# Online Decision-Making: Applications in Hiring, Inventory Placement, and Matching

Boris Epstein

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2025

## Abstract

Online Decision-Making: Applications in Hiring, Inventory Placement, and Matching

Boris Epstein

This dissertation studies three online decision-making problems motivated by practical applications in hiring pipelines, inventory placement, and online matching. In all cases, a decision-maker must take actions sequentially, often under time or resource constraints, and with partial or stochastic information about the future. A unifying objective is to develop computationally tractable policies that perform near-optimally relative to natural benchmarks, such as hindsight-optimal solutions or adaptive policies with full information. To this end, we design and analyze approximation algorithms with provable guarantees, drawing on linear programming relaxations, surrogate optimization, and randomized rounding.

The first chapter considers three hiring problems that differ in how offers are made to candidates: sequentially, in parallel, or all at once. We introduce a linear programming framework that captures each of these settings and develop simple, non-adaptive policies that attain approximation factors of at least $1 - 1/e$, improving upon prior guarantees. Our results unify and extend existing prophet inequality models, and provide theoretical and numerical insights into how firms should sequence offers based on operational constraints.

In the second chapter, we address the inventory placement problem in e-commerce, where inventory must be allocated to warehouses before facing stochastic demand. We formulate this as a two-stage optimization problem and show that optimizing a surrogate "offline" objective yields good performance in the downstream online matching phase. A key technical contribution is the

development of tight approximation guarantees via randomized rounding of LP relaxations, even in the multi-product setting. We complement our theoretical findings with extensive simulations, highlighting when optimistic or pessimistic placement heuristics are preferable in practice.

The third chapter focuses on online bipartite matching under limited historical data. We compare a range of algorithmic approaches—model-based, data-agnostic, and simulation-trained neural policies—through the lens of sample efficiency and robustness. Our results show that when data is scarce, structured model-based policies generalize better and offer competitive performance. In contrast, parameterized neural policies perform best in data-rich regimes, but at the cost of longer training and testing times.

Across all three chapters, we emphasize algorithmic strategies that scale efficiently and adapt to real-world constraints. Thematic connections—particularly the use of LP relaxations, surrogate models, and randomized rounding—help bridge seemingly distinct applications, and suggest a broader framework for designing principled, data-aware online decision policies.

# Table of Contents

# List of Figures

# List of Tables

## Acknowledgements

I would like to express my deepest gratitude to my advisor, Will Ma, for his exceptional mentorship, unwavering support, and the thoughtful guidance he has offered throughout my PhD journey. I am also thankful to my fellow PhD students for their camaraderie, stimulating discussions, and the many warm and memorable lunch breaks we shared.

I am grateful to the rest of the Columbia Business School faculty for their dedication to teaching and research, and for fostering such a vibrant academic environment. I also wish to thank the administrative staff for their tireless support and for helping everything run smoothly behind the scenes.

## Dedication

I dedicate this thesis to all the people who have supported me with their love and companionship. First and foremost, to my wonderful wife, who makes every day special. To my brother, my mother, my father, and my big family, whom I can always count on. And last but not least, to all my friends, both in New York and back in Santiago — thank you for being part of this journey.

# Chapter 1: Introduction

Many decision-making problems in operations and computer science are inherently online: decisions must be made as information arrives, with only partial knowledge of future events. This dissertation focuses on three such problems—selection and offering in hiring pipelines, inventory placement for downstream fulfillment, and online matching—each of which illustrates the tension between operational constraints and uncertainty in inputs or responses.

While the specific modeling assumptions and applications vary, the chapters are unified by a common set of algorithmic tools and theoretical techniques. In particular, all three chapters develop tractable algorithms with strong performance for problems that are otherwise computationally or structurally hard. Linear programming relaxations are used as benchmarks or algorithmic backbones, and randomized rounding is a central tool in both the design and analysis of the proposed methods.

### Chapter Overview

**Chapter 2: Selection and Ordering Policies for Hiring Pipelines.** This chapter considers three natural models for making hiring decisions: (i) sequential interviewing, (ii) parallel offering, and (iii) simultaneous offering with overbooking penalties. In all three settings, a firm seeks to hire up to $k$ candidates while being constrained by a finite number of interviews or offers. After each interview or offer, the firm receives a stochastic realization of either the candidate's quality or acceptance.

For each setting, we design non-adaptive algorithms that achieve near-optimal performance guarantees by solving suitable linear programs and applying randomized rounding techniques. In the sequential setting, we derive a $(1 - e^{-k}k^k/k!)$-approximation using a non-adaptive, commit-

ted policy—improving upon existing adaptivity gaps. In the parallel setting, we extend previous models to heterogeneous positions and develop a non-adaptive policy with provable guarantees. Finally, in the simultaneous offering setting, we analyze value-threshold policies and derive approximation bounds that are tight under mild assumptions.

This chapter emphasizes a unifying LP-based framework that allows the same core techniques to be applied across different variants of the hiring problem. In particular, randomized rounding plays a key role in designing policies that are both tractable and effective.

**Chapter 3: Optimizing Inventory Placement for Downstream Matching.** This chapter studies a two-stage problem in which inventory must be placed across warehouses before an online matching phase that fulfills stochastic demand over time. The placement must be done before the demand sequence is revealed and affects the feasibility and performance of downstream fulfillment.

The central idea is to optimize a surrogate objective—the expected reward under an optimistic offline matching policy—and use randomized rounding to derive feasible placement decisions. We analyze three surrogates (Offline, Myopic, and Fluid), and show that the Offline surrogate admits strong approximation guarantees when paired with a good fulfillment algorithm. We further extend this approach to multi-product settings, improving over the best-known bounds in the literature.

The use of randomized rounding is again critical: it bridges the gap between fractional LP solutions and implementable integer placements, with tight bounds derived under mild assumptions. Our numerical experiments validate the theory and provide practical guidance on which surrogates to prefer under various assumptions about fulfillment quality.

**Chapter 4: Data-Driven Online Matching Under Limited Data.** This chapter focuses on online bipartite matching in the presence of limited historical data. Unlike Chapters 2 and 3, this setting assumes that the distribution governing arrivals is unknown, and the algorithm must rely on a finite sample of past realizations. This creates a natural tension between model complexity and data efficiency.

We benchmark three algorithmic families: data-agnostic heuristics, model-based policies (e.g.,

LP-based thresholds or dynamic programs), and simulation-trained parametrized policies (e.g., neural networks). We find that model-based policies perform reliably even when the model is misspecified, particularly in low-data regimes. In contrast, parametrized policies are more flexible but require significant data and tuning to perform well.

This chapter contributes to a growing literature on data-driven decision-making by quantifying the statistical efficiency and robustness of competing methods. While randomized rounding is not used explicitly, the evaluation of model-based policies connects back to the techniques developed in earlier chapters.

**Unifying Themes**

Three common threads run through this dissertation. First, each chapter addresses decision-making under uncertainty and limited feedback. Second, approximation guarantees are obtained through linear programming relaxations that serve either as optimization tools or theoretical benchmarks. Third, randomized rounding emerges as a unifying technique that allows for tractable, implementable solutions with provable performance in both offline and online settings.

Together, these chapters contribute to a coherent framework for online decision-making that balances analytical rigor with practical implementability. They illustrate how careful modeling and algorithmic design can yield robust and scalable policies in a wide range of operational contexts.

# Chapter 2: Selection and Ordering Policies for Hiring Pipelines via Linear Programming

## 2.1 Introduction

Hiring the right personnel is one of the most important factors in the success of an enterprise. That being said, carrying out an efficient and timely recruitment process can be challenging in practice. Several difficulties arise when hiring, such as (but not limited to) deciding when to carry out the process, dealing with imperfect information, and making operational decisions at the time that the process is being carried out. This last aspect of the recruitment process raises several questions that can be studied through an algorithmic lens.

A typical recruitment process starts with a firm making a call for applications. An application usually consists of a resume and potential complementary materials. Based on this (imperfect) information, the firm decides *who* is going to be interviewed and *in which order* are the interviews going to be conducted. Both of these aspects are relevant because a recruitment process cannot go on forever. That is, there is not enough time to interview every single applicant, and the firm can choose who to interview next depending on the outcomes of past interviews.

Applicants become candidates once they are interviewed. After all interviews are conducted, the firm sends offers to the candidates it wishes to hire, given a limited set of positions. However, there are many possible ways to send offers to candidates. The first natural approach would be to **sequentially** send offers to candidates. By this we mean: send an offer, wait for the response of the candidate, and (if there is still a position available) carry on with the next offer. As in the interviewing process, the *order* in which the offers are sent becomes a relevant decision. A second approach, applicable only to a firm hiring more than one person, is to save time by sending offers in **parallel**. This means, for each position remaining to fill: send an offer, wait for the response

of the candidate, and (if still unfilled) carry on with subsequent offers. The final approach, which may be desirable under a tight timeline or to avoid revealing preferences among the candidates sent offers, is to send all offers **simultaneously**. However, it runs the risk of hiring more people than positions available, which comes at a cost.

To answer these operational questions and compare the different modes of sending offers, we study three different models of hiring processes that build off existing work.

**First model: sequential interviewing, a.k.a. ProbeTop-$k$.** In the first model, we assume that the firm has to hire up to $k$ people from a pool of $n$ applicants. Each applicant has a random value unknown to the firm carrying out the hiring process. The firm has access to distributional knowledge of these values coming from the applicants' resumes and complementary material submitted. The firm can interview applicants to find out the realization of their values, but there is a limit of $T$ on the number of interviews conducted. The realization of the value of the applicant becomes known to the firm immediately after carrying out the interview. We note that this realization should be interpreted as the applicant's *expected* value to the firm given the interview (and conditional on them accepting the offer). We treat this value as deterministic, which does not lose generality for a risk-neutral firm that maximizes its expectation. We further assume that realizations from interviews are independent across applicants. After all interviews are carried out, the firm can choose the best $k$ interviewed candidates to be hired, who are assumed to accept their offers with probability 1. The goal of the firm is to maximize the expected sum of values of the hired personnel. This problem is exactly the ProbeTop-$k$ problem, as described in Fu, Li, and Xu (2018).

In this problem, we can further distinguish between different classes of policies. First, we distinguish between *adaptive* and *non-adaptive* policies. Adaptive policies can decide the order of the interviews on the fly, choosing who to interview next depending on the outcomes of previous interviews. Non-adaptive policies, in contrast, have to fix an interview order before the process starts, which although restrictive, is attractive from an ease-of-implementation perspective. We also distinguish between *committed* and *non-committed* policies. Committed policies have to irrevocably decide whether to hire each candidate immediately after interviewing them and discovering their

value. Non-committed policies, in contrast, can carry out all interviews and choose the $k$ highest realized values in hindsight. Using committed policies could be attractive from a practical point of view, as waiting until the end incurs the risk that candidates accept offers from competing firms in the meantime. We are interested in bounds on how costly it is to restrict the firm to use policies that are non-adaptive and committed. The former is quantified in the literature by the widely studied notion of *adaptivity gap*: the worst-case ratio between the performance of general policies vs. algorithms that are restricted to be non-adaptive. Our results will bound the "adaptivity-commitment gap", in which the algorithm is restricted to be both non-adaptive and committed.

*Relation to Free-Order Prophets.* The Free-Order Prophet Inequality problem is the special case of ProbeTop-$k$ where $T = n$ and only committed policies are allowed. (When $T = n$, the constraint of $T$ interviews is not binding, and hence non-committed policies can just trivially interview all applicants.) Typically in prophet inequalities, the benchmark can see all applicants' values in advance and simply choose to interview and hire the $k$ overall highest-valued candidates. We note that such a benchmark is *too powerful to compare against* in our more general problem if $n$ is much larger than $T$ since the benchmark sees all $n$ realizations while the algorithm can only interview $T$ applicants. This is why in our general ProbeTop-$k$ problem, we compare to an optimal (adaptive, non-committed) algorithm that is still bound by $T$ interviews that must be decided without any prophetic information, making our comparison different from prophet inequalities.

*Relation to Sequential Offering.* In the Sequential Offering model of Purohit, Gollapudi, and Raghavan (2019), applicants are assumed to have already been interviewed but have uncertainty about whether they accept an offer. The firm knows, for each candidate, how likely it is that they will accept an offer, and the value they add to the firm, should they accept. The firm has time to send at most $T$ offers and wants to maximize the expected total value of up to $k$ candidates who accept their offers. This setting is closely related to the special case of ProbeTop-$k$ with weighted Bernoulli distributions, where the values take a positive realization with some probability (representing an acceptance) or 0 with the remaining probability (representing a rejection). The subtle difference is that an accepted offer cannot be withdrawn by the firm in the Sequential Offering

model, whereas in the ProbeTop-$k$ model, the firm can turn down a candidate even if their value turns out to be positive. We will show that our algorithm satisfies properties that makes it admissible for this Sequential Offering problem too, and improve upon the results of Purohit, Gollapudi, and Raghavan (2019).

**Second model: Parallel Offering.** We also study a Parallel Offering model, in which a firm again has to hire people to fill $k$ positions. However, we now allow for heterogeneous positions, where a candidate may have different potential values for different job positions. We assume that all interviews have already been conducted, leaving us with a pool of $n$ desirable candidates. After conducting all interviews the firm learned, for each candidate, how valuable they are for each of the available positions and how likely it is for each candidate to accept an offer for each of the available positions. The firm must now decide how to send offers in $T$ parallel offering rounds. At each round, the firm can send an offer for each of the positions that have not yet been filled by a candidate. When a candidate receives an offer, they can either accept or reject it, with the assumption that they cannot receive an offer for another position if they rejects it (and hence candidates do not try to anticipate offers they might receive later). The goal of the firm is to maximize the expected sum of values of hired candidates. We develop a non-adaptive algorithm that can be computed efficiently and performs competitively in comparison to adaptive and even relaxed sequential algorithms.

This model generalizes a parallel offering model also introduced in Purohit, Gollapudi, and Raghavan (2019), which is similar but has $k$ identical instead of heterogeneous positions. Our Parallel Offering model is not only more general, but we also derive stronger performance guarantees.

**Third model: Simultaneous Offering.** Finally, we study the Simultaneous Offering model, again for a firm hiring to fill $k$ positions. As in the Parallel Offering model, the firm has already conducted all interviews, resulting in a pool of $n$ candidates for which it knows how valuable each candidate is to the firm and how likely each candidate is to accept an offer. The firm must decide on a subset of candidates who will receive an offer (all at the same time). This subset can be of any size, so a possible outcome is that more than $k$ candidates accept an offer. If that is the case, the

firm pays a penalty for each candidate who accepted an offer beyond the capacity $k$. This penalty can be thought of as the cost of withdrawing an offer, or the cost of creating a new position in the firm. We analyze the performance of value-ordered policies, which send offers to candidates above a value threshold (regardless of their probability of acceptance). We derive near-optimal approximation guarantees for these policies.

Table 2.1 contains a comparison of all the models studied/captured in this paper.

| Model | Action performed | Result of action | Actions per time step | Moment of hiring | Bound on total hires |
|---|---|---|---|---|---|
| ProbeTop-$k$ | Interview | Observe value of candidate | One | After last interview | Hard |
| Free-Order Prophets | Interview | Observe value of candidate | One | After each interview (irrevocable) | Hard |
| Sequential Offering | Send offer | Observe accept/reject decision | One | Upon acceptance of offer | Hard |
| Parallel Offering | Send offer(s) | Observe accept/reject decision(s) | One per position remaining | Upon acceptance of offer | Hard |
| Simultaneous Offering | Send offer(s) | Observe accept/reject decision(s) | $n$ | Upon acceptance of offer | Soft (linear penalty) |

Table 2.1: Comparison between models.

### 2.1.1 Outline of Results

We will generally say that our algorithm is $\alpha$-approximate if its expected total value collected is at least $\alpha$ times that of an optimal algorithm, from a larger class. We call $\alpha \in [0, 1]$ the approximation factor. We note that typically this terminology is used when comparing to the optimal algorithm from the *same* class. Our results imply lower bounds on the approximation factor $\alpha$ under the typical terminology, since we are comparing against a larger class. All of our results assume that distributions have *finite support* and are explicitly input in the form of (value, probability) pairs with binary encoding. All of our algorithms are polynomial-time under this form of input.

We should note that for the problem of computing the optimal algorithm within a fixed class, no computational hardness results are known for any of the problems we study (ProbeTop-$k$, Parallel Offering, Simultaneous Offering), to our knowledge. Nonetheless, our algorithms still have some of the currently best-known approximation factors, which also hold when comparing to a larger class.

**Sequential interviewing a.k.a. ProbeTop-$k$ problem.** We develop a polynomial-time algorithm that is non-adaptive, committed, and achieves a $(1 - e^{-k}k^k/k!)$ approximation factor relative to an optimal adaptive, non-committed algorithm (Theorem 1, Subsection 2.4.4). We note that the approximation factor of $(1 - e^{-k}k^k/k!)$ equals $1 - O(1/\sqrt{k})$ by Stirling's approximation, and is always at least $1 - 1/e \approx 0.632$ (when $k = 1$). We also note that it is tight relative to the LP benchmark we compare against (Appendix A.2.6).

Our results for this problem, while simple and clean, have broad implications. First, we can combine our algorithm with the work of Fu, Li, and Xu (2018) to obtain a PTAS for ProbeTop-$k$ (Corollary 1, Subsection 2.4.5). There exist PTAS's for ProbeTop-$k$ restricted to non-adaptive algorithms (Segev and Singla 2021) and ProbeTop-$k$ restricted to committed algorithms (Fu, Li, and Xu 2018), but a PTAS for the general (adaptive, non-committed) ProbeTop-$k$ problem appears unknown, assuming that $k$ is part of the input. Second, the best-known existing guarantee on the adaptivity gap for ProbeTop-$k$ was 1/2, due to Bradac, Singla, and Zuzic (2019) via an algorithm that is not necessarily polynomial-time. We improve the lower bound on the adaptivity gap for ProbeTop-$k$ from $1/2$ to $1 - 1/e$ using a non-adaptive, polynomial-time algorithm, and moreover show asymptotic optimality when the number of positions grows to infinity. Third, our $(1 - e^{-k}k^k/k!)$-approximation algorithm can be directly applied to the Sequential Offering problem without loss (Corollary 2, Subsection 2.4.6), improving the previous-best approximation factor and adaptivity gap of $1/2$ (Purohit, Gollapudi, and Raghavan 2019). As with Bradac, Singla, and Zuzic (2019), the adaptivity gap shown in Purohit, Gollapudi, and Raghavan (2019) was achieved through a non-constructive approach. Finally, our abstract treatment leads to an algorithm that works in the Free-Order Prophets setting (because it is committed), hence our results extend to a

free-order "prophet inequality" (comparing against a necessarily weaker benchmark) that allows an additional constraint of $T$ on the number of interviews.

**Parallel Offering problem.** For the Parallel Offering model, we develop an algorithm that is non-adaptive and achieves a $(1 - 1/e)$ approximation factor relative to an optimal (adaptive) algorithm (Theorem 2, Subsection 2.5.3). This result also both improves and generalizes existing results, in this case, the 1/8-approximate algorithm of Purohit, Gollapudi, and Raghavan (2019) that works in the special case where all positions are identical. It is also tight relative to the LP benchmark we compare against (Appendix A.3.5). We further show that our algorithm obtains at least $(1 - 1/e)$ times what an optimal algorithm would obtain in the Sequential Offering problem with $kT$ time steps, establishing a lower bound on the value of batching offers (Corollary 3, Subsection 2.5.4).

**Simultaneous Offering problem.** We analyze the performance of *value-ordered* policies, which send offers to all candidates whose value is above a threshold. The idea of having a value threshold is practically motivated and the optimal value-ordered policy can be computed efficiently. We show that with no assumptions on the values of the candidates, value-ordered policies can have arbitrarily poor performance (Example 1, Subsection 2.6.1). Consequently, we assume that the valuations of candidates are lower bounded by a parameter $\tau \in (0, 1)$, and provide a value-ordered policy that achieves at least a factor $\alpha_k^\tau$ of what an optimal policy could achieve, where $\alpha_k^\tau$ is an increasing function of $k$ and $\tau$ (Theorem 3, Subsection 2.6.4) that satisfies $\lim_{k\to\infty} \alpha_k^\tau = 1$ for every $\tau \in (0, 1)$ (Corollary 4, Subsection 2.6.4). We also provide an instance where no value-ordered policy can achieve a factor higher than $\beta_k^\tau$ of what an optimal policy could achieve, where $\beta_k^\tau$ is another increasing function of $k$ and $\tau$ (Theorem 4, Subsection 2.6.4). We further characterize the region where $\alpha_k^\tau = \beta_k^\tau$ (Proposition 6, Subsection 2.6.4). In particular, our bound is tight for all $k$ if $\tau \leq 1/2$.

**Numerical study.** In Appendix A.5, we numerically simulate the three different models of sending offers (sequential, parallel, simultaneous) using the same candidate pool generation process as Purohit, Gollapudi, and Raghavan (2019). We illustrate the improved performances offered

by our new policies, and compare the performances attainable across the different models, providing insight into when the firm should favor sending offers sequentially, in parallel, or simultaneously. For instance, there is surprisingly little value to gain from an algorithm that sends offers in parallel, as opposed to sending single offers sequentially and being highly adaptive to the number of remaining positions, unless the horizon for making offers is extremely short. For Simultaneous Offering, we demonstrate that value-ordered policies are generally desirable unless there is both a small number of initial positions and a high cost of overage. In that case, it is better to identify the highest-valued "safe" candidates (with a high probability of acceptance) to reduce the variance in the number of accepted offers. We acknowledge that these insights are based on the specific generative model of Purohit, Gollapudi, and Raghavan (2019) and they do not necessarily hold in general.

### 2.2 Related Work

**Stochastic Probing and Matching.** Our work is closely related to the general stochastic probing problem studied in Gupta and Nagarajan (2013) and Gupta, Nagarajan, and Singla (2016). These papers study the problem of sequentially probing elements in order to maximize the sum of the weights of a selected subset. In their setting they consider more general sets of 'outer' constraints to be satisfied by the probed elements, and 'inner' constraints to be satisfied by the selected elements. In this language, ProbeTop-$k$ considers the outer constraint to be the $T$-uniform matroid and the inner constraint to be the $k$-uniform matroid. Their work is different from ours in that it only allows the probe to have two outcomes (active or inactive) and active probes must be irrevocably included in the final subset. Bradac, Singla, and Zuzic (2019) introduce the multiple-type general stochastic probing problem. In this work, they only work with outer constraints and include the inner constraints by allowing submodular functions instead of modular functions.

In Bradac, Singla, and Zuzic (2019) they show that the adaptivity gap is exactly 1/2 for the stochastic probing problem with monotone submodular functions under prefix-closed probing constraints. Their proof is not constructive, in the sense that the algorithm for their lower bound re-

quires the optimal decision tree as an input. The best-known non-adaptive algorithm for which is known that it can be computed in polynomial time is due by Gupta, Nagarajan, and Singla (2017), which achieves a 1/3 approximation for submodular and XOS functions under prefix-closed probing constraints.

**PTAS-type results.** Fu, Li, and Xu (2018) develop PTAS's for a class of dynamic programs, that includes ProbeTop-1 over general (adaptive, non-committed) algorithms, and ProbeTop-$k$ over committed algorithms. They also provide a $(1-\varepsilon)-$approximation for ProbeTop-$k$ over adaptive, non-committed algorithms whose runtime is polynomial in $1/\varepsilon$ but exponential in $k$. Segev and Singla (2021) develop the improved notion of EPTAS's for several related problems, including ProbeTop-$k$ over non-adaptive algorithms. Their EPTAS works differently when $k$ is small or large. We use the same LP relaxation as theirs for the large $k$ case, although our rounding scheme and analysis are very different. We note that none of these PTAS-type results use non-adaptive, committed policies to approximate the best adaptive, non-committed policies, as in our paper.

**Prophet Inequalities.** Our work has the flavor of Prophet Inequalities in that we are deciding online whether to accept values drawn from known distributions, and comparing against a supernatural benchmark. Classical works in Prophet Inequalities (Krengel and Sucheston 1977; Krengel and Sucheston 1978; Samuel-Cahn 1984; Kleinberg and Weinberg 2012; Esfandiari et al. 2017) assume that the order is adversarial or random, while we are studying the *free-order* variant where the order can be decided (Hill 1983; Beyhaghi et al. 2021; Agrawal, Sethuraman, and Zhang 2020). However, due to the constraint of $T$ time steps, as discussed earlier, it is important to emphasize that the benchmark we are comparing against is also bound by $T$ time steps, so we are not comparing against (and it is impossible to compare against) the true prophet. For sequential interviewing, the approximation factor we derive of $(1-e^{-k}k^k/k!)$ has also been established for free-order by Yan (2011) and extended in Arnosti and Ma (2021) to random-order, but neither of these works allows for a constraint of $T$ time steps.

**Simultaneous Offering model.** Our Simultaneous Offering model and in particular the linear penalty cost is motivated by static overbooking models in revenue management (see e.g. Gallego

and Topaloglu 2019, Ch. 3). However, our decision is different in that we are selecting a *subset* of candidates, whereas they are setting a single booking limit (possibly one for each fare class). The simple family of heuristics for which we provide an approximation guarantee also has no analog in overbooking. Our problem also shares many aspects with the one studied in Cominetti et al. (2010), where a subset of customers are sent last-minute offers for a limited amount of items. Their model differs from ours in that if more items than available inventory are sold, they only collect value from a random subset of size $k$; whereas, our model collects the value of all acceptances but has to pay a linear penalty.

**Concurrent work.** Parallel to our work, Gallego and Segev (2022) have studied and obtained results for the ProbeTop-$k$ problem. Although both works aim to derive non-adaptive algorithms for the problem, theirs is different from ours in several aspects. First, their models assume that the valuations of the candidates come from independent random variables with either general distributions or continuous distributions, whereas our work focuses on random variables with finite support. While the bound they obtain for continuous random variables matches the $1 - e^{-k}k^k/k!$ bound we obtain for finitely-supported random variables, the bound they obtain for general random variables is $1/2$. Another aspect that distinguishes our work from Gallego and Segev 2022 is the relaxations used to benchmark the performance of algorithms. We use standard LP relaxations, whereas they introduce a novel benchmark consisting of a simple minimax problem.

## 2.3 Problem Formulations and Preliminaries

In this section, we formally state the problems studied in the paper. We first state the ProbeTop-$k$ problem in Subsection 2.3.1. We then state the Parallel Offering problem in Subsection 2.3.2. We close by stating the Simultaneous Offering problem in Subsection 2.3.3.

### 2.3.1 ProbeTop-k Problem

In the ProbeTop-$k$ (`ptk`) problem, a firm faces the challenge of filling $k$ positions out of a pool of $n$ applicants. Each applicant $i$ has a random, non-negative value $V_i \sim F_i$, and $F_i$ is known

to the firm. Before the firm hires an applicant, an interview must be conducted. When the firm interviews applicant $i$, the realization of $V_i$ becomes known to the firm. The firm can conduct at most $T$ interviews in total and then can choose any $k$ interviewed candidates to be hired. The goal of the firm is to maximize the sum of the values of the hired candidates. An instance of the problem is characterized by a tuple $I = (k, T, n, F)$, where $1 \leq k \leq T \leq n$, and $F = \{F_i\}_{i\in[n]}$ is a collection of probability distribution functions. In this work, we focus on distributions supported on a finite set of non-negative values $\{r_j\}_{j\in[J]}$ and we use $q_{ij}$ to denote $\mathbb{P}(V_i = r_j)$. We use $\mathcal{I}_{\texttt{ptk}}$ to denote the set of all possible instances for the ProbeTop-$k$ problem. We further use $\mathcal{I}^k_{\texttt{ptk}} \subseteq \mathcal{I}_{\texttt{ptk}}$ to denote the subset of instances where the number of positions is k.

For this problem, we define a policy as a function $\pi$ that maps the remaining budget of interviews, the set of applicants that have not yet been interviewed, and the realization of the values of the candidates that have already been interviewed to a decision of which applicant to interview next. Let $\Pi^{\texttt{ptk}}$ be the set of all policies for $\texttt{ptk}$. For a policy $\pi \in \Pi^{\texttt{ptk}}$ and an instance $I \in \mathcal{I}_{\texttt{ptk}}$, let $R_\pi(I)$ be the expected reward of using policy $\pi$ on the instance in question. For an instance $I \in \mathcal{I}_{\texttt{ptk}}$ define $\mathsf{OPT}_{\texttt{ptk}}(I) := \sup_{\pi\in\Pi^{\texttt{ptk}}} R_\pi(I)$ as the expected reward of using the best possible policy on instance $I$. We call a policy $\pi \in \Pi^{\texttt{ptk}}$ an $\alpha$-approximation if

$$\inf_{I\in\mathcal{I}_{\texttt{ptk}}} \frac{R_\pi(I)}{\mathsf{OPT}_{\texttt{ptk}}(I)} \geq \alpha.$$

We distinguish between *adaptive* and *non-adaptive policies*. A non-adaptive policy has to decide an order in which to conduct the interviews before the process starts. An adaptive policy, in contrast, conducts the interviews sequentially and can use the outcomes of previous interviews to decide which candidate to interview next. We also distinguish between *committed* and *non-committed* policies. After each interview, committed policies must irrevocably decide whether to hire the candidate or not. Non-committed policies, in contrast, can interview $T$ applicants and then choose the $k$ highest realizations among them. Among other questions, we are interested in how well can the firm perform when restricted to using non-adaptive policies, committed policies, or

both. To quantify this, we are interested in how good an approximation factor can be achieved by restricting the firm to using policies that are both non-adaptive and committed.

### 2.3.2 Parallel Offering Problem

The second setting we study in this paper is the Parallel Offering problem (par). Again consider the case of a firm that has to hire candidates in order to fill $k$ positions. Instead of conducting interviews, the firm has to send offers to candidates. The positions of the firm are now not identical, and for each candidate $i \in [n]$ and position $j \in [k]$ the firm knows $v_{ij}$: the reward collected by the firm if candidate $i$ accepts an offer for position $j$, and $p_{ij}$: the probability that the candidate $i$ accepts an offer for position $j$. The firm can carry out at most $T$ offering rounds. At each round, the firm can send an offer for each unfilled position in parallel. Each candidate can receive at most one offer in total. The goal of the firm is to maximize the expectation of the sum of the values of the accepted offers. An instance for this problem is defined by a tuple $I = (k, T, n, p, v)$, where $1 \leq k \leq T \leq n$, $p \in [0,1]^{n \times k}$ and $v \in \mathbb{R}_+^{n \times k}$. For this problem, we define a policy as a function $\pi$ that maps the remaining number of time steps, the remaining set of unfilled positions, and the set of candidates that have not yet received an offer, to an assignment of a subset of not-yet-offered candidates to unfilled positions. Let $\mathcal{I}_{\texttt{par}}$ be the set of all instances and $\Pi^{\texttt{par}}$ be the set of all policies for the Parallel Offering model. Again let $R_\pi(I)$ denote the expected reward of using policy $\pi \in \Pi^{\texttt{par}}$ on instance $I \in \mathcal{I}_{\texttt{par}}$. Let $\mathsf{OPT}_{\texttt{par}}(I) := \sup_{\pi \in \Pi^{\texttt{par}}} R_\pi(I)$ be the best possible expected reward that can be obtained from an instance $I \in \mathcal{I}_{\texttt{par}}$. We say that a policy $\pi \in \Pi^{\texttt{par}}$ is an $\alpha$-approximation if

$$\inf_{I \in \mathcal{I}_{\texttt{par}}} \frac{R_\pi(I)}{\mathsf{OPT}_{\texttt{par}}(I)} \geq \alpha.$$

For the purpose of analysis, we can imagine that before the offering rounds start, each candidate has already decided which offers they would accept. We allow for these decisions to be correlated across different positions for a single candidate, but we assume independence across candidates.

In this context, we define a non-adaptive policy as follows. For each position, a list of candidates is constructed in a way such that a candidate cannot appear in more than one list, and that

each list contains no more than $T$ candidates. Each list must also have a specified fixed ordering of its candidates. For each position, offers are sequentially sent to the candidates in their corresponding list, in its corresponding order, until either one of the candidates accepts or the list ends without any acceptance (independent of what happens with the other positions). The algorithm we develop for this problem falls under this definition, and thus we establish a lower bound on how well such non-adaptive policies can perform.

### 2.3.3 Simultaneous Offering Problem

The third and last setting we study is the Simultaneous Offering problem (`sim`). As in the previous models we study, a firm faces the challenge of filling $k$ positions out of a pool of $n$ candidates. There are two aspects that make this problem different from `ptk` and `par`. First, all offers must be made at the same time, and only once (hence the name Simultaneous Offering). The second aspect making this problem different is that the firm can end up hiring over capacity, but pays a linear cost for each candidate hired beyond $k$.[1]

Formally, we have the following problem. There are $n$ candidates. For each candidate $i \in [n]$, we know $p_i$: the probability that candidate $i$ accepts an offer if sent, and $v_i$: the value obtained if $i$ accepts the offer. The firm must select a subset $S \subseteq [n]$ of candidates to send offers to, which has no cardinality constraint. When the subset $S$ is decided, an offer is sent to each of the candidates in $S$. Each candidate accepts the offer independently with probability $p_i$, in which case they add value $v_i$ to the firm. For each candidate that exceeds the capacity $k$, the firm incurs a linear cost of $c > 0$. Let $A \subseteq S$ be the (random) subset of candidates that accepted their offers. The reward obtained by the firm is $\sum_{i \in A} v_i - c \cdot [|A| - k]^+$. By re-scaling values $v_i$ by $1/c$, we can assume without loss of generality that $c = 1$. The re-scaled values $v_i$ can be interpreted as the value that a candidate adds to the firm, relative to the cost of hiring a candidate over capacity.

With this notation, an instance of `sim` can be described by a tuple $I = (k, n, p, v)$, where $1 \leq k \leq n$, $p \in [0, 1]^n$ and $v \in \mathbb{R}^n_+$. We define a policy as a function $\pi$ that maps the problem

[1] Requiring the firm to hire at most $k$ candidates with probability 1 would make this problem trivial: the firm would send an offer to the $k$ candidates with the highest expected value $v_i p_i$.

instance $I$ to a (possibly random) subset of candidates $S \subseteq [n]$. Let $\mathcal{I}_{\texttt{sim}}$ denote the set of all instances and $\Pi^{\texttt{sim}}$ denote the set of all policies for the Simultaneous Offering problem. Let $R_\pi(I)$ denote the expected reward of using policy $\pi \in \Pi^{\texttt{sim}}$ on instance $I \in \mathcal{I}_{\texttt{sim}}$. Let $\mathsf{OPT}_{\texttt{sim}}(I)$ be the best possible expected reward that can be obtained from an instance $I \in \mathcal{I}_{\texttt{sim}}$. We call a policy $\pi$ an $\alpha$-approximation if

$$\inf_{I \in \mathcal{I}_{\texttt{sim}}} \frac{R_\pi(I)}{\mathsf{OPT}_{\texttt{sim}}(I)} \geq \alpha.$$

One thing worth noting is that, although the objective function of the problem is submodular, it is not monotone and can take negative values. Thus, we cannot use general results for approximating maximization problems with submodular objective functions.

### 2.4 ProbeTop-k Problem

In this section, we provide our algorithm and analysis for the ProbeTop-$k$ problem, where applicants are sequentially interviewed. We develop a non-adaptive, committed policy that achieves a $(1 - e^{-k}k^k/k!)$ approximation of any (adaptive, non-committed) policy. The algorithm first solves an LP relaxation that upper-bounds the performance of any (adaptive, non-committed) policy, and uses then the LP solution as an input to decide the applicants that will be interviewed, in which order, and whether to hire an interviewed applicant, given their value.

In Subsection 2.4.1 we state the LP in question. In Subsection 2.4.2 we introduce a simple dependent rounding scheme that will be used in our approximation algorithm. In Subsections 2.4.3 and 2.4.4 we introduce and analyze our approximation algorithm. In Subsection 2.4.5 we explain how to use our approximation algorithm as a subroutine for obtaining a PTAS. In Subsection 2.4.6 we treat the special case where values are distributed as weighted Bernoulli random variables and extend our results to the Sequential Offering problem. The analyses in this section, as well as those in Section 2.5 make use of correlation gap results by Yan (2011) and dependent rounding schemes by Gandhi et al. (2006). We recommend readers who are not familiar with these papers to visit Appendix A.1, which contains a summary of the relevant technical results contained in

the previously mentioned papers, before reading the proofs. Throughout this section (except for Subsection 2.4.6) we use the word applicant indistinctly from candidate to avoid any confusion, although we had previously distinguished that an applicant becomes a candidate after being interviewed.

### 2.4.1 Linear Program

Consider the linear program $\mathrm{LP}_{\mathtt{ptk}}$, which for any instance of ProbeTop-$k$, upper-bounds the performance of all possible policies. Linear programs in this spirit have been used exhaustively in the literature. Indeed, this relaxation is used to approach the same problem in Segev and Singla (2021), and it consists of a special case of the linear program used in Gupta and Nagarajan (2013). In $\mathrm{LP}_{\mathtt{ptk}}$, variable $y_i$ is to be interpreted as the probability that applicant $i$ is interviewed. The variable $x_{ij}$ is to be interpreted as the probability that $i$ is hired when $V_i = r_j$.

$$
\begin{aligned}
\mathrm{LP}_{\mathtt{ptk}}(I) = \max \quad & \sum_{i=1}^{n}\sum_{j=1}^{J} r_j x_{ij} & & \\
s.t. \quad & x_{ij} \le y_i q_{ij} & \forall i \in [n], j \in [J] & \quad (2.1)\\
& \sum_{i=1}^{n} y_i \le T & & \quad (2.2)\\
& \sum_{i=1}^{n}\sum_{j=1}^{J} x_{ij} \le k & & \quad (2.3)\\
& x_{ij} \ge 0 & \forall i \in [n], j \in [J] & \\
& 0 \le y_i \le 1 & \forall i \in [n]. &
\end{aligned}
$$

We first show that $\mathrm{LP}_{\mathtt{ptk}}$ upper-bounds the optimal (adaptive, non-committed) algorithm.

**Lemma 1** *For any instance $I \in \mathcal{I}_{\mathtt{ptk}}$, we have $\mathrm{LP}_{\mathtt{ptk}}(I) \ge \mathsf{OPT}_{\mathtt{ptk}}(I)$.*

Although this result is not new, we provide a proof of the claim in Appendix A.2.1 for completeness.

The following lemma establishes a convenient fact about the basic feasible solutions of the LP. This will let us develop a simple rounding scheme for selecting applicants to be interviewed.

**Lemma 2** *Let $I \in \mathcal{I}_{\texttt{ptk}}$ and let $y = (y_i)_{i\in[n]}, x = (x_{ij})_{i\in[n],j\in[J]}$ be a basic feasible solution of $\mathrm{LP}_{\texttt{ptk}}(I)$. Then $y$ has at most 2 non-integer components. If it has 2 non-integer components, they sum to 1.*

The proof of this lemma is deferred to Appendix A.2.2.

### 2.4.2 Dependent Rounding

We develop a simple dependent rounding scheme (which we call DR for short) that will be used as a subroutine of our approximation algorithm. DR receives an optimal solution $y = (y)_{i\in[n]}$ of $\mathrm{LP}_{\texttt{ptk}}$ as an input and returns a (possibly random) vector $Y \in \{0,1\}^n$. DR works differently depending on the number of fractional components of the optimal solution $y$. In any case, for all $i$ such that $y_i \in \{0,1\}$, it sets $Y_i = y_i$ with probability 1. If $y$ is integral, then $Y$ is deterministic. If there is exactly one fractional component $i'$, then it sets $Y_{i'} = 1$ with probability $y_{i'}$ (and it sets $Y_{i'} = 0$ otherwise). If there are exactly two fractional components $i_1$ and $i_2$, then it sets $Y_{i_1} = 1, Y_{i_2} = 0$ with probability $y_{i_1}$ and sets $Y_{i_1} = 0, Y_{i_2} = 1$ with probability $1 - y_{i_1} = y_{i_2}$. It is easy to see that the output of DR satisfies the following two properties: (P1) $\mathbb{E}(Y_i) = y_i$ for every $i \in [n]$, and (P2) $\sum_{i\in[n]} Y_i \le T$ with probability 1.

### 2.4.3 Approximation Algorithm

We propose the following algorithm for the ProbeTop-$k$ problem, which we call $\mathsf{ALG}_{\texttt{ptk}}$. Given an instance, we first solve $\mathrm{LP}_{\texttt{ptk}}(I)$. Let $x, y$ be an optimal solution. Define

$$p_i = \frac{\sum_{j=1}^{J} x_{ij}}{y_i}, \quad v_i = \frac{\sum_{j=1}^{J} r_j x_{ij}}{\sum_{j=1}^{J} x_{ij}}. \tag{2.4}$$

If any of the denominators are 0, then define these values as 0.[2] For reasons that will soon become clear, $p_i$ is to be interpreted as the probability that we accept applicant $i$ given that they get interviewed, and $v_i$ is to be interpreted as the expected value of $i$ given that they get hired. Assume that after solving the LP, we relabel the applicants so that $v_1 \geq v_2 \geq \cdots \geq v_n$.

The algorithm first chooses which applicant to interview and the order in which the interviews are going to take place. For the first task, we use the dependent rounding DR just introduced. Specifically, we feed vector $y$ as an input to DR, and obtain binary random variables $\{Y_i\}_{i\in[n]}$. The algorithm will choose to interview applicants $i$ with $Y_i = 1$. For the second task, the algorithm will always interview applicants in decreasing order of $v_i$. Upon interviewing applicant $i$ and observing their value $r_j$, the algorithm hires $i$ with probability $x_{ij}/(y_i q_{ij})$.

### 2.4.4 Analysis of $\mathsf{ALG}_{\texttt{ptk}}$

To analyze the algorithm let $Y_i$ be the indicator that applicant $i$ is included in the list of applicants to receive an interview. Let $Q_{ij}$ be the indicator of $V_i = r_j$. Let $P_i$ be the indicator that applicant $i$ would be hired if they were interviewed while still having unfilled positions. We colloquially refer to this event as $i$ 'making the cut'. We have

$$\mathbb{E}(P_i) = \sum_{j=1}^{J} \mathbb{E}(P_i|Q_{ij})q_{ij} = \sum_{j=1}^{J} \frac{x_{ij}}{q_{ij}y_i} q_{ij} = p_i.$$

Define $Z_i = Y_i \cdot P_i$ as the indicator that $i$ is included in the list of applicants to receive an interview and that $i$ makes the cut. Note that $Y_i$ is independent of $P_i$. Let $N_{i-1} = \sum_{\ell=1}^{i-1} Z_\ell$ be the number of applicants that are interviewed before applicant $i$ that would be hired if there are still positions available. We can therefore write the (random) reward of our algorithm as $\sum_i V_i \mathbb{1}\{N_{i-1} < k\}Z_i$. The following lemma allows us to write the expected reward of $\mathsf{ALG}_{\texttt{ptk}}$ in a way that will allow us to relate it to weighted rank functions of $k$-uniform matroids.

**Lemma 3** *For any $I \in \mathcal{I}_{\texttt{ptk}}$, we have $R_{\mathsf{ALG}_{\texttt{ptk}}}(I) = \sum_{i=1}^{n} v_i \mathbb{E}(\mathbb{1}\{N_{i-1} < k\}Z_i)$.*

[2] If $y_i = 0$, then $\sum_{j=1}^{J} x_{ij} = 0$ too. If $y_i = 0$ then the algorithm will never interview applicant $i$, so the definition of these values is not relevant for the analysis.

The proof of this lemma is deferred to Appendix A.2.3. Since the applicants are labeled such that $v_1 \geq v_2 \geq \cdots \geq v_n$, Lemma 3 implies that our algorithm has the same expected reward as an algorithm that would first sample $Z_i = Y_i P_i$ for all applicants and then collect the $k$ highest values $v_i$ among those applicants with $Z_i = 1$. This interpretation is only possible because $\mathsf{ALG}_{\texttt{ptk}}$ is committed. The same expressions can be derived for an algorithm that instead of sampling $Y_i$ using DR, samples $\tilde{Y}_i$ with the same marginal probabilities as $Y_i$, but independently. These expressions are useful for showing that our algorithm outperforms a *hypothetical algorithm* that decides to interview each applicant $i$ using the independent indicators $\tilde{Y}_i$ instead of the correlated indicators $Y_i$, potentially interviewing $T + 1$ applicants. The correlation induced by the constraint of $T$ interviews on the actual algorithm only works in our favor.

**Lemma 4** *Let $y = (y_i)_{i\in[n]}, x = (x_{ij})_{i\in[n],j\in[J]}$ be a basic feasible solution of $\mathrm{LP}_{\texttt{ptk}}(I)$ with two fractional components $i_1$ and $i_2$. Let $\tilde{Y}_i$ be independent Bernoulli random variables with mean $y_i$. Define $\tilde{Z}_i = P_i\tilde{Y}_i$ and $\tilde{N}_i = \min\{k, \sum_{\ell=1}^{i} \tilde{Z}_\ell\}$. Then*

$$\sum_{i=1}^{n} v_i \mathbb{E}\left(Z_i \mathbb{1}\{N_{i-1} < k\}\right) \geq \sum_{i=1}^{n} v_i \mathbb{E}\left(\tilde{Z}_i \mathbb{1}\{\tilde{N}_{i-1} < k\}\right). \tag{2.5}$$

The proof of this lemma is deferred to Appendix A.2.4.

To conclude our analysis we relate the performance of the hypothetical algorithm (with independent indicators) and the objective function of the LP, through the weighted rank function of $k$-uniform matroids. For a set $S \subseteq [n]$ we define the weighted rank function for the $k$-uniform matroid with weights $v$ as $v^*(S) = \max_{R\subseteq[S],\,|R|\leq k} \sum_{i\in R} v_i$.

Both the reward of the hypothetical independent algorithm and the objective of $\mathrm{LP}_{\texttt{ptk}}$ can be expressed as expectations of weighted rank functions of $k$-uniform matroids. We use this together with the correlation gap results by Yan (2011) to show the main result of this section.

**Theorem 1** *For any instance $I \in \mathcal{I}^k_{\texttt{ptk}}$, we have $R_{\mathsf{ALG}_{\texttt{ptk}}}(I) \geq \left(1 - \frac{e^{-k}k^k}{k!}\right) \mathrm{LP}_{\texttt{ptk}}(I)$.*

The proof of this theorem is deferred to Appendix A.2.5. In Appendix A.2.6 we show that the guarantee that our algorithm attains is tight.

It is worth noting that this algorithm can be de-randomized. Indeed, the algorithm will randomize between at most two fixed orders, so both of them can be evaluated and the best among them can be selected. Thus, we have a deterministic, non-adaptive, committed algorithm whose expected reward will be at least a $(1 - e^{-k}k^k/k!)$ factor of the expected reward of the optimal algorithm. Since our algorithm is non-adaptive and committed, it establishes a lower bound on how good an approximation factor can be achieved by restricting to these classes of policies.

### 2.4.5 PTAS for ProbeTop-$k$

We now explain how to combine $\mathsf{ALG}_{\texttt{ptk}}$ with the (adaptive, non-committed) approximation algorithm proposed by Fu, Li, and Xu (2018) to obtain a PTAS for ProbeTop-$k$. We know there is a $(1 - \varepsilon)$-approximation algorithm for ProbeTop-$k$ whose runtime is exponential in $k$ (Fu, Li, and Xu 2018, Theorem 4.2). The idea behind the combined PTAS is that for $k$ large enough, $\mathsf{ALG}_{\texttt{ptk}}$ obtains an approximation factor better than $1 - \varepsilon$, so for small values of $k$ we would run the algorithm by Fu, Li, and Xu (2018), and for large values of $k$ we would run our $\mathsf{ALG}_{\texttt{ptk}}$. This essentially allows us to treat $k$ as a constant in the algorithm of Fu, Li, and Xu (2018). We note that our algorithm has runtime polynomial in $k$, but it is without loss to assume that $k$ is input in unary, since if $k$ (capacity) exceeds the number of given distributions (applicants/candidates) then the problem is trivial.

Specifically, for $\varepsilon \in (0, 1/e)$, let $k^*$ be the smallest $k \geq 1$ such that $\varepsilon > e^{-k}k^k/(k!)$.[3] For $k < k^*$, the PTAS will run the algorithm by Fu, Li, and Xu (2018) to obtain a $1 - \varepsilon$ approximation in polynomial time, since $k^*$ is a constant. For $k \geq k^*$, the PTAS will run $\mathsf{ALG}_{\texttt{ptk}}$ and obtain, in polynomial time, an approximation factor of $1 - e^{-k}k^k/(k!) > 1 - \varepsilon$.

**Corollary 1** *There is a PTAS for ProbeTop-$k$.*

[3] We know that $k^*$ always exists since $e^{-k}k^k/(k!)$ is a decreasing function that equals $1/e$ when $k = 1$ and converges to 0 as $k \to \infty$.

### 2.4.6 Weighted Bernoulli Values and the Sequential Offering Problem

ProbeTop-$k$ is closely related to the Sequential Offering problem (`seq`) studied by Purohit, Gollapudi, and Raghavan (2019). In this problem, the hiring firm has already interviewed $n$ candidates and must decide the order in which to send offers to them. Each candidate $i$ has a probability $q_i$ of accepting an offer and adds a value $r_i$ to the firm if they accept an offer. The firm can hire at most $k$ candidates and send at most $T$ offers in total. This model is closely related to ProbeTop-$k$ when considering the special case where the value of applicant $i$ takes value $r_i$ with probability $q_i$ (representing acceptance of an offer), and 0 with probability $1 - q_i$ (representing rejection of an offer). The subtlety making `ptk` different from `seq` is that policies for `ptk` can reject an applicant $i$ even if they had a realization to value $r_i$, which is not allowed in `seq`. We show, however, that $\mathsf{ALG}_{\texttt{ptk}}$ can be forced to hire any applicant when their realized value is $r_i$ without loss, therefore making it an admissible algorithm for the Sequential Offering model with the same guarantee.

Indeed, we can rewrite $\mathrm{LP}_{\texttt{ptk}}$ for this special case as

$$\max \quad \sum_{i=1}^{n} r_i x_i$$

$$s.t. \quad x_i \le y_i q_i \ \ \forall i \in [n], \quad \sum_{i=1}^{n} y_i \le T, \quad \sum_{i=1}^{n} x_i \le k, \quad x_i \ge 0 \ \ \forall i \in [n], \quad 0 \le y_i \le 1 \ \ \forall i \in [n]$$

where $x_i$ is to be interpreted as the probability of hiring applicant $i$ and $V_i = r_i$ (we can omit the variable corresponding to $V_i = 0$ because it has a zero coefficient in the objective). The following lemma will let us restrict $\mathsf{ALG}_{\texttt{ptk}}$ to be admissible for the Sequential Offering problem.

**Lemma 5** *For weighted Bernoulli instances,* $\mathrm{LP}_{\texttt{ptk}}$ *has an optimal solution with* $x_i = y_i q_i \ \forall i \in [n]$.

The proof of this lemma is deferred to Appendix A.2.7.

Recall that after interviewing $i$, $\mathsf{ALG}_{\texttt{ptk}}$ will hire them with probability $x_i/(y_i q_i)$. Therefore, if we restrict to solutions with $x_i = y_i q_i$, then $\mathsf{ALG}_{\texttt{ptk}}$ will hire applicant $i$ with probability 1 if $V_i = r_i$, making it admissible for the Sequential Offering problem.

For this specific family of instances, equations (2.4) yield

$$p_i = \frac{x_i}{y_i} = x_i \frac{q_i}{x_i} = q_i, \quad v_i = \frac{r_i x_i}{x_i} = r_i.$$

We can use these expressions to further simplify the LP by removing variables $x_i$ and express it in terms of $p_i$ and $v_i$, leading to $\mathrm{LP}_{\texttt{seq}}$.

$$\begin{aligned} \mathrm{LP}_{\texttt{seq}}(I) = \max \quad & \sum_{i=1}^{n} v_i y_i p_i \\ s.t. \quad & \sum_{i=1}^{n} y_i \le T, \quad \sum_{i=1}^{n} y_i p_i \le k, \quad 0 \le y_i \le 1 \ \ \forall i \in [n]. \end{aligned}$$

Being consistent with the previously introduced notation, let $\mathcal{I}_{\texttt{seq}}$ be the set of all instances for the Sequential Offering problem. Let $\mathcal{I}^k_{\texttt{seq}}$ be the set of all instances of `seq` that have exactly $k$ positions to fill. Let $\mathsf{ALG}_{\texttt{seq}}$ be the modified version of $\mathsf{ALG}_{\texttt{ptk}}$ that, when faced with instances of weighted Bernoulli random variables, modifies the solution of the LP as in Lemma 5 such that the algorithm is admissible for `seq`. We then have the following corollary of Theorem 1.

**Corollary 2** *For any instance $I \in \mathcal{I}^k_{\texttt{seq}}$, we have $R_{\mathsf{ALG}_{\texttt{seq}}}(I) \ge \left(1 - \frac{e^{-k}k^k}{k!}\right) \mathrm{LP}_{\texttt{seq}}(I)$.*

### 2.5 Parallel Offering

We turn to the Parallel Offering model defined in Subsection 2.3.2. We provide an algorithm that achieves a $(1 - 1/e)$ approximation of the optimal policy. The algorithm works by solving a LP relaxation and rounding its solution to decide who to offer which position, and in which order.

In Subsection 2.5.1 we introduce the LP in question. In Subsection 2.5.2 we present our algorithm and in Subsection 2.5.3 we provide its analysis. In Subsection 2.5.4 we study the special case with identical positions introduced by Purohit, Gollapudi, and Raghavan (2019) and establish a connection between the Parallel and Sequential Offering problems.

### 2.5.1 Linear Program

For an instance $I \in \mathcal{I}_{\texttt{par}}$, we introduce $\text{LP}_{\texttt{par}}(I)$, with LP variables $y_{ij}$ for $i \in [n]$ and $j \in [k]$. Variable $y_{ij}$ is to be interpreted as the probability that candidate $i$ receives an offer for position $j$. As with $\text{LP}_{\texttt{ptk}}$, this LP only enforces that the problem's constraints are satisfied in expectation.

$$
\begin{aligned}
\text{LP}_{\texttt{par}}(I) = \max \quad & \sum_{j=1}^{k} \sum_{i=1}^{n} v_{ij} y_{ij} p_{ij} & & \\
s.t. \quad & \sum_{i=1}^{n} y_{ij} \leq T & \forall j \in [k] \quad & (2.6) \\
& \sum_{i=1}^{n} y_{ij} p_{ij} \leq 1 & \forall j \in [k] \quad & (2.7) \\
& \sum_{j=1}^{k} y_{ij} \leq 1 & \forall i \in [n] \quad & (2.8) \\
& 0 \leq y_{ij} \leq 1 & \forall (i,j) \in [n] \times [k]. &
\end{aligned}
$$

We start by formally proving that $\text{LP}_{\texttt{par}}$ upper-bounds the expected reward of any algorithm.

**Lemma 6** *For any instance $I \in \mathcal{I}_{\texttt{par}}$, we have $\text{LP}_{\texttt{par}}(I) \geq \mathsf{OPT}_{\texttt{par}}(I)$.*

The proof of this lemma is deferred to Appendix A.3.1.

### 2.5.2 Approximation Algorithm

We now present our algorithm for the Parallel Offering model, which we refer to as $\mathsf{ALG}_{\texttt{par}}$. The algorithm first solves $\text{LP}_{\texttt{par}}$ to produce an optimal solution $y$. With the solution at hand, the algorithm will round it to obtain a random binary matrix $Y = (Y_{ij})_{(i,j)\in[n]\times[k]}$. To round our solution here, we use the dependent rounding scheme developed by Gandhi et al. (2006) that is described in Appendix A.1.2. Specifically, the nodes of one side of the bipartite graph are the $n$ applicants, and the nodes of the other side are the $k$ positions. The weight of each edge $(i, j) \in [n] \times [k]$ is the fractional value of $y_{ij}$ from the optimal solution of $\text{LP}_{\texttt{par}}$. After the solution is rounded, the algorithm uses the rounded solution to make a sequential offering list for each position so that

candidate $i$ is included in the list for position $j$ if $Y_{ij} = 1$. The properties of the dependent rounding scheme by Gandhi et al. (2006) combined with the constraints of $\text{LP}_{\texttt{par}}$ will ensure that: (a) each list has at most $T$ candidates, and (b) each candidate is included in at most one list. After the lists are formed, all lists are run in parallel, and the order in which candidates $i$ for each position $j$ are sent offers is in decreasing order of $v_{ij}$.

### 2.5.3 Analysis of $\mathsf{ALG}_{\texttt{par}}$

For the analysis, let $L_j$ denote the expected value of the candidate that ends up being hired for position $j$. Define $Z_{ij} = P_{ij}Y_{ij}$ and $\tilde{Z}_{ij} = P_{ij}\tilde{Y}_{ij}$, where $(\tilde{Y}_{ij})_{i\in[n],j\in[k]}$ are independent Bernoulli random variables with $\mathbb{E}(\tilde{Y}_{ij}) = y_{ij}$. Let $D_j = \{i : Z_{ij} = 1\}$ and $\tilde{D}_j = \{i : \tilde{Z}_{ij} = 1\}$. Let $v_j^*(S) = \max_{i\in S} v_{ij}$. It is clear to see that $L_j = \mathbb{E}(v_j^*(D))$.

The first step for showing the guarantee of $\mathsf{ALG}_{\texttt{par}}$ is to show that the reward collected by a list is not lower than what we would collect if we rounded each component of $y$ independently. This is a consequence of the negative correlation property (P3) of the dependent rounding scheme by Gandhi et al. (2006). This result is formalized in the following lemma.

**Lemma 7** *For all $j \in [k]$, we have $\mathbb{E}(v_j^*(D)) \geq \mathbb{E}(v_j^*(\tilde{D}))$.*

The proof of this lemma is deferred to Appendix A.3.2. We note that Lemma 7 analyzes each position in isolation, even if two or more positions are identical. In fact, our proof does not appear to extend to multiple identical positions. Indeed, our proof only makes use of properties (P1) and (P3) from the dependent rounding scheme of Gandhi et al. (2006). We prove in Appendix A.3.3 that only using these properties is insufficient for proving an analogue of Lemma 7 for multiple positions.

Continuing with the main result, let $L_j^*$ denote $\sum_{i=1}^n v_{ij}y_{ij}p_{ij}$, so that the objective function of $\text{LP}_{\texttt{par}}$ can be expressed as $\sum_{j=1}^k L_j^*$. The following lemma helps establish the desired bound for each separate list; note that at most one candidate is hired from each list. Both the correlation gap results and the rounding scheme mentioned in Appendix A.1 are used in our analysis.

**Lemma 8** *For all $j \in [k]$, we have $\mathbb{E}(v_j^*(\tilde{D})) \geq (1 - 1/e)L_j^*$.*

The proof of this lemma is deferred to Appendix A.3.4.

By combining Lemma 7 and Lemma 8 we obtain the main result of the section.

**Theorem 2** *For any instance $I \in \mathcal{I}_{\texttt{par}}$, we have $R_{\mathsf{ALG}_{\texttt{par}}}(I) \geq (1 - 1/e)\mathrm{LP}_{\texttt{par}}(I)$.*

In Appendix A.3.5 we establish that the bound achieved by our algorithm is tight.

### 2.5.4 Identical Positions and the Cost of Batching

We turn to the special case where all positions are identical (i.e. $v_{ij} = v_i$ and $p_{ij} = p_i$ for all $i \in [n]$ and $j \in [k]$). For this case, we can obtain a connection between the Sequential Offering model and the Parallel Offering model through their respective LPs.

Let $\mathcal{I}^{=}_{\texttt{par}} \subseteq \mathcal{I}_{\texttt{par}}$ be the set of instances of `par` in which all positions are identical. For a given instance $I \in \mathcal{I}^{=}_{\texttt{par}}$, construct $I' \in \mathcal{I}_{\texttt{seq}}$ with the same candidates as in $I$, but with a budget of $kT$ sequential offers (instead of $T$ parallel offering rounds). We obtain the following lemma.

**Lemma 9** $\mathrm{LP}_{\texttt{par}}(I) = \mathrm{LP}_{\texttt{seq}}(I')$.

The proof of this lemma is deferred to Appendix A.3.6.

Lemma 9 implies the following corollary, which we refer to as the cost of batching.

**Corollary 3** *For any $I \in \mathcal{I}^{=}_{\texttt{par}}$, we have $R_{\mathsf{ALG}_{\texttt{par}}}(I) \geq (1 - 1/e)\mathsf{OPT}_{\texttt{seq}}(I')$.*

This corollary helps us understand how costly it can be to send offers in batches instead of one by one like in the Sequential Offering problem. By reducing to $T$ parallel offering rounds instead of $kT$ sequential offering rounds, we know that we cannot be worse by more than a factor of $(1-1/e)$.

## 2.6 Simultaneous Offering

In this section, we study the Simultaneous Offering problem described in Subsection 2.3.3. We are interested in a class of "value-ordered" policies. We develop such a policy that, when faced

with instances where all values of candidates are lower-bounded by some $\tau \in (0, 1)$, achieves an $\alpha_k^\tau$-approximation, where $\alpha_k^\tau$ is an increasing function that maps the number of positions $k \in \mathbb{N}$ and the lower bound $\tau$ to a real number between 0 and 1. Our algorithm first solves an LP relaxation. It then uses a modification of the optimal solution to decide which candidates will receive offers.

In Subsection 2.6.1 we define the class of value-ordered policies, which we show can perform arbitrarily poorly without the assumption of the lower bound $\tau$. In Subsection 2.6.2 we introduce an LP relaxation that is used in our algorithm. In Subsections 2.6.3 and 2.6.4 we introduce and analyze our algorithm, providing lower bounds on how well value-ordered policies can perform.

### 2.6.1 Value-ordered Policies

For this problem, we focus on a natural class of policies, which we call *value-ordered policies*. Assume that $v_1 \geq v_2 \geq \cdots \geq v_n$. A value-ordered policy is defined by an integer $m \in [n]$, and sends offers to the $m$ candidates with the highest values (i.e. $S = \{1, \ldots, m\}$). This family of policies is practically well-motivated because a firm generally does not want to withhold sending offers to high-value candidates whom it deems "too good" for itself. The optimal value-ordered policy also can be obtained efficiently by solving a dynamic program (see Appendix A.4.1 for details).

As natural as they seem, value-ordered policies can achieve an arbitrarily poor approximation factor. Consider the following simple example.

**Example 1** *Consider an instance with $k = 1$ and $n = 2$. For small $\varepsilon$, candidate 1 has $v_1 = p_1 = \varepsilon$, while candidate 2 has $v_2 = \varepsilon(1-\varepsilon)$ and $p_2 = 1$. There are three possible policies for this instance: $\{1, 2\}$, $\{1\}$ and $\{2\}$. The two first are value-ordered policies. The reward for using $\{1, 2\}$ is 0, for using $\{1\}$ is $\varepsilon^2$, and for using $\{2\}$ is $\varepsilon(1-\varepsilon)$. For $\varepsilon < 1/2$ this means that the optimal policy is $\{2\}$ and the optimal value-ordered policy is $\{1\}$. The approximation factor achieved by value-ordered policies in this instance is equal to $\varepsilon/(1-\varepsilon)$, which can be made arbitrarily small by taking $\varepsilon \to 0$.*

A similar example can be constructed given any amount of positions $k$, as we show later in Theorem 4. The intuition behind the previous example is that when $\varepsilon$ is small, adding a candidate

over capacity provides negligible benefit compared to cost, making the capacity $k$ almost a hard constraint. And although the difficulties of this specific example can be avoided by ordering the candidates in descending order of $v_i p_i$ instead of $v_i$, or using the greedy policy which starts with an empty set of candidates and iteratively adds the candidate with the highest marginal benefit to the set, we show that these alternate algorithmic ideas can also achieve an arbitrarily poor approximation in Appendix A.4.2.

We hereafter aim for parametric guarantees for value-ordered policies under the assumption that the values of the candidates are lower-bounded by a parameter $\tau > 0$. Let $\mathcal{I}^{\tau}_{\texttt{sim}} = \{I \in \mathcal{I}_{\texttt{sim}} : v_i \geq \tau \ \forall i \in [n]\}$ be the set of $\tau$-bounded instances. Since we normalized $c$ (the cost of hiring each candidate over capacity) to be 1, the value of $\tau$ can be interpreted as how hard or soft the capacity constraint of $k$ is in practice. For example, if $\tau$ is close to 1, then the capacity constraint can be viewed as soft, since the cost of hiring over capacity would be almost fully compensated by the value of any candidate. If the capacity constraint is hard in practice, then $\tau$ could be close to 0.

We remark that if $\tau$ is only small because of "irrelevant" low-value candidates who would never receive an offer, then it does not negatively affect our results. More precisely, our guarantee depends on the lowest value of a candidate with positive mass in the solution of the LP introduced in Subsection 2.6.2. We ignore this distinction in our definition of $\tau$-bounded instances for simplicity.

We say that a policy $\pi$ is an $\alpha$-approximation for $\tau$-bounded instances if

$$\inf_{I \in \mathcal{I}^{\tau}_{\texttt{sim}}} \frac{R_\pi(I)}{\mathsf{OPT}_{\texttt{sim}}(I)} \geq \alpha.$$

Notice that if $\tau \geq 1$ then the problem is trivial: it is optimal to send an offer to every candidate. We will therefore restrict our attention to $\tau \in (0, 1)$.

Before we carry on, we define $V_{\texttt{high}} = \{i \in [n] : v_i > 1\}$ to be the set of candidates whose value is greater than 1. These are candidates that we would like to hire even if we know they would violate capacity, essentially making the number of positions random, as they will all receive offers

but it is uncertain how many would accept. To our understanding, there is no easy way to reduce the problem to one where they do not exist, when both $V_{\texttt{high}}$ and $[n] \setminus V_{\texttt{high}}$ are non-empty.

### 2.6.2 LP Relaxation

In order to obtain approximation factors for value-ordered policies, we introduce a linear programming relaxation of $\mathsf{OPT}_{\texttt{sim}}$ which we call $\mathrm{LP}_{\texttt{sim}}$.

$$\begin{aligned} \mathrm{LP}_{\texttt{sim}}(I) = \max_{y,z} \quad & \sum_{i\in[n]} v_i p_i y_i + z \\ s.t. \quad & z \le k - \sum_{i\in[n]} p_i y_i \\ & z \le 0 \\ & 0 \le y_i \le 1 \quad \forall i \in [n]. \end{aligned}$$

As with the previous linear programs, $y_i$ is to be interpreted as the probability that candidate $i$ receives an offer. This linear program optimizes over randomized policies that pay a penalty for the difference between the expected number of candidates hired and $k$, rather than the realized number of candidates hired in excess of $k$. Thus, by applying Jensen's inequality we can show that $\mathrm{LP}_{\texttt{sim}}$ is an upper bound of $\mathsf{OPT}_{\texttt{sim}}$.

**Lemma 10** *For any $I \in \mathcal{I}_{\texttt{sim}}$, we have $\mathrm{LP}_{\texttt{sim}}(I) \ge \mathsf{OPT}_{\texttt{sim}}(I)$.*

The proof of this lemma is deferred to Appendix A.4.3.

We proceed to show a property about optimal solutions of $\mathrm{LP}_{\texttt{sim}}$ that closely relates it to a value-ordered policy.

**Lemma 11** *Let $I \in \mathcal{I}_{\texttt{sim}}$. There exists an optimal solution $(y, z)$ of $\mathrm{LP}_{\texttt{sim}}(I)$ with an index $j$ such that $y_i = 1$ for $1 \le i \le j-1$, $y_j > 0$, and $y_i = 0$ for $j+1 \le i \le n$.*

The proof of this lemma follows from the fact that, for any fixed $z$, $\mathrm{LP}_{\texttt{sim}}$ corresponds to an instance of the Fractional Knapsack problem. It is well-known that optimal solutions to this problem

satisfy the structure described in Lemma 11 (Goodrich and Tamassia 2001, Chapter 5), so it is true for any optimal choice of $z$. Given Lemma 11, we can easily construct a randomized value-ordered policy from an optimal solution of $\mathrm{LP}_{\mathtt{sim}}$. Indeed, we could simply send an offer to each candidate $i$ independently with probability $y_i$. This policy randomizes between two value-ordered policies: sending offers to candidates $\{1, \ldots, j\}$ or to candidates $\{1, \ldots, j-1\}$. We will make use of this idea when constructing our actual approximation algorithm.

An important object in the definition and analyses of our policies is the "total mass" of a solution of $\mathrm{LP}_{\mathtt{sim}}$. For a feasible solution $y$, its total mass is given by $\sum_{i\in[n]} y_i p_i$. This can be interpreted as the expected amount of candidates that would be hired if we were to send an offer to each candidate $i$ with probability $y_i$. Lemma 11 also implies that an optimal solution is completely determined by its total mass. This is because the solution can be constructed by "filling" the components from smallest to largest index until we obtain the desired total mass.

The following lemma allows us to determine the total mass of optimal solutions of $\mathrm{LP}_{\mathtt{sim}}$ based on aggregate parameters of instances and will be useful later in the analysis.

**Lemma 12** *Let $I \in \mathcal{I}_{\mathtt{sim}}$. Let $(y, z)$ be an optimal solution of $\mathrm{LP}_{\mathtt{sim}}(I)$ that satisfies the structure given in Lemma 11. Let $P_{\mathtt{high}} = \sum_{i\in V_{\mathtt{high}}} p_i$ and $P_{\mathtt{total}} = \sum_{i\in[n]} p_i$. The following holds:*

1. *If $P_{\mathtt{high}} > k$, then $y_i = 1$ for all $i \in V_{\mathtt{high}}$ and $y_i = 0$ otherwise. Therefore, $\sum_{i\in[n]} y_i p_i = P_{\mathtt{high}}$;*
2. *If $P_{\mathtt{high}} \leq k$, then $\sum_{i\in[n]} y_i p_i = \min\{k, P_{\mathtt{total}}\}$.*

The proof of this lemma is deferred to Appendix A.4.4.

### 2.6.3 Approximation Algorithm

Let us introduce our approximation algorithm for $\mathtt{sim}$, which we call $\mathsf{ALG}^s_{\mathtt{sim}}$. The policy takes as input an optimal solution $y$ of $\mathrm{LP}_{\mathtt{sim}}$ and a parameter $s \in [0, 1]$. The idea of the policy is to truncate the optimal solution of the LP. This truncation is done by scaling down the total mass of the optimal LP solution by a factor $s$ and then using this mass to "fill" the new variables

from smallest to largest index. We then use this truncated solution to decide which candidates will receive offers.

Formally, the first step is to construct an alternative solution $y'$ by truncating $y$ the following way. Let $\kappa = s \sum_{i\in[n]} y_i p_i$ be the total mass of the optimal LP solution, scaled down by a factor $s$. Let $j$ be the first index such that $\sum_{i=1}^{j+1} p_i > \kappa$. We let $y'_i = 1$ for $1 \le i \le j$, $y'_i = 0$ for $j+2 \le i \le n$, and set $y'_{j+1}$ such that $\sum_{i=1}^{j+1} y'_i p_i = \kappa$. After solution $y'$ is constructed, the offers are sent independently to each candidate $i$ with probability $y'_i$. Given the structure of $y'$, $\mathsf{ALG}^s_{\mathtt{sim}}$ clearly randomizes between two value-ordered policies: sending offers to candidates 1 through $j$, and sending offers to candidates 1 through $j+1$. For any $k$ and $\tau$, parameter $s$ can be optimized to maximize the algorithm's performance, and we let $s^*$ denote the optimal value. Specifically, $s^*$ is defined as an optimal solution to the problem

$$\sup_{0\le s\le 1} \left( s - \frac{s}{\tau} + \frac{\mathbb{E}[\min\{\mathrm{Pois}(sk), k\}]}{\tau k} \right).$$

We will use $s^*$ only in our analysis to lower-bound the performance of the best value-ordered policy.

### 2.6.4 Analysis of $\mathsf{ALG}_{\mathtt{sim}}$

The analysis of $\mathsf{ALG}^s_{\mathtt{sim}}$ and $\mathsf{ALG}^{s^*}_{\mathtt{sim}}$ is done in two cases. One case is $\sum_{i\in V_{\mathtt{high}}} p_i \le k$, in which sending an offer to all candidates in $V_{\mathtt{high}}$ hires an expected number of candidates less than or equal to $k$. The other case is $\sum_{i\in V_{\mathtt{high}}} p_i > k$.

**Case 1:** $\sum_{i\in V_{\mathtt{high}}} p_i \le k$**.** In this case, we can show a lower bound for the performance of $\mathsf{ALG}^s_{\mathtt{sim}}$ that depends on the choice of $s$.

**Lemma 13** *For any $s \in [0,1]$, $\tau \in (0,1)$, and $I \in \mathcal{I}^\tau_{\mathtt{sim}}$ with $\sum_{i\in V_{\mathtt{high}}} p_i \le k$, we have*

$$R_{\mathsf{ALG}^s_{\mathtt{sim}}}(I) \ge \left( s - \frac{s}{\tau} + \frac{\mathbb{E}[\min\{\mathrm{Pois}(sk), k\}]}{\tau k} \right) \mathrm{LP}_{\mathtt{sim}}(I).$$

The proof of this lemma is deferred to Appendix A.4.5.

**Case 2:** $\sum_{i \in V_{\texttt{high}}} p_i > k$**.** In this case, we can show that for an optimally chosen parameter $s^*$, the guarantee obtained by $\mathsf{ALG}^{s^*}_{\texttt{sim}}$ cannot be worse than the previous case where $\sum_{i \in V_{\texttt{high}}} p_i \leq k$. A key observation for analyzing this new case is that as Lemma 12 states, all candidates $i$ whose value $y_i$ in the optimal LP solution will have $v_i \geq 1$, so $s^* = 1$. Indeed, setting $s < 1$ can only decrease the chances of sending offers to candidates in $V_{\texttt{high}}$, and these are candidates that the firm would always want to hire (even over capacity). With this observation in hand, we prove the following, via the construction of an auxiliary instance in which the probabilities of acceptance are deflated.

**Lemma 14** *Let $I = (k, n, p, v) \in \mathcal{I}^{\tau}_{\texttt{sim}}$ be an instance with $\sum_{i \in V_{\texttt{high}}} p_i > k$. Then there exists an instance $I' = (k, n, p', v) \in \mathcal{I}^{\tau}_{\texttt{sim}}$ such that $\sum_{i \in V_{\texttt{high}}} p'_i = k$ and*

$$\frac{R_{\mathsf{ALG}^{s^*}_{\texttt{sim}}}(I)}{\mathrm{LP}_{\texttt{sim}}(I)} \geq \frac{R_{\mathsf{ALG}^{s^*}_{\texttt{sim}}}(I')}{\mathrm{LP}_{\texttt{sim}}(I')}$$

The proof of this lemma is deferred to Appendix A.4.6.

By combining Lemma 13 and Lemma 14 for an optimally chosen parameter $s^*$, we obtain the main result of this section.

**Theorem 3** *For any $\tau \in (0, 1)$ and $I \in \mathcal{I}^{\tau}_{\texttt{sim}}$ we have $R_{\mathsf{ALG}^{s^*}_{\texttt{sim}}}(I) \geq \alpha^{\tau}_k \mathrm{LP}_{\texttt{sim}}(I)$, where*

$$\alpha^{\tau}_k = \sup_{0 \leq s \leq 1} \left( s - \frac{s}{\tau} + \frac{\mathbb{E}[\min\{\mathrm{Pois}(sk), k\}]}{\tau k} \right).$$

The performance of $\mathsf{ALG}^{s^*}_{\texttt{sim}}$ clearly dominates that of $\mathsf{ALG}^{1}_{\texttt{sim}}$, where $\mathsf{ALG}^{1}_{\texttt{sim}}$ directly follows the LP solution. By Lemma 13, $\mathsf{ALG}^{1}_{\texttt{sim}}$ has a guarantee of at least $1 - \frac{1}{\tau} + \frac{\mathbb{E}[\min\{\mathrm{Pois}(k), k\}]}{\tau k}$, which equals $1 - e^{-k} k^k / (k! \tau)$, as we derive in Appendix A.1.3. By combining this last expression with Stirling's approximation, we obtain the following result about the asymptotic optimality of $\mathsf{ALG}^{s^*}_{\texttt{sim}}$ when the number of positions grows large and the values of candidates are bounded away from 0.

**Corollary 4** *For any $\tau \in (0,1)$ and any instance $I \in \mathcal{I}^{\tau}_{\texttt{sim}}$, we have*

$$\mathsf{ALG}^{s^*}_{\texttt{sim}} \geq \left(1 - O\left(1/\sqrt{k}\right)\right) \mathrm{LP}_{\texttt{sim}}(I),$$

*where $k$ is the number of positions in the instance.*

We now provide an upper bound for the guarantee that can be achieved using value-ordered policies. We construct an instance consisting of $n = j^2 + k$ candidates[4], with $j$ being a (large) integer. The first $j^2$ candidates are of type 1, who have $p_i = k/j$ and $v_i = \tau + \delta$, where $\delta > 0$ is small. The remaining $k$ candidates are of type 2, who have $p_i = 1$ and $v_i = \tau$. The idea behind this result is that any optimal value-ordered policy will only send offers to type 1 candidates. On the other hand, the optimal policy does not perform worse than a policy that sends offers to $k$ type 2 candidates and zero type 1 candidates.

**Theorem 4** *For any $\varepsilon > 0$ and $\tau \in (0,1)$, there exists an instance $I \in \mathcal{I}^{\tau}_{\texttt{sim}}$ such that no value-ordered policy can have an expected reward greater than $\left(\beta^{\tau}_k + \varepsilon\right) \mathsf{OPT}_{\texttt{sim}}(I)$, where*

$$\beta^{\tau}_k = \sup_{s \geq 0} \left( s - \frac{s}{\tau} + \frac{\mathbb{E}[\min\{\mathrm{Pois}(sk), k\}]}{\tau k} \right).$$

The proof of this theorem is deferred to Appendix A.4.7. In Appendix A.4.8 we analyze the region in which our bounds coincide, together with plots showing $\alpha^{\tau}_k$ and $\beta^{\tau}_k$ for several values of $k$ and $\tau$.

## 2.7 Future Directions

We close the paper by pointing out natural follow-up research questions that arise from our work. An interesting open direction is to combine the sequential interviewing problem with the offering problem. This can be done by generalizing the sequential interviewing problem to a setting

[4]We do not need exactly $j^2$ candidates of type 1 for showing the result. Any amount of candidates $\ell_j$ with $\ell_j/j \to \infty$ as $j \to \infty$ will suffice.

where candidates are not assured to accept offers and instead have a probability of accepting which might depend on factors such as their realized value. The firm can, at the end of each period, decide to send offers to any candidate who has already been interviewed. Another open direction is to combine the Simultaneous Offering problem with the Parallel Offering problem, as implicitly suggested in Purohit, Gollapudi, and Raghavan (2019). In this setting, the firm could, at each round, send more offers than positions remaining and face the risk of hiring over capacity (at a cost). One last future direction on the modeling side is how the firm should behave when the problem parameters are inaccurate. In particular, acceptance probabilities can be very difficult to estimate, so developing algorithms that are robust to these parameters' misspecification could be of great interest to firms.

On the technical side, we believe the Simultaneous Offering problem introduced is a parsimonious new variant of overbooking. Although our focus was to analyze the performance of value-ordered policies, we do not know of any hardness or algorithmic results for finding the best offer set. It would be interesting if an optimal or near-optimal (i.e., PTAS) algorithm could be found. The question of whether we can improve on $1 - 1/e$ for the sequential offering problem (or maybe even obtain a PTAS) is also open.

# Chapter 3: Optimizing Inventory Placement for a Downstream Online Matching Problem

## 3.1 Introduction

The supply chain of an e-commerce retailer is a large and complicated system whose overall operation results from an extensive stream of interdependent decisions. These decisions range all the way from long-term strategic decisions, including network design and sourcing, to mid-term tactical decisions, including inventory replenishment and placement, to real-time operational decisions, including order fulfillment and last-mile delivery (Chen and Graves 2021). The optimality of upstream decisions depends on how downstream decisions will be made, and vice-versa. However, jointly optimizing different stages is computationally and practically difficult (Ryzin 2024; Maggiar, Dicker, and Mahoney 2024). Therefore, a common approach for making upstream decisions is to abstractly model the downstream dynamics, or make simplifying assumptions on how downstream decisions will be made.

In this paper, we study the problem of an e-commerce retailer faced with making *inventory placement* decisions followed by on-the-fly *fulfillment* decisions. The e-commerce retailer manages a network of warehouses that can deliver goods to several different last-mile delivery hubs. Before orders arrive, the e-commerce retailer must decide how to split an incoming purchase of inventory units across the multiple warehouses it manages. This corresponds to the inventory placement problem. Afterward, the e-commerce retailer faces a stream of customer orders during a time horizon, which is random and therefore not known in advance. Whenever an order is made, the e-commerce retailer must decide which warehouse to deliver from, depleting a unit of inventory and collecting a reward. This sequential decision-making problem corresponds to the fulfillment problem. The goal of the e-commerce retailer is to maximize the expected reward collected dur-

ing the time horizon by deciding how to place inventory and how to fulfill demand. These two decisions are tightly linked together. On one hand, fulfillment decisions depend on the current inventory levels at each warehouse, which in turn depend on the initial inventory placement. On the other hand, the optimal inventory placement depends on how fulfillment decisions are going to be made. Despite this connection, only recently has there been increased attention on how these decisions interact and how to make them in a joint or coordinated manner (Govindarajan, Sinha, and Uichanco 2021b; Bai et al. 2022; Chen et al. 2022; Arlotto, Keskin, and Wei 2023; Jasin, Liu, and Zhao 2024).

As an isolated problem, online fulfillment has been well-studied by the operations research and computer science communities (Xu, Allgor, and Graves 2009; Acimovic and Graves 2015; Jasin and Sinha 2015). The main tool of analysis for this problem is modeling it as an online bipartite matching problem with stochastic inputs (Feldman et al. 2009; Alaei, Hajiaghayi, and Liaghat 2012; Brubach et al. 2020). These models take the initial inventories at different warehouses as input, and the policies developed aim to obtain a reward as close as possible to the best-achievable reward with that inventory. On the other hand, the inventory placement problem has been less studied. Here, a fixed quantity of an item that has already been purchased is to be distributed among a set of warehouses. A fundamental difference between the placement and fulfillment problems is that there are no downstream decisions after fulfillment is done, whereas the effectiveness of an inventory placement directly depends on the fulfillment policy being deployed. For a fixed inventory placement, finding the optimal fulfillment policy can be computationally challenging (Papadimitriou et al. 2021), which means that even evaluating the potential value of a proposed placement is hard.

To overcome this difficulty, an approach taken by the literature and also in practice is to (exactly or approximately) optimize a surrogate function that approximates the value of the optimal fulfillment policy. The following surrogates and corresponding placement procedures have been considered.

1. **Fluid Placement**: optimizes inventory placement for a fluid relaxation of the problem, in

which the total demand of each node is assumed to be deterministic and equal to its expected value. This procedure only requires knowing the first moments of demand distributions and is also the least computationally expensive. It has been used to prove some approximation guarantees for the joint placement and fulfillment problem (Bai, Rusmevichientong, and Topaloglu 2022; Chen et al. 2022).

2. **Offline Placement**: optimizes the expected reward obtained by a fulfillment policy that knows the total demand for each node in the given random scenario in advance and thus makes hindsight optimal fulfillment decisions. Govindarajan, Sinha, and Uichanco (2021b) and DeValve et al. (2023) advocate for this procedure based on their experiments. This procedure requires the distributional knowledge of total demands and the ability to enumerate over it, which can be difficult. Therefore, it is often approximated by the sampling of demand scenarios.

3. **Myopic Placement**: makes the simplifying assumption that the downstream fulfillment decisions will be made according to a greedy policy that maximizes immediate rewards, ignoring the opportunity cost of depleting a unit of inventory at a given warehouse. This is an approach present in the lines of work of Acimovic and Graves (2017) and Chen (2017), which presents it as a natural first step. Unlike the first two procedures, the surrogate optimized here is rather pessimistic about the sophistication of the downstream fulfillment team. The output of myopic fulfillment depends on the order in which demand arrives, so this information has to be incorporated when optimizing inventory placement for this surrogate, which is usually done through simulation.

Despite the aforementioned work, there is no clear consensus on which placement procedure is best. Fluid Placement comes with theoretical guarantees, but is shown to perform substantially worse than Offline Placement in real-world experiments (DeValve et al. 2023). Offline Placement, on the other hand comes with no theoretical guarantees, and the number of samples required for it to prescribe a good decision is also unclear. Finally, Myopic Placement is the most realistic

in terms of not expecting the fulfillment team to foretell demand in advance, but the speed and stability of simulation procedures is questionable.

In this paper, we take a deep dive into understanding how these three common approaches for optimizing inventory placement compare. Our contributions come both from deriving theoretical results and from numerical simulations. On the theoretical side, we show that when coupled with an appropriate, high-quality fulfillment policy, Offline Placement offers constant-factor guarantees for the joint placement and fulfillment problem. Moreover, our guarantee is strictly better than the best guarantees obtained by using Fluid Placement for the inventory placement, complementing the empirical findings of DeValve et al. (2023), and it holds on a strictly larger class of demand models. This result formalizes the notion that Offline Placement, a procedure that can be interpreted as being optimistic about the quality of the downstream fulfillment decisions, is desirable when the fulfillment policy being deployed is of high quality. Our numerical simulations corroborate our theoretical finding.

### 3.1.1 Theoretical Results and Techniques

The core of our theoretical contribution is an approximation algorithm for inventory placement optimization under the Offline surrogate, which is developed in **Section 3.3**. This is a fundamental integer programming problem related to matching, defined as follows. We are given a bipartite graph with warehouses $i \in [n] := \{1, \ldots, n\}$, demand nodes $j \in [m]$, and rewards $r_{ij} \geq 0$ such that $|\{i : r_{ij} > 0\}| \leq d$ for all $j$. Here, $d$ measures fulfillment flexibility, which can also be interpreted as an upper bound on the number of warehouses that receive a positive reward for serving any given demand location. (Our results are relevant even if $d = \infty$, representing full flexibility.) A demand vector $D = (D_1, \ldots, D_m)$ is drawn from an arbitrary distribution over $\mathbb{Z}^m_{\geq 0}$. The problem is to decide an inventory placement $x = (x_1, \ldots, x_n) \in \mathbb{Z}^n_{\geq 0}$, subject to a constraint $x_1 + \cdots + x_n = Q$ where the total inventory $Q$ has been decided already, to maximize the expected

value of

$$\begin{aligned}
\mathsf{OFF}(x, D) = \max_{y \geq 0} \quad & \sum_{i \in [n]} \sum_{j \in [m]} y_{ij} r_{ij} \\
\text{s.t.} \quad & \sum_{i \in [n]} y_{ij} \leq D_j \quad \forall j \in [m], && (3.1) \\
& \sum_{j \in [m]} y_{ij} \leq x_i \quad \forall i \in [n], && (3.2)
\end{aligned}$$

over the randomly-drawn demand vector $D$. Assuming for now that the distribution of $D$ has small support, the fractional relaxation of the placement problem,

$$\max_{x \in \mathbb{R}^m_{\geq 0}, \sum_i x_i = Q} \mathbb{E}_D[\mathsf{OFF}(x, D)], \tag{3.3}$$

can be formulated as an LP by enumerating the possible realizations of $D$. It is important to note that the optimal solution of (3.3) can indeed be fractional even though the bipartite matching LP for any fixed $D$ is integral. In fact, we show that the value of (3.3) can shrink by a factor of $1 - (1 - 1/d)^d$ when $x$ is constrained to lie in $\mathbb{Z}^n_{\geq 0}$ (Lemma 17 in **Subsection 3.3.1**). Our main result is to show that this factor is in fact tight, i.e. any fractional placement $x$ can be rounded into an integer placement $R(x) = (R_1(x), \ldots, R_n(x)) \in \mathbb{Z}^n_{\geq 0}$ such that $\mathbb{E}_D[\mathsf{OFF}(R(x), D)] \geq (1 - (1-1/d)^d)\mathbb{E}_D[\mathsf{OFF}(x, D)]$ (Theorem 5 in **Subsection 3.3.1**). We develop a randomized rounding for this problem based on applying the fundamental rounding procedure of Gandhi et al. (2006) in two iterations. In the first iteration, we round the inventory placements obtained in (3.3) and in the second iteration we round the LP variables defining $\mathsf{OFF}(x, D)$ for every possible outcome of the demand vector $D$. This second iteration rounding must depend on the outcome of the first iteration, which requires a new rounding subroutine that we describe in **Subsection 3.3.1**.

Our use of the Offline relaxation assumes that we can enumerate over the support of $D$, whereas in general, the support of $D$ has exponential size. To fix this, we show that given $K$ independent and identically distributed (IID) samples, we can still obtain an approximation ratio of $1-(1-1/d)^d - O(Q\sqrt{(n \log n)/K})$ when Offline Placement solves a sample-average approxima-

tion of (3.3) using the $K$ samples, whose size is polynomial in $K$. We use Rademacher complexity and a vector contraction inequality (Maurer 2016) to show that the sample-average approximation provides a uniform approximation of $\mathbb{E}_D[\mathsf{OFF}(x, D)]$ over the continuous feasible region $\{x \in \mathbb{R}^m_{\geq 0} : \sum_i x_i = Q\}$ (Theorem 6 in **Subsection 3.3.2**), which completes the argument. Although this argument is simple in hindsight, we remark that a typical analysis of sample-average approximation for optimization problems (Shapiro and Nemirovski 2005, e.g.) does not go through uniform convergence. Consequently, we would lose an additional factor of 2 in the approximation ratio instead of losing a factor that is vanishing as $K \to \infty$ if the optimization problem is not solved to optimality.

In **Section 3.4** we apply these results to the joint placement and fulfillment problem under the assumption that our demand model admits an $\alpha$-competitive online fulfillment policy. This means that the setting is such that, no matter what initial inventories we have, we can efficiently compute a fulfillment policy that collects on average at least $\alpha$ times what we would collect if we knew the stream of customer orders in advance and made hindsight optimal fulfillment decisions. We show that, when paired with this $\alpha$-competitive fulfillment policy, Offline Placement collects a reward that is at least $\alpha(1-(1-1/d)^d)$ times what the optimal joint placement and fulfillment procedures would collect, where $d$ is the maximum number of warehouses that can fulfill a single demand node (Theorem 7 in **Subsection 3.4.2**). Our general treatment of demand models allows us to recover existing and obtain new results in relevant settings. For example, the Temporal Independence model (classic revenue management and online matching model with deterministic, known time horizon and independent arrivals; e.g., Alaei, Hajiaghayi, and Liaghat (2012)) allows a $1/2$-competitive fulfillment policy. In this setting, we recover the $(1-1/e)/2$-approximation implied by Bai, Rusmevichientong, and Topaloglu (2022) and strictly improve it for small values of $d$. To obtain their result, Bai, Rusmevichientong, and Topaloglu (2022) uses Fluid Placement, so our result also provides theoretical justification for why Offline appears to outperform Fluid Placement in the experiments of DeValve et al. (2023). Another example setting is the Spatial Independence model of Aouad and Ma (2022). Here, the authors establish a $1/2$-competitive fulfillment policy, so our

result implies a $(1-(1-1/d)^d)/2$-approximation in this setting. Importantly, the 1/2-competitive fulfillment policy of Aouad and Ma (2022) only works against the weaker Offline surrogate, so previous approaches would not have obtained any constant-factor approximation ratios at all using the Fluid surrogate.

In **Section 3.5** we show how our randomized rounding result for the Offline surrogate extends to the multi-SKU setting from Bai, Rusmevichientong, and Topaloglu (2022). In this setting, the e-tailer stores and faces demand for $s$ different SKUs that must share capacities at the different warehouses. The key observation for extending our result to this setting is that after using the algorithm of Gandhi et al. (2006) in the first iteration to round the inventory placement, the fulfillment decisions can be separated across SKUs. Therefore, we can use the same second iteration rounding for all pairs of SKUs and demand vectors for the particular SKU. That way, our rounding guarantee of $1-(1-1/d)^d > 1-1/e$ still holds in this multi-SKU setting. Importantly, any attempt to prove this result using submodularity would now have to consider the intersection of two partition matroids, for which the best-known approximation factor is only 1/2 (Lee et al. 2010), worse than our guarantee even if $d = \infty$.

### 3.1.2 Numerical Results From Simulations

In **Section 3.6**, we complement our theoretical results with simulations. We evaluate how different placement and fulfillment procedures perform when deployed together in a wide array of instances with different network structures, demand models, and load factors (ratio between expected demand and number of inventory units). The main difference between our theoretical results and our simulations is that we drift from the adversarial nature. Indeed, whereas our theoretical results provide worst-case guarantees for all possible network structures and demand distributions for a given demand model, our simulations consider particular instances motivated by the fulfillment literature.

We evaluate four different placement procedures: Offline Placement, Myopic Placement, and two versions of Fluid Placement. The fulfillment procedures evaluated are Myopic Fulfillment

and several variants of linear programming-based policies that use the shadow prices of their constraints to approximate and incorporate the opportunity costs of depleting inventory from specific warehouses. The variants depend on the linear program being solved (fluid or offline) and whether they include re-solving or not.

Consistent with DeValve et al. (2023), we find that the fulfillment policy that uses the shadow prices of the offline linear program and re-solves periodically has the best performance. Moreover, when this fulfillment policy is deployed, Offline Placement achieves the best performance among the four placement procedures we benchmark. This result is robust to varying network structures, demand models, and load factors. This again backs the notion that deploying Offline Placement leads to good performance, given that the fulfillment policy is of high quality. We also find that Myopic Placement outperforms both versions of Fluid Placement in most of our instances, suggesting that using a placement procedure that takes into account the uncertainty of the demand (Offline Placement or Myopic Placement) is consistently better than using deterministic approximations that use average demand as input (Fluid Placement).

Table 3.1 presents a summary of how the three placement procedures of interest compare and how our results contribute to this understanding.

### 3.1.3 Further Related Work

**Joint inventory placement and fulfillment.** Chen et al. (2022) studies the joint inventory placement and fulfillment problem with multiple items, obtaining a $(1 - 1/e)/4$ worst-case guarantee, and demonstrating its applicability at Anheuser Busch InBev (ABI). Bai, Rusmevichientong, and Topaloglu (2022) study the joint inventory placement, promise, and fulfillment problem with multiple items and warehouse capacity constraints, obtaining a $1/4$ worst-case guarantee. Their proposed inventory placement uses submodular optimization techniques to obtain an approximate solution for a fluid linear program. Bai et al. (2022) studies the more general problem where the downstream decisions correspond to deciding an assortment, obtaining a $(1 - 1/e)/4$-approximation. Govindarajan, Sinha, and Uichanco (2021b) study the joint distribution and place-

| **Placement Procedure** | **Offline Placement** (OFF) | **Myopic Placement** (MYO) | **Fluid Placement** (FLU) |
|---|---|---|---|
| Information Required | Medium (all moments of aggregate demand for each node) | High (exact demand arrival sequence) | Low (first moment of aggregate demand for each node) |
| Computation Time | Requires sampling | Requires simulation | Fastest (still no poly-time algorithm) |
| Surrogate Approximation | $1-(1-1/d)^d$ **(∗)** (LP and rounding) | No known guarantees | $1-1/e$ (Bai, Rusmevichientong, and Topaloglu 2022) (Submodular optimization) |
| Overall Approximation | $\alpha(1-(1-1/d)^d)$ **(∗)** (Wtih $\alpha$-approximate fulfillment policy against Offline) | No known guarantees | $\alpha(1-1/e)$ (Bai, Rusmevichientong, and Topaloglu 2022) (Wtih $\alpha$-approximate fulfillment policy against Fluid) |
| Empirical Justification | Beats FLU (DeValve et al. 2023) Beats MYO given a good fulfillment policy **(∗)** | Beats FLU **(∗)** | |

Table 3.1: Comparison among three placement procedures. Entries accompanied by a **(∗)** represent contributions of this paper. Our surrogate approximation guarantee also extends seamlessly to the multi-SKU constrained-warehouse setting.

ment problem in the context of an omni-channel retailer. They provide an inventory placement heuristic that is, as the one we analyze, based on solving the optimal inventory placement problem assuming fulfillment will be hindsight-optimal, although they do not provide theoretical guarantees. Arlotto, Keskin, and Wei (2023) study the single-item joint inventory placement and fulfillment problem through the lens of regret minimization. They obtain sublinear regret for two state-of-the-art fulfillment policies, assuming that the initial inventory is optimal for the specific policy. All of these models use the classic Temporal Independence demand model, whereas the results we obtain also hold in more general settings. Jasin, Liu, and Zhao (2024) study how Offline Placement performs in the cost minimization version of the joint placement and fulfillment problem. They provide several instances where Offline Placement can perform arbitrarily badly if coupled with Myopic Fulfillment, highlighting the importance of deploying this placement with a high-quality fulfillment policy.

**Other coordinated decisions in e-commerce retail.** The recent trend of studying coordinated decisions in e-commerce retail goes beyond inventory placement and fulfillment. Lei, Jasin, and Sinha (2018) study the problem of jointly pricing and fulfillment for different items of a catalog during a fixed time horizon. Lei et al. (2022) study the more general problem of joint product framing and order fulfillment under the multinomial logit model. They solve a deterministic approximation for the stochastic control problem and develop a randomized rounding scheme for this solution. Jasin et al. (2022) studies a joint inventory replenishment, pricing, and fulfillment problem with multiple stores and warehouses. They develop a Lagrangian based heuristic that achieves sub-linear regret in the time horizon. DeValve et al. (2023) primarily studies the effects of flexibility in e-commerce retail supply chains, but their work takes a look at how several aspects interact. Through numerical experiments using real-world data, they study the interaction of fulfillment, placement, and network design. In particular, they find that if the deployed fulfillment policy is myopic, adding flexibility to the network can increase total costs. DeValve, Pekeč, and Wei 2023 study the two-stage problem of deciding a network structure followed by fulfillment decisions, which they assume can be made in an offline fashion, and derive constant factor guarantees. Zhao

et al. (2023) studies an inventory replenishment and fulfillment problem in a network structure that resembles the JD.com network that we study. DeValve 2023 studies, among other problems, the multi-item joint inventory buying, placement, and fulfillment problem with the presence of fixed costs, where orders can contain several items. They show that it is NP-hard to approximate this problem up to a constant factor. They also derive approximation guarantees for a broad range of problems.

**Newsvendor Networks.** Van Mieghem (1998) and Mieghem and Rudi (2002) study a multi-dimensional newsvendor model, closely related to optimizing inventory assuming that fulfillment will be hindsight optimal. The difference with inventory placement is that in the latter, the total number of units, $Q$, has already been decided exogenously. Closely related is Govindarajan, Sinha, and Uichanco (2021a), where they study a multi-location newsvendor network model where the only information known about the joint demand distribution is the mean vector and covariance matrix. They take a distributionally robust approach to find the inventory decision that minimizes the worst-case expected cost of fulfilling demand. Birge and DeValve (2025) study the problem of purchasing and placing inventory units on a network before facing uncertain demand under the assumption that fulfillment will be hindsight optimal, deriving complexity and approximation results. For recent developments and an in-depth literature review related to Newsvendor networks, we refer the reader to DeValve and Myles (2023).

### 3.2 Problem Statement

We study the two-stage problem faced by an e-commerce retailer (e-tailer) who has to make an inventory placement decision at the beginning of a time horizon (first stage), followed by sequential fulfillment decisions during that time horizon (second stage). The e-tailer manages a supply chain network consisting of warehouses $i \in [n]$ that can hold inventory and serve different demand nodes $j \in [m]$. Demand node $j$ can be thought of as a specific district or ZIP code that we call $j$. When demand from node $j$ arrives, the e-tailer must decide from which warehouse (if any) that demand is going to be satisfied. If the e-tailer decides to deliver the item from warehouse $i$, they collect

a reward of $r_{ij} \in [0, 1]$. (Here, we are assuming that the rewards are bounded and normalized to lie between 0 and 1.) We say that warehouse $i$ can serve demand of node $j$ if $r_{ij} > 0$. For each demand node $j \in [m]$, we define its degree $d_j := |\{i \in [n] : r_{ij} > 0\}|$ as the number of warehouses that can serve the demand of this node. We further define $d := \max_{j \in [m]} d_j$. Parameter $d$ is an indicator of fulfillment flexibility in the network, and our guarantees will depend on it. (Here we assume a single SKU, but we extend our results to the multi-SKU constrained-warehouse setting in Section 3.5.)

**First stage problem – Inventory placement.** In the inventory placement problem, the e-tailer must distribute an incoming shipment of $Q \in \mathbb{Z}_{\geq 0}$ inventory units across the $n$ distribution centers they manage, where $\mathbb{Z}_{\geq 0}$ are the non-negative integers. Formally, the e-tailer must decide on an inventory placement in $\mathcal{X} := \{x \in \mathbb{Z}^n_{\geq 0} : \sum_{i \in [n]} x_i = Q\}$. We note that $Q$ is not a decision variable since we assume that an upstream team has already decided the total quantity to stock. (Including the total quantity $Q$ to stock at a unit cost of $c$, and investigating the overall performance of the system under different surrogates is an interesting research direction.) As a remark, all of our results directly extend to the setting where each warehouse has a capacity limiting the number of stock units that can be placed there, and where each warehouse also has a starting inventory. We omit these aspects for the sake of exposition, and refer the reader to Appendix B.3 for a brief explanation on how to extend our algorithms to this setting.

**Second stage problem – Fulfillment.** The fulfillment problem takes an inventory placement as input and faces a random stream of demand arriving from different nodes. We say that the total number of demand arrivals is $T$, which we call the time horizon and is generally random. We characterize the sequence of demand arrivals by a random vector $J = (j_t)_{t=1}^T$, where $j_t = j$ if and only if the $t$-th demand arrival comes from node $j \in [m]$. This sequence is random, so it is unknown to the e-tailer, who only gets to observe demand arrivals one by one as they arrive. Sequence $J$ follows a probability distribution called the *demand model*. We make the standard assumption that demand is independent of our inventory placement and fulfillment decisions. We define $D_j = \sum_{t=1}^T \mathbb{1}\{j_t = j\}$ to be the total number of demand arrivals from node $j$ during the time

horizon, and we further define $D = (D_j)_{j\in[m]}$ to be the vector containing these total demands. Our approach allows us to analyze and obtain results for a general family of random demand models. The only assumptions that we make are that the demand allows a fulfillment policy with a constant factor competitive ratio and that the total demand vector $D$ is exogenous to our placement and fulfillment decisions (this is further explained in Subsection 3.4.1). The e-tailer has an initial inventory $x \in \mathcal{X}$ decided in the first stage, and starts observing the demand sequence one by one. When the $t$-th demand request arrives, the e-tailer observes its origin $j_t$, and must irrevocably decide at most one warehouse $i \in [n]$ with remaining inventory from which to fulfill the request. Once the warehouse is chosen, the e-tailer depletes one unit of inventory, collects a reward $r_{ij_t}$, and observes the next arrival. We say that these online decisions are prescribed by a fulfillment policy $\pi$, and let $\Pi$ denote the class of all admissible fulfillment policies that make decisions without knowing the future. In its most general form, a policy $\pi$ decides from which warehouse to fulfill each demand arrival based on the node from where it arrives and the current inventory status together with the history of all previous demand arrivals and fulfillment decisions, and it is allowed to randomize. Reductions to the policy class can be made when further structure is imposed on the demand model.

An instance of the joint placement and fulfillment problem includes the total inventory to be placed $Q$, the sets of warehouses $[n]$ and demand nodes $[m]$, the rewards $(r_{ij})_{i,j\in[n]\times[m]}$, plus any demand parameters (concrete examples are provided in Subsection 3.4.1). We use $\mathsf{ONL}(x, \pi)$ to denote the expected reward collected by deploying an inventory placement $x$ and a fulfillment policy $\pi$, where the expectation is over the random demand and inner randomness of the inventory placement and fulfillment policy. We further define $\mathsf{OPT}(x) := \max_{\pi\in\Pi} \mathsf{ONL}(x, \pi)$ as the expected reward obtained by the best possible policy for a given instance and initial inventory. The overall goal of the e-tailer is to maximize the expected total reward collected during the time horizon by jointly deciding on an inventory placement and a fulfillment policy. To decide on an inventory placement, the e-tailer uses a *placement procedure* $P$ that takes in an instance and outputs a (potentially random) inventory placement $x^P \in \mathcal{X}$. To decide on a fulfillment policy, the e-tailer uses a

*fulfillment procedure* $F$ that takes in an instance along with an inventory placement $x$ and outputs a fulfillment policy $\pi^F \in \Pi$. Our goal is to derive a placement procedure $P$ along with a fulfillment procedure $F$ such that

$$\mathsf{ONL}(x^P, \pi^F) \geq \gamma \max_{x \in \mathcal{X}} \sup_{\pi \in \Pi} \mathsf{ONL}(x, \pi) - \varepsilon = \gamma \max_{x \in \mathcal{X}} \mathsf{OPT}(x) - \varepsilon$$

for all instances, with *approximation ratio* $\gamma \in [0, 1]$ being as large as possible, and sampling error $\varepsilon \geq 0$ (to be specified) being as small as possible.

### 3.3 Optimizing the Offline Surrogate

A solution to the joint placement and fulfillment problem consists of both a placement and fulfillment procedure. This section focuses on deriving the placement procedure, which optimizes the Offline surrogate.

The Offline surrogate represents the expected reward from Offline fulfillment—a hypothetical (non-implementable) algorithm that knows demand $D$ in advance and makes optimal hindsight fulfillment decisions. Given $D$ and initial inventory $x$, the reward from Offline fulfillment is $\mathsf{OFF}(x, D)$, the optimal value of a bipartite matching LP defined in Subsection 3.1.1. The fulfillment decisions correspond to the optimal LP variables $y_{ij}$, representing the number of units delivered from warehouse $i$ to demand node $j$. Since this value depends only on $D$ (not the arrival order), $\mathsf{OFF}(x, D)$ serves as an upper bound on any online algorithm's reward, as online decisions also lie in the matching LP polytope. Slightly abusing notation, we define the Offline surrogate as $\mathsf{OFF}(x) := \mathbb{E}_D[\mathsf{OFF}(x, D)]$, averaging over demand realizations. Both $\mathsf{OFF}(x, D)$ and $\mathsf{OFF}(x)$ extend to fractional placements $x$ in the convex hull $\mathrm{CH}(\mathcal{X}) := \{x \in \mathbb{R}^n_+ : \sum_{i \in [n]} x_i = Q\}$, allowing us to consider relaxations where $x$ is fractional.

Ideally, we would solve $\max_{x \in \mathcal{X}} \mathsf{OFF}(x)$ exactly, but two challenges arise: integrality constraints in $\mathcal{X}$ and an exponentially large sample space for $D$. To address integrality, we use randomized rounding, solving the relaxed problem $\max_{x \in \mathrm{CH}(\mathcal{X})} \mathsf{OFF}(x)$ and rounding to an integer while

preserving a constant factor. To handle the large sample space, we apply sample-average approximation (SAA), sampling a small number of demand realizations and solving the SAA problem. The placement procedure consists of (1) sampling demand, (2) solving the SAA linear relaxation, and (3) randomly rounding the solution.

The rest of this section is organized as follows. In Subsection 3.3.1, we describe how to round solutions of $\max_{x \in \mathrm{CH}(\mathcal{X})} \mathsf{OFF}(x)$ while preserving a constant factor of the original value. In Subsection 3.3.2, we apply statistical learning theory to show that an arbitrarily good solution can be obtained from our sample-average approximation using a polynomial number of samples.

### 3.3.1 Randomized Rounding

A key step in our inventory placement procedure is to round a (potentially fractional) solution $x \in \mathrm{CH}(\mathcal{X})$ without losing too much objective value when evaluating it in $\mathsf{OFF}$. To do so we apply the (star graph case of the) dependent rounding algorithm presented in Gandhi et al. (2006). This algorithm takes a finite set $A$ and a weight $w_i \in [0, 1]$ for every $i \in A$ as input. It outputs a vector of rounded random variables $W_i \in \{0, 1\}$ for all $i \in A$ that satisfies three properties:

- **Marginal Distribution**: $\mathbb{E}[W_i] = w_i$ for all $i \in A$.
- **Degree Preservation**: Let $\delta = \sum_{i \in A} w_i$. Then $\sum_{i \in A} W_i \in \{\lfloor \delta \rfloor, \lceil \delta \rceil\}$ with probability 1.
- **Negative Correlation**: For any subset $S \subseteq A$ and for any $b \in \{0, 1\}$:

$$\mathbb{P}\left(\bigcap_{i \in S} W_i = b\right) \leq \prod_{i \in S} \mathbb{P}(W_i = b).$$

We use this rounding algorithm as a subroutine, where we feed a solution $x \in \mathrm{CH}(\mathcal{X})$ and obtain a rounded solution $R(x) \in \mathcal{X}$. In our application, we will only round the fractional layer of the solution $x \in \mathrm{CH}(\mathcal{X})$, where we define the fractional part of $x_i$ as $x_i^f := x_i - \lfloor x_i \rfloor$. To round the solution, we input the set of warehouses, i.e. $A = [n]$, with weights $w_i = x_i^f$ for all $i$ to the rounding algorithm by Gandhi et al. (2006). Let $(W_i)_{i \in [n]}$ be the output of the rounding subroutine. We then

set $R_i(x) = \lfloor x_i \rfloor + W_i$ for all $i \in [n]$, where $R_i(x)$ is the $i$-th component of the rounded solution $R(x)$. In this application, the three properties of the randomized rounding algorithm by Gandhi et al. (2006) translate to:

**(P1) Marginal Distribution**: $\mathbb{E}[R_i(x)] = x_i$ for all $i \in [n]$,

**(P2) Degree Preservation**: $\sum_{i\in[n]} R_i(x) = Q$ and $R_i(x) \in \{\lfloor x_i \rfloor, \lfloor x_i \rfloor + 1\}$ for all $i \in [n]$ with probability 1,

**(P3) Negative Correlation**: For any subset $S \subseteq [n]$ and for any $b \in \{0, 1\}$:

$$\mathbb{P}\left(\bigcap_{i\in S} R_i(x) = \lfloor x_i \rfloor + b\right) \leq \prod_{i\in S} \mathbb{P}(R_i(x) = \lfloor x_i \rfloor + b).$$

We establish the following guarantee for this rounding procedure on our problem.

**Theorem 5** *Consider an instance such that the maximum degree across demand nodes is $d$. For any fractional inventory placement $x \in CH(\mathcal{X})$, and any rounding procedure $R$ that satisfies Properties* (P1)-(P3)*, it holds that*

$$\mathbb{E}_R[\mathsf{OFF}(R(x))] \geq \left(1 - \left(1 - \frac{1}{d}\right)^d\right) \mathsf{OFF}(x). \tag{3.4}$$

To prove this, the first thing to notice is that $\mathbb{E}_R[\mathsf{OFF}(R(x))] = \mathbb{E}_{D,R}[\mathsf{OFF}(R(x), D)]$ (because $D$ is independent from $R$) and $\mathsf{OFF}(x) = \mathbb{E}_D[\mathsf{OFF}(x, D)]$. Therefore, to prove (3.4), it suffices to show

$$\mathbb{E}_R[\mathsf{OFF}(R(x), D)] \geq \left(1 - \left(1 - \frac{1}{d}\right)^d\right) \mathsf{OFF}(x, D) \tag{3.5}$$

for any fixed realization of $D$. Given $D$, we will randomly construct a feasible solution to the LP defining $\mathsf{OFF}(R(x), D)$ that satisfies (3.5) in expectation over the randomly rounded inventory and the random construction of the solution. For the analysis, we will split the demand of each node $j$ into $D_j$ identical nodes with one unit of demand each. After doing so, we end up with the same

problem except that now $D_j = 1$ for all $j \in [T]$ under the new indexing. (Note that we cannot split inventory units at warehouses in the same way, because that would change the parameter $d$ in our guarantee.) Hereafter, we fix a demand realization $D$, split $D$ into unit demand nodes, and let $(y_{ij})_{(i,j)\in[n]\times[T]}$ refer to an optimal solution to the LP defining $\mathsf{OFF}(x, D)$ (after the splitting of nodes so that $y_{ij} \in [0, 1]$ for all $i, j$).

We use $(Z_{ij})_{(i,j)\in[n]\times[T]}$ to denote our proposed feasible solution for $\mathsf{OFF}(R(x), D)$. Its construction can be summarized in the following three steps.

1. Round the feasible solution $x$ and obtain $R(x)$.

2. For each warehouse $i \in [n]$, preliminarily assign its $R_i(x)$ inventory units across demand nodes $j \in [T]$. Let $Y_{ij}$ be an indicator for whether a unit from $i$ is preliminarily assigned to $j$. This step applies another iteration of the rounding algorithm from Gandhi et al. (2006), with $Y_{ij}$ depending on whether $x_i$ was rounded up or down.

3. For each demand node $j \in [T]$, finalize assignments to construct $(Z_{ij})_{(i,j)\in[n]\times[T]}$. Demand node $j$ selects the preliminarily assigned unit yielding the highest reward, setting $Z_{ij} = 1$ if and only if $i = \arg\max_{i:Y_{ij}=1} r_{ij}$, with arbitrary tie-breaking.

Step 1 is independent of $D$, while Steps 2 and 3 depend on $D$ and $R(x)$. In Step 3, each demand node selects at most one assigned unit, wasting the rest. This explains why the guarantee worsens as $d$ increases—larger $d$ means demand nodes can be served by more warehouses, increasing inventory waste.

We now elaborate on Step 2: defining a preliminary assignment of the $R_i(x)$ inventory units of each warehouse $i$. Conditional on $R(x)$, this will be done independently for every $i \in [n]$, and to do this we again resort to the dependent rounding algorithm by Gandhi et al. (2006). For each $i \in [n]$, we feed the rounding subroutine the set $[T]$ with weights that depend on whether $W_i = 1$ or $W_i = 0$. We denote these weights by $(y^{\mathrm{H}}_{ij})_{j\in[T]}$ (for high inventory, $W_i = 1$) and $(y^{\mathrm{L}}_{ij})_{j\in[T]}$ (for low inventory, $W_i = 0$). For the analysis to hold, we require that for all $i \in [n]$, these weights

(probabilities) satisfy the following constraints:

$$y_{ij}^{\mathrm{H}} x_i^f + y_{ij}^{\mathrm{L}}(1 - x_i^f) = y_{ij} \qquad \forall j \in [T], \tag{3.6}$$

$$0 \le y_{ij}^{\mathrm{L}} \le y_{ij}^{\mathrm{H}} \le 1 \qquad \forall j \in [T], \tag{3.7}$$

$$\sum_{j \in [T]} y_{ij}^{\mathrm{L}} \le \lfloor x_i \rfloor, \tag{3.8}$$

$$\sum_{j \in [T]} y_{ij}^{\mathrm{H}} \le \lfloor x_i \rfloor + 1. \tag{3.9}$$

Equation (3.6) will make $\mathbb{E}[Y_{ij}] = y_{ij}$ for all $j \in [T]$. Equation (3.7) is a logical monotonicity constraint that allows to transfer the negative correlation property of the rounding $(W_i)_{i \in [n]}$ to the preliminary assignment $(Y_{ij})_{i \in [n]}$, for every $j \in [T]$. Equations (3.8) and (3.9) will make it so that we can always apply the procedure of Gandhi et al. (2006) without assigning more inventory units than what we have in hand, regardless of whether $R_i(x) = \lfloor x_i \rfloor$ or $R_i(x) = \lfloor x_i \rfloor + 1$. We note that naive ways of setting these probabilities such as scaling, sampling, or splitting fail:

1. **Scaling**: Setting $y_{ij}^{\mathrm{H}} = y_{ij}(\lceil x_i \rceil / x_i), y_{ij}^{\mathrm{L}} = y_{ij}(\lfloor x_i \rfloor / x_i)$ for each $j$ satisfies all the conditions except that it could create values of $y_{ij}^{\mathrm{H}}$ that are greater than 1 and hence invalid probabilities.

2. **Proportional Sampling**: Another plausible strategy is to sample $R_i(x)$ times without replacement from $[T]$, with probabilities proportional to $y_{i1}, \ldots, y_{iT}$. However, this does not preserve marginals—take for example $y_{i1} = 0.9, y_{i2} = 0.6, x_i = 1.5$. This procedure would induce $y_{i1}^{\mathrm{H}} = 1$ and $y_{i1}^{\mathrm{L}} = 3/5$, but does not satisfy $0.5 y_{i1}^{\mathrm{H}} + 0.5 y_{i1}^{\mathrm{L}} = y_{i1}$.

3. **Further Splitting**: Naive methods would work if we could split warehouses so that $x_i \le 1$ for all $i$; however, this would change the value of $d$ that we are parametrizing by.

Given the failure of these naive methods, we instead show fundamentally that a feasible adjustment $(y_{ij}^{\mathrm{L}}, y_{ij}^{\mathrm{H}})_{j \in [T]}$ exists.

**Lemma 15** *For all $i \in [n]$, there exist vectors of probabilities $(y_{ij}^{\mathrm{H}})_{j \in [T]}$ and $(y_{ij}^{\mathrm{L}})_{j \in [T]}$ that satisfy Equations* (3.6) *to* (3.9)*, assuming that $\sum_{j \in [T]} y_{ij} \le x_i$ and $y_{ij} \in [0, 1]$ for each $j \in [T]$.*

The proof of this lemma is deferred to Appendix B.1.1.

We proceed by lower-bounding the reward that can be obtained from each demand node by our proposed solution $(Z_{ij})_{(i,j)\in[n]\times[T]}$, in expectation over the randomized placement and assignment, in a way that allows us to relate it to the objective value of $\mathsf{OFF}(x, D)$.

**Lemma 16** *Fix a realization of the demand vector $D$. It holds for all demand nodes $j \in [T]$ that*

$$\mathbb{E}\left[\sum_{i\in[n]} r_{ij} Z_{ij}\right] \geq \left(1 - \left(1 - \frac{1}{d_j}\right)^{d_j}\right) \sum_{i\in[n]} r_{ij} y_{ij},$$

*where the expectation is taken over the randomized rounding $R(x)$ and random assignments determined by $Y_{ij}$ and $Z_{ij}$.*

We prove Lemma 16 in Appendix B.1.2, noting that a similar proof has appeared in Brubach et al. (2021). With these results, we can complete the proof of Theorem 5, presented in Appendix B.1.3. Importantly, we note that the approximation from Theorem 5 is tight.

**Lemma 17** *There exists a family of instances for which*

$$\frac{\max_{x\in\mathcal{X}} \mathsf{OFF}(x)}{\max_{x\in\mathrm{CH}(\mathcal{X})} \mathsf{OFF}(x)} \leq 1 - \left(1 - \frac{1}{d}\right)^{d}.$$

The proof of this lemma is deferred to Appendix B.1.4.

We close this subsection by noting another approach to approximating $\max_{x\in\mathcal{X}} \mathsf{OFF}(x)$: submodular optimization. However, this fails to yield better approximations for small $d$. Bai, Rusmevichientong, and Topaloglu (2022) show that $\mathsf{OFF}(x, D)$ is submodular over $x \in \mathcal{X}$ for fixed $D$. Since submodularity is preserved under convex combinations, $\mathsf{OFF}(x)$ is also submodular over $x \in \mathcal{X}$. Moreover, $\mathsf{OFF}(x)$ is non-decreasing in $x$, allowing a $(1 - 1/e)$-approximation of $\max_{x\in\mathcal{X}} \mathsf{OFF}(x)$ via greedy inventory placement (Nemhauser, Wolsey, and Fisher 1978). More precisely, the submodular approach gives an approximation of $(1 - (1 - 1/Q)^{Q})$, where $Q$ is the inventory units placed. In applications, $Q$ is large, making this close to $1 - 1/e$. This bound cannot improve even if $d$ is small, as shown in Appendix B.2. Another advantage of rounding over

submodular optimization is that the $1 - (1 - 1/d)^d$ randomized rounding guarantee extends to a general multi-SKU setting (Section 3.5), whereas submodular techniques only ensure $1/2$.

### 3.3.2 Sample Average Approximation

The rounding results in Subsection 3.3.1 are contingent on being able to solve the problem $\max_{x \in \mathrm{CH}(\mathcal{X})} \mathsf{OFF}(x)$. However, since $\mathsf{OFF}(x)$ takes an expectation over $D$ and the number of possible realizations of $D$ grows exponentially under a general demand model, the problem cannot be solved exactly. Instead, we solve a sample-average approximation of the problem $\max_{x \in \mathrm{CH}(\mathcal{X})} \mathsf{OFF}(x)$, and prove that the sample-average approximation is not too sensitive to the samples drawn.

To elaborate, we independently sample $K$ vectors $D^1, \ldots, D^K$ for the total demand according to the given demand model. We define

$$\widehat{\mathsf{OFF}}(x) = \frac{1}{K} \sum_{k=1}^{K} \mathsf{OFF}(x, D^k),$$

and solve for $\hat{x}$, an optimal solution to the problem $\max_{x \in \mathrm{CH}(\mathcal{X})} \widehat{\mathsf{OFF}}(x)$. In general, we will use the symbol $\wedge$ to refer to anything that depends on the samples $D^1, \ldots, D^K$, and we will use $\mathbb{E}_\wedge$ when we are taking an expectation over the random samples. It will also be understood that the sample consists of exactly $K$ demand realizations.

We can solve $\max_{x \in \mathrm{CH}(\mathcal{X})} \widehat{\mathsf{OFF}}(x)$ by solving the following linear program:

$$\begin{aligned}
\max_{x \in \mathbb{R}^n_+, y \in \mathbb{R}^{[n]\times[m]\times[K]}_+} \quad & \frac{1}{K} \sum_{k=1}^{K} \sum_{i \in [n]} \sum_{j \in [m]} r_{ij} y^k_{ij} & \\
\text{s.t.} \quad & \sum_{i=1}^{n} y^k_{ij} \le D^k_j & \forall j \in [m], k \in [K], \\
& \sum_{i=j}^{m} y^k_{ij} \le x_i & \forall i \in [n], k \in [K], \\
& \sum_{i=1}^{n} x_i = Q. &
\end{aligned}$$

Notice that this problem has a polynomial number of variables and constraints if $K$ is polynomial in the instance parameters. By contrast, the problem $\max_{x \in \mathrm{CH}(\mathcal{X})} \mathsf{OFF}(x)$ would have required an exponentially-sized $K$ to capture the support of $D$.

The main result of this subsection, which will allow us to use the sample-average solution $\hat{x}$ to obtain our approximation result, is the following.

**Theorem 6** *For a random sample of K IID demand realizations, we have*

$$\mathbb{E}_{\wedge} \left[ \sup_{x \in \mathrm{CH}(\mathcal{X})} \widehat{\mathsf{OFF}}(x) - \mathsf{OFF}(x) \right] = O\left( Q \sqrt{\frac{n \log n}{K}} \right).$$

Theorem 6 provides a uniform convergence guarantee on the generalization error of how much $\widehat{\mathsf{OFF}}$ can overestimate the value of any solution $x$ and its proof uses tools from statistical learning theory. We note that the space of fractional solutions $\mathrm{CH}(\mathcal{X})$ is continuous and infinite; however, we can apply a vector contraction inequality after showing the function $\mathsf{OFF}(x)$ to be Lipschitz in $x$. The proof of Theorem 6 is presented in Appendix B.1.5.

## 3.4 Approximating the Joint Problem

In this section, we build on the results from Section 3.3 to derive a placement procedure and fulfillment procedure that jointly achieve an $\alpha(1-(1-1/d)^d)$-approximation of the optimal joint solution. This can only be possible if the demand model admits a fulfillment approximation guarantee in the first place, as we describe in Subsection 3.4.1. In Subsection 3.4.2, we present the joint result, provide a sketch of its proof, and provide some remarks and discussion.

### 3.4.1 Assumptions on Demand Model

As mentioned in Section 3.2, our analysis applies to a broad class of demand models, defined as a distribution over sequences of demand nodes (of random length). We assume the following.

**Assumption 1** *The demand model is such that:*

*(a) for any instance, and any starting inventory $x \in \mathcal{X}$, we can efficiently compute a fulfillment policy $\pi^\alpha$ such that $\mathsf{ONL}(x, \pi^\alpha) \geq \alpha\mathsf{OFF}(x)$,*

*(b) the total demand vector $D$ does not depend on the placement and fulfillment policy, and*

*(c) a sample of the total demand vector $D$ can be drawn in polynomial time.*

Assumption (a) is necessary for our analysis (though it could be relaxed for placements generated by our procedure). Assumption (b) is standard; independence applies only to total demand, allowing adversarial arrival orders. Assumption (c) is required as our placement procedure relies on sampling $D$.

We mention two demand models in the literature that satisfy these assumptions as examples.

**Temporal Independence model.** A classic model in revenue management and online matching (Alaei, Hajiaghayi, and Liaghat 2012), where the time horizon $T$ is deterministic and known, and each demand node $j$ arrives independently at time $t$ with probability $p_{tj}$. Some time steps may have no arrivals, represented by a dummy demand node with zero reward. Here, $\sum_{j\in[m]} p_{tj} = 1$

for all $t \in [T]$. Assumptions (a), (b), and (c) hold, with $\alpha = 1/2$ (Alaei, Hajiaghayi, and Liaghat 2012), as $D$ follows a multinomial distribution with $T$ trials.

**Spatial Independence model.** Recently introduced by Aouad and Ma (2022), this model samples each node's total demand $D_j$ from independent distributions $(G_j)_{j\in[m]}$. An adversary then determines the arrival order to minimize the e-tailer's reward. Here, $T = \sum_{j=1}^{m} D_j$ is random and unknown. Assumptions (a), (b), and (c) hold, with $\alpha = 1/2$ (Aouad and Ma 2022), provided each $G_j$ can be sampled efficiently.

### 3.4.2 Approximation for the Join Fulfillment and Placement Problem

We now present the joint placement and fulfillment solution. The placement procedure, $R(\hat{x})$, follows three steps:

1. Sample $K$ IID demand realizations $D_1, \ldots, D_K$.
2. Solve for $\hat{x}$, the optimal solution of $\max_{x\in \mathrm{CH}(\mathcal{X})} \widehat{\mathsf{OFF}}(x)$.
3. Randomly round $\hat{x}$ using $R$ to obtain $R(\hat{x})$.

The fulfillment policy used is $\pi^{\alpha}$, which exists by Assumption 1 and is $\alpha$-competitive. We obtain the following guarantee:

**Theorem 7** *For any instance where at most d warehouses serve a single demand node, we efficiently obtain an inventory placement $R(\hat{x})$ and a fulfillment policy $\pi^{\alpha}$ satisfying*

$$\mathbb{E}_{\wedge,R}\left[\mathsf{ONL}(R(\hat{x}), \pi^{\alpha})\right] \geq \alpha\left(1 - \left(1 - \frac{1}{d}\right)^{d}\right) \max_{x\in\mathcal{X}} \mathsf{OFF}(x) - \varepsilon,$$

*using $K = O\left(\frac{Q^2 n \log n}{\varepsilon^2}\right)$ demand samples.*

Since $\mathsf{OFF}(x) \geq \mathsf{OPT}(x)$ for all $x \in \mathcal{X}$, this provides an $\alpha(1 - (1 - 1/d)^d)$-approximation for the joint problem. The proof (deferred to Appendix B.1.6) builds on the three-step approximation framework of Bai, Rusmevichientong, and Topaloglu (2022) and Bai et al. (2022), which involves:

1) defining a surrogate function $f$ such that $f(x) \geq \mathsf{OPT}(x)$ for all $x \in \mathcal{X}$, 2) finding an inventory placement $x'$ with $f(x') \geq \beta \max_{x \in \mathcal{X}} f(x)$, and 3) designing a fulfillment policy $\pi$ such that $\mathsf{ONL}(\pi, x') \geq \alpha f(x')$.This yields an $\alpha\beta$ approximation:

$$\mathsf{ONL}(\pi, x') \geq \alpha f(x') \geq \alpha\beta \max_{x \in \mathcal{X}} f(x) \geq \alpha\beta \max_{x \in \mathcal{X}} \mathsf{OPT}(x). \tag{3.10}$$

In our case, the surrogate function is $\mathsf{OFF}$. For step 2, we apply randomized rounding, achieving the $(1 - (1 - 1/d)^d)$-approximation from Theorem 5. To handle the large demand space, we use sample-average approximation. Since this prevents a direct application of Equation (3.10), we incorporate learning theory techniques. Finally, step 3 follows from the assumed existence of the $\alpha$-competitive fulfillment policy.

We would like to remark on the advantage of using $\mathsf{OFF}(x)$ as the surrogate function instead of using a fluid approximation, approach used by Bai, Rusmevichientong, and Topaloglu (2022). Let $\mathsf{FLU}(x)$ be the value of this fluid approximation with inventory placement $x$, defined as

$$\begin{aligned} \mathsf{FLU}(x) := \max_{y \geq 0} \quad & \sum_{i \in [n]} \sum_{j \in [m]} y_{ij} r_{ij} \\ \text{s.t.} \quad & \sum_{i \in [n]} y_{ij} \leq \mathbb{E}[D_j] \quad \forall j \in [m], \\ & \sum_{j \in [m]} y_{ij} \leq x_i \quad \forall i \in [n]. \end{aligned}$$

That is, $\mathsf{FLU}(x)$ is equal to $\mathsf{OFF}(x, \mathbb{E}[D])$. Since $\mathsf{OFF}(x, D)$ is concave in $D$, Jensen's inequality implies that $\mathsf{FLU}(x) \geq \mathsf{OFF}(x)$ for all $x \in \mathrm{CH}(\mathcal{X})$. Hence, to carry out our same analysis we would require that $\pi^\alpha$ satisfies the stronger assumption

$$\mathsf{ONL}(x, \pi^\alpha) \geq \alpha \mathsf{FLU}(x) \quad \forall x \in \mathcal{X}. \tag{3.11}$$

In fact, if we consider demand models that additionally satisfy this assumption, we obtain a strictly

smaller subset than we would get only imposing Assumption 1. For instance, the Spatial Independence model described in Subsection 3.4.1 does not allow any policy that satisfies Equation (3.11) with $\alpha$ bounded away from 0, but it does admit a policy that satisfies Assumption 1 with $\alpha = 1/2$ (Aouad and Ma 2022). Thus, our approach does not only obtain a strictly better competitive ratio, but it also works for a strictly larger set of demand models.

We highlight the practical aspects of our placement procedure in Theorem 7. First, it is entirely sample-driven, allowing past demand data to be used without fitting an exact distribution. Second, it requires only aggregate demand samples over time, independent of arrival order. Third, it is easily obtained by solving a linear program, with rounding needed only for rare fractional solutions. When the optimal solution is integer, our guarantee improves to $\alpha$.

All in all, Theorem 7 formalizes that offline fulfillment yields strong outcomes when paired with an effective fulfillment policy. Our placement procedure optimizes as if the fulfillment team knew the future exactly. The result shows this approach achieves state-of-the-art approximation guarantees when fulfillment is highly competitive relative to offline. This aligns with intuition—a strong fulfillment team allows the placement team to be optimistic about effective inventory distribution.

## 3.5 Extension of Randomized Rounding to Multiple SKUs Setting

We now change our focus to the multi-SKU setting of Bai, Rusmevichientong, and Topaloglu (2022). We show how to apply our result for the single-SKU setting in this case without loss in performance, and therefore obtain state-of-the-art guarantees for approximating the inventory placement problem under the Offline surrogate in the presence of multiple SKUs.

### 3.5.1 Problem Setting

In the multi-SKU setting, we have $s$ different SKUs, indexed by $\ell \in [s]$. Each warehouse $i$ is equipped with an initial inventory of $x_{i\ell}$ units of SKU $\ell$. To avoid introducing additional notation, we will again use $x$ to denote this (now) $n \times s$ inventory matrix, and use $x^\ell := (x_{i\ell})_{i\in[n]}$

to denote its $\ell$-th column (associated to SKU $\ell$). Random demand is now represented by an $m \times s$ matrix $D$, where $D_{j\ell}$ are the units of SKU $\ell$ demand originating from demand node $j$. Let $D^\ell := (D_{j\ell})_{j\in[m]}$ represent the $\ell$-th column of the demand matrix. There is a reward of $r_{ij}^\ell$ for fulfilling a unit of demand for SKU $\ell$ originating from demand node $j$ using inventory in warehouse $i$. (In this problem, we do not account for potential reward gains from package consolidation. This is consistent with Bai, Rusmevichientong, and Topaloglu (2022).) Given an initial inventory matrix $x$ and random demand realization matrix $D$, the value of offline fulfillment is given by the following linear program

$$\begin{aligned}
\mathsf{OFF}_{\mathsf{multi}}(x, D) = \max_{y\geq 0} \quad & \sum_{\ell\in[s]} \sum_{i\in[n]} \sum_{j\in[m]} y_{ij}^\ell r_{ij}^\ell \\
\text{s.t.} \quad & \sum_{i\in[n]} y_{ij}^\ell \leq D_{j\ell} \quad \forall j \in [m],\, \forall \ell \in [s], && (3.12)\\
& \sum_{j\in[m]} y_{ij}^\ell \leq x_{i\ell} \quad \forall i \in [n],\, \forall \ell \in [s]. && (3.13)
\end{aligned}$$

Here, Constraint (3.12) says that for each SKU and each demand node, we cannot fulfill more than the demand originating from said node. Constraint (3.13) says that we cannot use more than the available inventory for any warehouse and SKU pair. As in the single-SKU setting, the value of the Offline surrogate corresponds to the expected value of the Offline fulfillment LP, taken over the random demand distribution: $\mathsf{OFF}_{\mathsf{multi}}(x) = \mathbb{E}_D[\mathsf{OFF}_{\mathsf{multi}}(x, D)]$. Note that the optimization problem defining $\mathsf{OFF}_{\mathsf{multi}}(x, D)$ can be decoupled across SKUs $\ell$.

We are interested in finding an optimal inventory placement for the Offline surrogate in the multi-SKU setting. Here, for each SKU $\ell$, we let $Q_\ell$ denote the number of inventory units to be placed. Each warehouse $i$ has a capacity $C_i$ that is crucially shared across SKUs, coupling the inventory placement decisions across $\ell$. Again, extending the notation from previous sections, we

define the set of feasible inventory placements as

$$\mathcal{X}_{\mathsf{multi}} = \{x \in \mathbb{N}^{n\times s} : \sum_{i\in[n]} x_{i\ell} = Q_\ell \quad \forall \ell \in [s],$$
$$\sum_{\ell\in[s]} x_{i\ell} \le C_i \quad \forall i \in [n]\}.$$

We also define the convex hull of the feasible placements set $\mathrm{CH}(\mathcal{X}_{\mathsf{multi}})$ as $\mathcal{X}_{\mathsf{multi}}$ after relaxing the integrality constraints. Finding an optimal inventory placement for the Offline surrogate formally corresponds to solving $\max_{x\in\mathcal{X}_{\mathsf{multi}}} \mathsf{OFF}_{\mathsf{multi}}(x)$. This is a hard combinatorial problem, and therefore we resort to obtaining good approximations.

### 3.5.2 Rounding a Fraction Solution

We proceed by showing how to extend our randomized rounding analysis from Subsection 3.3.1 to obtain state-of-the-art approximation guarantees for the Offline surrogate in the multi-SKU setting.

As we did in the single-SKU setting, we solve the linear relaxation $\max_{x\in\mathrm{CH}(\mathcal{X}_{\mathsf{multi}})} \mathsf{OFF}_{\mathsf{multi}}(x)$ and round the optimal solution obtained. To round the solution, we now apply the general bipartite graph case of the dependent rounding algorithm presented in Gandhi et al. (2006). This algorithm takes a bipartite graph $G = (A, B, E)$ and weights $w_{i\ell} \in [0, 1]$ for every edge $(i, \ell) \in E$ as input. It outputs a vector of random variables $W_{i\ell} \in \{0, 1\}$ for all $(i, \ell) \in E$ that satisfies three properties:

- **Marginal Distribution**: $\mathbb{E}[W_{i\ell}] = w_{i\ell}$ for every edge $(i, \ell) \in E$.
- **Degree Preservation**: For any node $i \in A \cup B$ define $\Delta(i)$ the set of nodes adjacent to it, and let $\delta_i = \sum_{\ell\in\Delta(i)} w_{i\ell}$ be its fractional degree. Then, for any node $i \in A \cup B$ we have $\sum_{\ell\in\Delta(i)} W_{i\ell} \in \{\lfloor\delta_i\rfloor, \lceil\delta_i\rceil\}$ with probability 1.
- **Negative Correlation**: For any node $i \in A \cup B$, any subset $S \subseteq \Delta(i)$ of nodes adjacent to $i$, and for any $b \in \{0, 1\}$:
$$\mathbb{P}\left(\bigcap_{\ell\in S} W_{i\ell} = b\right) \le \prod_{\ell\in S} \mathbb{P}(W_{i\ell} = b).$$

We use this rounding algorithm as a subroutine, where we feed a solution $x \in \mathrm{CH}(\mathcal{X}_{\mathsf{multi}})$ and obtain a rounded solution $R(x) \in \mathcal{X}_{\mathsf{multi}}$. In our application, we will only round the fractional layer of the solution $x \in \mathrm{CH}(\mathcal{X}_{\mathsf{multi}})$, where we define the fractional part of $x_{i\ell}$ as $x^f_{i\ell} := x_{i\ell} - \lfloor x_{i\ell} \rfloor$. To round the solution, we give the following input to the rounding algorithm by Gandhi et al. (2006). The bipartite graph will have the set of warehouses on one side and the set of SKUs on the other side, and it will be complete. That is, $G = ([n], [s], [n] \times [s])$. For the weights, we use the fractional layer of our linear relaxation: $w_{i\ell} = x^f_{i\ell}$ for all $(i, \ell) \in [n] \times [s]$. Let $(W_{i\ell})_{(i,\ell)\in[n]\times[s]}$ be the output of the rounding subroutine. We then set $R_{i\ell}(x) = \lfloor x_{i\ell} \rfloor + W_{i\ell}$ for all $(i, \ell) \in [n] \times [s]$, where $R_{i\ell}(x)$ is the $(i, \ell)$-th entry of the rounded solution matrix $R(x)$. In this application, the marginal distribution property translates to $\mathbb{E}[R_{i\ell}(x)] = x_{i\ell}$ for all $(i, \ell) \in [n] \times [s]$. The degree preservation property implies that $\sum_{i\in[n]} R_{i\ell}(x) = Q_\ell$ and $\sum_{\ell\in[s]} R_{i\ell}(x) \leq C_i$ with probability 1.

The main result of this section is that the rounding guarantee of Theorem 5 holds in the multi-SKU setting.

**Corollary 5** *For any instance with multiple SKUs such that the maximum degree across demand nodes is $d$ and for any fractional inventory placement $x \in CH(\mathcal{X}_{\mathsf{multi}})$, it holds that*

$$\mathbb{E}_R[\mathsf{OFF}_{\mathsf{multi}}(R(x))] \geq \left(1 - \left(1 - \frac{1}{d}\right)^d\right) \mathsf{OFF}_{\mathsf{multi}}(x). \tag{3.14}$$

The key observation to prove this corollary is that $\mathsf{OFF}_{\mathsf{multi}}(x)$ can be decomposed into $s$ sub-problems, one for each SKU. Indeed, we have

$$\begin{aligned}\mathsf{OFF}_{\mathsf{multi}}(x) = E_D[\mathsf{OFF}_{\mathsf{multi}}(x, D)] &= \sum_{\ell\in[s]} \mathbb{E}_D[\mathsf{OFF}(x^\ell, D^\ell)] \\ &= \sum_{\ell\in[s]} \mathbb{E}_{D^\ell}[\mathsf{OFF}(x^\ell, D^\ell)] = \sum_{\ell\in[s]} \mathsf{OFF}_\ell(x^\ell),\end{aligned}$$

where we implicitly define $\mathsf{OFF}_\ell(x^\ell) := E_{D^\ell}[\mathsf{OFF}(x^\ell, D^\ell)]$. Therefore we have

$$\mathbb{E}_R[\mathsf{OFF}_{\mathsf{multi}}(R(x))] = \sum_{\ell\in[s]} \mathbb{E}_R[\mathsf{OFF}_\ell(R^\ell(x))],$$

where $R^\ell(x)$ is the $\ell$-th column of the rounded inventory matrix. In order to obtain the corollary, it suffices to show that

$$\mathbb{E}_R[\mathsf{OFF}_\ell(R^\ell(x))] \geq (1 - (1 - 1/d))\,\mathsf{OFF}_\ell(x^\ell) \tag{3.15}$$

for any $\ell \in [s]$. In order to do that, we simply notice that $R^\ell(x)$ satisfies Properties (P1)-(P3), the hypotheses of Theorem 5. Therefore, we can invoke this theorem and establish (3.15), which ultimately implies Corollary 5.

The approximation guarantee in Corollary 5 is at least $1 - 1/e$, which improves upon the state-of-the-art guarantee of $1/2$ obtained by Bai, Rusmevichientong, and Topaloglu (2022). In this paper, the authors obtain a $1/(2+\varepsilon)$ approximation of the Offline surrogate by showing that $\mathsf{OFF}_{\mathsf{multi}}(x)$ is monotone and submodular in $x$. Then, $\max_{x\in\mathcal{X}} \mathsf{OFF}_{\mathsf{multi}}(x)$ corresponds to maximizing a monotone, submodular function subject to two partition constraints, for which Lee et al. (2010) offers a $1/(2+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$.

### 3.6 Experiments with Synthetic Data

Generally speaking, an instance of the e-commerce placement and fulfillment problem consists of a quantity $Q$ to be placed across warehouses, a network defined by rewards $r_{ij}$ between each warehouse $i$ and demand location $j$, and a distribution over arrival sequences from different demand locations. In Section 3.3 we analyzed worst-case instances for network structure and the distribution over arrival sequences. In this section we explore how our proposed methods perform when deployed on network structures and demand models that are common in the literature. Computationally, we corroborate the intuition behind Theorem 7: choosing an inventory placement by optimizing the Offline surrogate leads to a good overall performance if a high-quality fulfillment policy is deployed downstream. Indeed, Offline Placement consistently performs among the highest rewards when paired with such a fulfillment policy.

In order to answer this question, we evaluate the performance of four different inventory place-

ment procedures together with five fulfillment policies on a range of 90 instances. Consistent with DeValve et al. (2023), we find that a policy based on the shadow prices of the Offline surrogate for quantifying opportunity costs, and that periodically recomputes these shadow prices based on current inventories and observed demands, consistently outperforms all of the benchmarks. We compare the reward collected by this high-quality fulfillment policy when using the four different inventory placements and establish that the Offline Placement has the most robust performance.

The rest of this section is structured as follows. In Subsection 3.6.1, we describe the different placement and fulfillment procedures we evaluate, and in Subsection 3.6.2, we describe all the instances we use to evaluate them. In Subsection 3.6.3, we outline the steps we follow to conduct our experiments. In Subsection 3.6.4 and Subsection 3.6.5, we report the obtained results for comparing fulfillment policies and placement procedures, respectively.

### 3.6.1 Placement Procedures and Fulfillment Policies

We evaluate four inventory placements by optimizing different surrogates, and couple them with five different fulfillment procedures. For a detailed description of these procedures, we refer the reader to Appendix B.4.2 and Appendix B.4.3.

**Offline Placement.** This placement procedure optimizes the Offline surrogate by solving its LP relaxation and rounding the solution. (We choose this way of solving the surrogate based on the findings in Appendix B.4.1, where the LP rounding solution has the same performance as the greedy solution, but it is orders of magnitude faster to obtain.) Since the support of demand realizations is exponentially sized for all demand models, we solve the sample-average approximation (SAA) instead. Remarkably, all the obtained solutions were integral, so no rounding was required.

**Myopic Placement.** This placement procedure optimizes the Myopic surrogate: the expected reward collected by a myopic fulfillment policy that chooses to deliver from the warehouse with remaining inventory that provides the highest immediate reward. This is optimized through simulation, using a greedy algorithm, and with SAA.

**Fluid Placement.** This placement procedure optimizes the Fluid surrogate, attempting to solve

$\max_{x \in \mathcal{X}} \mathsf{FLU}(x)$. This is done by solving the linear relaxation and greedily rounding the obtained solution. Since we know the expected values for the aggregate demands, this procedure does not require sampling.

**Scaled Demand Fluid Placement.** This placement procedure optimizes a modified Fluid surrogate. In this variant, we scale the expected demand from each demand node so that the total expected demand is equal to the available inventory $Q$. This is intended as a fix to Fluid Placement when $Q$ exceeds the expected demand, leading to degenerate solutions.

**Fulfillment Policies.** We evaluate five different fulfillment policies. The first one is a simple *myopic policy* that maximizes the immediate collected reward, without accounting for future demand. We also study two different fulfillment policies that account for the opportunity costs of depleting inventory units in certain warehouses by penalizing the collected rewards. We use the shadow prices of the inventory constraints of two different LPs as these penalties. Specifically, we use the Fluid LP and the Offline LP, leading to the *fluid shadow price policy* (F-SP) and *offline shadow price policy* (O-SP), respectively. The three policies described above define, for each demand node, a preference order of warehouses from which to deliver. This order is fixed across the time horizon, which could be suboptimal given particular realizations of demand samples. To improve this, we also evaluate the re-solving variants of these shadow price policies, where the shadow prices are recomputed taking into account the observed demand (because we consider non-independent demand models, Bayesian updating is needed) and current inventories. We call refer to these as F-SP-R and O-SP-R, for the Fluid and Offline LPs, respectively. To decide when to re-solve, we simulate arrival times associated to arrivals (explained in Subsection 3.6.2). In our experiments, we include two re-solving epochs: one after we observe the last arrival with an arrival time of $1/3$, and a second one after we observe the last arrival with an arrival time of $2/3$.

### 3.6.2 Description of Instances

Below is a description of the different instances we use in our experiments. A total of 90 instances arise from combinations of 3 network structures, 6 demand models, and 5 values for the

total inventory $Q$.

**Description of Networks.**

Recall that a fulfillment network is described by a set of warehouses, a set of demand nodes, and the rewards between them.

**Long-chain Network.** This follows the structure introduced by Jordan and Graves (1995), with $n = m = 5$ warehouses and demand nodes. Warehouse $i \in \{1, \ldots, 4\}$ connects to demand nodes $j = i$ and $j = i + 1$, while warehouse $i = 5$ connects to $j = 5$ and $j = 1$. Rewards for these edges are sampled from Uniform$(0, 1)$, with all other pairs set to 0. This low-flexibility network allows each demand node to be served by at most two warehouses ($d = 2$).

**RDC-FDC Network.** Inspired by DeValve et al. (2023), this network models JD.com's operations with $n = m = 5$ warehouses and demand nodes. Warehouse $i = 1$ is the Regional Distribution Center (RDC), while $i = 2, \ldots, 5$ are Front Distribution Centers (FDCs). Each FDC serves only its local demand node ($j = i$) with a reward of 1. The RDC can fulfill any demand: locally ($j = 1$) with a reward of 1, and to $j = 2, \ldots, 5$ with rewards sampled from Uniform$(0, 1)$. This low-flexibility network allows each demand node at most two warehouses ($d = 2$).

**Complete Network.** This graph consists of $n = 5$ warehouses and $m = 15$ demand nodes, randomly placed in $[0, 1]^2$ and $[0.2, 0.8]^2$, respectively. Every demand node is reachable from any warehouse, with rewards based on their Euclidean distance: $r_{ij} = 1 - d_{ij}/\sqrt{2}$ for all $(i, j) \in [n] \times [m]$.

**Description of Demand Models.**

A demand model specifies a distribution over demand arrival sequences. We consider three models: the classic temporal independence model, its extension with a random time horizon, and a spatial independence variant with a random arrival order. Each demand node $j$ has a weight $p_j \in [0, 1]$ representing its average arrival proportion, with $\sum_{j \in [m]} p_j = 1$. We define two weight-setting methods, yielding six stochastic arrival models.

**Deterministic-Horizon Temporal Independence (DH-TI).** Here, demand always consists of $T = 60$ arrivals, each assigned to demand node $j$ with probability $p_j$. This IID model is well-studied in revenue management and online matching (Gallego and Van Ryzin 1994; Feldman et al. 2009).

**Random-Horizon Temporal Independence (RH-TI).** This model extends DH-TI by making $T$ random (Aouad and Ma 2022). Here, we sample $T$ from a Geometric$(q)$ distribution with $q = 1/61$, ensuring $\mathbb{E}[T] = 60$. Each arrival is then assigned to demand node $j$ with probability $p_j$. This high-variance distribution has a standard deviation roughly equal to its mean.

**Random-Order Spatial Independence (RO-SI).** Here, total demand per node $D_j$ follows a Geometric$(q_j)$ distribution, ensuring $\mathbb{E}[D_j] = 60p_j$ and $\mathbb{E}[T] = 60$. Once sampled, arrivals occur in random order. This modifies the Indep model from Aouad and Ma (2022) by introducing randomness instead of an adversarial order.

We define two weight-setting methods: *uniform weights*, where $p_j = 1/m$, and *reward-based weights*, where $p_j$ is proportional to the total reward from warehouse-to-node edges. The latter increases demand near warehouses, aligning with real-world facility location strategies.

For RH-TI and RO-SI, we also simulate *arrival times* for fulfillment policies. Given $T$ arrivals, we sample and sort $T$ Uniform$(0, 1)$ variables as arrival times. These inform better decisions; e.g., if early arrivals exceed expectations, later demand is likely higher as well.

**Initial Inventory.**

We choose the total inventory $Q$ to take the values in $\{30, 45, 60, 75, 90\}$. Since the expected number of demand arrivals is 60 in all demand models, these initial inventories are such that the ratio between the available inventory and the expected total demand lies within $\{0.50, 0.75, 1.00, 1.25, 1.50\}$.

### 3.6.3 Experimental Setup

To answer our questions, we execute the following experiment. For each of the 90 instances, we:

1. **Inventory placements:** Compute four placements. Fluid placements use exact demand distributions, while Offline and Myopic placements rely on 1000 sampled demand sequences to solve their SAA versions.

2. **Fulfillment policies:** Construct priority lists for each demand node, fulfillment algorithm (myopic, fluid, offline), and starting inventory. Non-myopic policies depend on initial inventory since different levels lead to varying shadow prices. The Offline LP uses the same 1000 demand samples as in step 1.

3. **Performance simulation:** Generate 1000 fresh demand sequences and compute the average reward for each policy across all 20 combinations of 4 inventory placements and 5 fulfillment policies.

We normalize collected rewards by the *prophet's reward*, defining the *competitive ratio* as this ratio. The prophet's reward is the linear relaxation $\max_{x \in \mathrm{CH}(\mathcal{X})} \widehat{\mathsf{OFF}}(x)$, computed using the 1000 demand samples from step 3. It represents the reward of an offline-fulfilling prophet who knows the empirical demand distribution. This value, independent of inventory placement, serves as an upper bound on expected rewards, ensuring the competitive ratio lies between 0 and 1.

### 3.6.4 Results: Comparison Between Fulfillment Policies

We start by comparing the performance obtained by deploying different fulfillment policies combined with different inventory placements.

As an aggregate measurement of performance, Table 3.2 shows the competitive ratio obtained by each pair of solutions, averaged over the 90 instances. The main observation from this table is that the offline shadow prices policy with re-solving achieves the best average competitive ratio for all inventory placements. This dominance goes beyond the average: this fulfillment policy achieves the best performance in 310 out of the 360 pairs of instances and inventory placement procedures. Moreover, even when it is not the best policy, its collected reward is always within 0.5% of the best policy for any instance and inventory placement pair. This is consistent with the

findings in DeValve et al. (2023), where they also find that this particular policy performs the best among other practically motivated policies.

| | Offline Placement | Fluid Placement | Scaled Demand Fluid Placement | Myopic Placement |
|---|---|---|---|---|
| Myopic | 0.956 | 0.933 | 0.960 | 0.966 |
| F-SP | 0.936 | 0.870 | 0.939 | 0.943 |
| O-SP | 0.974 | 0.944 | 0.965 | 0.969 |
| F-SP-R | 0.957 | 0.912 | 0.955 | 0.960 |
| O-SP-R | 0.986 | 0.954 | 0.974 | 0.978 |

Table 3.2: Average competitive ratio obtained by each pair of placement procedure and fulfillment policy.

We can also observe that for each fulfillment policy, the best performance is obtained with either the Offline Placement (O-SP, O-SP-R) or the Myopic Placement (Myopic, F-SP, F-SP-R). This suggests that inventory placements that account for randomness in the demand tend to outperform placement procedures with a certainty-equivalence approach.

### 3.6.5 Results: Comparison Between Placement Procedures

We now turn our focus to a more fine-grained comparison between inventory placements. Since we established that the offline shadow prices policy with re-solving is best-performing, we hereafter focus on results under it, to reflect a high-quality fulfillment policy.

Figure 3.1 shows the competitive ratio for the 45 instances with uniform weights. (The plots for reward-based weights are presented in Appendix B.4.4. The shapes of the graphs and the insights remain unchanged.) The main conclusion drawn from these plots is the superior robustness of Offline Placement. It can be seen that this placement procedure obtains the best or nearly the best performance across all of the displayed instances. Moreover, there is no other procedure that is always close to Offline Placement. For instance, in the long-chain network, the only procedure that is close to Offline Placement for $Q \leq 60$ is the Fluid Placement. On the other hand, if $Q \geq 60$, Fluid Placement has a far worse performance, but Myopic Placement and Scaled Demand Fluid Placement become competitive.

Another important observation is that placement procedures that account for randomness in demand (Offline and Myopic) generally outperform the certainty equivalence counterparts (Fluid and Scaled Demand Fluid). The only exception occurs in the long-chain network with $Q < 60$.

One last observation is that Fluid Placement seems to be especially brittle. This is expected for instances where $Q > 60$ as the total inventory is greater than the total expected demand, which leads to "leftover" inventory. This leftover inventory can be placed arbitrarily and still obtain an optimal solution, so in these cases the solution to the Fluid LP becomes degenerate. This can be fixed for $Q > 60$ by scaling up the demand so it is equal to the total inventory, but this comes at a cost of potential low performance when $Q < 60$, as observed in the long-chain network. A mixed procedure that scales the demand if $Q > 60$ and does not if $Q \leq 60$ also fails to obtain the robustness seen for Offline Placement. This can be observed in the RDC-FDC network, where Fluid Placement performs worse than its Scaled Demand counterpart even if $Q < 60$.

In broader terms, the complete graph seems to be the hardest instance in terms of maximizing the competitive ratio. In the long-chain and RDC-FDC networks, Offline Placement achieves a competitive ratio of at least 0.98, regardless of the starting inventory. For the complete network, this ratio can be as low as 0.94 for low inventory regimes. This could be due to the high flexibility of the complete network, where more subtle hindsight optimal assignments can be constructed. In terms of demand models, RH-TI and RO-SI appear to be harder than DH-TI. Which demand model is the hardest to approximate depends on the graph structure.

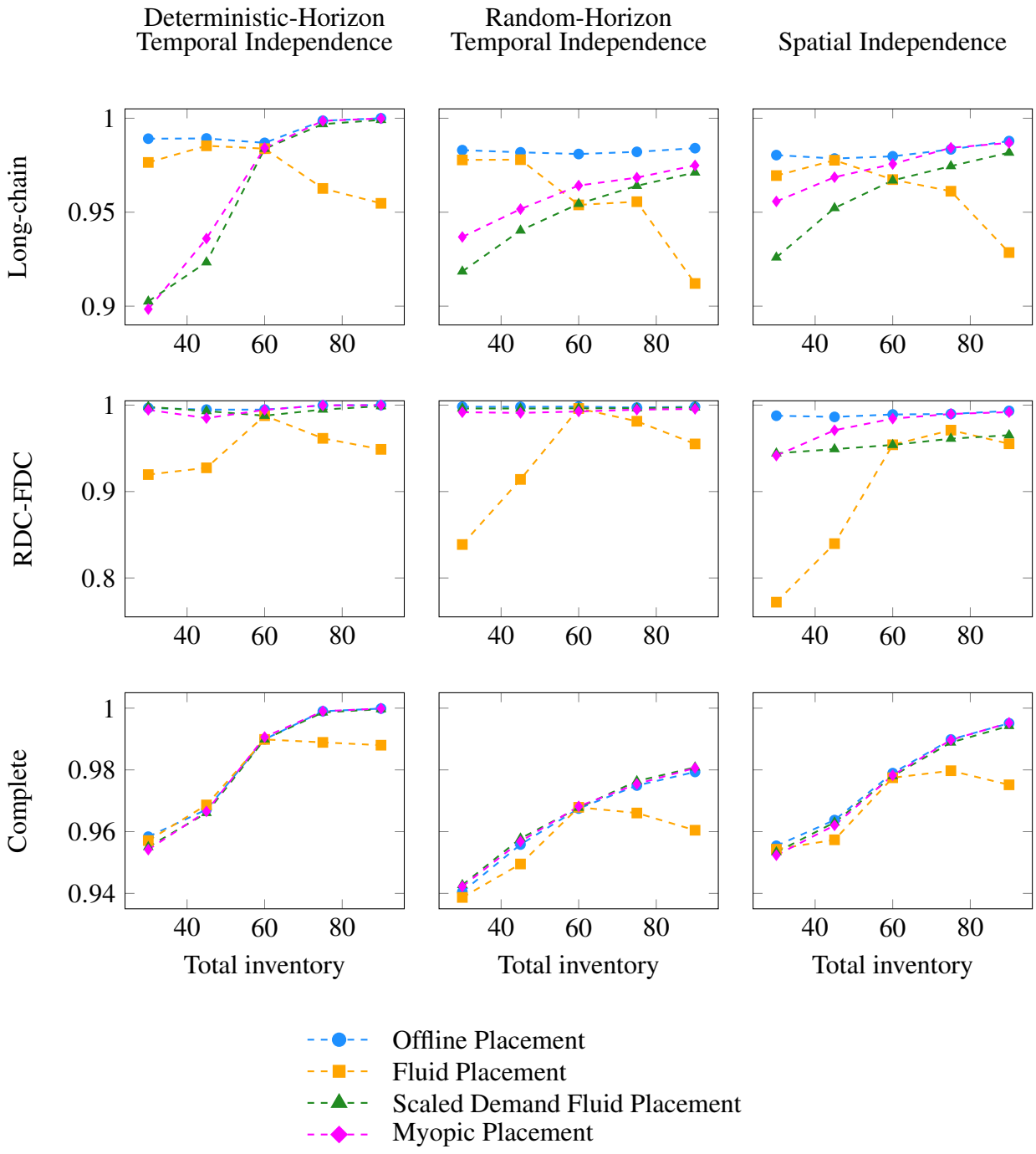


Figure 3.1: Competitive ratio as a function of total inventory, for all combinations of graphs and demand models with uniform weights. Each row corresponds to a different network structure and each column corresponds to a different demand model.

# Chapter 4: Data-Driven Online Matching: A Comparison of Policy Approaches Under Limited Data

## 4.1 Introduction

Data-driven decision-making is increasingly prevalent in online allocation and matching problems, particularly with the surge of reinforcement learning and simulation-based optimization techniques (Azagirre et al. 2024). These advances have empowered decision makers to leverage historical or simulated data to optimize complex dynamic systems. However, despite this progress, a central question remains largely unanswered: how much data is necessary for these approaches to outperform simpler, data-agnostic heuristics? Moreover, in which scenarios does data availability change the relative performance among model-based, model-free, and heuristic strategies?

In this work, we aim to shed light on these questions in the context of the online bipartite matching problem (Karp, Vazirani, and Vazirani 1990). In this setting, an agent with a set of available resources faces an (a priori unknown) sequence of request types, with each request type offering a different reward for each of the resources. When faced with a request type, the agent must irrevocably match it to at most one available resource to maximize the total reward. We assume that this sequence is drawn from a distribution that is unknown to the agent. Instead, the agent has access to sample realizations of request sequences but does not know the underlying distribution explicitly.

We compare a broad set of algorithmic strategies, including model-based policies that assume specific parametric structures on the demand process, data-agnostic heuristics that ignore demand information, and model-free policies trained through simulation and black-box optimization. For data-agnostic policies, we consider a basic greedy algorithm, and the Multi-price Balance algorithm by Ma and Simchi-Levi (2020). The model-based policies encompass optimal policies under

the assumption of three sequence distributions; these policies are computed via dynamic programming. Another model-based approach we study is that of linear programming (LP) dual methods, where the dual solutions are used to guide decisions as estimators of opportunity costs of depleting resources (DeValve et al. 2023). For the model-based approaches, we develop a parametrized generalization of Multi-price Balance and train these parameters through simulation and black-box optimization. With the exception of the dynamic programming optimal policies, all of the above work by greedily selecting the resource with the highest "pseudo reward", defined as a reward minus some penalizer. This algorithmic design motivates our last model-free approaches, where we use the sample sequences to train a neural network that outputs these penalizers in a way that allows us to collect as much reward as possible.

We are particularly interested in understanding the statistical efficiency, robustness,[1] and practical considerations associated with each approach. Our findings indicate that model-based algorithms are highly sample-efficient when well-specified but are brittle under model misspecification. In contrast, model-free approaches, while statistically inefficient in low-data regimes, can outperform when trained on abundant data. Data-agnostic heuristics, despite their simplicity, can be surprisingly competitive in some settings.

Our work contributes to the growing literature that systematically benchmarks data-driven approaches for online matching under stochastic uncertainty. It also provides practical guidelines for choosing the appropriate matching policy depending on the data regime and system characteristics.

### 4.1.1 Further Related Work

**Online Bipartite Matching**. From a theoretical perspective, online matching problems have been extensively studied in the worst-case adversarial setting (Karp, Vazirani, and Vazirani 1990; Mehta et al. 2007; Aggarwal et al. 2011; Fahrbach et al. 2022). The pessimistic nature of adversarial inputs combined with the availability of historical data motivated the study of stochastic inputs

[1] In this work, we say that a policy is robust if it achieves a satisfactory performance across varied instances under the assumption that the sample sequences come from the true underlying distribution. An interesting research direction is to design policies that are robust to distribution shifts between training and testing samples.

drawn from a known distribution (Goel and Mehta 2008; Feldman et al. 2009; Manshadi, Gharan, and Saberi 2012). These models have motivated the development of model-based policies that rely on LP relaxations (Jaillet and Lu 2014). Shadow-price based heuristics, such as those used in the LP-resolving policies, are closely related to this line of work.

**Data-driven variants**. A rich stream of literature studies the online bipartite matching without assuming full distributional knowledge. Karande, Mehta, and Tripathi (2011) studies the performance of Ranking under the unknown distribution with and random order model. A recent trend of papers studies online bipartite matching with machine learning augmented algorithms. Here, algorithms have additional input in the form of a machine learning prediction about the graph or sequences, which is learned from previous data. Examples include predictions about the nodes' degrees (Chen and Indyk 2021) and suggested matchings for the two-stage matching problem (Jin and Ma 2022).

**Experimental Benchmarking**. Several works have explored data-driven approaches for online bipartite matching. Alomrani, Moravej, and Khalil (2021) propose a deep reinforcement learning approach under the i.i.d. demand model, while Borodin, Karavasilis, and Pankratov (2020) perform an extensive experimental comparison of algorithms assuming a known i.i.d. distribution. Our work generalizes these settings by considering multiple demand models and analyzing performance across varying data regimes.

**Further Algorithmic Approaches**. The families of algorithms that we benchmark is by no means exhaustive. Other approaches include, and are not limited to primal sampling methods (Banerjee, Gurvich, and Vera 2020), proportional allocation (Agrawal, Zadimoghaddam, and Mirrokni 2018), and gradient-based methods Balseiro et al. 2023.

### 4.2 Problem Statement

We study the online bipartite matching problem, where a decision maker faces an unknown stream of requests that have to be irrevocably matched to resources to maximize collected rewards. In this problem, there is an underlying bipartite graph $G = ([n], [m], E)$. Here, $[n]$ are the *offline*

*nodes* or *resources*, which we index using $i$. We use $k_i$ to denote the number of copies of resource $i \in [n]$ available at the beginning of the process. Set $[m]$ contains all possible *online nodes* or *request types*, which we index with $j$.

The decision maker faces a *sequence* of $T$ requests $J = (j_t)_{t=1}^{T}$, unknown in advance, where $j_t = j$ if and only if the $t$-th arrival is of type $j \in [m]$. We call $T$ the time horizon, which is deterministic and known to the decision maker. At each time step $t$, the decision maker observes a request type $j_t$ and must choose to match it to a neighboring offline node $i \in \delta(j_t) := \{i \in [n] : (i, j_t) \in E\}$, or leave it unmatched. If the decision maker matches $j_t$ with $i$, he collects a reward of $r_{i,j_t} \in [0, 1]$ and depletes a unit of resource $i$. If $j_t$ is unmatched, no reward is collected, and no resource is depleted. The decision maker's goal is to maximize the sum of the collected rewards.

In this work, we focus on stochastic sequences that follow a *demand model*, defined as a distribution over sequences $J$ of $T$ requests. A simple example of a demand model is the iid model, where at each time step $t$ the request will be of type $j$ with probability $p_j$, independent of the past and the future. More examples can be found in Subsection 4.4.1. The demand model is unknown to the decision maker, who instead has access to $N$ samples of arrival sequences $J^{(1)}, \ldots, J^{(N)}$. The goal of this work is to analyze and benchmark the performance of different data-driven online bipartite matching algorithms in terms of their robustness, statistical efficiency and practicality.

## 4.3 Algorithms

In its most general form, an algorithm for data-driven online bipartite matching maps, at each step $t \in [T]$, the training data $(J^{(s)})_{s=1}^{N}$ and the history observed up to that step to a matching decision. The history contains the partial sequence of arrivals $(j_{t'})_{t'=1}^{t}$, the leftover units of inventory $\ell_{it}$ available for each $i \in [n]$, the actual time step $t$, and the previous matching decisions.

### 4.3.1 Algorithm Classification

In this subsection, we propose a several classifications for algorithms in terms of the information they make use of, the logic behind their decision-making, the assumptions they make, and the

way the are computed/optimized.

**Information used**

We start by distinguishing *data-driven* and *data-agnostic* algorithms. Simply put, data-driven algorithms make use of the samples $(J^{(s)})_{s=1}^{N}$ to make decisions, and data-agnostic algorithms ignore the samples. We distinguish *inventory-, time-, demand-, and decision-aware* algorithms. As their names suggest, these algorithms make use of the current inventory levels, current time step, partially observed demand, and previous decisions to make their decisions, respectively. An algorithm that only uses the information of whether a resource is completely depleted to deem it unfeasible is not considered inventory-aware. Notice that these sets of information are related. For instance, depleted inventory can indicate that more time has passed. We still make these distinctions because explicit use of the information can be useful for guiding decisions, and for practical considerations, since using different sets of information requires keeping track of different metrics throughout the process.

**Model-based vs. Model-free**

Model-based algorithms make assumptions on the ground truth (e.g., iid arrivals), and optimize accordingly. Model-free algorithms, on the other hand, make no assumptions about the demand model, but take the form of a parametrized policy. These parameters are trained using black-box optimization and simulation.

### 4.3.2 Algorithmic Benchmarks

We proceed by listing all the algorithmic benchmarks we implement.

**Myopic**

This is the simplest benchmark, solving for

$$\arg \max_{i \in [n]: \ell_{it} > 0} r_{ij_t}.$$

It can be interpreted as a pseudo-reward greedy with penalizers all equal to 0. It is data-agnostic and neither demand, inventory, nor time aware.

**Multi-Price Balance**

This algorithm, developed in Ma and Simchi-Levi (2020), aims to balance the resource consumption by incorporating a penalizing function that is increasing in the fraction of the resource depleted. That is, if $w_{it} = (k_i - \ell_{it})/x_i$, then this algorithms solves for

$$\arg \max_{i \in \delta(j_t): \ell_{it} > 0} r_{ij_t} - \phi_i(w_{it}).$$

Here, $\phi_i$ is an increasing function that depends on the set of rewards associated to resource $i$. Namely, $P_i = \{r_{ij} \forall j : *(i, j) \in E\}$. Therefore, this is a data-agnostic, inventory-aware algorithm. It is not demand- or time-aware.

**Enhanced Multi-Price Balance**

This algorithm builds upon Multi-Price Balance by incorporating $n + 1$ learnable parameters, which makes it time-aware. In particular, this parametrized policy learns $(\theta_i)_{i \in [n]}$ and $\gamma$, and chooses

$$\arg \max_{i \in \delta(j_t): \ell_{it} > 0} r_{ij_t} - (\phi_i(w_{it}) + \theta_i)(1 - e^{\gamma\left(\frac{t}{T} - 1\right)}).$$

Parameter $\gamma$ allows to decrease the penalziers over time, even if inventory is not being depleted (which Multi-Price Balance cannot do). Parameters $(\theta_i)_{i \in [n]}$ serve as an offset and can capture resource-level effects present in demand samples (which $\phi_i$ cannot by definition). The parame-

ters are learned using black-box optimization and simulation. This is a model-free, data-driven, inventory and time-aware policy.

### Neural Network Opportunity Costs

This algorithm leverages deep learning to learn the penalizing functions $f(x_i)$ that yield the highest expected reward. It solves for

$$\arg \max_{i \in \delta(j_t): \ell_{it} > 0} r_{ij_t} - f(x_i),$$

where $f$ is a multi-layer perceptron. We have two variants of this neural network, each one with a different set of inputs. The first one consists in three inputs: $w_{it}$, $t/T$, and a third engineered input, defined as follows. Let $d_{jt}$ be the number of requests of type $j$ that have arrived up to time $t$. We define $\bar{r}_{it} := \sum_{j:(i,j)\in E} r_{ij} d_{jt}/t$, and it can be interpreted as a proxy for the value of a unit of resource $i$. This neural network then has 3 inputs. The other variant includes the three mentioned inputs, plus a one-hot encoding of the resource $i$, thus having an input of dimension $n + 3$. We call these variants *Neural Network Opportunity Costs*, and *Neural Network Opportunity Costs with ID*, respectively

Both neural networks have a simple architecture with a linear layer mapping the input to an 8-dimensional vector, followed by a ReLu layer, a linear layer that maps 8-dimensions to 1, and a final sigmoid so that the output is always between 0 and 1. (Recall the rewards are bounded to lie between 0 and 1.)

These are model-free data-driven policies that are time, demand, and inventory aware.

### Fluid LP Shadow Prices

This policy computes the penalizers for each resource using the dual variables of the Fluid Linear Program (Gallego and Van Ryzin 1997). For a fixed sequence $J$, define $D_j := \sum_{t=1}^{T} \mathbb{1}\{j_t = j\}$ as the total number of type $j$ online nodes that arrive in the sequence. Notice this is a random

variable. The fluid linear program is as follows.

$$\begin{aligned}
\max_{y \geq 0} \quad & \sum_{i \in [n]} \sum_{j \in [m]} y_{ij} r_{ij} & \\
\text{s.t.} \quad & \sum_{i \in [n]} y_{ij} \leq \mathbb{E}[D_j] & \forall j \in [m], \\
& \sum_{j \in [m]} y_{ij} \leq k_i & \forall i \in [n]. \qquad (4.1)
\end{aligned}$$

In our algorithm, $\mathbb{E}[D_j]$ is estimated from the data, and we set the penalizer for resource $i$ to be equal to $\lambda_i$, the dual variable of its corresponding constraint (4.1). Furthermore, this algorithm will re-solve for these dual variables as the time horizon goes by. In particular, let $D_{jt} : \sum_{t'=t}^{T} \mathbb{1}\{j_t = j\}$ as the number of type $j$ arrivals from time $t$ onward. If there is a re-solving epoch at time $t$, then the expected demand in constraint (4.1) is replaced by $\mathbb{E}[D_{j,t+1}]$, and the inventory levels in (4.1) are replaced with the current inventory levels $\ell_{it}$.

It can be argued that this is a model-based algorithm, in the sense that the way in which it estimates expected demands assumes independence across time steps. Apart from that, it is data-driven, time-aware and inventory-aware.

**Offline LP Shadow Prices**

This algorithm, like the previous one, uses the dual variables of a linear program as penalizers. The difference is that this algorithm uses the Offline linear program, presented below. In the LP, $D_j^s$ is the aggregate demand computed from the sample sequence $J^{(s)}$.

$$\begin{aligned}
\max_{x,y \geq 0} \quad & \frac{1}{N} \sum_{s=1}^{N} \sum_{i \in [n]} \sum_{j \in [m]} r_{ij} y_{ij}^s & \\
\text{s.t.} \quad & \sum_{i=1}^{n} y_{ij}^s \leq D_j^s & \forall j \in [m], s \in [N], \\
& \sum_{i=j}^{m} y_{ij}^s \leq k_i & \forall i \in [n], s \in [N]. \qquad (4.2)
\end{aligned}$$

The value of this LP is the reward collected from a hindsight optimal matching, in average over the training samples. In this algorithm, we use $\lambda_i^s$ to denote the dual variable associated with constraint (4.2), and define the penalizer as $\lambda_i := \sum_{s=1}^{N} \lambda_i^s$. As in the previous algorithm, the shadow prices are periodically re-calculated using the remaining time steps of all sample sequences.

This model can also be thought of as model-based, but to a lower extent than its Fluid counterpart. This is because when resolving at time $t$, it uses all samples from time $t+1$ onward, whereas some of these samples can be less likely to resemble the current sample-path. On the other hand, it does not assume independence from $t+1$ onward. This is a data-driven, inventory and time aware policy. This algorithm has been used as a benchmark in, e.g., DeValve et al. (2023).

**Simple Interpolation LP Shadow Prices**

This policy works similarly to the Fluid LP Shadow Prices algorithm, but the expected demands are computed by simple interpolation instead of from the data. Therefore, this is a data-agnostic policy. Let again $d_{jt}$ be the proportion of demand that has been of type $j$ up to time $t$. When resolving at time $t$, this algorithm sets $\mathbb{E}[D_{jt}] = d_{jt} * (T-t)/t$ in constraint (4.1) of the Fluid Linear Program, where $d_{jt}$ is the number of type $j$ arrivals observed up to time $t$. This is a data-agnostic, inventory, demand and time aware policy.

**IID Dynamic Program**

This algorithm assumes that the demand model is IID. That is, on each time step, the arriving type will be $j$ with probability $p_j$. It estimates these probabilities $p_j$ form the data, and solves the optimal dynamic program. Here, the optimal dynamic program is given by the Bellman Equations:

$$V[t,\ell,j] = \max\left\{\sum_{j'\in[m]} p_{j'} V[t,\ell,j'], \max_{i\in\delta(j_t):\ell_{it}>0} r_{ij} + \sum_{j'\in[m]} p_{j'} V[t,\ell-e_i,j']\right\}$$

**Independent Non-Identical Dynamic Program**

This algorithm assumes that the demand model is independent and non-identical. That is, on time step $t$, the arriving type will be $j$ with probability $p_{jt}$. It estimates these probabilities $p_{jt}$ form the data, and solves the optimal dynamic program. Here, the optimal dynamic program is given by the Bellman Equations:

$$V[t, \ell, j] = \max \left\{ \sum_{j' \in [m]} p_{j',t+1} V[t, \ell, j'], \max_{i \in \delta(j_t): \ell_{it} > 0} r_{ij} + \sum_{j' \in [m]} p_{j',t+1} V[t, \ell - e_i, j'] \right\}$$

**Markovian Dynamic Program**

This algorithm assumes that the demand model is Markovian in the latest arrival. That is, on time $t = 1$, the arrival type will be given by an initial distribution $a_j$. Afterward, on any given time step $t \geq 2$ where the arrival on $t - 1$ was of type $j'$, the arriving type will be $j$ with probability $p_{j'j}$. It estimates these probabilities $p_{j'j}$ form the data, and solves the optimal dynamic program. Here, the optimal dynamic program is given by the Bellman Equations:

$$V[t, \ell, j] = \max \left\{ \sum_{j' \in [m]} p_{j,j'} V[t, \ell, j'], \max_{i \in \delta(j_t): \ell_{it} > 0} r_{ij} + \sum_{j' \in [m]} p_{j,j'} V[t, \ell - e_i, j'] \right\}$$

## 4.4 Experimental Setup

In this section, we describe the experimental setup that we use to compare and benchmark the different online bipartite matching algorithms. We are interested in comparing the performance and statistical efficiency of the different algorithms across different lengths of time horizon, demand-to-supply ratios, and ground truth demand models.

The experiments consist roughly in the following steps. First, a sample of $N$ training sequences drawn from the demand model. We use the sample to train our algorithms, and then evaluate them out-of-sample using a test set of 5000 sequences, drawn from the same ground truth distribution.

We use sample sizes $N \in \{1, 5, 25, 125\}$. For each sample size, we generate 6 training samples in order to average out idiosyncrasies of too favorable or unfavorable samples.

### 4.4.1 Demand Models

We carry out our experiments in three different demand models. We test all models with $T \in \{6, 12, 18\}$, and with load factors (defined as total resources over total arrivals) equal to $1/2$ and 1. For independent nonidentical models, a load factor of $1/2$ means setting $k_i = T/(2n)$ for all $i \in [n]$, while a load factor of 1 means setting $k_i = T/n$ for all $i \in [n]$.

For the independent non-identical and Markovian models, we generate 10 random instances (in terms of graph structure and demand model parameters), and for the Bimodal Time Horizon we only use one crafted instance.

#### Independent Non-identical

In this model, each time step $t$ has a distribution over types given by $(p_{jt})_{j\in[m]}$ (Gallego and Van Ryzin 1994; Jasin and Kumar 2012). At each step, the type is drawn from this distribution, independent of the past and the future. Notice that this is exactly the model assumed by the Independent non-identical dynamic program, so in this case it will be correctly specified.

#### Markovian

This model is specified by an initial distribution $(a_j)_{j\in[m]}$ and a transition matrix with entries $p_{jj'}$ for every $(j, j')$ in $[m] \times [m]$. The first arrival will be drawn from the initial distribution $a$, and the next random arrivals will be drawn according to the transition matrix. Notice that this is exactly the model assumed by the Markovian dynamic program, so in this case it will be correctly specified.

**Bimodal Time Horizon**

This model, based on the Correl model in Aouad and Ma (2022), is constructed in a somewhat adversarial way to make the model-based dynamic programs misspecified. It will be accompanied by a specific graph structure with $n = 1$ and $m = 3$. The first online node will be connected to the resource and have a reward of 1, whereas the second online node will have a reward of $1/2$. The third online node will not be connected to the resource, representing an empty arrival. With probability $1/2$, the arrivals will be IID during the $T$ time steps, with probability $1/5$ of having online node 1 (high reward) and probability $4/5$ of having online node 2 (low reward). With probability $1/2$, only $T/3$ IID arrivals will occur, and the remaining $2T/3$ will correspond to type 3 arrivals, which cannot be matched.

### 4.4.2 Instance Generation

For the first two demand models (independent non-identical and Markovian), we generate 10 random instances and take the median performance to avoid the results being affected by idiosyncrasies of particular instances.

**Random Graph Generation**

The first step is to generate 10 random graphs. To generate a random graph, we first fix $n = 3$ and $m = 15$. For each resource $i$, we generate a price set $P_i = \{r_i^{(1)}, r_i^{(2)}\}$ by sampling $r_i^{(1)}$ and $r_i^{(2)}$ as independent Uniform$(0, 1)$ random variables. Then, fixing this $i$, for each $j \in [m]$ we either:

- Include $(i, j) \in E$ with $r_{ij} = r_i^{(1)}$ with probability $1/3$,
- Include $(i, j) \in E$ with $r_{ij} = r_i^{(2)}$ with probability $1/3$,
- Do not include $(i, j) \in E$ with probability $1/3$.

We repeat this for all $i \in [n]$ and the resulting graph is used for the experiments. We remark that the same 10 graphs are used for both demand models.

**Random Demand Model Parameter Generation**

For generating random distributions, we use the fact that we can sample a uniform point from the $m$-dimensional simplex by sampling $n$ iid Exponential random variables with mean 1 and scaling them so they add up to 1 (Devroye 1986).

For the independent non-identical demand model, we use this sampling from the simplex to generate each of the distributions $(p_{tj})_{j\in[m]}$, for every $t \in [T]$. For the Markovian demand model, we set the initial distribution to be uniform. For the transition matrix, every row will be sampled using the exponential scheme, with a slight modification. To obtain some degree of auto correlation, we add a bias term of 3 to the diagonal terms in the transition matrix before scaling it down to the simplex.

## 4.5 Results

We present the obtained results in terms of performance, training time, and testing time.

### 4.5.1 Performance

We measure the performance of a policy using its competitive ratio, defined as the average reward collected on the testing set, divided by the average hindsight optimal reward that could be collected from the testing set. The results for the independent non-identical, Markovian, and Bimodal demand models are presented in Figures 4.1 to 4.3, respectively. Each figure consists of six plots. The two columns represent the load-factor of 0.5 or 1, respectively, and the columns represent the time horizons of 6, 12, and 18 respectively.

**Load factor**. A first observation, valid across all scenarios, is that the competitive ratios get better for all policies when the load factor increases. This is to be expected, since having less inventory makes it easier to make 'mistakes' with respect to a hindsight optimal matching. Another interesting behavior is that data-agnostic policies become more competitive with a higher load factor.

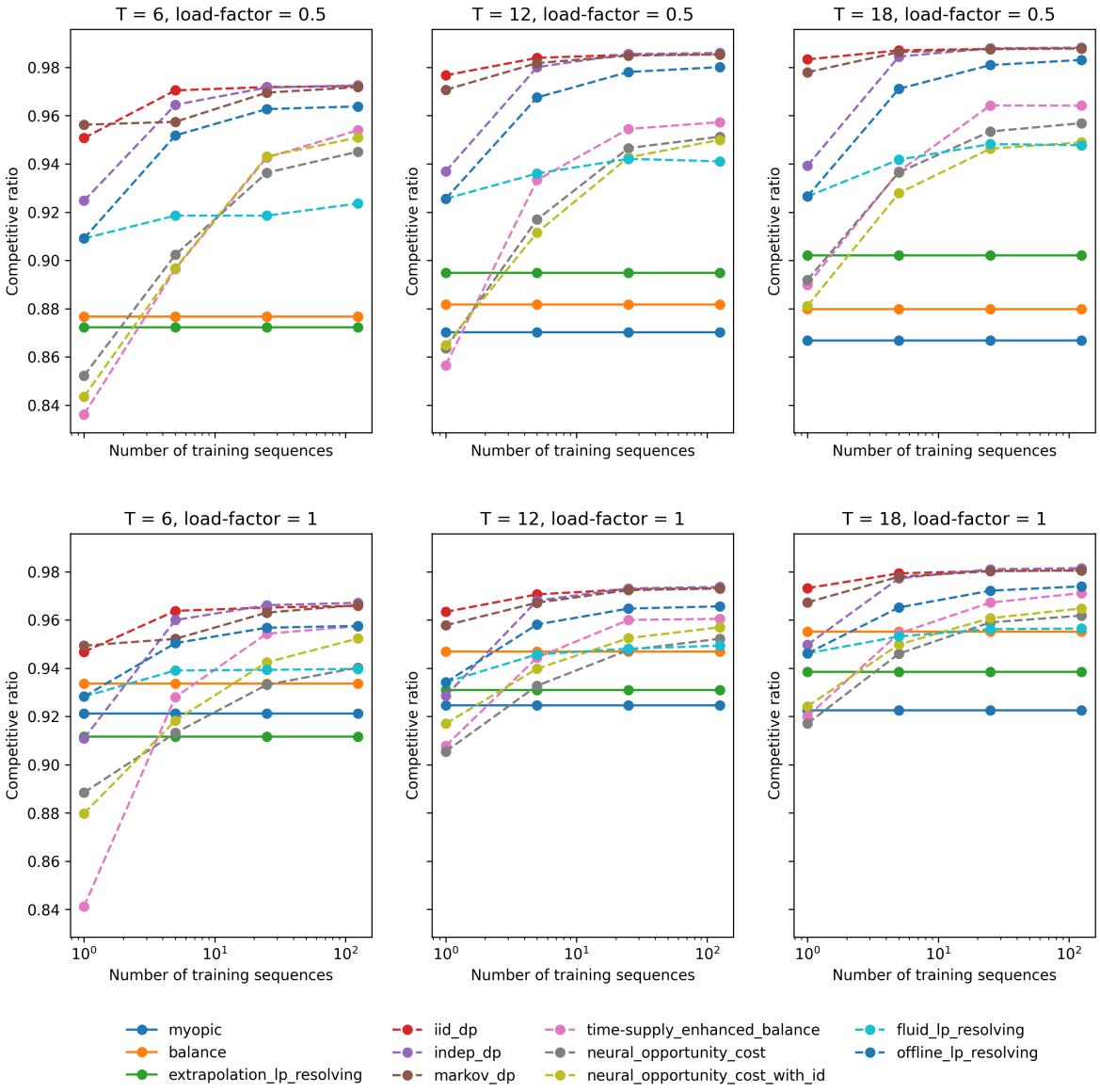


Figure 4.1: Performance results for the independent non-identical model.

**Time Horizon**. Something similar happens with the time horizon. For a fixed load-factor, the competitive ratios tend to go upwards as the time horizon gets longer. This behavior can be explained by the fact that a longer time horizon provides more data for some models. An exception to this rule is the myopic policy, which tends to get worse as the time horizon gets longer.

**Model-based Dynamic Programs**. We observe that for the independent non-identical and the Markovian demand models, their corresponding dynamic programs have the best performance. This is expected for when the model is correctly specified. Even when mis-specified, the iid dynamic program obtains optimal or near-optimal performance in the first two demand models, apart from being usually the most statistically efficient. Model-based dynamic programs do not achieve the best performance in the bimodal demand model, where they are severely mis-specified. They still converge to their asymptotic performance, suggesting high bias and low variance.

**LP re-solving algorithms**. Out of the algorithms that compute penalizers based on dual variables of linear programs, the Offline Shadow prices has the most dominant performance. The only exceptions are two Bimodal instances, in which the extrapolation LP slightly outperforms the Offline LP. For the independent non-identical and Markovian models, the Offline LP Shadow Prices obtain the best performance of all policies that are not model-based dynamic programs for any sample size larger than 1. One interesting behavior is that the Extrapolation LP gets better as the time horizon gets longer. This can be explained by the fact that only 5 re-solving epochs are deployed, no matter the length of the time horizon. Therefore, each re-solving epoch has more information about demand if the time horizon is longer.

**Model-free policies**. In the first two demand models, these policies exhibit the worst statistical efficiency, but they achieve a competitive performance in the rich data regime (125 samples). In this regime, they achieve the best performance after the dynamic programs and the Offline Shadow prices. Moreover, in the first two demand models, the enhanced multi-price balance algorithm seems to outperform the neural networks. This is to be expected in the independent non-identical model, since enhanced multi-priced balance is not demand-aware, whereas the neural nets are, giving them space to overfit to the training sample. This effect is also observed in the Markovian

model, which shares the explanation to some extent. For the Bimodal model, on the other hand, the best performance is obtained by the neural networks. This even holds for the smaller data regimes ($N = 5, 25$) in some scenarios. This is explained by the fact that the neural networks are demand-aware, and these demand patterns seem to allow to detect in when to accept the low rewards and when to wait for the high rewards. Another explanation for the dominant performance of the neural networks in this setting is that there is only one resource, so there is no loss of having partial inventory information when estimating opportunity costs. Surprisingly, enhanced multi-priced balance has a very good performance in this setting, even if it is not demand-aware. This demonstrates the advantages of simulation based optimization, where some rewards can indirectly be optimized to detect and capitalize from patterns in the sample sequences.

**Data-agnostic policies**. We observe that multi-priced balance always outperforms myopic. The difference becomes larger when the load factor is larger and the time horizon is larger. Interpolation LP has the worst performance when $T = 6$. For the rest, Interpolation LP performs better than Multi-priced Balance when the load factor is $1/2$, and worse than Multi-priced balance when the load-factor is 1.

### 4.5.2 Training Time

To illustrate the different training times, we observe at the independent non-identical model, with $T = 12$ and load factor equal to $1/2$. The results are displayed in Figure 4.4.

We observe that the highest training times are constant for all policies except for the simulation-based policies and the Offline LP. For simulation-based, this is because all samples are processed sequentially, and this could be partially alleviated with parallel processing of the samples. This would not reduce the increasing nature of training time on the time horizon. For the Offline LP, this is because obtaining the initial dual variables becomes slower when more samples are introduced to the LP.

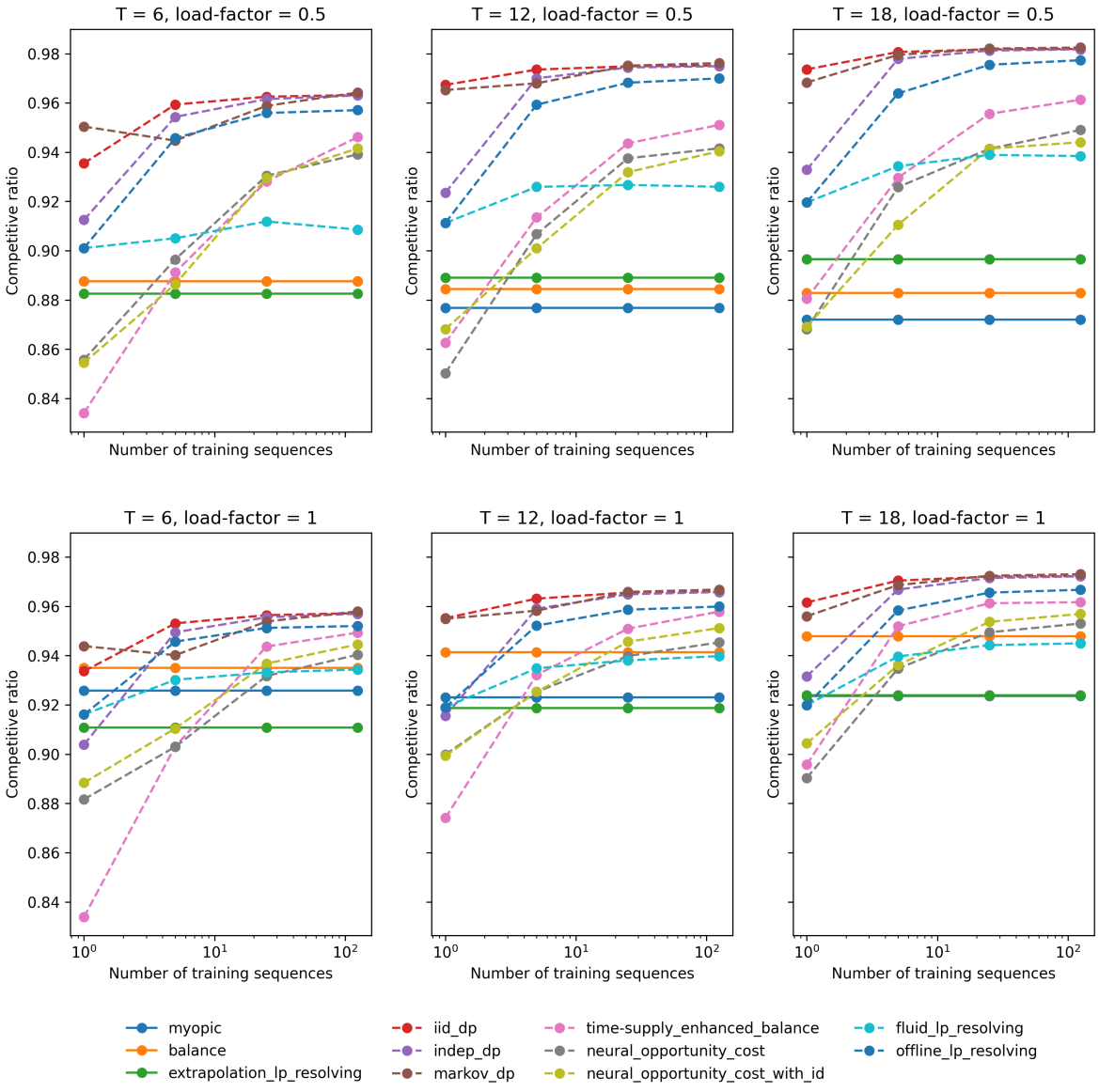


Figure 4.2: Performance results for the Markovian model.

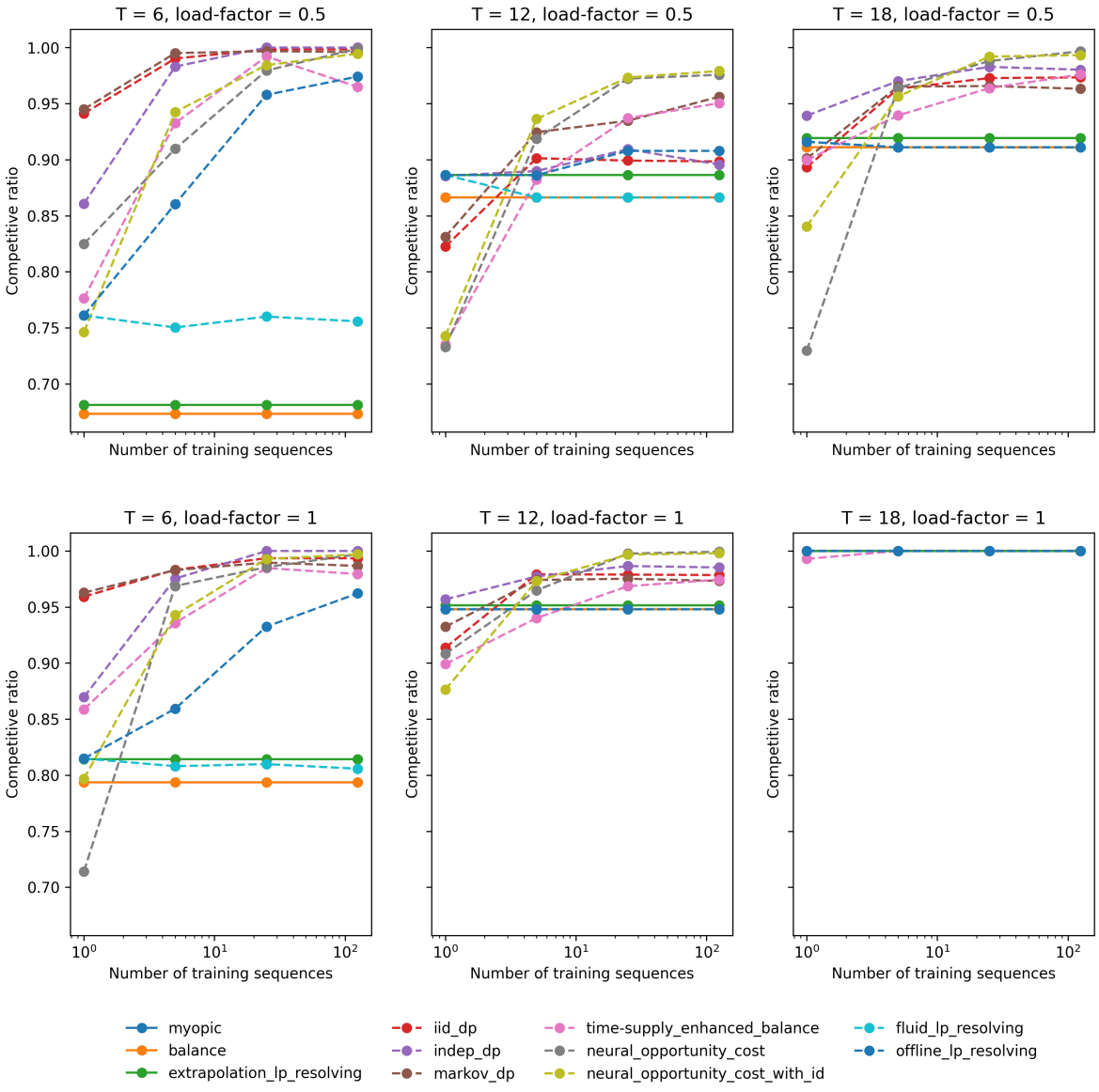


Figure 4.3: Performance results for the bimodal model.

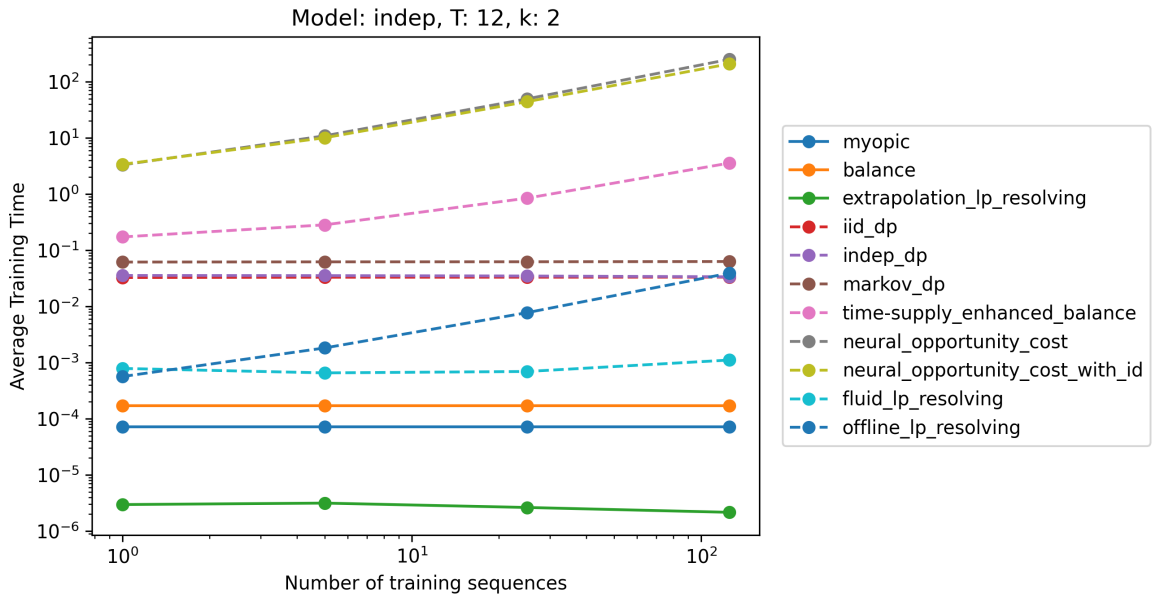


Figure 4.4: Training time for different policies under the independent non-identical distribution, $T = 12$, and load-factor= $1/2$.

### 4.5.3 Testing Time

We again use the independent non-identical model, with $T = 12$ and load factor equal to $1/2$ to illustrate the testing times of different policies. This corresponds to the average time (in seconds) it takes to process one sequence. The results are displayed in Figure 4.5.

We observe that the LP-based policies are the slowest, and the Offline LP becomes even slower when the sample size increases. The second-slowest are the neural net policies, followed by Multi-Priced Balance and Enhanced Multi-Priced Balance. The testing time has important implications, since this is usually a bottleneck in applications where decisions have to be made in fractions of seconds. In that sense, model-free, simulation-based policies offer a sweet spot between robustness and inference time.

## 4.6 Practical Considerations

We close this chapter by discussing practical considerations of the different policies we benchmark. **Data-agnostic** policies require the least amount of engineering, and by definition they do not

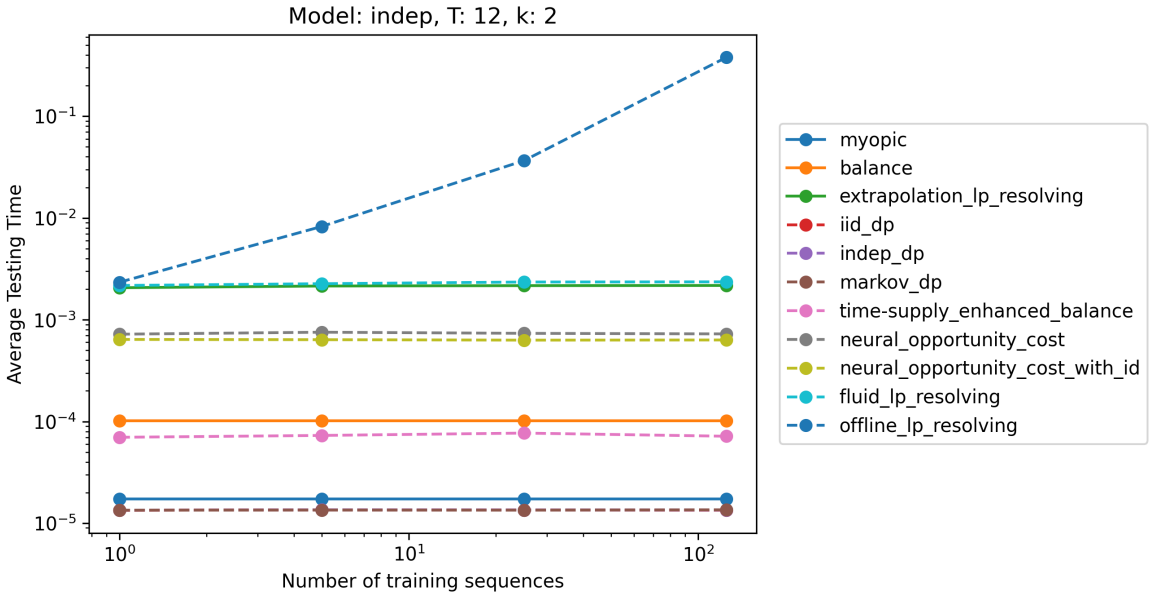


Figure 4.5: Testing time for different policies under the independent non-identical distribution, $T = 12$, and load-factor= $1/2$.

require data, but exhibit low robustness in performance. **Model-based dynamic programs** exhibit the best performance when well specified (or close to well specified) and they exhibit high sample efficiency, but they are impractical. They heavily depend on the time discretizations, and computing them for a large time horizon is computationally infeasible. From the **LP based** policies, the Offline LP achieves high performance and robustness. That being said, its base deployment time is costly, and it scales with the sample size. This makes it undesirable for applications that require fast responses. **Model-free, simulation-based** optimization appears to be in the sweet spot of robustness, training time, and testing time, provided the data regime is medium-to-high. The particular simulation-based policies we study can also directly generalize to continuous time settings, provided a notion of a time horizon.

# References


Fu, Hao, Jian Li, and Pan Xu (2018). "A PTAS for a Class of Stochastic Dynamic Programs". In: *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Purohit, Manish, Sreenivas Gollapudi, and Manish Raghavan (2019). "Hiring under uncertainty". In: *International Conference on Machine Learning*. PMLR, pp. 5181–5189.

Segev, Danny and Sahil Singla (2021). "Efficient approximation schemes for stochastic probing and prophet problems". In: *Proceedings of the 22nd ACM Conference on Economics and Computation*, pp. 793–794.

Bradac, Domagoj, Sahil Singla, and Goran Zuzic (2019). "(Near) Optimal Adaptivity Gaps for Stochastic Multi-Value Probing". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Gupta, Anupam and Viswanath Nagarajan (2013). "A stochastic probing problem with applications". In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer, pp. 205–216.

Gupta, Anupam, Viswanath Nagarajan, and Sahil Singla (2016). "Algorithms and adaptivity gaps for stochastic probing". In: *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*. SIAM, pp. 1731–1747.

— (2017). "Adaptivity gaps for stochastic probing: Submodular and XOS functions". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, pp. 1688–1702.

Krengel, Ulrich and Louis Sucheston (1977). "Semiamarts and finite values". In: *Bulletin of the American Mathematical Society* 83.4, pp. 745–747.

— (1978). "On semiamarts, amarts, and processes with finite value". In: *Probability on Banach spaces* 4, pp. 197–266.

Samuel-Cahn, Ester (1984). "Comparison of threshold stop rules and maximum for independent nonnegative random variables". In: *the Annals of Probability*, pp. 1213–1216.

Kleinberg, Robert and Seth Matthew Weinberg (2012). "Matroid prophet inequalities". In: *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pp. 123–136.

Esfandiari, Hossein et al. (2017). “Prophet secretary”. In: *SIAM Journal on Discrete Mathematics* 31.3, pp. 1685–1701.

Hill, TP (1983). “Prophet inequalities and order selection in optimal stopping problems”. In: *Proceedings of the American Mathematical Society* 88.1, pp. 131–137.

Beyhaghi, Hedyeh et al. (2021). “Improved Revenue Bounds for Posted-Price and Second-Price Mechanisms”. In: *Operations Research* 69.6, pp. 1805–1822.

Agrawal, Shipra, Jay Sethuraman, and Xingyu Zhang (2020). “On optimal ordering in the optimal stopping problem”. In: *Proceedings of the 21st ACM Conference on Economics and Computation*, pp. 187–188.

Yan, Qiqi (2011). “Mechanism design via correlation gap”. In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, pp. 710–719.

Arnosti, Nick and Will Ma (2021). “Tight guarantees for static threshold policies in the prophet secretary problem”. In: *arXiv preprint arXiv:2108.12893*.

Gallego, Guillermo and Huseyin Topaloglu (2019). *Revenue management and pricing analytics*. Vol. 209. Springer.

Cominetti, Roberto et al. (2010). “Optimal selection of customers for a last-minute offer”. In: *Operations research* 58.4-part-1, pp. 878–888.

Gallego, Guillermo and Danny Segev (2022). “A Constructive Prophet Inequality Approach to The Adaptive ProbeMax Problem”. In: *arXiv preprint arXiv:2210.07556*.

Gandhi, Rajiv et al. (2006). “Dependent rounding and its applications to approximation algorithms”. In: *Journal of the ACM (JACM)* 53.3, pp. 324–360.

Goodrich, Michael T and Roberto Tamassia (2001). *Algorithm design: foundations, analysis, and internet examples*. John Wiley & Sons.

Chen, Annie I and Stephen C Graves (2021). “Item aggregation and column generation for online-retail inventory placement”. In: *Manufacturing & Service Operations Management* 23.5, pp. 1062–1076.

Ryzin, Garrett van (2024). *Consensus planning protocol (CPP): A market-based approach to coordination at Amazon*. Plenary talk presented at the 25th ACM Conference on Economics and Computation. Plenary talk.

Maggiar, Alvaro, Lee Dicker, and Michael Mahoney (2024). “Consensus planning with primal, dual, and proximal agents”. In: *arXiv preprint arXiv:2408.16462*.

Govindarajan, Aravind, Amitabh Sinha, and Joline Uichanco (2021b). "Joint inventory and fulfillment decisions for omnichannel retail networks". In: *Naval Research Logistics (NRL)* 68.6, pp. 779–794.

Bai, Yicheng et al. (2022). "Coordinated inventory stocking and assortment personalization". In: *Available at SSRN 4297618*.

Chen, Xingxing et al. (2022). "Approximation schemes for the joint inventory selection and online resource allocation problem". In: *Production and Operations Management* 31.8, pp. 3143–3159.

Arlotto, Alessandro, Irem Nur Keskin, and Yehua Wei (2023). "Online Demand Fulfillment Problem with Initial Inventory Placement: A Regret Analysis". In: *Available at SSRN 4666493*.

Jasin, Stefanus, Sheng Liu, and Jinglong Zhao (2024). "Inventory Allocation under the Greedy Fulfillment Policy: The (Potential) Perils of the Hindsight Approach". In: *Available at SSRN 4815848*.

Xu, Ping Josephine, Russell Allgor, and Stephen C Graves (2009). "Benefits of reevaluating real-time order fulfillment decisions". In: *Manufacturing & Service Operations Management* 11.2, pp. 340–355.

Acimovic, Jason and Stephen C Graves (2015). "Making better fulfillment decisions on the fly in an online retail environment". In: *Manufacturing & Service Operations Management* 17.1, pp. 34–51.

Jasin, Stefanus and Amitabh Sinha (2015). "An LP-based correlated rounding scheme for multi-item ecommerce order fulfillment". In: *Operations Research* 63.6, pp. 1336–1351.

Feldman, Jon et al. (2009). "Online stochastic matching: Beating 1-1/e". In: *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, pp. 117–126.

Alaei, Saeed, MohammadTaghi Hajiaghayi, and Vahid Liaghat (2012). "Online prophet-inequality matching with applications to ad allocation". In: *Proceedings of the 13th ACM Conference on Electronic Commerce*, pp. 18–35.

Brubach, Brian et al. (2020). "Online stochastic matching: New algorithms and bounds". In: *Algorithmica* 82.10, pp. 2737–2783.

Papadimitriou, Christos et al. (2021). "Online stochastic max-weight bipartite matching: Beyond prophet inequalities". In: *Proceedings of the 22nd ACM Conference on Economics and Computation*, pp. 763–764.

Bai, Yicheng, Paat Rusmevichientong, and Huseyin Topaloglu (2022). "Joint placement, delivery promise and fulfillment in online retail". In: *Delivery Promise and Fulfillment in Online Retail (May 3, 2022)*.

DeValve, Levi et al. (2023). "Understanding the value of fulfillment flexibility in an online retailing environment". In: *Manufacturing & service operations management* 25.2, pp. 391–408.

Acimovic, Jason and Stephen C Graves (2017). "Mitigating spillover in online retailing via replenishment". In: *Manufacturing & Service Operations Management* 19.3, pp. 419–436.

Chen, Annie I-An (2017). "Large-scale optimization in online-retail inventory management". PhD thesis. Massachusetts Institute of Technology.

Maurer, Andreas (2016). "A vector-contraction inequality for rademacher complexities". In: *Algorithmic Learning Theory: 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings 27*. Springer, pp. 3–17.

Shapiro, Alexander and Arkadi Nemirovski (2005). "On complexity of stochastic programming problems". In: *Continuous optimization: Current trends and modern applications*, pp. 111–146.

Aouad, Ali and Will Ma (2022). "A nonparametric framework for online stochastic matching with correlated arrivals". In: *arXiv preprint arXiv:2208.02229*.

Lee, Jon et al. (2010). "Maximizing nonmonotone submodular functions under matroid or knapsack constraints". In: *SIAM Journal on Discrete Mathematics* 23.4, pp. 2053–2078.

Lei, Yanzhe, Stefanus Jasin, and Amitabh Sinha (2018). "Joint dynamic pricing and order fulfillment for e-commerce retailers". In: *Manufacturing & Service Operations Management* 20.2, pp. 269–284.

Lei, Yanzhe et al. (2022). "Joint product framing (display, ranking, pricing) and order fulfillment under the multinomial logit model for e-commerce retailers". In: *Manufacturing & Service Operations Management* 24.3, pp. 1529–1546.

Jasin, Stefanus et al. (2022). "Joint Inventory, Pricing, and Fulfillment Control for a Multi-Warehouse Multi-Store Problem: An Asymptotically Optimal Lagrangian Policy". In: *Sentao and Roy, Debjit and Sun, Zhiyuan, Joint Inventory, Pricing, and Fulfillment Control for a Multi-Warehouse Multi-Store Problem: An Asymptotically Optimal Lagrangian Policy (December 10, 2022)*.

DeValve, Levi, Saša Pekeč, and Yehua Wei (2023). "Approximate submodularity in network design problems". In: *Operations Research* 71.4, pp. 1021–1039.

Zhao, Yanyang et al. (2023). "Managing multi-tier inventory networks with expediting under normal and disrupted modes". In: *Available at SSRN 4204008*.

DeValve, Levi (2023). "Cost Balancing for General Inventory/Fulfillment Networks with Applications to ATO and Multi-item E-retail Problems". In: *Available at SSRN 3961613*.

Van Mieghem, Jan A (1998). "Investment strategies for flexible resources". In: *Management Science* 44.8, pp. 1071–1078.

Mieghem, Jan A Van and Nils Rudi (2002). "Newsvendor networks: Inventory management and capacity investment with discretionary activities". In: *Manufacturing & Service Operations Management* 4.4, pp. 313–335.

Govindarajan, Aravind, Amitabh Sinha, and Joline Uichanco (2021a). "Distribution-free inventory risk pooling in a multilocation newsvendor". In: *Management Science* 67.4, pp. 2272–2291.

Birge, John R and Levi DeValve (2025). "Inventory placement on a network". In: *Operations Research Letters* 60, p. 107240.

DeValve, Levi and Jabari Myles (2023). "Approximation Algorithms for Dynamic Inventory Management on Networks". In: *Available at SSRN 4187297*.

Brubach, Brian et al. (2021). "Improved guarantees for offline stochastic matching via new ordered contention resolution schemes". In: *Advances in Neural Information Processing Systems* 34, pp. 27184–27195.

Nemhauser, George L, Laurence A Wolsey, and Marshall L Fisher (1978). "An analysis of approximations for maximizing submodular set functions—I". In: *Mathematical programming* 14, pp. 265–294.

Jordan, William C and Stephen C Graves (1995). "Principles on the benefits of manufacturing process flexibility". In: *Management science* 41.4, pp. 577–594.

Gallego, Guillermo and Garrett Van Ryzin (1994). "Optimal dynamic pricing of inventories with stochastic demand over finite horizons". In: *Management science* 40.8, pp. 999–1020.

Azagirre, Xabi et al. (2024). "A better match for drivers and riders: Reinforcement learning at lyft". In: *INFORMS Journal on Applied Analytics* 54.1, pp. 71–83.

Karp, Richard M, Umesh V Vazirani, and Vijay V Vazirani (1990). "An optimal algorithm for on-line bipartite matching". In: *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pp. 352–358.

Ma, Will and David Simchi-Levi (2020). "Algorithms for online matching, assortment, and pricing with tight weight-dependent competitive ratios". In: *Operations Research* 68.6, pp. 1787–1803.

Mehta, Aranyak et al. (2007). "Adwords and generalized online matching". In: *Journal of the ACM (JACM)* 54.5, 22–es.

Aggarwal, Gagan et al. (2011). “Online vertex-weighted bipartite matching and single-bid budgeted allocations”. In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, pp. 1253–1264.

Fahrbach, Matthew et al. (2022). “Edge-weighted online bipartite matching”. In: *Journal of the ACM* 69.6, pp. 1–35.

Goel, Gagan and Aranyak Mehta (2008). “Online budgeted matching in random input models with applications to Adwords.” In: *SODA*. Vol. 8, pp. 982–991.

Manshadi, Vahideh H, Shayan Oveis Gharan, and Amin Saberi (2012). “Online stochastic matching: Online actions based on offline statistics”. In: *Mathematics of Operations Research* 37.4, pp. 559–573.

Jaillet, Patrick and Xin Lu (2014). “Online stochastic matching: New algorithms with better bounds”. In: *Mathematics of Operations Research* 39.3, pp. 624–646.

Karande, Chinmay, Aranyak Mehta, and Pushkar Tripathi (2011). “Online bipartite matching with unknown distributions”. In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pp. 587–596.

Chen, Justin Y and Piotr Indyk (2021). “Online bipartite matching with predicted degrees”. In: *arXiv e-prints*, arXiv–2110.

Jin, Billy and Will Ma (2022). “Online bipartite matching with advice: Tight robustness-consistency tradeoffs for the two-stage model”. In: *Advances in Neural Information Processing Systems* 35, pp. 14555–14567.

Alomrani, Mohammad Ali, Reza Moravej, and Elias B Khalil (2021). “Deep policies for online bipartite matching: A reinforcement learning approach”. In: *arXiv preprint arXiv:2109.10380*.

Borodin, Allan, Christodoulos Karavasilis, and Denis Pankratov (2020). “An experimental study of algorithms for online bipartite matching”. In: *Journal of Experimental Algorithmics (JEA)* 25, pp. 1–37.

Banerjee, Siddhartha, Itai Gurvich, and Alberto Vera (2020). *Constant regret in online allocation: On the sufficiency of a single historical trace*.

Agrawal, Shipra, Morteza Zadimoghaddam, and Vahab Mirrokni (2018). “Proportional allocation: Simple, distributed, and diverse matching with high entropy”. In: *International Conference on Machine Learning*. PMLR, pp. 99–108.

Balseiro, Santiago R et al. (2023). “Robust budget pacing with a single sample”. In: *International Conference on Machine Learning*. PMLR, pp. 1636–1659.

Gallego, Guillermo and Garrett Van Ryzin (1997). “A multiproduct dynamic pricing problem and its applications to network yield management”. In: *Operations research* 45.1, pp. 24–41.

Jasin, Stefanus and Sunil Kumar (2012). “A re-solving heuristic with bounded revenue loss for network revenue management with customer choice”. In: *Mathematics of Operations Research* 37.2, pp. 313–345.

Devroye, Luc (1986). *Non-Uniform Random Variate Generation*. New York: Springer-Verlag.

Agrawal, Shipra et al. (2010). “Correlation robust stochastic optimization”. In: *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, pp. 1087–1096.

Mitzenmacher, Michael and Eli Upfal (2017). *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2018). *Foundations of machine learning*. MIT press.

Cornuejols, Gerard, Marshall L Fisher, and George L Nemhauser (1977). “Exceptional paper—location of bank accounts to optimize float: An analytic study of exact and approximate algorithms”. In: *Management science* 23.8, pp. 789–810.

# Appendix A: Chapter 2

## A.1 Useful Known Results

In this section, we provide some concepts and known results that will be used in our algorithms and analyses. We start by defining weighted rank functions for $k$-uniform matroids, which have a close connection to both the expected rewards of our algorithms and our LP benchmarks. We use correlation gaps for this family of functions to relate these two quantities. We also review the dependent rounding scheme developed by Gandhi et al. (2006), which we use both in the implementation of our algorithms and as a tool for analysis. We close by providing a closed-form formula for the expectation of a truncated Poisson random variable that is used throughout the paper.

### A.1.1 Weighted Rank Functions and Correlations Gaps

Let $(w_i)_{i\in[n]}$ be a vector of non-negative weights and $D \subseteq [n]$. Define the weighted rank function for the $k$-uniform matroid $w^* : 2^{[n]} \to [0, \infty)$ as

$$w^*(D) := \max_{R\subseteq D, |R|\leq k} \sum_{i\in R} w_i.$$

Loosely speaking, the reward obtained by our algorithms can be expressed as weighted rank functions of $k$-uniform matroids. Here, $D$ is to be interpreted as the set of candidates eligible for hire, the weights are to be interpreted as the expected reward collected from each candidate conditional on being eligible, and the weighted rank function selects the $k$ highest rewards from the set $D$.

Let $D \sim \mathcal{D}$, with $\mathcal{D}$ a distribution over $2^{[n]}$, and $q_i = \mathbb{P}(i \in D)$. Let $\tilde{D} \sim \mathcal{I}(\mathcal{D})$, where $\mathcal{I}(\mathcal{D})$ is a distribution over $2^{[n]}$ in which each $i \in D$ independently with probability $q_i$. The correlation

gap of a set function $f : 2^{[n]} \to [0, \infty)$ is defined by

$$\sup_{\mathcal{D}} \frac{\mathbb{E}_{D\sim\mathcal{D}}(f(D))}{\mathbb{E}_{D\sim\mathcal{I}(\mathcal{D})}(f(D))}.$$

This concept was first formalized by Agrawal et al. (2010). In words, it quantifies how much we can win by correlating the outcome of $\{i \in D\}$ for $i \in [n]$ while maintaining the marginal probabilities. We use a result from Yan (2011) concerning the correlation gap of weighted rank functions for $k$-uniform matroids.

**Proposition 1 (Lemma 4.4 in Yan (2011))** *For any $n, k \geq 1$, the correlation gap for the weighted rank function of a $k$-uniform matroid of size $n$ is at most* $(1 - \frac{e^{-k}k^k}{k!})^{-1}$.

### A.1.2 GKPS Dependent Rounding

Gandhi et al. (2006) developed a dependent rounding scheme that we use for establishing our results. This dependent rounding scheme receives as an input a bipartite graph $(A, B, E)$ with weights $x_{ij} \in [0, 1]$ for all edges $(i, j) \in E$. The output is, for each edge, a random variable $X_{ij} \in \{0, 1\}$. For any vertex $i \in A$, define the fractional degree $\ell_i = \sum_{j:(i,j)\in E} x_{ij}$. Analogously, for $j \in B$ define $\ell_j = \sum_{i:(i,j)\in E} x_{ij}$. The outputs satisfy the following three properties:

(P1) Marginal distribution: $E(X_{ij}) = x_{ij}$ for every $(i, j) \in E$,

(P2) Degree preservation: with probability 1 it holds that $\sum_{j:(i,j)\in E} X_{ij} \in \{\lfloor \ell_i \rfloor, \lceil \ell_i \rceil\}$ for all $i \in A$ and $\sum_{i:(i,j)\in E} X_{ij} \in \{\lfloor \ell_j \rfloor, \lceil \ell_j \rceil\}$ for all $j \in B$,

(P3) Negative correlation: For any vertex $i \in A \cup B$, any subset $S$ of edges incident in $i$ and any $b \in \{0, 1\}$, it holds that

$$\mathbb{P}\left(\bigcap_{e\in S}\{X_e = b\}\right) \leq \prod_{e\in S}\mathbb{P}(X_e = b).$$

For the special case when the bipartite graph is a star graph (i.e. $A = [n], |B| = 1$) the input are simply weights $x_i \in [0, 1]$ for all $i \in [n]$, and the negative correlation property can be stated as:

for any subset $M \subseteq [n]$ and any $b \in \{0, 1\}$ it holds that

$$\mathbb{P}\left(\bigcap_{i \in M} \{X_i = b\}\right) \leq \prod_{i \in M} \mathbb{P}(X_i = b).$$

### A.1.3 Expectation of Truncated Poisson Random Variable

We provide an elementary calculation used for deriving a closed-form formula for the expectation of a truncated Poisson random variable.

**Proposition 2** *For $k \in \mathbb{N}$, let $Z$ be a Poisson random variable with mean $k$. Then*

$$\mathbb{E}(\min\{Z, k\}) = k\left(1 - e^{-k}\frac{k^k}{k!}\right).$$

*Proof.*

$$\begin{aligned}
\mathbb{E}(\min\{Z, k\}) &= \sum_{j=1}^{k} j\frac{e^{-k}k^j}{j!} + k\left(1 - \sum_{j=0}^{k}\frac{e^{-k}k^j}{j!}\right) \\
&= k + ke^{-k}\left(\sum_{j=1}^{k}\frac{k^{j-1}}{(j-1)!} - \sum_{j=0}^{k}\frac{k^j}{j!}\right) \\
&= k + ke^{-k}\left(\sum_{j=0}^{k-1}\frac{k^j}{j!} - \sum_{j=0}^{k}\frac{k^j}{j!}\right) \\
&= k\left(1 - e^{-k}\frac{k^k}{k!}\right). \square
\end{aligned}$$

## A.2 Proofs of Section 2.4

### A.2.1 Proof of Lemma 1

Consider any algorithm and let $Y_i$ be the indicator of applicant $i$ getting an interview and $X_{ij}$ the indicator of $i$ being hired and $V_i = r_j$. Also let $Q_{ij}$ be the indicator of $V_i = r_j$. We will show that $x_{ij} = \mathbb{E}(X_{ij})$ and $y_i = \mathbb{E}(Y_i)$ is feasible, and that the objective function is equal to the expected reward of the algorithm.

In order to hire an applicant $i$ with $V_i = r_j$, we need to interview the applicant and that the applicant has value $r_j$. This translates to $X_{ij} \leq Y_i Q_{ij}$, and by taking expectation and using that $Y_i$ and $Q_{ij}$ are independent we get constraint (2.1). The algorithm can interview at most $T$ applicants, so $\sum_{i\in[n]} Y_i \leq T$. Again taking expectation we get constraint (2.2). The algorithm can hire at most $k$ applicants, so $\sum_{(i,j)\in[n]\times[J]} X_{ij} \leq k$, by taking expectation we get constraint (2.3). The remaining constraints are clearly satisfied.

Finally, we have that the reward of the algorithm equals $\sum_{(i,j)\in[n]\times[J]} X_{ij} r_j$, so the expected reward of the algorithm is equal to the objective function of the linear program.

### A.2.2 Proof of Lemma 2

The linear program has $Jn + n$ variables and $2Jn + 2n + 2$ constraints. This proof relies on two observations, the first one being that constraints $y_i \geq 0$ and $y_i \leq 1$ cannot be tight simultaneously. The second observation is that constraints $x_{ij} \geq 0$ and $x_{ij} \leq y_i q_{ij}$ cannot be tight simultaneously unless $y_i = 0$. If that is the case, then we have that constraints

$$y_i \geq 0, \quad x_{ij} \geq 0, \quad x_{ij} \leq q_{ij} y_i$$

are tight, but linearly dependent. With these two observations, we conclude that

(i) for each $i \in [n]$, we can only count one of $y_i \geq 0$ and $y_i \leq 1$ as a linearly independent tight constraint,

(ii) for each $(i, j) \in [n] \times [J]$, we can only count one of $x_{ij} \geq 0$ and $x_{ij} \leq q_{ij} y_i$ as a linearly independent tight constraint.

As we need $Jn + n$ linearly independent tight constraints for $x, y$ to be a basic feasible solution and we only have two other constraints, we can drop at most 2 tight constraints out of the ones listed in points (i) and (ii). This shows that at most 2 components of $y$ are different from 0 or 1.

We still need to show that if there are two fractional components, they will add up to 1. This holds because if there are two fractional components, then constraint (2.2) is necessarily tight. This

implies that $T-1$ components of $y$ are equal to 1, as setting $T-2$ or less would contradict the tightness of (2.2), and setting $T$ would contradict the two fractional components. Combining this with the tightness of (2.2) gives

$$T = \sum_{i\in[n]:y_i=1} y_i + \sum_{i\in[n]:0<y_i<1} y_i = T-1+\sum_{i\in[n]:0<y_i<1} y_i,$$

from which we conclude that

$$\sum_{i\in[n]:0<y_i<1} y_i = 1.$$

### A.2.3 Proof of Lemma 3

When taking expectations we get

$$\begin{aligned}\mathbb{E}\left(V_i\mathbb{1}\{N_{i-1}<k\}Z_i\right) &= \mathbb{E}\left(V_i|\mathbb{1}\{N_{i-1}<k\}Z_i=1\right)\mathbb{E}(\mathbb{1}\{N_{i-1}<k\}Z_i)\\ &= v_i\mathbb{E}(\mathbb{1}\{N_{i-1}<k\}Z_i).\end{aligned}$$

This expression holds because $\mathbb{1}\{N_{i-1}<k\}Z_i=1$ is equivalent to applicant $i$ being hired and $v_i$ is the expected value of applicant $i$ given that they were interviewed and the algorithm would decide

that they are hired:

$$\mathbb{E}(V_i|\mathbb{1}\{N_{i-1} < k\}Z_i = 1) = \mathbb{E}(V_i|Z_i = 1) \tag{A.1}$$

$$= \sum_{j=1}^{J} r_j \mathbb{P}(Q_{ij} = 1|Z_i = 1)$$

$$= \sum_{j=1}^{J} r_j \mathbb{P}(Z_i = 1|Q_{ij} = 1)\frac{\mathbb{P}(Q_{ij} = 1)}{\mathbb{P}(Z_i = 1)}$$

$$= \sum_{j=1}^{J} r_j \mathbb{P}(Z_i = 1|Q_{ij} = 1)\frac{q_{ij}}{p_i y_i} \tag{A.2}$$

$$= \sum_{j=1}^{J} r_j \mathbb{P}(Y_i = 1|Q_{ij} = 1)\mathbb{P}(P_i = 1|Q_{ij} = 1)\frac{q_{ij}}{y_i p_i} \tag{A.3}$$

$$= \sum_{j=1}^{J} r_j y_i \mathbb{P}(P_i = 1|Q_{ij} = 1)\frac{q_{ij}}{y_i p_i} \tag{A.4}$$

$$= \sum_{j=1}^{J} r_j y_i \frac{x_{ij}}{y_i q_{ij}}\frac{q_{ij}}{y_i p_i} \tag{A.5}$$

$$= \frac{\sum_{j=1}^{J} r_i x_{ij}}{\sum_{j=1}^{J} x_{ij}} = v_i. \tag{A.6}$$

In equality A.1 line we simply use that the value of an applicant is independent of the remaining positions left when they would be interviewed. In equality A.2, we replaced the know probabilities in the fraction: $\mathbb{P}(Q_{ij} = 1) = q_{ij}$ and $\mathbb{P}(Z_i = 1) = \mathbb{P}(P_i = 1)\mathbb{P}(Y_i = 1) = p_i y_i$ since they are independent. In equality A.3, we use that $Y_i$ and $P_i$ are independent. In equality A.4, we replace $\mathbb{P}(Y_i = 1|Q_{ij} = 1) = \mathbb{P}(Y_i = 1) = y_i$, since $Y_i$ is to be decided before knowing the realization of the values. In equality A.5 we replace $\mathbb{P}(P_i = 1|Q_{ij} = 1) = x_{ij}/(y_i q_{ij})$. In equality A.6, we simply replace the definition of $p_i$ and rearrange the terms to obtain the desired equality.

With this equivalence, we get the following expression for the reward of $\mathsf{ALG}_{\texttt{ptk}}$:

$$\sum_{i=1}^{n} v_i \mathbb{E}(\mathbb{1}\{N_{i-1} < k\}Z_i).$$

### A.2.4 Proof of Lemma 4

The right-hand side of (2.5) can be interpreted as the reward of a policy that samples from $\tilde{Y}$ instead of $Y$ for deciding the applicants that will be interviewed. Notice that $\mathsf{ALG}_{\texttt{ptk}}$ selects a set of $T$ applicants to interview, but the independent analog might select an order of $T+1$ applicants (if $\tilde{Y}_{i_1} = \tilde{Y}_{i_2} = 1$). Define $V := \sum_{i=1}^{n} v_i Z_i \mathbb{1}\{N_{i-1} < k\}$ and $\tilde{V} := \sum_{i=1}^{n} v_i \tilde{Z}_i \mathbb{1}\{\tilde{N}_{i-1} < k\}$. Let $\mathcal{F}$ be the sigma algebra generated by $\{P_i\}_{i\in[n]}$ (i.e. all the randomness except $Y_{i_1}$, $\tilde{Y}_{i_1}$, $Y_{i_2}$ and $\tilde{Y}_{i_2}$, as the remaining $Y_i$ and $\tilde{Y}_i$ are deterministic).

For proving the lemma we show that $\mathbb{E}(V - \tilde{V}|\mathcal{F}) \geq 0$, which implies the result by using the law of iterated expectations. We do this by an exhaustive sample path analysis establishing the inequality for all possible events in $\mathcal{F}$.

**Case 1:** $P_{i_1} = P_{i_2} = 0$**.** The first thing to notice is that if $P_{i_1} = P_{i_2} = 0$, then $V$ and $\tilde{V}$ are identical. Indeed, neither $i_1$ or $i_2$ will be hired, and $Z_i = \tilde{Z}_i$ for the remaining applicants.

**Case 2:** $P_{i_1} = 1, P_{i_2} = 0$**.** If this is the case, then $V$ and $\tilde{V}$ have the same expectation. Indeed, the values of $Y_{i_2}$ and $\tilde{Y}_{i_2}$ will not have any effect on the applicants hired by the algorithm. The only difference in the selection could be made by $Y_{i_1}$ and $\tilde{Y}_{i_1}$, which have the same marginal distribution, so the expectation is equal.

**Case 3:** $P_{i_1} = 0, P_{i_2} = 1$**.** This case is symmetric to Case 2.

**Case 4:** $P_{i_1} = P_{i_2} = 1$**.** To analyze this case, define $\phi := \{i \in [n] \setminus \{i_1, i_2\} : i < i_2, Z_i = 1\}$, so $|\phi|$ is the amount of applicants other than $i_1$ scheduled to get an interview before $i_2$ and would make the cut. Clearly $\phi \in \mathcal{F}$. We further distinguish 3 sub-cases, depending on the number of elements in $\phi$.

*Subcase 4.1:* $|\phi| \geq k$. In this case we have $\mathbb{E}((V - \tilde{V})|\mathcal{F}) = 0$. Indeed, as $|\phi| \geq k$, then no matter what happens with applicant $i_1$, applicant $i_2$ will never get an interview. The only difference can be made by different values of $Y_{i_1}$ and $\tilde{Y}_{i_1}$, whose marginal distributions are identical, so the expectation must also be.

*Subcase 4.2:* $|\phi| = k - 1$. In this subcase, $V$ will select everything in $\phi$ plus either $i_1$ or $i_2$. On the other hand, $\tilde{V}$ might select only $i_1$, only $i_2$, both, or neither. In the last case, the last selected

index in $\tilde{V}$ will be $i' := \inf\{i > i_2 : Z_i = 1\}$. Notice that $i'$ might not exist, in that case, we say $i' = n + 1$ and $v_{n+1} = 0$. This way we write

$$\mathbb{E}(V|P_{i_1} = P_{i_2} = 1, \phi, |\phi| = k - 1) = \sum_{i \in \phi} v_i + y_{i_1} v_{i_1} + y_{i_2} v_{i_2},$$
$$\mathbb{E}(\tilde{V}|P_{i_1} = P_{i_2} = 1, \phi, |\phi| = k - 1) = \sum_{i \in \phi} v_i + y_{i_1} v_{i_1} + (1 - y_{i_1})(y_{i_2} v_{i_2} + (1 - y_{i_2}) v_{i'}),$$

so the difference is

$$\begin{aligned}\mathbb{E}(V - \tilde{V}|P_{i_1} = P_{i_2} = 1, \phi, |\phi| = k - 1) &= y_{i_2} v_{i_2} - y_{i_2}(y_{i_2} v_{i_2} + y_{i_1} v_{i'}) \\ &\geq y_{i_2}(v_{i_2} - v_{i_2}) = 0,\end{aligned}$$

as $y_{i_1} + y_{i_2} = 1$ and $v_{i_2} \geq v_{i'}$.

*Subcase 4.3:* $|\phi| \leq k - 2$. Define $\phi' := \{i \neq i_1, i_2 : \sum_{j=1, j \neq i_1, i_2}^{i} Z_j \leq k - 2\}$ to be the indices of applicants that will be certainly selected both in $V$ and in $\tilde{V}$. Define $j_1 := \inf\{i \in [n] \setminus \{i_1, i_2\} : Z_i = 1, \sum_{j=1, j \neq i_1, i_2}^{i} Z_j = k - 1\}$. If the set defining $j_1$ is empty we again say $j_1 = T + 1$. Notice that $j_1 \geq i_2$ because $|\phi| \leq k - 2$. Applicant $j_1$ will always be selected by $V$, as it will select the $k - 2$ elements in $\phi'$, either $i_1$ or $i_2$, and $j_1$, which will be the next best applicant remaining. On the other hand, $\tilde{V}$ will also select all applicants in $\phi'$, but the remaining two positions can be filled in more ways. It can be the case that $\tilde{V}$ picks either $i_1$ or $i_2$ plus $j_1$, but it also might happen that it picks both $i_1$ and $i_2$ and not pick $j_1$, or it can also pick neither of $i_1$ and $i_2$ and pick instead $j_1$ plus

another another which we define as $j_2 := \inf\{i > j_1 : Z_i = 1\}$. With this in mind, we can express

$$\begin{aligned}
\mathbb{E}(V|P_{i_1} = P_{i_2} = 1, \phi', |\phi| \leq k-2) &= \sum_{i\in\phi'} v_i + y_{i_1}v_{i_1} + y_{i_2}v_{i_2} + v_{j_1}, \\
\mathbb{E}(\tilde{V}|P_{i_1} = P_{i_2} = 1, \phi', |\phi| \leq k-2) &= \sum_{i\in\phi'} v_i + y_{i_1}y_{i_2}(v_{i_1} + v_{i_2}) \\
&\quad + (y_{i_1})^2(v_{i_1} + v_{j_1}) + (y_{i_2})^2(v_{i_2} + v_{j_1}) + y_{i_1}y_{i_2}(v_{j_1} + v_{j_2}) \\
&= \sum_{i\in\phi'} v_i + y_{i_1}v_{i_1} + y_{i_2}v_{i_2} + y_{i_1}v_{j_1} + y_{i_2}v_{j_2}.
\end{aligned}$$

The difference is

$$\mathbb{E}(V - \tilde{V}|P_{i_1} = P_{i_2} = 1, \phi', |\phi| \leq k-2) = v_{j_1} - y_{i_1}v_{j_1} - y_{i_2}v_{j_2} \geq v_{j_1} - v_{j_1} = 0,$$

### A.2.5 Proof of Theorem 1

To show the result we first show how to express the expected reward of the independent version of our algorithm, $\mathsf{ALG}'_{\texttt{ptk}}$, and the objective function of $\mathrm{LP}_{\texttt{ptk}}$ as expectations of weighted rank functions.

For the first one, we use the interpretation of the algorithm collecting value $v_i$ when applicants make the cut. Let $M = \{i : \tilde{Z}_i = 1\}$. As the algorithm will interview applicants in decreasing order of $v_i$, we can express the expected reward as $R_{\mathsf{ALG}'_{\texttt{ptk}}} = v^*(M)$. Notice that $\mathbb{P}(i \in M) = \mathbb{E}(Z_i) = y_i p_i$.

For expressing the objective function of the linear program as a weighted rank function we rewrite

$$\sum_{i=1}^{n}\sum_{j=1}^{J} r_{ij}x_{ij} = \sum_{i=1}^{n} y_i \frac{\sum_{j=1}^{J} r_{ij}x_{ij}}{\sum_{j=1}^{J} x_{ij}} \frac{\sum_{j=1}^{J} x_{ij}}{y_i} = \sum_{i=1}^{n} y_i v_i p_i.$$

This way, we can use the dependent rounding procedure presented in Gandhi et al. (2006) to

generate indicators $W_i$ with probabilities $w_i = y_i p_i$. Constraint (2.3) can be re-written as

$$\sum_{i=1}^{n}\sum_{j=1}^{J} x_{ij} = \sum_{i=1}^{n} y_i \sum_{j=1}^{J} \frac{x_{ij}}{y_i} = y_i p_i \leq k,$$

so the dependent rounding procedure ensures that $D = \{i \in [n] : W_i = 1\}$ has cardinality at most $k$. With this, we can use the linearity of the expectation to write

$$\sum_{i=1}^{n} v_i y_i p_i = \sum_{i=1}^{n} v_i \mathbb{P}(W_i = 1) = \max_{R \subseteq D, |R| \leq k} \sum_{i \in R} v_i = v^*(D).$$

The second equality holds because $D$ has a cardinality of at most $k$.

With these expressions, for any instance $I \in \mathcal{I}_{\texttt{ptk}}$ we can write

$$\frac{R_{\mathsf{ALG}'_{\texttt{ptk}}}(I)}{\mathrm{LP}_{\texttt{ptk}}(I)} = \frac{v^*(M)}{v^*(D)} \geq 1 - \frac{e^{-k}k^k}{k!}.$$

The inequality follows from Proposition 1 by Yan (2011), since $M$ and $D$ have the same marginal distributions.

The last step is to invoke Lemma 4 for the inequality $R_{\mathsf{ALG}_{\texttt{ptk}}}(I) \geq R_{\mathsf{ALG}'_{\texttt{ptk}}}(I)$.

### A.2.6 Tightness of Guarantee

We provide an instance that shows that the guarantee in Theorem 1 is tight. Indeed, construct the instance with $n$ applicants with i.i.d valuations. Each applicant will have $V_i = 1$ with probability $k/n$ or $V_i = 0$ with probability $1 - k/n$. The implied support of the valuations' distributions is $\{r_0, r_1\} = \{0, 1\}$. There is no time constraint for this algorithm, i.e. $T = n$. Denote the described instance by $I_{UB}^{\texttt{ptk}}$.

**Proposition 3** *For any $\pi \in \Pi^{\texttt{ptk}}$,*

$$R_\pi(I_{UB}^{\texttt{ptk}}) \leq \left(1 - \frac{e^{-k}k^k}{k!}\right) \mathrm{LP}_{\texttt{ptk}}(I_{UB}^{\texttt{ptk}}).$$

*Proof.* An optimal solution for $\mathrm{LP}_{\mathtt{ptk}}(I^{\mathtt{ptk}}_{UB})$ will have $y_i = 1$, $x_{i1} = k/n$ and $x_{i0} = 0$ for all $i$, and attain an objective value of $k$. On the other hand, an optimal algorithm will interview all applicants one by one, until $k$ of them have value 1 and are hired or there are no applicants left to interview. The expected reward of the algorithm will be $\mathbb{E}(\min\{k, B_{n,k}\})$, where $B_{n,k} \sim Bin(n, k/n)$. If we make $n$ go to infinity, the expected reward will converge to

$$\mathbb{E}(\min\{\text{Poisson}(k), k\}) = k\left(1 - \frac{e^{-k}k^k}{k!}\right)$$

by Proposition 2 in Appendix A.1.3. □

It is worth noting that this tightness is only with respect to $\mathrm{LP}_{\mathtt{ptk}}$. Indeed, $\mathsf{ALG}_{\mathtt{ptk}}$ is optimal when applied to $I^{\mathtt{ptk}}$.

### A.2.7 Proof of Lemma 5

Let $x, y$ be an optimal solution of $\mathrm{LP}_{\mathtt{ptk}}$. Construct $\tilde{x}, \tilde{y}$ with $\tilde{x}_i = x_i$ and $\tilde{y}_i = x_i/q_i$ for all $i \in [n]$. It is clear to see that $\tilde{x}, \tilde{y}$ is feasible and that the value of the objective function remains unchanged.

## A.3 Proofs of Section 2.5

### A.3.1 Proof of Lemma 6

For an arbitrary policy let $Y_{ij}$ be the indicator that candidate $i$ receives an offer for position $j$. Let $P_{ij}$ be the indicator that candidate $i$ would accept an offer for position $j$ should they receive one.

Any policy satisfies that for each position $j$, at most $T$ offers can be sent. In terms of the indicators:

$$\sum_{i=1}^{n} Y_{ij} \leq T,$$

so constraint (2.6) follows by taking expectation. Similarly, each candidate can receive at most one

offer:

$$\sum_{j=1}^{k} Y_{ij} \leq 1,$$

so constraint (2.8) follows by expectation. Next, for each position $j$, at most one candidate can be hired. In terms of our indicators:

$$\sum_{i=1}^{n} Y_{ij} P_{ij} \leq 1,$$

so constraint (2.7) is obtained by taking expectation and using that $P_{ij}$ and $Y_{ij}$ are independent. Constraints of the form $0 \leq y_{ij} \leq 1$ are clearly satisfied for all $i \in [n]$ and $j \in [k]$.

Finally, the reward collected by the policy is

$$\sum_{j=1}^{k} \sum_{i=1}^{n} v_{ij} Y_{ij} P_{ij}.$$

Again by taking expectation and using the independence of $Y_{ij}$ and $P_{ij}$ we obtain that the expected reward of the policy is equal to the objective function of the linear program.

### A.3.2 Proof of Lemma 7

For simplicity assume that $v_{1,j} \geq v_{2,j} \geq \cdots \geq v_{n,j}$. We can write

$$v_j^*(D) = v_{1,j} + \sum_{i=2}^{n} (v_{i-1,j} - v_{i,j}) \mathbb{1}\{\cap_{\ell=1}^{i-1} \{\ell \in D\}).$$

By taking expectation we get

$$\mathbb{E}(v_j^*(D)) = v_{1,j} + \sum_{i=2}^{n} (v_{i,j} - v_{(i+1),j}) \mathbb{P}(\cap_{\ell=1}^{i-1} \{\ell \in D\}).$$

Now, using the negative correlation property (P3) together with (P1) and $(v_{i-1,j} - v_{i,j}) \leq 0$ we can conclude

$$\begin{aligned}
\mathbb{E}(v_j^*(D)) &\geq v_{1,j} + \sum_{i=2}^{n} (v_{i,j} - v_{(i+1),j}) \prod_{\ell=1}^{i-1} \mathbb{P}(\{\ell \in D\}) \\
&= v_{1,j} + \sum_{i=2}^{n} (v_{i,j} - v_{(i+1),j}) \prod_{\ell=1}^{i-1} \mathbb{P}(\{\ell \in \tilde{D}\}) \\
&= \mathbb{E}(v_j^*(\tilde{D})).
\end{aligned}$$

### A.3.3 Counterexample for Lemma 7 if $k \geq 2$

**Example 2** *Consider four elements a, b, c and d, all with identical weights equal to 1. Consider the following rounding scheme: pick any subset from* $\{\{a\}, \{b\}, \{c\}, \{d\}, \{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}\}$ *with equal probability. This implies that any element will be included in the set of eligible elements D with probability 1/2. It is not hard to see that this rounding scheme satisfies (P3). Indeed, for subsets of cardinality 1, the right-hand side of (P3) is 1/2. The left-hand side is also 1/2: it can be obtained by choosing the corresponding singleton, plus either of the three subsets of cardinality 3 that contain the element in question. By the same simple counting argument, the property will also be satisfied with equality for subsets of sizes 2 and 3. For the full set, however, the right-hand side of (P3) will be 1/16, while the left-hand side will be 0.*

*The expected reward collected by choosing the* $k = 2$ *highest weights with this rounding scheme is 3/2: with probability 1/2 we will have one element in the subset, and with probability 1/2 we will have three elements in the subset, of which we can only choose 2. However, if we formed the eligible subset by including each of the elements independently with probability 1/2, the reward collected would be*

$$\mathbb{E}(\min\{B_{4,1/2}, 2\}) = 1 \cdot \frac{4}{16} + 2 \cdot (\frac{6}{16} + \frac{4}{16} + \frac{1}{16}) = \frac{13}{8} > \frac{3}{2},$$

*where* $B_{4,1/2}$ *refers to a Binomial random variable with 4 independent trials of probability 1/2.*

### A.3.4 Proof of Lemma 8

Let $W_{ij}$ be the output of the Gandhi et al. (2006) dependent rounding with input $y_{ij}p_{ij}$. These indicators satisfy $\mathbb{E}(W_{ij}) = y_{ij}p_{ij}$ and $\sum_{i=1}^{n} W_{ij} \leq 1$ with probability 1, since $y$ satisfies constraint (2.7). Let $M_j = \{i : W_{ij} = 1\}$. We can then express

$$L_j^* = \sum_{i=1}^{n} y_{ij}p_{ij} = \mathbb{E}(\max_{i \in M} v_{ij}) = \mathbb{E}(v_j^*(M)).$$

The lemma follows from Proposition 1, since $v_j^*$ is a weighted rank function for a 1-uniform matroid, and $\tilde{D}$ has the same marginal distributions as $M$, but the elements are sampled independently.

### A.3.5 Tightness of Guarantee

In this section, we show that the guarantee obtained in Theorem 2 is tight. Consider an instance with $k$ identical positions and $kT$ identical candidates. In particular, $v_{ij} = 1$ and $p_{ij} = 1/T$ for all $(i, j) \in [kT] \times [k]$. Denote this instance by $I_{UB}^{\texttt{par}}$.

**Proposition 4** *For any $\pi \in \Pi^{\texttt{par}}$,*

$$R_\pi(I_{UB}^{\texttt{par}}) \leq \left(1 - \frac{1}{e}\right) \mathrm{LP}_{\texttt{par}}(I_{UB}^{\texttt{par}})$$

*Proof.* A feasible solution for $\mathrm{LP}_{\texttt{par}}$ is to set $y_{ij} = 1/k$ for all $(i, j) \in [kT] \times [k]$, achieving an objective of $k$. On the other hand, since all candidates are identical, forming $k$ parallel lists with $T$ candidates each, and running each list in parallel is clearly optimal. The reward collected by a single list is

$$\mathbb{E}(\min\{B_{T,1/T}, 1\}) \to_{T\to\infty} \mathbb{E}(\min\{\text{Poisson}(1), 1\}) = 1 - \frac{1}{e},$$

where $B_{T,1/T} \sim Bin(T, 1/T)$. The last equality is obtained by applying Proposition 2 in Appendix A.1.3. Since there are $k$ identical lists, the ratio between the performance of the algorithm and the linear program is exactly $(1 - 1/e)$. □

### A.3.6 Proof of Lemma 9

Let $y$ be a solution for $\mathrm{LP}_{\mathtt{seq}}$. Let us construct $y'$, a solution for $\mathrm{LP}_{\mathtt{par}}$. In particular, we let $y'_{ij} = y_i/k$ for all $(i, j) \in [n] \times [k]$. It is straightforward to see that the solution is feasible and that the values of the objective functions are equal, so $\mathrm{LP}_{\mathtt{par}}(I) \geq \mathrm{LP}_{\mathtt{seq}}(I')$. For the other direction, let $z$ be a solution for $\mathrm{LP}_{\mathtt{par}}$. We can construct $z'$ by setting $z'_i = \sum_{j=1}^{k} z_{ij}$. It is again straightforward that this solution is feasible in $\mathrm{LP}_{\mathtt{seq}}$ and that the values of the objective functions are equal, so so $\mathrm{LP}_{\mathtt{par}}(I) \leq \mathrm{LP}_{\mathtt{seq}}(I')$.

## A.4 Proofs and Complementary Material of Section 2.6

### A.4.1 Computation of the Optimal Value-ordered Policy

In this section, we explain how to obtain the optimal value-ordered policy for the Simultaneous Offering problem. The first step is to compute the distribution of the number of candidates that accept an offer, when the set of candidates that receive an offer is $S = \{1, \ldots, m\}$, for all possible values of $m \in [n]$. For this, we use a dynamic program. For $j \leq m$, let $Q(j, m)$ be the probability that $j$ candidates accept an offer when candidates 1 through $m$ get offers. We can obtain these values with $O(n^2)$ computations by solving the following system of equations using recursion:

$$
\begin{aligned}
Q(1, 1) &= p_1 \\
Q(0, 1) &= 1 - p_1 \\
Q(j, m+1) &= p_{m+1} V(j-1, m) + (1 - p_{m+1}) V(j, m) \qquad \forall m \in [n-1], \quad j \leq m+1.
\end{aligned}
$$

Now let $V(m)$ denote the expected reward for sending offers to candidates in $S = \{1, \ldots, m\}$. We can now compute

$$
U(m) = \sum_{i=1}^{m} v_i p_i - \sum_{j=k+1}^{m} (j - k) \cdot Q(j, m).
$$

The last step is to pick $m^* \in \arg\max\{U(m) : m \in [n]\}$. Thus, we can find the optimal value-ordered policy in runtime $O(n^2)$.

### A.4.2 Example Illustrating Expected-value-ordered Policies' Bad Performance

**Example 3** *Consider an instance with $k = 1$ positions to fill and $n + 1$ candidates. Candidate 1 has $v_1 = 1/n$ and $p_1 = 1$. Candidates $i = 2, \ldots, n+1$ have $v_i = 1 - 1/n$ and $p_i = 1/n$. Candidate 1 has a higher expected value than any other candidate, so any expected-value-ordered policy must include her. Since $p_1 = 1$, including any other candidate can only decrease the collected reward, so the optimal expected-value-ordered policy is to only send an offer to candidate 1 and obtain a reward of $1/n$ which vanishes as n grows large. On the other hand, a valid policy is to send an offer to all candidates in $\{2, \ldots, n+1\}$. Let $Z \sim Bin(n, 1/n)$. The expected reward of this policy is*

$$
\begin{aligned}
& n\left(1-\frac{1}{n}\right)\frac{1}{n} - \mathbb{E}\left[\max\{Z-1, 0\}\right] \\
=& \left(1-\frac{1}{n}\right) - \sum_{i=2}^{n}(i-1)\mathbb{P}(Z=i) \\
=& \left(1-\frac{1}{n}\right) - \left[\sum_{i=2}^{n} i\mathbb{P}(Z=i) - \sum_{i=2}^{n}\mathbb{P}(Z=i)\right] \\
=& \left(1-\frac{1}{n}\right) - \left[(\mathbb{E}(Z) - \mathbb{P}(Z=1)) - (1 - \mathbb{P}(Z=1) - \mathbb{P}(Z=0))\right] \\
=& 1 - \frac{1}{n} - \left(1-\frac{1}{n}\right)^{n} \underset{n\to\infty}{\to} 1 - \frac{1}{e}.
\end{aligned}
$$

*This shows that for n large enough, the ratio between the optimal expected-value-ordered policy and another policy can be made arbitrarily close to 0. Note that the greedy policy would also add candidate 1 first, since candidate 1 has the highest marginal benefit when the offer set is initially empty, hence the greedy policy would also perform as poorly as expected-value-ordered policies.*

### A.4.3 Proof of Lemma 10

By noting that $f(x) = \max\{x, a\}$ for some constant $a$ is a convex function, we can apply Jensen's inequality to bound the objective function in $\mathsf{OPT}_{\texttt{sim}}$. For any policy, let $Y_i$ denote the indicator that candidate $i$ receives an offer, and let $y_i = \mathbb{E}(Y_i)$. Similarly, let $P_i$ be the indicator that candidate $i$ would accept an offer if they receive one, with $\mathbb{E}(P_i) = p_i$. We can write the expected

reward obtained by the policy as

$$\mathbb{E}\left(\sum_{i\in[n]} v_i Y_i P_i - \max\left\{\sum_{i\in[n]} Y_i P_i - k, 0\right\}\right)$$

$$\leq \sum_{i\in[n]} v_i \mathbb{E}(Y_i P_i) - \max\left\{\sum_{i\in[n]} \mathbb{E}(Y_i P_i) - k, 0\right\} \tag{A.7}$$

$$= \sum_{i\in[n]} v_i p_i y_i - \max\left\{\sum_{i\in[n]} p_i y_i - k, 0\right\} \tag{A.8}$$

$$= \sum_{i\in[n]} v_i p_i y_i + \min\left\{k - \sum_{i\in[n]} p_i y_i, 0\right\} \tag{A.9}$$

where in inequality A.7 we use Jensen's inequality and in equation A.8 we use the fact that $Y_i$ and $P_i$ are independent. We know that for any instance $I \in \mathcal{I}_{\texttt{sim}}$, and any policy $\pi \in \Pi^{\texttt{sim}}$, $R_\pi(I)$ is upper-bounded by the expression in equation A.9, so by optimizing this expression over randomized policies, represented by $\{y_i\}_{i\in[n]}$, we obtain an upper bound on $\mathsf{OPT}_{\texttt{sim}}(I)$. We can represent this optimization problem as a linear program by introducing auxiliary variable $z \in \mathbb{R}$, which will take the value of $\min\left\{k - \sum_{i\in[n]} p_i y_i, 0\right\}$. This can be easily accomplished by requiring $z \leq 0$ and $z \leq k - \sum_{i\in[n]} p_i y_i$. Since we are maximizing and $z$ appears with a positive sign in equation A.9, one of the two mentioned constraints will always be binding.

### A.4.4 Proof of Lemma 12

We prove both statements by guessing a solution and showing that any other solution with the structure given in Lemma 11 achieves either the same or a worse objective value. Notice that this solution structure implies that the whole solution is completely determined by its total mass. In all cases we can assume that $z = \min\{0, k - \sum_{i\in[n]} y_i p_i\}$.

We start with the case $P_{\texttt{high}} > k$. Let $j$ be the last index of $V_{\texttt{high}}$. This is the same as saying that $j+1$ is the first index satisfying $v_{j+1} < 1$. We can modify the guessed solution in two ways: by either increasing or decreasing the total mass. This first translates to either decreasing $y_j$ or

increasing $y_{j+1}$. Since our guessed solution satisfies $\sum_{i\in[n]} y_i p_i = P_{\texttt{high}} > k$, then $z > 0$. If we decrease $y_j$ by $\delta > 0$, the change in the objective function is upper-bounded by $\delta p_j(1 - v_j) \leq 0$, so the objective does not improve. The same will happen with each index $i < j$, the change in the objective function will be upper-bounded by $\delta p_i(1 - v_i) \leq 0$, so reducing the mass of our guessed solution cannot improve the objective value. On the other hand, if we increase $y_{j+1}$ by $\delta > 0$, because $z > 0$ and it will not decrease, the change in the objective function will be $\delta p_{j+1}(v_{j+1} - 1) \leq 0$. If we further increase indices $i > j + 1$ by $\delta > 0$, the change in the objective value will be $\delta p_i(v_i - 1) \leq 0$. This way, cannot improve the objective value of our guessed solution by increasing its total mass and is therefore optimal.

The case when $P_{\texttt{high}} < k$ follows the same logic. First, if $P_{\texttt{total}} < k$, then we can make $y_i = 1$ for all $i \in [n]$ without incurring in any cost (since the total mass will always be less than $k$ and therefore $z = 0$). If $P_{\texttt{total}} > k$, then our guessed solution will have mass $k$ and $z = 0$. Let $j$ be the last index in our guessed solution such that $y_j > 0$. If we reduce the mass of the solution by $0 < \delta \leq y_j$, we will lose $p_j y_j$ and we will not reduce any cost since $z$ is already 0. The same will happen if we further decrease the mass from indices $i < j$. If we increase the mass of the solution by $\delta < 1 - y_j$, then $z$ will increase by the same amount and the change in the objective value will be $\delta p_j(v_j - 1) \leq 0$. The same happens if we further increase the mass. We conclude that our guessed solution is also optimal in this case.

### A.4.5 Proof of Lemma 13

Let $y$ be an optimal solution of $\text{LP}_{\texttt{sim}}(I)$ and $y'$ be the alternative solution constructed by $\mathsf{ALG}^s_{\texttt{sim}}$. Let $M = \sum_{i\in[n]} y_i p_i$, which is at most $k$ by Lemma 12. By construction, $\sum_{i\in[n]} y'_i p_i = sM$. Let $\{D_i\}_{i=1}^n$ and $\{D'_i\}_{i=1}^n$ be collections of independent Bernoulli random variables with parameters $y_i p_i$ and $y'_i p_i$, respectively. Since $\mathsf{ALG}^s_{\texttt{sim}}$ independently sends offers to candidates with

probabilities $y'_i$, we interpret $D'_i$ as the indicator that candidate $i$ is hired. We can write

$$\begin{aligned}
R_{\mathsf{ALG}^s_{\texttt{sim}}}(I) &= \sum_{i\in[n]} v_i \mathbb{E}(D'_i) - \mathbb{E}\left(\max\left\{\sum_{i\in[n]} D'_i - k, 0\right\}\right) \\
&= \sum_{i\in[n]} v_i \mathbb{E}(D'_i) - \sum_{i\in[n]} \mathbb{E}(D'_i) + \mathbb{E}\left(\min\left\{\sum_{i\in[n]} D'_i, k\right\}\right) \\
&\geq \sum_{i\in[n]} v_i \mathbb{E}(D'_i) - \sum_{i\in[n]} \mathbb{E}(D'_i) + \frac{\mathbb{E}\left(\min\left\{\mathrm{Pois}(sM), k\right\}\right)}{sM} \sum_{i\in[n]} \mathbb{E}(D'_i) \\
&= \sum_{i\in[n]} v_i \mathbb{E}(D'_i) + \sum_{i\in[n]} \mathbb{E}(D'_i) \left(-1 + \frac{\mathbb{E}\left(\min\left\{\mathrm{Pois}(sM), k\right\}\right)}{sM}\right).
\end{aligned}$$

In the inequality, we use a folklore splitting argument (see, e.g., Lemma 4.2 in Yan (2011)) saying that if $\sum_{i\in[n]} \mathbb{E}(D'_i) \leq sM$, then

$$\frac{\mathbb{E}\left(\min\{\sum_{i\in[n]} \mathbb{E}(D'_i), k\}\right)}{\sum_{i\in[n]} \mathbb{E}(D'_i)} \geq \frac{\mathbb{E}(\min\{\mathrm{Pois}(sk), k\})}{sM}.$$

The splitting argument uses the fact that $\mathbb{E}[\min\{\mathrm{Pois}(x), k\}]/x$ is decreasing in $x$. With this last fact combined with Lemma 12, which gives us that $M \leq k$, we can further bound

$$R_{\mathsf{ALG}^s_{\texttt{sim}}}(I) \geq \sum_{i\in[n]} v_i \mathbb{E}(D'_i) + \sum_{i\in[n]} \mathbb{E}(D'_i) \left(-1 + \frac{\mathbb{E}\left(\min\left\{\mathrm{Pois}(sk), k\right\}\right)}{sk}\right). \tag{A.10}$$

Since $\sum_{i\in V_{\texttt{high}}} p_i \leq k$, Lemma 12 gives us that $z = 0$, thus we can write

$$\mathrm{LP}_{\texttt{sim}}(I) = \sum_{i\in[n]} v_i y_i p_i = \sum_{i\in[n]} v_i \mathbb{E}(D_i).$$

By combining this with equation A.10 we can further bound

$$\begin{aligned}
\frac{R_{\mathsf{ALG}^s_{\texttt{sim}}}(I)}{\mathrm{LP}_{\texttt{sim}}(I)} &\geq \frac{\sum_{i\in[n]} v_i \mathbb{E}(D'_i)}{\sum_{i\in[n]} v_i \mathbb{E}(D_i)} + \frac{\sum_{i\in[n]} \mathbb{E}(D'_i)}{\sum_{i\in[n]} v_i \mathbb{E}(D_i)} \left(-1 + \frac{\mathbb{E}\left(\min\left\{\mathrm{Pois}(sk), k\right\}\right)}{sk}\right) \\
&\geq s + \frac{\sum_{i\in[n]} D'_i}{\sum_{i\in[n]} v_i D_i} \left(-1 + \frac{\mathbb{E}\left(\min\left\{\mathrm{Pois}(sk), k\right\}\right)}{sk}\right) \qquad \text{(A.11)} \\
&\geq s + \frac{\sum_{i\in[n]} D'_i}{\tau \sum_{i\in[n]} D_i} \left(-1 + \frac{\mathbb{E}\left(\min\left\{\mathrm{Pois}(sk), k\right\}\right)}{sk}\right) \qquad \text{(A.12)} \\
&= s + \frac{s}{\tau} \left(-1 + \frac{\mathbb{E}\left(\min\left\{\mathrm{Pois}(sk), k\right\}\right)}{sk}\right). \qquad \text{(A.13)}
\end{aligned}$$

In inequality A.11 we use that $v_i$ are decreasing in $i$, and since the solution $y'$ is constructed by truncating $y$, so

$$\frac{\sum_{i\in[n]} v_i \mathbb{E}(D'_i)}{\sum_{i\in[n]} v_i \mathbb{E}(D_i)} \geq \frac{\sum_{i\in[n]} \mathbb{E}(D'_i)}{\sum_{i\in[n]} \mathbb{E}(D_i)} = s.$$

In inequality A.12, we use that

$$-1 + \frac{\mathbb{E}\left(\min\left\{\mathrm{Pois}(sk), k\right\}\right)}{sk} \leq -1 + \frac{\mathbb{E}\left(\mathrm{Pois}(sk)\right)}{sk} = 0,$$

so the whole second term is non-positive and can only be reduced by bounding $v_i$'s by $\tau$ in the denominator. In equality A.13, we simply use the construction of $y'$ which scales the total mass of $y$ by a factor $s$.

### A.4.6 Proof of Lemma 14

Let $I = (k, n, p, v) \in \mathcal{I}^{\tau}_{\texttt{sim}}$ be such that $\sum_{i\in V_{\texttt{high}}} p_i = m > k$. Construct $I^{\texttt{alt}} = (k, n, p^{\texttt{alt}}, v)$ by setting $p_i^{\texttt{alt}} = p_i k/m$. This way $\sum_{i\in V_{\texttt{high}}} p_i^{\texttt{alt}} = k$. We first show that for both of these instances, $s^* = 1$. Since the optimal solutions of both linear programs will only have positive components for candidates in $V_{\texttt{high}}$, we can without loss of generality assume that the instances only contain the candidates in $V_{\texttt{high}}$. This can be easily seen by inspecting the expression

$$s - \frac{s}{\tau} + \frac{\mathbb{E}[\min\{\mathrm{Pois}(sk), k\}]}{\tau k}.$$

Since $\tau > 1$, the expression is clearly increasing in $s$, thus setting $s^* = 1$ is optimal.

Now let $D_i$ be Bernoulli variables with parameters $p_i$ and $D_i^{\mathtt{alt}}$ be Bernoulli random variables with parameters $p_i^{\mathtt{alt}}$. For the original instance, we can write

$$R_{\mathsf{ALG}^1_{\mathtt{sim}}}(I) = \sum_{i \in V_{\mathtt{high}}} v_i p_i - \mathbb{E}\left(\max\{\sum_{i \in V_{\mathtt{high}}} D_i - k, 0\}\right)$$

and

$$\mathrm{LP}_{\mathtt{sim}}(I) = \sum_{i \in V_{\mathtt{high}}} v_i p_i - (m - k).$$

For the alternative instance, we can also write

$$R_{\mathsf{ALG}^1_{\mathtt{sim}}}(I^{\mathtt{alt}}) = \sum_{i \in V_{\mathtt{high}}} v_i p_i \frac{k}{m} - \mathbb{E}\left(\max\{\sum_{i \in V_{\mathtt{high}}} D_i^{\mathtt{alt}} - k, 0\}\right)$$

and

$$\mathrm{LP}_{\mathtt{sim}}(I^{\mathtt{alt}}) = \sum_{i \in V_{\mathtt{high}}} v_i p_i \frac{k}{m}.$$

We will show that

$$R_{\mathsf{ALG}^1_{\mathtt{sim}}}(I) - R_{\mathsf{ALG}^1_{\mathtt{sim}}}(I^{\mathtt{alt}}) \geq \mathrm{LP}_{\mathtt{sim}}(I) - \mathrm{LP}_{\mathtt{sim}}(I^{\mathtt{alt}}), \tag{A.14}$$

which will imply the result.

Equation (A.14) is equivalent to

$$m - k \geq \mathbb{E}\left(\max\{\sum_{i \in V_{\mathtt{high}}} D_i - k, 0\}\right) - \mathbb{E}\left(\max\{\sum_{i \in V_{\mathtt{high}}} D_i^{\mathtt{alt}} - k, 0\}\right).$$

The right-hand side of this inequality can be re-written as

$$\left[\sum_{i\in V_{\texttt{high}}} \mathbb{E}(D_i) + \mathbb{E}\left(\max\{-k, -\sum_{i\in V_{\texttt{high}}} D_i\}\right)\right] - \left[\sum_{i\in V_{\texttt{high}}} \mathbb{E}(D_i^{\texttt{alt}}) + \mathbb{E}\left(\max\{-k, -\sum_{i\in V_{\texttt{high}}} D_i^{\texttt{alt}}\}\right)\right]$$
$$= m - k + \left[\mathbb{E}\left(\max\{-k, -\sum_{i\in V_{\texttt{high}}} D_i\}\right) - \mathbb{E}\left(\max\{-k, -\sum_{i\in V_{\texttt{high}}} D_i^{\texttt{alt}}\}\right)\right].$$

Finally, $\mathbb{E}\left(\max\{-k, -\sum_{i\in V_{\texttt{high}}} D_i^{\texttt{alt}}\}\right)$ stochastically dominates $\mathbb{E}\left(\max\{-k, -\sum_{i\in V_{\texttt{high}}} D_i\}\right)$, so

$$\mathbb{E}\left(\max\{-k, -\sum_{i\in V_{\texttt{high}}} D_i\}\right) - \mathbb{E}\left(\max\{-k, -\sum_{i\in V_{\texttt{high}}} D_i^{\texttt{alt}}\}\right) \leq 0,$$

which concludes the proof.

### A.4.7 Proof of Theorem 4

Recall that the instance has $n = j^2 + k$ candidates, with $j$ being a large integer. The first $j^2$ candidates are of type 1 and have $p_i = k/j$ and $v_i = \tau + \delta$, where $\delta = \varepsilon\tau/j$ is small. The remaining $k$ candidates are of type 2, who have $p_i = 1$ and $v_i = \tau$.

We first note that a feasible policy is to send offers to all type 2 candidates, which yields a reward of $k\tau$. Thus, $\mathsf{OPT}_{\texttt{sim}}(I) \geq k\tau$.

Now we show that an optimal value-ordered policy will never send offers to candidates of type 2 and that the number of offers sent to candidates of type 1 cannot be very large with respect to $j$. To make this formal, consider a value-ordered policy that sends offers only to candidates of type 1. The policy must decide the amount of type 1 candidates who are going to receive offers. Let $m_j$ be the optimal number of offers to be sent to these type of candidates. Notice that $0 \leq m_j \leq j^2$. Define $s_j := m_j/j$, $s = \lim_{j\to\infty} s_j$. We will show that $s < \infty$, since having $s_j \to \infty$ as $j \to \infty$ would imply that the policy has a reward that diverges to $-\infty$. Let $A_j$ denote the event $\{\text{Bin}(m_j, k/j) > k\}$. The reward of a policy that sends offers to $m_j$ type 1 candidates when there

are $j^2$ of them available is

$$\begin{aligned}
\mathsf{ALG}_{s_j} &= (\tau+\delta)\mathbb{E}(\text{Bin}(m_j, k/j)) + \mathbb{E}(\min\{0, k - \text{Bin}(m_j, k/j)\}) \\
&= (\tau+\delta)k\frac{m_j}{j} + \mathbb{E}(\min\{0, k - \text{Bin}(m_j, k/j)\}) \qquad \text{(A.15)} \\
&= (\tau+\delta)k\frac{m_j}{j} + \mathbb{E}(k - \text{Bin}(m_j, k/j)|A_j)\mathbb{P}(A_j) \\
&\leq k + (\tau+\delta)k\frac{m_j}{j} - \mathbb{E}(\text{Bin}(m_j, k/j))\mathbb{P}(A_j) \\
&= k + k\frac{m_j}{j}(\tau+\delta-\mathbb{P}(A_j)).
\end{aligned}$$

In the inequality we use the fact that we are subtracting the Binomial random variable in the expectation, so removing the conditional $\{\text{Bin}(m_j, k/j) > k\}$ can only make our expression larger. Now suppose that $m_j/j \to \infty$ as $j$ grows to infinity. That implies that the mean of $\text{Bin}(m_j, k/j)$ also diverges, so $\mathbb{P}(A_j) = \mathbb{P}(\text{Bin}(m_j, k/j) > k)$ grows to 1. This in turn means that for $j$ large enough, $(\tau+\delta-\mathbb{P}(A_j)) < 0$, so the expected reward is upper-bounded by a sequence that diverges to $-\infty$. This has two implications: first, an optimal value-ordered policy will never send offers to candidates of type 2, since they can only add at most a constant reward for being included. Second, for upper-bounding the expected reward of any value-ordered policy in this instance, we can assume without loss of generality that $s < \infty$.

To establish our upper bound let us resume from Equation (A.15) and develop

$$\begin{aligned}
\mathsf{ALG}_{s_j} &= (\tau+\delta)k\frac{m_j}{j} + \mathbb{E}(\min\{0, k - \text{Bin}(m_j, k/j)\}) \\
&= (\tau+\delta)k\frac{m_j}{j} - \mathbb{E}(\text{Bin}(m_j, k/j)) + \mathbb{E}(\min\{k, \text{Bin}(m_j, k/j)\}) \qquad \text{(A.16)} \\
&= \tau\frac{m_j k}{j} + \frac{\varepsilon k\tau m_j}{j^2} - \frac{m_j k}{j} + \mathbb{E}(\min\{k, \text{Bin}(m_j, k/j)\}) \qquad \text{(A.17)} \\
&\leq \varepsilon k\tau + k\frac{m_j}{j}(\tau-1) + \mathbb{E}(\min\{k, \text{Bin}(m_j, k/j)\}) \qquad \text{(A.18)} \\
&\underset{j\to\infty}{\to} \varepsilon k\tau + \tau k s - sk + \mathbb{E}(\min\{\text{Pois}(sk), k\}). \qquad \text{(A.19)}
\end{aligned}$$

In the equation A.16, we rearrange terms. In the equation A.17, we replace the definition of $\delta$, the expectation of the Binomial random variable, and rearrange terms. In equation A.18, we use the fact that $m \leq j^2$ and $\varepsilon > 0$. In the equation A.19 we use the fact that $\lim_{j\to\infty} s_j < \infty$, so $\text{Bin}(m, k/j)$ converges in distribution to $\text{Pois}(sk)$ and $\min\{x, k\}$ is a continuous bounded function.

Putting all together, we obtain that

$$\frac{\mathsf{ALG}_s}{\mathsf{OPT}_{\mathtt{sim}}(I)} \leq \varepsilon + s - \frac{s}{\tau} + \frac{\mathbb{E}(\min\{\text{Pois}(sk), k\})}{k\tau}.$$

The proof is concluded by taking the supremum over $s$.

### A.4.8 Tightness Analysis

In this section, we provide conditions for our derived bounds to be tight. Notice that the only difference between the optimization problems defining $\alpha_k^\tau$ and $\beta_k^\tau$ is that the former has the constraint $s \leq 1$. Neither set of constants has a closed-form value, but since their objective functions are concave, we can characterize their optimal solutions using first-order conditions.

**Proposition 5** *For $\tau \in (0, 1)$, let $s_\alpha^*$ and $s_\beta^*$ be optimal solutions for $\alpha_k^\tau$ and $\beta_k^\tau$, respectively. Then $s_\alpha^* = \min\{s_\beta^*, 1\}$ and $s_\beta^*$ is the solution of*

$$\tau = 1 - e^{-sk} \sum_{j=0}^{k-1} \frac{(sk)^j}{j!}.$$

*Proof.* We first prove the characterization of $s_\beta^*$. For that, we show that the objective function of the optimization problems is concave. Let us develop:

$$\begin{aligned} s - \frac{s}{\tau} + \frac{\mathbb{E}[\min\{\text{Pois}(sk), k\}]}{k\tau} &= s - \frac{s}{\tau} + \frac{1}{k\tau}\left[\sum_{j=0}^{k-1} j\frac{e^{-sk}(sk)^j}{j!} + k\left(1 - \sum_{j=0}^{k-1}\frac{e^{-sk}(sk)^j}{j!}\right)\right] \\ &= s - \frac{s}{\tau} + \frac{1}{\tau} - \frac{1}{k\tau}\sum_{j=0}^{k-1}(k-j)\frac{e^{-sk}(sk)^j}{j!}. \end{aligned}$$

By taking derivative with respect to $s$ we get

$$
\begin{aligned}
&1-\frac{1}{\tau}-\frac{1}{k\tau}\sum_{j=0}^{k-1}\frac{(k-j)}{j!}\left[\frac{je^{-sk}(sk)^j}{s}-ke^{-sk}(sk)^j\right]\\
=&1-\frac{1}{\tau}-\frac{e^{-sk}}{\tau}\sum_{j=1}^{k-1}(k-j)\frac{(sk)^{j-1}}{(j-1)!}+\frac{e^{-sk}}{\tau}\sum_{j=0}^{k-1}(k-j)\frac{(sk)^j}{(j)!}\\
=&1-\frac{1}{\tau}-\frac{e^{-sk}}{\tau}\sum_{j=0}^{k-2}(k-(j+1))\frac{(sk)^j}{j!}+\frac{e^{-sk}}{\tau}\sum_{j=0}^{k-1}\frac{(k-j)}{(j)!}(sk)^j\\
=&1-\frac{1}{\tau}-\frac{e^{-sk}}{\tau}\sum_{j=0}^{k-2}(k-(j+1))\frac{(sk)^j}{j!}+\frac{e^{-sk}}{\tau}\sum_{j=0}^{k-1}\frac{(k-j)}{(j)!}(sk)^j\\
=&1-\frac{1}{\tau}+\frac{e^{-sk}}{\tau}\sum_{j=0}^{k-2}\frac{(sk)^j}{j!}-\left[\frac{e^{-sk}}{\tau}\sum_{j=0}^{k-2}(k-j)\frac{(sk)^j}{j!}-\frac{e^{-sk}}{\tau}\sum_{j=0}^{k-1}\frac{(k-j)}{(j)!}(sk)^j\right]\\
=&1-\frac{1}{\tau}+\frac{1}{\tau}\sum_{j=0}^{k-1}e^{-sk}\frac{(sk)^j}{j!}.
\end{aligned}
\tag{A.20}
$$

The second derivative of the objective function is

$$
\begin{aligned}
&\frac{1}{\tau}\left[\sum_{j=0}^{k-1}\frac{je^{-sk}(sk)^j}{j!s}-\sum_{j=0}^{k-1}ke^{-sk}(sk)^j\right]\\
=&\frac{1}{\tau}\left[\sum_{j=1}^{k-1}\frac{e^{-sk}(sk)^j}{j!s}-\sum_{j=0}^{k-1}\frac{e^{-sk}(sk)^{j+1}}{(j-1)!s}\right]\\
=&\frac{1}{\tau}\left[\sum_{j=0}^{k-2}\frac{e^{-sk}(sk)^{j+1}}{j!s}-\sum_{j=0}^{k-1}\frac{e^{-sk}(sk)^{j+1}}{j!s}\right]\\
=&-\frac{s^{k-1}e^{-sk}k^k}{\tau(k-1)!}\le 0
\end{aligned}
$$

for all $s \ge 0$, establishing the concavity. Also observe that Equation (A.20) is positive for $s$ close to 0, so either the problem is unbounded, or first-order conditions will imply optimality. It is easy to see that the problem is not unbounded. This is because $\mathbb{E}(\min\{\text{Pois}(sk), k\}) \le k$ and $\tau < 1$, so with $s$ large enough the objective function will become negative. We can conclude that the optimal value $s^*_\beta$ is the solution of making Equation (A.20) equal to 0.

The form of $s^*_\alpha$ also comes from the fact that the objective function is concave. Thus, it will either satisfy the first-order condition in the $[0, 1]$ interval, or it will be weakly increasing in the $[0, 1]$ interval. In both cases the result holds. □

The form of $s^*_\alpha$ is implied by the concavity of the objective function combined with the constraint that $s \leq 1$. We characterize the region where $s^*_\beta \leq 1$, in which $s^*_\alpha = s^*_\beta$ and our guarantee is tight.

**Proposition 6** *For any $k \in \mathbb{N}$ and $\tau \in (0, 1)$ we have we have $s^*_\alpha = s^*_\beta$ if and only if*

$$\tau \leq 1 - e^{-k} \sum_{j=0}^{k-1} \frac{k^j}{j!}.$$

*This condition holds for all $k$ if $\tau \leq 1/2$.*

*Proof.* Proposition 5 implies that $s^*_\alpha = s^*_\beta$ if and only if $s^*_\beta \leq 1$. We will show that the condition of Proposition 6 is equivalent to $s^*_\beta \leq 1$.

To show this we again resort to the concavity of the objective function in $\beta^\tau_k$. Since the function is concave, we have that $s^*_\beta \leq 1$ if and only if the derivative in Equation (A.20) non-positive at $s = 1$. Indeed, if it is non-positive, concavity implies that the first-order condition is satisfied and some $s' \leq 1$. On the other hand, if the derivative is positive at $s = 1$, then the first-order condition is satisfied at some $s' > 1$ and $s^*_\beta > s^*_\alpha$. This establishes the first statement of the lemma.

For the second statement, we see that

$$1 - \sum_{j=0}^{k-1} \frac{e^{-k} k^j}{j!} = \mathbb{P}(\text{Pois}(k) \geq k) \geq \frac{1}{2},$$

since $k$ is an integer (Mitzenmacher and Upfal 2017, Exercise 5.14). □

Figure A.1 gives a graphic representation of $\alpha^\tau_k$ and $\beta^\tau_k$ for several values of $k$ and all $\tau \in (0, 1)$. As stated in proposition 6, it can be shown that for all values of $\tau < 1/2$, the bound is tight. It can also be observed that as $k$ grows, the guarantee becomes better, and that the gap between the upper and lower bounds becomes smaller.

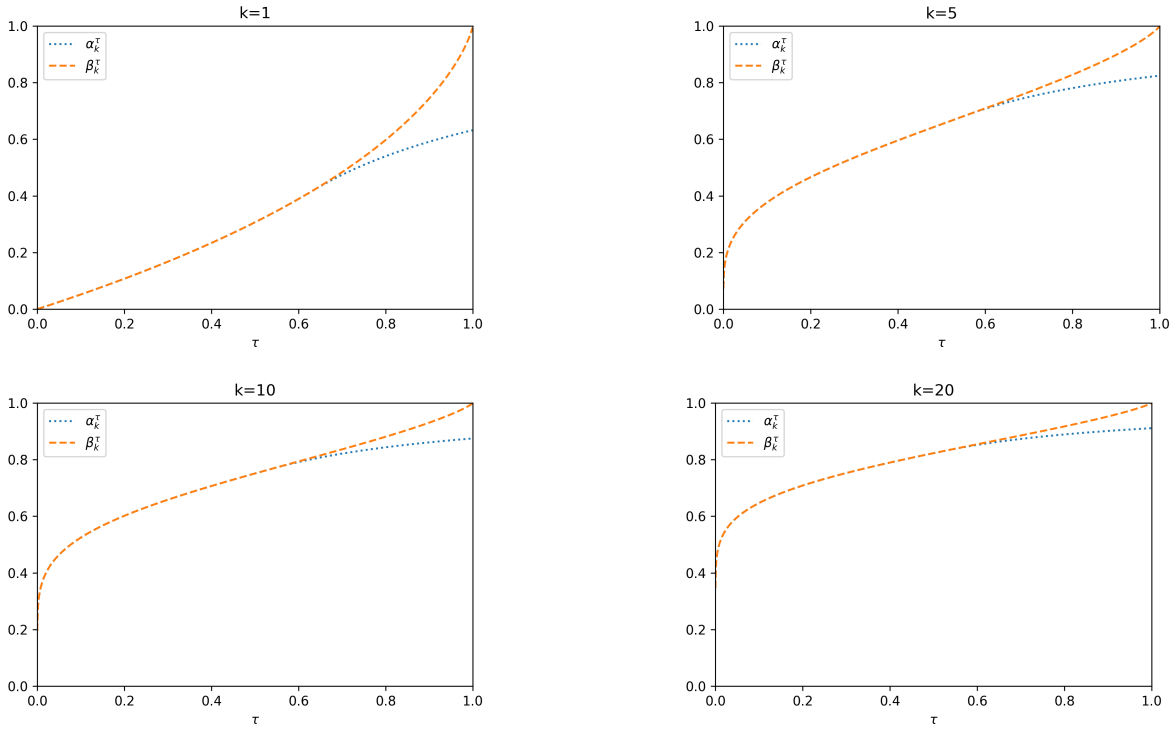


Figure A.1: Values of $\alpha_k^\tau$ and $\beta_k^\tau$ for $\tau \in (0, 1)$ and $k \in \{1, 5, 10, 20\}$.

## A.5 Numerical Study

In this section, we provide numerical experiments that complement our theory and allow us to compare the rewards obtained under the three different models of hiring: `seq`, `par` and `sim`.

Our experimental setting is based on the one by Purohit, Gollapudi, and Raghavan (2019). In our experiments, we sample 50 randomly generated pools of $n = 100$ candidates. As in Purohit, Gollapudi, and Raghavan (2019), we generate candidates such that there is a *negative correlation* between each candidate's value and their probability of acceptance, motivated by there being more market competition for the high-valued candidates. We also provide results where the sampling of values and acceptance probabilities is independent.

The candidate pools are sampled in the following way. For each candidate $i$, $v_i$ is sampled independently from a Uniform(0,1) distribution. In the 'negative correlation' setting the acceptance probability of candidate $i$, $p_i$, is sampled from a Beta$(10(1 - v_i), 10v_i)$ distribution. In the 'independent' setting, $p_i$ is sampled from a Uniform(0,1) distribution, independent of $v_i$. This very

specific sampling procedure is the one used in Purohit, Gollapudi, and Raghavan 2019, and as such we acknowledge that the insights we obtain in these experiments do not necessarily hold in general. We choose to sample 50 candidate pools and average out the performance of the different heuristics and benchmarks across these pools. This helps to smooth out idiosyncratic choices made by algorithms in specific instances that may be misleading.

### A.5.1 Implemented Policies

We implement four heuristics for `seq` and one heuristic for `par`. The first sequential heuristic is $\mathsf{ALG}'_{\texttt{seq}}$, a slight variant of $\mathsf{ALG}_{\texttt{seq}}$ which is de-randomized, and in the case that the original algorithm would send less than $T$ offers, it fills those gaps with candidates with high $v_i$. The second heuristic was introduced by Purohit, Gollapudi, and Raghavan (2019) and we call it 'Adaptive sequential'. It corresponds to the optimal *adaptive* algorithm that *sends offers in decreasing order of* $v_i$. Since any non-adaptive algorithm can be improved by sending offers in decreasing order of $v_i$, Adaptive sequential serves as an upper bound for the performance of any non-adaptive algorithm, including our $\mathsf{ALG}_{\texttt{seq}}$. The other two heuristics that we implement for `seq` are what we call the naive non-adaptive policies: 'Value-ordered' ($\mathsf{VO}_{\texttt{seq}}$) and 'Expected-value-ordered' ($\mathsf{EO}_{\texttt{seq}}$). These policies greedily send offers to the next candidate with the highest $v_i$ and $v_i p_i$, respectively. For reference, we include the value of $\mathrm{LP}_{\texttt{seq}}$. For `par` we implement $\mathsf{ALG}'_{\texttt{par}}$, a de-randomized heuristic based on $\mathsf{ALG}_{\texttt{par}}$.

For `sim` we implement three heuristics. The first one is 'Value-ordered' ($\mathsf{VO}_{\texttt{sim}}$), corresponding to the optimal value-ordered policy that sends offers to the first $m$ candidates with the highest $v_i$ (and $m$ is optimized). Similarly, 'Expected-value-ordered' ($\mathsf{EO}_{\texttt{sim}}$) corresponds to the optimal policy that sends offers to the first $m$ candidates with the highest $v_i p_i$ (and $m$ is optimized). We also implement a greedy heuristic and include the value of $\mathrm{LP}_{\texttt{sim}}$ for reference.

A detailed description of the implementation of these policies can be found in Appendix A.5.4.

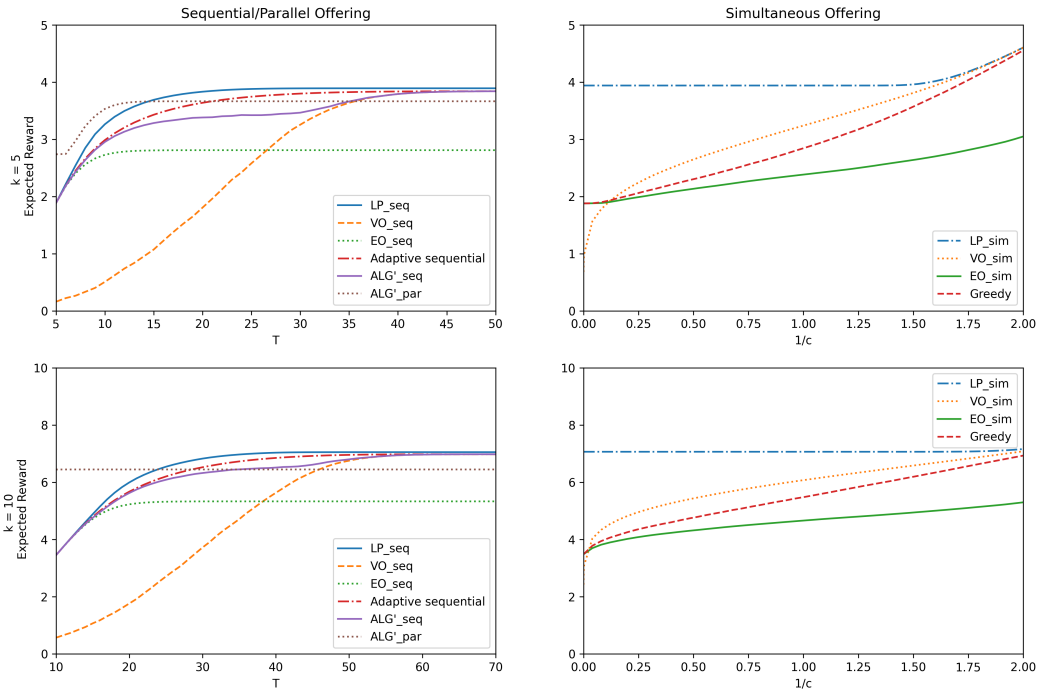


Figure A.2: Numerical results for negatively correlated $v_i$ and $p_i$, averaged over the 50 random candidate pools.

### A.5.2 Results for the Negative Correlation Setting

Figure A.2 shows the results obtained for the implemented heuristics. The left column shows how the expected reward changes as the number of offering rounds $T$ increases in the sequential and parallel offering settings. The right column shows how the expected reward changes as $1/c$ varies in the simultaneous setting. Recall that $c$ is the penalty for hiring each candidate over capacity $k$. We make the $x$-axis $1/c$ instead of $c$ so that the plots are more comparable to the sequential/parallel setting. The top row shows the results for $k = 5$ positions and the bottom row shows the results for $k = 10$ positions.

**Insights about Sequential and Parallel Offering.**

The plots for `seq` and `par` show how the expected reward of different heuristics grows as $T$ increases.

*Difficult values of T for sequential hiring.* A first observation is that intermediate values of $T$ are harder to approximate, both for the adaptive heuristic and the non-adaptive heuristics. This can be concluded by observing a higher gap between the heuristics and the benchmark upper bounds. It comes with no surprise, since extreme values of $T$ have trivial solutions. On one hand, if $T = n$, then the optimal policy is to send offers to all candidates in decreasing order of $v_i$. If $T = k$, then the optimal policy is to send offers to the $k$ candidates with the highest $v_i p_i$. Consequently, $\mathsf{VO}_{\texttt{seq}}$ is (near-)optimal when $T$ is large, and $\mathsf{EO}_{\texttt{seq}}$ is (near-)optimal when $T$ is small. The intermediate values produce more separation in the performance of different algorithms.

*Virtue of our non-adaptive sequential policy.* We observe that $\mathsf{ALG}'_{\texttt{seq}}$ outperforms both naive policies for all values of $T$. Given the constraints in $\mathrm{LP}_{\texttt{seq}}$, our heuristic $\mathsf{ALG}'_{\texttt{seq}}$ will behave like Expected-value-ordered when $T = k$. There is a value of $T$ where the constraint $\sum_{i\in[n]} y_i \leq T$ stops binding, so the LP solution would remain constant for higher values of $T$, and the output of our algorithm would make less than $T$ offers. At this point, $\mathsf{ALG}'_{\texttt{seq}}$ includes candidates in decreasing order of $v_i$ until $T$ offers are to be sent, and the policy starts behaving like $\mathsf{VO}_{\texttt{seq}}$. This non-adaptive policy based on our LP correctly transitions from $\mathsf{EO}_{\texttt{seq}}$ to $\mathsf{VO}_{\texttt{seq}}$ as $T$ grows, and strictly dominates them on the difficult, intermediate values of $T$.

*More positions help the heuristics.* As $k$ becomes larger, our heuristics achieve better performance with respect to the available benchmarks: a smaller gap can be observed between $\mathsf{ALG}'_{\texttt{seq}}$ and either the Adaptive heuristic or LP upper bound. A smaller gap can also be observed between the Adaptive heuristic and the LP upper bound. This is consistent with our theoretical results, where the approximation guarantee of our algorithm grows with $k$.

*Parallel vs. Sequential Offering.* There is value in the ability to send more offers by sending them in parallel, and this value increases with $k$. This is natural since parallel algorithms are allowed to send as many offers as positions available, so the more available positions in the begin-

ning, the more potential offers there are to be sent. In fact, when $k$ is small (there are two or three positions) the firm can gain more value by using a sequential and adaptive policy than by using a parallel, non-adaptive policy, even with the same amount of allowed offering rounds $T$. There is a point where $T$ becomes large enough such that the Adaptive heuristic, $\mathsf{ALG}'_{\texttt{seq}}$ and $\mathsf{VO}_{\texttt{seq}}$ outperform our parallel heuristic. This is because our parallel heuristic balances out the good candidates along $k$ lists, so when a candidate accepts (and no other candidate in their list receives an offer) many good candidates are left out. With $T$ large enough, the sequential policies (except $\mathsf{EO}_{\texttt{seq}}$) are hedged against this because they have enough time to send offers to all the good candidates. An interesting direction is to design parallel algorithms that adaptively choose the next candidates.

**Insights about Simultaneous Offering.**

In the plots for `sim` we show how the expected reward of the implemented heuristics as $1/c$ grows.[1]

*Best policy depends on c.* For small values of $c$, $\mathsf{VO}_{\texttt{sim}}$ outperforms Greedy and $\mathsf{EO}_{\texttt{sim}}$, while for large values of $c$, the opposite occurs. This is natural since a very large value of $c$ is analogous to having a hard constraint of $k$ on the number of candidates hired. As $c$ becomes smaller, $\mathsf{VO}_{\texttt{sim}}$ becomes better since there is room for error, and sending offers to candidates with high value with the risk of going over capacity becomes more valuable than hiring lower value candidates with a lower chance of not going over capacity. There is a value of $c$ low enough (namely $\min_{i\in[n]} v_i$) when all policies agree to send an offer to every candidate $i \in [n]$.

*Larger k helps Value-ordered.* We see that if $k = 10$, $\mathsf{VO}_{\texttt{sim}}$ outperforms the other two heuristics even for larger values of $c$. This can be explained by the fact that when $k = 10$, there is more margin to capture all the high-value candidates without too much risk of going over capacity.

*Simultaneous vs. Sequential Offering.* This experiment allows us to compare which values of $T$ and $c$ achieve a comparable expected reward. For instance, if $k = 10$, $\mathsf{ALG}_{\texttt{seq}}$ and $\mathsf{VO}_{\texttt{seq}}$ with

[1]The choice of showing the results with respect to $1/c$ instead of $c$ is because it allows showing large values of $c$ while keeping the scale of the graph reasonable. We can use $1/c \in (0, 2]$ to concisely show $c \in [1/2, \infty)$, a range that allows us to compare the performance of $\mathsf{VO}_{\texttt{sim}}$ against the performance of the sequential heuristics.

$T = 30$ time steps achieve rewards comparable to $\mathsf{VO}_{\mathtt{sim}}$ and Greedy in the simultaneous setting with $c \approx 0.6$. A finer mapping from $T$ to $c$ can be done if one fixes a specific policy for each setting. Note that as $c \to \infty$, $\mathsf{EO}_{\mathtt{sim}}$ converges to the same value as $\mathsf{EO}_{\mathtt{seq}}$ when only $T = k$ offers can be sent, as having a large value of $c$ is equivalent to $k$ being a hard constraint.

### A.5.3 Results for the Independent Setting

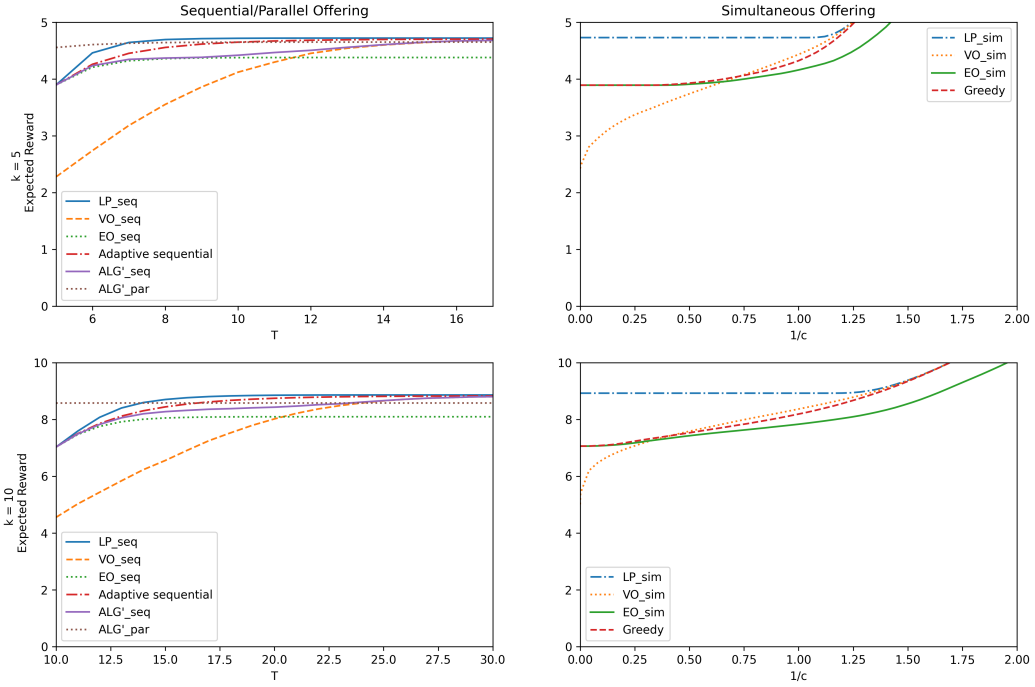


Figure A.3: Numerical results for independent $v_i$ and $p_i$, averaged over the 50 random candidate pools. The left column shows how the expected reward changes as the number of offering rounds $T$ increases in the sequential and parallel offering settings. The right column shows how the expected reward changes as $1/c$ varies in the simultaneous setting.[2]The top row shows the results for $k = 5$ positions and the bottom row shows the results for $k = 10$ positions.

Figure A.3 shows the results obtained for the numerical experiments in the setting where $v_i$ and $p_i$ are independently generated (instead of negatively correlated).

Many of the insights of the negative correlation case also appear here: larger $k$ helps all heuristics, intermediate values of $T$ are the most difficult to approximate (sequential) and the optimal

simultaneous policy depends on the value of $c$. However, some differences can be observed. These differences are mainly explained by the fact that the independent sampling of $p_i$ and $v_i$ generates more high-value, high-probability candidates, as opposed to the negative correlation case where fewer 'unanimous' candidates –who will indisputably receive an offer– appear.

*Sequential - Less time required for optimality.* A smaller time horizon $T$ is required for $\mathsf{ALG}'_{\texttt{seq}}$ and $\mathsf{VO}_{\texttt{seq}}$ to achieve nearly the same performance as the upper bound benchmarks. This can be explained by the existence of more high-value, high-probability candidates, who will fill up the positions and all policies will agree to send offers to them. This also explains the fact that $\mathsf{ALG}'_{\texttt{seq}}$ only slightly outperforms $\mathsf{EO}_{\texttt{seq}}$ for $k = 5$ and small values of $T$.

*Simultaneous - Value-ordered performs worse.* It can be seen that $\mathsf{VO}_{\texttt{seq}}$ performs poorly with respect to the benchmarks, only outperforming Greedy in a slim range of $c$. This can again be explained by the existence of more high-value, high-probability candidates. Since $\mathsf{VO}_{\texttt{seq}}$ defines a threshold, in order to include all the high-value, high-probability candidates it must also include high-value, low-probability candidates, who will make the policy pay a higher penalty.

### A.5.4 Detailed Description of Heuristics

**Sequential Heuristics**

- $\mathsf{ALG}'_{\texttt{seq}}$: This heuristic is based on $\mathsf{ALG}_{\texttt{seq}}$. It first solves $\mathrm{LP}_{\texttt{seq}}$ and forms the two possible resulting lists of candidates that DR could have as an outcome. Call these lists $L_1$ and $L_2$. If any list $L_j$ contains less than $T$ candidates, we fill out the remaining slots with candidates from $[n] \setminus L_j$ in decreasing order of $v_i$. After both lists contain $T$ candidates, the policy evaluates the expected reward obtained by sending offers to the candidates of each list in decreasing order of $v_i$ and selects the list with the highest reward.

- Adaptive Sequential: This policy by Purohit, Gollapudi, and Raghavan (2019) corresponds to the optimal adaptive policy, out of all the adaptive policies that send offers in decreasing order of $v_i$. Specifically, it solves the following dynamic program. Let $S(i, \ell, t)$ be the

expected reward of hiring at most $\ell$ candidates in $t$ time steps by only considering candidates in $\{i, \ldots, n\}$. The optimal value is given by $S(1, k, T)$ and the recursion solved is

$$S(i, \ell, t) = \max\{p_i(v_i + S(i+1, \ell-1, t-1) + (1-p_i)S(i+1, \ell, t-1), S(i+1, \ell, t)).\}$$

- Value-ordered ($\mathsf{VO}_{\mathtt{seq}}$): This policy greedily chooses the next candidate to send an offer to, selecting the one with the highest $v_i$ out of the remaining candidates.
- Expected-value-ordered ($\mathsf{EO}_{\mathtt{seq}}$): This policy greedily chooses the next candidate to send an offer to, selecting the one with the highest $v_i p_i$ out of the remaining candidates.

**Parallel Heuristic**

- $\mathsf{ALG}'_{\mathtt{par}}$: This policy first solves $\mathrm{LP}_{\mathtt{seq}}$ with $kT$ time periods instead of $T$. Two possible pools are constructed (the two possible outcomes of DR), and if they contain less than $kT$ candidates they are filled in the same fashion as in $\mathsf{ALG}'_{\mathtt{seq}}$. For each of the pools, we construct $k$ different lists as follows. Initiate the lists empty and initiate $M_j = 0$. For each of the candidates in the pool, in decreasing order of $v_i$, assign Candidate $i$ to the list $j$ that contains less than $T$ candidates and has the lowest $M_j$, and update $M_j \leftarrow M_j + p_i$. The idea is to spread out the high-value candidates among the lists while maintaining a balanced expected number of acceptances for each list. Once both sets of lists are constructed, the policy evaluates both of them and chooses the one with the highest expected value.

**Simultaneous Heuristics**

- Value-ordered ($\mathsf{VO}_{\mathtt{sim}}$): Given that the candidates are labeled such that $v_1 \geq v_2 \geq \cdots \geq v_n$, this policy chooses the optimal $m \in \{1, \ldots, n\}$ such that all candidates in $\{1, \ldots, m\}$ receive offers.
- Expected-value-ordered ($\mathsf{EO}_{\mathtt{sim}}$): Given that the candidates are labeled such that $v_1 p_1 \geq v_2 p_2 \geq \cdots \geq v_n p_n$, this policy chooses the optimal $m \in \{1, \ldots, n\}$ such that all candidates

in $\{1, \ldots, m\}$ receive offers.

- Greedy: This policy starts with an empty solution and iteratively adds the candidate that adds the highest marginal expected reward to the solution. It stops at the point that adding any candidate would reduce the expected reward obtained.

# Appendix B: Chapter 3

## B.1 Proofs of Section 3.3 and Section 3.4

### B.1.1 Proof of Lemma 15

For brevity, we remove index $i$ from the proof. In particular, we will use $x$ to denote the LP variable for inventory in the $i$-th warehouse, rather than the complete vector. Before proving Lemma 15, we prove the following auxiliary claim.

**Claim 1**

$$\sum_{j\in[T]} \max\left\{0, \frac{y_j - x^f}{1-x^f}\right\} \le \lfloor x \rfloor.$$

*Proof.* Let $S := \{j : y_j \ge x^f\}$, $V := \sum_{j\in S} y_j$, and $V^f := V - \lfloor V \rfloor$. We have

$$\sum_{j\in[T]} \max\left\{0, \frac{y_j - x^f}{1-x^f}\right\} = \sum_{j\in S} \frac{y_j - x^f}{1-x^f} = \frac{V^f + \lfloor V \rfloor - |S| x^f}{1-x^f} \le \frac{V^f + \lfloor V \rfloor - \lceil V \rceil x^f}{1-x^f}, \quad (\ddagger)$$

since $|S| \ge \lceil V \rceil$. We distinguish 2 cases:

Case 1: $V^f = 0$. In this case, $\lceil V \rceil = \lfloor V \rfloor$, so the right-hand side of ($\ddagger$) is equal to $\lfloor V \rfloor \le \lfloor x \rfloor$.

Case 2: $V^f > 0$. In this case, $\lceil V \rceil = \lfloor V \rfloor + 1$ and the right-hand side of ($\ddagger$) is equal to

$$\lfloor V \rfloor + \frac{V^f - x^f}{1-x^f}.$$

We further distinguish two subcases.

Subcase 2.1: $V \ge \lfloor x \rfloor$. Here, $V^f \le x^f$ because $V \le x$. Therefore, the right-hand side of ($\ddagger$) is at most $\lfloor V \rfloor \le \lfloor x \rfloor$.

Subcase 2.2: $V < \lfloor x \rfloor$. In this case, $\lfloor V \rfloor \leq \lfloor x \rfloor - 1$, so the right-hand side of (‡) is at most

$$\lfloor x \rfloor - 1 + \frac{1 - x^f}{1 - x^f} = \lfloor x \rfloor. \square$$

*Proof of Lemma 15.* The result is trivial if $\sum_{j \in [T]} y_j \leq \lfloor x \rfloor$. Indeed, we can simply set $y_j^{\mathrm{L}} = y_j^{\mathrm{H}} = y_j$ for all $j \in [T]$. Therefore, assume $\sum_{j \in [T]} y_j > \lfloor x \rfloor$. We propose the following solution: for all $j \in [T]$ set

$$y_j^{\mathrm{L}} = \max \left\{ \frac{y_j - x^f}{1 - x^f}, 0, y_j - \lambda^* \right\} \text{ and } y_j^{\mathrm{H}} = \frac{y_j - (1 - x^f) y_j^{\mathrm{L}}}{x^f},$$

where $\lambda^* \in [0, 1]$ is such that $\sum_{j \in [T]} y_j^{\mathrm{L}} = \lfloor x \rfloor$. We first show the existence of such $\lambda^*$. Define

$$\Delta(\lambda) := \sum_{j \in [T]} y_j^{\mathrm{L}} = \sum_{j \in [T]} \max \left\{ \frac{y_j - x^f}{1 - x^f}, 0, y_j - \lambda \right\}.$$

Clearly $\Delta(\lambda)$ is a continuous decreasing function. We have by assumption that

$$\Delta(0) \geq \sum_{j \in [T]} y_j > \lfloor x \rfloor.$$

On the other hand, by Claim 1 it holds that

$$\Delta(1) = \sum_{j \in [T]} \max \left\{ 0, \frac{y_j - x^f}{1 - x^f} \right\} \leq \lfloor x \rfloor.$$

Therefore, the Intermediate Value Theorem says that there exists $\lambda^* \in [0, 1]$ such that $\Delta(\lambda^*) = \lfloor x \rfloor$.

We now show that our proposed solution satisfies Equations (3.6) to (3.9). Equations Equa-

tions (3.6) and (3.8) are satisfied by definition. To see that Equation (3.9) is verified, see that

$$\sum_{j\in[T]} y_j^{\mathrm{H}} = \frac{\sum_{j\in[T]} y_j - (1-x^f)\sum_{j\in[T]} y_j^{\mathrm{L}}}{x^f} = \frac{\sum_{j\in[T]} y_j - (1-x^f)\lfloor x\rfloor}{x^f}$$
$$\leq \frac{x - (1-x^f)\lfloor x\rfloor}{x^f} = \frac{x^f + \lfloor x\rfloor - (1-x^f)\lfloor x\rfloor}{x^f} = 1 + \lfloor x\rfloor.$$

To see that Equation (3.7) is satisfied, first notice that $y_j^{\mathrm{L}} \geq 0$ for all $j \in [T]$. We also have that $y_j^{\mathrm{L}} \leq y_j$, which combined with Equation (3.6) implies that $y_j^{\mathrm{L}} \leq y_j^{\mathrm{H}}$. Finally, for all $j \in [T]$ we have

$$y_j^{\mathrm{H}} = \frac{y_j - (1-x^f)y_j^{\mathrm{L}}}{x^f} \leq \frac{y_j - (y_j - x^f)}{x^f} = 1.\square$$

### B.1.2 Proof of Lemma 16

Before proving Lemma 16, we prove the following claim. This allows to transfer the negative correlation property of $(W_i)_{i\in[n]}$ across $i$ to the preliminary assignments $(Y_{ij})_{i\in[n]}$, for each demand node $j \in [T]$.

**Claim 2** *For any $j \in [T]$ and for any subset $S \subseteq [n]$, it holds that*

$$\mathbb{P}\left(\bigcap_{i\in S} Y_{ij} = 0\right) \leq \prod_{i\in S} \mathbb{P}(Y_{ij} = 0).$$

*Proof.* Fix $j \in [T]$. We can write

$$Y_{ij} = W_i Y_{ij}^{\mathrm{H}} + (1-W_i)Y_{ij}^{\mathrm{L}} = Y_{ij}^{\mathrm{H}} - (1-W_i)(Y_{ij}^{\mathrm{H}} - Y_{ij}^{\mathrm{L}}),$$

where $Y_{ij}^{\mathrm{H}}$ and $Y_{ij}^{\mathrm{L}}$ are Bernoulli random variables with means $y_{ij}^{\mathrm{H}}$ and $y_{ij}^{\mathrm{L}}$, respectively, independent across $i$, and coupled such that $Y_{ij}^{\mathrm{L}} \leq Y_{ij}^{\mathrm{H}}$ with probability 1 (we can do this because $y_{ij}^{\mathrm{L}} \leq y_{ij}^{\mathrm{H}}$).

Hence,

$$\begin{aligned}
\mathbb{P}\left(\bigcap_{i\in S} Y_{ij} = 0\right) &= \mathbb{E}\left[\prod_{i\in S}(1 - Y_{ij})\right] \\
&= \mathbb{E}\left[\prod_{i\in S}(1 - Y_{ij}^{\mathrm{H}} + (1 - W_i)(Y_{ij}^{\mathrm{H}} - Y_{ij}^{\mathrm{L}}))\right] \\
&= \sum_{T\subseteq S} \mathbb{E}\left[\prod_{i\in T}(1 - Y_{ij}^{\mathrm{H}}) \prod_{i\in S\setminus T}(1 - W_i)(Y_{ij}^{\mathrm{H}} - Y_{ij}^{\mathrm{L}})\right] \\
&= \sum_{T\subseteq S} \prod_{i\in T}(1 - y_{ij}^{\mathrm{H}}) \prod_{i\in S\setminus T}(y_{ij}^{\mathrm{H}} - y_{ij}^{\mathrm{L}}) \mathbb{E}\left[\prod_{i\in S\setminus T}(1 - W_i)\right] \qquad \text{(B.1)} \\
&\leq \sum_{T\subseteq S} \prod_{i\in T}(1 - y_{ij}^{\mathrm{H}}) \prod_{i\in S\setminus T}(y_{ij}^{\mathrm{H}} - y_{ij}^{\mathrm{L}})(1 - x_i^f) \qquad \text{(B.2)} \\
&= \prod_{i\in S}(1 - y_{ij}^{\mathrm{H}} + (1 - x_i^f)(y_{ij}^{\mathrm{H}} - y_{ij}^{\mathrm{L}})) \\
&= \prod_{i\in S}(1 - y_{ij}) \\
&= \prod_{i\in S} \mathbb{P}(Y_{ij} = 0).
\end{aligned}$$

In Equation (B.1) we use that $Y_{ij}^{\mathrm{H}}, Y_{ij}^{\mathrm{L}}$ are independent across $i$, and that for each term in the sum we are multiplying $(1 - Y_{ij}^{\mathrm{H}})$ and $Y_{ij}^{\mathrm{H}} - Y_{ij}^{\mathrm{L}}$ on disjoint sets. In Inequality (B.2) we use the negative correlation property (P3) of the dependent rounding scheme by Gandhi et al. (2006). □

*Proof of Lemma 16.* Fix $j$ and re-label the warehouses so that

$$r_{1j} \geq r_{2j} \geq \cdots \geq r_{d_j j} \geq r_{d_j+1,j} = \cdots = r_n = 0.$$

Since the unit of demand of node $j$ will be assigned to the warehouse that offers the highest reward, we can write $Z_{ij} = Y_{ij} \prod_{i'<i}(1 - Y_{i'j})$ for all $i = 1, \ldots, d_j$. In words, to assign $j$ to $i$ we need $Y_{ij} = 1$

and $Y_{i'j} = 0$ for all $i'$ with higher reward than $i$. Thus,

$$\mathbb{E}\left[\sum_{i\in[n]} r_{ij}Z_{ij}\right] = \mathbb{E}\left[\sum_{i=1}^{d_j} r_{ij}Y_{ij}\prod_{i'<i}(1-Y_{i'j})\right]$$

$$= \sum_{i=1}^{d_j}(r_{i,j}-r_{i+1,j})\left(1-\mathbb{P}\left(\bigcap_{i'\le i} Y_{i'j}=0\right)\right) \tag{B.3}$$

$$\ge \sum_{i=1}^{d_j}(r_{i,j}-r_{i+1,j})\left(1-\prod_{i'\le i}(1-y_{i'j})\right) \tag{B.4}$$

$$\ge \sum_{i=1}^{d_j}(r_{i,j}-r_{i+1,j})\left(1-\left(1-\frac{1}{i}\right)^i\right) \tag{B.5}$$

$$\ge r_{1j}\left(1-\left(1-\frac{1}{d_j}\right)^{d_j}\right) \tag{B.6}$$

$$\ge \left(1-\left(1-\frac{1}{d_j}\right)^{d_j}\right)\sum_{i\in[n]} r_{ij}y_{ij}. \tag{B.7}$$

In Claim B.3 we expressed the expected reward using increments, where we gain $r_{i,j} - r_{i+1,j}$ if and only if at least one of $Y_{i'j}$ for $i' \le i$ turns out to be 1. In Inequality (B.4) we use the negative correlation property from Claim 2. In Inequality (B.5) we use $\sum_{i\in[n]} y_{ij} \le 1$ (since $D_j = 1$ for all $j$ with the new indexing), from which it follows elementarily that the product $\prod_{i'\le i}(1-y_{i'j})$ is maximized by setting $y_{i'j} = 1/i$ for $i' \le i$ and 0 otherwise. Inequality (B.6) follows by a telescoping sum after using the fact that expression $1-(1-1/i)^i$ is minimized when $i = d_j$. Finally, in Inequality (B.7) we again use that $\sum_{i\in[n]} y_{ij} \le 1$ and that $r_{1j} \ge r_{i'j}$ for all $i' \ne i$. □

### B.1.3 Proof of Theorem 5

Recall we just need to prove on a fixed realization of $D$ that

$$\mathbb{E}_R[\mathsf{OFF}(R(x), D)] \ge \left(1-\left(1-\frac{1}{d}\right)^d\right)\mathsf{OFF}(x, D)$$

We proceed by showing that, for any demand realization $D$, $(Z_{ij})_{(i,j)\in[n]\times[T]}$ is a feasible solution for the LP defining $\mathsf{OFF}(R(x), D)$. Indeed, the degree preservation property (P2) gives us

$$\sum_{j\in[T]} Z_{ij} \leq \sum_{j\in[T]} Y_{ij} \leq R_i(x)$$

for all $i \in [n]$, since the randomized rounding weights satisfy Equations (3.8) and (3.9) so we will not exceed the inventory constraints either if inventory gets rounded up or down. We also have $\sum_{i\in[n]} Z_{ij} \leq D_j = 1$ for all $j \in [T]$. Indeed, we have $Z_{ij} = Y_{ij} \prod_{i'\leq i}(1 - Y_{i'j})$ which will only be 1 for at most one $i \in [n]$ because we assign the demand node to the single warehouse with the highest reward. It follows that, with probability 1,

$$\mathsf{OFF}(R(x), D) \geq \sum_{j\in[T]} \sum_{i\in[n]} r_{ij} Z_{ij}.$$

By taking expectations we get

$$\begin{aligned}
\mathbb{E}_R[\mathsf{OFF}(R(x), D)] &= \mathbb{E}_R[\mathbb{E}_Z[\mathsf{OFF}(R(x), D)|R(x)]] \\
&\geq \mathbb{E}_R\left[\mathbb{E}_Z\left[\sum_{j\in[m]} \sum_{i\in[n]} r_{ij} Z_{ij} \middle| R(x)\right]\right] \\
&= \sum_{j\in[T]} \sum_{i\in[n]} r_{ij} \mathbb{E}_Z[Z_{ij}] && \text{(B.8)} \\
&\geq \sum_{j\in[T]} \left(1 - \left(1 - \frac{1}{d_j}\right)^{d_j}\right) \sum_{i\in[n]} r_{ij} y_{ij} && \text{(B.9)} \\
&\geq \left(1 - \left(1 - \frac{1}{d}\right)^{d}\right) \sum_{j\in[T]} \sum_{i\in[n]} r_{ij} y_{ij} && \text{(B.10)} \\
&= \left(1 - \left(1 - \frac{1}{d}\right)^{d}\right) \mathsf{OFF}(x, D), \square
\end{aligned}$$

where in Equation (B.8) we use linearity of expectation and law of iterated expectations, in Inequality (B.9) we use Lemma 16 and in Inequality (B.10) we use that $\left(1 - \left(1 - \frac{1}{d_j}\right)^{d_j}\right)$ is decreasing in

$d_j$. □

### B.1.4 Proof of Lemma 17

Let $\{n_k\}_{k\geq 1}$ be an increasing sequence of natural numbers such that $n_k/d$ is an integer. Consider the following family of instances indexed by $k$, with $n_k$ warehouses and $\binom{n_k}{d}$ demand nodes, each one of them served by a different subset of $d$ warehouses. The rewards are all equal to 1, and the demand distribution is such that only one of the demand nodes will have demand equal to 1, and the remaining will have demand equal to 0. The total inventory to be distributed is $Q = n_k/d$. The optimal solution for $\max_{x\in \mathrm{CH}(\mathcal{X})} \mathsf{OFF}(x)$ is to set $x_i = 1/d$ for all $i \in [n]$, yielding an optimal value of 1. On the other hand, an optimal integer placement places 1 unit of inventory in $n_k/d$ different warehouses. The reward collected by the offline algorithm is equal to 1 minus the probability of the realized demand node being one of the nodes that can be served given the inventory placement. That is,

$$1 - \prod_{\ell=0}^{d-1} \frac{n_k - n_k/d - \ell}{n_k - \ell} \underset{k\to\infty}{\to} 1 - \left(1 - \frac{1}{d}\right)^d .\square$$

### B.1.5 Proof of Theorem 6

To establish theorem 6, we first define the Rademacher complexity of our sample as:

$$\widehat{\mathcal{R}} := \mathbb{E}_\sigma \left[ \sup_{x\in \mathrm{CH}(\mathcal{X})} \frac{1}{K} \sum_{k=1}^{K} \sigma_k \left(1 - \frac{\mathsf{OFF}(x, D^k)}{Q}\right)\right], \tag{B.11}$$

where $(\sigma_k)_{k\in[K]}$ are independent Rademacher random variables, defined as random variables that are either +1 or −1, with probability $1/2$ each. We note that in (B.11) we are normalizing all values $\mathsf{OFF}(x, D^k)$ to lie in [0,1] (all rewards $r_{ij}$ lie in [0,1] and we have only $Q$ units, so the total reward is upper-bounded by $Q$) and subtracting it from 1 to get a loss function, to obtain the following result.

**Lemma 18**

$$\mathbb{E}\left[\sup_{x\in \mathrm{CH}(\mathcal{X})} \widehat{\mathsf{OFF}}(x) - \mathsf{OFF}(x)\right] \leq 2Q\mathbb{E}_{\wedge}\left[\widehat{\mathcal{R}}\right].$$

*Proof.* This inequality follows by applying a result shown within the proof of Theorem 3.3 in Mohri, Rostamizadeh, and Talwalkar (2018) for a general loss and a general family of functions to choose from. In our case, the loss corresponds to $1 - \mathsf{OFF}(x, D)/Q$, and the family of functions to choose from is $\mathrm{CH}(\mathcal{X})$. By plugging this in, we obtain

$$\mathbb{E}\left[\sup_{x\in \mathrm{CH}(\mathcal{X})} \mathbb{E}_D\left[1 - \frac{\mathsf{OFF}(x, D)}{Q}\right] - \frac{1}{K}\sum_{k=1}^{K}\left(1 - \frac{\mathsf{OFF}(x, D^k)}{Q}\right)\right] \leq 2\mathbb{E}_{\wedge}\left[\widehat{\mathcal{R}}\right].$$

The lemma follows by recognizing $\mathsf{OFF}(x) = \mathbb{E}_D[\mathsf{OFF}(x, D)]$, $\widehat{\mathsf{OFF}}(x) = \sum_{k=1}^{K} \mathsf{OFF}(x, D^k)/K$, rearranging and multiplying both sides by $Q$. □

The following two results that we directly quote will help us upper-bound the Rademacher complexity, thus giving an upper bound to the right-hand side in Lemma 18.

**Proposition 7 (Corollary 4 from Maurer (2016))** *If $(h_k(x))_{k=1}^{K}$ are functions of $x$ that are $L$-Lipschitz in the $\ell_2$ norm, we then have*

$$\mathbb{E}_{\sigma}\left[\sup_{x\in \mathrm{CH}(\mathcal{X})} \sum_{k=1}^{K} \sigma_k h_k(x)\right] \leq L\sqrt{2}\mathbb{E}_{\sigma'}\left[\sup_{x\in \mathrm{CH}(\mathcal{X})} \sum_{i=1}^{n} x_i \sum_{k=1}^{K} \sigma'_{ik}\right],$$

*where $(\sigma_k)_{k\in[K]}$ and $(\sigma'_{ki})_{(k,i)\in[K]\times[n]}$ are independent Rademacher variables.*

**Proposition 8 (Corollary D.11 from Mohri, Rostamizadeh, and Talwalkar (2018))**

$$\mathbb{E}_{\sigma'}\left[\max_{i\in[n]} \sum_{k=1}^{K} \sigma'_{ik}\right] \leq \sqrt{2K\log n},$$

*where $(\sigma'_{ki})_{(k,i)\in[K]\times[n]}$ are independent Rademacher variables.*

We now show that our objective function of interest is indeed Lipschitz in the placement $x$.

**Lemma 19** *Functions* $(h_k(x))_{k\in[K]}$ *defined as*

$$h_k(x) = 1 - \mathsf{OFF}(x, D^k)/Q$$

*are* $(\sqrt{n}/Q)$*-Lipschitz in the* $\ell_2$ *norm.*

*Proof.* Consider two fractional inventory placements $x, x' \in \mathrm{CH}(\mathcal{X})$. We have

$$h_k(x) - h_k(x') = \frac{\mathsf{OFF}(x', D^k) - \mathsf{OFF}(x, D^k)}{Q}.$$

To bound $|\mathsf{OFF}(x', D^k) - \mathsf{OFF}(x, D^k)|$ notice that each if we increase only one component $x_i$ of $x$ by a quantity $\eta$, the objective value of the LP cannot increase by more than $\eta$ because the rewards $r_{ij}$ are upper bounded by 1. With this in mind, define $\eta_i = x_i - x'_i$ and write:

$$\mathsf{OFF}(x, D^k) - \mathsf{OFF}(x', D^k) = \sum_{i=1}^{n} \mathsf{OFF}\left(x + \sum_{i'=1}^{i-1} \eta_{i'} e_{i'}, D^k\right) - \mathsf{OFF}\left(x + \sum_{i'=1}^{i} \eta_{i'} e_{i'}, D^k\right),$$

where $e_{i'}$ is a vector of zeros and a 1 in the $i'$-th component. Since $x + \sum_{i'=1}^{i-1} \eta_{i'} e_{i'}$ and $x + \sum_{i'=1}^{i} \eta_{i'} e_{i'}$ only differ in component $i$, we can use triangular inequality and bound

$$\begin{aligned}
|\mathsf{OFF}(x, D^k) - \mathsf{OFF}(x', D^k)| &\le \sum_{i=1}^{n} \left| \mathsf{OFF}\left(x + \sum_{i'=1}^{i-1} \eta_{i'} e_{i'}, D^k\right) - \mathsf{OFF}\left(x + \sum_{i'=1}^{i} \eta_{i'} e_{i'}, D^k\right) \right| \\
&\le \sum_{i=1}^{n} |\eta_i| = \sum_{i=1}^{n} |x_i - x'_i| = ||x - x'||_1 \le \sqrt{n}||x - x'||_2. \square
\end{aligned}$$

*Proof of Theorem 6.* To show the result we will upper bound the Rademacher complexity of a sample: $\widehat{\mathcal{R}}$. For this, we first combine Lemma 19 with Proposition 7 to bound

$$\sup_{x\in\mathcal{X}} \sum_{k=1}^{K} \sigma_k \left(1 - \frac{\mathsf{OFF}(x, D^k)}{Q}\right) \le \frac{\sqrt{2n}}{Q} \mathbb{E}_{\sigma'} \left[ \sup_{x\in \mathrm{CH}(X)} \sum_{i=1}^{n} x_i \sum_{k=1}^{K} \sigma'_{ik} \right] = \frac{\sqrt{2n}}{Q} Q \mathbb{E}_{\sigma'} \left[ \max_{i\in[n]} \sum_{k=1}^{K} \sigma'_{ik} \right],$$

where in the last equality we simply solve the supremum by placing all mass on the index $i$ with the highest $\sum_{k=1}^{k} \sigma'_{ik}$. Now, by Proposition 8, the right-hand side of the above inequality is at most

$\sqrt{2K \log n}$. Putting it all together, we get

$$\widehat{\mathcal{R}} \leq \frac{1}{K}\sqrt{2n} \cdot \sqrt{2K \log n} = 2\sqrt{\frac{n \log n}{K}}.$$

To conclude we apply this upper bound to Lemma 18 and obtain

$$\mathbb{E}\left[\sup_{x \in \mathcal{X}} \widehat{\mathsf{OFF}}(x) - \mathsf{OFF}(x)\right] \leq 2Q\mathbb{E}_{\wedge}\left[\widehat{\mathcal{R}}\right] \leq 4Q\sqrt{\frac{n \log n}{K}}.\square$$

### B.1.6 Proof of Theorem 7

We can lower-bound the left-hand side in Theorem 7 as follows:

$$\mathbb{E}_{\wedge,R}[\mathsf{ONL}(\pi^{\alpha}, R(\hat{x}))] \geq \alpha \mathbb{E}_{\wedge,R}[\mathsf{OFF}(R(\hat{x}))] \tag{B.12}$$

$$\geq \alpha\left(1 - \left(1 - \frac{1}{d}\right)^{d}\right)\mathbb{E}_{\wedge}\left[\mathsf{OFF}(\hat{x})\right], \tag{B.13}$$

where in Inequality (B.12) we used Assumption 1 and in Inequality (B.13) we used Theorem 5. Now, let $x^*$ denote an optimal solution to $\max_{x \in \mathrm{CH}(X)} \mathsf{OFF}(x)$, and write

$$\mathsf{OFF}(x^*) - \mathsf{OFF}(\hat{x}) = \left(\mathsf{OFF}(x^*) - \widehat{\mathsf{OFF}}(x^*)\right) + \left(\widehat{\mathsf{OFF}}(x^*) - \widehat{\mathsf{OFF}}(\hat{x})\right) + \left(\widehat{\mathsf{OFF}}(\hat{x}) - \mathsf{OFF}(\hat{x})\right). \tag{B.14}$$

We know from linearity of expectation that $\mathbb{E}_{\wedge}[\mathsf{OFF}(x^*) - \widehat{\mathsf{OFF}}(x^*)] = 0$ (note that $x^*$ is fixed and does not depend on the samples), and by definition $\widehat{\mathsf{OFF}}(x^*) - \widehat{\mathsf{OFF}}(\hat{x}) \leq 0$. For the last term in Equation (B.14), we use Theorem 6 to bound

$$\mathbb{E}_{\wedge}\left[\widehat{\mathsf{OFF}}(\hat{x}) - \mathsf{OFF}(\hat{x})\right] \leq \mathbb{E}_{\wedge}\left[\sup_{x \in \mathrm{CH}(X)} \widehat{\mathsf{OFF}}(x) - \mathsf{OFF}(x)\right] = O\left(Q\sqrt{\frac{n \log n}{K}}\right).$$

By taking expectations over $\wedge$ on Equation (B.14) and rearranging we get

$$\mathbb{E}_{\wedge}[\mathsf{OFF}(\hat{x})] \geq \mathsf{OFF}(x^*) - O\left(Q\sqrt{\frac{n\log n}{K}}\right).$$

Carrying on from Inequality (B.13), we get

$$\begin{aligned}
\alpha\left(1-\left(1-\frac{1}{d}\right)^d\right)\mathbb{E}_{\wedge}\left[\mathsf{OFF}(\hat{x})\right] &\geq \alpha\left(1-\left(1-\frac{1}{d}\right)^d\right)\left(\mathsf{OFF}(x^*) - O\left(Q\sqrt{\frac{n\log n}{K}}\right)\right) \\
&= \alpha\left(1-\left(1-\frac{1}{d}\right)^d\right)\mathsf{OFF}(x^*) - O\left(Q\sqrt{\frac{n\log n}{K}}\right) \\
&\geq \alpha\left(1-\left(1-\frac{1}{d}\right)^d\right)\max_{x\in X}\mathsf{OFF}(x) - O\left(Q\sqrt{\frac{n\log n}{K}}\right). \square
\end{aligned}$$

### B.2 Greedy does not improve if $d = 2$

The sequence of examples in Cornuejols, Fisher, and Nemhauser (1977, Thm. 3) showing Greedy to be at best an $(1-(1-1/Q)^Q)$-approximation for any positive integer $Q$ can all be represented by an instance of our problem with $d = 2$.

We first illustrate on the small case of $Q = 3$. There are 9 demand locations arranged in a grid. The realized demand will be 1 at a uniformly chosen location, and 0 everywhere else. There are 5 warehouses: one that serves each row, and one that serves each of the first two columns. The rewards for serving any location in the first, second, and third columns are 3/9, 2/9, and 4/9 respectively, irrespective of the warehouse used. The reward is 0 if a warehouse does not serve a location. Slightly perturbing rewards as necessary, Greedy would place a unit of inventory in the first column warehouse, followed by the second column warehouse, followed by a row warehouse. It would cover 7 locations with rewards $\frac{3}{9}, \frac{3}{9}, \frac{3}{9}, \frac{2}{9}, \frac{2}{9}, \frac{2}{9}, \frac{4}{9}$ and hence earn expected reward $19/81$. The optimal placement, on the other hand, is to place one unit of inventory in each row warehouse, covering all locations and earning expected reward $27/81$. The approximation ratio is $19/27 = 1-(1-1/3)^3$, even though $d = 2$ (each location is only served by its row warehouse and column warehouse).

This argument can be extended for general $Q$. Let there again be $Q^2$ demand locations arranged in a $Q$ by $Q$ grid, and the realized demand will be 1 at a uniformly chosen location and 0 everywhere else. There are $2Q - 1$ warehouses: one for each row, serving all locations within the row, and one for each column except the last one, serving all locations within the column. We define the reward $r_i$ for all locations in column $i$ as follows, allowing them to exceed 1 without loss of generality (we can scale all rewards down by $Q^Q$ without changing the result.) For the last column, the reward is $r_Q = (Q-1)^{Q-1}$, and for the remaining columns the reward is defined recursively according to the following equations:

$$Qr_i = \sum_{j=i}^{Q} r_j \quad \forall i \in [Q-1]. \tag{B.15}$$

This equation can be interpreted as imposing that the sum of the rewards in column $i$ must be equal to the sum of the rewards in each row, summing from $i$ all the way to the right up to $Q$. It can be verified that $\sum_{i=1}^{Q} r_i = Q^{Q-1}$, so the sum of the rewards in the grid is equal to $Q^Q$.

If we slightly perturb the rewards so that the left hand side in Equation (B.15) dominates, greedy will put a unit of inventory in every column and one row. This placement covers all locations except $Q - 1$ of them in the last row, yielding an expected reward of

$$\frac{Q^Q - (Q-1)r_Q}{Q^2} = \frac{Q^Q - (Q-1)^Q}{Q^2}.$$

On the other hand, an optimal placement places a unit in every row, yielding a reward of $Q^Q/Q^2$. Thus, the ratio between both is

$$\frac{Q^Q - (Q-1)^Q}{Q^Q} = 1 - \left(1 - \frac{1}{Q}\right)^Q.$$

### B.3 Incorporating Warehouse Capacity and Initial Inventories

As mentioned in Section 3.2, all of our results directly extend to the setting where each warehouse has a capacity limiting the number of stock units that can be placed there, and where each

warehouse also has a starting inventory. Let $C_i$ be the maximum number of units that can be stored at warehouse $i$, and let $S_i$ be the initial inventory at warehouse $i$ (that cannot be relocated). All of our results hold in this setting by solving the following generalization of $\max_{x \in \mathrm{CH}(\mathcal{X})} \widehat{\mathsf{OFF}}(x)$:

$$
\begin{aligned}
\max_{x \in \mathbb{R}^n_+, y \in \mathbb{R}_+^{[n]\times[m]\times[K]}} \quad & \frac{1}{K} \sum_{k=1}^{K} \sum_{i \in [n]} \sum_{j \in [m]} r_{ij} y_{ij}^k & \\
\text{s.t.} \quad & \sum_{i=1}^{n} y_{ij}^k \leq D_j^k & \forall j \in [m], k \in [K], \\
& \sum_{i=j}^{m} y_{ij}^k \leq x_i + S_i & \forall i \in [n], k \in [K], \\
& x_i \leq C_i & \forall i \in [n], \\
& \sum_{i=1}^{n} x_i = Q. &
\end{aligned}
$$

All of the proofs follow without change after this modification. We have to be careful in that, given the new constraints $x_i \leq C_i$ for all $i \in [n]$, if $Q$ is too big then constraint $\sum_{i=1}^{n} x_i = Q$ could make the problem infeasible. We can come around this by reducing $Q$ to the largest $Q'$ that makes the problem feasible. This change only lowers the number of samples required to obtain a sampling error of $\varepsilon$ in Theorem 7.

## B.4 Experiments

### B.4.1 LP vs Greedy for optimizing Offline surrogate

In Subsection 3.3.1 we mention that an alternative approach for optimizing the Offline surrogate is a greedy algorithm. We compare the solutions obtained through rounding the linear relaxation and the greedy algorithm by deploying them in our 90 instances. These are obtained with different combinations of networks (complete, long-chain, RDC-FDC), demand models (DH-TI, RH-TI, RO-SI), weights settings (uniform, rewards-based), and total inventory $Q \in \{30, 45, 60, 75, 90\}$.

For each of the 90 instances, we generate 500 samples of demand sequences and find solutions for $\max_{x\in\mathcal{X}} \widehat{\mathsf{OFF}}(x)$ using both methods. Remarkably, all of the solutions found by the linear relaxation were integer, so no rounding was needed and the solutions were optimal. For the greedy procedure, we start with all warehouses empty. In each iteration, we solve the linear program definining $\widehat{\mathsf{OFF}}(x)$ and place a unit of inventory in the warehouse whose inventory dual variable is the maximum. We iterate this procedure until we have placed all of the $Q$ units.

We find that both procedures produce very similar solutions in terms of placement and performance. For instance, the output of both procedures is exactly the same for 49 out of the 90 instances. For the instances where this is not the case, the output are still very alike. Out of the 41 instances with different placements, on average 2.8% of the units are placed differently.

The solutions are hard to distinguish in terms of performance too. Since the optimal solution of the LP is integer, the reward obtained by the LP solution is always greater than the one obtained by the greedy procedure. That being said, across the 90 instances, the expected reward collected by the greedy solution is at least 99.75% of the expected reward collected by the LP solution. To control for potential overfitting to the samples used to generate the solutions, we sampled 500 fresh demand samples and evaluated the expected rewards. We find that, in 90 instances, the average out-of-sample reward collected by the greedy solution is always within 99.63% and 100.09% of what the LP solution collects out-of-sample.

The only substantial difference that we found between both solutions was the time required to obtain each of them. For each instance, we look at the ratio between the runtime of the LP and the runtime of the greedy procedure. This ratio was at least 4 and at most 155, with an average value of 43. Given these findings, from this point on we will continue to use the LP relaxation and (potentially) rounding the solution to obtain the Offline placement.

### B.4.2 Description of Placement Procedures

**Fluid Placement.** This inventory placement solves the linear relaxation $\max_{x\in\mathrm{CH}\mathcal{X}} \mathsf{FLU}(x)$ and rounds the obtained solution using a greedy rounding scheme. The explicit linear relaxation that

this inventory placement solves is the following:

$$\begin{aligned}
\max_{x,y\geq 0} \quad & \sum_{i\in[n]}\sum_{j\in[m]} y_{ij} r_{ij} \\
\text{s.t.} \quad & \sum_{i\in[n]} y_{ij} \leq \mathbb{E}[D_j] && \forall j \in [m], \\
& \sum_{j\in[m]} y_{ij} \leq x_i && \forall i \in [n], \\
& \sum_{i\in[n]} x_i = Q.
\end{aligned}$$

In order to round a fractional solution, a greedy approach is taken. All fractional components are ordered from largest to smallest, and are rounded upwards one by one until the total inventory placement is exactly $Q$.

**Demand-scaled Fluid Placement.** This placement procedure again solves a fluid relaxation, but it modifies the expected demands so that they match the total inventory. Specifically, define $\rho = Q/\sum_{j\in[m]} \mathbb{E}[D_j]$ as the ratio between the total inventory and the expected total demand across all demand nodes. Then, the linear relaxation this placement procedure solves is

$$\begin{aligned}
\max_{x,y\geq 0} \quad & \sum_{i\in[n]}\sum_{j\in[m]} y_{ij} r_{ij} \\
\text{s.t.} \quad & \sum_{i\in[n]} y_{ij} \leq \rho\mathbb{E}[D_j] && \forall j \in [m], \\
& \sum_{j\in[m]} y_{ij} \leq x_i && \forall i \in [n], \\
& \sum_{i\in[n]} x_i = Q.
\end{aligned}$$

Again, if a fractional solution is obtained, it will be rounded greedily.

**Offline Placement.** This placement procedure optimizes the sample-average approximation of the Offline Placement surrogate. Specifically, $K$ demand vectors are sampled (according to the corresponding demand model). Let $D_j^k$ be the demand originating from demand node $j$ on sample

$k$. Then, the linear problem this surrogate solves is the following:

$$\begin{aligned}
\max_{x,y\geq 0} \quad & \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in[n]}\sum_{j\in[m]} r_{ij}y_{ij}^{k} & \\
\text{s.t.} \quad & \sum_{i=1}^{n} y_{ij}^{k} \leq D_j^k & \forall j \in [m], k \in [K], \\
& \sum_{i=j}^{m} y_{ij}^{k} \leq x_i & \forall i \in [n], k \in [K], \\
& \sum_{i=1}^{n} x_i = Q. &
\end{aligned}$$

As a remark, all of the solutions obtained by solving this linear program were integral, so no rounding was required.

**Myopic Placement.** This placement procedures uses the Myopic Surrogate. That is, it optimizes the inventory placement as if the fulfillment algorithm to be deployed downstream was myopic. This myopic algorithm chooses, out of the warehouses with remaining inventory, the one that offers the maximum immediate reward. This surrogate is optimized using a greedy procedure. Specifically, let $\mathsf{MYO}(x)$ denote the expected reward collected by the myopic fulfillment algorithm. The algorithm first initializes $x^{(0)} = (0)_{i\in[n]}$. Then, we iteratively set $x^{(\ell+1)} = \max_{i\in[n]} \mathsf{MYO}(x^{(\ell)} + e_i)$, where $e_i$ is a vector of zeros with a one on component $i$. The algorithm stops once $\sum_{i\in[n]} x_i^{(\ell)} = Q$, which is equivalent to stopping at $x^{(Q)}$. As with Offline Placement, this is solved through sample-average approximation, where $K$ demand sequences are sampled, and the value being optimized is the average reward collected by the myopic fulfillment policy across the $K$ samples.

### B.4.3 Description of Fulfillment Policies

**Myopic Fulfillment.** In this policy, each demand node $j \in [m]$ has a fixed priority list over the warehouses $i \in [n]$, decreasing in $r_{ij}$. When demand from node $j$ arrives, the warehouse $i$ with the highest $r_{ij}$ and remaining inventory is chosen for fulfillment.

**Fluid Shadow Prices.** This policy considers opportunity costs for depleting a unit of inventory in a certain warehouse through the shadow prices of the Fluid linear program. Before the online process begins, it solves the Fluid LP:

$$\begin{aligned} \max_{y \geq 0} \quad & \sum_{i \in [n]} \sum_{j \in [m]} y_{ij} r_{ij} \\ \text{s.t.} \quad & \sum_{i \in [n]} y_{ij} \leq \mathbb{E}[D_j] && \forall j \in [m], \\ & \sum_{j \in [m]} y_{ij} \leq x_i && \forall i \in [n]. \end{aligned} \tag{B.16}$$

Let $\lambda_i$ be the dual variable for constraint (B.16). Then, whenever demand from node $j$ arrives, the warehouse $i$ that has the highest $r_{ij} - \lambda_i$ and remaining inventory is chosen for fulfillment.

**Offline Shadow Prices.** This policy considers opportunity costs for depleting a unit of inventory in a certain warehouse through the shadow prices of the Fluid linear program. Before the online process begins, it solves the Fluid LP:

$$\begin{aligned} \max_{x,y \geq 0} \quad & \frac{1}{K} \sum_{k=1}^{K} \sum_{i \in [n]} \sum_{j \in [m]} r_{ij} y_{ij}^k \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_{ij}^k \leq D_j^k && \forall j \in [m], k \in [K], \\ & \sum_{i=j}^{m} y_{ij}^k \leq x_i && \forall i \in [n], k \in [K]. \end{aligned} \tag{B.17}$$

Let $\lambda_{ik}$ be the dual variable for constraint (B.17). Then, whenever demand from node $j$ arrives, the warehouse $i$ with the highest $r_{ij} - \sum_{k=1}^{K} \lambda_{ik}$ and remaining inventory is chosen for fulfillment.

**Re-solving.** All of the above policies are static in the sense that they do not update their beliefs of the opportunity costs of depleting inventory units. This is suboptimal for two reasons. First, certain demand realizations can produce imbalanced inventory consumptions, producing scarcity in warehouses that started with abundant inventory, which should increase its opportunity cost. Second, information about demand arrivals during the time horizon can change our beliefs of

future demand arrivals. For that, we also implement the re-solving versions of the Fluid and Offline shadow prices policies. In particular, we re-solve the corresponding LPs after the first arrival with arrival time of $\alpha \in \{1/3, 2/3\}$, considering the current inventory levels and updating our beliefs about the demand distributions (except for the DH-TI model).

**Posterior update.** The following Bayesian update is used to update the beliefs of the demand distributions. Let $T$ be a Geometric random variable with parameter $p$, representing the total number of arrivals, and let $X_1, \ldots, X_T$ be the arrival times of each demand arrival (the order statistics of $T$ IID Uniform$[0, 1]$ random variables). Let $T^\alpha$ be the number of random variables with arrival time at most $\alpha$. That is, $T_\alpha = |\{i : X_i <= \alpha\}|$. Let $T^{1-\alpha}$ be such that $T = T^\alpha + T^{1-\alpha}$. Then $T^{1-\alpha}|T^\alpha$ follows a negative binomial distribution with parameters $T^\alpha + 1$ and $1 - (1 - p)(1 - \alpha)$. This posterior is applied differently for the RH-TI and RO-SI models. In the former, we update our belief of the total number of demand arrivals, and they preserve the probabilities that they arrive from each demand node. In the latter, we update our beliefs for each demand node independently.

### B.4.4 Results for Reward-based Weights

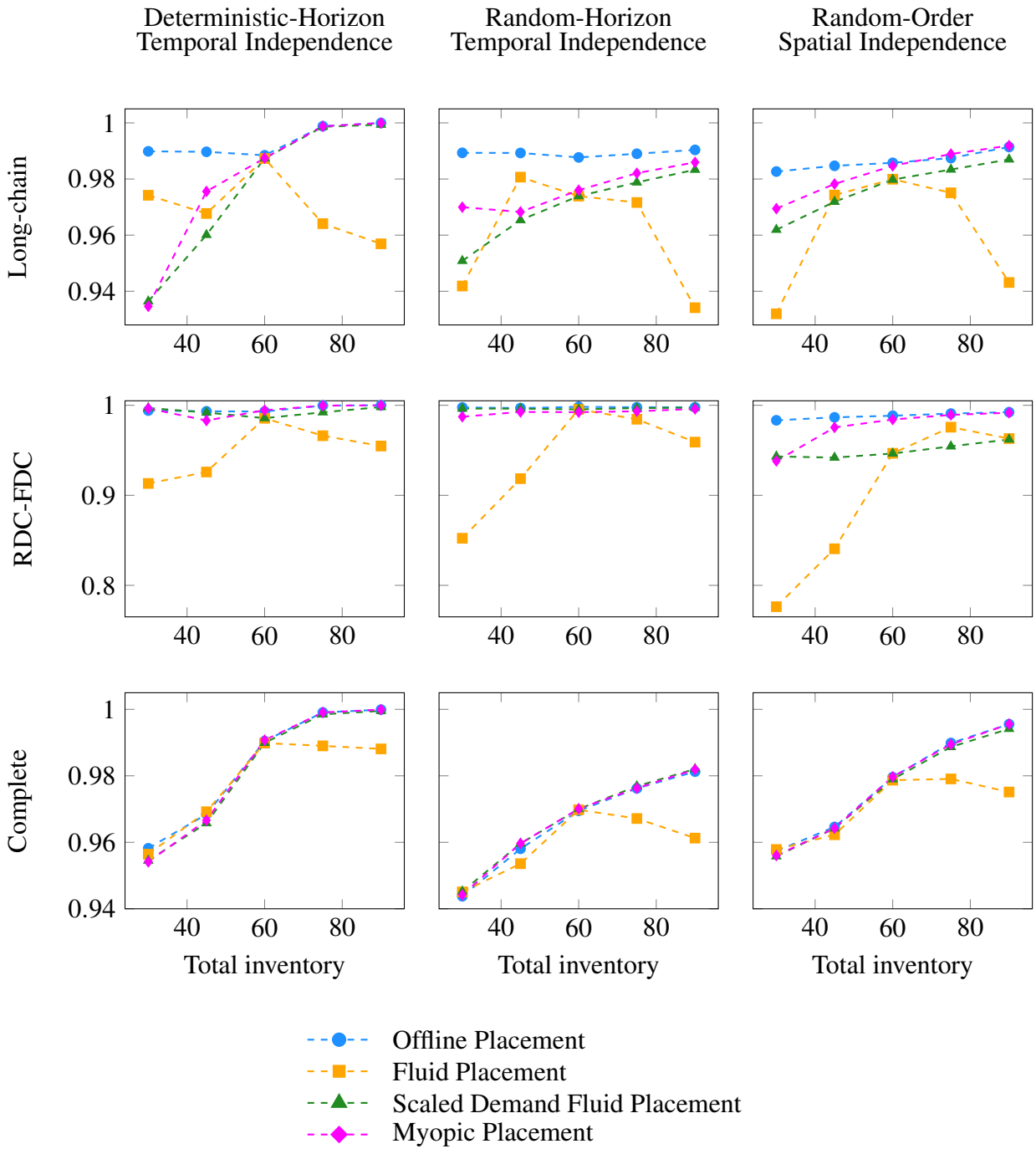


Figure B.1: Competitive ratio as a function of total inventory for all combinations of graphs and demand models with reward-based weights. Each row corresponds to a different network structure and each column corresponds to a different demand model.