NEW BENCHMARKING TECHNIQUES IN RESOURCE ALLOCATION PROBLEMS: THEORY AND APPLICATIONS IN CLOUD SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Sebastian Walter Perez Salazar

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

H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology

August 2022

NEW BENCHMARKING TECHNIQUES IN RESOURCE ALLOCATION PROBLEMS: THEORY AND APPLICATIONS IN CLOUD SYSTEMS

Thesis committee:

Dr. Mohit Singh
H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. Siva Theja Maguluri
H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. Alejandro Toriello
H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. Prasad Tetali
Department of Mathematical Sciences
Carnegie Mellon University

Dr. Santanu Dey
H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology

Date approved: May 2, 2022

To my friends and family

ACKNOWLEDGMENTS

I could not have written this thesis without the support of many crucial people. First, I would like to thank my parents, Maria Salazar and Arturo Perez, for allowing me to pursue my dreams and loving me unconditionally.

I want to express my sincere gratitude to my advisors, Mohit Singh and Alejandro Toriello, for their guidance during my Ph.D. studies. I was very fortunate to have the opportunity to work with them. They allowed me to explore my interests and they constantly pushed me to try new ideas and envision new horizons. Both have become role models for me in every aspect. Thank you, guys!

My thanks to my friends from the Ph.D.: Adrian, Alejandro, Anatoliy, Sheng-Tao, Daniela, Adam, Anthony, Yuliia, Reem, Mohamed, Francisco, Ignacio, Idil, Beste, Bastian, Hassan, Minas, Georgios, Jiaming, Sajad, Majid, Digvijay, Kaizhao, Shixuan, Woody, Jana, Jorge, Seyma, and to all the people that I met at GT. Also, I would like to thank the "Chileans", Alfredo, Ramon, Matias, and Adolfo, for all the lunches and the parties together. My special thanks to my friends from DIM: Enrique, Pato, Pablo, Pipe Campos, Garrido, Pancho Arana, Pancho Venegas, Hasson, Emilio, Cabezas, Bob, Bustos, and Martin. Thanks to my good ol' friends from karate: Ricardo, Paulina, and Jhonatan, for being an anchor to Chile.

I want to express my gratitude to my dear Eyes Kareeratana for her unconditional love. Without her support, company, and spicy food over these years, my life would have been only salt and pepper, and this thesis would not have been possible.

I want to thank Santanu Dey, Siva Theja Maguluri, and Prasad Tetali for serving on my committee and providing insightful comments. A big shout out to Rachel Cummings and Ishai Menache for being excellent collaborators who shared their wisdom with me. And a big thank you to my former advisor, Ivan Rapaport, who introduced me to academic life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Motivated by different e-commerce applications such as allocating virtual machines to servers and online ad placement, we study new models that aim to capture unstudied tensions faced by decision-makers. In online/sequential models, future information is often unavailable to decision-makers—e.g., the exact demand of a product for next week. Sometimes, these unknowns have regularity, and decision-makers can fit random models. Other times, decision-makers must be prepared for any possible outcome. In practice, several solutions are based on classical models that do not fully consider these unknowns. One reason for this is our present technical limitations. Exploring new models with adequate sources of uncertainty could be beneficial for both the theory and the practice of decision-making. For example, cloud companies such as Amazon WS face highly unpredictable demands of resources. New management planning that considers these tensions have improved capacity and cut costs for the cloud providers. As a result, cloud companies can now offer new services at lower prices benefiting thousands of users. In this thesis, we study three different models, each motivated by an application in cloud computing and online advertising.

From a technical standpoint, we apply either worst-case analysis with limited information from the system or adaptive analysis with stochastic results learned after making an irrevocable decision. A central aspect of this work is dynamic benchmarks as opposed to static or offline ones. Static and offline viewpoints are too conservative and have limited interpretation in some dynamic settings. A dynamic criterion, such as the value of an optimal sequential policy, allows comparisons with the best that one could do in dynamic scenarios. Another aspect of this work is multi-objective criteria in dynamic settings, where two or more competing goals must be satisfied under an uncertain future. We tackle the challenges introduced by these new perspectives with fresh theoretical analyses, drawing inspiration from linear and nonlinear optimization and stochastic processes.

CHAPTER 1

INTRODUCTION

The rise of new problems in e-commerce has spurred revitalized interest in cost-effective resource allocation. In the last couple of decades, companies such as Amazon, Google, Microsoft, and Meta (ex Facebook Inc.) have discovered the benefits of applying cutting-edge academic developments in practical scenarios. As a result, tech companies have grown their R&D departments steadily, hiring research scientists with specialized skills. For example, Revelio Labs reported that Google employs the highest number of people with a Ph.D. in STEM—at least 3000 of them having a Ph.D. in Computer Science. The runner-ups in the list are Intel, Apple, and Microsoft [1].

In numbers, Amazon WS, Amazon's public cloud computing subsidiary, has increased its revenue from approximately 3 billion USD in 2013 to more than 62 billion USD in 2021 [2]. Likewise, Microsoft Azure, the second-largest public cloud provider, reported about 50 billion USD in revenue in 2021 [3]. One of the reasons for the success of cloud enterprises has been the introduction of virtualization in the early 2000s, which triggered a rapid economy of scale. Recently, with never-ending improvements, such as the optimization of operative systems' kernels [4], more energy-efficient integrated chips and customized airflow [5], and new virtualization technologies such as containers [6], costs and energy consumption have decreased even further for cloud providers. Thus cloud computing has become an example of the success of integrated advancements coming from multiple areas of knowledge.

Another example of success is online advertising. Meta reported that roughly 98% of the

company's revenue comes from online ad placement—this was approximately 84 billion USD for 2020. The following year, 2021, Meta's revenue increased to 114 billion USD [7]. In the 90s, when the internet was in its infancy, every user visiting a webpage would observe the same ads or banners. With the increasing use of personalized ads, auction-based algorithms, and new classes of contracts such as pay-per-click, platforms started offering better online experiences for both the advertisers and the users visiting webpages. Underlying these advancements in cloud computing and online advertising, we can find fascinating resource allocation models such as knapsack, bin packing, and bipartite matching, to name a few.

Every mathematical model is an approximation of a real-world problem. Limited by our current technology and present-day ingenuity, we simplify further these models to make them tractable. For instance, one simplification is the reduction of random settings to averaged models, which gives us fewer variables and numbers to work with but a less expressive model. The loss of expressiveness—and possibly applicability—by simplifications is the price paid for tractability. The field of operations research has come a long way since its modern inception during World War II. For example, now we can solve linear models with millions of variables in a matter of seconds [15]. Likewise, modeling perspectives have become more complex. However, more complex models are good as long as they preserve tractability. Applying models that do not consider basic behaviors of the real-world problem leaves a lot to fix for practitioners and decision-makers (DM). Depending on the risk aversion of DMs, some might be willing to go through post-processing steps to try to remediate mistakes. On the other hand, the risk-averse DMs will often underutilize resources to avoid losses. A better understanding of the sources of uncertainty and how to react appropriately to them can help DMs reduce costs and improve the services for final end-users. In this work, we aim to formulate new models for resource allocation, motivated by applications in cloud computing and online advertising. These models extend some of the ideas present in classical models but consider unstudied sources of uncertainty faced by

practitioners. We seek to provide solutions to the motivating application that are simple to analyze and implement.

In the next section, we formalize the research questions posed by this thesis. These questions are later on answered by our proposed models.

1.1 Research Problems and Objectives

This thesis aims to shed light into the following fundamental questions:

1. How can we allocate resources while satisfying a baseline requirement under a highly unpredictable environment? Uncertainty is at the core of many sequential problems, and most of the time, we have to make decisions without knowing future data or outcomes. Frequently, DMs agree to give end-users a minimum quality of service, which most of the time translates into a hard baseline requirement. For instance, cloud users buy resources and expect to find them whenever they connect to the cloud. On the other side, DMs have their own goals, which often compete with the baseline requirements. For example, it is common for cloud providers to aim for high utilization of resources; however, this goal competes with the baseline requirements for cloud users. Indeed, just meeting baseline requirements can lead to poor utilization of the resources, and just concentrating on utilization may deprive users of computational resources. We propose a model for this question in Chapter 2. See also the next section for a summary of that chapter.

2. How should we face the allocation of resources when the source of uncertainty is revealed after an irrevocable decision? For example, it is hard to preempt the exact resources users need in cloud computing before they start using their resources. In another example, it is hard to know which user will click on an ad in online advertising. Unfortunately, many models ignore the effects of the uncertainty and leave

the practitioners to deal with it. We tackle this question with dynamic models that consider by design that sources of uncertainty are revealed after making irrevocable decisions. This introduces new burdens, such as observing outcomes predicted by the model but undesirables for the DM. We provide in this thesis solutions that can recourse under any possible outcome while staying close to an optimal target solution. Moreover, from a complexity standpoint, these solutions are easy to compute and implement.

3. How should we benchmark algorithms in online/sequential environments? Benchmarking a solution is essential to communicate its quality. Competitive analysis [13] has been the gold standard for benchmarking algorithms for a long time. In this setting, we often compare the output value of an algorithm against the optimal output of a (possibly nonexistent) algorithm. In several settings, this metric is uninformative (e.g., see Chapter 3 for an example). Therefore, we require more precise metrics depending on the model. We explore in this thesis different metrics for different models, and we justify our choice in each case. Sometimes, this choice is based on community standards, while in other cases, this choice will be based on the tightest value that gives some meaningful information about our model.

1.2 Contributions of the Thesis

We answer these questions by studying different models for cloud computing and online advertising applications. Despite the straightforward application of the models, each of them is presented as general as possible to make them suitable beyond the motivational application. The following is a summary of the contributions of this thesis:

1.2.1 Dynamic Resource Allocation in the Cloud

In Chapter 2, we develop an online algorithm for allocating a resource in shared systems while satisfying a baseline requirement. In past years, multiple companies have started offering *public cloud services* at a low cost, e.g., Amazon WS and Microsoft Azure. Consequently, common end-users now have easy access to high-end computing technology. Sharing resources efficiently, such as CPU and bandwidth, between different users and taking advantage of economies of scale make this business model viable. Each cloud user has different demand patterns and requirements. The offered service often comes with a service-level agreement (SLA) that specifies the amount of resources a user is entitled to. Usually, providers would like to operate resources at high utilization while satisfying all the SLAs. Yet, these goals compete with each other. Just meeting SLAs can lead to poor utilization, and just concentrating on utilization may deprive users of computational resources.

Contributions

We propose a discrete-time online model for a single shared resource that considers the following: (1) We digress from known stochastic models to *adversarial* users' demand model, i.e., we make no assumption over the demands. (2) We consider *limited feedback* from the system—common in many cloud settings—indicating users with non-empty backlogs, namely, the *active users*. (3) We introduce the notion of *SLA satisfaction*, which occurs when work done for a user in any window of time is at least the work done by her SLA.

We introduce an algorithm with provable near-optimal resource utilization and approximate SLA satisfaction, for each user. We base our algorithm on multiplicative updates. Since we are unable to see the lengths of users' backlogs, we pretend that active users have gigantic backlogs. From here, we extract a multiplicative update rule that boosts active users; however, active users who are assigned less than their SLA are boosted slightly more than the

rest. We provide regret-like guarantees against any *dynamic* offline solution—rather than static solutions often found in regret analysis. We use our algorithm's dynamic to construct dual feasible solutions of a dynamic work-maximization LP. For SLA satisfaction, using the boosted update, we show that users with a backlog recover their SLA requirements in a few steps. Finally, we present extensive empirical studies of our algorithm on synthetic data and real data from production services in Microsoft's cloud.

The results of this chapter appeared in the INFORMS Journal of Operations Research [139]. This was joint work with M. Singh, A. Toriello and I. Menache (Microsoft Research). A preliminary version of this article was a runner-up for the 2019 ICS Student Paper Prize.

1.2.2 Adaptive Bin Packing with Overflow

In Chapter 3, we examine a general stochastic bin packing problem. In the standard online bin packing formulation, n items with sizes in $[0, 1]$ arrive in an online fashion, and we aim to pack the items into the fewest possible unit-capacity bins. Applications appear in virtual machine (VM) allocation, scheduling problems, and others. In many cases, the items' sizes are uncertain and modeled via probability distributions, with most of the models assuming that sizes are known upon arrival. Yet, in many practical cases, this is unrealistic. In VM allocation into servers, the resources required by VMs are often uncertain and deviate from their typical utilization. VMs with utilization higher than expected can destabilize the server, affecting operating costs; however, we can only learn the resources used after allocating the VM.

Contributions

We introduce an online bin packing problem that considers the following: (1) Arrivals are adversarial distributions and the length of the item sequence is unknown to the decision-maker. (2) In contrast to existing work in stochastic bin packing, when an item arrives, the decision-maker only sees a distribution of its size. (3) The decision-maker learns the

item's actual size only after irrevocably placing it in a bin; hence, overflowing a bin is a possibility. (4) An overflowed bin incurs in a large penalty and renders the bin unusable from that point on—e.g. a server overload. The goal is to minimize the expected cost given by the sum of the number of open bins and the overflow penalty.

We design an online algorithm that incurs an expected cost of at most a constant factor times the cost incurred by the optimal packing policy when item sizes are an i.i.d. sequence. Our algorithm keeps the number of overflowed bins bounded by a small fraction of the bins used. This is appealing for risk-averse applications where overflowing bins might affect largely the operating costs. Our proofs are based on careful modifications of decision trees that transform any decision tree into the decision tree of our algorithm, by incurring in a constant multiplicative loss. We also consider the sequential offline version of the packing problem, where we know all the distributions in advance. We design a polynomial-time approximation scheme policy with a small additional bin capacity. We also show that computing the optimal cost is $\#\mathbf{P}$-hard.

The results of this chapter were published in the INFORMS Journal of Mathematics of Operations Research [141]. This article was joint work with M. Singh and A. Toriello.

1.2.3 Robust Online Selection with Uncertain Offer Acceptance

In Chapter 4, we digress from cloud computing applications, and we move to a model motivated by online advertising. Effective online advertising has been crucial for companies like Meta and Google. However, not only big companies benefit from displaying ads online. Numerous small businesses now have access to global platforms and higher chances to reach the right customers. It has been estimated that the click-through rates (CTR) are relatively small, of the order of 1% [71]. That means that out of 100 online users who observe the same ad on a web page, only one will click on it on average. The current models used to design algorithmic solutions for online advertising either ignore the effect of CTR

or average it to the model's objective. Studying models and implementing solutions that take into account CTR and the impact of users ignoring ads could potentially increase the revenue for the platforms and the service for the advertisers.

Contributions

We model this problem as a secretary problem, where candidates (the users) can reject the offer (the ad). A sequence of competing candidates arrives online in random order. Upon arrival, we can assess the quality of the candidate compared to previously observed ones, and we either extend an offer to the candidate or move on to assess the next one, without the possibility of recalling previously observed candidates. The candidate can accept the offer with probability p and the process ends, or reject it, in which case we move on to assess the next one. Suppose that we knew that a top k candidate is willing to accept an offer, then we would like to maximize the probability of making an offer to one of such candidates. In reality, we do not know k, thus we consider a *robust objective* that *maximizes* the minimum of all these probability-based scenarios for all k. A robust objective of at least γ guarantees a probability of success at least γ in any of these possible scenarios where a top k candidate is willing to accept an offer.

We provide an optimal algorithm, which attains the optimal robust objective. The algorithm solves a linear program (LP) and implements its optimal solution in the input sequence. We use a Markov decision processes framework to deduce our LP, and this deduction is generalizable to other online selection problems. We further our analysis by providing closed-form bounds for the robust objective and near-optimal policies. We do this by using an infinite LP, the limit of our LP when the number of candidates goes to infinite. Moreover, in utility settings, we show that our algorithm is optimal among all algorithms that can make decisions based on the rank of the values and attains a fraction of at least the robust objective of the optimal offline value.

Part of this chapter is in the preprint [140]. This is joint work with M. Singh and A. Toriello.

CHAPTER 2

DYNAMIC RESOURCE ALLOCATION IN THE CLOUD WITH NEAR-OPTIMAL EFFICIENCY

Cloud computing has motivated renewed interest in resource allocation problems with new consumption models. A common goal is to share a resource, such as CPU or I/O bandwidth, among distinct users with different demand patterns as well as different quality of service requirements. To ensure these service requirements, cloud offerings often come with a service level agreement (SLA) between the provider and the users. An SLA specifies the amount of a resource a user is entitled to utilize. In many cloud settings, providers would like to operate resources at high utilization while simultaneously respecting individual SLAs. There is typically a tradeoff between these two objectives; for example, utilization can be increased by shifting away resources from idle users to "scavenger" workload, but with the risk of the former then becoming active again. We study this fundamental tradeoff by formulating a resource allocation model that captures basic properties of cloud computing systems, including SLAs, highly limited feedback about the state of the system, and variable and unpredictable input sequences. Our main result is a simple and practical algorithm that achieves near-optimal performance on the above two objectives. First, we guarantee nearly optimal utilization of the resource even if compared to the omniscient offline dynamic optimum. Second, we simultaneously satisfy all individual SLAs up to a small error. The main algorithmic tool is a multiplicative weight update algorithm, and a primal-dual argument to obtain its guarantees. We also provide numerical validation on real data to demonstrate the performance of our algorithm in practical applications.

The content of this chapter appeared in the INFORMS Journal of Operations Research,

2021 [139]. It was joint work with M. Singh, A. Toriello, and I. Menache. This work was also supported by the U.S. National Science Foundation via grants CMMI 1552479, AF 1910423 and AF 1717947.

2.1 Introduction

Cloud computing has motivated renewed interest in resource allocation, manifested in new consumption models (e.g., AWS spot pricing), as well as the design of resource-sharing platforms [99, 170]. These platforms need to support a heterogenous set of users, also called tenants, that share the same physical computing resource, e.g., CPU, memory, I/O bandwidth. Providers such as Amazon, Microsoft and Google offer cloud services with the goal of benefiting from economies of scale. However, the inefficient use of resources – over-provisioning on the one hand or congestion on the other – could result in a low return on investment or in loss of customer goodwill, respectively. Hence, resource allocation algorithms are key for efficiently utilizing cloud resources.

To ensure quality of service, cloud offerings often come with a *service level agreement* (SLA) between the provider and the users. An SLA specifies the amount of resource the user is entitled to consume. Perhaps the most common example is renting a virtual machine (VM) that guarantees an explicit amount of CPU, memory, etc. Naturally, VMs that guarantee more resources are more expensive. In this context, a simple allocation policy is to assign each user the resources specified by their SLAs. However, such an allocation can be wasteful, as users may not need the resource at all times. In principle, a dynamic allocation of resources can increase the total efficiency of the system. However, allocating resources dynamically without carefully accounting for SLAs can lead to user dissatisfaction.

Recent scheduling proposals address these challenges through work-maximizing yet fair schedulers [176, 82]. However, such schedulers do not have explicit SLA guarantees. On the other hand, other works focus on enforcing SLAs [54, 104, 88], but do not explicitly

optimize the use of extra resources.

Our goal in this work is to understand the fundamental tradeoff between high utilization of resources and SLA satisfaction of individual users. In particular, we design algorithms that guarantee *both* near optimal utilization as well as the satisfaction of individual SLAs, simultaneously. To that end, we formulate a basic model for online dynamic resource allocation. We focus on a single divisible resource, such as CPU or I/O bandwidth, that has to be shared among multiple users. Each user also has an SLA that specifies the fraction of the resource it expects to obtain. The actual demand of the user is in general time-varying, and may exceed the fraction specified in the SLA. As in many real systems, the demand is not known in advance, but rather arrives in an online manner. Arriving demand is either processed or queued up, depending on the resource availability. In many real-world scenarios, it is difficult to measure the actual demand size (see, e.g., [133]). Accordingly, we assume that the system (and the underlying algorithm) receives only a simple *binary feedback* per user at any given time: whether the user queue is empty (the user's work arriving so far has been completed), or not. This is a plausible assumption in many systems, because one can observe workload activity, yet anticipating how much of the resource a job will require is more difficult. Additionally, it also models settings where demands are not known in advance.

While online dynamic resource allocation problems have been studied in different contexts and communities (see Section 2.1.3 for an overview), our work aims to address the novel aspects arising in the cloud computing paradigm, particularly the presence of SLAs, the highly limited feedback about the state of the system, and a desired robustness over arbitrary input sequences. For the algorithm design itself, we pay close attention to practicality; our approach involves fairly simple computations that can be implemented with minimal overhead of space or time. Our algorithm achieves nearly optimal utilization of the resource, as well as approximately satisfying the SLA of each individual user. We see two

main use-cases for the algorithm:

- In enterprise settings ("private cloud"), different applications or organizations share the same infrastructure. These often have SLAs, but providers would still like to maximize the ROI by maximizing utilization [148].

- In public clouds, users buy VMs, which are offered at different "sizes" (which is practically the SLA). In addition, the service providers offer "best-effort" alternatives, such as Azure Batch (MS) or Spot instances (AWS). In our model, these services can be modeled by giving an SLA of zero. Here, satisfying the VM SLAs and achieving high utilization are both important; indeed, the provider is paid for the best-effort workloads only if it completes these jobs. Our work can be viewed as a principled way to accommodate such services, and even give VMs better service than expected, an important consideration as public cloud offerings gradually become commoditized.

2.1.1 The Model

We consider the problem of having multiple tenants or users sharing a single resource, such as CPU, I/O or networking bandwidth. For simplicity, we assume that the total resource capacity is normalized to 1. We have N users sharing the resource, a finite but possibly unknown discrete time horizon indexed $t = 1, \ldots, T$, and an underlying queuing system. For each user i, we are also given an expected share of resource $\beta(i) \geq 0$ satisfying $\sum_{i=1}^{N} \beta(i) \leq 1$. The input is an online sequence of workloads $L_1, \ldots, L_T \in \mathbf{R}_+^N$, where $L_t(i) \geq 0$ corresponds to i's workload arising at time t. The system maintains a queue $Q_t(i)$, denoting i's remaining work at time t. In our model, the decision maker does *not* have direct access to the values of the queues or the workloads. This allows us to consider settings where the job sizes are not known in advance and minimal information is available about the underlying system, a regular occurrence in many cloud settings. At time t, the

following happens:

1. **Feedback:** The decision maker observes which queues are non-empty (the set of users i with $Q_t(i) > 0$, the *active* users), and which are empty ($Q_t(i) = 0$, the *inactive* users).

2. **Decision:** The decision maker updates user resource allocations $h_t(i)$, satisfying $\sum_i h_t(i) \leq 1$.

3. **Update:** The load $L_t(i)$ for each i arrives and each user processes as much of the work from the queue plus the arriving workload as possible. The work completed by user i in step t is

$$w_t(i) := \min\{h_t(i), L_t(i) + Q_t(i)\}.$$

   The queues at the end of the time step are updated accordingly,

$$Q_{t+1}(i) = \max\{0, L_t(i) + Q_t(i) - h_t(i)\}.$$

We assess the performance of any algorithm based on two measures.

1. **Work Maximization.** The algorithm should maximize the total work completed over all users, and thus utilize the resource as much as possible.

2. **SLA Satisfaction.** The algorithm should (approximately) satisfy the SLAs in the following manner. The work completed by user i up to any time $1 \leq t \leq T$ should be no less than the work completed for this user up to t if it were given a constant $\beta(i)$ fraction of the resource over the whole horizon.

Achieving either of the criteria on their own is straightforward. A greedy strategy that takes away resources from an idle user and gives them to any user whose queue is non-empty is approximately work-maximizing (see Appendix 2.A.5 for details). On the other

hand, to satisfy the SLAs, we give each user a *static* assignment of $h_t(i) := \beta(i)$ for all t. Naturally, the two criteria compete with each other; the following examples illustrate why these simple algorithms do not satisfy both simultaneously.

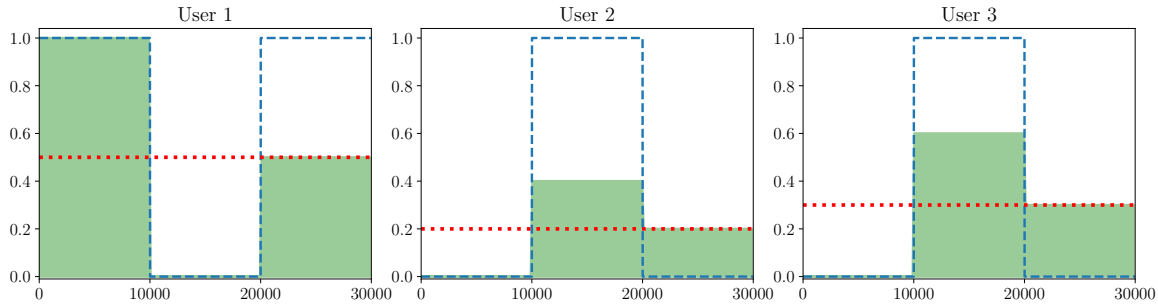

Figure 2.1: Example of loads and work for 3 users. The blue dashed lines show each user's workload. The green areas represent the work done by the users. The red dashed lines depict SLAs.

Example 1. We have a shared system with three users and corresponding SLAs $\beta(1) = 0.5, \beta(2) = 0.2$ and $\beta(3) = 0.3$. Loads are defined by

$$L_t(1) = \begin{cases} 1 & t = 1, \ldots, T/3, 2T/3 + 1, \ldots, T \\ 0 & t = T/3 + 1, \ldots, 2T/3 \end{cases}, \text{ and } \quad L_t(2) = L_t(3) = 1 - L_t(1).$$

We assume that T is divisible by 3. In Figure 2.1 we show the three users' loads in blue dashed lines and the corresponding SLAs in dotted red lines. The static solution given by the SLAs, i.e. $h_t(i) = \beta(i)$ for all t, ensures a total of $5T/6$ work done. However, the dynamic policy shown in the green line, given by

t	$[0, T/3]$	$[T/3 + 1, 2T/3]$	$[2T/3 + 1, T]$
$h_t(1)$	1	0	0.5
$h_t(2)$	0	0.4	0.2
$h_t(3)$	0	0.6	0.3

ensures T work is done (the green area). Moreover, it also ensures SLA satisfaction at all times. An alternative policy is

t	$[0, T/3]$	$[T/3+1, 2T/3]$	$[2T/3+1, T]$
$h_t(1)$	1	0	0
$h_t(2)$	0	1	0
$h_t(3)$	0	0	1

which is also work maximizing. However, it does not ensure SLA satisfaction. Indeed, this policy does not satisfy user 3's SLA at any time in $(T/3, 2T/3]$.

We remark that achieving both criteria is relatively simple if we allow the decision maker to observe demand or even the queue length. This can be achieved by first allocating to each user as much of the resource as necessary up to their SLA, and then distributing the remaining resource arbitrarily among users with additional demand. The versatility of our setting stems from the limited feedback in the form of binary information about idle and busy users. In our cloud computing context, full demand information or even visible queue lengths are unrealistic assumptions.

2.1.2 Results and Contributions

We design a simple and efficient online algorithm that achieves approximate work maximization as well as approximate SLA satisfaction even in the limited feedback model that we consider. For work maximization, we analyze the performance by comparing our algorithm to the *optimal offline dynamic allocation* that knows all the data up front. In contrast, our online algorithm receives limited feedback even in an online setting. Thus, our aim is to minimize the quantity

$$\text{work}_{\mathbf{h}_1^*,\ldots,\mathbf{h}_T^*} - \text{work}_{\text{alg}} = \sum_{t=1}^{T}\sum_{i} w_t^*(i) - \sum_{t=1}^{T}\sum_{i} w_t(i),$$

where $\text{work}_{\mathbf{h}_1^*,\ldots,\mathbf{h}_T^*}$ is the optimal offline work done by dynamic allocations $\mathbf{h}_1^*, \ldots, \mathbf{h}_T^*$, $\mathbf{w}_t^* = (w_t^*(1), \ldots, w_t^*(N))$ is the work done at time t by these allocations, and work_{alg} is the work done by the algorithm with allocations $\mathbf{h}_1, \ldots, \mathbf{h}_T$ and work $\mathbf{w}_t = (w_t(1), \ldots, w_t(N))$

at time t. The objective of the decision maker is to *minimize* this quantity by constructing a sequence of good allocations that approach the best allocations in hindsight. Note that our benchmark is *dynamic*, rather than the more common *static* offline optimum usually considered in regret minimization [16, 96, 155]. Similarly, for SLA satisfaction, our benchmark is the total work done for a user if they were given $\beta(i)$ resources for each time $1 \leq t \leq T$. We give a bi-criteria online algorithm that achieves nearly the same performance as the benchmarks if the resources for the latter are slightly more constrained than that of the algorithm. Algorithm 1, which we formally describe in Section 2.2, follows a multiplicative weight approach. The idea is to boost the allocations of active users by a factor greater than 1, with more emphasis on users with current allocation below their SLA. This intuition translates into a simple update that ensures high utilization of the resource and SLA satisfaction; formally:

Theorem 2.1. *For any input parameter $0 < \varepsilon \leq 1/10$, SLAs $\beta = (\beta(1), \ldots, \beta(N))$ satisfying $\beta(i) \geq 2\frac{\varepsilon}{N}$, and online loads $L_1, \ldots, L_T \in \mathbf{R}^N_{\geq 0}$, Algorithm 1 achieves the following guarantees:*

1. ***Approximate Work Maximization.*** *Let $\mathbf{h}^*_1, \ldots, \mathbf{h}^*_T \in [0,1]^N$ be an optimal offline sequence of allocations such that $\sum_i h^*_t(i) = 1$ for all $1 \leq t \leq T$. Then*

$$\text{work}_{alg} \geq (1-\varepsilon)\,\text{work}_{\mathbf{h}^*_1,\ldots,\mathbf{h}^*_T} - \mathcal{O}\left(N\varepsilon^{-2}\log(N/\varepsilon)\right).$$

2. ***Approximate SLA Satisfaction.*** *There exists $\tilde{p} = \tilde{p}(N,\varepsilon) = \mathcal{O}\left(N^2\varepsilon^{-3}\log(N/\varepsilon)\right)$ such that for any user i and time t, if we take $\mathbf{h}'_1, \ldots, \mathbf{h}'_T \in [0,1]^N$ to be any sequence of allocations with $h'_t(i) \leq \beta(i)$, then*

$$\sum_{\tau=1}^{t} w_\tau(i) \geq (1-2\varepsilon)\sum_{\tau=1}^{t} w'_\tau(i) - \beta(i)\tilde{p},$$

where $\mathbf{w}'_t$ *is the work performed by the allocations* $\mathbf{h}'_1, \ldots, \mathbf{h}'_T$.

The first part of Theorem 2.1 asserts that if T is known for the system, we can achieve $\text{work}_{\text{offline}} - \text{work}_{\text{alg}} \leq \mathcal{O}(T^{2/3} \log T)$ with $\varepsilon = \Theta\left(\frac{N^{1/3}}{T^{1/3}}\right)$. However, this choice of ε is suboptimal for SLA satisfaction. We can achieve an improved bound for SLA satisfaction (when $t = T$) by picking $\varepsilon = \Theta\left(\frac{N^{1/3}}{T^{1/4}}\right)$. If T is unknown, we can use a standard doubling trick, see for example [155]. Summarizing, we obtain the following result.

Corollary 2.1.1. *For* $\varepsilon = \Theta\left(\frac{N^{1/3}}{T^{1/4}}\right)$*, Algorithm 1 guarantees*

$$\text{work}_{\mathbf{h}^*_1,\ldots,\mathbf{h}^*_T} - \text{work}_{alg} \leq \mathcal{O}(N^{1/3}T^{3/4} + N^{1/3}T^{1/2} \log(NT)) = \mathcal{O}(N^{1/3}T^{3/4}),$$

where $\mathbf{h}^*_1, \ldots, \mathbf{h}^*_T$ *are optimal offline dynamic allocations.*

As T grows, Corollary 2.1.1 guarantees that the *rate* of work done by our algorithm $\frac{\text{work}_{\text{alg}}}{T}$, approaches the rate of work done by the optimal dynamic solution $\frac{\text{work}_{\mathbf{h}^*_1,\ldots,\mathbf{h}^*_T}}{T}$. We emphasize again that this is a stronger guarantee using the much more powerful optimal dynamic solution as a benchmark, rather than the typical static allocation used in regret analysis for online algorithms. Such a guarantee can be obtained in our model because the incomplete work remains stored in the queues until we are able to finish it; this allows the algorithm to *catch up* with the incomplete work.

The second result in Theorem 2.1 states that the work done by any individual user is comparable to the work done by their promised SLA. In other words, the user's queue length is not much larger than it would be under a static SLA allocation. By using $\varepsilon = \Theta\left(\frac{N^{1/3}}{T^{1/4}}\right)$ we obtain the following result.

Corollary 2.1.2. *Let* $\varepsilon = \Theta\left(\frac{N^{1/3}}{T^{1/4}}\right)$*. For a user* i*,* $Q_T(i) \leq Q'_T(i) + \mathcal{O}(NT^{3/4} \log(NT))$*, where* Q_t *is the queue given by Algorithm 1 and* Q'_t *is the queue induced by any dynamic policy* $\mathbf{h}'_1, \ldots, \mathbf{h}'_T$ *with* $h'_t(i) \leq \beta(i)$*.*

We remark that we need to bound the SLAs away from $\frac{\varepsilon}{N}$; in Theorem 2.1 we use the bound $\beta(i) \geq 2\frac{\varepsilon}{N}$. This condition is necessary in our analysis to guarantee that there is an over-provisioned ($h_t(i) > \beta(i)$) user from which we can move allocation to an under-provisioned user. See Theorem 2.5 for details and a more relaxed bound.

Corollary 2.1.1's guarantee is near-optimal asymptotically in T in terms of work maximization, as the following result shows. The proof of this result appears in Appendix 2.A.6.

Theorem 2.2. *For any online deterministic algorithm $\mathcal{A}$ for our model, there is a sequence of online loads $L_1, \ldots, L_T$ such that* $\text{work}_{\mathbf{h}_1^*,\ldots,\mathbf{h}_T^*} - \text{work}_{\mathcal{A}} = \Omega\left(\sqrt{T}\right)$*, where $\mathbf{h}_1^*, \ldots, \mathbf{h}_T^*$ are optimal offline dynamic allocations.*

Our algorithm follows a mirror descent approach [24, 96]. Unable to see the lengths of the queues, a (naive) approach is to pretend that active users have gigantic queues. From this approach we extract a simple update rule that multiplicatively boosts active users; however, active users who are under their SLA are boosted slightly more than other active users. If inactive users are assigned more than ε of the resource, where ε is the algorithm parameter, active users ramp up their allocation in few iterations. In the opposite case, at least $1 - \varepsilon$ of the resource is assigned to active users, and the slight boost to users below their SLA ensures a healthy re-balancing of the resource. We show that this efficient heuristic strategy is enough to achieve approximate work maximization and SLA satisfaction. We remark that the mirror descent interpretation is only used to provide intuition for the algorithm, and our proofs follow a different path than the usual mirror descent analysis. Later, we detail a slight modification that enjoys the same theoretical guarantees as Algorithm 1 but in practice exhibits more desirable behavior (see Algorithm 2). Intuitively, among active users, the modified algorithm tries to keep allocations proportional to their SLAs. This behavior is appealing; for example, a user A with twice the SLA of another user B would expect in practice to perform at least twice as much work. Similarly, user B would expect to receive no less than half of user A's allocation. This second algorithm exhibits another in-

teresting feature; it can be applied to over-committed regimes with $\sum_{i=1}^{N} \beta(i) > 1$, remain work-maximizing, and satisfy a normalized version of SLA satisfaction (see Section 2.3.4).

The analysis of the algorithm relies on a primal-dual fitting approach. For work maximization, we can write the offline dynamic optimal allocation as a solution to a linear program and then construct feasible dual solutions with objective value close to the algorithm's resource utilization. A crucial ingredient of the algorithm is the use of entropic projection on the *truncated* simplex, which ensures every user gets at least a ε/N fraction of the resource at all times. Intuitively, this means any user with a non-empty queue will recover their SLA requirement in a few steps.

We do an extensive analysis of our algorithm on synthetic data as well as real data obtained from CPU traces of a production service in Microsoft's cloud. We aim to quantify the performance of the algorithm on three objectives, (i) work maximization, (ii) SLA guarantee and (iii) queue behaviour. While our theoretical results give guarantees for these objectives, we show experimentally that the algorithm exceeds these guarantees. We benchmark the algorithm against natural online algorithms, such as a static allocation as given by the SLA guarantee, or the algorithm that aims to proportionally distribute the resource among active clients (according to their SLA). We also benchmark against offline algorithms that know all input data up front; our algorithm's performance on various measures is comparable to the offline algorithms.

This work is organized as follows. In Section 2.2, we present the preliminaries and the basic version of the multiplicative weight algorithm. Section 2.3 contains the proof of Theorem 2.1 in the bi-criteria form. We split the proof into two parts: work maximization in Section 2.3.2 and SLA satisfaction in 2.3.3. In Section 2.3.4, we present the extension of our algorithm and its guarantees. Finally, in Section 2.4 we present numerical experiments that empirically validate our results, and conclude in Section 2.5.

2.1.3 Related Work

There has been growing interest in resource allocation problems arising from cloud computing applications, both from a practical as well as a theoretical standpoint [88, 99, 148, 104, 54, 133, 170, 134, 130]. The focus of many of these works has been to understand the trade-offs between efficiency and ensuring guarantees to individual users.

On the more theoretical side, the cloud computing allocation problem has been modeled as a stochastic allocation problem [126, 125, 124]. The underlying models draw inspiration from a large body of work on stochastic network control, originating from the seminal work of [165, 166], followed by additional related research, e.g., on uplink and downstream scheduling in wireless networks [135, 136]. The analytical results in these papers characterize the stability region of the arrival processes under certain stochastic assumptions (e.g., i.i.d. processes), and suggest algorithms that achieve maximal throughput. The main distinction between these works and ours is that we assume an *adversarial* input, i.e., we do not make stationary distributional assumptions on the input. Another difference is that our model centers on the notion of an SLA which is known to the algorithm. This allows us to address the over-committed case (see Algorithm 2), which is especially relevant in cloud settings.

Despite these modeling differences, there are some parallels worth mentioning. For example, we design algorithms with rate of work $\frac{\text{work}_{\text{alg}}}{T}$ approaching the optimal offline rate of work; this translates to the average delay $\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} Q_t(i)$ converging to the optimal offline average delay. This result can be compared to the limiting behavior of the Markov process in many of the stochastic network models. For example, [136] studies a system of N users connecting to a server via ON/OFF channels. The paper presents an algorithm that ensures bounded average delay $\limsup_T \frac{1}{T}\,\mathbf{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{N} Q_i(t)\right]$ for any input within the interior of the stability region; here $Q_i(t)$ is the i-th user's backlog (queue length) at time t. Much of the stochastic network literature assumes full knowledge of queue lengths, e.g.

the LCQ policy in [166], although there are studies that limit the information available to the decision maker in a similar fashion to our model (see [118, 135, 124, 157]).

More broadly, the general problem of online resource allocation has been studied in both stochastic and adversarial settings; we refer the reader to the books [13, 29, 163] on the topic. Our work differs from the aforementioned lines of research by combining the three following elements already present in the literature. First, as mentioned above, we digress from the stochastic arrival model to the adversarial setting and worst-case analysis. This makes our algorithm robust against unpredictable users' demands. For instance, demands in the morning could be totally different from demands in the afternoon or the morning of the next day. We are able to provide a single strategy that adapts easily to any scenario. Second, we provide simple online algorithms that perform well even under *limited feedback*, a typical situation in cloud systems in which we can only determine the utilization of a resource after it has been allocated. Finally, we consider SLA satisfaction as a measure of user contentment, and seek to satisfy it up to a small error.

There is now an extensive literature devoted to the pricing of cloud computing services. In [123] the authors study a genetic model for generating a suitable pricing function in the cloud market. In [81], pricing is studied via a revenue management formulation to address resource provisioning decisions. See also [138, 156] for more pricing models. A closely related topic is fairness in resource allocation [176, 82]. Although we do not directly consider pricing, work maximization could be interpreted as a way to obtain extra revenue by allocating unused resources to active users.

More recently, there has been work considering over-commitment in the cloud [49, 86, 55], that is, selling resources beyond server capacity. One of the objectives of over-commitment is to reduce the number of servers opened in order to minimize energy consumption. In our basic model, we do not assume over-commitment, yet our algorithm can still be applied to that setting (see Algorithm 2). Specifically, we obtain a normalized version of SLA

satisfaction in the over-commitment setting, where the guarantees depend on how much the system is over-committed (see Section 2.3.4).

A fundamental tool in our design is *mirror descent* algorithms [24]. These first-order iterative algorithms have been widely used in optimization [24], online optimization and machine learning [96, 131, 155] to generate update policies under limited feedback. Similarly, multiplicative weights algorithms have been widely studied in optimization [142, 16], online convex optimization [96], online competitive analysis [35] and learning theory [76, 155]. Our results bear some resemblance to regret analysis, where typically the benchmark is the optimal offline static policy [155, 96, 8, 33, 76]; the use of a dynamic benchmark (as in our work) is scarcer in the literature, see e.g. [93, 132, 178, 177].

2.2 Algorithm

2.2.1 Preliminaries

For $N \geq 1$, we identify the set of *users* with the set $[N] = \{1, \ldots, N\}$. For $0 < \varepsilon < 1$, we call an allocation $\mathbf{h} = (h(1), \ldots, h(N)) \in [0,1]^N$ a $(1-\varepsilon)$-*allocation* if $\sum_i h(i) \leq 1 - \varepsilon$. We assume $N \geq 2$, that is, the system consists of at least two users.

For any t, we define the set of *active users* at that time as the set of users with non-empty queue, and denote this set by A_t. Observe that $h_t(i) = w_t(i)$ for all active users. Let $B_t = [N] \setminus A_t$ be the sets of users with empty queues at time t; we call these users *inactive*. A_t and B_t correspond to the feedback given to the decision maker. Also, let $A_t^1 = \{i \in A_t : h_t(i) < \beta(i)\}$ be the set of active users with allocation below their SLA and $A_t^2 = A_t \setminus A_t^1$ be the set of active users receiving at least their SLAs.

We assume without loss of generality that the allocations set by the decision maker always add up to 1. We propose an algorithm that uses a multiplicative weight strategy to boost a subset of users by multiplying their allocation by factor greater than one. Because the

allocations do not sum to one after applying the update, we then project them onto the simplex using the KL-divergence metric. Furthermore, to ensure no user gets an allocation arbitrarily close to zero, we in fact project onto the *truncated simplex*,

$$\Delta_\varepsilon = \{\mathbf{x} = (x(1), \ldots, x(N)) : \|\mathbf{x}\|_1 = 1, x(i) \geq \varepsilon/N, \forall i\}.$$

To fix notation, let $\pi_{\Delta_\varepsilon}(\cdot)$ be the *projection function* onto Δ_ε using Kullback–Leibler divergence (KL-divergence for short), i.e., $\pi_{\Delta_\varepsilon}(\mathbf{y}) := \operatorname{argmin}_{\mathbf{x}\in\Delta_\varepsilon} \sum_i x(i)\log(x(i)/y(i))$, where $\mathbf{y} = (y(1), \ldots, y(N)) \in \mathbf{R}^N_{\geq 0}$. In Appendix 2.A.1, we show how to efficiently compute this projection. The following proposition states some basic facts that are useful in our analysis. The proof appears in Appendix 2.A.1.

Proposition 2.3. *Let* $\mathbf{y} \in \mathbf{R}^N_+$, $\mathbf{x} = \pi_{\Delta_\varepsilon}(\mathbf{y})$, *and* $S = \{i : x(i) = \varepsilon/N\}$. *Then:*

(a) If $y(1) \leq y(2) \leq \cdots \leq y(N)$, *then* $S = \{1, \ldots, k\}$ *for some* $k \geq 0$.

(b) $x(i) = y(i)e^{\mu_i}C$, *where* $C = \left(\frac{1-\frac{\varepsilon}{N}|S|}{\sum_{j\notin S} y(j)}\right)$, $\mu_i \geq 0$ *for all* i *and* $\mu_i = 0$ *for* $i \notin S$.

(c) $\mathbf{x}$ *can be computed in* $\mathcal{O}(N \log N)$ *time.*

2.2.2 The Multiplicative Weight Algorithm

We propose an algorithm that follows a multiplicative weight strategy (see Algorithm 1). We describe here the basic approach given by the mirror descent algorithm. In Section 2.3.4 we present an extension of the algorithm that in practice shows a better relation between the allocations and the ratios between the SLAs.

Algorithm 1: Multiplicative Weight Update Algorithm

Input: Parameters $0 < \varepsilon \leq \frac{1}{10}, 0 < \eta < \frac{1}{3}$.

Initialization: $\mathbf{h}_1$ any allocation over Δ_ε and $\lambda = \frac{\varepsilon^2}{8N}$.

for $t = 1, \ldots, T$ **do**

Set allocation $\mathbf{h}_t$.

Read active and inactive users A_t and B_t. $A_t^1 = \{i \in A_t : h_t(i) < \beta(i)\}$, $A_t^2 = A_t \setminus A_t^1$.

Set gain function $g_t(i) = \begin{cases} 1+\lambda & i \in A_t^1 \\ 1 & i \in A_t^2 \\ 0 & i \in B_t \end{cases}$.

Update allocation:

$$\widehat{h}_{t+1}(i) = h_t(i) e^{\eta g_t(i)}.$$

$$\mathbf{h}_{t+1} = \pi_{\Delta_\varepsilon}(\widehat{h}_{t+1}).$$

Intuitively, the algorithm boosts active users at the expense of inactive ones, and boosts users slightly more if they are currently under their SLA. The algorithm update rule comes from a mirror descent approach applied to a Lagrangian relaxation of a work-maximization linear function at time t. More formally, under the assumption that active users have a huge queue, we aim to maximize the objective $\sum_{i \in A_t} w_t(i)$ subject to $w_t(i) \geq \beta(i)$ for $i \in A_t$. The update rule is obtained after applying a mirror descent with a KL-divergence distance generating function over the simplex to the Lagrangian relaxation of the previous problem (see [31, 24]). We restrict the projection to the truncated simplex so no user gets an allocation too close to 0. We use this update rule solely to guide the algorithm's decisions; however, the proofs of work maximization and SLA satisfaction do not follow from the standard analysis of mirror descent.

2.3 Analysis

To give the analysis of the algorithm and prove Theorem 2.1, we prove the following stronger guarantees about Algorithm 1. We compare its performance to the optimal offline dynamic strategy that uses at most a $1 - 4\varepsilon$ fraction of the resources at each time step.

Theorem 2.4. *Given loads $L_1, \ldots, L_T$, for any $\varepsilon > 0$ and $\eta > 0$ such that $\varepsilon \leq 1/10$, Algorithm 1 guarantees*

$$\mathrm{work}_{\mathbf{h}_1^*,\ldots,\mathbf{h}_t^*} - \mathrm{work}_{alg,t} \leq 8\frac{N}{\varepsilon^2\eta}\ln(N/\varepsilon),$$

for any time $1 \leq t \leq T$, where $\mathbf{h}_1^*, \ldots, \mathbf{h}_T^*$ is the optimal offline sequence of $(1-4\varepsilon)$-allocations and $\mathrm{work}_{alg,t} = \sum_i \sum_{\tau=1}^{t} w_\tau(i)$ is the work done by Algorithm 1 until time t.

The first guarantee of Theorem 2.1 regarding work maximization now follows simply from Theorem 2.4. Given any offline dynamic policy $\mathbf{h}_1, \ldots, \mathbf{h}_T$ such that $\sum_i h_t(i) = 1$, we define $\bar{\mathbf{h}}_t := (1-4\varepsilon)\mathbf{h}_t$, which satisfies the assumption of Theorem 2.4. Now we have

$$\begin{aligned}
\mathrm{work}_{\mathrm{alg}} &\geq \mathrm{work}_{\bar{\mathbf{h}}_1,\ldots,\bar{\mathbf{h}}_T} - 8\frac{N}{(\varepsilon/4)^2\eta}\ln(4N/\varepsilon) \\
&\geq (1-4\varepsilon)\cdot \mathrm{work}_{\mathbf{h}_1,\ldots,\mathbf{h}_T} - 2000\frac{N}{\varepsilon^2\eta}\ln(N/\varepsilon),
\end{aligned}$$

where the first inequality follows from Theorem 2.4. To argue the second, let $\mathbf{w}_1, \ldots, \mathbf{w}_T$ and $\overline{\mathbf{w}}_1, \ldots, \overline{\mathbf{w}}_T$ respectively denote the work performed by allocations $\mathbf{h}$ and $\bar{\mathbf{h}}$. Then $(1-4\varepsilon)\mathbf{w}_1, \ldots, (1-4\varepsilon)\mathbf{w}_T$ are feasible work patterns that the allocations $\overline{\mathbf{h}}_1, \ldots, \overline{\mathbf{h}}_T$ could do, since in this setting we have $1 - 4\varepsilon$ capacity and the same workload. Therefore, $(1-\varepsilon)\,\mathrm{work}_{\mathbf{h}_1,\ldots,\mathbf{h}_T} \leq \mathrm{work}_{\overline{\mathbf{h}}_1,\ldots,\overline{\mathbf{h}}_T}$, because the users try to use their allocations at maximum.

$$
\begin{array}{l|l}
(P_\varepsilon) \quad \max \quad \sum_{i=1}^{N}\sum_{t=1}^{T} w_t(i) & (D_\varepsilon) \quad \min \quad \sum_{i=1}^{N}\sum_{t=1}^{T} L_t(i)\gamma_t(i) + (1-\varepsilon)\sum_{t=1}^{T}\beta_t \\
\text{s.t.} & \text{s.t.} \\
\forall t, i \quad \sum_{s=1}^{t} w_s(i) \le \sum_{s=1}^{t} L_s(i) \quad (2.1) & \forall t, i \quad \gamma_t(i) + \beta_t \ge 1 \quad (2.4) \\
\forall t \quad \sum_{i=1}^{N} w_t(i) \le 1-\varepsilon \quad (2.2) & \forall t \quad \gamma_t \ge \gamma_{t+1} \quad (2.5) \\
\forall t \quad \mathbf{w}_t \ge 0 \quad (2.3) & \forall t \quad \beta_t, \gamma_t \ge 0 \quad (2.6)
\end{array}
$$

Figure 2.2: The primal and dual LP formulation for the maximum work problem.

Similarly, for SLA satisfaction, we prove a stronger bi-criteria result that implies the SLA guarantee in Theorem 2.1.

Theorem 2.5. *Let $0 < \varepsilon \le 1/10$ and $0 < \eta \le 1/3$. Take any SLAs $\beta(1), \ldots, \beta(N)$ such that $\beta(i) \ge e^{\eta(1+\lambda)}\frac{\varepsilon}{N}$, where $\lambda = \frac{\varepsilon^2}{8N}$ and let $\tilde{p} = 32\frac{N^2}{\varepsilon^3\eta}\ln(N/\varepsilon)$. Then, for any user i and time $t \le T - \widetilde{p}$, if we take $\mathbf{h}'_1, \ldots, \mathbf{h}'_T$ to be allocations such that $h'_t(i) = (1-2\varepsilon)\,\beta(i)$, the work done by Algorithm 1 for user i satisfies*

$$\sum_{\tau=1}^{t+\tilde{p}} w_\tau(i) \ge \sum_{\tau=1}^{t} w'_\tau(i),$$

where $\mathbf{w}'_t$ is the work done by the allocations $\mathbf{h}'_1, \ldots, \mathbf{h}'_T$. Moreover, $\sum_{\tau=1}^{t} w_\tau(i) \ge \sum_{\tau=1}^{t} w'_\tau(i) - \beta(i)\tilde{p}$.

2.3.1 The Offline Formulation

Before presenting the proof of Theorem 2.4, we state the offline LP formulation of the maximum work problem for $(1-\varepsilon)$-allocations. We denote by $\mathbf{w}_t = (w_t(1), \ldots, w_t(N))$ the work done for each user at time t. Given loads $L_1, \ldots, L_T$, the offline formulation and its dual LP are given in Figure 2.2. As written, the dual LP includes a change of variable; see Appendix 2.A.2 for details. Constraints (2.1) state that the work done for any user up to time t by the allocation cannot exceed the user's loads up to that time. Constraints (2.2) limit the work performed at time t to at most a $1-\varepsilon$ fraction of the resource. The LP (D_ε)

will be of special importance in the analysis. Using our algorithm, we will construct a dual feasible solution.

Observe that (P_ε) is feasible and bounded since the feasible region is a non-empty polytope. Let v_{P_ε} be the optimal value of (P_ε). The following proposition gives a simple characterization of v_{P_ε}; the proof appears in Appendix 2.A.2.

Proposition 2.6. $v_{P_\varepsilon} = \min_{0 \le t \le T} \left(\sum_{s=1}^{t} \sum_i L_s(i) + (1-\varepsilon)(T-t) \right)$.

2.3.2 Work Maximization

In this section we prove Theorem 2.4. Our first Lemma characterizes the implications of the update rule. The proof follows from a careful analysis of the dynamics using the KL-divergence and appears in Appendix 2.A.2.

The first result of the lemma shows the behavior of active users' allocations when the system is under-utilized ($\le 1-\varepsilon$). In this case, all the active users receive a multiplicative boost in their allocation. The second result shows a more general behavior (see also Lemma 2.10). In this case, active users with allocation below their SLA do not decrease their allocations while the other active users might decrease their allocation, but in this case, the multiplicative penalization will be less severe.

Lemma 2.7. *Let $c = \frac{\varepsilon\eta}{4N}$. Then Algorithm 1 satisfies the following:*

1. *Suppose $\sum_{i \in A_t} h_t(i) \le 1-\varepsilon$. If $i \in A_t$, then $h_{t+1}(i) \ge h_t(i)(1+c)$.*
2. *In general, Algorithm 1 satisfies $h_{t+1}(i) \ge h_t(i)$ for $i \in A_t^1$ and $h_{t+1}(i) \ge h_t(i)(1-\varepsilon c)$ for $i \in A_t^2$.*

Proof of Theorem 2.4. Given loads $L_1, \ldots, L_T \in \mathbf{R}_+^N$, consider the following $\{0,1\}$-matrix M of dimension $N \times T$ that encodes the information about the status of queues obtained

while running Algorithm 1

$$M_{i,t} = \begin{cases} 0 & i\text{'s queue is empty at } t, Q_t(i) = 0, \\ 1 & i\text{'s queue is not empty at } t, Q_t(i) > 0. \end{cases}$$

Let $\widetilde{s} = \frac{\ln(N/\varepsilon)}{\varepsilon c}$, where c is defined in Lemma 2.7. Now, pick $s^\star$ to be the maximum non-negative integer s (including 0) such that

$$\sum_{t=1}^{s} \sum_{i} L_t(i) \leq \sum_{t=1}^{s+\widetilde{s}} \sum_{i} w_t(i) \tag{2.7}$$

Claim 2.8. *Consider any block of time $[r, r+\tilde{s}]$ where $r > s^\star$; then there exists a user i such that $M_{i,r'} = 1$ for all $r' \in [r, r+\tilde{s}]$.*

Proof. Suppose not. Then we claim that $s = r$ satisfies condition (2.7). Consider any user i and let $r_i' \in [r, r+\tilde{s}]$ be such that $M_{i,r_i'} = 0$. Then work done by the user i up to time $r+\tilde{s}$ is at least

$$\sum_{t=1}^{r+\tilde{s}} w_t(i) \geq \sum_{t=1}^{r_i'} w_t(i) = \sum_{t=1}^{r_i'} L_t(i) \geq \sum_{t=1}^{r} L_t(i).$$

Now summing over all i, we get the desired contradiction. □

We now prove the following claim that shows that the algorithm ensures that, on average, the total resource utilization after $s^\star$ is close to $1 - 4\varepsilon$. The proof of the Claim relies on Lemma 2.7 and appears in Appendix 2.A.2.

Claim 2.9. *Let $B = [r, r+\tilde{s})$ with $r > s^\star$ be a consecutive block of $\tilde{s}$ timesteps and let $B' = \{t \in B : \sum_{j \in A_t} h_t(j) \leq 1 - \varepsilon\}$ be the time steps in B with low utilization. Then $|B'| \leq 4\varepsilon\tilde{s}$ and therefore, $\sum_{t=s^\star+1}^{T} \sum_i w_t(i) \geq (1-4\varepsilon)(T - s^\star) - \widetilde{s}$.*

Now, consider the following feasible dual solution of $(D_{4\varepsilon})$: $\gamma_t(i) = 1, \beta_t = 0$ for all users i and $t = 1, \ldots, s^\star$, and $\gamma_t(i) = 0, \beta_t = 1$ for all users i and $t = s^\star + 1, \ldots, T$. Observe that $\sum_{t=1}^{T} \beta_t = T - s^\star$. For optimal $(1 - 4\varepsilon)$-allocations $\mathbf{h}_1^* \ldots, \mathbf{h}_T^*$ we obtain

$$\begin{aligned}
\text{work}_{\mathbf{h}_1^*,\ldots,\mathbf{h}_T^*} &\leq v_{\text{dual}}(\gamma_1, \ldots, \gamma_T, \beta_1, \ldots \beta_T) && \text{(weak duality)} \\
&= \sum_{t=1}^{s^\star} \sum_i L_t(i) + (1 - 4\varepsilon)(T - s^\star) \\
&\leq \sum_{t=1}^{s^\star + \widetilde{s}} \sum_i w_t(i) + (1 - 4\varepsilon)(T - s^\star) && \text{(choice of } s^\star\text{)} \\
&\leq \sum_{t=1}^{s^\star} \sum_i w_t(i) + \widetilde{s} + \sum_{t=s^\star+1}^{T} \sum_i w_t(i) + \widetilde{s} && \text{(Claim 2.9)} \\
&= \text{work}_{\text{alg}} + 8\frac{N}{\varepsilon^2 \eta} \ln(N/\varepsilon).
\end{aligned}$$

where we have used $\sum_{t=s^\star+1}^{s^\star+\widetilde{s}} w_t(i) \leq \widetilde{s}$ and the definition of $\widetilde{s}$. □

2.3.3 SLA Satisfaction

In this section, we prove Theorem 2.5. Recall that $\lambda = \frac{\varepsilon^2}{8N}$ and $A_t^1 = \{i \in A_t : h_t(i) < \beta(i)\}$ is the set of active users receiving less than their SLAs and $A_t^2 = A_t \setminus A_t^1$ is the set of active user receiving at least their SLA. Analogous to Lemma 2.7, we have the following lemma, whose proof appears in Appendix 2.A.2.

Lemma 2.10. *Assume $\varepsilon \leq 1/10$, $\eta \leq 1/3$ and $\beta(i) \geq 2\frac{\varepsilon}{N}$ for all users. Then for any $i \in A_t^1$, Algorithm 1 guarantees $h_{t+1}(i) \geq h_t(i)(1 + c')$, where $c' = \frac{\varepsilon\eta\lambda}{2N}$.*

Proof of Theorem 2.5. Let $\widetilde{p} = \left\lceil \frac{\ln(N/\varepsilon)}{\ln(1+c')} \right\rceil$, where c' is defined in Lemma 2.10. Now, we proceed by induction on t to prove that $\sum_{\tau=1}^{t+\widetilde{p}} w_\tau(i) \geq \sum_{\tau=1}^{t} w'_\tau(i)$, where $\mathbf{w}'_t$ is the work done by the allocations $\mathbf{h}'_1, \ldots, \mathbf{h}'_T$. Clearly, the case $t = 0$ is direct.

Take $t \geq 1$ and suppose the result is true for $t - 1$. If there exists $r \in [t, t + \widetilde{p}]$ such that

user i's queue is empty, then

$$\sum_{\tau=1}^{t+\widetilde{p}} w_\tau(i) \geq \sum_{\tau=1}^{r} w_\tau(i) = \sum_{\tau=1}^{r} L_\tau(i) \geq \sum_{\tau=1}^{t} w'_\tau(i).$$

Therefore, assume that for all $\tau \in [t, t+\widetilde{p}]$ we have that user i's queue is non-empty. By the induction hypothesis

$$\sum_{\tau=1}^{t-1+\widetilde{p}} w_t(i) \geq \sum_{\tau=1}^{t-1} w'_\tau(i).$$

In order to complete the proof, we need to prove that $w_{t+\widetilde{p}}(i) \geq w'_t(i)$. We proceed as follows. Suppose that for all $\tau \geq t$ we have $w_\tau(i) < (1-\varepsilon)\beta(i)$. By Lemma 2.10, at each time $\tau \in [t, t+\widetilde{p}]$ the allocation of user i increases multiplicatively by a rate $(1+c')$. Therefore,

$$w_{t+\widetilde{p}}(i) \geq \frac{\varepsilon}{N}(1+c')^{\widetilde{p}} \geq 1 \geq \beta(i),$$

a contradiction. From the previous analysis we obtain the existence of $\tau^\star \in [t, t+\widetilde{p}]$ such that $w_{\tau^\star}(i) \geq (1-\varepsilon)\beta(i)$. By using Lemmas 2.7 and 2.10, we can show that the allocation $h_\tau(i)$ will never go below $(1-\varepsilon c)(1-\varepsilon)\beta(i)$ for all $\tau \geq \tau^\star$. In particular $w_{t+\widetilde{p}}(i) \geq (1-\varepsilon c)(1-\varepsilon)\beta(i) \geq (1-2\varepsilon)\beta(i) \geq w'_t(i)$. □

2.3.4 Extension to Proportionality and Over-commitment

In the previous subsections, we have introduced the first version of the multiplicative weights algorithm. We explained how we deduced our algorithm using mirror descent and proved its theoretical guarantees. Even though Algorithm 1 guarantees individual SLA satisfaction, this simple policy can lead to undesirable results that do not respect the ratio between allocations. If one user has an SLA twice the size of another, it would be reasonable for the former to expect allocations at least twice as big as the latter's if both are consistently busy. Likewise, the second user would expect allocations no less than half of the first user's. In other words, both users should expect shares that respect the ratio

between their SLAs.

To illustrate this unsatisfactory behavior in Algorithm 1, we run it with three users having SLAs $\beta(1) = 0.5$, $\beta(2) = 0.3$ and $\beta(3) = 0.2$. We set $\eta \leq 1/3$ and $\varepsilon \leq 1/10$. For simplicity, the initial allocation will be uniform. In our example, user 1 is always idle. User 2 consistently demands 1 unit of resource. User 3 begins idle and remains so until user 2's allocation reaches $1 - \varepsilon$. This takes roughly $\frac{1}{\eta\varepsilon}$ time steps; call this time t_0. Starting at time t_0, user 3 demands unit loads every time step for the rest of the horizon. Initially, the allocations are uniformly $1/3$ for everyone. Between time 1 and t_0, user 2's allocation increases until it hits $1 - \varepsilon$, since they are the only active user. After t_0, user 3 becomes active, and has an allocation below their SLA. Therefore, the algorithm redistributes allocation from user 2 to 3 until user 3's allocation hits 0.2. After this, allocations remain stable at approximately $h_t(1) = \frac{\varepsilon}{3}$, $h_t(2) = 0.8 - \frac{\varepsilon}{3}$ and $h_t(3) = 0.2$. User 2 receives about 4 times the allocation of user 3 if ε is small enough. However, a better allocation for user 2 and 3 is $\frac{\beta(2)}{\beta(2)+\beta(3)} = \frac{3}{5}$ and $\frac{\beta(3)}{\beta(2)+\beta(3)} = \frac{2}{5}$ respectively. These allocations reflect the ratio $\frac{\beta(2)}{\beta(3)}$ between active users.

Given this, we propose a slight modification of Algorithm 1, shown in Algorithm 2. As before, the plan is always to benefit active users. However, this time, we boost active users slightly more if they fall behind their "proportional SLA" among active users. Intuitively, if there are $n < N$ active users for a long period of time, the allocation of these active users should converge to their proportional share.

Algorithm 2: Extended Multiplicative Weight Update Algorithm

Input: Parameters $0 < \varepsilon \leq 1/10, 0 < \eta < 1/3$.

Initialization: $\mathbf{h}_1$ any allocation over Δ_ε and $\lambda = \frac{\varepsilon^2}{8N}$.

for $t = 1, \ldots, T$ **do**

  Set allocation $\mathbf{h}_t$.

  Read active and inactive users A_t and B_t.

  $A_t^1 = \left\{ i \in A_t : h_t(i) < (1-\varepsilon)\frac{\beta(i)}{\sum_{j \in A_t} \beta(j)} \right\}$, $A_t^2 = A_t \setminus A_t^1$.

  Set gain function $g_t(i) = \begin{cases} 1+\lambda & i \in A_t^1 \\ 1 & i \in A_t^2 \\ 0 & i \in B_t \end{cases}$.

  Update allocation: $\widehat{h}_{t+1}(i) = h_t(i) e^{\eta g_t(i)}, \forall i$ and $\mathbf{h}_{t+1} = \pi_{\Delta_\varepsilon}(\widehat{h}_{t+1})$

For technical reasons, the set A_t^1, the active users with allocation below their proportional share among active users at time t, has to be defined as

$$\left\{ i \in A_t : h_t(i) < (1-\varepsilon)\frac{\beta(i)}{\sum_{j \in A_t} \beta(j)} \right\}.$$

The reason behind this choice is to ensure that if $A_t^1 \neq \emptyset$ and the resource is nearly fully utilized, i.e., $\sum_{i \in A_t} h_t(i) \geq 1 - \varepsilon$, then there is a different active user $j \neq i$ from which we can move allocation to i. This is fundamental in the proof of Theorem 2.12 below.

In terms of work maximization and SLA satisfaction, Algorithm 2 provides exactly the same guarantees as Algorithm 1.

Theorem 2.11. *Given loads $L_1, \ldots, L_T$, for any $\varepsilon > 0$ and $\eta > 0$ such that $\varepsilon \leq 1/10$, Algorithm 2 guarantees*

$$\text{work}_{\mathbf{h}_1^*, \ldots, \mathbf{h}_t^*} - \text{work}_{alg,t} \leq 8\frac{N}{\varepsilon^2 \eta} \ln(N/\varepsilon),$$

for any time $1 \leq t \leq T$*, where* $\mathbf{h}_1^*, \ldots, \mathbf{h}_T^*$ *is an optimal offline sequence of* $(1-4\varepsilon)$*-allocations and* $\mathrm{work}_{alg,t} = \sum_i \sum_{\tau=1}^{t} w_\tau(i)$ *is the overall work done by Algorithm 2 until time* t*.*

The proof of Theorem 2.11 is exactly the same as the proof of Theorem 2.4. To see this, observe that the proof of Theorem 2.4 only uses the fact that the allocations of *every* active user get a multiplicative boost whenever the usage is below $1-\varepsilon$. The last statement is true since Lemma 2.7 also holds in this case.

For SLA satisfaction, we have the following stronger statement.

Theorem 2.12. *Let* $0 < \varepsilon \leq 1/10$*,* $0 < \eta \leq 1/3$*,* $\lambda = \frac{\varepsilon^2}{8N}$ *and* $\tilde{p} = 32\frac{N^2}{\varepsilon^3\eta}\ln(N/\varepsilon)$*. Take any SLAs* $\beta(1), \ldots, \beta(N)$ *such that* $\frac{\beta(i)}{\sum_k \beta(k)} \geq \frac{e^{\eta(1+\lambda)}}{1-\varepsilon}\frac{\varepsilon}{N}$*. Then, for any user* i *and time* t*, if we take* $\mathbf{h}'_1, \ldots, \mathbf{h}'_T$ *to be the allocations such that* $h'_t(i) = (1-2\varepsilon)\frac{\beta(i)}{\sum_k \beta(k)}$*, the work done by Algorithm 1 for user* i *satisfies*

$$\sum_{\tau=1}^{t+\tilde{p}} w_\tau(i) \geq \sum_{\tau=1}^{t} w'_\tau(i),$$

where $\mathbf{w}'_t$ *is the work done by the allocations* $\mathbf{h}'_1, \ldots, \mathbf{h}'_T$*. Moreover, we have* $\sum_{\tau=1}^{t} w_\tau(i) \geq \sum_{\tau=1}^{t} w'_\tau(i) - \frac{\beta(i)}{\sum_k \beta(k)}\tilde{p}$*.*

The proof of this result is similar to the proof of Theorem 2.5. A subtle difference is that the analogue of Lemma 2.10 holds if we add the hypothesis $\sum_{i\in A_t} h_t(i) > 1-\varepsilon$. We skip the proof for brevity.

Lemma 2.13. *Assume* $\varepsilon \leq 1/10$*,* $\eta \leq 1/3$ *and* $\frac{\beta(i)}{\sum_k \beta(k)} \geq \frac{e^{\eta(1+\lambda)}}{1-\varepsilon}\frac{\varepsilon}{N}$ *for all users. In Algorithm 2, if* $\sum_{k\in A_t} h_t(k) > 1-\varepsilon$ *then for any* $i \in A_t^1$ *we have* $h_{t+1}(i) \geq (1+c')h_t(i)$*, where* $c' = \frac{\varepsilon\eta\lambda}{2N}$*.*

Lemma 2.7 and 2.13 ensure that any active user gets a multiplicative boost of at least

$(1 + c')$. Therefore, any user that is active $\widetilde{p}$ consecutive times will have an allocation of at least $(1 - 2\varepsilon)\frac{\beta(i)}{\sum_k \beta(k)}$. Then, by following the same inductive proof of Theorem 2.5, we obtain Theorem 2.12.

If the resource is not over-committed, $\sum_k \beta(k) \leq 1$, this result implies that Algorithm 2 ensures for each user i an amount of work comparable with $\frac{\beta(i)}{\sum_k \beta(k)} \geq \beta(i)$; that is, we obtain the standard SLA satisfaction guarantee. In the over-commitment regime, $\sum_{i=1}^{N} \beta(i) > 1$, we do retain some performance guarantees. The update according to almost-normalized SLAs in Algorithm 2 still works and Theorem 2.11's work maximization guarantee still applies, as its proof does not rely on the SLAs. On the other hand, Theorem 2.12 states that individually, each user does work comparable to their normalized SLA. If the level of over-commitment is not large, each user is still guaranteed service "almost" at their SLA; for example, if the resource is over-committed by 10%, each user receives service comparable to $1.1^{-1} \approx 91\%$ of their SLA.

Another interesting byproduct of the work maximization guarantee is the following result.

Corollary 2.13.1. *Under the assumptions of Theorem 2.5, suppose there is a time $1 \leq \tau \leq T$ with $\sum_{t=1}^{\tau} \sum_i w'_\tau(i) = \sum_{t=1}^{\tau} \sum_i L_t(i)$; i.e. the optimal offline $(1-4\varepsilon)$-allocation is able to finish all work up until τ. Then, the sum of queue lengths at time τ induced by Algorithm 2 is at most $8\frac{N}{\varepsilon^2\eta}\ln(N/\varepsilon)$. In particular, at time τ each user's queue length is at most this value.*

In practical settings, it is commonplace to assume an arrival rate is lower than the work processing rate. In stochastic settings, stationary states cannot be achieved without this assumption; see, e.g., [165]. In our context, this can be reinterpreted as having times within the operating horizon where the optimal offline solution is able to finish all work arriving up until that time. At these particular times, the corollary guarantees that Algorithm 2's queue lengths are constant. In other words, the algorithm does not starve individual users

to achieve work maximization and keeps their queues short, an appealing property in cloud systems.

2.4 Experiments

In this section, we empirically test Algorithm 2 against a family of offline and online algorithms. We aim to measure the performance on both synthetic data as well as real CPU traces from a production service in Microsoft's cloud. We quantify the performance on the following three criteria.

- **Work maximization.** We compare the overall work done by Algorithm 2 against various benchmark algorithms.
- **SLA guarantee.** We examine the extent to which our algorithm achieves the cumulative work of the static SLA policy for each user. We do so by measuring the cumulative work over plausible time windows.
- **Queue length.** We compare the 2-norm of the individual queues over time. We use this metric as a proxy for the system latency, which is not captured by our theoretical results.

We consider the following *online algorithms*, against which we benchmark our algorithm.

- **Static SLA Policy (Static)**. Each user gets their SLA as a constant, static allocation. We call this algorithm Static.
- **Proportional Online (PO)**. In each iteration, every active user will get their SLA normalized by the sum of SLAs of active users (just their SLA if there are no active users). This simple algorithm seems suitable for a practical implementation; however, its performance can be arbitrarily bad in terms of work maximization. The formal description appears in Algorithm 4 in Appendix 2.A.7. We call this Algorithm

PO.

- **Online Work Maximizing (OWM)**. This algorithm divides users into three categories: A, B and I (active users with allocation, actives users without allocation and inactive users). At each iteration, the resource is divided uniformly among users in A. If a user in A becomes inactive, they are moved to I. If a user in I becomes active, they are moved to B. When A becomes empty, we move all users from B to A. In Appendix 2.A.5 we prove that this method is work maximizing. However, this greedy strategy is not guaranteed to satisfy SLA constraints for general input loads. We call this algorithm OWM.

We also consider the following *offline algorithms*, against which we benchmark our algorithm.

* **Optimal** 1**-allocations (PG)**. The optimal offline solution to the work maximization problem. This solution is computed using Algorithm 3, which we call Proportional Greedy (PG). This algorithm can be considered as the offline counterpart of Proportional Online.

* **Optimal** $(1-\varepsilon)$**-allocation (restPG)**. Offline solution to the work maximization problem with resource restricted to $1-\varepsilon$, where ε is the parameter of Algorithm 2.

2.4.1 Synthetic Experiment

Description of the Experiment

In this experiment, we consider a synthetically generated input sequence, which we use to examine how online algorithms adapt to different conditions. Specifically, our system consists of three users with SLAs of $\beta(1) = 0.2$, $\beta(2) = 0.3$ and $\beta(3) = 0.5$. We consider a time horizon of $T = 3{,}000{,}000$. The load input sequence is divided into six periods: $P_i = \left[\frac{(i-1)T}{6}, \frac{iT}{6}\right)$ for $i = 1, \ldots, 6$. In each period, only two users demand new resources.

During the first 3 periods, the random demand has a mean proportional to the users' SLA. In the following 3 periods, the random demand changes to a distribution with uniform mean among users demanding resources. Specifically:

- During P_1, only users 2 and 3 demand the following loads. At the beginning of P_1, i.e., $t = 1$, user 2 demands a large load of $L_1(2) \sim \frac{T}{6} \cdot \text{Gamma}\left(2000, \frac{1}{2000} \cdot \frac{\beta(2)}{\beta(2)+\beta(3)}\right)$ and $L_1(3) = 0$. During the rest of period P_1, $L_t(2) = 0$ and the other load will be $L_t(3) \sim \text{Gamma}\left(2000, \frac{1}{2000} \cdot \frac{\beta(3)}{\beta(2)+\beta(3)}\right)$. User 1 demands nothing during this entire period. Similar loads are set for period P_2 and P_3.

- Similarly, during P_4 users 2 and 3 demand $L_t(i) \sim \text{Gamma}\left(2000, \frac{1}{2000} \cdot \frac{1}{2}\right)$ for $i = 2, 3$. During P_5 users 1 and 2 demand $L_t(i) \sim \text{Gamma}\left(2000, \frac{1}{2000} \cdot \frac{1}{2}\right)$ for $i = 1, 2$. During P_6 users 1 and 3 demand $L_t(i) \sim \text{Gamma}\left(2000, \frac{1}{2000} \cdot \frac{1}{2}\right)$ for $i = 1, 3$.

The expectation of a Gamma(k, θ) random variable is given by $k\theta$ and the variance is given by $k\theta^2$ (see e.g. [73]). For example, in period P_1 user 2's expected load is $\frac{T}{6}\frac{\beta(2)}{\beta(2)+\beta(3)}$, with variance $\frac{1}{2000}\left(\frac{\beta(2)}{\beta(2)+\beta(3)}\right)^2$. Similarly, user 3's expected total load is $\frac{T}{6}\frac{\beta(3)}{\beta(2)+\beta(3)}$. Thus, the expected overall load is $T/6$, exactly the length of the period. A brief summary of Gamma distribution's properties is given in Appendix 2.A.8.

We instantiate Algorithm 2 with $\eta = \frac{1}{3}$, $\varepsilon = 0.02$ and $T = 3{,}000{,}000$.

Results

Work maximization. In Figure 2.3 we present the cumulative work difference between PG and Algorithm 2 (solid blue line with star), restPG and Algorithm 2 (red dashed line with triangle), PO and Algorithm 2 (solid magenta line), Static and Algorithm 2 (solid green line with small circle) and OWM and Algorithm 2 (solid cyan line with large circle). Intuitively, one positive unit of difference implies the corresponding algorithm is ahead of

Algorithm 2 by one unit of time.

First, we consider the comparison to online algorithms Static, PO and OWM. Algorithm 2 outperforms Static significantly, by roughly 700,000 units of time. During the first half of the experiment PO shows good performance, but in the second half of the experiment (when the load distribution changes), Algorithm 2 outperforms PO. This shows that Algorithm 2 can adapt to changing input sequences that PO cannot adapt to. Finally, OWM surpasses Algorithm 2 during the whole experiment, with a performance similar to PG; this is an expected result since OWM is work maximizing.

With respect to offline algorithms, PG outperforms Algorithm 2 by roughly 10,000 time units. On the other hand, Algorithm 2 surpasses restPG by approximately 20,000 units. This shows that Algorithm 2 indeed performs better than the offline optimum with a slightly reduced amount of resources, as guaranteed by the theoretical results.

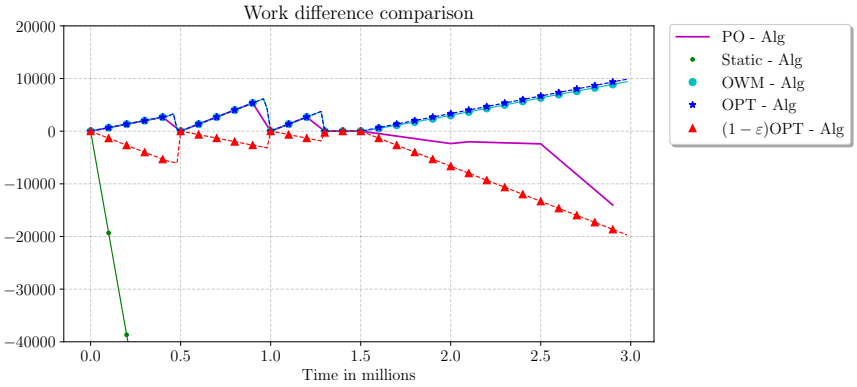

Figure 2.3: Difference between algorithms' work. Alg is short for Algorithm 2. Observe that the differences "OPT - Alg" "and "OWM - Alg" slightly overlap.

SLA satisfaction. Among online algorithms, Static and PO satisfy SLA restrictions by design, but as seen earlier they are not competitive in terms of work maximization. We focus on the comparison between OWM and Algorithm 2 as far as SLA satisfaction is concerned. Although OWM performs extremely well in work maximization, this comes at a significant price in SLA satisfaction. In Figure 2.4 we depict empirically this behavior by plotting the

instantaneous work done by users 2 and 3 by OWM and Algorithm 2 during period P_1. We empirically observe that Algorithm 2 approximately satisfies user 3's SLA, but OWM does not allocate the user any resources. Such an extreme behavior arises because OWM is geared towards work maximization rather than SLA satisfaction.

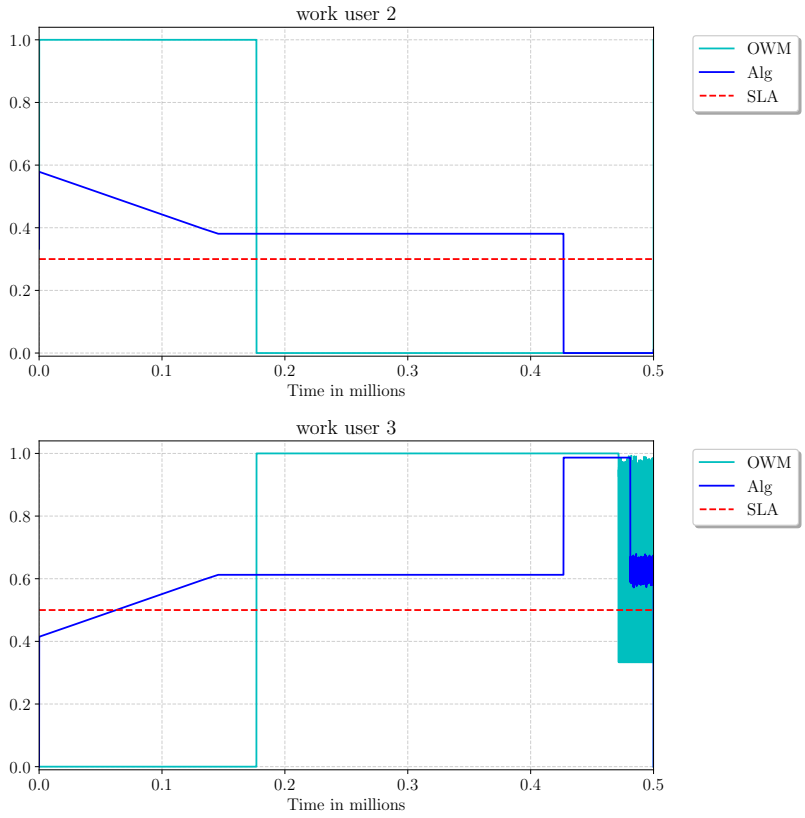

Figure 2.4: Instantaneous work for user 2 and 3 during period P_1. User 3 does not receive any allocation in OWM until user 2 finishes all of their work.

Queue lengths. In Figure 2.5 we present the 2-norm of queues induced by Algorithm 2 (solid blue), Static (solid magenta with star), PO (solid cyan with large circle) and OWM (solid black with triangle). Once again, we can interpret one unit of norm as one unit of latency. Experimentally, we observe that Static shows the worst performance with a final 2-norm of 524,000 units. Algorithm 2 ends with a 2-norm of 10,000 units, PO with 26,970 units, and OWM with 381 units. As remarked above, even though OWM induces very small queues, this comes at the cost of not satisfying SLA requirements.

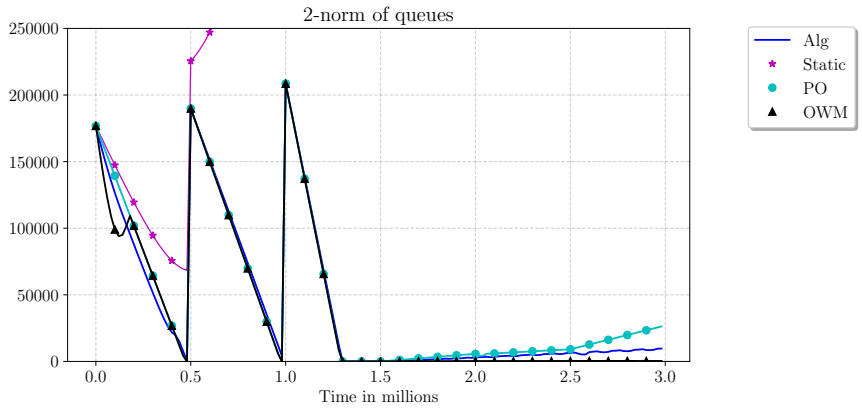

Figure 2.5: 2-norm of queues.

Summary. Among all online algorithms, Algorithm 2 is able to best balance work maximization and SLA satisfaction. In particular, Algorithm 2 is only slightly worse in terms of work maximization compared to OWM; this is expected since Algorithm 2 always reserves a small fraction of the resource for each user, regardless of activity. Furthermore, our results show that the actual total work done by Algorithm 2 is much better than the theoretical guarantee of Theorem 2.11, as it substantially outperforms $(1 - \varepsilon)$OPT (restPG). Finally, we observe small queues throughout the entire horizon, which is key for maintaining reasonable latencies.

2.4.2 Experiment with Real Data

We next describe the results of our computational study using real world data. For these experiments, we obtained CPU traces of a production service on Azure, Microsoft's public cloud. The data consists of demand traces of six different users over a time window of approximately ten days.

To show Algorithm 2's robustness with respect to (short-term) real data, we also consider the following measurement:

- **Instantaneous SLA.** We focus on a modified SLA satisfaction criterion because of the relatively short horizon (about $14,000$ minutes); we assess Algorithm 2's perfor-

mance in the following way. For a user i and any time t, we compare the cumulative work done by user i in Algorithm 2 during a time window $[t, t+\tau)$ versus the cumulative work done by the same user i under a Static SLA Policy during the time window $[t, t+\tau)$. In order to have a meaningful comparison, at time t, we run the Static SLA Policy with the queues of Algorithm 2 at time t. The motivating question is, *what happens if at time t and the next τ time steps we run the static policy instead of Algorithm 2?* For this experiment, we used $\tau = 500$ minutes.

The data set consists of demand traces of six users of exactly $14{,}628$ minutes (approximately ten days). Each user is assigned their normalized average workload as SLA. (Since the data is proprietary, we cannot disclose actual SLAs.) For the purpose of the experiment, we run Algorithm 2 with parameters $\varepsilon = 0.01$ and $\eta = \frac{1}{3}$.

Results

Work maximization. We depict in Figure 2.6 the following differences: cumulative work done until time t by optimal 1-allocations (PG) and Algorithm 2, $(1-\varepsilon)$-allocations (restPG) and Algorithm 2, PO and Algorithm 2, Static and Algorithm 2, and OWM and Algorithm 2. In a similar fashion to the previous experiment, one positive unit can be interpreted as Algorithm 2 being one unit (minute) of work behind, and one negative unit means Algorithm 2 is ahead by a minute.

We observe that Algorithm 2 outperforms all online benchmarks. Against Static, the final difference is 105 units, with a maximum difference of 167 units. Against PO the final difference is 25 units with a maximum difference of 37. Finally, against OWM, the final difference is 20 with a maximum difference of 21. Surprisingly, for this data set Algorithm 2 is able to surpass even OWM.

Regarding the offline algorithms, PG surpasses Algorithm 2 during the whole experiment as expected, with a final difference of 14 units. On the other hand, Algorithm 2 outperforms

restPG by 26 units by the end of the experiment.

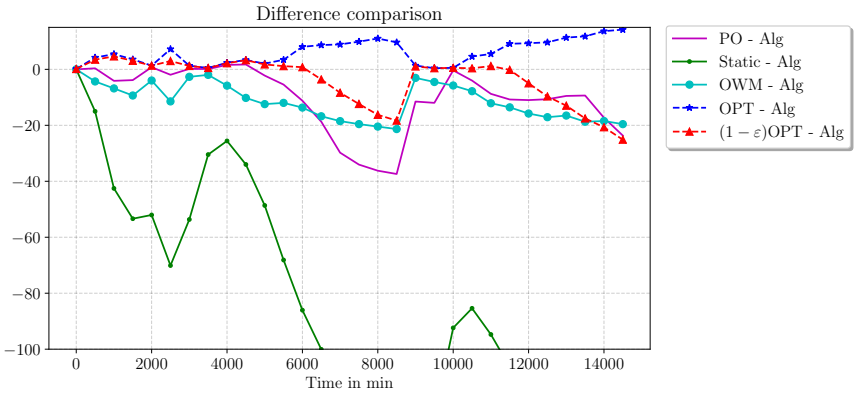

Figure 2.6: Difference of cumulative works.

Instantaneous SLA. In Table 2.1 we report statistics on the differences between the cumulative work done in time windows $[t, t+\tau)$ by Static and Algorithm 2 for each user. For each t and each user i, the exact formula is $r_i(t) = \sum_{r=t}^{t+\tau-1} w'_r(i) - \sum_{r=t}^{t+\tau-1} w_r(i)$, where $w_r(i)$ is the work done by user i under Algorithm 2 and $w'_r(i)$ is the work done by user i with Static (with queues at t given by Algorithm 2). The table shows the min, max, average and standard deviation of $\{r_i(t)\}_t$ when $\tau = 500$ minutes. A positive unit means Algorithm 2 is outperformed by Static during $[t, t+\tau)$ under the same initial conditions by one unit of time. In general, we observe that all users show negative empirical average difference. This result empirically suggests that Algorithm 2 ensures approximate SLA satisfaction, even for small time windows. For instance, user 3 occasionally has a high difference (46.4 units), mostly due to times t where Algorithm 2 allocates the user a small amount of resource but a huge load is incoming during the window $[t, t+\tau)$. The experiment tells us that averaging out these "bad" times ensures good performance under the SLA criterion. Furthermore, we tested values of $\tau = 60$, 500 and 1000 minutes; larger windows improve the results, with lower maximum and average values.

Queue lengths. In Figure 2.7 we present the 2-norms of queues given by the online benchmark algorithms and Algorithm 2. As usual, we can interpret one unit as the respective

Table 2.1: Statistics for the difference between the cumulative works of our algorithm and static over time windows $[t, t+\tau)$. For each user i we present the min, max, average, and standard deviation over the sequence of differences $\{r_i(t)\}_t$.

User	min	max	mean	std
User 1	-31.1	21.7	-4.6	13.8
User 2	-122.3	46.9	-31.2	49.2
User 3	-90.8	46.4	-7.2	29.5
User 4	-49.7	18.5	-0.8	12.9
User 5	-42.6	22.6	-5.85	14.0
User 6	-21.9	14.9	-0.6	6.4

algorithm's latency, that is, lateness with respect to the overall users' demand. Compared against the online algorithms, we empirically observe the superiority of Algorithm 2, as it has the smallest latency most of the time. Algorithm 2 ends with a 2-norm of roughly 44 units, average length of 22 and a maximum length of 92. PO ends with a 2-norm of approximately 68, average length of 32 and a maximum length of 126. OWM ends with a 2-norm of 68, average of 32 and maximum of 113. Finally, Static shows the worst behavior, with a final 2-norm of 103, average of 91 and maximum of 231. For this data set, Algorithm 2 shows a remarkable performance, considering particularly that Algorithm 2 always reserves ε/N resource for each user.

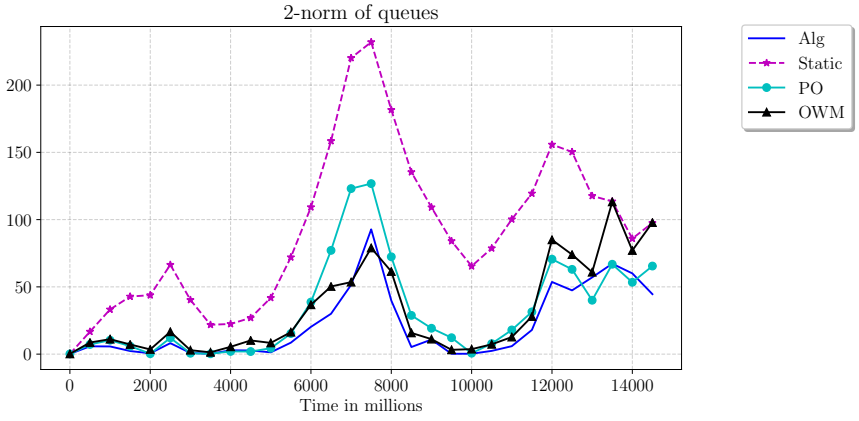

Figure 2.7: 2-norm of queues.

Summary. Algorithm 2 performs very well in terms of work maximization compared to all other online algorithms, and exceeds the theoretical guarantees. Furthermore, SLA re-

quirements are typically satisfied, even when measured over relatively short time windows. Finally, as in the previous experiment, the algorithm maintains small queues compared to other online algorithms.

2.5 Conclusion

We have proposed a new online model for dynamic resource allocation of a single divisible resource in a shared system. Our framework captures basic properties of cloud systems, including SLAs, limited system feedback and unpredictable (even adversarial) input sequences. We designed an algorithm that is near-optimal in terms of both work maximization and SLA satisfaction (Theorems 2.1, 2.11 and 2.12). Furthermore, our second algorithm, Algorithm 2, can be applied in an over-commitment regime with similar guarantees, which could be of additional merit for some applications. We derived a simple expression for the offline work maximization problem that allowed us to reinterpret the algorithm's dynamics as an approximate solution of the optimal (offline) work maximization LP. Numerical experiments show that our algorithm is indeed able to achieve a desirable trade-off between work maximization and SLA satisfaction. In particular, comparisons with offline algorithms (PG and restPG) indicate that our algorithm is empirically work maximizing. Further, unlike other plausible online algorithms, our algorithm is able to quickly adapt to unexpected changes in demand and still approximately satisfy the underlying user SLAs. Our model and results may be extended in various directions of interest to the operations research and cloud computing communities.

A natural extension for single-resource systems is to model priority among users. Typically, users with higher priority should be given resources before their lower-priority counterparts. A challenge in this setting is how to define the metric corresponding to work maximization. One possibility is to have different weights for different users, corresponding to their priority, and then to maximize the weighted total work while satisfying SLAs.

Directly extending our algorithms to this case means a user's multiplicative boost depends on their priority. However, our analysis in this chapter does not apply, as we use the fact that users' work is interchangeable, whereas the identity of who performs the work is critical in the prioritized case.

Another challenging extension is the management of multiple resources (e.g., CPU, I/O bandwidth, memory), where different users or jobs may require the resources in different proportions. This extension requires a fundamental redefinition of our model, where work done for a user is a function of the multiple resources allocated, and may also depend on a particular job's characteristics. In many real-world scenarios, a job's resource demands are often complementary, e.g. RAM and CPU usage. This observation may motivate a possible extension in which we still treat all users' loads as one-dimensional quantities, and the work performed by a user is a relatively simple function of their allocations, e.g., a concave non-decreasing function.

2.A Appendix

2.A.1 Projecting on Δ_ε

Proof of Proposition 2.3. Let $\mathbf{y} \in \mathbf{R}_+^N$. The projection of $\mathbf{y}$ on Δ_ε corresponds to the solution of the convex problem

$$(Q) \qquad \begin{array}{rcl} \min \ \sum_i x(i) \ln\left(\frac{x(i)}{y(i)}\right) & & \\ \sum_i x(i) & = & 1 \\ x(i) & \geq & \varepsilon/N \end{array}$$

Its Lagrangian (see [31]) is

$$\mathcal{L}(\mathbf{x}, \lambda, \mu) = \sum_i x(i) \ln\left(\frac{x(i)}{y(i)}\right) - \lambda\left(\sum_i x(i) - 1\right) - \sum_i \mu_i \left(x(i) - \frac{\varepsilon}{N}\right).$$

Using the FO conditions:

$$\forall i : x(i) = y(i)e^{\mu_i+\lambda-1} = y(i)e^{\mu_i}C.$$

and the SO conditions:

$$\mu_i \geq 0, \forall i, \quad \text{and} \quad x(i) > \frac{\varepsilon}{N} \implies \mu_i = 0.$$

Let $S = \{i : x(i) = \varepsilon/N\}$ and $T = [N] \setminus S$. Then, using $\sum_i x(i) = 1$ we obtain

$$e^{\lambda-1} = \frac{1 - \frac{\varepsilon}{N}|S|}{\sum_{i\in T} y(i)}.$$

This proves part (b). Now, suppose we have $y(1) \leq \cdots \leq y(N)$. If $i, j \in T$, then $x(i) = y(i)e^{\lambda-1}$ and $x(j) = y(j)e^{\lambda-1}$ and then

$$x(i) \leq x(j) \iff y(i) \leq y(j).$$

That is, in T the variables preserve their ordering.

If $i \in S$ and $j \in T$, then $y(i)e^{\lambda-1+\mu_i} = x(i) = \frac{\varepsilon}{N} < x(j) = y(j)e^{\lambda-1}$, which implies $y(i) < y(j)$ using that $\mu_i \geq 0$. Now, let $k = \min\{i \in T\}$ which is a well-defined number using constraint $\sum_{i=1}^{N} x(i) = 1$. We claim that for any $j \geq k$, $j \in T$, that is, T corresponds to the interval $[k, N]$. By contradiction, suppose that $j > k$ does not belong to T, then $y(j) < y(k)$ by previous calculus. However $y(j) \geq y(k)$ by the ordering of $\mathbf{y}$. A contradiction. With this, the algorithm to project is clear, we sort $\mathbf{y}$ and then we test increasingly the possible set $S = \{1, \ldots, k-1\}$ for $k = 1, \ldots, N$ and select the best candidate. This proves (a).

We say that S is *feasible* if there is a feasible solution $\mathbf{x}$ such that $S = \{i : x(i) = \varepsilon/N\}$.

In the following paragraphs we prove that the first feasible solution found in this process is the right one.

Observe that once $S = \{1, \ldots, k\}$ is feasible, then $S' = \{1, \ldots, j\}$ remains feasible for all $j \geq k$. Indeed, if $S = \{1, \ldots, k\}$ is feasible, then

$$1 = \frac{\varepsilon}{N} k + \sum_{i \in T} x(i).$$

Now, increasing S to $S' = \{1, \ldots, k+1\}$ means that we pick $x(k+1) > \frac{\varepsilon}{N}$ and we decrease it to $\frac{\varepsilon}{N}$. Therefore, $x(k+2), \ldots, x(N)$ must increase. Therefore, S' remains feasible. The proof for general case $j \geq k$ follows by induction.

Now, we claim that if $S = \{1, \ldots, k\}$ is feasible, then $S' = \{1, \ldots, k+1\}$ cannot have better optimal value. Indeed, the difference between the objective S' and S is

$$\begin{aligned}
&\sum_{i=1}^{k+1} \frac{\varepsilon}{N} \ln \frac{\varepsilon}{N y(i)} + \Big(1 - \frac{\varepsilon}{N}(k+1)\Big) \ln \frac{1 - \frac{\varepsilon}{N}(k+1)}{\sum_{i \geq k+2} y(i)} - \sum_{i=1}^{k} \frac{\varepsilon}{N} \ln \frac{\varepsilon}{N y(i)} \\
&\quad - \Big(1 - \frac{\varepsilon}{N} k\Big) \ln \frac{1 - \frac{\varepsilon}{N} k}{\sum_{i \geq k+1} y(i)} \\
=&\frac{\varepsilon}{N} \ln \frac{\varepsilon}{N y(k+1)} + \Big(1 - \frac{\varepsilon}{N}(k+1)\Big) \ln \frac{1 - \frac{\varepsilon}{N}(k+1)}{\sum_{i \geq k+2} y(i)} - \Big(1 - \frac{\varepsilon}{N} k\Big) \ln \frac{1 - \frac{\varepsilon}{N} k}{\sum_{i \geq k+1} y(i)}
\end{aligned}$$

The function $f(x) = x \ln x$ is convex for $x > 0$. Now, pick $x = \frac{\varepsilon}{N y(k+1)}$, $y = \frac{1 - \frac{\varepsilon}{N}(k+1)}{\sum_{i \geq k+2} y(i)}$ and $\lambda = \frac{y(k+1)}{\sum_{i \geq k+1} y(i)}$. Then

$$\begin{aligned}
\lambda x + (1-\lambda) y &= \frac{y(k+1)}{\sum_{i \geq k+1} y(i)} \left(\frac{\varepsilon}{N y(k+1)} \right) + \frac{\sum_{i \geq k+2} y(i)}{\sum_{i \geq k+1} y(i)} \left(\frac{1 - \frac{\varepsilon}{N}(k+1)}{\sum_{i \geq k+2} y(i)} \right) \\
&= \frac{1 - \frac{\varepsilon}{N} k}{\sum_{i \geq k+1} y(i).}
\end{aligned}$$

Then, using the convexity of f we obtain the result. This implies that the first feasible prefix S that we find is the optimal one. Therefore, by ordering $\mathbf{y}$ in $\mathcal{O}(N \log N)$ time and then running binary search we can find S in $\mathcal{O}(N \log N)$ time. This finishes the proof of

(c). □

2.A.2 Missing Proofs From Section 2.3.1

Here we present dual stated in the offline formulation of the maximum work problem. We have the LP

$$
\begin{array}{lrcll}
 & \max & \sum_{i=1}^{N}\sum_{t=1}^{T} w_t(i) & & \\
(P_\varepsilon) & \sum_{s=1}^{t} w_s(i) & \leq & \sum_{s=1}^{t} L_s(i) & \forall t,i \quad (1)\\
 & \sum_{i=1}^{N} w_t(i) & \leq & 1-\varepsilon & \forall t \quad (2)\\
 & \mathbf{w}_t & \geq & 0 & \forall t
\end{array}
$$

Using the variables $\alpha_t(i)$ for constraint (1) and β_t for constraint (2) we obtain the dual

$$
\begin{array}{lrcll}
 & \min & \sum_{i=1}^{N}\sum_{t=1}^{T} \alpha_t(i)\sum_{s=1}^{t} L_s(i) + (1-\varepsilon)\sum_{t=1}^{T}\beta_t & & \\
(D_\varepsilon) & \sum_{s=t}^{T}\alpha_s(i) + \beta_t & \geq & 1 & \forall t,i \quad (1')\\
 & \alpha,\beta & \geq & 0 &
\end{array}
$$

Using the change of variable $\gamma_t(i) = \sum_{s=t}^{T}\alpha_s(i)$ we obtain the stated dual

$$
\begin{array}{lrcll}
 & \min & \sum_{i=1}^{N}\sum_{t=1}^{T} L_t(i)\gamma_t(i) + (1-\varepsilon)\sum_{t=1}^{T}\beta_t & & \\
 & \gamma_t(i) + \beta_t & \geq & 1 & \forall t,i \quad (1')\\
(D_\varepsilon) & \gamma_t & \geq & \gamma_{t+1} & \forall t \quad (2')\\
 & \beta,\gamma & \geq & 0 &
\end{array}
$$

Proof of Proposition 2.6. We prove each inequality separately. Let $0 \leq t^\star \leq T$ be such that $\sum_{s=1}^{t^\star}\sum_i L_s(i) + (1-\varepsilon)(T-t^\star) = \min_{0\leq t\leq T}\sum_{s=1}^{t}\sum_i L_s(i) + (1-\varepsilon)(T-t)$. Consider the dual solution (β,γ) such that $\gamma_t = 1$, $\beta_t = 0$ for $t = 1,\ldots,t^\star$ and $\gamma_t = 0, \beta_t = 1$ for $t = t^\star+1,\ldots,T$. Then, by weak duality,

$$
v_{P_\varepsilon} \leq v_{\text{dual}}(\beta,\gamma) = \sum_{s=1}^{t^\star}\sum_i L_s(i) + (1-\varepsilon)(T-t^\star) = \min_{0\leq t\leq T}\sum_{s=1}^{t}\sum_i L_s(i) + (1-\varepsilon)(T-t).
$$

Now, consider the greedy algorithm that, in each iteration, gives enough allocation to the users in order to complete their work starting with user 1, then user 2, and so on. We restrict the algorithms' allocations to $(1-\varepsilon)$-allocations. We denote by $\text{work}_{\text{greedy}}$ the work done by this algorithm. As usual, we denote by $\mathbf{w}_t$ the vector of work done at time t. Let $t^\star$ be the maximum non-negative t such that $\sum_i w_t(i) < 1-\varepsilon$. Observe that $\sum_{s=1}^{t^\star}\sum_i w_s(i) = \sum_{s=1}^{t^\star}\sum_i L_s(i)$. Then

$$\min_{0\le t\le T}\sum_{s=1}^{t}\sum_i L_s(i)+(1-\varepsilon)(T-t) \le \sum_{s=1}^{t^\star}\sum_i L_s(i)+(1-\varepsilon)(T-t^\star) = \text{work}_{\text{greedy}} \le v_{P_\varepsilon},$$

since v_{P_ε} is the optimal solution. □

Remark 1. This max-min result shows that the greedy algorithm is optimal for solving (P_ε) and also shows how to compute the dual variables. Finally, solving (P_ε) can be done efficiently in $\mathcal{O}(NT)$ by running the greedy algorithm.

2.A.3 Missing Proofs From Section 2.3.2

In what follows, we denote by S_t the users with allocation $\frac{\varepsilon}{N}$ at time t.

Proof of Lemma 2.7. 1. First, for $i \in A_t$ we have

$$h_{t+1}(i) = \frac{h_t(i)e^{\eta g_t(i)}e^{\mu_i}\left(1-\frac{\varepsilon}{N}|S_{t+1}|\right)}{\sum_{j\in\overline{S}_{t+1}}\widehat{h}_{t+1}(j)} \ge h_t(i)\frac{\left(1-\frac{\varepsilon}{N}|S_{t+1}|\right)}{e^{-\eta}\sum_{j\in\overline{S}_{t+1}}\widehat{h}_{t+1}(j)}.$$

We divide the analysis into two cases: $B_t \cap \overline{S}_{t+1} \neq \emptyset$ and $B_t \subseteq S_{t+1}$.

For $B_t \cap \overline{S}_{t+1} \neq \emptyset$ we have

$$\begin{aligned}
e^{-\eta} \sum_{j \in \overline{S}_{t+1}} \widehat{h}_{t+1}(j) &\leq e^{\lambda\eta} \sum_{j \in \overline{S}_{t+1} \cap A_t} h_t(j) + e^{-\eta} \sum_{j \in \overline{S}_{t+1} \cap B_t} h_t(j) \\
&= e^{\lambda\eta} \sum_{j \in \overline{S}_{t+1}} h_t(j) - (e^{\lambda\eta} - e^{-\eta}) \sum_{j \in B_t \cap \overline{S}_{t+1}} h_t(j) \\
&\leq e^{\lambda\eta} \left(1 - \frac{\varepsilon}{N}|S_{t+1}|\right) - \frac{\varepsilon}{N}(e^{\lambda\eta} - e^{-\eta}) \qquad \text{(since } h_t(i) \geq \tfrac{\varepsilon}{N}\text{)} \\
&\leq (1 + 2\lambda\eta) \left(1 - \frac{\varepsilon}{N}|S_{t+1}|\right) - \frac{\varepsilon}{N} \left(\lambda\eta + \eta - \frac{\eta^2}{2}\right) \\
&\qquad \text{(using } 1 + \lambda\eta \leq e^{\lambda\eta} \leq 1 + 2\lambda\eta, e^{-\eta} \leq 1 - \eta + \tfrac{\eta^2}{2}\text{)} \\
&\leq 1 - \frac{\varepsilon}{N}|S_{t+1}| + 2\lambda\eta - \frac{\varepsilon}{N}\eta \left(1 - \frac{\eta}{2}\right) \\
&\leq 1 - \frac{\varepsilon}{N}|S_{t+1}| - \frac{\varepsilon\eta}{4N}. \qquad \text{(} \eta \leq 1 \text{ and } 2\lambda \leq \tfrac{\varepsilon}{4N}\text{)}
\end{aligned}$$

Therefore,

$$\frac{\left(1 - \frac{\varepsilon}{N}|S_{t+1}|\right)}{e^{-\eta} \sum_{j \in \overline{S}_{t+1}} \widehat{h}_{t+1}(j)} \geq \frac{1 - \frac{\varepsilon}{N}|S_{t+1}|}{1 - \frac{\varepsilon}{N}|S_{t+1}| - \frac{\varepsilon\eta}{4N}} \geq \frac{1}{1 - \frac{\varepsilon\eta}{4N}} \geq 1 + \frac{\varepsilon\eta}{4N},$$

using $\frac{1}{1-x} \geq 1 + x$ when $x \in (0, 1)$. Hence $h_{t+1}(i) \geq h_t(i)(1 + \frac{\varepsilon\eta}{4N})$.

Now, if $B_t \subseteq S_{t+1}$, then $\overline{S}_{t+1} \subseteq A_t$. We have

$$\begin{aligned}
\frac{e^{-\eta} \sum_{j \in \overline{S}_{t+1}} \widehat{h}_{t+1}(j)}{1 - \frac{\varepsilon}{N}|S_{t+1}|} &\leq \frac{(1 - \varepsilon)e^{\lambda\eta}}{1 - \frac{\varepsilon}{N}|S_{t+1}|} \\
&\leq \frac{(1 - \varepsilon)e^{\lambda\eta}}{1 - \varepsilon + \frac{\varepsilon}{N}} \\
&\leq \frac{(1 - \varepsilon)(1 + 2\lambda\eta)}{1 - \varepsilon + \frac{\varepsilon}{N}} \qquad (e^{\lambda\eta} \leq 1 + 2\lambda\eta \text{ since } 2\lambda\eta < 1) \\
&\leq \frac{1 - \varepsilon + 3\lambda\eta}{1 - \varepsilon + \frac{\varepsilon}{N}} = 1 - \frac{\frac{\varepsilon}{N} + 3\lambda\eta}{1 - \varepsilon + \frac{\varepsilon}{N}}.
\end{aligned}$$

Since $\frac{\frac{\varepsilon}{N}+3\lambda\eta}{1-\varepsilon+\frac{\varepsilon}{N}} \geq \frac{\varepsilon}{N}$ we obtain

$$h_{t+1}(i) \geq h_t(i)\frac{1}{1-\frac{\varepsilon}{N}} \geq h_t(i)\left(1+\frac{\varepsilon}{N}\right) \geq h_t(i)\left(1+\frac{\varepsilon\eta}{4N}\right).$$

2. The monotonicity of $h_t(i)$ with $i \in A_t^1$ is easy to see. Let us prove the second statement:

$$\begin{aligned}
\frac{e^{-\eta}\sum_{j\in\overline{S}_{t+1}}\widehat{h}_{t+1}(j)}{1-\frac{\varepsilon}{N}|S_{t+1}|} &\leq \frac{e^{\lambda\eta}\sum_{j\in\overline{S}_{t+1}}h_t(j)}{1-\frac{\varepsilon}{N}|S_{t+1}|} \\
&\leq \frac{e^{\lambda\eta}\left(1-\frac{\varepsilon}{N}|S_{t+1}|\right)}{1-\frac{\varepsilon}{N}|S_{t+1}|} \\
&\leq e^{\lambda\eta} \\
&\leq 1+2\lambda\eta = 1+\varepsilon c,
\end{aligned}$$

since $\lambda = \frac{\varepsilon^2}{8N}$. Then, for $i \in A_t^2$,

$$h_{t+1}(i) \geq h_t(i)\frac{e^{\eta}(1-\frac{\varepsilon}{N}|S_{t+1}|)}{\sum_{j\in\overline{S}_{t+1}}\widehat{h}_{t+1}(j)} \geq h_t(i)\frac{1}{1+\varepsilon c} \geq h_t(i)(1-\varepsilon c).$$

□

Proof of Claim 2.9. Now, let $[s^\star+1,\ldots,T]$ and let us divide this interval into blocks of length $\widetilde{s}$ with a possible last piece of length of length at most $\widetilde{s}$. Let L be one of these blocks and let i be the user given by claim 2.8, that is, $M_{i,r}=1$ for all $r \in L$. Consider $L' = \{t \in L : \sum_{j\in A_t} h_t(j) < 1-\varepsilon\}$. By using part 1 of Lemma 2.7, user i increases her allocation multiplicatively in L' by a factor of $(1+c)$. Observe that for $t \notin L'$, user's i allocation can increase or decrease depending on $h_t(i)$. However, by Lemma by part 2 of 2.7, we know that $h_t(i)$ will not decrease by a huge amount. Let $k' = |L'|$, then i increases her allocation for k' times and decreases it for at most $\widetilde{s}-k'$ times. Therefore, k' maximum

value is such that

$$\frac{\varepsilon}{N}(1+c)^{k'}(1-\varepsilon c)^{\widetilde{s}-k'}=1$$

and therefore,

$$\begin{aligned}
k' &\leq \frac{\ln(N/\varepsilon)+\widetilde{s}\ln(1-\varepsilon c)^{-1}}{\ln((1+c)/(1-\varepsilon c))} \\
&\leq \frac{1+c}{c(1+\varepsilon)}\ln(N/\varepsilon)+\frac{\varepsilon(1+c)}{(1+\varepsilon)(1-\varepsilon c)}\widetilde{s} && (\ln\tfrac{1+c}{1-\varepsilon c}\geq\tfrac{c(1+\varepsilon)}{1+c},\ln\tfrac{1}{1-\varepsilon c}\leq\tfrac{\varepsilon c}{1-\varepsilon c}) \\
&\leq \frac{\varepsilon(1+c)}{(1+\varepsilon)}\widetilde{s}+\frac{\varepsilon(1+c)}{(1+\varepsilon)(1-\varepsilon c)}\widetilde{s} && (\widetilde{s}=\tfrac{\ln(N/\varepsilon)}{\varepsilon c}) \\
&= \varepsilon\frac{1+c}{1+\varepsilon}\left(1+\frac{1}{1-\varepsilon c}\right)\widetilde{s} \\
&\leq 3\varepsilon\widetilde{s}. && (\text{for } N\geq 2 \text{ and } \varepsilon\leq\tfrac{1}{10})
\end{aligned}$$

Hence, L' is at most a fraction of $\widetilde{s}$ and with this

$$\sum_{t\in L}\sum_{i} w_t(i)\geq(1-\varepsilon)(\widetilde{s}-k')\geq(1-\varepsilon)\,(1-3\varepsilon)\,\widetilde{s}\geq(1-4\varepsilon)|L|.$$

Summing over all blocks we conclude the desired result. □

2.A.4 Missing Proof From Section 2.3.3

Proof of Lemma 2.10. If $A_t^1=\emptyset$, the result is vacuously true. Suppose that $A_t^1\neq\emptyset$. First, we prove that under the assumption of Lemma 2.10, we have $\overline{A}_t^1\cap\overline{S}_{t+1}\neq\emptyset$. For $j\in S_{t+1}$ we have

$$\begin{aligned}
\frac{\varepsilon}{N} &= h_{t+1}(j) \\
&= \frac{\widehat{h}_{t+1}(j)e^{\mu_i}(1-\frac{\varepsilon}{N}|S_{t+1}|)}{\sum_{k\in\overline{S}_{t+1}}\widehat{h}_{t+1}(k)} \\
&\geq \frac{h_t(j)(1-\frac{\varepsilon}{N}|S_{t+1}|)}{\sum_{k\in\overline{S}_{t+1}}h_t(k)e^{\eta(1+\lambda)}}, && (\text{since } \mu_i\geq 0)
\end{aligned}$$

This implies $h_t(j) \leq \frac{\varepsilon}{N} e^{\eta(1+\lambda)} < \beta(j)$. Since $\sum_j h_t(j) = 1$, $i \in A_t^1$ and $\sum_j \beta(j) = 1$ we must have that there is a user $j \neq i$ with allocation $h_t(j) \geq \beta(j)$. Clearly, $j \notin S_{t+1}$ and $j \notin A_t^1$. Therefore, $\overline{A}_t^1 \cap \overline{S}_{t+1} \neq \emptyset$.

Following the proof of Lemma 2.7, for $i \in A_t^1$, we have

$$\begin{aligned}
e^{-\eta(1+\lambda)} \sum_{j \in \overline{S}_{t+1}} \widehat{h}_{t+1}(j) &\leq \sum_{j \in \overline{S}_{t+1} \cap A_t^1} h_t(j) + e^{-\eta\lambda} \sum_{j \in \overline{S}_{t+1} \cap \overline{A}_t^1} h_t(j) \\
&= \sum_{j \in \overline{S}_{t+1}} h_t(j) - (1 - e^{-\lambda\eta}) \sum_{j \in \overline{S}_{t+1} \cap \overline{A}_t^1} h_t(j) \\
&\leq 1 - \frac{\varepsilon}{N} |S_{t+1}| - \frac{\varepsilon}{N}(1 - e^{-\lambda\eta}) \qquad \text{(since } \overline{S}_{t+1} \cap \overline{A}_t^1 \neq \emptyset\text{)} \\
&\leq 1 - \frac{\varepsilon}{N} |S_{t+1}| - \frac{\varepsilon\lambda\eta}{2N}. \qquad (1 - e^{-x} \geq \tfrac{x}{2} \text{ for } x \in [0,1])
\end{aligned}$$

Therefore,

$$h_{t+1}(i) \geq h_t(i) \frac{1 - \frac{\varepsilon}{N}|S_{t+1}|}{1 - \frac{\varepsilon}{N}|S_{t+1}| - \frac{\varepsilon\lambda\eta}{2N}} \geq h_t(i) \left(1 + \frac{\varepsilon\lambda\eta}{2N}\right).$$

□

2.A.5 Greedy Online Algorithm

In this section, we prove that the following greedy allocation strategy is almost optimal in work maximization. The algorithm divides the users into 3 categories: A non-empty queue users with non-zero allocation, B non-empty queue users with zero allocation and I empty queue users with zero allocation. At time t, a user $i \in A$ is left in A if she still has non-empty queue, otherwise we will move her to I; a user $i \in I$ will be moved to B if her queue is becomes non-empty, otherwise she will remain in I; finally, if all users from A are moved to I, then we will move all B to A, otherwise we will left B untouched. In any case, we will distribute uniformly among the users that remain in A.

Users move from A to I, I to B and B to A. Let $\mathbf{w}_t$ be the work done by the algorithm and

let $\mathbf{w}'_t$ be the optimal offline work.

Theorem 2.14. *For any loads $L_1, \ldots, L_T \in \mathbf{R}^N_{\geq 0}$, and any $\varepsilon > 0$, this greedy Algorithm guarantees*

$$\sum_{t=1}^{T} \sum_i w_t(i) + 2\frac{N^2}{\varepsilon} \geq \sum_{t=1}^{T} \sum_i w'_t(i)$$

where $\mathbf{w}'_t$ is the work done by the optimal offline sequence of $(1 - 2\varepsilon/N)$-allocations.

Proof. Let $t^\star$ be the maximum $t \geq 0$ such that $\sum_{s=1}^{t+N^2} \sum_i w_s(i) \geq \sum_{s=1}^{t} \sum_i L_s(i)$. By claim 2.8 we know that each interval $[r, r + N^2/\varepsilon)$, with $r > t^\star$, has a user with no-empty queue. As in claim 2.9 we divide the interval $[t^\star + 1, T]$ into blocks of length N^2/ε with a last block of length at most N^2/ε. Pick any of these blocks, say L, and let $L' = \{t \in L : \sum_i w_t(i) < 1\}$. It is easy to see that $|L'| \leq 2N$ and therefore, summing over all block, we have $\sum_{t \geq t^\star+1} \sum_i w_t(i) + N^2/\varepsilon \geq (T - t^\star)(1 - 2\varepsilon/N)$. The conclusion follows applying weak duality to $(P_{2\varepsilon/N})$. □

Remark 1. Against the best 1-allocations we can optimize ε and obtain $\varepsilon = \sqrt{N^3/T}$. This greedy strategy will be $\mathcal{O}(\sqrt{NT})$ far from the optimal dynamic work. Observe that this matches the lower bound in Theorem 2.15.

2.A.6 Lower Bound

Theorem 2.15. *For any online deterministic algorithm $\mathcal{A}$ setting at each time 1-allocations, with an underlying queuing system, and with the same limited feedback as Algorithm 1, there exists a sequence of online loads $L_1, \ldots, L_T$ such that $\mathrm{work}_{\mathbf{h}^*_1, \ldots, \mathbf{h}^*_T} - \mathrm{work}_{\mathcal{A}} = \Omega\left(\sqrt{T}\right)$, where $\mathbf{h}^*_1, \ldots, \mathbf{h}^*_T$ are the optimal offline dynamic 1-allocations.*

Proof. We consider the case with $N = 2$ users, the general case reduces to $N = 2$ by loading jobs only to two users. Let $\mathcal{A}$ be an online algorithm for allocating a divisible

resource for 2 users and with underlying queuing system and limited feedback. Without loss of generality, we can assume that the allocations sets by $\mathcal{A}$ always add up to 1 at every time step.

We will construct a sequence of loads $L_t = (L_t(1), L_t(2))$ that at every time will add up to 1. This will ensure that the overall work done by the optimal offline dynamic policy will be T. On the other hand, we will show that this sequence of loads will lead to large queue length for at least one of the users. The main ingredient is to use the fact the algorithm receives limited feedback about the state of the system, i.e., which users have empty queue. In particular, this implies that if there are two distinct set of load vectors L_t and L'_t for some interval $t \in [r, s]$ such that the queues remain non-empty on both these sequences, then the resource allocation to the users in the two load sequences must be identical.

We will divide the time window $[1, T]$ into phases. Each phase will begin with a configuration of queues, say $Q = (Q(1), Q(2))$, where one of the queues is empty and the other one nonempty. We set $q = Q(1) + Q(2)$ and we denote by q_i the q at phase i. We define $q_0 = 0$. We will prove that at the end of each phase $i \geq 1$, $q_{i+1} \geq q_i + \frac{1}{4}$ with all q_{i+1} cumulated in one queue and the other queue empty.

Initially, the algorithm has a fixed deterministic allocation $\mathbf{h}_1 = (h_1(1), h_1(2))$. If $h_1(1) \leq h_1(2)$, then we load $L_1 = (0, 1)$. Otherwise, we load $L_1 = (1, 0)$. In any case, we have $q_1 \geq \frac{1}{2}$ and all q_1 in one queue.

Now, we will describe how the general phases work. For the sake of simplicity, we will describe the phase starting at time $t = 1$. We have queue configuration $Q = (Q(1), Q(2))$ with $q > 0$. By the initial phase, we can assume $q \geq 1/2$. Moreover, we can assume that only one of the queues is nonempty, this point will be clear after we describe how the phase works and it is clearly true for phase 1. Phase i with $q = q_i$ will last at most $2q + 2$ time steps.

Suppose that $Q(1) = 0$ and $Q(2) > 0$. If $h_1(1) = 1$, then we load $L_1 = (0, 1)$ and the phase ends with q increased by 1 and user 1's queue empty. Therefore, we can assume $h_1(1) < 1$. Our first load will be $L_1 = (h_1(1) + \varepsilon, h_2(2) - \varepsilon)$ with $0 < \varepsilon < \frac{1}{4}$ small enough and such that both queues are nonempty. The following loads will be $L_t = \mathbf{h}_t$, the allocation of $\mathcal{A}$ at time t. Observe that the first load will ensure that both users see nonempty queues until the end of the phase. Moreover, user 1 always has exactly ε remaining in her queue.

- If there is a time $\tau^\star \in [1, 2q + 1]$ such that $h_{\tau^\star}(1) \geq 1/2$, then we change the load at time $\tau^\star$ for $L'_{\tau^\star} = (0, 1)$. This will increment q by at least $1/2 - \varepsilon \geq 1/4$ and the phase ends. Observe that $Q_{\tau^\star+1}(1) = 0$.

- We can assume now that for the loads $L_t = \mathbf{h}_t$ we always have $h_t(1) < 1/2$ for all $t \in [2, 2q + 1]$. We change the loads to $L'_t = (1, 0)$ until time $\tau^\star$ in which user 2 empties her queue. Recall that the feedback of the algorithm is only the set of empty queues at every time step. Thus the behavior of $\mathcal{A}$ under L_t and L'_t will be the same until time $\tau^\star$. Now, we change load $L_{\tau^\star}$ by $L'_{\tau^\star} = (1-h_{\tau^\star}(2)+Q_{\tau^\star}(2)-\varepsilon', h_{\tau^\star}(2)-Q_{\tau^\star}(2)+\varepsilon')$ with $0 < \varepsilon' < 1/4$ small enough. This will ensure that queue 2 will be exactly ε'. Now, in an extra step, we load $L_{\tau^\star+1} = (1, 0)$. Again, we have q increased by at least $1/4$ and this ends the phase. Observe that $Q_{\tau^\star+2}(2) = 0$.

The analysis is similar for $Q(1) > 0$ and $Q(2) = 0$. Observe that at the end of each phase, only one queue is nonempty and the other one is empty. In any case, we have the desired increment. With this, we can set the following recurrence, $q_0 = 0, \quad q_{i-1} + \frac{1}{4} \leq q_i \quad \forall i \geq 1$. We deduce that $q_i \geq i/4$. Now, let m be the number of phases. By construction, each phase last at most $2q_i + 2$. Then $T \leq \sum_{i=1}^{m}(2q_i + 2) \leq 40q_m^2$, where we have used $4q_m \geq 4q_i \geq i \geq 1$. From here we deduce that $q_m \geq \sqrt{T/40}$.

Now, the work done by the algorithm and the unfulfilled work in the queues must add up the overall load. Then $q_m + \text{work}_{\mathcal{A}} = T = \text{work}_{\mathbf{h}_1^*,\ldots,\mathbf{h}_T^*}$ from which we obtain the result. $\square$

2.A.7 Additional Algorithms

Algorithm 3: Proportional Greedy

Input: Sequence of loads $(L_t)_{t=1}^T$ and SLAs $\beta(1), \ldots, \beta(N)$.

for $t = 1, \ldots, T$ **do**
  $w_t(i) \leftarrow 0, \forall i \in [N]$.
  $\mathrm{Rem} \leftarrow 1$.
  $\mathrm{Rem}(i) \leftarrow Q_{t-1}(i) + L_t(i), \forall i \in [N]$.
  repeat
    $A \leftarrow \{i : \mathrm{Rem}(i) > 0\}$.
    $\Sigma \leftarrow \sum_{i \in A_t} \beta(i)$.
    $i^* \leftarrow \mathrm{argmin}_{k \in A} \mathrm{Rem}(k)$.
    if $\mathrm{Rem}(i) < \frac{\beta(i)}{\Sigma}\mathrm{Rem}$ **then**
      $w_t(i^*) \leftarrow w_t(i^*) + \mathrm{Rem}(i^*)$.
      $\mathrm{Rem} \leftarrow \mathrm{Rem} - \mathrm{Rem}(i^*)$.
      $\mathrm{Rem}(i^*) \leftarrow 0$.
    else
      for $k \in A$ **do**
        $w_t(k) \leftarrow w_t(k) + \frac{\beta(k)}{\Sigma}\mathrm{Rem}$.
        $\mathrm{Rem}(k) \leftarrow \mathrm{Rem}(k) - \frac{\beta(k)}{\Sigma}\mathrm{Rem}$.
      $\mathrm{Rem} \leftarrow 0$.
  until $\mathrm{Rem} = 0$ *or* $\{i : \mathrm{Rem}(i) > 0\} = \emptyset$;

Algorithm 4: Online Proportional

Input: Sequence of loads $(L_t)_{t=1}^T$ and SLAs $\beta(1), \ldots, \beta(N)$.

Initial distribution $\mathbf{h}_1 = (\beta(1), \ldots, \beta(N))$,

for $t = 1, \ldots, T$ **do**

  Set $\mathbf{h}_t$ and obtain $A_t = \{i : Q_t(i) \neq 0\}$,

  Update $h_{t+1}(i) = \frac{\beta(i)}{\sum_{j \in A} \beta(j)}$

  If $A = \emptyset$, then $\mathbf{h}_{t+1} = \mathbf{h}_1$.

2.A.8 Gamma Distribution

Recall that a Gamma distribution ([73]) is characterized by two parameters: the *shape* $k > 0$ and the *scale* $\theta > 0$. The PDF of a Gamma(k, θ) is given by $\frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-x/\theta}$ where $\Gamma(k) = \int_0^\infty u^{k-1} e^{-u} \mathrm{d}u$ is the standard Gamma function. See Figure 2.8 for PDFs of different Gamma distribution for various choices of k and θ.

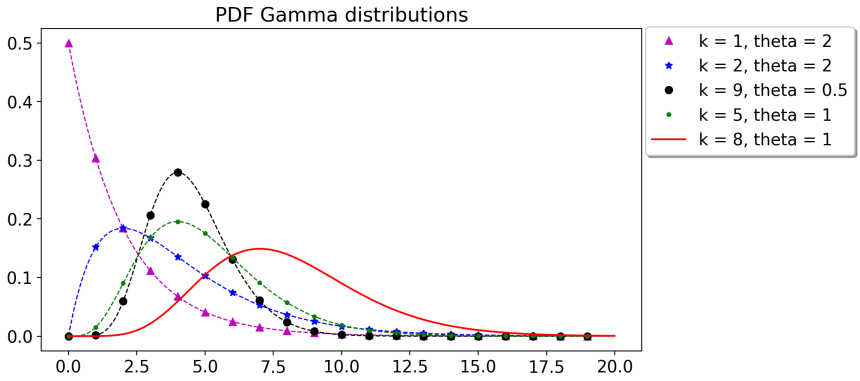

Figure 2.8: PDF of different Gamma distributions.

Proposition 2.16. *Let* $X \sim \mathrm{Gamma}(k, \theta)$. *Then,* $\mathbf{E}[X] = k\theta$ *and* $\mathrm{Var}(X) = k\theta^2$.

CHAPTER 3

ADAPTIVE BIN PACKING WITH OVERFLOW

Motivated by the allocation of virtual machines to servers in the cloud [90], this chapter considers the online problem of packing items with random sizes into unit-capacity bins. Items arrive sequentially, but upon arrival an item's actual size is unknown; only its probabilistic information is available to the decision maker. Without knowing this size, the decision maker must irrevocably pack the item into an available bin or place it in a new bin. Once packed in a bin, the decision maker observes the item's actual size, and overflowing the bin is a possibility. An overflow incurs a large penalty cost and the corresponding bin is unusable for the rest of the process. In practical terms, this overflow models delayed services, failure of servers, and/or loss of end-user goodwill. The objective is to minimize the total expected cost given by the sum of the number of opened bins and the overflow penalty cost. We present an online algorithm with expected cost at most a constant factor times the cost incurred by the optimal packing policy when item sizes are drawn from an i.i.d. sequence of unknown length. We give a similar result when item size distributions are exponential with arbitrary rates. We also study the offline model, where distributions are known in advance but must be packed sequentially. We construct a soft-capacity PTAS for this problem, and show that the complexity of computing the optimal offline cost is $\#\mathbf{P}$-hard. Finally, we provide an empirical study of our online algorithm's performance.

The content of this chapter appears in the INFORMS Journal of Mathematics of Operations Research, 2022 [141]. This work was partially supported by the U.S. National Science Foundation via grants CMMI 1552479, AF 1910423 and AF 1717947.

3.1 Introduction

Bin Packing is one of the oldest problems in combinatorial optimization, and has been studied by multiple communities in a variety of forms. In the classical online formulation, n items with sizes in $[0, 1]$ arrive in an online fashion, and the objective is to pack the items into the fewest possible number of unit-capacity bins. The model has wide applicability in areas including cargo shipping [174], assigning virtual machines to servers [173], a variety of scheduling problems [44, 80, 171], and so on. In many of these applications, the items' sizes may be uncertain, with this uncertainty often modeled via probability distributions. In much of the stochastic bin packing literature, an item's size is observed before it must be packed, e.g. [90, 159]. Nevertheless, in many applications this assumption is unrealistic. For instance, in bandwidth allocation, connection requests are often bursty and deviate from their typical utilization. If the utilization of the request is higher than expected, it can jeopardize the stability of other connections sharing the same channel. Moreover, the only way to observe the actual traffic required by the connection is to first allocate the request and then observe the traffic pattern.

Motivated by these considerations, we introduce an online adaptive bin packing problem that takes into account the following ingredients:

1. Arrivals are adversarial distributions and the length of the item sequence is unknown to the decision maker.

2. In contrast to existing work in the online and/or stochastic bin packing literature, when an item arrives, the decision maker only observes a probability distribution of its size.

3. The decision maker observes the item's actual size only after irrevocably placing it in a bin; therefore, overflowing a bin is possible.

4. An overflowed bin incurs a penalty and renders the bin unusable from that point on. The

objective is to minimize the expected cost given by the sum of the number of open bins and overflow penalty.

3.1.1 Motivating Applications

The online adaptive bin packing problem captures the uncertainty introduced by the online nature of the problem, and also the uncertainty introduced by learning the size of an item *after* it is packed in a bin. While the variant of the bin packing problem we consider is general and widely applicable, the following examples give some concrete applications:

Bandwidth Allocation. An operator is in charge of assigning sequentially arriving independent connection requests. The operator can open new fixed-capacity connections (bins) of unit cost or try to use one of the available connections to pack the incoming request. Traffic on a connection may be bursty, requiring more than the available bandwidth. In this case, the connection suffers from the overflow of the channel, which could represent a monetary penalty or extra work involved in reassigning the request(s) to other connection(s). See also [109].

Freight Shipping. A dispatcher in a fulfillment center is in charge of packing items into trucks for delivery. Truckloads must comply with a maximum weight limit, and our model applies when the dispatcher assigns items into trucks before their final weighing. An overweight truck incurs a penalty representing additional labor or possible fines. See also [111, 112, 137].

Cloud Computing. A controller is in charge of assigning virtual machines (VM) to servers. The controller has statistical knowledge of the amount of resource a VM will utilize (CPU, RAM, I/O bandwidth, energy, etc.), learned via historical data. The actual resource usage is observed once the VM runs in a server. Excessive consumption of a resource by the VM could compromise the stability of the server and negatively affect other VM's sharing the same infrastructure. See also [90].

Operation Room Scheduling. In hospitals, an administrator is in charge of assigning incoming surgeries to different operation rooms. There may be a statistical estimation of a procedure's duration, but the real time spent in the room is only learned once the operation has finished. Over-allocating a room could incur economic penalties and loss of patients' good will. See also [62, 66].

3.1.2 The Model

We consider the problem of sequentially packing items arriving in an online fashion into homogeneous bins of unit capacity. The input consists of a sequence of n nonnegative independent random variables $X_1, \ldots, X_n$, observed sequentially one at a time. Similar to the bin packing literature, we refer to items interchangeably either by their index i or their corresponding random variable X_i. At iteration i, random variable X_i arrives and we observe its distribution but not its outcome. We decide irrevocably to pack X_i into an available bin with nonnegative remaining capacity (if any), or to place X_i in a new bin and pay a unit cost. Once packed, we observe the outcome of the random variable $X_i = x_i$, and the chosen bin's capacity is reduced by this amount. A bin overflows when the sizes of items packed in it sum to more than one; when this happens, we incur an additional cost $C \geq 1$ and the overflowed bin becomes unavailable for future iterations.

We measure the performance of an algorithm $\mathcal{P}$ based on the expected overall cost incurred and denote it $\mathrm{cost}(\mathcal{P})$. Because of the online nature of the problem, we cannot expect to compute the optimal cost for an arbitrary sequence of distributions. Even if we knew all distributions in advance, computing the minimum-cost packing is still computationally challenging; the deterministic version reduces to the $\mathbf{NP}$-hard offline bin packing problem. To quantify the quality of an online algorithm, we compare the expected cost incurred by the algorithm against the expected cost incurred by an optimal adaptive packing policy that knows all distributions in advance. This benchmark knows all size distributions in advance but not their outcomes, and must pack the items sequentially in the same order as the online

algorithm[1]. This measure of quality differs from the traditional online competitive ratio, cf. [13, 29]. In the latter, we would compare our online algorithm against a more powerful optimal offline algorithm that knows all item sizes in advance, or only knows distributions but can pack the items in an adaptive order. We provide two examples showing that against either benchmark, any sequential policy is a factor $\Omega(C)$ more expensive than the optimal solution.

Example 1. Consider n i.i.d. random variables, where $X_i = 1$ with probability $1/C$, and $X_i = 1/n$ with the remaining probability. We expect n/C random variables to realize to 1. Therefore, the expected cost of an offline solution that observes the sizes is at most $n/C+1$. In contrast, the cost incurred by any online algorithm (or even an offline algorithm that observes distributions but not sizes) is at least n. Therefore, when measured against the most powerful benchmark, no online algorithm can have a bounded competitive ratio. The next example shows a similar result for a benchmark that can pack items adaptively.

Example 2. Consider the following sequence of random variables: $X_i = 2^{-i}$ with probability $1 - 2/C$ and $X_i = 1 - 2^{-i}$ with the remaining probability. If the items are packed in the order $X_1, X_2, \ldots, X_n$, the expected optimal cost is n, because at each time step it is better to open a new bin than to use one of the available bins. At time $i > 1$, if we pack item i into a previously opened bin with remaining capacity $1 - x$, the expected cost of this action is $C\,\mathbf{P}(X_i > 1 - x) \geq C \cdot (2/C) \cdot \mathbf{1}_{\{x \geq 2^{-i}\}} \geq 2$, which is larger than the cost of packing item i in a new bin. On the other hand, the optimal expected cost of packing the items in the order $X_n, X_{n-1}, \ldots, X_1$ is at most $2n/C + 1$. Consider the policy that packs items into the same bin until its usage exceeds $1/2$, at which point it opens a new bin. This policy does not overflow any bin, and the expected number of used bins is at most the expected number of items with size $1/2$ or greater, plus one additional bin to accommodate the items placed in the last bin. This example implies that no online algorithm can have a

[1]See Section 3.3 for a more detailed description of policies.

bounded competitive ratio against an adversary that can adaptively choose the order of the items, even if the adversary's item order is fixed beforehand. These two negative results motivate us to use a more refined benchmark that knows distributions but not outcomes in advance, and must pack items in the same order as our algorithm. In addition, we show that computing the cost of this optimal offline policy is $\#\mathbf{P}$-hard (Theorem 3.5).

It is worth mentioning that simple greedy strategies based only on a bin's used capacity can perform poorly compared to the optimal offline policy. One such strategy is the *Greedy Algorithm* that compares the instantaneous expected cost of packing the incoming item in an available bin, $C \cdot \mathbf{P}(X_i \text{ overflows bin})$, versus the unit cost of opening a new bin, selecting the cheapest available choice. This strategy performs poorly in general, even for i.i.d. input sequences.

Example 3. Consider n i.i.d. items, with $X_i \sim \text{Bernoulli}(1/C)$. The optimal policy incurs an expected cost of at most $n/C + 1$: This corresponds to the policy that stops utilizing a bin after observing an item of size 1. On the other hand, Greedy incurs an expected cost of at least $n/2$, since it will keep trying to pack items in a bin until breaking it. Intuitively, in a sequence of Bernoulli trials the expected time to observe two items of size 1 is $2C$; therefore, every $2C$ items (in expectation), Greedy pays a penalty, incurring an expected cost of roughly $n/2$.

Another simple choice for a heuristic packing policy is a *Threshold Algorithm*, which establishes a threshold $\alpha \in (0, 1)$ such that a bin filled to more than α of its capacity is not used again. Notice that for any $\alpha \in (0, 1)$, the optimal policy and the threshold policy incur roughly the same cost for the i.i.d. input $X_i \sim \text{Bernoulli}(1/C)$. We now argue that these policies can perform poorly if the threshold α is not adaptively chosen based on the distribution.

Example 4. Assume that $\alpha \leq 1/2$ (the case $\alpha > 1/2$ is handled similarly) and consider

the i.i.d. input

$$X_i = \begin{cases} 0 & \text{w.p. } 1 - 1/C \\ \alpha & \text{w.p. } 1/2C \\ 1 - \alpha/2 & \text{w.p. } 1/2C. \end{cases}$$

The optimal policy incurs an expected cost of at most $n/C + 1$, since the policy that stops using a bin upon observing a positive outcome incurs at most this cost. On the other hand, the Threshold Algorithm incurs an expected cost of at least $n/24 - C$; we sketch an argument here to obtain this bound, ignoring the $-C$ term for the sake of clarity: The expected number of positive outcomes is n/C. A bin is overflowed by the Threshold Algorithm when an item of size α is followed by another of size $1 - \alpha/2$ (regardless of the number of items of size 0 in between). Focusing solely on the positive outcomes, the number of expected disjoint triplets of the form $(1-\alpha/2, \alpha, 1-\alpha/2)$ is at least a fraction $(1/8) \times (1/3) = 1/24$ of these positive outcomes, from which the bound follows.

We include a brief discussion of threshold policies for i.i.d. input sequences in Appendix 3.C. If the common distribution of the input sequence is finite, a threshold policy can be computed as a function of the distribution, with expected cost a constant factor of the optimal expected cost.

Until now, we have presented examples in which the optimal policies do not break any bin. To not give the false impression that optimal policies do not risk breaking bins, we present the following example.

Example 5. Consider n i.i.d. items, where $X_i = 1$ with probability $1/C^2$ and $X_i = 1/n$ with the remaining probability. The optimal policy has expected cost no more than $n/C+1$, far less than the policy that does not break any bins, which incurs an expected cost of n.

In deterministic bin packing problems, one of the most useful bounds for the number of

used bins is the sum of the item sizes. It is known that this value is at least half the number of bins used by any greedy algorithm [48]. In our stochastic setting, the expected sum of item sizes could be far from the number of bins used. Indeed, for the random variables considered in Example 1, we have $\sum_{i=1}^{n} \mathbf{E}[X_i] = (n-1)/C + 1$, while the expected cost of any policy is at least n for this input sequence.

3.1.3 Results and Contributions

We propose a heuristic algorithm called Budgeted Greedy and denoted ALG (Algorithm 5). Budgeted Greedy uses a *risk budget* in each bin as a way to control the risk of overflowing the bins. If we consider packing item i in bin j, this action's risk is equal to the probability of overflowing the bin; Budgeted Greedy maintains a bin's risk below its risk budget. At every step, similar to the bin's capacity, when an item is packed in a bin, the bin's risk budget is reduced by the probability of the current item overflowing the bin. If no currently opened bin has enough risk budget left, then a new bin is opened. Observe that the risk of packing item i into any available bin depends on the realized sizes of items $1, \ldots, i-1$ and these items' assignments.

The risk as defined above can be calculated for any policy. While there are instances where the optimal policy incurs a large risk for certain bins it opens, our first structural result shows that *any* policy can be converted to one with budgeted risk with at most a constant factor loss.

Theorem 3.1. *Let $X_1, \ldots, X_n$ be an arbitrary sequence of independent nonnegative random variables (not necessarily identically distributed). For any $\gamma > 0$ and for any policy $\mathcal{P}$ that sequentially packs $X_1, \ldots, X_n$, there exists a risk-budgeted policy $\mathcal{P}'$ packing the*

same items, such that no bin surpasses the risk budget γ/C, and with expected cost

$$\operatorname{cost}(\mathcal{P}') \leq (1 + 2/\gamma) \operatorname{cost}(\mathcal{P}).^2$$

Theorem 3.1 is obtained by updating policy $\mathcal{P}$'s decision tree whenever the risk budget is violated by opening a new bin. The extra cost of the new opened bins is paid by a delicate charging argument. Notice that as $\gamma \to \infty$, we recover the original cost of the policy.

While the cost of any policy involves two terms, the expected number of open bins and the expected penalty for overflowed bins, we show (Lemma 3.9) that for a *budgeted policy*, the cost of overflowed bins is at most a constant factor of the number of opened bins in expectation, with the constant depending on the budget γ/C. This allows us to exclusively focus on the number of bins opened by the budgeted policy. A consequence of these structural results is the following.

Theorem 3.2. *If the input sequence $X_1, \ldots, X_n$ is i.i.d., for any γ, the Budgeted Greedy algorithm with this γ minimizes the expected number of opened bins among all risk-budgeted policies with the same γ. In particular, for $\gamma = \sqrt{2}$ we obtain $\operatorname{cost}(\textsc{Alg}) \leq (3 + 2\sqrt{2}) \operatorname{cost}(\textsc{Opt})$, where $\textsc{Opt}$ denotes the optimal policy that knows n in advance.*

In many bin packing models, it is possible to construct an explicit lower bound for the cost of the optimal benchmark (e.g. the expected sum of the item sizes). In our setting, even though these bounds still hold, they can be a factor of C away from $\operatorname{cost}(\textsc{Opt})$, even for i.i.d. input sequences (see Example 5). We therefore take a different approach for the i.i.d. case; we transform policies into the policy induced by Budgeted Greedy, incurring only small, multiplicative losses.

This i.i.d. model can be interpreted in the following manner. Suppose there is a probability

[2] If there are items X_i with $\mathbf{P}(X_i > 1) > \gamma/C$, these are packed individually. Bins not containing these items have risk bounded by γ/C. See Section 3.4.

distribution over the nonnegative real numbers. There are n item sizes independently drawn from this distribution, $x_1, \ldots, x_n$. For each $i = 1, \ldots, n$, we are asked to pack the i-th item without observing its size. This is indeed a model for basic allocation systems where only a population distribution is known about the item's size, which is a typical occurrence in practical applications if more granular information is not available.

As a consequence of Theorem 3.2, we can also show the existence of instance-dependent threshold policies with similar guarantees as Budgeted Greedy.

Corollary 3.2.1. *If the input sequence $X_1, \ldots, X_n$ is i.i.d. with finite support, there is a threshold $\alpha \in [0, 1]$ that depends on the common distribution of the X_i, such that the threshold policy $\mathcal{P}_\alpha$, which stops using bins when their capacity exceeds α, satisfies $\operatorname{cost}(\mathcal{P}_\alpha) \leq (3 + 2\sqrt{2}) \operatorname{cost}(\textsc{Opt}) + 1$, where* Opt *denotes the optimal policy that knows n in advance.*

The proof is based on the theory of discounted Markov decision processes (see [146]). We need the finiteness of the support of the distribution to show the existence of a fixed point, which is crucial for the Bellman recursion in the discounted setting. The proof of this corollary appears in the Appendix 3.C.

As a second contribution, we show that for arbitrary exponential distributions, i.e. a sequence of random variables $X_1, \ldots, X_n$ with $\mathbf{P}(X_i > x) = e^{-\lambda_i x}$, Budgeted Greedy incurs a cost that is at most a factor $\mathcal{O}(\log C)$ times the benchmark cost. Moreover, if the exponential random variables are sufficiently small, this factor can be reduced to a constant.

Theorem 3.3. *If each X_i is exponentially distributed with rate $\lambda_i > 0$, Budgeted Greedy satisfies*

$$\operatorname{cost}(\textsc{Alg}) \leq \mathcal{O}(\log C) \operatorname{cost}(\textsc{Opt}).$$

Furthermore, if $\lambda_i \geq 2 \log C$ for all $i = 1, \ldots, n$, $\operatorname{cost}(\textsc{Alg}) \leq \mathcal{O}(1) \operatorname{cost}(\textsc{Opt})$.

We show that Budgeted Greedy opens bin $i+1$ if it either packs at least $\Omega(1/\log C)$ in bin i, or the risk in bin i is bounded by a small constant, which we obtain via an auxiliary non-convex maximization problem. With this, Budgeted Greedy's cost is bounded by $\mathcal{O}(\log C)\sum_i \mathbf{E}[X_i] \leq \mathcal{O}(\log C)\,\mathrm{cost}(\textsc{Opt})$. When the exponential random variables are small enough, Budgeted Greedy opens bin $i+1$ if a constant amount of mass in bin i is packed, thereby reducing the $\log C$ factor to a constant. We also give a $\Omega\big(\sqrt{\log C}\big)$ lower bound for Budgeted Greedy's competitive ratio in the case of exponentially distributed sizes.

Offline Model Although our motivation for studying the bin packing model is an online application, the offline sequential version of the problem is interesting in its own right, as it interpolates the online setting and the completely offline setting, where items are packed in an arbitrary order. In the offline sequential version of the problem, an ordered list of random variables is given to the decision maker, and the objective is to design a sequential policy to minimize expected cost, in time polynomial in the number of items and possibly $\log C$. As in the online model, right after the decision maker packs a random variable (item) into a bin, the actual size is revealed to her. The optimal offline expected cost computed here corresponds to the benchmark we consider in the online setting.

In this offline framework, we present two main contributions. Following the resource augmentation literature [79, 119], the first contribution states that there is a polynomial-time approximation scheme (PTAS) for computing a policy when the capacity of the bins is extended by ε.

Theorem 3.4. *For any $0 < \varepsilon \leq (\sqrt{15}-3)/\sqrt{6}$, there is an algorithm running in $\mathcal{O}\left(n^{2(6/\varepsilon)^5}/\varepsilon^{10}\right)$ time that computes a polynomial-size policy $\mathcal{P}$ packing items into bins of size $1+\varepsilon$, and incurring an expected cost of at most $(1+\varepsilon)\,\mathrm{cost}(\textsc{Opt})$, where* Opt *is the optimal policy packing items into bins with unit capacity.*

The algorithm uses a discretization of possible item sizes similar to [119]. This allows us to find an optimal policy for discretized outcomes via dynamic programming in polynomial time. The cost of this policy is almost the original optimal cost. We recover a policy for the original items by a tracking argument simulating the discretized policy in parallel. The policy follows the discretized policy's decision to pack items in a bin j as long as the error between the sizes in j and its discrete version remains small. When this fails, the policy opens a new copy of j and keeps following the discretized policy as before. This tracking is enough to guarantee similar cost between the two policies.

Our second result for the offline model relates the complexity of computing the optimal value to counting problems. Specifically, we show that computing the optimal offline cost is $\#\mathbf{P}$-hard—hence, the optimal *online* benchmark is also $\#\mathbf{P}$-hard to compute.

Theorem 3.5. *It is* $\#\mathbf{P}$*-hard to minimize* $\operatorname{cost}(\mathcal{P})$.

The proof of this result is divided into two parts. First, we show that counting solutions of symmetric logic formulas in 4CNF[3] is $\#\mathbf{P}$-hard (Theorem 3.46). From a symmetric 4CNF formula we construct a stochastic input of the stochastic bin packing problem, where $\min_{\mathcal{P}} \operatorname{cost}(\mathcal{P})$ allows us to count the solutions of the 4CNF formula. The proof resembles the reduction from the Partition problem to the Bin Packing problem. Intuitively, randomized items model outcomes of variables in the 4CNF formula, one item for each positive and negative literal. The main step in the proof is to correlate the outcomes of the positive/negative literals corresponding to the same variable. The proof of Theorem 3.5 is deferred to Appendix 3.B.

The rest of the chapter is organized as follows. We follow this introduction with a brief literature review. In Section 3.3, we present the Budgeted Greedy algorithm and introduce the necessary notation for the rest of the chapter. Section 3.4 focuses on the i.i.d. case,

[3]Logic formula in conjunctive normal form with 4 literals in each clause.

including the proofs of Theorem 3.1 and Theorem 3.2. In Section 3.5 we turn to exponentially distributed item sizes, with the proof of Theorem 3.3 and the construction of the corresponding lower bound. Section 3.6 discusses the offline case, including the proof of Theorem 3.4. In Section 3.7 we present a numerical study of our algorithms, comparing it with natural benchmarks.

3.2 Related Work

In the classic one-dimensional bin packing problem, n items with sizes $x_1, \ldots, x_n$ in $[0, 1]$ must be packed in the fewest unit-capacity bins without splitting any item into two or more bins. This is a well-studied $\mathbf{NP}$-complete problem spanning more than sixty years of work [57, 80, 84, 101, 103, 106, 151]. For excellent surveys see [42, 48]. In the online version, the list of items $L = (x_1, \ldots, x_n)$ is revealed online one item at a time. In round t, we observe item x_t and must irrevocably decide whether to pack it in an open bin with enough remaining space or to open a new unit-capacity bin at unit cost, without knowledge of future arrivals. It is standard to measure an online algorithm's performance via its *(asymptotic) competitive ratio* [13, 29, 48] $\limsup_{|L|\to\infty} \mathrm{cost}_{\mathrm{alg}}(L)/\mathrm{cost}_{\mathrm{OPT}}(L)$, where $\mathrm{cost}_{\mathrm{alg}}(L)$ is the cost incurred by the online algorithm with input L, and $\mathrm{cost}_{\mathrm{OPT}}(L)$ is the cost incurred by the optimal offline solution that knows L in advance. The best known competitive ratio is 1.57829 [20], and the best current lower bound is 1.5403 [21]; see also [172, 175]. Our model subsumes the deterministic case, and therefore inherits the aforementioned lower bound.

In several real-world applications, exact item sizes are unknown to the decision maker at the time of insertion [61, 171]. This uncertainty is typically modeled via probability distributions of the items' sizes. Several online and offline bin packing models introducing stochastic components have been studied. These include, for instance, models where item sizes are drawn from know distributions and observed before packing [45, 53, 91, 149,

159, 158], or chance-constrained models where all sizes are revealed to the decision maker after committing to a packing [85, 119, 109]. These stochastic models have revealed connections with balls-into-bins problems [159], sums of squares [53], queuing theory [47], Poisson approximation [119], etc. For the online case, common to all these models is the assumption that the item size is observed before packing it. Nevertheless, observing the item's size before deciding where to pack it is unrealistic in many scenarios. For instance, in cloud computing, before running a job in a cluster, we may have some statistical knowledge of the amount of resource the job will utilize. However, the only way to observe the real utilization is to start the job in the cluster. In this work, we propose a new model variant where items' size distributions are revealed in an online fashion, but each outcome is observed only after packing the item. We therefore relax the strict capacity constraint by allowing each bin to overflow at most once, at the expense of a penalty.

Our model also shares similarities with adaptive combinatorial optimization, particularly stochastic knapsack models introduced in [63]. Recent treatment began with the work of Dean, Goemans, and Vondrák [59] from an approximation algorithms viewpoint; however, their work focuses primarily on the power of adaptivity in the stochastic knapsack as opposed to the design of adaptive policies. Since then, a large body of work has studied the adaptive knapsack model (and more general problems) from several perspectives: information-relaxation in MDP [22], improved approximations via policy analysis [25, 119], improved approximations via linear programming [89, 122] and new linear programs using MDP relaxations [26, 27, 28]. In these models, overflowing the knapsack stops the packing of items but does not directly penalize the decision maker. Conversely, in the so-called *blackjack knapsack problem*, overflowing the knapsack results in zero value for the decision maker [79, 117]. This could be interpreted as a penalty similar to the one incurred by the decision maker in our model. Most of these works assume complete knowledge of the input distributions, and online treatments are scarcer in the literature. Here we highlight work in online generalized assignment [12] and stochastic bipartite matching [87, 129].

A related area of work is the extensible bin packing problem [46, 60]. Roughly speaking, a fixed number of bins are given and a set of items must be packed into them. The cost of bin B corresponds to $\max\left\{\sum_{i\in B} x_i, 1\right\}$, a fixed unit cost and a linear excess cost. The objective is to design packings with small overall cost; even though we do not allow bins to be utilized after overflowing, we could interpret our model as a nonlinear version of a stochastic extensible bin packing problem. Such models capture situations in which a fixed cost is incurred to complete a task with finite resources (e.g. workers) but the resources can be "stretched" with some additional linear cost (e.g. overtime pay). For a generalization to different costs and bin capacities, see [116]. For a stochastic approach similar to our posterior observability, see [152].

3.3 The Algorithm

3.3.1 Preliminaries

The problem's input consists of n independent nonnegative random variables $X_1, \ldots, X_n$. The (possible) bins to utilize are denoted by $B_1, B_2, \ldots, B_n$. A *state* $\mathbf{s}$ for round $i \in [n+1]$ is a sequence $(x_1, 1 \to j_1)(x_2, 2 \to j_2)\cdots(x_{i-1}, i-1 \to j_{i-1})$, where x_k is an outcome of X_k for all $k < i$. The pair $(x_k, k \to j)$ represents round k, and refers to packing X_k in bin j and observing outcome $X_k = x_k$. A state for round i represents the path followed by a decision maker packing items $X_1, \ldots, X_n$ sequentially into bins and the outcomes for each of these decisions until round $i-1$. States have a natural recursive structure: $\mathbf{s} = \mathbf{s}'(x_{i-1}, i-1 \to j_{i-1})$, where $\mathbf{s}'$ is the state for round $i-1$. The *initial state* $\mathbf{s}_0$ is the empty state. Bin B_j is *open* by state $\mathbf{s}$ if some $(x_k, k \to j)$ appears in $\mathbf{s}$. The items packed into bin B_j by state $\mathbf{s}$ are $B_j(\mathbf{s}) = \{k : (x_k, k \to j) \text{ appears in } \mathbf{s}\}$. The number of bins opened by state $\mathbf{s}$ is $|\{j : (x_k, k \to j) \text{ appears in } \mathbf{s}\}|$. The *usage* of bin B_j at the beginning of round i in state $\mathbf{s}$ is

$$S_j^{i-1}(\mathbf{s}) = \sum_{\substack{k \le i-1 \\ (x_k, k \to j) \in \mathbf{s}}} x_k,$$

the sum of sizes of items packed in bin j. A bin B_j is *broken* or *overflowed* in $\mathbf{s}$ if $S_j^{i-1}(\mathbf{s}) > 1$. In our model, we stop using bins that overflow. A state $\mathbf{s}$ for round i is *feasible* if any overflowed bin by round k is never used again after k, for any $k < i$. The *state space* is the set of *all feasible states*, denoted $\mathbf{S}$. The set of all feasible states for round $i \leq n$ is denoted by $\mathbf{S}_i$.

A *policy* $\mathcal{P}$ is a function $\mathcal{P} : \mathbf{S}_n \to [n]$ such that for a feasible state $\mathbf{s} \in \mathbf{S}_n$ for round i, $\mathcal{P}(\mathbf{s}) = j$ indicates that item i is packed into bin j; we write this as $i \to j$ when the policy and state are clear from the context. The policy is *feasible* if $\mathbf{s}' = \mathbf{s}(x_i, i \to \mathcal{P}(\mathbf{s}))$ is a feasible state for any feasible $\mathbf{s} \in \mathbf{S}_n$ for round i and outcome x_i of X_i. From now on, we only consider feasible policies. A state $\mathbf{s}' \in \mathbf{S}$ is *reachable* by the policy if $\mathbf{s}' = \mathbf{s}_0$ or $\mathbf{s}' = \mathbf{s}(x_i, i \to \mathcal{P}(\mathbf{s}))$ with $\mathbf{s}$ reachable, $\mathbf{s}$ for round i and x_i an outcome of X_i. For a reachable state $\mathbf{s}$ for round $i \in [n]$, we say that $\mathcal{P}$ *opens* bin j if $\mathcal{P}(\mathbf{s}) = j$ and B_j is not open in $\mathbf{s}$. We say that the policy *overflows* bin B_j at state $\mathbf{s}$ if B_j overflows for $\mathbf{s}' = \mathbf{s}(x_i, i \to \mathcal{P}(\mathbf{s}))$ but B_j is not overflowed in $\mathbf{s}$. We set the *cost* of a policy as

$$\mathrm{cost}(\mathcal{P}) = \mathbf{E}[N_{\mathcal{P}}] + C\,\mathbf{E}[O_{\mathcal{P}}],$$

where $N_{\mathcal{P}}$ is the number of bins opened and $O_{\mathcal{P}}$ is the number of bins broken by reachable states for round $n + 1$. The randomness is over the items' outcomes. Notice that non-reachable states in $\mathbf{S}$ are unimportant for $\mathrm{cost}(\mathcal{P})$, hence we can always assume $\mathcal{P}(\mathbf{s}) = n$ for non-reachable $\mathbf{s} \in \mathbf{S}_n$. A policy specifies the actions to apply in any epoch of the sequential decision-making problem. Note that our states are typically considered *histories* in the Markov decision processes literature [146]. We use our description of states to keep close track of policies' actions in the subsequent analysis.

Any policy $\mathcal{P}$ has a natural $(n + 1)$-level decision tree representation $\mathcal{T}_{\mathcal{P}}$, which we call the *policy tree*. The root, denoted r, is at level 1 and represents item X_1 and state $\mathbf{s}_0$. A

node at level $i \in [n]$ is labeled with $\mathcal{P}(i, \mathbf{s})$ where $\mathbf{s}$ is the state of the system obtained by following the path from the root to the current node. There is a unique arc going out of the node for every possible outcome of X_i directed to a unique node in level $i+1$. Nodes at level $n+1$ are *leaves* denoting that the computation has ended. Nodes in levels $i \in [n]$ are called *internal nodes*. To compute the $\mathrm{cost}(\mathcal{P})$ using the policy-tree $\mathcal{T}_{\mathcal{P}}$, we add two labels to the tree:

- For an internal node u, $\ell_u = 1$ if $\mathcal{P}$ opens a new bin in node u; 0 otherwise. For leaves we define $\ell_u = 0$.
- For arcs $a = (u, v)$, we define $c_a = C$ if the outcome of the random variable belonging to the level where u is located overflows the bin chosen by the policy at node u; 0 otherwise.

We refer to this tree as cost-labeled tree $\mathcal{T}_{\mathcal{P}}$ with cost vectors (ℓ, c), or simply cost-labeled tree $\mathcal{T}_{\mathcal{P}}$ if the costs are clear from the context. The tree structure gives us a recursive way of computing the cost of the policy. Let $\mathcal{T}_{\mathcal{P}}(u)$ be the cost-labeled sub-tree of $\mathcal{T}_{\mathcal{P}}$ rooted at node u; then

$$\mathrm{cost}_{\ell,c}(\mathcal{T}_{\mathcal{P}}(u)) = \begin{cases} \ell_u + \mathbf{E}_{X_i}[c_{(u,u_{X_i})} + \mathrm{cost}_{\ell,c}(\mathcal{T}_{\mathcal{P}}(u_{X_i}))] & \text{if } u \text{ is at level } i = 1, \ldots, n \\ 0 & \text{if } u \text{ is at level } n+1 \end{cases},$$

where u_{X_i} is the node at level $i+1$ connected to u. Thus, $\mathrm{cost}(\mathcal{P}) = \mathrm{cost}_{\ell,c}(\mathcal{T}_{\mathcal{P}}(r))$. We define $\textsc{Opt} = \mathrm{argmin}_{\mathcal{P}}\, \mathrm{cost}(\mathcal{P})$ as the optimal policy for sequentially packing items $X_1, \ldots, X_n$. This policy might not exist in cases where the number of states is uncountable, for example, when $X_1, \ldots, X_n$ have continuous distributions. In this case, the policy tree has uncountably many edges emanating from nodes, corresponding to all possible realizations of X_i. Nevertheless, a ε-optimal policy is guaranteed to exist, i.e. a policy $\mathcal{P}$ that ensures $\mathrm{cost}(\mathcal{P}) \leq \inf_{\mathcal{P}} \mathrm{cost}(\mathcal{P}) + \varepsilon$. We abuse notation by calling $\textsc{Opt}$ the optimal policy (or an arbitrarily good approximation if it does not exist).

Note that we defined only *deterministic* policies, since the action $\mathcal{P}(\mathbf{s})$ is deterministic. If $\mathcal{P}(\mathbf{s})$ were a probability distribution over $[n]$, then we would have a randomized policy. A standard result from Markov decision processes theory ensures that any randomized policy has a deterministic counterpart incurring the same cost; hence, we only focus on deterministic policies. For more details see [143, 146].

The following proposition characterizes the expected number of bins overflowed by a policy. The proof appears in Appendix 3.A.

Proposition 3.6. *Let $X_1, \ldots, X_n$ be nonnegative independent random variables, and let $\mathcal{P}$ be any policy that sequentially packs these items. The expected number of bins overflowed by the policy $\mathcal{P}$ is*

$$\mathbf{E}[O_{\mathcal{P}}] = \sum_{j=1}^{n} \mathop{\mathbf{E}}_{X_1,\ldots,X_n} \left[\sum_{i=1}^{n} \mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1) \mathbf{1}_{\{i \to j\}}^{\mathcal{P}} \right],$$

where S_j^{i-1} is the usage of bin j at the beginning of iteration i and $\mathbf{1}_{\{i\to j\}}^{\mathcal{P}}$ is the indicator random variable of the event in which $\mathcal{P}$ packs item X_i into bin j.

If we interpret $\mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1)$ as the *risk* that X_i overflows bin j if packed there, the result says that the number of overflowed bins is the expected aggregation of these risks. We define the *risk* of a bin j as $\mathrm{Risk}(B_j) = \sum_{i=1}^{n} \mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1)\mathbf{1}_{\{i\to j\}}$. Then $\mathbf{E}[O_{\mathcal{P}}] = \sum_{j=1}^{n} \mathbf{E}[\mathrm{Risk}(B_j)]$. A policy $\mathcal{P}$ is *risk-budgeted* or simply *budgeted* with *risk budget* $r > 0$ if no bin incurs a risk larger than r, $\mathrm{Risk}(B_j) \leq r$ for $j \in [n]$.

A deterministic online algorithm induces a policy, with non-reachable states simply mapped to $\emptyset$. Since online algorithms are not aware of the number of items n, we label the j-th bin opened by an online algorithm as B_j in this case. The cost of an online algorithm is naturally defined as the cost of the corresponding induced policy.

We use the notation $z(B) = \sum_{i\in B} z_i$ for a vector $z = (z_1, \ldots, z_n)$. If $X = (X_1, \ldots, X_n)$

is the vector of random variables and $B = B_j$, then $X(B) = S_j^n$ is the usage of bin B_j. The following propositions are probabilistic analogues of the well-known size lower bound for deterministic bin packing. We use them in Sections 3.5 and 3.6. The proofs are deferred to Appendix 3.A.

Proposition 3.7. *For any sequence of nonnegative i.i.d. random variables $X_1, \ldots, X_n$, for any bin $B = B_j$ and any policy $\mathcal{P}$, we have*

$$\mathbf{E}\left[\sum_{i\in B}\mathbf{E}[X_i \wedge 1]\right] = \mathbf{E}\left[\sum_{i\in B}(X_i \wedge 1)\right] \leq 2\,\mathbf{P}(\mathcal{P} \textit{ opens bin } B),$$

where $X_i \wedge 1 = \min\{X_i, 1\}$.

When all items sizes are aggregated, we can improve the factor of 2 as follows.

Proposition 3.8. *For any sequence of nonnegative i.i.d. random variables $X_1, \ldots, X_n$, for any policy, we have*

$$\mathbf{E}\left[\sum_{i=1}^{n}(X_i \wedge 1)\right] \leq \mathrm{cost}(\mathcal{P}).$$

3.3.2 The Budgeted Algorithm

In the *Budgeted Greedy* algorithm, we keep a risk budget for each bin that is initialized as γ/C, where $\gamma \geq 1$ is an algorithm parameter. We pack items in a bin as long as the usage of the bin is at most 1 and its risk budget has not run out. More formally, when opening a bin, say bin j at round i, we initialize its risk of overflow at $r_j^{i-1} = 0$. At round i, when item X_i arrives, we find a bin j such that $r_j^{i-1} + p_i(S_j^{i-1}) \leq \gamma/C$, where r_j^{i-1} is the accumulated risk of overflowing the bin until $i-1$, S_j^{i-1} is the usage of the bin j until the previous round and $p_i(S_j^{i-1}) = \mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1)$ is the risk that X_i overflows bin j. If that bin j exists, we pack the incoming item into bin j, breaking ties arbitrarily, and we update the risk of overflow as $r_j^i = r_j^{i-1} + p_i(S_j^{i-1})$ and $r_{j'}^i = r_{j'}^{i-1}$ for any $j' \neq j$. Such a bin may not exist,

in which case we open a new bin k with $r_k^i = p_i(0)$. Strictly speaking, Budgeted Greedy is not a budgeted policy with risk budget γ/C unless all items satisfy $\mathbf{P}(X_i > 1) \leq \gamma/C$; items with $\mathbf{P}(X_i > 1) > \gamma/C$ are packed into individual bins. In Algorithm 5, we formally present the description of Budgeted Greedy.

Algorithm 5: BUDGETED-GREEDY$(\gamma, X_1, \ldots, X_n)$

Initialize: $I = \emptyset$.
for $i = 1 \ldots, n$ **do**
  if $\exists j \in I$ *such that* $r_j^{i-1} + p_i(S_j^{i-1}) \leq \gamma/C$ **then**
    $S_j^i = S_j^{i-1} + X_i$.
    $r_j^i = r_j^{i-1} + p_i(S_j^{i-1})$.
  else
    Define $r_j^i = p_i(0)$ for j such that $j = \inf\{j \geq 0 : j \notin I\}$.
    $S_j^i = X_i$.
    Update $I = I \cup \{j\}$.
  end
  for $j' \neq j$ **do**
    $S_{j'}^i = S_{j'}^{i-1}$.
    $r_{j'}^i = r_{j'}^{i-1}$.
  end
end

Lemma 3.9. *Let $\gamma \geq 1$ and assume that for all i, $\mathbf{P}(X_i > 1) \leq \gamma/C$. For any bin j, Algorithm 5 guarantees*

$$\mathbf{P}(\text{ALG } \textit{breaks bin } j) \leq \frac{\gamma}{C}\,\mathbf{P}(\text{ALG } \textit{opens bin } j).$$

Proof. Proof. Using Proposition 3.6,

$$\begin{aligned}\mathbf{P}(\text{ALG breaks bin } j) &= \mathbf{E}\left[\left(\sum_{i=1}^{n}\mathbf{P}(X_i + S_j^{i-1} > 1)\mathbf{1}^{\text{ALG}}_{\{t\to j\}}\right)\mathbf{1}_{\{\text{ALG opens bin } j\}}\right] \\ &= \mathbf{E}\left[\text{Risk}(B_j)\mathbf{1}_{\{\text{ALG opens bin } j\}}\right] \le \frac{\gamma}{C}\,\mathbf{P}(\text{ALG opens bin } j),\end{aligned}$$

since once the bin has been opened, its risk never goes beyond γ/C. □

As a result, we have the following corollary, which implies that we only need to bound the expected number of bins opened by Budgeted Greedy in our analysis.

Corollary 3.9.1. *Under the same assumptions as Lemma 3.9,* $\text{cost}(\text{ALG}) \le (1+\gamma)\,\mathbf{E}[N_{\text{ALG}}]$.

3.4 A Policy-Tree Analysis for I.I.D. Random Variables

In this section we prove Theorem 3.1 for general input distributions in Theorem 3.10. We use this result to prove Theorem 3.2, which gives the Budgeted Greedy guarantee for the i.i.d. case. Theorem 3.10 states that any policy can be converted into a budgeted version, where a risk budget is never surpassed for any bin. This transformation can be carried out while only incurring a small multiplicative loss. The proof relies on a charging scheme in the cost paid by overflowing bins. Starting with the original policy tree, we increase the cost paid by overflowing bins by an amount $\delta > 0$. The overall cost of the tree increases multiplicatively by at most $(1 + \delta/C)$. We show that this additional δ allows us to pay for new bins whenever the risk of the bin goes beyond γ/C, for an appropriate choice of δ and γ.

Theorem 3.10. *Let $X_1, \ldots, X_n$ be an arbitrary sequence of independent, nonnegative random variables that are not necessarily identical. Fix $\gamma > 0$. For any policy $\mathcal{P}$ that sequentially packs items $X_1, \ldots, X_n$, there exists a policy $\mathcal{P}'$ for the same items such that:*

- *$\mathcal{P}'$ packs items with $\mathbf{P}(X_i > 1) > \gamma/C$ into individual bins, and bins not containing these items never exceed the risk budget γ/C.*

- *$\mathcal{P}'$ satisfies* $\text{cost}(\mathcal{P}') \leq (1 + 2/\gamma)\,\text{cost}(\mathcal{P})$.

In particular, if all items satisfy $\mathbf{P}(X_i > 1) \leq \gamma/C$, $\mathcal{P}'$ is a risk-budgeted policy with risk budget γ/C.

Proof. Proof. The proof follows two phases. In the first and longest phase, we show that we can modify the policy $\mathcal{P}$ in such a way that the risk of each bin exceeds γ/C at most once. In the second phase, we show that the item surpassing the risk budget in each bin can be packed into an individual bin. At the end, no item with $\mathbf{P}(X_i > 1) \leq \gamma/C$ can exceed the risk γ/C.

In the rest of the proof we utilize the tree representation of the policy. Let $\delta = C/\gamma > 0$. We proceed as follows:

1. In the cost labeled tree $\mathcal{T}_{\mathcal{P}}$, increase the cost of overflowing the bins from C to $C + 2\delta$. That is, $\widehat{c}_{(u,v)} = C + 2\delta$ if $c_{(u,v)} = C$ and 0 otherwise for any arc (u, v) in $\mathcal{T}_{\mathcal{P}}$. Then,

$$\text{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(r)) \leq \left(1 + 2\frac{\delta}{C}\right) \text{cost}_{\ell,c}(\mathcal{T}_{\mathcal{P}}(r)) = \left(1 + 2\frac{\delta}{C}\right) \text{cost}(\mathcal{P}).$$

2. Starting at the root of this new cost-labeled tree, find a node u at level $i = 1, \ldots, n$ where the policy $\mathcal{P}$ decides to open a new bin, say bin j. In each of the branches starting at node u and directed to some leaf, find the sequence of nodes $u_1 = u, u_2, \ldots, u_k$ where the policy packs items into bin j and node u_k corresponds to the first node in the branch where the risk budget γ/C is surpassed for bin j. Define u_k as a leaf if in the branch the risk budget is not surpassed for bin j. Let $i_1 = i, i_2, \ldots, i_k$ be the items packed into bin

j in this branch; that is, node u_ℓ is at level i_ℓ. Then, we have

$$\sum_{m=1}^{k-1} \mathbf{P}_{X_{i_m}}(X_{i_m} + S_j^{i_m-1} > 1) \leq \frac{\gamma}{C}, \quad \text{and} \quad \sum_{m=1}^{k} \mathbf{P}_{X_{i_m}}(X_{i_m} + S_j^{i_m-1} > 1) > \frac{\gamma}{C}$$

if node u_k is not a leaf. Here $S_j^{i_m-1}$ represents the usage of bin j at node i_m.

Consider the following modifications to the cost-labeled tree $\mathcal{T}_\mathcal{P}$: We start with the same tree as $\mathcal{P}$ but in the subtree rooted at u, bin j is utilized only in nodes $u_1, \ldots, u_k$ for the different branches. Any future utilization of bin j after passing through node u_k is moved to a new bin j'. Now, we update the cost labels as follows. For all the branches, we reduce the cost of $C + 2\delta$ appearing in the arcs going out from nodes $u_1, \ldots, u_k$ to $C + \delta$. We label the first node appearing after node u_k where the bin j' is opened with a 1. We reduce the labels of arcs going out of nodes using bin j' if they do not overflow the bin j' anymore (bin j' has smaller usage than bin j). Formally, for any branch and nodes $u_1, \ldots, u_k$ defined as before,

$$c'_{(a,b)} = \begin{cases} \frac{C+\delta}{C+2\delta}\widehat{c}_{(a,b)} & a = u_m \text{ for some branch starting at } u \\ \widehat{c}_{a,b} & \text{otherwise} \end{cases},$$

and for nodes,

$$\ell'_a = \begin{cases} 1 & a \text{ is the first node packed into bin } j' \text{ in the subtree } \mathcal{T}_\mathcal{P}(u_k) \\ \ell_a & \text{otherwise} \end{cases}.$$

We denote this new policy by $\mathcal{P}'$. Figure 3.1 displays the modification process.

We now argue that the changes applied to the cost-labeled tree $\mathcal{T}_\mathcal{P}$ to transform it into $\mathcal{T}_{\mathcal{P}'}$ do not increase the cost function $\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_\mathcal{P})$. Since we only modified labels in the subtree $\mathcal{T}_\mathcal{P}(u)$ it is enough to study the cost change in this specific subtree for bins j and

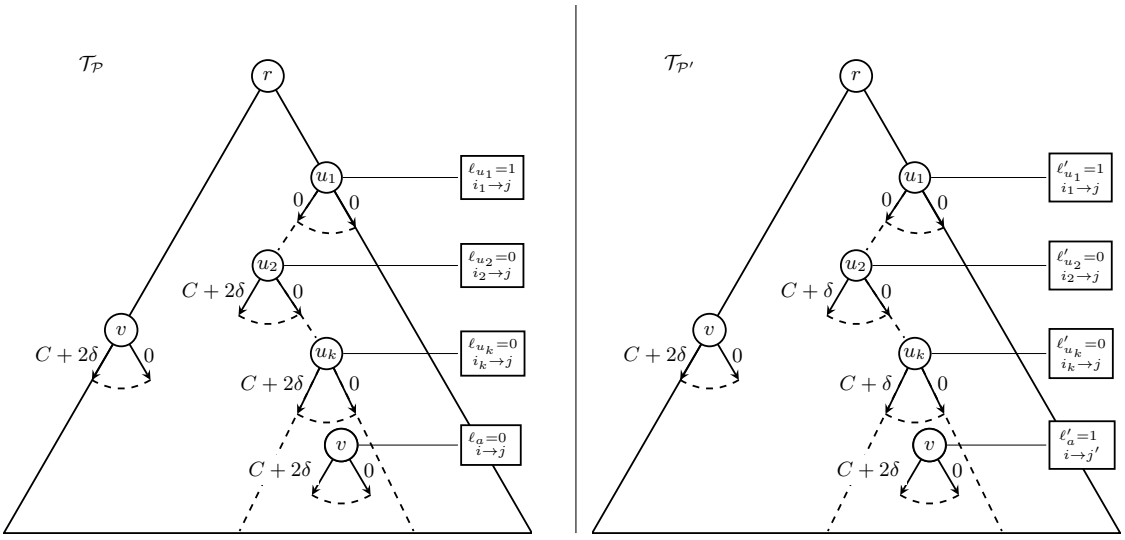

Figure 3.1: Policy tree modification. On the left, we display the original tree with augmented cost from C to $C + 2\delta$. On the right, we show the modified labels after opening a new bin in node u_k. Observe that we only decrease the costs of arcs related to bin j going out of nodes $u_1, \ldots, u_k$ in all branches starting at node u.

bin j'.

Lemma 3.11. $\text{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(u)) \geq \text{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(u))$.

The proof of this lemma appears in Appendix 3.A. With this result we have

$$\begin{aligned}&\text{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(r)) - \text{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(r))\\ =&\,\mathbf{E}\left[\mathbf{E}\left[\text{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(u)) - \text{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(u)) \mid \text{Reach node } u\right]\right] \geq 0\end{aligned}$$

3. Now, starting from policy $\mathcal{P}'$ and cost-labeled tree $\mathcal{T}_{\mathcal{P}'}$ with labels ℓ' in the nodes and c' in the arcs, repeat step 2 until every bin exceeds the risk budget γ/C at most once.

With the previous method, we construct a policy, which we still call $\mathcal{P}'$ for simplicity, in addition to its policy tree $\mathcal{T}_{\mathcal{P}'}$ and labels ℓ' and c'. This policy exceeds each bin's risk budget at most once. Note that labels c' take values in $\{0, C + \delta, C + 2\delta\}$; we modify the label of arc (a, b) to $\min\{c'_{(a,b)}, C + \delta\}$ thus labels take values only in $\{0, C + \delta\}$. This does not

increases the cost of the policy tree.

For the second phase, we further modify $\mathcal{P}'$: If the policy tries to exceed some bin's risk budget, we open a new bin for that item, unless there is only one item packed in the bin, in which case we move to modify another bin. Using the notation of the first phase, this means that whenever the policy reaches node u_k in some branch starting at u, instead of packing the item in node u_k into bin j, it opens a new bin j'' for it. We call this new policy $\mathcal{P}''$. We modify the cost labels accordingly to accommodate this new cost. We label all nodes u_k in the subtree $\mathcal{T}_{\mathcal{P}'}(u)$ that are not leaves (i.e. the risk goes beyond γ/C at u_k) with $+1$ (the cost to open a new bin). All arcs going out of paths $u_1, \ldots, u_k$ are relabeled from $C + \delta$ to C. Formally, we define the new labels

$$c''_{(a,b)} = \begin{cases} \frac{C}{C+\delta} c'_{(a,b)} & a = u_m \text{ for some branch starting at } u \\ c'_{(a,b)} & \text{otherwise} \end{cases}$$

and for nodes

$$\ell''_a = \begin{cases} 1 & a = u_k \text{ and } a \text{ is not a leaf} \\ \ell'_a & \text{otherwise} \end{cases}.$$

Using the same argument as in Lemma 3.11, we can show that

$$\text{cost}_{\ell'',c''}(\mathcal{T}_{\mathcal{P}''}(u)) \leq \text{cost}_{c',\ell'}(\mathcal{T}_{\mathcal{P}'}(u)).$$

We repeat this procedure as many times as necessary, and we obtain a policy $\mathcal{P}''$ that satisfies

$$\text{cost}(\mathcal{P}'') \leq \text{cost}(\mathcal{P}') \leq \left(1 + 2\frac{\delta}{C}\right) \text{cost}(\mathcal{P}).$$

For ease of reading, we present the proof in an iterative manner; the proof's steps can be followed to obtain the result for finite and countably infinite policy trees. We now sketch

how to generalize the proof for uncountable policy trees, focusing on the first phase of the proof. We note that the proof of Lemma 3.11 is general and does not require any iterative argument. Recursively,

$$\mathrm{cost}_{\ell,c}(\mathcal{T}_{\mathcal{P}}(r)) = \mathop{\mathbf{E}}_{X_1,\ldots,X_{i-1}}\left[\sum_{k=1}^{i-1}\ell_{U_k} + \sum_{k=1}^{i-1} c_{(U_k,U_{k+1})} + \mathrm{cost}_{\ell,c}(\mathcal{T}_{\mathcal{P}}(U_i))\right]$$

for any $i = 1, \ldots, n$, where U_i is the (random) node at level i. Starting at the root, we apply Lemma 3.11 to all nodes at level i where a bin is opened. Therefore, we have

$$\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(u)) \geq \mathrm{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(u))$$

for all nodes u at level i. Using the previous equation,

$$\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(r)) - \mathrm{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}}(r)) = \mathop{\mathbf{E}}_{X_1,\ldots,X_{i-1}}\left[\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(U_i)) - \mathrm{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(U_i))\right] \geq 0.$$

Doing this for all levels $i = 1, \ldots, n$, we conclude the first phase of the proof. The second phase is completely analogous and omitted for brevity. □

Theorem 3.10 is a general result that does not depend on item size distributions. In the following, we use it to analyze the performance of Budgeted Greedy.

I.I.D. Input When the input is an i.i.d. sequence of nonnegative random variables, Budgeted Greedy induces a policy tree that packs one bin at a time: When a bin is opened, the policy never again uses previously opened bins. This simple fact is crucial in the proof of our next result. The next lemma shows that among all budgeted policies, Budgeted Greedy opens the minimum expected number of bins when the input is an i.i.d. sequence of random variables. Intuitively, if we ignore the penalty paid by overflowing bins and all items are i.i.d., the optimal way to minimize the expected number of opened bins is by packing as many items as possible in each bin, as long as the risk budget is satisfied. This can of

course be done sequentially, one bin at a time, which is what Budgeted Greedy does.

Lemma 3.12. *Suppose $X_1, \ldots, X_n$ are nonnegative i.i.d. random variables. Then,*

$$\mathbf{E}[N_{\mathrm{ALG}}] = \min_{\substack{\mathcal{P} \text{ budgeted with} \\ \text{risk budget } \gamma/C}} \mathbf{E}[N_{\mathcal{P}}].$$

That is, among all risk-budgeted policies with budget γ/C, Budgeted Greedy (Algorithm 5) opens the minimum expected number of bins.

Proof. Proof. Consider any policy $\mathcal{P}$ for packing items such that the risk budget of each bin γ/C is never surpassed. Consider its tree representation $\mathcal{T}_{\mathcal{P}}$. We modify the policy tree so only one bin is utilized at a time. For this, we exhibit a sequence of operations ensuring that, whenever bin j is opened, bins $1, \ldots, j-1$ are never utilized again. In the tree, this is equivalent to saying that any branch starting from the root directed to any leaf has labels $1 \to j_1, 2 \to j_2, \ldots, n \to j_n$ where $1 = j_1 \leq j_2 \leq \cdots \leq j_n$, where we recall that $i \to j$ means the policy packs item i into bin j.

Claim 3.13. *Let $j = 1, \ldots, n$ be any bin opened by the policy. Suppose that node u in level k is labeled $k \to j'$, where $j' \neq j$. Furthermore, suppose that at node u, bin j is open, its usage does not exceed 1, and its risk budget can accommodate k. Then u can be relabeled $k \to j$ without increasing the expected number of bins opened by the policy.*

Before proving this claim, we show how to use it to conclude the result. Starting at the root r of the policy tree $\mathcal{T}_{\mathcal{P}}$, find the closest node u to the root where we have the label $k \to j$, $j \neq 1$, but the usage of bin 1 is no more than 1 and its risk budget can accommodate k. Use the claim to relabel this node $k \to 1$ without increasing the expected number of open bins. Repeat this process until there are no nodes u in this category. After this process has been finished, all branches starting at the root have the form $1 \to 1, 2 \to 1, \ldots, i \to 1, i+1 \to$

$2, \ldots$ and from $i+1$ onward, bin 1 is overflowed or does not have enough risk budget to receive any additional item. We repeat this process with bin $2, 3, \ldots$. After this process has been carried out, the resulting policy is the one induced by Budgeted Greedy.

We prove the claim by backward induction on the level of node u in the policy tree. Fix an opened bin $j = 1, \ldots, n$ and pick any node u at level n with label $n \to j'$, $j' \neq j$, and suppose bin j is open and satisfies the hypothesis – its usage is one or less, and it has enough risk budget to receive item X_n. Re-labeling this node $n \to j$ does not worsen the number of bins opened since the cost of bin j has already been paid at some previous node. Recall that we are only taking into account the cost paid by opening bins and not the cost of breaking bins.

Suppose the result holds for all levels $k+1, k+2, \ldots, n$. Pick a node u at level k with label $k \to j'$, $j' \neq j$, and such that bin j is open and satisfies the hypothesis. If all its children are labeled $k+1 \to j$ then relabel all its children with $k+1 \to j'$ and relabel u with $k \to j$. The cost remains the same after this operation since the distribution of X_k is the same as X_{k+1}. Now, suppose that some child of u, say v, is labeled $k+1 \to m$ with $m \neq j$. Since at node u bin j still has usage not exceeding one and sufficient risk budget left, at node v bin j still satisfies this condition. Therefore, by induction, we can relabel v with $k+1 \to j$ without increasing the cost. We can repeat this for any children of u until all of its children have been labeled $k+1 \to j$. We conclude by swapping the label of u with the label of its children as in the previous case. □

Theorem 3.14. *For $\gamma = \sqrt{2}$ and i.i.d. nonnegative random variables $X_1, \ldots, X_n$, we have*

$$\operatorname{cost}(\textsc{Alg}) \leq (3 + 2\sqrt{2}) \operatorname{cost}(\textsc{Opt}).$$

Proof. If $\mathbf{P}(X_1 > 1) > \sqrt{2}/C$, $\operatorname{cost}(\textsc{Alg}) = n(1 + C\,\mathbf{P}(X_1 > 1))$. On the other hand, $\operatorname{cost}(\textsc{Opt}) \geq nC\,\mathbf{P}(X_1 > 1)$ since each item incurs at least this expected cost. Therefore,

$\text{cost}(\textsc{Alg}) \leq 2\,\text{cost}(\textsc{Opt})$.

Now assume $\mathbf{P}(X_1 > 1) \leq \sqrt{2}/C$. Using Theorem 3.10 and Lemma 3.12, we have

$$\text{cost}(\textsc{Alg}) \leq (1+\gamma)\,\mathbf{E}[N_{\mathcal{P}}] \leq (1+\gamma)\,\text{cost}(\textsc{Opt}^*) \leq (1+\gamma)\left(1+\frac{2}{\gamma}\right)\text{cost}(\textsc{Opt}),$$

where $\textsc{Opt}^*$ is the (γ/C)-budgeted version of $\textsc{Opt}$. The expression $(1+\gamma)(1+2/\gamma)$ is minimized at $\gamma = \sqrt{2}$, which gives the desired result. □

3.5 Exponential Random Variables

In this section, we show that Budgeted Greedy incurs an expected cost at most $\mathcal{O}(\log C)$ times the optimal expected cost when the item sizes are exponentially distributed. That is, for any X_i in the input sequence,

$$\mathbf{P}(X_i > x) = e^{-\lambda_i x}, \tag{3.1}$$

for any $x \geq 0$, where $\lambda_i > 0$ is the rate. Recall that $\mathbf{E}[X_i] = 1/\lambda_i$.

The proof is divided into two parts: First, similarly to deterministic bin packing, we show that $\mathbf{E}\left[\sum_{i=1}^{n} \min\{X_i, 1\}\right]$ is a lower bound for $\text{cost}(\mathcal{P})$, for any policy $\mathcal{P}$. In the next step, we show that the probability that Algorithm 5 opens bin $k \geq 2$ is related to the amount of mass packed into bin $k-1$. Roughly speaking, we show that the probability that Algorithm 5 opens bin $k \geq 2$ is at most $\mathcal{O}(\log C)$ times the expected mass packed into bin $k-1$. Moreover, in Subsection 3.5.2 we show that, when the rates governing the item sizes are sufficiently large, $\lambda_i \geq 2\log C$, the amount of mass packed into bin $k-1$ is at least a constant, thereby improving the algorithm's approximation factor to a constant. Finally, we show that our analysis of Algorithm 5 for exponential random variables is almost tight by exhibiting an input sequence that forces Budgeted Greedy to incur a cost $\Omega(\sqrt{\log C})$ times the optimal cost.

3.5.1 Arbitrary Exponential Random Variables

Here we show that $\text{cost}(\text{ALG}) \leq \mathcal{O}(\log C)\,\text{cost}(\text{OPT})$ when the input is an arbitrary sequence of exponential random variables. Using Proposition 3.6, we can assume $\mathbf{P}(X_i > 1) \leq 1/C$ for all $i = 1, \ldots, n$ at the expense of an extra multiplicative loss of 2 in the cost incurred. This assumption translates into $\lambda_i \geq \log C$ for all i.

The next result shows that the probability that Algorithm 5 opens a bin, besides the first bin, is related to the amount of mass packed in the previous bin.

Proposition 3.15. *Suppose $\gamma = 2$. Then, for any $k \geq 2$, Budgeted Greedy guarantees*

$$\begin{aligned}\mathbf{P}(\text{ALG } \textit{opens bin } B_k) \leq & 5 \log C\, \mathbf{E}\left[\sum_{i \in B_{k-1}} X_i \wedge 1\right] \\ & + \left(\frac{1}{3} + 8\sqrt{5}\sqrt{\frac{\log C}{C}}\right) \mathbf{P}(\text{ALG } \textit{opens bin } B_{k-1}).\end{aligned}$$

Proof. Since the item sizes are continuous random variables, we have $\mathbf{P}(\text{ALG opens bin } B_k) = \mathbf{P}(X(B_k) > 0)$. Now, we have

$$\mathbf{P}(\text{ALG opens bin } B_k) \leq \mathbf{P}\left(X(B_{k-1}) > \frac{1}{5 \log C}\right) + \mathbf{P}\left(X(B_{k-1}) \leq \frac{1}{5 \log C}, X(B_k) > 0\right).$$

We bound each term separately. To bound the first term we use Markov's inequality:

$$\mathbf{P}\left(X(B_{k-1}) > \frac{1}{5 \log C}\right) = \mathbf{P}\left(X(B_{k-1}) \wedge 1 > \frac{1}{5 \log C}\right) \leq 5 \log C\, \mathbf{E}[X(B_{k-1}) \wedge 1].$$

For the second term, we proceed as follows. Let E be the event "*all items packed in* B_{k-1}

have rate $\lambda_i \geq 2\log C$." Then

$$\mathbf{P}\left(X(B_{k-1}) \leq \frac{1}{5\log C}, X(B_k) > 0\right) \leq \mathbf{P}\left(X(B_{k-1}) \leq \frac{1}{5\log C}, X(B_k) > 0, E\right) + \mathbf{P}\left(X(B_{k-1}) \leq \frac{1}{5\log C} \mid \overline{E}\right)\mathbf{P}(X(B_{k-1}) > 0),$$

since $\overline{E}$, the event that *some item in* B_{k-1} *has rate* $\leq 2\log C$, is contained in the event "*Algorithm 5 opens bin* B_{k-1}."

Claim 3.16. $\mathbf{P}\left(X(B_{k-1}) \leq \frac{1}{5\log C} \mid \overline{E}\right) \leq 1 - e^{-2/5} \leq 1/3.$

Proof. If $X_{i_1}, \ldots, X_{i_m}$ are all the *large* items with rates $\lambda_{i_p} \leq 2\log C$, then, the events

$$M_p = \{X_{i_p} \text{ is the first large item packed into } B_{k-1}\}$$

satisfy $\overline{E} = \bigcup_{p=1}^{m} M_p$. Then,

$$\begin{aligned}
\mathbf{P}\left(X(B_{k-1}) \leq \frac{1}{5\log C} \mid \overline{E}\right) &= \sum_{p=1}^{m} \mathbf{P}\left(X(B_{k-1}) \leq \frac{1}{5\log C} \mid \overline{E}, M_p\right)\mathbf{P}(M_p \mid \overline{E}) \\
&\leq \sum_{p=1}^{m} \mathbf{P}\left(X_{i_p} \leq \frac{1}{5\log C}\right)\mathbf{P}(M_p \mid \overline{E}) \\
&= \sum_{p=1}^{m} (1 - e^{-\lambda_{i_p}/5\log C})\,\mathbf{P}(M_p \mid \overline{E}) \qquad \text{(Using (3.1))} \\
&\leq (1 - e^{-2/5}) \sum_{p=1}^{m} \mathbf{P}(M_p \mid \overline{E}). \qquad \text{(Using } \lambda_{i_p} \geq 2\log C\text{)}
\end{aligned}$$

The proof follows because the events M_p are disjoint and form $\overline{E}$. □

Claim 3.17. $\mathbf{P}\left(X(B_{k-1}) \leq \frac{1}{5\log C}, X(B_k) > 0, E\right) \leq 20\sqrt{\log C/C}\,\mathbf{P}(X(B_{k-1}) > 0).$

Proof. In this case, bin B_k has been opened even though B_{k-1} still has available space. That means that the element that opens bin B_k surpasses the budget of B_{k-1}. From here

we obtain,

$$\frac{2}{C} < \mathrm{Risk}(B_{k-1}) + \mathbf{P}(X_t > 1 - X(B_{k-1})) \le \mathrm{Risk}(B_{k-1}) + \frac{e^{1/5}}{C},$$

where X_t is the first item packed into B_k and we use $\lambda_t \ge \log C$ and (3.1), thus $\mathbf{P}(X_t > 1 - X(B_{k-1})) \le e^{1/5}/C$. Let F_β be the event "$\sum_{i \in B_{k-1}} \mathbf{E}[X_i] > \beta$"; by Markov's inequality,

$$\begin{aligned}
\mathbf{P}(F_\beta) &\le \frac{1}{\beta}\,\mathbf{E}\left[\sum_{i \in B_{k-1}} \mathbf{E}[X_i]\right] \\
&\le \frac{C}{C-1}\frac{1}{\beta}\,\mathbf{E}\left[\sum_{i \in B_{k-1}} \mathbf{E}[X_i \wedge 1]\right] && (\mathbf{E}[X_i \wedge 1] = (1 - e^{-\lambda_i})\,\mathbf{E}[X_i]) \\
&\le 2\frac{C}{C-1}\frac{1}{\beta}\,\mathbf{P}(X(B_{k-1}) > 0). && \text{(Proposition 3.7)}
\end{aligned}$$

Thus,

$$\begin{aligned}
&\mathbf{P}\left(X(B_{k-1}) \le \frac{1}{5\log C}, X(B_k) > 0, E\right) \\
&\le \mathbf{P}\left(X(B_{k-1}) \le \frac{1}{5\log C}, \mathrm{Risk}(B_{k-1}) > \frac{2 - e^{1/5}}{C}, E\right) \\
&\le \frac{2C}{\beta(C-1)}\,\mathbf{P}(X(B_{k-1}) > 0) \\
&\quad + \mathbf{P}\left(X(B_{k-1}) \le \frac{1}{5\log C}, \mathrm{Risk}(B_{k-1}) > \frac{2 - e^{1/5}}{C}, \overline{F}_\beta, E\right) \\
&\le \frac{2C}{\beta(C-1)}\,\mathbf{P}(X(B_{k-1}) > 0) \\
&\quad + \frac{C}{2 - e^{1/5}}\,\mathbf{E}\left[\mathrm{Risk}(B_{k-1}) \mid X(B_{k-1}) \le \frac{1}{5\log C}, E, \overline{F}_\beta\right].
\end{aligned}$$

Claim 3.18. $\mathbf{E}\left[\mathrm{Risk}(B_{k-1}) \mid X(B_{k-1}) \le \frac{1}{5\log C}, E, \overline{F}_\beta\right] \le 10\beta C^{-2}\log C\,\mathbf{P}(X(B_{k-1}) > 0)$.

Proof. Given $X(B_{k-1}) \le 1/5\log C$, the event E, the event $\overline{F}_\beta$ *and* the event $X(B_{k-1}) > 0$, the value $\mathrm{Risk}(B_{k-1}) = \sum_{i=1}^n \mathbf{P}(X_i + S_{k-1}^{i-1} > 1)\mathbf{1}^{\mathrm{ALG}}_{\{i \to k-1\}} \le \sum_{i=1}^n e^{-\lambda_i(1 - 1/5\log C)}\mathbf{1}^{\mathrm{ALG}}_{\{i \to k-1\}}$

can be upper bounded by the non-convex problem:

$$\max_{x_1,\ldots,x_n} \left\{ \sum_{i=1}^{n} e^{-x_i(1-1/5\log C)} : \sum_{i=1}^{n} 1/x_i \le \beta, x_i \ge 2\log C, \forall i = 1,\ldots,n \right\} \le 10\beta \frac{\log C}{C^2}.$$

The inequality follows because the maximum of a convex function is attained in the boundary of the feasible set. Indeed, the maximum is attained by setting the maximum variables to the bound $2\log C$—which are at most $2\beta\log C$—and the rest of the variables to $+\infty$. □

Therefore, by upper bounding $10/(2-e^{1/5})$ by 20 and $C/(C-1)$ by 2, we obtain

$$\mathbf{P}\left(X(B_{k-1}) \le \frac{1}{5\log C}, X(B_k) > 0, E \right) \le \left(\frac{4}{\beta} + 20\beta\frac{\log C}{C} \right) \mathbf{P}(X(B_{k-1}) > 0).$$

The right-hand side is minimized at $\beta = \sqrt{C/5\log C}$. □

Putting Claims 3.16 and 3.17 together we obtain

$$\mathbf{P}(X(B_k) > 0) \le 5\log C\, \mathbf{E}\left[\sum_{i\in B_{k-1}} X_i \wedge 1 \right] + \left(\frac{1}{3} + 8\sqrt{5}\sqrt{\frac{\log C}{C}} \right) \mathbf{P}(X(B_{k-1}) > 0).$$

□

Proposition 3.19. *Let $X_1,\ldots,X_n$ be arbitrary exponential random variables with $\lambda_i \ge \log C$. Algorithm 5 with $\gamma = 2$ guarantees*

$$\mathrm{cost}(\textsc{Alg}) \le \frac{15\log C}{2/3 - 8\sqrt{5\log C/C}}\,\mathrm{cost}(\textsc{Opt}) + \frac{3}{2/3 - 8\sqrt{5\log C/C}},$$

where Opt *is the optimal policy that knows n and the rates of all the sizes $X_1,\ldots,X_n$ in advance.*

Proof. Using Proposition 3.15 we obtain

$$\begin{aligned}
\mathbf{E}[N_{\mathrm{ALG}}] &= 1 + \sum_{k\geq 2} \mathbf{P}(X(B_k > 0)) \\
&\leq 1 + 5\log C \sum_{k\geq 2} \mathbf{E}\left[\sum_{i\in B_{k-1}^{\mathrm{ALG}}} X_i \wedge 1\right] + \left(\frac{1}{3} + 8\sqrt{5}\sqrt{\frac{\log C}{C}}\right) \sum_{k\geq 2} \mathbf{P}(X(B_{k-1}) > 0) \\
&\leq 1 + 5\log C \sum_{k\geq 1} \mathbf{E}\left[\sum_{i\in B_{k}^{\mathrm{ALG}}} X_i \wedge 1\right] + \left(\frac{1}{3} + 8\sqrt{5}\sqrt{\frac{\log C}{C}}\right) \mathbf{E}[N_{\mathrm{ALG}}] \\
&= 1 + 5\log C\, \mathbf{E}\left[\sum_{i} X_i \wedge 1\right] + \left(\frac{1}{3} + 8\sqrt{5}\sqrt{\frac{\log C}{C}}\right) \mathbf{E}[N_{\mathrm{ALG}}].
\end{aligned}$$

For any policy $\mathcal{P}$ we have $\mathbf{E}\left[\sum_{i=1}^{n} X_i \wedge 1\right] \leq \mathrm{cost}(\mathcal{P})$, using Proposition 3.8. Then,

$$\mathbf{E}[N_{\mathrm{ALG}}] \leq \frac{5\log C}{2/3 - 8\sqrt{5\log C/C}}\, \mathbf{E}[N_{\mathcal{P}}] + \frac{1}{2/3 - 8\sqrt{5\log C/C}}$$

The conclusion follows from here using $\mathrm{cost}(\textsc{Alg}) \leq 3\,\mathbf{E}[N_{\mathrm{ALG}}]$ (Corollary 3.9.1). □

3.5.2 Small Exponential Random Variables

We next show that $\mathrm{cost}(\textsc{Alg}) \leq \mathcal{O}(1)\,\mathrm{cost}(\textsc{Opt})$ whenever the item sizes are independent exponential random variables with rates satisfying $\lambda_i \geq 2\log C$. In this case, $\mathbf{E}\left[\sum_i X_i \wedge 1\right]$ is a better approximation for $\mathbf{E}[N_{\mathrm{ALG}}]$ than in the general case. The following results shows that we can improve Proposition 3.15 by a logarithmic factor.

Proposition 3.20. *Let $\gamma = 1$. For $k \geq 2$,*

$$\mathbf{P}(\textsc{Alg} \textit{ opens bin } k) \leq 4\,\mathbf{E}\left[\sum_{i\in B_{k-1}} X_i \wedge 1\right] + 8\frac{\sqrt{\log C}}{C^{1/4}}\,\mathbf{P}(\textsc{Alg} \textit{ opens bin } k-1).$$

Proof. We have

$$\begin{aligned}\mathbf{P}(\text{ALG opens bin } B_{k+1}) &= \mathbf{P}(X(B_{k+1}) > 0) \\ &\leq \mathbf{P}(X(B_k) > 1/4) + \mathbf{P}(X(B_{k+1}) > 0, X(B_k) \leq 1/4) \\ &\leq \mathbf{P}(X(B_k) \wedge 1 > 1/4) + \mathbf{P}(X(B_{k+1}) > 0, X(B_k) \leq 1/4) \\ &\leq 4\,\mathbf{E}\left[X(B_k) \wedge 1\right] + \mathbf{P}(X(B_{k+1}) > 0, X(B_k) \leq 1/4).\end{aligned}$$

We only focus on bounding the second term in the rest of the proof. Algorithm 5 opens bin B_{k+1} ($X(B_{k+1}) > 0$) if there are no available bins ($\forall i \leq k$, $X(B_i) \geq 1$) or there is an item that does not fit because of the budget. The first case cannot happen when the event $X(B_k) \leq 1/4$ happens so we are only left with the budget case. In particular, for bin k, we open bin B_{k+1} because for some item X_t we have

$$\frac{1}{C} < \text{Risk}(B_k) + \mathbf{P}(X_t + X(B_k) > 1) \leq \text{Risk}(B_k) + \mathbf{P}(X_t > 3/4) \leq \text{Risk}(B_k) + \frac{1}{C^{3/2}}$$

where we used the information from the event $X(B_{k+1}) \leq 1/4$. Therefore,

$$\begin{aligned}&\mathbf{P}(X(B_{k+1}) > 0, X(B_k) \leq 1/4) \leq \mathbf{P}(\text{Risk}(B_k) > 1/C - 1/C^{3/2}, 0 < X(B_k) < 1/4) \\ &\leq \frac{C^{3/2}}{C^{1/2} - 1}\,\mathbf{E}[\text{Risk}(B_k) \mid X(B_k) < 1/4, X(B_k) > 0]\,\mathbf{P}(X(B_k) > 0).\end{aligned}$$

Now, as in the previous proof, let $F_\beta = \left\{\sum_{i \in B_k} \mathbf{E}[X_i] > \beta\right\}$; by Markov's inequality and Proposition 3.7,

$$\mathbf{P}\left(F_\beta\right) \leq \frac{2C^2}{\beta(C^2 - 1)}\,\mathbf{P}(X(B_k) > 0).$$

Claim 3.21. $\mathbf{E}[\text{Risk}(B_k)\mathbf{1}_{\{X(B_k) < 1/4\}} \mid X(B_k) < 1/4, X(B_k) > 0, \overline{F}_\beta] \leq \frac{2\beta \log C}{C^{3/2}}$.

Proof. Given $X(B_k) < 1/4, X(B_k) > 0, \overline{F}_\beta$, the risk

$$\mathrm{Risk}(B_k) = \sum_{i=1}^{n} \mathbf{P}(X_i + S_k^{i-1} > 1)\mathbf{1}_{\{i \to k\}} \leq \sum_{i=1}^{n} e^{-3\lambda_i/4}\mathbf{1}_{\{i \to k\}}$$

is bounded by the non-convex problem,

$$\max_{x_1,\ldots,x_n} \left\{ \sum_{i=1}^{n} e^{-3x_i/4} : \sum_{i=1}^{n} \frac{1}{x_i} \leq \beta, x_i \geq 2\log C, \forall i = 1,\ldots,n \right\} \leq \frac{2\beta \log C}{C^{3/2}},$$

which we bound as before. □

With this claim,

$$\begin{aligned}\mathbf{E}[\mathrm{Risk}(B_k) \mid X(B_k) < 1/4, X(B_k) > 0] &\leq \frac{1}{C}\,\mathbf{P}(F_\beta) + \frac{2\beta\log C}{C^{3/2}}\,\mathbf{P}(\overline{F}_\beta) \\ &\leq \frac{2C}{\beta(C^2-1)} + \frac{2\beta\log C}{C^{3/2}},\end{aligned}$$

since $\mathrm{Risk}(B_k) \leq 1/C$. Thus,

$$\mathbf{P}(X(B_{k+1}) > 0, X(B_k) \leq 1/4) \leq \frac{C^{3/2}}{C^{1/2}-1}\left(\frac{2C}{\beta(C^2-1)} + \frac{2\beta\log C}{C^{3/2}}\right)\mathbf{P}(X(B_k) > 0).$$

Now, optimizing over β with $\beta = \frac{C^{5/4}}{\sqrt{C^2-1}\sqrt{\log C}}$ we obtain

$$\mathbf{P}(X(B_{k+1}) > 0, X(B_k) \leq 1/4) \leq 8\frac{\sqrt{\log C}}{C^{1/4}}\,\mathbf{P}(X(B_k) > 0).$$

□

Proposition 3.22. *Suppose $\lambda_i \geq 2\log C$ for all $i = 1,\ldots,n$. For $\gamma = 1$, Algorithm 5 guarantees*

$$\mathrm{cost}(\textsc{Alg}) \leq \frac{8}{1 - 8\sqrt{\log C}/C^{1/4}}\,\mathrm{cost}(\textsc{Opt}) + \frac{2}{1 - 8\sqrt{\log C}/C^{1/4}},$$

where OPT *is the optimal policy that knows* n *and all item size rates in advance.*

Proof. Using Proposition 3.20 we have

$$\begin{aligned}\mathbf{E}[N_{\mathrm{ALG}}] &= \sum_{k=1}^{n} \mathbf{P}(\text{ALG opens bin } k) \\ &\leq 1 + \sum_{k=2}^{n} 4\,\mathbf{E}\left[\sum_{i\in B_{k-1}} X_i \wedge 1\right] + 8\frac{\sqrt{\log C}}{C^{1/4}}\,\mathbf{P}(\text{ALG opens bin } k-1) \\ &\leq 1 + 4\,\mathbf{E}\left[\sum_{i=1}^{n} X_i \wedge 1\right] + 8\frac{\sqrt{\log C}}{C^{1/4}}\,\mathbf{E}[N_{\mathrm{ALG}}].\end{aligned}$$

Using Proposition 3.8,

$$\mathbf{E}[N_{\mathrm{ALG}}] \leq \frac{4}{1 - 8\sqrt{\log C}/C^{1/4}}\,\mathrm{cost}(\mathcal{P}) + \frac{1}{1 - 8\sqrt{\log C}/C^{1/4}}$$

for any policy $\mathcal{P}$. The result follows by using $\mathrm{cost}(\text{ALG}) \leq 2\,\mathbf{E}[N_{\mathrm{ALG}}]$ and optimizing over $\mathcal{P}$. □

3.5.3 A Lower Bound for the Algorithm with Exponential Random Variables

In this subsection, we present a hard input of exponential random variables for Budgeted Greedy. The sequence contains two kind of independent exponential random variables, those with rates $\mu = \beta \log C$, $\beta \geq 2$ and those with rates $\lambda = (1+\varepsilon)\log C$, with $\varepsilon \in (0,1)$. This sequence has n_1 items with rate λ and $n_2 = kn_1$ items with rate μ, presented to Algorithm 5 as,

$$X^{\mu}_{1,1} \cdots X^{\mu}_{1,k} X^{\lambda}_{1} X^{\mu}_{1,1} \cdots X^{\mu}_{2,k} X^{\lambda}_{2} \quad \cdots \quad X^{\mu}_{n_1,1} \cdots X^{\mu}_{n_1,k} X^{\lambda}_{n_1},$$

where $X^{\mu}_{i,j} \sim \exp(\mu)$ and $X^{\lambda}_{i} \sim \exp(\lambda)$ for all i, j. With the choices of $\beta = 6\frac{n_1 \log C}{\varepsilon}$ and $k = 3\varepsilon\mu = 18 n_1 (\log C)^2$, we show that Budgeted Greedy incurs an expected cost of at least $\frac{1}{2}n_1$. For the same choices of β and k and optimizing over the choice of ε, we show

that $\mathrm{cost}(\textsc{Opt}) \leq \mathcal{O}\left(1/\sqrt{\log C}\right) n_1$. This choice of ε is independent of n_1, which allows us to scale the result for any input size.

We prove each bound separately; the main results are stated here.

Proposition 3.23. *Let $\varepsilon > 0$ and set $\beta = 6\varepsilon^{-1} n_1 \log C$ and $k = 3\varepsilon\mu$. Then, running Budgeted Greedy with $\gamma = 1$ on the input described above yields*

$$\mathrm{cost}(\textsc{Alg}) \geq \frac{1}{2} n_1.$$

Proposition 3.24. *Using the same parameters as in the previous proposition, for any $\varepsilon > 0$ such that $\varepsilon \log C \geq 4$, we have*

$$\mathrm{cost}(\textsc{Opt}) \leq 48 n_1 \left(\frac{k}{\beta \log C} + \frac{1}{\varepsilon \log C} \right) = 48 n_1 \left(3\varepsilon + \frac{1}{\varepsilon \log C} \right).$$

The result now follows by taking $\varepsilon = 1/\sqrt{3 \log C}$. The proofs are in Appendix 3.A.

3.6 Offline Sequential Adaptive Bin Packing

In this section, we move to the offline sequential model, where random variables are known in advance and the packing occurs sequentially in the fixed order $1, \ldots, n$. We present the proof of Theorem 3.4 that guarantees a soft-capacity polynomial time approximation scheme (PTAS) for the offline problem.

3.6.1 Approximation of a Sequential Policy

Consider $X_1, \ldots, X_n$ independent random variables with bounded support $[0, 1 + \varepsilon]$. We can reduce the general case to this case by moving all the probability mass of the corresponding random variable in $[1+\varepsilon, \infty)$ to the point $1+\varepsilon$. We aim to show a polynomial time

approximation scheme with resource augmentation. In particular, we consider a policy operating on bins with size or capacity $c \geq 1$; a bin overflows if the total size of items packed into it exceeds c. We use the notation $\mathrm{cost}_c(\mathcal{P}, Z)$ to denote the expected cost incurred by a policy $\mathcal{P}$ packing items $Z = (Z_1, \ldots, Z_n)$ into bins of capacity c.

Theorem 3.25. *There is a policy that can be computed in $\mathcal{O}\left(n^{2/\varepsilon^5}/\varepsilon^{10}\right)$ time packing items $X_1, \ldots, X_n$ sequentially into bins of size $1 + 6\varepsilon$, and incurring expected cost of at most $(1 + 4\varepsilon)\,\mathrm{cost}_1(\textsc{Opt}, X)$, where* Opt *is an optimal policy with respect to bins of unit size.*

To prove Theorem 3.25, we proceed as follows in the remainder of the section:

1. First, we discretize the input random variables $X_1, \ldots, X_n$ into random variables $\widehat{X}_1, \ldots, \widehat{X}_n$ with support in $\{0, \varepsilon^5, \ldots, \lceil 2/\varepsilon^5 \rceil \varepsilon^5\}$. This allows us to compute an optimal policy in polynomial time via dynamic programming.

2. We then show that for any policy $\mathcal{P}$ for $X_1, \ldots, X_n$, we can construct a policy $\widehat{\mathcal{P}}$ for $\widehat{X}_1, \ldots, \widehat{X}_n$ such that

$$\mathrm{cost}_{1+4\varepsilon}\left(\widehat{\mathcal{P}}, \widehat{X}\right) \leq (1 + \varepsilon)\,\mathrm{cost}_1(\mathcal{P}, X).$$

3. Next, we show how to obtain a policy $\mathcal{P}$ for $X_1, \ldots, X_n$ from a policy $\widehat{\mathcal{P}}$ for items $\widehat{X}_1, \ldots, \widehat{X}_n$, such that

$$\mathrm{cost}_{1+6\varepsilon}(\mathcal{P}, X) \leq (1 + \varepsilon)\,\mathrm{cost}_{1+4\varepsilon}\left(\widehat{\mathcal{P}}, \widehat{X}\right).$$

4. Finally, we show that we can compute the optimal policy $\widehat{\mathcal{P}}$ for discretized items $\widehat{X}_1, \ldots, \widehat{X}_n$ in $\mathcal{O}\left(n^{2/\varepsilon^5}/\varepsilon^{10}\right)$ time. The policy $\mathcal{P}$ follows immediately from here.

3.6.2 Discretization Process

We perform the discretization in two steps, similarly to the discretization in [119]. In the first step, we discretize the small outcomes of $X_1, \ldots, X_n$, meaning that values not exceeding ε^4 now behave as a scaled Bernoulli random variable with scaling factor ε^4, and the appropriate success probability such that this discretization preserves the expectation of the original random variable. In the second step, we discretize the large outcomes by rounding up all values to multiples of ε^5.

This discretization allows us to construct a state space based on the number of bins at level $k\varepsilon^5$, $k = 1, \ldots, \lceil 2/\varepsilon^5 \rceil$. The number of states is roughly $\mathcal{O}(n^{2/\varepsilon^5})$, which is polynomial in n.

Step 1 of discretization Let $q_i = \mathbf{E}[X_i \mid X_i \leq \varepsilon^4]$. Then, the first discretization is

$$X_i' = \begin{cases} 0 & \text{if } X_i \leq \varepsilon^4, \text{ w.p. } 1 - q_i/\varepsilon^4 \\ \varepsilon^4 & \text{if } X_i \leq \varepsilon^4, \text{ w.p. } q_i/\varepsilon^4 \\ X_i & \text{if } X_i > \varepsilon^4. \end{cases}$$

Note that we have $|X_i - X_i'| \leq \varepsilon^4$ almost surely, $\mathbf{E}[X_i' \mid X_i' \leq \varepsilon^4] = \mathbf{E}[X_i \mid X_i \leq \varepsilon^4]$ and $\mathbf{E}[X_i] = \mathbf{E}[X_i']$.

Step 2 of discretization Now consider

$$\widehat{X}_i = \mathbf{1}_{\{X_i' \leq \varepsilon^4\}} X_i' + \mathbf{1}_{\{X_i' > \varepsilon^4\}} \lceil X_i'/\varepsilon^5 \rceil \varepsilon^5.$$

Clearly, $X_i' \leq \widehat{X}_i$, since the large outcomes are rounded up. Moreover, if $b > \varepsilon^4$, then $\lceil b/\varepsilon^5 \rceil \varepsilon^5 \leq (b/\varepsilon^5 + 1)\,\varepsilon^5 = b + \varepsilon^5 \leq (1+\varepsilon)b$. Hence, $X_i' \leq \widehat{X}_i \leq (1+\varepsilon)X_i'$.

3.6.3 From Regular Policy to Discretized Policy

In this subsection we show the following result:

Theorem 3.26. *For any policy $\mathcal{P}$ that sequentially packs items $X_1, \ldots, X_n$ into bins of unit size, there exists a policy $\widehat{\mathcal{P}}$ packing items $\widehat{X}_1, \ldots, \widehat{X}_n$ into bins of size $1 + 4\varepsilon$ such that*

$$\mathrm{cost}_{1+4\varepsilon}\left(\widehat{\mathcal{P}}, \widehat{X}\right) \leq (1+\varepsilon)\,\mathrm{cost}_1\left(\mathcal{P}, X\right).$$

To prove the theorem, we first introduce an intermediate policy $\mathcal{P}'$ that packs items $X'_1, \ldots, X'_n$ and satisfying

$$\mathrm{cost}_{1+2\varepsilon}(\mathcal{P}', X') \leq (1+\varepsilon)\,\mathrm{cost}_1(\mathcal{P}, X).$$

Policy $\widehat{\mathcal{P}}$ is obtained from policy $\mathcal{P}'$ by adding additional capacity to the bins.

We assume that $\mathcal{P}$ is a deterministic function of the capacity of the bins and the current element to be packed. Let us construct a policy $\mathcal{P}'$ for items $X'_1, \ldots, X'_n$ with bin capacity $1 + 2\varepsilon$ that simulates and follows policy $\mathcal{P}$ in the following way. Upon arrival of item X'_i, policy $\mathcal{P}'$ does what $\mathcal{P}$ would have done at this point in time to item X_i. We couple X_i and X'_i, so $X_i = X'_i$ if $X_i > \varepsilon^4$ and otherwise we have the Bernoulli behavior in X'_i. We pass the outcome of X'_i to $\mathcal{P}'$ and the outcome of X_i to $\mathcal{P}$. (Strictly speaking, $\mathcal{P}'$ receives the outcome of X'_i and from it, the policy samples X_i coupled with X'_i and passes this outcome to $\mathcal{P}$.) For each bin B_j that policy $\mathcal{P}$ opens, policy $\mathcal{P}'$ opens a bin $B'_{j,1}$ and packs items in $B'_{j,1}$ as policy $\mathcal{P}$ would do in bin B_j as long as $|X(B'_{j,1}) - X'(B'_{j,1})| \leq \varepsilon$ holds. If this difference is violated, policy $\mathcal{P}'$ opens a new bin $B'_{j,2}$ and continues following $\mathcal{P}$ as long as $|X(B'_{j,2}) - X'(B'_{j,2})| \leq \varepsilon$ holds, and so on.

Notice that $\mathcal{P}'$ is undefined if some $B'_{j,k}$ breaks but B_j is not broken by policy $\mathcal{P}$. Fortunately, this event cannot occur, as the following proposition guarantees.

Proposition 3.27. *If $\mathcal{P}$ overflows $B'_{j,k}$ for some k, then $\mathcal{P}$ must have overflown B_j. In particular, at most one of the $B'_{j,k}$ is overflowed by $\mathcal{P}'$.*

Proof. Let $B'_{j,k,t}$ be the items packed into bin $B'_{j,k}$ up to time t. Suppose that $X'(B'_{j,k,t}) > 1 + 2\varepsilon$ ($B'_{j,k}$ is overflowed at time t). Notice that $X'_t \leq 1 + \varepsilon$, therefore $B'_{j,k}$ was opened before t and

$$|X'(B'_{j,k,t-1}) - X(B'_{j,k,t-1})| \leq \varepsilon$$

otherwise $\mathcal{P}'$ would not have tried to pack X_t into $B'_{j,k}$. Now, since $|X'_t - X_t| \leq \varepsilon^4$ we have

$$\begin{aligned} X(B_j) &\geq X(B'_{j,k,t}) \\ &= X(B'_{j,k,t-1}) + X_t \geq X'(B_{j,k,t-1}) - \varepsilon + X'_t - \varepsilon^4 \\ &= X'(B'_{j,k,t}) - \varepsilon - \varepsilon^4 \\ &> 1 + \varepsilon - \varepsilon^4 \\ &> 1. \end{aligned}$$

For the second part, we notice that once $B'_{j,k}$ is overflowed, then B_j is overflowed as well and so $\mathcal{P}$ does not pack any item in B_j. Then, after $B'_{j,k}$ no more bins $B'_{j,k+1}, \ldots$ are open. □

Let $O_{\mathcal{P}',j}$ be the number of bins $B'_{j,k}$ that policy $\mathcal{P}'$ breaks; we just showed that $O_{\mathcal{P}',j} \leq \mathbf{1}^{\mathcal{P}}_{\{X(B_j)>1\}}$. Then, $O_{\mathcal{P}'} = \sum_{j=1}^{n} O_{\mathcal{P}',j}$, the number of bins overflowed by $\mathcal{P}'$, satisfies the following equality.

Proposition 3.28. $\mathbf{E}[O_{\mathcal{P}'}] \leq \mathbf{E}[O_{\mathcal{P}}]$.

Proof. By the previous proposition, at most one of the $B'_{j,1}, \ldots, B'_{j,n}$ breaks, and when it does then B_j must have been broken as well. □

Next, we show that the number of bins opened by $\mathcal{P}'$ is not much larger than the number of bins opened by $\mathcal{P}$. Let $N_{\mathcal{P}',j}$ be the number of bins $B'_{j,1}, B'_{j,2}, \ldots$ that policy $\mathcal{P}'$ uses, i.e. the number of copies of bin B_j used by policy $\mathcal{P}$. Let $N_{\mathcal{P}'}$ be the number of bins opened by policy $\mathcal{P}'$, $N_{\mathcal{P}'} = \sum_{j=1}^{n} N_{\mathcal{P}',j}$.

Consider the family of events $\mathcal{E}'_{j,k} = \{|X(B'_{j,k}) - X'(B'_{j,k})| > \varepsilon\}$ for $k \geq 1$ and for $k = 0$ define $\mathcal{E}'_{j,0} = \{\mathcal{P}' \text{ opens bin } B'_{j,1}\} = \{\mathcal{P} \text{ opens } B_j\}$. Notice then, for $\ell \geq 1$,

$$\{N_{\mathcal{P}',j} \geq \ell\} \subseteq \mathcal{E}'_{j,0} \cap \mathcal{E}'_{j,1} \cap \cdots \cap \mathcal{E}'_{j,\ell-1}. \tag{3.2}$$

Proposition 3.29. *For any $k \geq 1$,*

$$\mathbf{P}(\mathcal{E}'_{j,k} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}) \leq 6\varepsilon^2 \, \mathbf{P}(\mathcal{P}' \textit{ opens } B'_{j,k} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}).$$

Proof. Using Chebychev's inequality,

$$\begin{aligned}
&\mathbf{P}(\mathcal{E}'_{j,k} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}) \\
&\leq \frac{1}{\varepsilon^2} \mathbf{E}\left[\left(X(B'_{j,k}) - X'(B'_{j,k})\right)^2 \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}\right] \\
&= \frac{1}{\varepsilon^2} \mathbf{E}\left[\left(\sum_{i=1}^{n} (X_i - X'_i) \mathbf{1}^{\mathcal{P}'}_{\{i \to (j,k)\}}\right)^2 \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}\right] \\
&= \frac{1}{\varepsilon^2} \sum_{i=1}^{n} \mathbf{E}\left[(X_i - X'_i)^2 \mathbf{1}^{\mathcal{P}'}_{\{i \to (j,k)\}} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}\right] \\
&\quad + \frac{2}{\varepsilon^2} \sum_{i<\ell} \mathbf{E}\left[(X_i - X'_i)(X_\ell - X'_\ell) \mathbf{1}^{\mathcal{P}'}_{\{i \to (j,k), \ell \to (j,k)\}} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}\right].
\end{aligned}$$

Claim 3.30. *For $i < \ell$, $\mathbf{E}\left[(X_i - X'_i)(X_\ell - X'_\ell) \mathbf{1}^{\mathcal{P}'}_{\{i \to (j,k), \ell \to (j,k)\}} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}\right] = 0$.*

Proof. If $\mathbf{P}(i \to (j,k), \ell \to (j,k) \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}) = 0$ the result is clearly true,

while in the opposite case

$$\begin{aligned}
&\mathbf{E}\left[(X_i - X_i')(X_\ell - X_\ell')\mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k),\ell\to(j,k)\}} \mid \mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}\right] \\
&\qquad = \mathbf{E}\left[(X_i - X_i')(X_\ell - X_\ell') \mid i\to(j,k),\ell\to(j,k),\mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}\right] \\
&\qquad\quad \times \mathbf{P}(i\to(j,k),\ell\to(j,k) \mid \mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}) \\
&\qquad = \mathbf{E}[X_\ell - X_\ell']\,\mathbf{E}\left[X_i - X_i' \mid i\to(j,k),\ell\to(j,k),\mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}\right] \\
&\qquad\quad \times \mathbf{P}(i\to(j,k),\ell\to(j,k) \mid \mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}) \\
&\qquad = \mathbf{E}\left[\mathbf{E}[X_\ell - X_\ell'](X_i - X_i')\mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k),\ell\to(j,k)\}} \mid \mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}\right] \\
&\qquad = 0,
\end{aligned}$$

the last result since $\mathbf{E}[X_\ell] = \mathbf{E}[X_\ell']$. Note that from the second to the third equality, we utilized the fact that given that ℓ is packed into $B'_{j,k}$, the outcome of $X_\ell - X_\ell'$ is independent of previous $\mathcal{E}'_{j,m}$, $m < k$. □

Claim 3.31. *For any i,*

$$\mathbf{E}\left[(X_i - X_i')^2\mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k)\}} \mid \mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}\right] \le 2\varepsilon^4\,\mathbf{E}\left[\mathbf{E}[X_i]\mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k)\}} \mid \mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}\right].$$

Proof. We have $|X_i - X_i'| \le \varepsilon^4$, so

$$\begin{aligned}
&\mathbf{E}\left[(X_i - X_i')^2\mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k)\}} \mid \mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}\right] \\
&\qquad\qquad \le \varepsilon^4\,\mathbf{E}\left[|X_i - X_i'|\mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k)\}} \mid \mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}\right] \\
&\qquad\qquad \le \varepsilon^4\,\mathbf{E}\left[(X_i + X_i')\mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k)\}} \mid \mathcal{E}'_{j,k-1},\ldots,\mathcal{E}'_{j,1},\mathcal{E}'_{j,0}\right].
\end{aligned}$$

The sizes of X_i and X_i' are independent of the policy $\mathcal{P}'$ packing i into bin $B'_{j,k}$. Furthermore, if X_i' is packed into bin $B'_{j,k}$, its size does not depend on previous events $\mathcal{E}'_{j,\ell}$, $\ell < k$.

Therefore,

$$\begin{aligned}
&\mathbf{E}\left[X_i' \mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k)\}} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}\right] \\
&= \mathbf{E}[X_i'] \, \mathbf{P}(\mathcal{P}' \text{ packs } i \text{ into } (j,k) \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}) \\
&= \mathbf{E}\left[\mathbf{E}[X_i] \mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k)\}} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}\right]
\end{aligned}$$

since $\mathbf{E}[X_i'] = \mathbf{E}[X_i]$. Similarly,

$$\mathbf{E}\left[X_i \mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k)\}} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}\right] = \mathbf{E}\left[\mathbf{E}[X_i] \mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k)\}} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}\right].$$

□

Putting these two claims together in the previous inequality gives us

$$\begin{aligned}
\mathbf{P}(\mathcal{E}'_{j,k} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}) &\le \frac{1}{\varepsilon^2} \sum_{i=1}^{n} 2\varepsilon^4 \, \mathbf{E}\left[\mathbf{E}[X_i] \mathbf{1}^{\mathcal{P}'}_{\{i\to(j,k)\}} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}\right] \\
&\le 2\varepsilon^2 \, \mathbf{E}\left[\sum_{i \in B'_{j,k}} \mathbf{E}[X_i] \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}\right] \\
&\le 6\varepsilon^2 \, \mathbf{P}(\mathcal{P}' \text{ opens } B'_{j,k} \mid \mathcal{E}'_{j,k-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}).
\end{aligned}$$

In the last inequality, we used Proposition 3.7, $X_i \le 1+\varepsilon$ for all i, and the bins $B'_{j,k}$ having capacity $1 + 2\varepsilon$. □

Recall that $N_{\mathcal{P}',j}$ is the number of bins $B'_{j,1}, \ldots$ that policy $\mathcal{P}'$ uses. We have,

Proposition 3.32. *For any $j = 1, \ldots, n$,*

$$\mathbf{E}[N_{\mathcal{P}',j}] \le (1+\varepsilon) \, \mathbf{P}(\mathcal{P} \textit{ opens } B_j).$$

Proof. Using the inclusion (3.2) and the previous proposition,

$$\begin{aligned}\mathbf{P}(N_{\mathcal{P}',j} \geq \ell) &\leq \mathbf{P}(\mathcal{E}'_{j,\ell-1}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}) \\ &= \mathbf{P}(\mathcal{E}'_{j,\ell-1} \mid \mathcal{E}'_{j,\ell-2}, \ldots, \mathcal{E}'_{j,1}, \mathcal{E}'_{j,0}) \cdots \mathbf{P}(\mathcal{E}'_{j,1} \mid \mathcal{E}'_{j,0})\, \mathbf{P}(\mathcal{E}'_{j,0}) \\ &\leq (6\varepsilon^2)^{\ell-1}\, \mathbf{P}(\mathcal{P} \text{ opens } B_j).\end{aligned}$$

Thus,

$$\begin{aligned}\mathbf{E}[N_{\mathcal{P}',j}] &= \sum_{\ell \geq 1} \mathbf{P}(N_{\mathcal{P}',j} \geq \ell) \\ &\leq \sum_{\ell \geq 1} (6\varepsilon^2)^{\ell-1}\, \mathbf{P}(\mathcal{P} \text{ opens } B_j) \\ &= \frac{1}{(1-6\varepsilon^2)}\, \mathbf{P}(\mathcal{P} \text{ opens } B_j) \\ &\leq (1+\varepsilon)\, \mathbf{P}(\mathcal{P} \text{ opens } B_j).\end{aligned}$$

For the last inequality we require $\varepsilon \leq (\sqrt{15} - 3)/\sqrt{6} \approx 0.1454$. □

Corollary 3.32.1. $\mathbf{E}[N_{\mathcal{P}'}] \leq (1+\varepsilon)\, \mathbf{E}[N_{\mathcal{P}}]$.

Lemma 3.33. $\mathrm{cost}_{1+2\varepsilon}(\mathcal{P}', X') \leq (1+\varepsilon)\, \mathrm{cost}_1(\mathcal{P}, X)$.

Proof. This follows from $\mathrm{cost}_{1+2\varepsilon}(\mathcal{P}', X') = \mathbf{E}[N_{\mathcal{P}'}] + C\, \mathbf{E}[O_{\mathcal{P}'}]$ and the previous results. □

Proof of Theorem 3.26. Let $\widehat{\mathcal{P}}$ be the policy constructed from $\mathcal{P}'$ in the following manner. We simulate policy $\mathcal{P}'$ in parallel. To pack item $\widehat{X}_i$, $\widehat{\mathcal{P}}$ imitates what $\mathcal{P}'$ does to item X'_i. Random variables $\widehat{X}_i$ and X'_i (and also X_i) are assumed to be coupled in the standard manner. The outcome of $\widehat{X}_i$ goes to $\widehat{\mathcal{P}}$ and the outcome of X'_i goes to $\mathcal{P}'$.

We denote by $\widehat{B}_{j,k}$ the bins opened by $\widehat{\mathcal{P}}$. Since $\widehat{X}_i \leq (1+\varepsilon)X'_i$,

$$\widehat{X}(B) \leq (1+\varepsilon)X'(B)$$

for any set of items B. Therefore, if $\widehat{X}(\widehat{B}_{j,k}) > 1+4\varepsilon$, then $X'(B'_{j,k}) > 1+2\varepsilon$ and so bin $B'_{j,k}$ must have been broken by $\mathcal{P}'$. Then,

$$\mathrm{cost}_{1+4\varepsilon}\left(\widehat{\mathcal{P}}, \widehat{X}\right) \leq \mathrm{cost}_{1+2\varepsilon}(\mathcal{P}', X')$$

and we obtain the desired result. □

3.6.4 From Discretized Policy to Regular Policy with Resource Augmentation

The main result of this section is the following.

Theorem 3.34. *For any policy $\widehat{\mathcal{P}}$ that sequentially packs items $\widehat{X}_1, \ldots, \widehat{X}_n$ into bins of size $1+4\varepsilon$, there exists a policy $\mathcal{P}$ that sequentially packs items $X_1, \ldots, X_n$ into bins of size $1+6\varepsilon$ such that*

$$\mathrm{cost}_{1+6\varepsilon}(\mathcal{P}, X) \leq (1+\varepsilon)\,\mathrm{cost}_{1+4\varepsilon}\left(\widehat{\mathcal{P}}, \widehat{X}\right).$$

Given a policy $\widehat{\mathcal{P}}$ for the discretized items $\widehat{X}_1, \ldots, \widehat{X}_n$, we recover a policy $\mathcal{P}$ for items $X_1, \ldots, X_n$ with an extra 2ε in the bins' capacities. We couple the variables $\widehat{X}_i$ with X'_i and X_i. Policy $\mathcal{P}$ simulates policy $\widehat{\mathcal{P}}$ in the following manner. For each bin $\widehat{B}_j$ that policy $\widehat{\mathcal{P}}$ opens, $\mathcal{P}$ opens a bin $B_{j,1}$ and packs items in $B_{j,1}$ as policy $\widehat{\mathcal{P}}$ would do in bin $\widehat{B}_j$, as long as $|X(B_{j,1}) - X'(B_{j,1})| \leq \varepsilon$ holds. (Note that the comparison is between random variables X and X'.) If this difference is violated, policy $\mathcal{P}$ opens a new bin $B_{j,2}$, and continues following $\widehat{\mathcal{P}}$ as long as $|X(B_{j,2}) - X'(B_{j,2})| \leq \varepsilon$ holds, and so on.

As before, we need to show that policy P is well defined, in the sense that bin $B_{j,k}$ is not broken if $\widehat{B}_j$ has not been broken.

Proposition 3.35. *If $\mathcal{P}$ overflows bin $B_{j,k}$ for some k, then $\widehat{\mathcal{P}}$ must have overflowed bin $\widehat{B}_j$. Moreover, at most one of the bins $B_{j,1}, B_{j,2}, \ldots$ can be overflowed.*

Proof. Let $B_{j,k,t}$ be the items packed into bin $B_{j,k}$ by policy $\mathcal{P}$ up to time t. Suppose that at time t policy $\mathcal{P}$ breaks bin $B_{j,k}$; then $X(B_{j,k,t}) > 1 + 6\varepsilon$. Now,

$$\begin{aligned}
\widehat{X}(\widehat{B}_j) &\geq \widehat{X}(B_{j,k,t}) \\
&\geq X'(B_{j,k,t}) \\
&= X'(B_{j,k,t-1}) + X'_t \\
&\geq (X(B_{j,k,t-1}) - \varepsilon) + \left(X_t - \varepsilon^4\right) \\
&> 1 + 5\varepsilon - \varepsilon^4 \\
&> 1 + 4\varepsilon.
\end{aligned}$$

Therefore, $\widehat{\mathcal{P}}$ must have overflowed bin $\widehat{B}_j$. □

As a consequence we have the following result.

Proposition 3.36. $\mathbf{E}\left[O_{\mathcal{P}}\right] \leq \mathbf{E}\left[O_{\widehat{\mathcal{P}}}\right]$.

For $k \geq 1$, consider the family of events $\mathcal{E}_{j,k} = \{|X(B_{j,k}) - X'(B_{j,k})| > \varepsilon\}$ and for $k = 0$ define $\mathcal{E}_{j,0} = \{\mathcal{P} \text{ opens bin } B_{j,1}\} = \{\widehat{\mathcal{P}} \text{ opens bin } \widehat{B}_j\}$. Then,

Proposition 3.37. *For any $k \geq 1$,*

$$\mathbf{P}(\mathcal{E}_{j,k} \mid \mathcal{E}_{j,k-1}, \ldots, \mathcal{E}_{j,1}, \mathcal{E}_{j,0}) \leq (6\varepsilon^2)\, \mathbf{P}(\mathcal{P} \textit{ opens } B_{j,k} \mid \mathcal{E}_{j,k-1}, \ldots, \mathcal{E}_{j,1}, \mathcal{E}_{j,0}).$$

Proof. The proof is identical to Proposition 3.29. □

Let $N_{\mathcal{P},j}$ be the number of bins $B_{j,1}, B_{j,2}, \ldots$ that policy $\mathcal{P}$ opens. Then, $N_{\mathcal{P}} = \sum_{j=1}^{n} N_{\mathcal{P},j}$ is the number of bins used by policy $\mathcal{P}$. Following the proof strategy used for Proposition 3.32, we obtain the following proposition.

Proposition 3.38. $\mathbf{E}\left[N_{\mathcal{P},j}\right] \leq (1+\varepsilon)\,\mathbf{P}(\widehat{\mathcal{P}}$ *opens* $\widehat{B}_j)$.

Corollary 3.38.1. $\mathbf{E}[N_{\mathcal{P}}] \leq (1+\varepsilon)\,\mathbf{E}[N_{\widehat{\mathcal{P}}}]$.

Proof of Theorem 3.34. The proof is direct from the previous results. □

3.6.5 Computing an Optimal Discretized Policy via Dynamic Programming

We can write a dynamic program (DP) that computes $\min_{\widehat{P}} \mathrm{cost}_{1+4\varepsilon}\left(\widehat{\mathcal{P}}, \widehat{X}\right)$, solved by backward induction in $\mathcal{O}\left(n^{2/\varepsilon^5}/\varepsilon^{10}\right)$ time. The states are pairs (t, S), where $t = 1, \ldots, n+1$ and $S = (k_0, k_1, \ldots, k_r)$ is a vector of non-negative integers such that $k_0 + k_1 + \cdots + k_r \leq t-1$. Here k_j represents the number of bins currently at capacity $j \cdot \varepsilon^5$, $j = 1, \ldots, \lceil 2/\varepsilon^5 \rceil$. The number of states (t, S) is at most $\mathcal{O}(n^{2/\varepsilon^5})$. Then, the DP recursion becomes

$$v(t,S) = \min\Bigg\{1 + \mathop{\mathbf{E}}_{\widehat{X}_t}\left[v(t+1, S + e_{\widehat{X}_t/\varepsilon^5})\right],$$
$$C\,\mathbf{P}(k_j + \widehat{X}_t > 1+4\varepsilon) + \mathop{\mathbf{E}}_{\widehat{X}_t}\left[v(t+1, S + e_{j+\widehat{X}_t/\varepsilon^5} - e_j)\right] : 0 < j\varepsilon^5 \leq 1+4\varepsilon, k_j \geq 1\Bigg\},$$

with the boundary condition $v(n+1, S) = 0$ for any S. Here, e_j is the canonical vector in $\mathbf{R}^{r+1}$ with a 1 in the j-th coordinate and 0 elsewhere. The recursion for $v(t, S)$ includes the two possible choices for a decision maker: Pack the item into a new bin and incur a cost of 1 or use one of the previously opened and available bins.

Given access to $v(t+1, S')$ for any valid S', we can compute $v(t, S)$ in $\mathcal{O}(1/\varepsilon^{10})$ time, since we need to compute the corresponding expectations in time $\mathcal{O}(1/\varepsilon^5)$. There are $\mathcal{O}(1/\varepsilon^5)$ of these terms inside the minimum operator, so we can compute $v(t+1, S')$ in $\mathcal{O}(1/\varepsilon^{10})$

time. Finally, given that there are $\mathcal{O}(n^{2/\varepsilon^5})$ states, we obtain the stated running time.

3.7 Numerical Experiments

In this section, we empirically validate the Budgeted Greedy (BG) algorithm. We quantify an algorithm's performance via the ratio of its cost to the cost incurred by some reference algorithm. When computationally possible, the reference algorithm is the optimal offline sequential policy. Otherwise, the reference is BG itself. We compare BG against the online benchmarks Full Greedy (FG), Fixed-Threshold (FT) and Fixed-Threshold Greedy (FTG).

- **Full Greedy (FG)** is the myopic policy that for each item i compares the instantaneous cost of opening a new bin (unit cost) and the expected cost of packing the item in one of the previously opened bins, $C\,\mathbf{P}(\text{overflow})$. The policy selects the cheapest option.

- **Fixed-Threshold (FT(α))** is the policy that has a threshold $\alpha \in (0, 1]$, and packs items into a bin as long as its usage does not exceed α. Note that this policy uses one bin at a time.

- **Fixed-Threshold-Greedy (TG(α))** combines the myopic policy FG with a capacity threshold α. The policy behaves as FG, but bins with usage greater than α are discarded. Note that FG corresponds to TG(1).

We test BG on four kinds of instances, one i.i.d. sequence of random variables, and three arbitrary exponential random variable input sequences. In all the instances we set the penalty to $C = 50$ and input length to $n = 10^5$. We simulate each instance $1,000$ times and report the sample mean.

- **I.I.D. Sequence**. In this experiment, we consider an i.i.d. input sequence with three-point

support given by

$$X_i = \begin{cases} 0 & \text{w.p. } 1 - 1/C \\ 0.4 & \text{w.p. } 1/2C \\ 0.61 & \text{w.p. } 1/2C. \end{cases}$$

With this input, we aim to compare BG against TG(α) with threshold $\alpha \geq 0.4$. For $\alpha < 0.4$, TG(α) is near-optimal, therefore we do not study this case because we already include the optimal offline policy as a reference. Furthermore, it suffices to consider the case $\alpha = 0.4$, since TG(α) for $\alpha \in (0.4, 1)$ is exactly the same. For TG(1), we recover FG. In addition, we test different values of γ for BG, denoted BG(γ). We test $\gamma = 1, 2$ and the value $\gamma = \sqrt{2}$ which optimizes the γ in the proof of Theorem 3.2. In this experiment we do not test FT, since it behaves exactly as TG for thresholds $\alpha < 1$, and FT(1) has an expected cost of at least n.

- **Exponential Distributions**. We consider input random variables $X_1, \ldots, X_n$ that follow exponential distributions, $\mathbf{P}(X_i > x) = e^{-\lambda_i x}$. We perform three different experiments:

  1. First, we consider an input sequence of exponential random variables with increasing rates. The smallest rate starts at $\lambda_1 = \log C$ and the largest rate is $\lambda_n = 3 \log C$. In general, we set $\lambda_i = \left(1 + 2\frac{i-1}{n-1}\right) \log C$ for $i = 1, \ldots, n$.

  2. Second, we consider an input sequence with decreasing rates. The largest rate is $\lambda_1 = 3 \log C$ and the smallest rate is $\lambda_n = \log C$. In this case, we have $\lambda_i = \left(3 - 2\frac{i-1}{n-1}\right) \log C$ for $i = 1, \ldots, n$.

  3. Finally, we consider an input sequence divided into three sections, each section with an i.i.d. sequence. The first section, for $i = 1, \ldots, \lfloor n/3 \rfloor$, considers the fixed rate $\lambda_i = \log C$. The second section, for $i = \lfloor n/3 \rfloor + 1, \ldots, \lfloor 2n/3 \rfloor$, considers the fixed rate $\lambda_i = 2 \log C$. The final section, for $i = \lfloor 2n/3 \rfloor + 1, \ldots, n$ considers the fixed rate

$$\lambda_i = \log C.$$

Theorem 3.3 guarantees that BG has a constant multiplicative factor loss if the rates are $2 \log C$ or greater. In these experiments, we empirically test the expected cost incurred by BG when the rates are in $[\log C, 3 \log C]$, where Theorem 3.3 can only guarantee a multiplicative loss of $\mathcal{O}(\log C)$. Moreover, this guarantee theoretically applies to large C (see Section 3.5); here we test the algorithm on the relatively small penalty $C = 50$.

3.7.1 Results

I.I.D. Random Variables Figure 3.2 presents the ratio of the sample mean of the cost incurred by the algorithms and the sample mean of the optimal offline sequential cost; this latter quantity is roughly n/C. We empirically confirm that the expected cost of TG policies is at least $n/8$; BG(γ) with $\gamma = 1, \sqrt{2}, 2$ exhibits better performance. As n grows, BG(1) has a ratio of roughly 1.8, BG(2) one of roughly 2.75, and BG$(\sqrt{2})$ a ratio of roughly 1.9; theoretically we can guarantee a ratio of $\sqrt{3} + 2\sqrt{2} \approx 4.5604$.

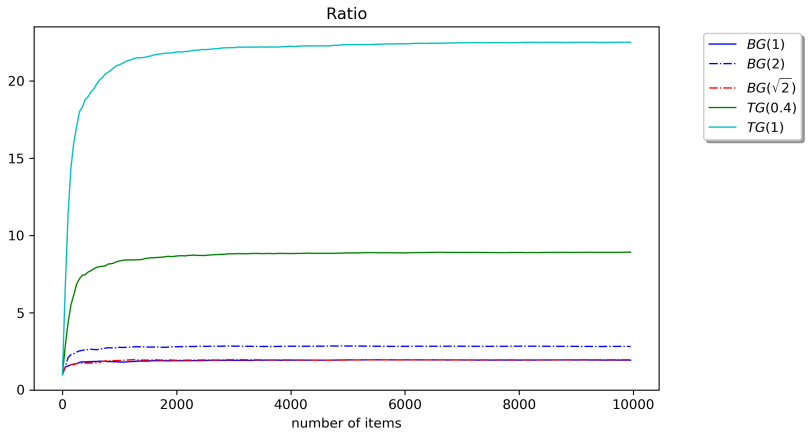

Figure 3.2: Ratio to optimal expected cost incurred by the algorithms BG, FG and TG. Note that BG$(\sqrt{2})$ overlaps with BG(1); the difference is roughly 0.1 units.

Exponential Random Variables Figures 3.3, 3.4 and 3.5 present the empirical results of our experiments in the case of increasing rates, decreasing rates and block-input rates,

respectively. In these experiments, we used BG(2) as a reference, because computing the offline benchmark was too computationally expensive. We used $\gamma = 2$ because it has a $\mathcal{O}(\log C)$ approximation guarantee compared to the optimal offline expected cost (see Proposition 3.15.) Smaller values of γ do not improve the performance of BG in a significant manner; as the results show, being greedy seems suited to exponential distributions. On the other hand, larger values of γ make BG's performance resemble FG.

Figure 3.3 displays the ratio of cost sample means between the benchmark algorithms and BG(2) for increasing rates. We empirically observe that BG performs significantly better against all FT policies. Similarly, BG performs better that most of TG policies, with the exception of TG(0.5) and TG(1). Until approximately the $5,000$-th item, the ratio TG(1)/BG(2) is the best among all greedy strategies, and afterwards the ratio TG(0.5)/BG(2) becomes the best. Moreover, by the end of the sequence, FT(0.5) becomes better than TG(1). During the whole input sequence, we empirically observe that BG(2) is able to balance the behavior of TG(0.5) and TG(1), surpassing the performance of TG(1) in the second half of the input sequence. For the entire sequence, the best performing algorithm's expected cost ratio is above 0.8, which means BG(2) is within 25% of the best algorithm for all input sizes. Furthermore, around the $5,000$-th item BG performs the best among all tested strategies.

Figure 3.4 displays the ratios of the tested algorithms and BG(2) for the decreasing rates experiment. In this case, most of the TG/BG-curves and FT/BG-curves overlap, with the exception of TG(1)/BG and FT(1)/BG. For almost the entire sequence, all plots lie above 0.4, indicating that BG's expected cost is at most 2.5 times the best performing algorithm's cost for all input sizes. BG's performance decreases until around the $5,000$-th item and improves thereafter. As in the previous experiment, BG performs better for larger rates, which coincides with our theoretical findings.

Figure 3.5 displays the ratios between the tested algorithms and BG(2) for the partitioned

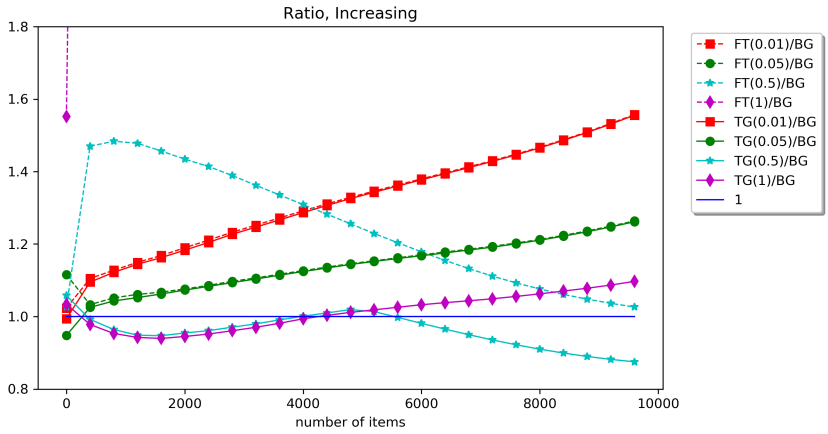

Figure 3.3: Ratio of cost incurred in the exponential case for increasing rates.

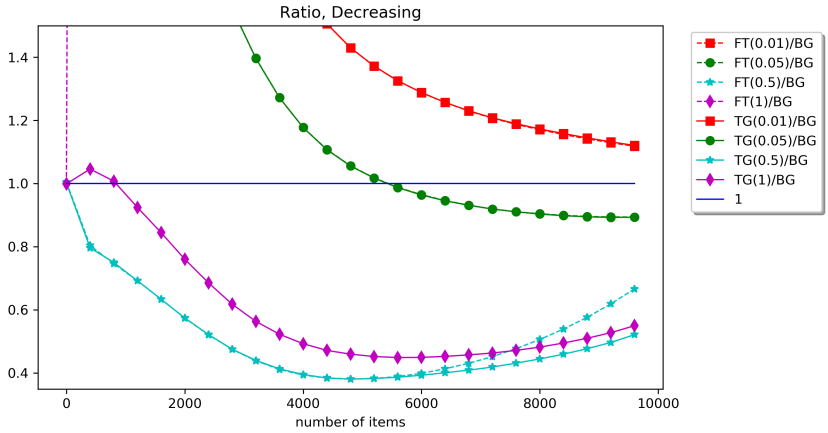

Figure 3.4: Ratio of cost incurred in the exponential case for decreasing rates.

input sequence. The best performing algorithm over the entire sequence is TG(1); this algorithm's plot and all others lie above 0.6, indicating BG is within 67% of the best performing algorithm for any input size. During the first interval of the sequence, the ratios are roughly constant; the main differentiation occurs with the transition to the second interval, where TG(1) and TG(0.5) outperform BG. In the last interval, BG's performance again improves.

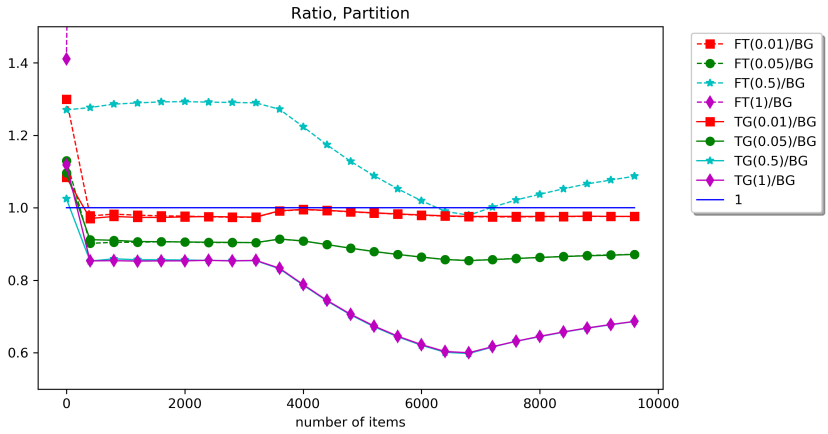

Figure 3.5: Ratio of cost incurred in the exponential case block input.

3.8 Concluding Remarks

In this chapter, we introduced the adaptive bin packing problem with overflow. We introduced the notion of risk as a proxy for a capacity threshold, as typically used in deterministic settings. We showed that Budgeted Greedy incurs an expected cost at most a constant factor times the optimal expected cost of an offline policy when the input is an i.i.d. sequence of random variables. In the more general setting, we give similar results for arbitrary exponential random variables.

We extended the discussion by studying the offline sequential adaptive bin packing problem, in which the decision maker knows the sequence of random variables in advance and must pack them in this order. We devised a soft-capacity PTAS by utilizing a policy tracking argument, and showed that computing the cost of the optimal policy is $\#\mathbf{P}$-hard by relating it to counting problems. This offline cost corresponds to the online benchmark.

Unfortunately, Budgeted Greedy does not guarantee a constant approximation factor for general input sequences. Consider the input sequence $X_1, X_2, X_3, \ldots, X_n$ defined as $X_1 =$

$1/n$, $X_{2i} \sim \text{Bernoulli}(1/C)$ and

$$X_{2i+1} = \begin{cases} 1/n & \text{w.p. } 1 - 1/C^2 \\ 1 & \text{w.p. } 1/C^2. \end{cases}$$

Budgeted Greedy incurs an expected cost of $\Theta(n)$, while the optimal offline policy incurs an expected cost of at most $n/C + 1$.

This example motivates either seeking a general algorithm exhibiting a bounded competitive ratio, or showing an impossibility result. In [12], the authors study the online generalized assignment problem with a similar stochastic component as in our model. They are able to show a $1 - 1/\sqrt{k}$ competitive ratio for general arriving distributions. However, they assume large capacity, in the sense that no item takes up more than $1/k$ fraction from any bin. It is not clear how to utilize their techniques in a bin packing setting, as they are able to discard distributions that they deem unimportant. Moreover, in the bin packing problem a large capacity assumption would immediately imply a policy with constant approximation factor, by simply filling up the bins until some desired fraction of capacity.

3.A Appendix: Missing Proofs

3.A.1 Missing Proofs From Section 3.3

Proof of Proposition 3.6. For simplicity, we write $\mathbf{1}_{\{i \to j\}}$ to denote $\mathbf{1}^{\mathcal{P}}_{\{i \to j\}}$. First, we show that $\mathbf{P}(\mathcal{P} \text{ breaks bin } j) = \mathbf{E}\left[\sum_{i=1}^{n} \mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1)\mathbf{1}_{\{i \to j\}}\right]$. We have,

$$\mathbf{E}\left[\sum_{i=1}^{n} \mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1)\mathbf{1}_{\{i \to j\}}\right] = \sum_{i=1}^{n} \mathbf{E}\left[\mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1)\mathbf{1}_{\{i \to j\}}\right]$$

Observe that $S_j^{i-1} = \sum_{k \le i-1} X_k \mathbf{1}_{\{k \to j\}}$ and $\mathbf{1}_{\{i \to j\}}$ only depend on the outcomes of the r.v.'s $X_1, \ldots, X_{i-1}$. Therefore,

$$\begin{aligned}
\mathbf{E}\left[\mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1)\mathbf{1}_{\{i \to j\}}\right] &= \mathop{\mathbf{E}}_{X_1,\ldots,X_{i-1}}\left[\mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1)\mathbf{1}_{\{i \to j\}}\right] \\
&= \mathop{\mathbf{E}}_{X_1,\ldots,X_{i-1}}\left[\mathop{\mathbf{E}}_{X_i}\left[\mathbf{1}_{\{X_i + S_j^{i-1} > 1\}}\right]\mathbf{1}_{\{i \to j\}}\right] \\
&= \mathop{\mathbf{E}}_{X_1,\ldots,X_i}\left[\mathbf{1}_{\{X_i + S_j^{i-1} > 1\}}\mathbf{1}_{\{i \to j\}}\right].
\end{aligned}$$

Clearly, $\{X_i + S_j^{i-1} > 1, i \to j\} = \{X_i \text{ breaks bin } j\}$. Thus,

$$\mathbf{E}\left[\mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1)\mathbf{1}_{\{i \to j\}}\right] = \mathbf{P}(X_i \text{ breaks bin } j),$$

hence,

$$\mathbf{E}\left[\sum_{i=1}^{n} \mathbf{P}_{X_i}(X_i + S_j^{i-1} > 1)\mathbf{1}_{\{i \to j\}}\right] = \sum_{i=1}^{n} \mathbf{P}(X_i \text{ breaks bin } j) = \mathbf{P}(\mathcal{P} \text{ breaks bin } j)$$

since the last sum uses the fact that bins are overflowed at most once; hence, the events $\{X_i \text{ breaks bin } j\}_i$ are disjoint. □

Proof of Proposition 3.7. The proof follows from a result in [59], which we replicate here for completeness. Let $\mu_i = \mathbf{E}[X_i \wedge 1]$ be the normalized expected size of an item. Let B^t be the (random) items that the policy packs into bin B by time t. We are interested in the expectation of $\mu(B) = \sum_{i \in B} \mu_i = \mu(B^n)$. The random variables $\mu(B^t)$ are nondecreasing in t; by the monotone convergence theorem,

$$\mathbf{E}\left[\mu(B)\right] = \sup_{t \ge 0} \mathbf{E}[\mu(B^t)].$$

Now, the random variables $Z^t = \sum_{i \in B^t} (X_i \wedge 1) - \mu_i$ form a martingale. Indeed,

$$\mathbf{E}[Z^t \mid Z^{t-1}, t \to j] = Z^{t-1} + \mathbf{E}[X_t \wedge 1] - \mu_t = Z^{t-1};$$

then, for all t, $\mathbf{E}[Z^t] = Z^0 = 0$ and so $\mathbf{E}[\mu(B^t)] = \mathbf{E}\left[\sum_{i \in B^t} X_i \wedge 1\right] \leq 2\,\mathbf{P}(\mathcal{P} \text{ opens bin } B)$; this last inequality holds because we break the bin at most once and we must have opened the bin. Therefore

$$\mathbf{E}\left[\mu(B)\right] = \sup_{t \geq 0} \mathbf{E}[\mu(B^t)] \leq 2\,\mathbf{P}(\mathcal{P} \text{ opens bin } B).$$

□

Proof of Proposition 3.8. Note that by Proposition 3.7 we have

$$\mathbf{E}\left[\sum_{i=1}^{n}(X_i \wedge 1)\right] = \sum_{j=1}^{n} \mathbf{E}\left[\sum_{i \in B_j^{\mathcal{P}}}(X_i \wedge 1)\right] = \sum_{j=1}^{n} \mathbf{E}\left[\sum_{i \in B_j^{\mathcal{P}}} \mathbf{E}[X_i \wedge 1]\right] = \mathbf{E}\left[\sum_{i=1}^{n} \mathbf{E}[X_i \wedge 1]\right].$$

We only need to show that $\sum_{i=1}^{n} \mathbf{E}[X_i \wedge 1]$ is a lower bound for $\mathrm{cost}(\textsc{Opt})$. We can compute $\mathrm{cost}(\textsc{Opt})$ recursively via dynamic programming as follows. We define the states as vectors $S \in (\mathbf{R} \cup \{\emptyset\})^n$ where $S_j \in \mathbf{R}$ is the usage of j-th bin and $S_j = \emptyset$ means that bin j is closed. We consider $\emptyset$ as an special symbol such that $a + \emptyset = a$ for any $a \in \mathbf{R}$. With this, the optimal cost can be computed via the following recursions:

$$v_t(S) = \inf\left\{\mathop{\mathbf{E}}_{X_t}\left[v_{t+1}(S + X_t e_j)\right] : j = 1, \ldots, n, S_j \in [0,1] \cup \{\emptyset\}\right\}, \quad \forall t = 1, \ldots, n, \forall S$$
$$v_{n+1}(S) = \sum_{j=1}^{n} \mathbf{1}_{\{S_j \neq \emptyset\}} + C \sum_{j=1}^{n} \mathbf{1}_{\{S_j > 1\}}, \quad \forall S.$$

The second equation measures the overall cost accumulated at the end of processing the sequence $X_1, \ldots, X_n$. The first equation takes actions that minimizes the mean cost of sample paths. Note that items can only be packed into bins not opened ($\emptyset$) or bins with

usage ≤ 1. Using MDP theory, we can show $v_1(\emptyset, \ldots, \emptyset) = \mathrm{cost}(\textsc{Opt})$, which we skip here for brevity.

Now, consider the functions $u_t(S) = \sum_{\tau=t}^{n} \mathbf{E}[X_\tau \wedge 1] + \sum_{j=1}^{n} (S_j \wedge 1)\mathbf{1}_{\{S_j \neq \emptyset\}}$ for any S. We show by backward induction in $t = n+1, \ldots, 1$ that $u_t(S) \leq v_t(S)$. For $t = n+1$ we have

$$u_{n+1}(S) = \sum_{j=1}^{n} (S_j \wedge 1)\mathbf{1}_{\{S_j \neq \emptyset\}} \leq \sum_{j=1}^{n} \mathbf{1}_{\{S_j \neq \emptyset\}} \leq v_{n+1}(S).$$

Now, assume the result is true for $t+1$ and let us show it for t. Let $j = 1, \ldots, n$ with $S_j \in [0,1] \cup \{\emptyset\}$, then

$$\begin{aligned}
\mathop{\mathbf{E}}_{X_t}\left[v_{t+1}(S + X_t e_j)\right] &\geq \mathop{\mathbf{E}}_{X_t}\left[\sum_{\tau=t+1}^{n} \mathbf{E}[X_\tau \wedge 1] + \sum_{\substack{k=1 \\ k \neq j}}^{n} (S_k \wedge 1)\mathbf{1}_{\{S_j \neq \emptyset\}} + (S_j + X_t) \wedge 1\right] \\
&\geq \sum_{\tau=t+1}^{n} \mathbf{E}[X_\tau \wedge 1] + \mathop{\mathbf{E}}_{X_t}[X_t \wedge 1] = u_t(S).
\end{aligned}$$

Taking minimum in j, we conclude $v_t(S) \geq u_t(S)$ for any S.

Now, for $t = 1$ we have $\mathrm{cost}(\textsc{Opt}) = v_1(\emptyset, \ldots, \emptyset) \geq u_1(\emptyset, \ldots, \emptyset) = \sum_{t=1}^{n} \mathbf{E}[X_t \wedge 1]$ which finishes the proof. □

3.A.2 Missing Proofs From Section 3.4

Proof of Lemma 3.11. We define

$$\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(u))_j = 1 + \mathbf{E}\left[(C + 2\delta)\sum_{i=1}^{n} \mathbf{1}^{\mathcal{P}}_{\{i \to j\}} \mathbf{1}^{\mathcal{P}}_{\{X_i + S_j^{i-1} > 1\}} \mid \text{Reach node } u\right]$$

which is the original cost paid in $\mathcal{T}_{\mathcal{P}}$ when packing items into bin j after reaching node u in the tree. We also define

$$\begin{aligned}\mathrm{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(u))_j =& 1 + \mathbf{E}\left[(C+\delta)\sum_{i=1}^{n}\mathbf{1}^{\mathcal{P}'}_{\{i\to j\}}\mathbf{1}^{\mathcal{P}'}_{\{X_i+S_j^{i-1}>1\}}\right.\\ &\left.+(C+2\delta)\left(\sum_{i=1}^{n}\mathbf{1}^{\mathcal{P}'}_{\{i\to j'\}}\mathbf{1}^{\mathcal{P}'}_{\{X_i+S_{j'}^{i-1}>1\}}\right)+\mathbf{1}^{\mathcal{P}'}_{\{\text{Open bin } j'\}} \mid \text{Reach node } u\right]\end{aligned}$$

which is the new cost paid by $\mathcal{T}_{\mathcal{P}'}$ when packing items into bin j after reaching node u and the new cost incurred by packing items into bin j'.

Therefore, the variation of the cost $\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(u)) - \mathrm{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(u))$ is given by

$$\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(u)) - \mathrm{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(u)) = (\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(u))_j - \mathrm{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(u))_j).$$

Now, we always have

$$\mathbf{1}^{\mathcal{P}}_{\{i\to j\}} = \mathbf{1}^{\mathcal{P}'}_{\{i\to j\}} + \mathbf{1}^{\mathcal{P}'}_{\{i\to j'\}},$$

for all $i = 1,\dots,n$. Indeed, if we are in a branch not containing u, then $\mathcal{P}$ and $\mathcal{P}'$ behave the same and there is no bin j'. If we are in a branch containing u, and if we pack i into j, we either pack i into j before surpassing the risk budget in which case $\mathcal{P}$ and $\mathcal{P}'$ behave the same or we do it after surpassing the risk budget in which case i goes to j'. With this fact we have,

$$\begin{aligned}\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(u))) - \mathrm{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(u)) &= (\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(u))_j - \mathrm{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(u))_j)\\ &\geq \mathbf{E}\left[\delta\left(\sum_{i=1}^{n}\mathbf{1}^{\mathcal{P}'}_{\{i\to j\}}\mathbf{1}^{\mathcal{P}'}_{\{X_i+S_j^{i-1}>1\}}\right)-\mathbf{1}^{\mathcal{P}'}_{\{\text{Open bin } j'\}} \mid \text{Reach node } u\right],\end{aligned}$$

in the last inequality we used the fact that the cost of breaking the bin j' is smaller than the cost of breaking j at that point of the computation. This is true since the usage of bin j' is

at most the usage of j at the same point of computation. Now, for $i \geq j$,

$$\begin{aligned}
&\mathbf{E}\left[\mathbf{1}^{\mathcal{P}'}_{\{i\to j\}}\mathbf{1}_{\{X_i+S_j^{i-1}>1\}} \mid \text{Reach node } u\right] \\
&= \mathop{\mathbf{E}}_{X_1,\ldots,X_{i-1}}\left[\mathop{\mathbf{E}}_{X_i}\left[\mathbf{1}^{\mathcal{P}'}_{\{i\to j\}}\mathbf{1}^{\mathcal{P}'}_{\{X_i+S_j^{i-1}>1\}}\right] \mid \text{Reach node } u\right] \\
&= \mathop{\mathbf{E}}_{X_1,\ldots,X_{i-1}}\left[\mathbf{1}^{\mathcal{P}'}_{\{i\to j\}}\mathop{\mathbf{E}}_{X_i}\left[\mathbf{1}_{\{X_i+S_j^{i-1}>1\}}\right] \mid \text{Reach node } u\right] \\
&= \mathop{\mathbf{E}}_{X_1,\ldots,X_{i-1}}\left[\mathbf{1}^{\mathcal{P}'}_{\{i\to j\}}\mathbf{P}_{X_i}(X_i+S_j^{i-1}>1) \mid \text{Reach node } u\right].
\end{aligned}$$

This is because the event $\{\text{Reach node } u\}$ is determined by the outcomes of $X_1,\ldots,X_{j-1}$. While for $i \leq j-1$ we have

$$\mathbf{E}\left[\mathbf{1}^{\mathcal{P}'}_{\{i\to j\}}\mathbf{1}_{\{X_i+S_j^{i-1}>1\}} \mid \text{Reach node } u\right] = 0$$

since bin j is opened at node u at level j. Thus,

$$\begin{aligned}
&\mathrm{cost}_{\ell,\widehat{c}}(\mathcal{T}_{\mathcal{P}}(u)) - \mathrm{cost}_{\ell',c'}(\mathcal{T}_{\mathcal{P}'}(u)) \\
&= \mathbf{E}\left[\delta\sum_{i=j}^{n}\mathbf{1}^{\mathcal{P}'}_{\{i\to j\}}\mathbf{P}(X_i+S_j^{i-1}>1) - \mathbf{1}_{\{\text{Open bin } j'\}} \mid \text{Reach node } u\right] \\
&= \mathbf{E}\left[\mathrm{Risk}(B_j) - \mathbf{1}^{\mathcal{P}'}_{\{\text{Open bin } j'\}} \mid \text{Reach node } u\right] \\
&\geq \left(\delta\frac{\gamma}{C}-1\right) = 0. && \text{(Using } \gamma = \tfrac{C}{\delta}\text{.)}
\end{aligned}$$

□

3.A.3 Missing Proofs From Section 3.5

Proof of Proposition 3.23. We show that (w.h.p.) Algorithm 5 packs each item X_i^{λ} individually. This is achieved by showing that in between two X_i^{λ} and X_{i+1}^{λ}, there is enough mass introduced by the elements $X_{i,j}^{\mu}$, therefore not allowing the items X_i^{λ} to be packed together.

Now, let $k = 3\varepsilon\mu = 3\varepsilon\beta \log C$. Then,

$$\mathbf{E}\left[\sum_{i=1}^{k} X_i^{\mu}\right] = \frac{k}{\beta \log C} = 3\varepsilon,$$

thus

$$\mathbf{P}\left(\sum_{i=1}^{k} X_i^{\mu} \leq 2\varepsilon\right) \leq \mathbf{P}\left(\left|\sum_{i=1}^{k} X_i^{\mu} - 3\varepsilon\right| \geq \varepsilon\right) \leq \frac{1}{\varepsilon^2}\frac{k}{(\beta \log C)^2} = \frac{3}{\varepsilon\beta \log C}.$$

(Chebyshev inequality)

Pick $\beta = 6(n_1 \log C)/\varepsilon$ and so,

$$\mathbf{P}\left(\exists j = 1, \ldots, n_1 : \sum_{i=1}^{k} X_{j,i}^{\mu} \leq 2\varepsilon\right) \leq n_1 \cdot \frac{3}{\varepsilon\beta \log C} = \frac{1}{2}.$$

That is, with probability at least $1/2$, all blocks $X_{j,1}^{\mu}, \ldots, X_{j,k}^{\mu}$ add at least 2ε mass. Consider the event

$$E = \left\{\forall j = 1, \ldots, n_1 : \sum_{i=1}^{k} X_{j,i}^{\mu} > 2\varepsilon\right\}.$$

then, we just proved that $\mathbf{P}(E) \geq 1/2$.

Claim 3.39. *Given event E, Algorithm 5 with $\gamma = 1$ never packs X_i^{λ} and X_{i+1}^{λ} together for any i.*

Proof. Suppose that Algorithm 5 packs X_i^{λ} and X_{i+1}^{λ} together for some i. This means that the algorithm had enough budget and space to allocate X_{i+1}^{λ}. Since $\mathbf{P}(X_{i+1}^{\lambda} > 1 - x) = e^{-\lambda(1-x)} > e^{-\mu(1-x)} = \mathbf{P}(X_{ij}^{\mu} > 1 - x)$ for any $x > 0$, then Budgeted Greedy must have packed all the X_{ij}^{μ} in between X_i^{λ} and X_{i+1}^{λ}. However, under event E, these items increase the usage of the bin by at least 2ε. Then, the budget utilized by X_{i+1}^{λ} is at least

$$\mathbf{P}(X_{i+1}^{\lambda} > 1 - 2\varepsilon) = e^{-\lambda(1-2\varepsilon)} > e^{-(1-\varepsilon^2)\log C} > \frac{1}{C}$$

which is a contradiction to the risk budget of Budgeted Greedy. □

Claim 3.40. *Using the same choices of β and k as before, we have* $\mathrm{cost}(\textsc{Alg}) \geq n_1/2$.

Proof. By the previous result, under event E, no X_i^λ and X_{i+1}^λ are packed together. Therefore, at least n_1 open bins are needed. Since E occurs w.p. $\geq 1/2$ we conclude the desired result. □

□

Proof of Proposition 3.24. In order to show an upper bound for $\mathrm{cost}(\textsc{Opt})$ it is enough to exhibit a policy with cost bounded by the desired value. We consider the following budgeted policy $\mathcal{P}$ with risk budget $2/C$: Pack items with rate λ separately of items with rate μ. We are going to show that $\mathcal{P}$ opens at most

$$\frac{n_2}{\beta \log C} + \frac{n_1}{\varepsilon \log C} = n_1 \left(\frac{k}{\beta \log C} + \frac{1}{\varepsilon \log C} \right)$$

bins in expectation (up to a constant). Since $\mathcal{P}$ is budgeted with budget $2/C$ we have $\mathrm{cost}(\textsc{Opt}) \leq \mathrm{cost}(\mathcal{P}) \leq 3\,\mathbf{E}[N_{\mathcal{P}}]$ from which the result follows. In what follows we prove the bound over the number of bins.

Let us analyze the policy $\mathcal{P}$. Policy $\mathcal{P}$ opens two kind of bins; the first kind of bins only contain items following exponentials distribution of rate λ; the second kind of bins only contain items following exponential distribution of rate μ. We have $N_{\mathcal{P}} = N_{\mathcal{P}}^1 + N_{\mathcal{P}}^2$ where $N_{\mathcal{P}}^1$ is the number of bins of type 1 and $N_{\mathcal{P}}^2$ is the number of bins of type 2. An equivalent way to see this process is that policy $\mathcal{P}$ runs two copies of Algorithm 5, one for the rate λ and one for the rate μ. Then, $N_{\mathcal{P}}^1$ equals N_{ALG} over the sequence $X_1^\lambda, \ldots, X_{n_1}^\lambda$ and $N_{\mathcal{P}}^2$ equals N_{ALG} over the sequence $X_{1,1}^\mu, \ldots, X_{n_1,k}^\mu$.

The following lemma is a general result that allows us to bound the number of bins used in a nonnegative i.i.d. sequence of items under Algorithm 5. For the sake of clarity, the proof has been moved to the end of this subsection.

Lemma 3.41. *Suppose the r.v.'s $X_1, \ldots, X_n$ form an i.i.d. sequence of items, then $\mathbf{E}[N_{\mathrm{ALG}}] \leq (2n-1)/\mathbf{E}[|B_1|]$ where $|B_1|$ is the number of items packed in the first bin.*

Claim 3.42. *Let $X_1, \ldots, X_n$ be n independent exponential r.v.'s with rate $\lambda = (1+\varepsilon)\log C$, with $\varepsilon \log C \geq 4$, then*

$$\mathbf{E}[|B_1|] \geq \frac{1}{8}\varepsilon \log C,$$

where $|B_1|$ is the number of items $X_1, \ldots, X_n$ packed in the first bin by Algorithm 5 with risk budget $= 2/C$.

Proof. Let $\ell = \frac{\varepsilon}{4}\log C \geq 1$. Then,

$$\begin{aligned}\mathbf{P}(|B_1| \leq \ell) &= \mathbf{P}\left(|B_1| \leq \ell, X(B_1) > \frac{\varepsilon}{2(1+\varepsilon)}\right) + \mathbf{P}\left(|B_1| \leq \ell, X(B_1) \leq \frac{\varepsilon}{2(1+\varepsilon)}\right) \\ &\leq \mathbf{P}\left(\sum_{i=1}^{\ell} X_i > \frac{\varepsilon}{2(1+\varepsilon)}\right) + \mathbf{P}\left(|B_1| \leq \ell, X(B_1) \leq \frac{\varepsilon}{2(1+\varepsilon)}\right).\end{aligned}$$

We bound each term separately. First, we have

$$\mathbf{P}\left(\sum_{i=1}^{\ell} X_i > \frac{\varepsilon}{2(1+\varepsilon)}\right) \leq \frac{2(1+\varepsilon)}{\varepsilon}\,\mathbf{E}\left[\sum_{i=1}^{\ell} X_i\right] = \frac{2(1+\varepsilon)}{\varepsilon}\ell\frac{1}{(1+\varepsilon)\log C} = \frac{1}{2}.$$

For the other term we have that $|B_1| \leq \ell$, given $X(B_1) \leq \frac{\varepsilon}{2(1+\varepsilon)}$, only if B_1 runs out of budget. That is,

$$\frac{2}{C} \leq \sum_{i=1}^{\ell+1} \mathbf{P}(X_i > 1-\alpha_i) \leq (\ell+1)\,\mathbf{P}\left(X_i > 1 - \frac{\varepsilon}{2(1+\varepsilon)}\right) \leq \left(\frac{\varepsilon}{4}\log C + 1\right)\frac{1}{C^{1+\varepsilon/2}} < \frac{2}{C}$$

using the assumption $\varepsilon \log C \geq 2$. From here we obtain that $\mathbf{P}(|B_1| \leq \ell, X(B_1) \leq$

$\frac{\varepsilon}{2(1+\varepsilon)}) = 0$. Therefore,

$$\mathbf{P}(|B_1| \leq \ell) \leq \frac{1}{2}.$$

Then,

$$\mathbf{E}[|B_1|] \geq \frac{1}{2}\ell = \frac{1}{8}\varepsilon \log C.$$

□

Claim 3.43. *Let $X_1, \ldots, X_m$ be m independent exponential r.v.'s with rate $\mu = \beta \log C$, $\beta \geq 4$, then*

$$\mathbf{E}[|B_1|] \geq \frac{1}{8}\beta \log C$$

where $|B_1|$ is the number of items $X_1, \ldots, X_m$ packed in the first in by Algorithm 5 with risk budget $= 2/C$.

Proof. Let $\ell = \frac{\beta}{4} \log C$. Then,

$$\begin{aligned}\mathbf{P}\left(|B_1| \leq \ell\right) &= \mathbf{P}(|B_1| \leq \ell, X(B_1) > 1/2) + \mathbf{P}(|B_1| \leq \ell, X(B_1) \leq 1/2) \\ &\leq \mathbf{P}\left(\sum_{i=1}^{\ell} X_i > \frac{1}{2}\right) + \mathbf{P}\left(|B_1| \leq \ell, X(B_1) \leq 1/2\right)\end{aligned}$$

Now, given the event $X(B_1) \leq 1/2$, the only way that $|B_1| \leq \ell$ is by running out of budget. We have then

$$\frac{2}{C} \leq \sum_{i=1}^{\ell+1} \mathbf{P}(X_i > 1 - \alpha_i) \leq (\ell+1)\,\mathbf{P}(X_i > 1/2) \leq \left(\frac{\beta}{4}\log C + 1\right)\frac{1}{C^{\beta/2}} < \frac{1}{C} \qquad (\beta \geq 4)$$

which cannot happen. Therefore, $\mathbf{P}(|B_1| \leq \ell, X(B_1) \leq 1/2) = 0$ and then

$$\mathbf{P}(|B_1| \leq \ell) \leq 2\,\mathbf{E}\left[\sum_{i=1}^{\ell} X_i\right] = 2\ell\frac{1}{\beta \log C} = \frac{1}{2}.$$

Therefore,

$$\mathbf{E}[|B_1|] \geq \frac{1}{2}\ell = \frac{\beta}{8}\log C.$$

□

□

Claim 3.44. *The cost of $\mathcal{P}$ is* $\operatorname{cost}(\mathcal{P}) \leq 3\,\mathbf{E}[N_{\mathcal{P}}] \leq 48n_1\left(k/(\beta\log C) + 1/(\varepsilon\log C)\right)$

Proof. Putting all the results together we obtain

$$\begin{aligned}
\mathbf{E}[N_{\mathcal{P}}] &= \mathbf{E}[N^1_{\mathcal{P}}] + \mathbf{E}[N^2_{\mathcal{P}}] \\
&\leq \frac{2n_1}{\mathbf{E}[|B^\lambda_1|]} + \frac{2n_2}{\mathbf{E}[|B^\mu_1|]} && \text{(Proposition 3.41)} \\
&\leq 16\frac{n_1}{\varepsilon\log C} + 16\frac{n_2}{\beta\log C} \\
&= 16n_1\left(\frac{1}{\varepsilon\log C} + \frac{k}{\beta\log C}\right).
\end{aligned}$$

□

Here we present the proof of Lemma 3.41.

Proof of Lemma 3.41. We use the following fictitious experiment. Consider n independent copies of the random variables $X_1, \ldots, X_n$ and run Algorithm 5 until its first bin is closed or the sequence fits entirely on the first bin. We denote by $\widetilde{B}_i$ the items packed in the first bin in the i-th trial of this experiment. The process $|\widetilde{B}_1|, \ldots, |\widetilde{B}_n|$ is i.i.d..

We have the following identities:

$$\begin{aligned}
|B_1| &= |\widetilde{B}_1| \\
|B_2| &= \min\{n - |B_1|, \widetilde{B}_2\} \\
&\vdots \\
|B_n| &= \min\{n - |B_1| - \cdots - |B_{n-1}|, \widetilde{B}_n\}.
\end{aligned}$$

Observe that $N_{\text{ALG}} = \min\left\{k : \sum_{i=1}^{k} |B_i| = n\right\}$ is a stopping time for $|B_1|, \ldots, |B_n|$ so also is a stopping time for $|\widetilde{B}_1|, \ldots, |\widetilde{B}_n|$. By Wald's equation (see Theorem 3.45 below) we have

$$\mathbf{E}[N_{\text{ALG}}]\,\mathbf{E}[|\widetilde{B}_1|] = \mathbf{E}\left[\sum_{i=1}^{N_{\text{ALG}}} |\widetilde{B}_i|\right].$$

Additionally, we have $\mathbf{E}[|\widetilde{B}_1|] = \mathbf{E}[|B_1|]$ by construction. Now, until time $N_{\text{ALG}} - 1$ we must have $|B_1| = |\widetilde{B}_1|, \ldots, |B_{N_{\text{ALG}}-1}| = |\widetilde{B}_{N_{\text{ALG}}-1}|$, all of these values at least 1. Then,

$$\sum_{i=1}^{N_{\text{ALG}}-1} |\widetilde{B}_i| = \sum_{i=1}^{N_{\text{ALG}}-1} |B_i| \leq n - 1.$$

Therefore,

$$\mathbf{E}[N_{\text{ALG}}]\,\mathbf{E}[|B_1|] = \mathbf{E}\left[\sum_{i=1}^{N_{\text{ALG}}-1} |B_i| + |\widetilde{B}_{N_{\text{ALG}}}|\right] \leq 2n - 1,$$

which concludes the proof. □

Wald's Equation

Theorem 3.45 (Wald's equation)**.** *If $X_1, X_2, \ldots$ are i.i.d. random variables with finite mean and N is a stopping time with $\mathbf{E}[N] < \infty$, then*

$$\mathbf{E}\left[\sum_{n=1}^{N} X_n\right] = \mathbf{E}[N]\,\mathbf{E}[X_1].$$

Proof can be found in [150].

3.B Appendix: #P-Hardness of Computing Minimum Cost of the Optimal Policy

In this section, we provide the proof of Theorem 3.5, i.e., it is $\#\mathbf{P}$-hard to compute $\min_{\mathcal{P}} \operatorname{cost}(\mathcal{P})$. We proceed as follows. We consider *symmetric* logic formulas, that is, $\phi(\mathbf{x}) = \phi(\overline{\mathbf{x}})$ for any $\mathbf{x}$, and we show that the problem $\#$SYM-4SAT—the problem of counting satisfying assignment of symmetric formulas in 4CNF—is $\#\mathbf{P}$-hard. Recall that a formula is in *conjunctive normal form* (CNF) if it is a conjunction of one or more clauses. When the clauses have k literals, we say that the formula is in kCNF.

Then, we provide a polynomial time reduction from symmetric formulas ϕ in 4CNF into instances of the stochastic bin packing problem such that

$$\min_{\mathcal{P}} \operatorname{cost}(\mathcal{P}) = \frac{5}{2} - \frac{2}{2^n} - \frac{s_\phi}{2^n},$$

where s_ϕ denotes the satisfying assignments of ϕ, i.e., $s_\phi = |\{\mathbf{x} = (x_1, \ldots, x_n) : \phi(\mathbf{x}) = 1\}|$.

We now proceed to show the hardness of $\#$SYM-4SAT.

Theorem 3.46. $\#$SYM-4SAT *is* $\#\mathbf{P}$*-hard.*

Proof. We show a reduction from the $\#\mathbf{P}$-hard problem $\#$2SAT [167]. We symmetrize a formula ϕ by extending the assignments $\mathbf{x} = (x_1, \ldots, x_n)$ in one variable, namely x_0. Let $\phi = \bigwedge_{j=1}^{m} C_j$ be a formula in 2CNF, i.e., each clause has the form $C_j = (\ell_{1,j} \vee \ell_{2,j})$ where $\ell_{i,j} \in \{x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n\}$ for $i = 1, 2$. Consider the formula

$$\widetilde{\phi}(x_0, \mathbf{x}) = (\overline{x}_0 \wedge \phi(\mathbf{x})) \vee (x_0 \wedge \phi(\overline{\mathbf{x}})).$$

Note that $\widetilde{\phi}$ is symmetric but not yet in CNF. Also, note that $s_{\widetilde{\phi}} = |\{(x_0, \mathbf{x}) : \widetilde{\phi}(x_0, \mathbf{x}) = 1\}| = 2s_\phi$. It remains to show that we can transform $\widetilde{\phi}$ into a symmetric 4CNF without altering the number of solutions. Using De Morgan's law, we rewrite the formula $\widetilde{\phi}$ as

$$\begin{aligned}
\widetilde{\phi}(x_0, \mathbf{x}) &= (x_0 \vee \phi(\mathbf{x})) \wedge (\overline{x}_0 \vee \phi(\overline{\mathbf{x}})) \wedge (\phi(\mathbf{x}) \vee \phi(\overline{\mathbf{x}})) \\
&= \left(x_0 \vee \bigwedge_{j=1}^{m} C_j(\mathbf{x})\right) \wedge \left(\overline{x}_0 \vee \bigwedge_{j=1}^{m} C_j(\overline{\mathbf{x}})\right) \wedge \left(\bigwedge_{j=1}^{m} C_j(\mathbf{x}) \vee \bigwedge_{j=1}^{m} C_j(\overline{\mathbf{x}})\right) \\
&= \underbrace{\left(\bigwedge_{j=1}^{m} (x_0 \vee C_j(\mathbf{x}))\right)}_{\text{(I)}} \wedge \underbrace{\left(\bigwedge_{j=1}^{m} (\overline{x}_0 \vee C_j(\overline{\mathbf{x}}))\right)}_{\text{(II)}} \wedge \underbrace{\left(\bigwedge_{j,k=1}^{m} C_j(\mathbf{x}) \vee C_k(\overline{\mathbf{x}})\right)}_{\text{(III)}}.
\end{aligned}$$

The first two terms (I) and (II) are clearly 3CNF. We extend them into 4CNF by repeating the variable x_0 or $\overline{x}_0$ accordingly. The last term (III) is already in 4CNF. We remove clauses that contain pairs $\ell \vee \overline{\ell}$ since they are trivially satisfied. □

Note that in the formula $\widetilde{\phi}$, each variable appears at most twice in each clause, either as $x \vee x$ or $\overline{x} \vee \overline{x}$. This is going to be utilized in the next proof.

Before going to the proof, and for the sake of explanation, we change the capacity of the bins to a capacity $B > 1$, to be defined later. This can be easily adjusted to our setting with capacity 1 by scaling down items sizes by the amount B. As we are going to see, it is clearer to introduce items > 1 than their fractional rescaled version.

3.B.1 Reduction #SYM-4SAT to Stochastic Bin Packing

The reduction is similar to the reduction from PARTITION-PROBLEM to BIN-PACKING. See [160] for an example. A similar reduction is used in [58] in the context of multidimensional stochastic knapsack. Given a symmetric 4CNF ϕ with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$, we are going to construct a list of nonnegative random variables, that will correspond to the input of the stochastic bin packing problem, that can be packed into

two bins whenever the outcomes of these random variable satisfy the 4CNF formula while in the opposite case, the number of bins requires is at least three.

We are going to utilize numbers with at most $4n + m$ digits in base 10. For convenience we assume that all number have exactly $4n + m$ digits by filling the unused corresponding significant digits with 0. Given a number with $4n + m$ digits in base 10, we split its representation into four blocks. The first block corresponds to the digits it positions $10^i \cdot 10^n \cdot 10^{2n} \cdot 10^m$ for $i = 0, \ldots, n-1$. We refer to the first block as the *variable block* and their intra significant digits as the *variables digits*. For instance, by variable digit x_i we refer to the digit in location $10^{n-i} \cdot 10^n \cdot 10^{2n} \cdot 10^m$.

The second block corresponds to digits at positions $10^i \cdot 10^{2n} \cdot 10^m$ for $i = 0, \ldots, n-1$. We refer to the second block as the *mirror variable block* and its intra digits as *mirror variable digits*. By *mirror digit* x_i we refer to the digit in location $10^{n-i} \cdot 10^{2n} \cdot 10^m$.

The third block corresponds to digits in positions $10^k \cdot 10^m$ for $k = 0, \ldots, 2n-1$. We refer to this block as the *equivalence block* and we split its digits into pairs of digits that we refer as *equivalence digits*. The positive equivalent digit x_i refers to the digit in position $10^{2n-2i+1} \cdot 10^m$ while the negative equivalence digit x_i refers to the digit in position $10^{2n-2i} \cdot 10^m$.

The last block corresponds to the digits at position 10^j for $j = 0, \ldots, m-1$. We refer to the this block as the *clauses block* and its intra digits as *clauses digits*. By clause digit C_j we refer to the digit located at position 10^{m-j}.

We set the capacity of the bins to be the number in base 10

$$B = \underbrace{11\cdots11}_{n\text{ 1's}} \mid \underbrace{11\cdots11}_{n\text{ 1's}} \mid \underbrace{11\cdots11}_{2n\text{ 1's}} \mid \underbrace{44\cdots44}_{m\text{ 4's}}.$$

We purposely separated the significant digits using the vertical bars " | " into the four afore-

mentioned blocks.

For a formula ϕ in 4CNF we now present the reduction. For each variable $i = 1, \ldots, n$ we construct the following four numbers a_i, b_i, c_i and d_i in base 10. The first two number are

$$a_i = 1\underbrace{0\cdots0}_{n-i} \mid \underbrace{00\cdots00}_{n} \mid \underbrace{00\cdots00}_{2i-2}01\underbrace{00\cdots00}_{2n-2i} \mid \underbrace{00\cdots00}_{m}$$

$$b_i = 1\underbrace{0\cdots0}_{n-i} \mid \underbrace{00\cdots00}_{n} \mid \underbrace{00\cdots00}_{2i-2}10\underbrace{00\cdots00}_{2n-2i} \mid \underbrace{00\cdots00}_{m}.$$

Number a_i and b_i have $4n - i + m + 1$ digits. Both of them have a common digit 1 in variable digit x_i. Moreover, a_i has another digit 1 in negative equivalent digit x_i; and b_i has a digit 1 in positive equivalence digit x_i.

The following two numbers are

$$c_i = \underbrace{\cdots}_{\text{no digits}} \mid \underbrace{\cdots}_{\text{no digits}} 1\underbrace{0\cdots0}_{n-i} \mid \underbrace{00\cdots00}_{2i-2}10\underbrace{00\cdots00}_{2n-2i} \mid \underbrace{00\cdots0c_i^{k_i}00\cdots00}_{m}$$

$$d_i = \underbrace{\cdots}_{\text{no digits}} \mid \underbrace{\cdots}_{\text{no digits}} 1\underbrace{0\cdots0}_{n-i} \mid \underbrace{00\cdots00}_{2i-2}01\underbrace{00\cdots00}_{2n-2i} \mid \underbrace{00\cdots000d_i^{k_i}\cdots00}_{m}.$$

Number c_i and d_i have $3n - i + m + 1$ digits, and note that we keep the separation between the blocks to emphasize where the nonzero digits appear. In other words, the variable block is completely missing from c_i and d_i. Now, both number c_i and d_i have a common digit 1 in mirror variable digit x_i (mirror numbers $x_{i'}$ with $i' < i$ are also missing). The number c_i has digit $c_i^{k_i} \in \{1, 2\}$ in all clauses digits C_{k_i} where literal x_i appears; the number d_i has digit $d_i^{k_i} \in \{1, 2\}$ in all clauses digits where literal $\overline{x}_i$ appears. The number c_i has a digit 1 in positive equivalence digit x_i; and d_i has a digit 1 in negative equivalence digit x_i.

For each clause C_j, $j = 1, \ldots, m$, we introduce three numbers f_j, g_j and h_j that are going

to serve as slacks:

$$f_j = g_j = h_j = \underbrace{\cdots}_{\text{no digits}} \mid \underbrace{\cdots}_{\text{no digits}} \mid \underbrace{\cdots}_{\text{no digits}} \mid \underbrace{\cdots}_{\text{no digits}} 1 \underbrace{0 \cdots 0}_{m-j}.$$

The three numbers have a unique digit 1 at clause digit C_j and they completely miss the variable, mirror variable and equivalence blocks.

Finally, we introduce a number needed for technical reasons:

$$h = \underbrace{\cdots}_{\text{no digits}} \mid \underbrace{\cdots}_{\text{no digits}} \mid \underbrace{11 \cdots 11}_{m} \mid \underbrace{00 \cdots 00}_{2n}.$$

A pictorial construction of the numbers appears in Figure 3.6.

Given these number, we construct an instance of the stochastic bin packing problem as follows. We define the following independent random variables

$$X_i = \begin{cases} a_i & \text{w.p. } 1/2 \\ b_i & \text{w.p. } 1/2 \end{cases} \quad \text{and} \quad X'_i = \begin{cases} a_i & \text{w.p. } 1/2 \\ b_i & \text{w.p. } 1/2 \end{cases}.$$

Now the instance is given by the sequence

$$\mathcal{L}_\phi = (X_1, X'_1, X_2, X'_2, X_3, X'_3, \ldots, X_n, X'_n, c_1, d_1, c_2, d_2, \ldots, c_n, d_n, h_1, g_1, \ldots, h_m, g_m, h).$$

Intuitively, we aim to simulate the random evaluation of ϕ when the values of $x_1, \ldots, x_n$ are chosen uniformly and independently of each other. Note that in this case,

$$\mathbf{P}_{\mathbf{x} \in_R \{0,1\}^n}(\phi(\mathbf{x}) = 1) = \frac{s_\phi}{2^n}.$$

Note that in instance $\mathcal{L}_\phi$, any policy incurs in a cost of at least 2 since X_1 and X'_1 if packed together incur in an expected cost of at least 2. This is because $C > 2$. As we did in

	x_1	x_2	x_3	$\cdots$	x_n	y_1	y_2	y_3	$\cdots$	y_n	x_1^+	x_1^-	x_2^+	x_2^-	$\cdots$	x_n^+	x_n^-	C_1	C_2	C_3	$\cdots$	C_m
a_1	1	0	0	$\cdots$	0	0	0	0	$\cdots$	0	0	1	0	0	$\cdots$	0	0	0	0	0	$\cdots$	0
b_1	1	0	0	$\cdots$	0	0	0	0	$\cdots$	0	1	0	0	0	$\cdots$	0	0	0	0	0	$\cdots$	0
a_2		1	0	$\cdots$	0	0	0	0	$\cdots$	0	0	0	0	1	$\cdots$	0	0	0	0	0	$\cdots$	0
b_2		1	0	$\cdots$	0	0	0	0	$\cdots$	0	0	0	1	0	$\cdots$	0	0	0	0	0	$\cdots$	0
a_3			1	$\cdots$	0	0	0	0	$\cdots$	0	0	0	0	0	$\cdots$	0	0	0	0	0	$\cdots$	0
b_3			1	$\cdots$	0	0	0	0	$\cdots$	0	0	0	0	0	$\cdots$	0	0	0	0	0	$\cdots$	0
$\vdots$				$\ddots$					$\vdots$						$\vdots$						$\vdots$	
a_n					1	0	0	0	$\cdots$	0	0	0	0	0	$\cdots$	0	1	0	0	0	$\cdots$	0
b_n					1	0	0	0	$\cdots$	0	0	0	0	0	$\cdots$	1	0	0	0	0	$\cdots$	0
c_1						1	0	0	$\cdots$	0	1	0	0	0	$\cdots$	0	0	2	0	0	$\cdots$	0
d_1						1	0	0	$\cdots$	0	0	1	0	0	$\cdots$	0	0	0	1	1	$\cdots$	0
c_2							1	0	$\cdots$	0	0	0	1	0	$\cdots$	0	0	1	1	0	$\cdots$	0
d_2							1	0	$\cdots$	0	0	0	0	1	$\cdots$	0	0	0	0	0	$\cdots$	0
c_3								1	$\cdots$	0	0	0	0	0	$\cdots$	0	0	0	0	1	$\cdots$	0
d_3								1	$\cdots$	0	0	0	0	0	$\cdots$	0	0	1	2	0	$\cdots$	0
$\vdots$									$\ddots$						$\vdots$						$\vdots$	
c_n										1	0	0	0	0	$\cdots$	1	0	0	0	1	$\cdots$	0
d_n										1	0	0	0	0	$\cdots$	0	1	0	0	0	$\cdots$	2
f_1																		1	0	0	$\cdots$	0
g_1																		1	0	0	$\cdots$	0
h_1																		1	0	0	$\cdots$	0
f_2																			1	0	$\cdots$	0
g_2																			1	0	$\cdots$	0
h_2																			1	0	$\cdots$	0
$\vdots$																					$\ddots$	
f_m																						1
g_m																						1
h_m																						1
h																		1	1	1	$\cdots$	1
B	1	1	1	$\cdots$	1	1	1	1	$\cdots$	1	1	1	1	1	$\cdots$	1	1	4	4	4	$\cdots$	4

Figure 3.6: Construction of numbers via ϕ 4CNF. The table is purposely divided into four blocks representing the digit blocks defined at the beginning of the subsection. The instance showed corresponds partially to the 4CNF $\phi(x_1, \ldots, x_n) = (x_1 \vee x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_3 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_3 \vee x_4 \vee x_n) \wedge \cdots \wedge (x_{n-2} \vee x_{n-1} \vee \overline{x}_n \vee \overline{x}_n)$, where clauses are $C_1, C_2, C_3, \ldots, C_m$ in the order they are displayed.

the main body of the article, we can assume that the policies are deterministic. Moreover, we can assume that policies never break a bin since the expected cost of breaking a bin is always greater than 1. This is because the probability of overflow always is either $0, 1/2$ or 1 by construction of $\mathcal{L}_\phi$.

Theorem 3.47. *For the instance $\mathcal{L}_\phi$ we have*

$$\min_{\mathcal{P}} \operatorname{cost}(\mathcal{P}) = \frac{5}{2} - \frac{2}{2^n} - \frac{1}{2^n} \mathbf{P}_{\mathbf{x} \in \{0,1\}^n}(\phi(\mathbf{x}) = 1).$$

Proposition 3.48. *For the instance $\mathcal{L}_\phi$ there is a policy with $\mathrm{cost}(\mathcal{P}) \leq 3$. Moreover, for any of such policies, there is a policy with same cost or better that packs items $X_1, \ldots, X_n$ into bin 1 and items $X'_1, \ldots, X'_n$ into bin 2.*

Proof. Consider the policy that packs item X_1 into bin 1; item X'_1 into bin 2; the rest of the items into bin 3. This policy is valid since

$$\sum_{i=2}^{n}(X_i + X'_i) + \sum_{i=1}^{n}(c_i + d_i) + \sum_{j=1}^{m}(h_j + g_j) + h \leq B.$$

This proves the first part of the proposition.

For the second part, consider any policy $\mathcal{P}$ with cost at most 3 that does not break any bin. We can assume, without loss of generality, that X_1 is packed into bin 1 and X'_1 is packed into bin 2. Now, starting at the root of the policy tree, find the first node u where X_i is not packed in bin 1, say bin $j \geq 2$. Note that up to that point, items X'_k must have been packed in a different bin than bin 1. If both children of u are packed into bin 1, that is, X'_i is packed into bin 1, then exchange the packing rule in node u by packing X_i into bin 1 and in its children to pack item X'_i into bin j. This does not change the cost since X_i and X'_i are identically distributed. Suppose now that some of the children of node u packs item X'_i into bin $j' \neq 1$, say children v. At this point, there is enough space in bin 1 to receive X'_i since X_i was packed into bin $j \geq 2$. In the subtree rooted at v, mark all nodes that pack their corresponding item into bin 1. Exchange the packing rule from these node to pack their items into bin j' and change the policy to pack item X'_i in node v from bin j' to bin 1. This does not increase the cost of the policy. With this, we can modify the policy to pack X'_i in both children of node u into bin 1 without increasing the expected cost. Now, like in the previous case, we can pack item X_i into bin 1 and item X'_i into bin $j \geq 2$. We can repeat this procedure for all i and at the end of this, we can ensure that all $X_1, \ldots, X_n$ are packed into bin 1 and the cost of the policy does not increase. With a similar argument,

we show that items $X'_1, \ldots, X'_n$ can be packed into bin 2 without increasing the cost of the policy. From the root, find the first node u' where X'_i is not packed into bin 2, say bin j. Note that j cannot be 1 since we already have that X_i has been packed into bin 1 and both random variables share the same variable digit x_i. Then, $j \geq 3$. In the subtree rooted at u' mark all items that are packed into bin 2. Repack those items into bin j and pack X'_i into bin 2. This does not increase the cost of the policy since we have enough space in bin j to receive any item in there if needed. Repeating this procedures for all i gives us the desired result. □

Consider the following three events: Let

$$\mathcal{E} = \{\forall i = 1, \ldots, n : X_i \neq X'_i\},$$

then $\mathbf{P}(\mathcal{E}) = 1/2^n$. Let

$$\mathcal{E}_a = \{\exists i = 1, \ldots, n : \forall k < i, X_k \neq X'_k, X_i = X'_i = a_i\}$$

and

$$\mathcal{E}_b = \{\exists i = 1, \ldots, n : \forall k < i, X_k \neq X'_k, X_i = X'_i = b_i\}.$$

Note that $\mathcal{E}_a$ and $\mathcal{E}_b$ are disjoint and $\mathcal{E}_a \cup \mathcal{E}_b = \overline{\mathcal{E}}$. Intuitively, $\mathcal{E}$ is the good event where the variables X_i and X'_i model opposite values in $\{a_i, b_i\}$. Also note that $\mathbf{P}(\mathcal{E}_a) = \mathbf{P}(\mathcal{E}_b) = (1 - 1/2^n)/2$.

We denote by $\mathrm{cost}(\mathcal{P} \mid \mathcal{A})$ the conditional expected cost of the policy $\mathcal{P}$ on the event $\mathcal{A}$.

Proposition 3.49. *For any policy $\mathcal{P}$ we have* $\mathrm{cost}(\mathcal{P} \mid \mathcal{E}_b) \geq 3$.

Proof. Note that given $\mathcal{E}_b$ we have (assuming the collision occurs at i, $X_i = X'_i = b_i$) we

have

$$\sum_{i=1}^{n}(X_i + X_i' + c_i + d_i) + \sum_{j=1}^{m}(h_j + g_j) + h$$

$$\geq 22\cdots22 \mid 22\cdots22 \mid \underbrace{22\cdots22}_{2i-2}311\cdots11 \mid 88\cdots88 > 2B$$

where B is the bin's capacity. Therefore, since we are assuming that $\mathcal{P}$ does not break bins, then the policy must have packed all the items into at least 3 bins which concludes the proof. □

Proposition 3.50. *For any policy $\mathcal{P}$ that packs items $X_1, \ldots, X_n$ into bin 1 and items $X_1', \ldots, X_n'$ into bin 2 and does not break any bin, we have* $\text{cost}(\mathcal{P} \mid \mathcal{E}) \geq 3 - \mathbf{P}_{\mathbf{x}\in_R\{0,1\}^n}(\phi(\mathbf{x}) = 1)$.

Proof. Consider the following random assignment $\mathbf{X} = (x_1, \ldots, x_n)$:

$$x_i = \begin{cases} 1 & \text{if } X_i = a_i \\ 0 & \text{if } X_i = b_i \end{cases}.$$

Note that $\mathbf{X}$ is uniformly distributed over $\{0,1\}^n$. Then, $\text{cost}(\mathcal{P} \mid \mathcal{E}, \phi(\mathbf{X}) = 0) \geq 3$. Indeed, if only 2 bins have been used after all items have been packed, this forces item c_i to be placed in bin 1 if $X_i = a_i$ and in bin 2 otherwise, while d_i is packed in the opposite bin to c_i. Since $\phi(\mathbf{X}) = 0 = \phi(\overline{\mathbf{X}})$ by symmetry of ϕ, then, after packing items $c_1, d_1, \ldots, c_n, d_n$ but before packing items $f_1, g_1, h_1, \ldots, f_m, g_m, h_m$, there must be a C_j digit in the utilization of bin 1 that is 0 and a $C_{j'}$ digit in the utilization of bin 2 that is also 0, $j \neq j'$. In particular, this implies that the usage of bin 2 at this point has a C_j-digit of 4. After packing items $f_1, g_1, h_1, \ldots, f_m, g_m, h_m$, one of the bins must have a 4 in its C_1-digit of usage, say bin 1. Therefore, bin 2 has a usage with C_1-digit 3 and item h cannot be packed into bin 1. Since $\mathcal{E}$ is given, we have $\sum_{i=1}^{n}(X_i + X_i') + \sum_{i=1}^{n}(c_i + d_i) + \sum_{j=1}^{m}(f_j +$

$g_j + h_j) + h = 2B$, then h can be packed into bin 2 only if all $C_1, \ldots, C_m$-digits are 3. However, this contradicts the fact that C_j-digit in bin 2 is 4. Similarly, if the bin 2 has usage with C_1-digit of 4 after packing items $f_1, g_1, h_1, \ldots, f_m, g_m, h_m$, we can obtain the same contradiction. Therefore, $\mathrm{cost}(\mathcal{P} \mid \mathcal{E}, \phi(\mathbf{X}) = 0) \geq 3$.

Now,

$$\begin{aligned}\mathrm{cost}(\mathcal{P} \mid \mathcal{E}) &\geq 3\,\mathbf{P}(\phi(\mathbf{X}) = 0 \mid \mathcal{E}) + 2\,\mathbf{P}(\phi(\mathbf{X}) = 1 \mid \mathcal{E}) = 3 - \mathbf{P}(\phi(\mathbf{X}) = 1 \mid \mathcal{E}) \\ &= 3 - \mathbf{P}_{\mathbf{x} \in_R \{0,1\}^n}(\phi(\mathbf{x}) = 1).\end{aligned}$$

□

Lemma 3.51. *For any policy $\mathcal{P}$,* $\mathrm{cost}(\mathcal{P}) \geq 5/2 - 1/2^{n-1} - \mathbf{P}_{\mathbf{x} \in_R \{0,1\}^n}(\phi(\mathbf{x}) = 1)/2^n$.

Proof. We can assume that the policy $\mathcal{P}$ packs items $X_1, \ldots, X_n$ into bin 1 while items $X'_1, \ldots, X'_n$ into bin 2 (see Proposition 3.48). Then, utilizing Propositions 3.49 and 3.50 we obtain

$$\begin{aligned}\mathrm{cost}(\mathcal{P}) &\geq \left(3 - \mathbf{P}_{\mathbf{x} \in_R \{0,1\}^n}(\phi(\mathbf{x}) = 1)\right) \mathbf{P}(\mathcal{E}) + 3\,\mathbf{P}(\mathcal{E}_b) + 2\,\mathbf{P}(\mathcal{E}_a) \\ &= \frac{5}{2}\left(1 - \frac{1}{2^n}\right) + \frac{1}{2^n}\left(3 - \mathbf{P}_{\mathbf{x} \in_R \{0,1\}^n}(\phi(\mathbf{x}) = 1)\right) \\ &= \frac{5}{2} - \frac{2}{2^n} - \frac{1}{2^n}\,\mathbf{P}_{\mathbf{x} \in_R \{0,1\}^n}(\phi(\mathbf{x}) = 1).\end{aligned}$$

□

Lemma 3.52. *There is a policy $\mathcal{P}$ such that* $\mathrm{cost}(\mathcal{P}) \leq 5/2 - 1/2^{n-1} - \mathbf{P}_{\mathbf{x} \in_R \{0,1\}^n}(\phi(\mathbf{x}) = 1)/2^n$.

Proof. Consider the following policy. Start packing items $X_1, \ldots, X_n$ into bin 1 and items $X'_1, \ldots, X'_n$ into bin 2. If there is a collision $X_i = X'_i$ and the first of these is $X_i = X'_i = b_i$,

then pack the rest of the items into bin 3; While if the first of these collisions is $X_i = X'_i = a_i$, then continue packing as follows. Pack $c_{i'}$ where $a_{i'}$ is packed and $d_{i'}$ where $b_{i'}$ is packed for $i' < i$. Pack c_i into bin 1 and d_i into bin 2. Pack the remaining $c_{i'}$ into bin 1 and $d_{i'}$ into bin 2. Since i is the first time there is a collision, bin 2 has a 0 in its positive equivalent digit x_i and a 2 in its negative equivalent digit x_i. Therefore it has enough space to receive the items packed after $d_{i'}$ has been packed. Then only 2 bins are utilized.

If no collision happens, then pack c_i where outcome a_i is packed (bin 1 if $X_i = a_i$ or bin 2 if $X'_i = a_i$) and pack d_i in the opposite bin (bin 2 if c_i is in bin 1 bin 1 otherwise). Now, utilize the slack items f_j, g_j, h_j to complete bin 1 and then bin 2. Now, for h there are two cases based on the value of ϕ on the satisfying assignment $\mathbf{x}$ given by

$$x_i = \begin{cases} 1 & \text{if } X_i = a_i \\ 0 & \text{if } X_i = b_i \end{cases}.$$

- If $\phi(\mathbf{x}) = 1$, then in bin 2, each C_j digit of the capacity used is at most 3. This is because the C_j digit of the capacity used bin 1 is 4 by construction. Now, pack h into bin 2 and finish the packing into 2 bins.

- If $\phi(\mathbf{x}) = 0$, we can retrace the proof of Proposition 3.50 to show that in this case, h cannot fit nor in bin 1 nor in 2, therefore forcing a bin 3.

Putting all these case together, we obtain

$$\begin{aligned} \operatorname{cost}(\mathcal{P}) &= \operatorname{cost}(\mathcal{P} \mid \mathcal{E})\, \mathbf{P}(\mathcal{E}) + \operatorname{cost}(\mathcal{P} \mid \mathcal{E}_a)\, \mathbf{P}(\mathcal{E}_a) + \operatorname{cost}(\mathcal{P} \mid \mathcal{E}_b)\, \mathbf{P}(\mathcal{E}_b) \\ &= \left(3 - \mathbf{P}_{\mathbf{x} \in_R \{0,1\}^n}(\phi(\mathbf{x}) = 1)\right) \frac{1}{2^n} + \frac{5}{2}\left(1 - \frac{1}{2^n}\right) \\ &= \frac{5}{2} - \frac{2}{2^n} - \frac{1}{2^n}\, \mathbf{P}_{\mathbf{x} \in \{0,1\}^n}(\phi(\mathbf{x}) = 1). \end{aligned}$$

□

Putting together Lemma 3.51 and 3.52 we obtain the proof of Theorem 3.47.

3.C Appendix: Threshold Policies for I.I.D. Random Variables with Finite Support

In this section we discuss the problem of designing a threshold algorithm that incurs in a constant factor loss whenever the input sequence is i.i.d. (with common distribution $\mathcal{D}$) with unknown time horizon n. A threshold algorithm observes the common random distribution of the incoming streams and computes a number $\alpha \in [0, 1]$ such that bins are utilized as long as their usage is at most α. If there are no such bins, then a new bin is opened upon an arrival. Note that in the i.i.d. setting, for threshold policies of this kind, at most *one* bin is kept active at a time.

We show that among all policies that keep at most one bin active at a time, threshold policies are optimal up to an additive loss of one. This is under the assumption that the common distribution $\mathcal{D}$ has finite support.

Theorem 3.53. *Let $\mathcal{D}$ be any distribution with finite support in $[0, \infty)$. There exists $\alpha \in [0, 1]$ such that the threshold policy $\mathcal{P}_\alpha$ with threshold α satisfies*

$$\mathrm{cost}(\mathcal{P}_\alpha) \leq \min_{\substack{\mathcal{P} \text{ has at most} \\ \text{one active bin}}} \mathrm{cost}(\mathcal{P}) + 1,$$

for any input sequence of i.i.d. random variables $X_1, \ldots, X_n$ with common distribution $\mathcal{D}$.

Note that policy $\mathcal{P}_\alpha$ is computed only with the information given by the distribution $\mathcal{D}$. An online algorithm that has access to $\mathcal{D}$ computes the threshold α given in Theorem 3.53 and implements the threshold policy. In the main body of the chapter we show that Budgeted Greedy keeps at most one bin active at a time in the i.i.d. setting (Lemma 3.12). Therefore, policies that keep at most one bin active at a time are within a constant factor of the optimal offline sequential cost. Using these facts, we conclude that an algorithm implementing a

threshold policy incurs an expected cost that is a constant factor of the optimum. This factor is at most $(3 + 2\sqrt{2})$, since we utilized the guarantee given by Budgeted Greedy (Theorem 3.2).

A major downside of Theorem 3.53 is that it does not give an efficient algorithm to compute the threshold α. Indeed, for the proof of Theorem 3.53, we utilize the framework of discounted reward Markov processes (see [146]), which can compute optimal stationary policies in time depending on the size of the state space. For us, the state space is the usage of the active bin, which can be exponentially large in the description of $\mathcal{D}$. Intuitively, since we aim to compute a policy that does not depend on the time horizon n and we only have one bin to use, the best we can do is to repeat the same process over and over.

3.C.1 Proof of Theorem 3.53

The proof is divided in a sequence of propositions. We briefly introduce the definitions used in infinite-time horizon discounted Markov decision processes. We later show that the optimal discounted cost induces a monotonic cost vector. From here, a threshold policy can be deduced which is later used to design a finite-time threshold policy.

We assume that the distribution $\mathcal{D}$ has finite support in $[0, 1] \cup \{1^+\}$ where the element 1^+ denotes a fixed upper bound over the values in $[0, 1]$ and any value that $\mathcal{D}$ could have taken above 1 with positive probability is mapped to 1^+. The state space, denoted $\mathbf{S}$ corresponds to all possible values ≤ 1 that the bin can take as combinations of number in the support of $\mathcal{D}$ in addition to the special state 1^+. Note that $\mathbf{S}$ is a finite set.

Fix a discount factor $\gamma \in (0, 1)$. In the infinite time-horizon discounted factor framework, a policy corresponds to a sequence of functions (or distributions if randomized) $\Pi = (\pi_1, \pi_2, \ldots)$ that dictates the behavior of the process. That is, $\pi_t : \mathbf{S} \to \{0, 1\}$ is the decision made by the policy at round t, where 0 indicates *keep using the current bin* while 1 indicates *open a new bin*, all this as a function of the state of the system. If π_t is

random, then $\pi_t : \mathbf{S} \to \Delta(\{0,1\})$, where $\Delta(\{0,1\})$ is the probability simplex over $\{0,1\}$. We define the *discounted cost* of the policy Π at time $t = 1, 2, \ldots$ by

$$V_t^{\Pi}(s) = \begin{cases} 1 + C\,\mathbf{P}(X > 1) + \gamma\,\mathbf{E}[V_{t+1}^{\Pi}(X \wedge 1^+)] & \pi_t(s) = 0 \\ C\,\mathbf{P}(X + s > 1) + \gamma\,\mathbf{E}[V_{t+1}^{\Pi}((X+s) \wedge 1^+)] & \pi_t(s) = 1 \end{cases}.$$

Let

$$c(s,a) = \begin{cases} C \cdot \mathbf{P}(X + s > 1) & a = 0 \\ 1 + C \cdot \mathbf{P}(X > 1) & a = 1 \end{cases}$$

and $T(s,0) = (X+s) \wedge 1^+$; $T(s,1) = X \wedge 1^+$. We can write $V_t^{\Pi}(s) = c(s, \pi(s)) + \gamma\,\mathbf{E}[V_{t+1}^{\Pi}(T(s,\pi(s)))]$.

If π_t are randomized then the previous values are replaced by expectations. The goal is to find $\min_{\Pi} V_1^{\Pi}(0)$. Markov Decision processes theory guarantees that this minimum is also a minimum over the history dependent randomized policies—policies that record previous outcomes. Moreover, the optimal policy for $\min_{\Pi} V_1^{\Pi}(0)$ is also the optimal policy for $\min_{\Pi} V_1^{\Pi}(s)$ for any $s \in \mathbf{S}$. The theory also guarantees that deterministic stationary policies are optimal. That is, $\min_{\Pi} V_1^{\Pi}(s) = \min_{\pi} V_1^{(\pi,\pi,\ldots)}(s)$. From now on, we only consider deterministic policies. By V_t^{π} we refer to V_t^{Π} where $\Pi = (\pi, \pi, \ldots)$. Note that $V_1^{\pi}(s) = V_2^{\pi}(s) = \cdots$ and so we can identify the temporal cost vector $\left(V_t^{\pi}(s)\right)_{\substack{s \in \mathbf{S} \\ t=1,2,\ldots}}$ by just the vector $V^{\pi} = (V^{\pi}(s))_{s \in \mathbf{S}}$. The optimal vector V^{π} satisfies the Bellman equation $V = \mathcal{T}^{\gamma} V$, where

$$\begin{aligned} \left(\mathcal{T}^{\gamma} V\right)(s) &= \min\{1 + C\,\mathbf{P}(X > 1) + \gamma\,\mathbf{E}[V(X \wedge 1^+)], C\,\mathbf{P}(X + s > 1) + \gamma\,\mathbf{E}[V((X+s) \wedge 1^+)]\} \\ &= \min_{a=0,1}\{c(s,a) + \gamma\,\mathbf{E}[V(T(s,a))]\}. \end{aligned}$$

Therefore, V^{π} is the fixed point of the Bellman operator $\mathcal{T}^{\gamma}$. For a detailed presentation of these results, see Chapter 6 in [146].

Proposition 3.54. *Consider the optimal solution V^{π} of the discounted cost problem. Then V^{π} is a monotone function of s. That is, $V^{\pi}(s') \leq V^{\pi}(s'')$ for any $0 \leq s' \leq s'' \leq 1^{+}$.*

Proof. By contradiction, suppose that for some $s' < s''$ we have $V^{\pi}(s') > V^{\pi}(s'')$. Among all such possible pairs $s' < s''$ choose the largest $s' \leq 1$, which exists because $\mathbf{S}$ is finite. Define the new function $\widehat{\pi}_1$ as $\widehat{\pi}_1(s) = \pi(s)$ if $s \neq s'$ and $\widehat{\pi}_1(s') = \pi(s'')$. For $t \geq 2$ we define $\widehat{\pi}_t = \pi$. Now consider the policy $\widehat{\Pi} = (\widehat{\pi}_1, \widehat{\pi}_2, \ldots)$. Then, using the definition of $V_1^{\widehat{\Pi}}$ we can show that $V^{\widehat{\Pi}}(s) = V^{\pi}(s)$ for $s \neq s'$. Now, let's analyze the case $s = s'$. Observe that

$$c(s', \pi(s'')) \leq c(s'', \pi(s''))$$

since $s' < s''$ and the function $c(\cdot, a)$ is nondecreasing for any fixed a. Then, we have two cases:

- If $\pi(s'') = 0$, then as s' is the largest state where monotonicity does not hold, we have

$$\begin{aligned}
V_1^{\widehat{\Pi}}(s') &= c(s', \pi_1(s')) + \gamma\, \mathbf{E}[V_2^{\widehat{\Pi}}(T(s', \pi_1(s')))] \\
&= c(s', 0) + \gamma\, \mathbf{E}[V_2^{(\pi,\dots)}((X + s') \wedge 1^+))] && (\pi_1(s') = \pi(s'') = 0) \\
&\le c(s'', 0) + \gamma\, \mathbf{E}[V^\pi((X + s') \wedge 1^+)] \\
& && (V_2^{(\pi,\dots)} = V^\pi \text{ and monotonicity of } c(\cdot, 0)) \\
&= c(s'', 0) + \gamma\, \mathbf{P}(X = 0) V^\pi(s') + \gamma\, \mathbf{E}[V((X + s') \wedge 1^+) \mid X > 0]\, \mathbf{P}(X > 0) \\
&\le c(s'', 0) + \gamma\, \mathbf{P}(X = 0) V^\pi(s') + \gamma\, \mathbf{E}[V((X + s'') \wedge 1^+) \mid X > 0]\, \mathbf{P}(X > 0) \\
& && (\text{As } s' \text{ is the largest value where monotonicity does not hold}) \\
&\le c(s'', 0) + \gamma\, \mathbf{P}(X = 0)(V^\pi(s') - V^\pi(s'')) + \gamma\, \mathbf{E}[V^\pi((X + s'') \wedge 1^+)] \\
&= V^\pi(s'') + \gamma\, \mathbf{P}(X = 0)(V^\pi(s') - V^\pi(s'')) \\
&= \gamma\, \mathbf{P}(X = 0) V^\pi(s') + (1 - \gamma\, \mathbf{P}(X = 0)) V^\pi(s'') \\
&< \gamma\, \mathbf{P}(X = 0) V^\pi(s') + (1 - \gamma\, \mathbf{P}(X = 0)) V^\pi(s') && (\text{Since } V^\pi(s') > V^\pi(s'')) \\
&= V^\pi(s').
\end{aligned}$$

- Similarly, if $\pi(s'') = 1$, then,

$$\begin{aligned}
V_1^{\widehat{\Pi}}(s') &= c(s', \pi_1(s')) + \gamma\, \mathbf{E}[V_2^{\widehat{\Pi}}(T(s', \pi_1(s')))] \\
&= c(s', 1) + \gamma\, \mathbf{E}[V^\pi(X \wedge 1^+)] \\
&\le c(s'', 1) + \gamma\, \mathbf{E}[V^\pi(X \wedge 1^+)] \\
&= V^\pi(s'') \\
&< V^\pi(s').
\end{aligned}$$

In any case, $V^{\widehat{\Pi}}(s') < V^\pi(s')$, which contradicts the optimality of π. □

The following result states that the optimal policy of the discounted cost problem is a threshold policy.

Proposition 3.55. *For the optimal V^π, there exists $\alpha \in [0,1]$ such that the stationary policy $\widetilde{\pi}(s) = 0$ if $s \leq \alpha$; $\widetilde{\pi}(s) = 1$ if $s > \alpha$, holds $\pi = \widetilde{\pi}$.*

Proof. Let $E = \{s \in \mathbf{S} : V^\pi(s) < 1 + C\,\mathbf{P}(X > 1) + \gamma\,\mathbf{E}[V^\pi(X \wedge 1^+)]\}$ be the states where the policy π decides to utilize the current bin. Note that $0 \in E$, hence $\alpha = \sup E$ is well-defined. Also, note that for $s = 1^+$ we have

$$1 + C\,\mathbf{P}(X > 1) + \gamma\,\mathbf{E}[V^\pi(X \wedge 1^+)] \leq C\,\mathbf{P}(X + 1^+ > 1) + \gamma\,\mathbf{E}[V^\pi((X + 1^+) \wedge 1^+)],$$

thus $\alpha < 1^+$. Using the monotonicity of V^π, we have $E = [0, \alpha]$.

By definition of α we have that for any $s > \alpha$, $V^\pi(s) \geq 1 + C\,\mathbf{P}(X > 1) + \gamma\,\mathbf{E}[V^\pi(X \wedge 1^+)]$. This immediately implies that for any $s > \alpha$, $V^\pi(s) = 1 + C\,\mathbf{P}(X > 1) + \gamma\,\mathbf{E}[V^\pi(X \wedge 1^+)]$. Since $V^\pi(\cdot)$ is monotone, then for any $s \leq \alpha$ we have $V^\pi(s) = C\,\mathbf{P}(X + s > 1) + \gamma\,\mathbf{E}[V^\pi((X + s) \wedge 1^+)]$. This shows that π is indeed $\widetilde{\pi}$. □

Note that in this discounted cost model we did not restrict the possible actions when the usage of the bin goes beyond 1, i.e, in state 1^+. The optimality and monotonicity of the optimal value V^π shows that the optimal policy never tries to utilize the overflowed bin again and it will always choose to open a new bin.

We now return, to our model without discounted cost. We prove Theorem 3.53. We show that, up to an additive factor of $+1$, the optimal policy that uses one bin at a time is a threshold policy. We refer to policies in our model by letters $\mathcal{P}$ while policies in the discounted model by Greek letters π and so on.

Proof of Theorem 3.53. Note that policies in our model are always defined to open a new bin at time 1. We modify this by assuming that at time 1 the bin is already given and we will charge this additional cost of $+1$ separately.

For $\gamma \in (0,1)$ we denote by π^γ the optimal threshold policy of the discounted cost problem with discount factor γ. Note that $\pi : \mathbf{S} \to \{0,1\}$ and the set of function from $\mathbf{S}$ to $\{0,1\}$ is finite. As $\gamma \to 1$, there is an optimal policy π that repeats infinitely often in the sequence $(\pi^\gamma)_\gamma$. We take a subsequence of $\gamma_k \in (0,1)$, $\gamma_k \to 1$ as $k \to \infty$, such that $\pi^k \doteq \pi^{\gamma_k} = \pi$. By the previous proposition, we can assume that π^k is a threshold policy with threshold $\alpha \in [0,1]$. Now, we recursively expand $V^\pi(s_0)$ to obtain

$$V^\pi(s_0) = \mathbf{E}\left[c(s_0,\pi(s_0)) + \gamma_k c(s_1,\pi(s_1)) + \cdots + \gamma_k^{n-1} c(s_{n-1},\pi(s_n)) + \gamma_k^n V^\pi(s_n)\right]$$

where $s_i = T(s_{i-1},\pi(s_{i-1}))$ is the i-th state obtained by the policy. By setting $s_0 = 0$ and using the monotonicity of V^π, we obtain

$$\sum_{i=0}^{n-1} \gamma_k^i \,\mathbf{E}[c(s_i,\pi(s_i))] \le (1-\gamma_k^n)\, V^\pi(0). \tag{3.3}$$

Let us define the policy $\mathcal{P}_\alpha$ that only uses one bin at a time and follows the actions of π at every time step. Then it is easy to see that

$$\begin{aligned}\operatorname{cost}(\mathcal{P}_\alpha) &= 1 + \sum_{i=0}^{n-1} \mathbf{E}[c(s_1,\pi(s_i))] = 1 + \lim_{k\to\infty} \sum_{i=0}^{n} \gamma_k^i \,\mathbf{E}[c(s_i,\pi(s_i))] \\ &\le 1 + \lim_{k\to\infty} (1-\gamma_k^n) V^\pi(0). \end{aligned} \tag{3.4}$$

Where in the last inequality we utilized inequality (3.3).

Next, by optimality of π, we have $V^\pi(0) \le V_1^\Pi(0)$ for any $\Pi = (\pi_1,\pi_2,\ldots)$. Let $\widehat{\mathcal{P}}$ be the optimal sequential packing policy of $X_1, X_2, \ldots, X_n$ that always keeps at most one bin active at a time. For $t = 1,\ldots,n$, consider the functions $\widehat{\pi}_t(s) = 0$ if $\mathcal{P}(s)$ uses the current

bin and $\widehat{\pi}_t(s) = 1$ otherwise. Now, consider the policy $\widehat{\Pi} = (\widehat{\pi}_1, \ldots, \widehat{\pi}_n, \widehat{\pi}_1, \ldots, \widehat{\pi}_n, \ldots)$ that repeats cyclically the actions of $\widehat{\mathcal{P}}$. Then, as before we can expand the recursion and write

$$\begin{aligned} V_1^{\widehat{\Pi}}(0) &= \sum_{i=1}^{n-1} \gamma_k^i \, \mathbf{E}[c(s_i, \widehat{\pi}_{i+1}(s_i))] + \gamma_k^n \, \mathbf{E}[V_{n+1}^{\widehat{\Pi}}(s_{n-1}, \widehat{\pi}_n(s_{n-1}))] \\ &= \sum_{i=0}^{n-1} \gamma_k^i \, \mathbf{E}[c(s_i, \widehat{\pi}_{i+1}(s_i))] + \gamma_k^n \, \mathbf{E}[V_1^{\widehat{\Pi}}(s_n)] \\ &\leq \sum_{i=0}^{n-1} \gamma_k^i \, \mathbf{E}[c(s_i, \widehat{\pi}_{i+1}(s_i))] + \gamma_k^n (V_1^{\widehat{\Pi}}(0) + 1) \end{aligned}$$

where the last inequality can be shown by optimality of $\mathcal{P}$. Then

$$(1 - \gamma_k^n) V_1^{\widehat{\Pi}}(0) \leq \gamma_k^n + \sum_{i=0}^{n-1} \gamma_k^i \, \mathbf{E}[c(s_i, \widehat{\pi}_{i+1}(s_i))]. \tag{3.5}$$

Moreover,

$$\mathrm{cost}(\widehat{\mathcal{P}}) = 1 + \sum_{i=0}^{n-1} \mathbf{E}[c(s_i, \widehat{\pi}_{i+1}(s_i))], \tag{3.6}$$

and then we obtain

$$\begin{aligned} \mathrm{cost}(\mathcal{P}_\alpha) &\leq \lim_{k \to \infty} (1 - \gamma_k^n) V^\pi(0) && \text{(By (3.4))} \\ &\leq \lim_{k \to \infty} (1 - \gamma_k^n) V_1^{\widehat{\Pi}}(0) && \text{(Optimality of } \pi\text{)} \\ &\leq 1 + \lim_{k \to \infty} \sum_{i=0}^{n-1} \gamma_k^i \, \mathbf{E}[c(s_i, \pi_{i+1}(s_i))] + \gamma_k^n && \text{(By (3.5))} \\ &= 1 + \sum_{i=0}^{n-1} \mathbf{E}[c(s_i, \pi_{i+1}(s_i))] + 1 \\ &= \mathrm{cost}(\mathcal{P}) + 1. && \text{(By (3.6))} \end{aligned}$$

□

CHAPTER 4

ROBUST ONLINE SELECTION WITH UNCERTAIN OFFER ACCEPTANCE

Online advertising has motivated interest in online selection problems. Displaying ads to the right users benefits both the platform (e.g., via pay-per-click) and the advertisers (by increasing their reach). In practice, not all users click on displayed ads, while the platform's algorithm may miss the users most disposed to do so. This mismatch decreases the platform's revenue and the advertiser's chances to reach the right customers. With this motivation, we propose a secretary problem where a candidate may or may not accept an offer according to a known probability p. Because we do not know the top candidate willing to accept an offer, the goal is to maximize a robust objective defined as the minimum over integers k of the probability of choosing one of the top k candidates, given that one of these candidates will accept an offer. Using Markov decision process theory, we derive a linear program for this max-min objective whose solution encodes an optimal policy. The derivation may be of independent interest, as it is generalizable and can be used to obtain linear programs for many online selection models. We further relax this linear program into an infinite counterpart, which we use to provide bounds for the objective and closed-form policies. For $p \geq p^* \approx 0.6$, an optimal policy is a simple threshold rule that observes the first $p^{1/(1-p)}$ fraction of candidates and subsequently makes offers to the best candidate observed so far.

4.1 Introduction

The growth of online platforms has spurred renewed interest in online selection problems, auctions and stopping problems [70, 121, 64, 11, 129]. Online advertising has particularly

benefited from developments in these areas. As an example, in 2005 Google reported about $6 billion in revenue from advertising, roughly 98% of the company's total revenue at that time; in 2020, Google's revenue from advertising grew to almost $147 billion. Thanks in part to the economic benefits of online advertisement, internet users can freely access a significant amount of online content and many online services.

Targeting users is crucial for the success of online advertising. Studies suggest that targeted campaigns can double *click-through rates* in ads [71] despite the fact that internet users have acquired skills to navigate the web while ignoring ads [40, 67]. Therefore, it is natural to expect that not every displayed ad will be clicked on by a user, even if the user likes the product on the ad, whereas the platform and advertiser's revenue depend on this event [145]. An ignored ad misses the opportunity of being displayed to another user willing to click on it and decreases the return on investment (ROI) for the advertiser, especially in cases where the platform uses methods like pay-for-impression to charge the advertisers. At the same time, the ignored ad uses the space of another, possibly more suitable ad for that user. In this work, we take the perspective of a single ad, and we aim to understand the right time to begin displaying the ad to users as a function of the ad's probability of being clicked.

Recent works have addressed this uncertainty from a full-information perspective [129, 87], where user's valuations over the ads are known in hindsight. However, in many e-commerce applications it is unrealistic to assume access to complete information, as users may approach the platform sequentially. Platforms that depend at least partially on display advertisement for revenue observe sequential user visits and need to irrevocably decide when to display an ad. If a platform displays an ad too soon, it risks not being able to learn users' true preferences, while spending too much time learning preferences risks missing a good opportunity to display an ad. Designing efficient policies for displaying ads to users is key for large internet companies such as Google and Meta, as well as for small businesses

advertising their products in these platforms.

We model the interaction between the platform and the users using a general online selection problem. We refer to it as the *secretary problem with uncertain acceptance* (**SP-UA** for short). Using the terminology of *candidate* and *decision maker*, the general interaction is as follows:

1. Similar to other secretary problems, a finite sequence of candidates of known length arrives online, in a random order. In our motivating application, candidates represent platform users.

2. Upon an arrival, the decision maker (DM) is able to assess the quality of a candidate compared to previously observed candidates and has to irrevocably decide whether to extend an offer to the candidate or move on to the next candidate. This captures the online dilemma the platform faces: the decision of displaying an ad to a user is based solely on information obtained up to this point.

3. When the DM extends an offer, the candidate accepts with a known probability $p \in (0, 1]$, in which case the process ends, or turns down the offer, in which case the DM moves on to the next candidate. This models the users, who can click on the ad or ignore it.

4. The process continues until either a candidate accepts an offer or the DM has no more candidates to assess.

A DM that knows in advance that at least one of the top k candidates is willing to accept the offer would like to maximize the probability of making an offer to one of these candidates. In reality, the DM does not know k; hence, the best she can do is maximize the minimum of all these scenario-based probabilities. We call the minimum of these scenario-based probabilities the *robust ratio* and our max-min objective the *optimal robust ratio* (see

Subsection 4.1.2 for a formal description). Suppose that the DM implements a policy that guarantees a robust ratio $\gamma \in (0, 1]$. This implies the DM will succeed with probability at least γ in obtaining a top k candidate, in any scenario where a top k candidate is willing to accept the DM's offer. This is an ex-ante guarantee when the DM knows the odds for each possible scenario, but the policy is independent of k and offers the same guarantee for any of these scenarios. Moreover, if the DM can assign a numerical valuation to the candidates, a policy with robust ratio γ can guarantee a factor at least γ of the optimal offline value. [164] also studies the **SP-UA** and considers the objective of maximizing the probability of selecting the best candidate willing to accept the offer. Applying his policy to value settings can also guarantee an approximation factor of the optimal offline cost; however, the policy with the optimal robust ratio attains the largest approximation factor of the optimal offline value among rank-based policies (see Proposition 4.5).

Contributions (1) We propose a framework and a robust metric to understand the interaction between a DM and competing candidates, when candidates can reject the DM's offer. (2) We state a linear program (LP) that computes the optimal robust ratio and the best strategy. We provide a general methodology to deduce our LP, and this technique is generalizable to other online selection problems. (3) We provide bounds for the optimal robust ratio as a function of the probability of acceptance $p \in (0, 1]$. (4) We present a family of policies based on simple threshold rules; in particular, for $p \geq p^* \approx 0.594$, the optimal strategy is a simple threshold rule that skips the first $p^{1/(1-p)}$ fraction of candidates and then makes offers to the best candidate observed so far. We remark that as $p \to 1$ we recover the guarantees of the standard secretary problem and its optimal threshold strategy. (5) Finally, for the setting where candidates also have non-negative numerical values, we show that our solution is the optimal approximation among rank-based algorithms of the optimal offline value, where the benchmark knows the top candidate willing to accept the offer. The optimal approximation factor equals the optimal robust ratio.

In the remainder of the section, we discuss additional applications of **SP-UA**, give a formal description of the problem, and summarize our technical contributions and results.

4.1.1 Additional Applications

SP-UA captures the inherent unpredictability in online selection, as other secretary problems do, but also the uncertainty introduced by the possibility of candidates turning down offers. Although our main motivation is online advertisement, **SP-UA** is broadly applicable; the following are additional concrete examples.

Data-driven selection problems When selling an item in an auction, buyers' valuations are typically unknown beforehand. Assuming valuations follow a common distribution, the aim is to sell the item at the highest price possible; learning information about the distribution is crucial for this purpose. In particular auction settings, the auctioneer may be able to sequentially observe the valuations of potential buyers, and can decide in an online manner whether to sell the item or continue observing valuations. Specifically, the auctioneer decides to consider the valuation of a customer with probability p and otherwise the auctioneer moves on to see the next buyer's valuation. The auctioneer's actions can be interpreted as an *exploration-exploitation* process, which is often found in bandit problems and online learning [37, 97, 77]. This setting is also closely related to data-driven online selection and the prophet inequality problem [36, 105, 107]; some of our results also apply in these models (see Section 4.6).

Human resource management As its name suggests, the original motivation for the secretary problem is in hiring for a job vacancy. Glassdoor reported in 2015 that each job posting receives on average 250 resumes. From this pool of resumes, only a small fraction of candidates are called for an interview. Screening resumes can be a time-consuming task that shift resources away from the day-to-day job in Human Resources. Since the advent of the internet, several elements of the hiring process can be partially or completely

automated; for example, multiple vendors offer automated resume screening [147], and machine learning algorithms can score and rank job applicants according to different criteria. Of course, a highly ranked applicant may nevertheless turn down a job offer. Although we consider the rank of a candidate as an absolute metric of their capacities, in reality, resume screening may suffer from different sources of bias [153], but addressing this goes beyond our scope. See also [161, 164, 169] for classical treatments. Similar applications include apartment hunting [32, 52, 144], among others.

4.1.2 Problem Formulation

A problem instance is given by a fixed probability $p \in (0, 1]$ and the number of candidates n. These are ranked by a total order, $1 \prec 2 \prec \cdots \prec n$, with 1 being the best or highest-ranked candidate. The candidate sequence is given by a random permutation $\pi = (R_1, \ldots, R_n)$ of $[n] \doteq \{1, 2, \ldots, n\}$, where any permutation is equally likely. At time t, the DM observes the partial rank $r_t \in [t]$ of the t-th candidate in the sequence compared to the previous $t - 1$ candidates. The DM either makes an offer to the t-th candidate or moves on to the next candidate, without being able to make an offer to the t-th candidate ever again. If the t-th candidate receives an offer from the DM, she accepts the offer with probability p, in which case the process ends. Otherwise, if the candidate refuses the offer (with probability $1 - p$), the DM moves on to the next candidate and repeats the process until she has exhausted the sequence. A candidate with rank in $[k]$ is said to be a *top k candidate*. The goal is to design a policy that maximizes the probability of extending an offer to a highly ranked candidate that will accept the DM's offer. To measure the quality of a policy $\mathcal{P}$, we use the *robust ratio*

$$\gamma_{\mathcal{P}} = \gamma_{\mathcal{P}}(p) = \min_{k=1,\ldots,n} \frac{\mathbf{P}(\mathcal{P} \text{ selects a top } k \text{ candidate, candidate accepts offer})}{\mathbf{P}(\text{At least one of the top } k \text{ candidates accepts offer})}. \quad (4.1)$$

The ratio inside the minimization operator equals the probability that the policy successfully selects a top k candidate given that some top k candidate will accept an offer. If we consider the game where an adversary knows in advance if any of the top k candidates will accept the offer, the robust ratio $\gamma_{\mathcal{P}}$ captures the scenario where the DM that follows policy $\mathcal{P}$ has the worst performance. When $p = 1$, the robust ratio $\gamma_{\mathcal{P}}$ equals the probability of selecting the highest ranked candidate, thus we recover the standard secretary problem. The goal is to find a policy that maximizes this robust ratio, $\gamma_n^* \doteq \sup_{\mathcal{P}} \gamma_{\mathcal{P}}$. We say that the policy $\mathcal{P}$ is γ-*robust* if $\gamma \leq \gamma_{\mathcal{P}}$.

4.1.3 Technical Contributions

Recent works have studied secretary models using linear programming (LP) methods [34, 38, 50, 68]. We also give an LP formulation that computes the best robust ratio and the optimal policy for our model. Whereas these recent approaches derive an LP formulation using ad-hoc arguments, our first contribution is to provide a general framework to obtain LP formulations that give optimal bounds and policies for different variants of the secretary problem. The framework is based on Markov decision process (MDP) theory [14, 146]. This is surprising since early literature on secretary problem used MDP techniques, e.g. [69, 120], though typically not LP formulations. In that sense, our results connect the early algorithms based on MDP methods with the recent literature based on LP methods. Specifically, we provide a mechanical way to obtain an LP using a simple MDP formulation (Section 4.4). Using this framework, we present a structural result that completely characterizes the space of policies for the **SP-UA**:

Theorem 4.1. *Any policy $\mathcal{P}$ for the **SP-UA** can be represented as a vector in the set*

$$\textsc{Pol} = \left\{ (\mathbf{x}, \mathbf{y}) \geq 0 : x_{t,s} + y_{t,s} = \frac{1}{t} \sum_{\sigma=1}^{t-1} \left(y_{t-1,\sigma} + (1-p) x_{t-1,\sigma} \right), \forall t > 1, s \in [t], x_{1,1} + y_{1,1} = 1 \right\}.$$

Conversely, any vector $(\mathbf{x}, \mathbf{y}) \in$ Pol represents a policy $\mathcal{P}$. The policy $\mathcal{P}$ makes an

offer to the first candidate with probability $x_{1,1}$ *and to the* t*-th candidate with probability* $tx_{t,s}/\left(\sum_{\sigma=1}^{t-1} y_{t-1,\sigma} + (1-p)x_{t-1,\sigma}\right)$ *if the* t*-th candidate has partial rank* $r_t = s$*.*

The variables $x_{t,s}$ represent the probability of reaching candidate t and making an offer to that candidate when that candidate has partial rank $s \in [t]$. Likewise, the variables $y_{t,s}$ represent the probability of reaching candidate t and moving on to the next candidate when the t-th candidate's partial rank is $s \in [t]$. We note that although the use of LP formulations in MDP is a somewhat standard technique, see e.g. [146], the recent literature in secretary problems and related online selection models does not appear to make an explicit connection between LP's used in analysis and the underlying MDP formulation.

Problems solved via MDP can typically be formulated as reward models, where each action taken by the DM generates some immediate reward. Objectives in classical secretary problems fit in this framework, as the reward (e.g. the probability of selecting the top candidate) depends only on the current state (the number t of observed candidates so far and the current candidate's partial rank $r_t = s$), and on the DM's action (make an offer or not); see Section 4.4.1 for an example. Our robust objective, however, cannot be easily written as a reward depending only on $r_t = s$. Thus, we split the analysis into two stages. In the first stage, we deal with the space of policies and formulate an MDP for our model with a generic utility function. The feasible region of this MDP's LP formulation corresponds to POL and is independent of the utility function chosen; therefore, it characterizes all possible policies for the **SP-UA**. In the second stage, we use the structural result in Theorem 4.1 to obtain a linear program that finds the largest robust ratio.

Theorem 4.2. *The best robust ratio* γ_n^* *for the* ***SP-UA*** *equals the optimal value of the linear program*

$$
\begin{aligned}
&\max_{\mathbf{x}\geq 0} && \gamma \\
&\text{s.t.} && \\
(LP)_{n,p} \qquad & x_{t,s} \leq \frac{1}{t}\left(1 - p\sum_{\tau=1}^{t-1}\sum_{\sigma=1}^{\tau} x_{\tau,\sigma}\right) && \forall t \in [n], s \in [t] \\
& \gamma \leq \frac{p}{1-(1-p)^k}\sum_{t=1}^{n}\sum_{s=1}^{t} x_{t,s}\,\mathbf{P}(R_t \leq k \mid r_t = s) && \forall k \in [n],
\end{aligned}
$$

where $\mathbf{P}(R_t \leq k \mid r_t = s) = \sum_{i=s}^{k\wedge(n-t+s)} \binom{i-1}{s-1}\binom{n-i}{t-s}/\binom{n}{t}$ *is the probability the* t*-th candidate is ranked in the top* k *given that her partial rank is* s*.*

Moreover, given an optimal solution $(\mathbf{x}^*, \gamma_n^*)$ *of* $(LP)_{n,p}$*, the (randomized) policy* $\mathcal{P}^*$ *that at state* (t, s) *makes an offer with probability* $tx^*_{t,s}/\big(1 - p\sum_{\tau=1}^{t-1}\sum_{\sigma=1}^{\tau} x^*_{\tau,\sigma}\big)$ *is* γ_n^**-robust.*

We show that $\gamma_{\mathcal{P}}$ can be written as the minimum of n linear functions on the $\mathbf{x}$ variables in POL, where these variables correspond to a policy's probability of making an offer in a given state. Thus our problem can be written as the maximum of a concave piecewise linear function over POL, which we linearize with the variable γ. By projecting the feasible region onto the $(\mathbf{x}, \gamma)$ variables we obtain $(LP)_{n,p}$.

As a byproduct of our analysis via MDP, we show that γ_n^* is non-increasing in n for fixed $p \in (0, 1]$ (Lemma 4.7), and thus $\lim_{n\to\infty} \gamma_n^* = \gamma_\infty^*$ exists. We show that this limit corresponds to the optimal value of an infinite version of $(LP)_{n,p}$ from Theorem 4.2, where n tends to infinity and we replace sums at time t with integrals (see Section 4.5). This allows us to show upper and lower bounds for γ_n^* by analyzing γ_∞^*. Our first result in this vein gives upper bounds on γ_∞^*.

Theorem 4.3. *For any* $p \in (0, 1]$*,* $\gamma_\infty^*(p) \leq \min\left\{p^{p/(1-p)}, 1/\beta\right\}$*, where* $1/\beta \approx 0.745$ *and* β *is the (unique) solution of the equation* $\int_0^1 (y(1 - \log y) + \beta - 1)^{-1}\mathrm{d}y = 1$*.*

To show $\gamma^*_\infty \leq p^{p/(1-p)}$, we relax all constraints in the robust ratio except $k = 1$. This becomes the problem of maximizing the probability of hiring the top candidate, which has a known asymptotic solution of $p^{1/(1-p)}$ [161]. For $\gamma^*_\infty(p) \leq 1/\beta$, we show that any γ-robust ordinal algorithm can be used to construct an algorithm for i.i.d. prophet inequality problems with a multiplicative loss of $(1 + o(1))\gamma$ and an additional $o(1)$ additive error. Using a slight modification of the impossibility result by [98] for the i.i.d. prophet inequality, we conclude that γ^*_∞ cannot be larger than $1/\beta$.

By constructing solutions of the infinite LP, we can provide lower bounds for γ^*_n. For $1/k \geq p > 1/(k+1)$ with integer k, the policy that skips the first $1/e$ fraction of candidates and then makes an offer to any top k candidate afterwards obtains a robust ratio of at least $1/e$. The following result gives improved bounds for $\gamma^*_\infty(p)$.

Theorem 4.4. *Let $p^* \approx 0.594$ be the solution of $p^{(2-p)/(1-p)} = (1-p)^2$. There is a solution of the infinite LP for $p \geq p^*$ that guarantees $\gamma^*_n \geq \gamma^*_\infty(p) = p^{p/(1-p)}$. For $p \leq p^*$ we have $\gamma^*_\infty(p) \geq (p^*)^{p^*/(1-p^*)} \approx 0.466$.*

To prove this result, we use the following general procedure to construct feasible solutions for the infinite LP. For any numbers $0 < t_1 \leq t_2 \leq \cdots \leq t_k \leq \cdots \leq 1$, there is a policy that makes offers to any candidate with partial rank $r_t \in [k]$ when a fraction t_k of the total number of candidates has been observed (Proposition 4.9). For $p \geq p^*$, the policy corresponding to $t_1 = p^{1/(1-p)}$ and $t_2 = t_3 = \cdots = 1$ has a robust ratio of at least $p^{p/(1-p)}$. For $p \leq p^*$, we show how to transform the solution for p^* into a solution for p with an objective value at least as good as the value $\gamma^*_\infty(p^*) = (p^*)^{p^*/(1-p^*)}$.

Figure 4.1 depicts the various theoretical bounds we obtain. For reference, we also include numerical results for γ^*_n computed by solving $(LP)_{n,p}$ in Theorem 4.2 for $n = 200$ and with p ranging from $p = 10^{-2}$ to $p = 1$, with increments of 10^{-3}. Since γ^*_n is nonincreasing in n, the numerical values obtained by solving $(LP)_{n,p}$ also provide an upper bound over γ^*_∞.

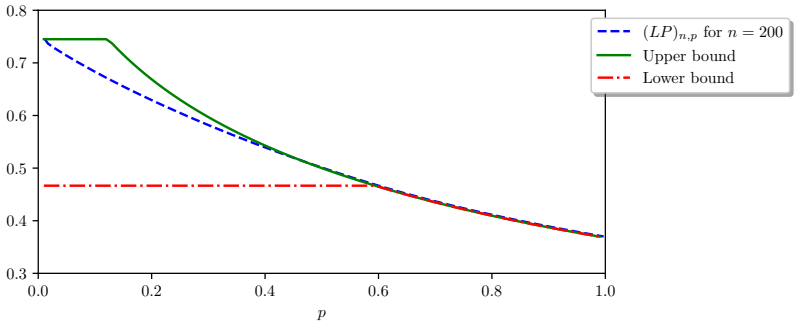

Figure 4.1: Bounds for γ^*_∞ as a function of p. The solid line represents the theoretical upper bound given in Theorem 4.3. The dashed-dotted line corresponds to the theoretical lower bound given in Theorem 4.4. In dashed line we present numerical results by solving $(LP)_{n,p}$ for $n = 200$ candidates.

We follow this introduction with a brief literature review. In Section 4.3 we present preliminaries, including MDP notation and an alternative characterization of the robust ratio in terms of utility functions. In Section 4.4 we present the MDP framework and use it to prove Theorems 4.1 and 4.2. In Section 4.5 we introduce the infinite relaxation of (LP), then prove Theorem 4.3 in Section 4.6. In Section 4.7 we prove Theorem 4.4. In Section 4.8 we present a numerical comparison between the policies obtained by solving $(LP)_{n,p}$ and other benchmarks policies. We conclude in Section 4.9, and an appendix includes proofs and analysis omitted from the main body of the chapter.

4.2 Related Work

Online advertising and online selection Online advertising has been extensively studied from the viewpoint of two-sided markets: advertisers and platform. There is extensive work in auction mechanisms to select ads (e.g. second-price auctions, the VCG mechanism, etc.), and the payment systems between platforms and advertisers (pay-per-click, pay-for-impression, etc.) [65, 70, 78]; see also [41] for a review. On the other hand, works relating the platform, advertisers, and web users have been studied mainly from a learning

perspective, to improve ad targeting [65, 71, 97]. In this work, we also aim to display an ad to a potentially interested user. Multiple online selection problems have been proposed to display ads in online platforms, e.g., packing models [18, 114], secretary problems and auctions [19], prophet models [11] and online models with "buyback" [17]. In our setting, we add the possibility that a user ignores the ad; see e.g. [40, 67]. Failure to click on ads has been considered in full-information models [87]; however, our setting considers only partial information, where the rank of an incoming customer can only be assessed relative to previously observed customers—a typical occurrence in many online applications. Our model is also disaggregated and looks at each ad individually. Our goal is to understand the right time to display an ad/make offers via the **SP-UA** and the robust ratio for each individual ad.

Online algorithms and arrival models Online algorithms have been extensively studied for adversarial arrivals [29]. This *worst-case* viewpoint gives robust algorithms against any input sequence, which tend to be conservative. Conversely, some models assume distributional information about the inputs [107, 113, 121]. The random order model lies in between these two viewpoints, and perhaps the most studied example is the secretary problem [69, 83, 120]. Random order models have also been applied in Adword problems [64], online LP's [10] and online knapsacks [18, 108], among others.

Secretary problems Martin Gadner popularized the secretary problem in his 1960 *Mathematical Games* column; for a historical review, see [74] and also the classical survey by [75]. For the classical secretary problem, the optimal strategy that observes the first n/e candidates and thereafter selects the best candidate was computed by [120, 83]. The model has been extensively studied in ordinal/ranked-based settings [120, 92, 168, 34] as well as cardinal/value-based settings [23, 110].

A large body of work has been dedicated to augment the secretary problem. Variations

include cardinality constraints [34, 168, 110], knapsack constraints [18], and matroid constraints [162, 72, 115]. Model variants also incorporate different arrival processes, such as Markov chains [100] and more general processes [68]. Closer to our problem are the data-driven variations of the model [50, 51, 105], where samples from the arriving candidates are provided to the decision maker. Our model can be interpreted as an online version of sampling, where a candidate rejecting the decision maker's offer is tantamount to a sample. This also bears similarity to the exploration-exploitation paradigm often found in online learning and bandit problems [37, 97, 77].

Uncertain availability in secretary problems The **SP-UA** is studied by [161] with the goal of selecting the top candidate — $k = 1$ in (4.1) — who gives an asymptotic probability of success of $p^{1/(1-p)}$. If the top candidate rejects the offer, this leads to zero value, which is perhaps excessively pessimistic in scenarios where other competent candidates could accept. [164] considers maximizing the probability of selecting the top candidate *among* the candidates that will accept the offer. Although more realistic, this objective still gives zero value when the top candidate that accepts is missed because she arrives early in the sequence. In our approach, we make offers to candidates even if we have already missed the top candidate that accepts the offer; this is also appealing in utility/value-based settings (see Proposition 4.5). We also further the understanding of the model and our objective by presenting closed-form solutions and bounds. See also [32, 144, 52].

Linear programs in secretary problems Early work in secretary problems mostly used MDPs [120, 161, 164]. Linear programming formulations were introduced by [34], and from there multiple formulations have been used to solve variants of the secretary problem [38, 50, 68]. We use MDP to derive the polyhedron that encodes policies for the **SP-UA**. The connection between linear programs and MDP has been explored in other online selection and allocation problems, such as network revenue management [9], knapsack and prophet problems [102], more general MDP models [56, 127, 146] and particularly in con-

strained MDP's [14, 94, 95]. To the best of our knowledge, these connections have not been explored for secretary problems.

4.3 Preliminaries

To discuss our model, we use standard MDP notation for secretary problems [69, 75, 120]. An instance is characterized by the number of candidates n and the probability $p \in (0, 1]$ that an offer is accepted. For $t \in [n]$ and $s \in [t]$, a *state* of the system is a pair (t, s) indicating that the candidate currently being evaluated is the t-th and the corresponding partial rank is $r_t = s$. To simplify notation, we add the states $(n+1, s)$, $s \in [n+1]$, and the state Θ as absorbing states where no decisions can be made. For $t < n$, transitions from a state (t, s) to a state $(t+1, \sigma)$ are determined by the random permutation $\pi = (R_1, \ldots, R_n)$. We denote by $S_t \in \{(t, s)\}_{s\in[t]}$ the random variable indicating the state in the t-th stage. A simple calculation shows

$$\mathbf{P}(S_{t+1} = (t+1, \sigma) \mid S_t = (t, s)) = \mathbf{P}(r_{t+1} = \sigma \mid r_t = s) = \mathbf{P}(S_{t+1} = (t+1, \sigma)) = 1/(t+1),$$

for $t < n$, $s \in [t]$ and $\sigma \in [t + 1]$. In other words, partial ranks at each stage are independent. For notational convenience, we assume the equality also holds for $t = n$. Let $\mathbf{A} = \{\textbf{offer}, \textbf{pass}\}$ be the set of *actions*. For $t \in [n]$, given a state (t, s) and an action $A_t = a \in \mathbf{A}$, the system transitions to a state S_{t+1} with the following probabilities :

$$P_{((t,s),a),(\tau,\sigma)} = \mathbf{P}(S_{t+1} = (\tau, \sigma) \mid S_t = (t, s), A_t = a) = \begin{cases} \frac{1-p}{t+1} & a = \textbf{offer}, \tau = t + 1, \sigma \in [\tau] \\ p & a = \textbf{offer}, (\tau, \sigma) = \Theta \\ \frac{1}{t+1} & a = \textbf{pass}. \end{cases}$$

The randomness is over the permutation π and the random outcome of the t-th candidate's decision. We utilize states $(n + 1, \sigma)$ as end states and the state Θ as the state indicating

that an offer is accepted from the state S_t. A *policy* $\mathcal{P} : \{(t, s) : t \in [n], s \in [t]\} \to \mathbf{A}$ is a function that observes a state (t, s) and decides to extend an offer ($\mathcal{P}(t, s) =$ **offer**) or move to the next candidate ($\mathcal{P}(t, s) =$ **pass**). The policy specifies the actions of a decision maker at any point in time. The *initial state* is $S_1 = (1, 1)$ and the computation (of a policy) is a sequence of state and actions $(1, 1), a_1, (2, s_2), a_2, (3, s_3), \ldots$ where the states transitions according to $P_{((t,s),a),(t+1,\sigma)}$ and $a_t = \mathcal{P}(t, s_t)$. Note that the computation always ends in a state $(n + 1, \sigma)$ for some σ or the state Θ, either because the policy was able to go through all candidates or because some candidate t accepted an offer.

We say that a policy reaches stage t or reaches the t-th stage if the computation of a policy contains a state $s_t = (t, s)$ for some $s \in [t]$. We also refer to stages as *times*.

A randomized policy is a function $\mathcal{P} : \{(t, s) : t \in [n], s \in [t]\} \to \Delta_{\mathbf{A}}$ where $\Delta_{\mathbf{A}} = \{(q, 1 - q) : q \in [0, 1]\}$ is the probability simplex over $\mathbf{A} = \{$**offer**, **pass**$\}$ and $\mathcal{P}(s_t) = (q_t, 1 - q_t)$ means that $\mathcal{P}$ selects the **offer** action with probability q_t and otherwise selects **pass**.

We could also define policies that remember previously visited states and at state (t, s_t) make decisions based on the *history*, $(1, s_1), \ldots, (t, s_t)$. However, MDP theory guarantees that it suffices to consider Markovian policies, which make decisions based only on (t, s_t); see [146].

We say that a policy $\mathcal{P}$ *collects* a candidate with rank k if the policy extends an offer to a candidate that has rank k *and* the candidate accepts the offer. Thus our objective is to find a policy that solves

$$\begin{aligned}\gamma_n^* &= \max_{\mathcal{P}} \min_{k \in [n]} \frac{\mathbf{P}(\mathcal{P} \text{ collects a candidate with rank} \le k)}{1 - (1 - p)^k} \\ &= \max_{\mathcal{P}} \min_{k \in [n]} \mathbf{P}(\mathcal{P} \text{ collects a top } k \text{ candidate} \mid \text{a top } k \text{ candidate accepts}).\end{aligned}$$

The following result is an alternative characterization of γ_n^* based on utility functions. We use this result to relate **SP-UA** to the i.i.d. prophet inequality problem; the proof appears in Appendix 4.A.1. Consider a nonzero utility function $U : [n] \to \mathbf{R}_+$ with $U_1 \geq U_2 \geq \cdots \geq U_n \geq 0$, and any rank0based algorithm ALG for the **SP-UA**, i.e., ALG only makes decisions based on the relative ranking of the values observed. In the value setting, if ALG collects a candidate with overall rank i, it obtains value U_i. We denote by $U(\text{ALG})$ the value collected by such an algorithm.

Proposition 4.5. *Let* ALG *be a γ-robust algorithm for **SP-UA**. For any $U : [n] \to \mathbf{R}_+$ we have $\mathbf{E}[U(\text{ALG})] \geq \gamma\, \mathbf{E}[U(\text{OPT})]$ where* OPT *is the maximum value obtained from candidates that accept. Moreover,*

$$\gamma_n^* = \max_{\text{ALG}} \min \left\{ \frac{\mathbf{E}\left[U(\text{ALG})\right]}{\mathbf{E}\left[U(\text{OPT})\right]} : U : [n] \to \mathbf{R}_+, U \neq 0, U_1 \geq U_2 \geq \cdots \geq U_n \geq 0 \right\}.$$

4.4 The LP Formulation

In this section, we present the proofs of Theorems 4.1 and 4.2. Our framework is based on MDP and can be used to derive similar LPs in the literature, e.g. [34, 38, 50]. As a byproduct, we also show that γ_n^* is a nonincreasing sequence in n (Lemma 4.7). For ease of explanation, we first present the framework for the classical secretary problem, then we sketch the approach for our model. Technical details are deferred to the appendix.

4.4.1 Warm-up: MDP to LP in the Classical Secretary Problem

We next show how to derive an LP for the classical secretary problem [34] using an MDP framework. In this model, the goal is to maximize the probability of choosing the top candidate, and there is no offer uncertainty.

Theorem 4.6 ([34])**.** *The maximum probability of choosing the top-ranked candidate in the*

classical secretary problem is given by

$$\max\left\{\sum_{t=1}^{n}\frac{t}{n}x_t : x_t \leq \frac{1}{t}\left(1-\sum_{\tau=1}^{t-1}x_\tau\right), \forall t \in [n], \mathbf{x}\geq 0\right\}.$$

We show this as follows:

1. First, we formulate the secretary problem as a Markov decision process, where we aim to find the highest ranked candidate. Let $v^*_{(t,s)}$ be the maximum probability of selecting the highest ranked candidate in $t+1,\ldots,n$ given that the current state is (t,s). We define $v^*_{(n+1,s)} = 0$ for any s. The value v^* is called the *value function* and it can be computed via the optimality equations [146]

$$v^*_{(t,s)} = \max\left\{\mathbf{P}(R_t = 1 \mid r_t = s), \frac{1}{t+1}\sum_{\sigma=1}^{t} v^*_{(t+1,\sigma)}\right\}. \tag{4.2}$$

   The first term in the max operator corresponds to the expected value when the **offer** action is chosen in state (t,s). The second corresponds to the expected value in stage $t+1$ when we decide to **pass** in (t,s). Note that $\mathbf{P}(R_t = 1 \mid r_t = s) = t/n$ if $s = 1$ and $\mathbf{P}(R_t = 1 \mid r_t = s) = 0$ otherwise. The optimality equations (4.2) can be solved via backwards recursion, and $v^*_{(1,1)} \approx 1/e$ (for large n). An optimal policy can be obtained from the optimality equations by choosing at each state an action that attains the maximum, breaking ties arbitrarily.

2. Using a standard argument [127], it follows that $v^* = (v^*_{(t,s)})_{t,s}$ is an optimal solution of the linear program (D) in Figure 4.2.

3. Taking the dual of (D), we obtain (P) in Figure 4.2. Variables $x_{t,s}$ are associated with constraints (4.3), $y_{t,s}$ with constraints (4.4). Take any solution $(\mathbf{x},\mathbf{y})$ of the problem (P) and note that the objective does not depend on $\mathbf{y}$. Incrementing $\mathbf{y}$ to tighten all constraints does not alter the feasibility of the solution and the objective does not change;

$$
\begin{array}{l|l}
(D) \quad \min_{v \geq 0} \quad v_{(1,1)} & (P) \quad \max_{x,y \geq 0} \quad \sum_{t=1}^{n} \frac{t}{n} x_{t,1} \\
\text{s.t.} & \text{s.t.} \\
\qquad v_{(t,s)} \geq \mathbf{P}(R_t = 1 \mid r_t = s) \quad \forall t \leq n, \forall s \quad (4.3) & \qquad x_{1,1} + y_{1,1} \leq 1 \quad (4.5) \\
\qquad v_{(t,s)} \geq \frac{1}{1+t} \sum_{\sigma=1}^{t+1} v_{(t+1,\sigma)} \quad \forall t, s \quad (4.4) & \qquad x_{t,s} + y_{t,s} \leq \frac{1}{t} \sum_{\sigma=1}^{t-1} y_{t-1,\sigma} \quad \forall t, s \quad (4.6)
\end{array}
$$

Figure 4.2: Linear program that finds value function v^* and its dual.

thus we can assume that all constraints are tight in (P). Here, $x_{t,s}$ is the probability that a policy (determined by $\mathbf{x}, \mathbf{y}$) reaches the state (t, s) and makes an offer, while $y_{t,s}$ is the probability that the same policy reaches state (t, s) but decides to pass.

4. Finally, projecting the feasible region of (P) onto the variables $(x_{t,1})_{t \in [n]}$, e.g. via Fourier-Motzkin elimination (see [154] for a definition), gives us Theorem 4.6. We skip this for brevity.

The same framework can be applied to obtain the linear program for the secretary problem with *rehire* [34] and the formulation for the (J, K)-*secretary problem* [34, 38]. It can also be used to derive an alternative proof of the result by [161].

4.4.2 Framework for the **SP-UA**

Next, we sketch the proof of Theorem 4.1 and use it to derive Theorem 4.2. Technical details are deferred to the appendix.

In the classical secretary problem, the objective is to maximize the probability of choosing the top candidate, which we can write in the recursion of the value function v^*. For our model, the objective $\gamma_{\mathcal{P}}$ corresponds to a multi-objective criteria, and it is not clear a priori how to write the objective as a reward. We present a two-step approach: (1) First, we follow the previous subsection's argument to uncover the polyhedron of policies; (2) second, we show that our objective function can be written in terms of variables in this polyhedron,

and we maximize this objective in the polyhedron.

The Polyhedron of Policies via a Generic Utility Function

When we obtained the dual LP (P) (Figure 4.2), anything related to the objective of the MDP is moved to the objective value of the LP, while anything related to the actions of the MDP remained in constraints (4.5)-(4.6). This suggests using a generic utility function to uncover the space of policies. Consider any vector $U : [n] \to \mathbf{R}_+$, and suppose that our objective is to maximize the utility collected, where choosing a candidate of rank i means obtaining $U_i \geq 0$ value. Let $v^*_{(t,s)}$ be the maximum value collected in times $t, t+1, \ldots, n$ given that the current state is (t, s), where $v^*_{(n+1,s)} = 0$. Then, the optimality equations yield

$$v^*_{(t,s)} = \max \left\{ pU_t(s) + (1-p)\frac{1}{t+1}\sum_{\sigma=1}^{t+1} v^*_{(t+1,\sigma)}, \frac{1}{t+1}\sum_{\sigma=1}^{t+1} v^*_{(t+1,\sigma)} \right\}, \qquad (4.7)$$

where $U_t(s) = \sum_{i=1}^{n} U_i \, \mathbf{P}(R_t = i \mid r_t = s)$. The term in the left side of the max operator is the expected value obtained by an **offer** action, while the term in the right corresponds to the expected value of the **pass** action. Using an approach similar to the one used in steps 2 and 3 from the previous subsection, we can deduce that

$$\textsc{Pol} = \left\{ (\mathbf{x}, \mathbf{y}) \geq 0 : x_{1,1} + y_{1,1} = 1, x_{t,s} + y_{t,s} = \frac{1}{t}\sum_{\sigma=1}^{t-1} \left(y_{t-1,\sigma} + (1-p)x_{t-1,\sigma} \right), \forall t > 1, s \in [t] \right\}$$

contains all policies (Theorem 4.1). A formal proof is presented in the appendix.

The Linear Program

Next, we consider Theorem 4.2. Given a policy $\mathcal{P}$, we define $x_{t,s}$ to be the probability of reaching state (t, s) and making an offer to the candidate, and $y_{t,s}$ to be the probability of

reaching (t, s) and passing. Then $(\mathbf{x}, \mathbf{y})$ belongs to POL. Moreover,

$$\mathbf{P}(\mathcal{P} \text{ collects a top } k \text{ candidate}) = p \sum_{t=1}^{n} \sum_{s=1}^{t} x_{t,s} \, \mathbf{P}(R_t \leq k \mid r_t = s). \tag{4.8}$$

Conversely, any point $(\mathbf{x}, \mathbf{y}) \in \text{POL}$ defines a policy $\mathcal{P}$: At state (t, s), it extends an **offer** to the t-th candidate with probability $x_{1,1}$ if $t = 1$, or probability $t x_{t,s} / \left(\sum_{\sigma=1}^{t-1} y_{t-1,\sigma} + (1-p) x_{t-1,\sigma} \right)$ if $t > 1$. Also, $\mathcal{P}$ satisfies (4.8). Thus,

$$\begin{aligned}
\gamma_n^* &= \max_{\mathcal{P}} \min_{k \in [n]} \frac{\mathbf{P}(\mathcal{P} \text{ collects a top } k \text{ candidate})}{1 - (1-p)^k} \\
&= \max_{(x,y) \in \text{POL}} \min_{k \in [n]} \frac{p \sum_{t=1}^{n} \sum_{s=1}^{t} x_{t,s} \, \mathbf{P}(R_t \leq k \mid r_t = s)}{1 - (1-p)^k} \\
&= \max \left\{ \gamma : (\mathbf{x}, \mathbf{y}) \in \text{POL}, \gamma \leq \frac{p \sum_{t=1}^{n} \sum_{s=1}^{t} x_{t,s} \, \mathbf{P}(R_t \leq k \mid r_t = s)}{1 - (1-p)^k}, \forall k \in [n] \right\}.
\end{aligned} \tag{4.9}$$

Projecting the feasible region of (4.9) as in step 4 onto the $(\mathbf{x}, \gamma)$-variables gives us Theorem 4.2. The details appear in Appendix 4.A.2.

Our MDP framework also allows us to show the following monotonicity result.

Lemma 4.7. *For a fixed $p \in (0, 1]$, we have $\gamma_n^* \geq \gamma_{n+1}^*$ for any $n \geq 1$.*

We sketch the proof of this result here and defer the details to Appendix 4.A.3. The dual of the LP (4.9) can be reformulated as

$$\text{(DLP)} \qquad \begin{aligned} \min_{\substack{u:[n] \to \mathbf{R}_+ \\ \sum_i u_i \geq 1}} \quad & v_{(1,1)} \\ \text{s.t.} \quad & v_{(t,s)} \geq \max \left\{ U_t(s) + \frac{1-p}{t+1} \sum_{\sigma=1}^{t=1} v_{(t+1,\sigma)}, \frac{1}{t+1} \sum_{\sigma=1}^{t=1} v_{(t+1,\sigma)} \right\} \quad \forall t \in [n], s \in [t] \\ & v_{(n+1,s)} = 0 \qquad \forall s \in [n+1], \end{aligned}$$

where $U_t(s) = p \sum_{j=1}^{n} \left(\sum_{k \geq j} u_k / \left(1 - (1-p)^k\right) \right) \mathbf{P}(R_t = j \mid r_t = s)$. The variables

$u_1, \ldots, u_n$ correspond to the constraints involving γ in the LP (4.9). Note that (DLP) is the minimum value that an MDP can attain when the utility functions are given by $U_i = p \sum_{k \geq i} u_k / \big(1 - (1-p)^k\big)$. Taking any weighting $u : [n] \to \mathbf{R}_+$ with $\sum_i u_i \geq 1$, we extend it to $\widehat{u} : [n+1] \to \mathbf{R}_+$ by setting $\widehat{u}_{n+1} = 0$. We define accordingly $\widehat{U}_i = p \sum_{k \geq i} \widehat{u}_k / \big(1 - (1-p)^k\big)$, and note that $U_i = \widehat{U}_i$ for $i \leq n$ and $\widehat{U}_{n+1} = 0$. Using a coupling argument, from any policy for utilities $\widehat{U}$ with $n+1$ candidates, we can construct a policy for utilities U, with n candidates, where both policies collect the same utility. Thus, the utility collected by the optimal policy for U upper bounds the utility collected by an optimal policy for $\widehat{U}$. The conclusion follows since γ^*_{n+1} is a lower bound for the latter value.

Since $\gamma^*_n \in [0,1]$ and $(\gamma^*_n)_n$ is a monotone sequence in n, $\lim_{n \to \infty} \gamma^*_n$ must exist. In the next section we show that the limit corresponds to the value of a continuous LP.

4.5 The Continuous LP

In this section we introduce the continuous linear program (CLP), and we show that its value γ^*_∞ corresponds to the limit of γ^*_n when n tends to infinity. We also state Proposition 4.9, which allows us to construct feasible solutions of (CLP) using any set of times $0 < t_1 \leq t_2 \leq \cdots \leq 1$. In the remainder of the section, *finite model* refers to the **SP-UA** with $n < \infty$ candidates, while the *infinite model* refers to **SP-UA** when $n \to \infty$.

We assume $p \in (0,1]$ fixed. The continuous LP (CLP) is an infinite linear program with variables given by a function $\alpha : [0,1] \times \mathbf{N} \to [0,1]$ and a scalar $\gamma \geq 0$. Intuitively, if in the finite model we interpret $x_{t,s}$ as weights and the sums of $x_{t,s}$ over t as Riemann sums, then the limit of the finite model, the infinite model, should have a robust ratio computed by the continuous LP (CLP):

$$\sup_{\substack{\alpha:[0,1]\times\mathbf{N}\to[0,1]\\ \gamma\geq 0}} \quad \gamma$$

s.t.

$$t\alpha(t,s) \leq 1 - p\int_0^t \sum_{\sigma\geq 1} \alpha(\tau,\sigma)\,\mathrm{d}\tau \qquad \forall t\in[0,1], s\geq 1 \tag{$(CLP)_p$}$$

$$\gamma \leq \frac{p\int_0^1 \sum_{s\geq 1}\alpha(t,s)\sum_{\ell=s}^{k}\binom{\ell-1}{s-1}t^s(1-t)^{\ell-s}\,\mathrm{d}t}{(1-(1-p)^k)} \qquad \forall k\geq 1 \tag{4.11}$$

We denote by $\gamma^*_\infty = \gamma^*_\infty(p)$ the objective value of $(CLP)_p$. The following result formalizes the fact that the value of the continuous LP $(CLP)_p$ is in fact the robust ratio of the infinite model. The proof is similar to other continuous approximations [38]; a small caveat in the proof is the restriction of the finite LP to the top $(\log n)/p$ candidates, as they carry most of the weight in the objective function. The proof is deferred to Appendix 4.A.4.

Lemma 4.8. *Let γ^*_n be the optimal robust ratio for n candidates and let γ^*_∞ be the value of the continuous LP $(CLP)_p$. Then $|\gamma^*_n - \gamma^*_\infty| \leq \mathcal{O}\left((\log n)^2/(p\sqrt{n})\right)$.*

The following proposition gives a recipe to find feasible solutions for $(CLP)_p$. We use it to construct lower bounds in the following sections.

Proposition 4.9. *Consider $0 \leq t_1 \leq t_2 \leq \cdots \leq 1$ and consider the function $\alpha : [0,1]\times\mathbf{N}\to[0,1]$ defined such that for $t\in[t_i,t_{i+1})$*

$$\alpha(t,s) = \begin{cases} T_i/t^{i\cdot p+1} & s\leq i \\ 0 & s>i, \end{cases}$$

where $T_i = (t_1\cdots t_i)^p$. Then α satisfies Constraint (4.10).

Proof. We verify that inequality (4.10) holds. We define $t_0 = 0$ and $T_0 = 0$. For $t \in [t_i, t_{i+1})$ we have

$$\begin{aligned}
1 - p\int_0^t \sum_{\sigma\geq 1} \alpha(\tau,\sigma)\,\mathrm{d}\tau &= 1 - p\left(\sum_{j=0}^{j-1}\int_{t_j}^{t_{j+1}} j\frac{T_j}{\tau^{jp+1}}\,\mathrm{d}\tau\right) - p\int_{t_i}^{t} i\frac{T_i}{\tau^{jp+1}}\,\mathrm{d}\tau \\
&= 1 - p\sum_{j=0}^{i-1} j\cdot T_j\cdot\frac{1}{-pj}\left(t_{j+1}^{-jp} - t_j^{-jp}\right) - pi\cdot T_i\cdot\frac{1}{-ip}\left(t^{-ip} - t_i^{ip}\right) \\
&= 1 + \sum_{j=0}^{i-1} T_j\left(t_{j+1}^{-jp} - t_j^{-jp}\right) + T_i\left(t^{-ip} - t_i^{ip}\right) \\
&= 1 + \left(\sum_{j=0}^{i-1} T_{j+1}t_{j+1}^{-(j+1)p} - T_j t_j^{-jp}\right) + T_i\left(t^{-ip} - t_i^{-ip}\right) \\
&\qquad\qquad \text{(Since } T_{j+1}t_{j+1}^{-jp} = T_{j+1}t_{j+1}^{-(j+1)p}\text{)} \\
&= 1 + T_i t_i^{-ip} - T_0 t_0^{-p} + T_i\left(t^{-ip} - t_i^{-ip}\right) \\
&= T_i t^{-ip} \geq t\alpha(t,s)
\end{aligned}$$

for any $s \geq 1$. This concludes the proof. □

We use this result to show lower bounds for γ_∞^*. For instance, if $1/k \geq p > 1/(k+1)$ for some integer k, and we set $t_1 = 1/e$ and $t_2 = t_3 = \cdots = 1$, we can show that $\gamma_\infty^*(p)$ is at least $1/e$. Thus, in combination with Lemma 4.7, we have that $\gamma_n^*(p) \geq 1/e$ for any n and $p > 0$; we skip this analysis for brevity. In Section 4.7, we use Proposition 4.9 to show exact solutions of γ_∞^* for large p.

4.6 Upper Bounds for the Continuous LP

We now consider upper bounds for (CLP) and prove Theorem 4.3, which states that $\gamma_\infty^*(p) \leq \min\left\{p^{p/(1-p)}, 1/\beta\right\}$, for any $p \in (0,1]$, where $1/\beta \approx 0.745$ and β is the unique solution of $\int_0^1 (y(1-\log y) + \beta - 1)^{-1}\mathrm{d}y = 1$ [107].

We show that γ_∞^* is bounded by each term in the minimum operator. For the first bound,

we have

$$\gamma_n^* = \max_{\mathcal{P}} \min_{k \in [n]} \frac{\mathbf{P}(\mathcal{P} \text{ collects a top } k \text{ candidate})}{1-(1-p)^k} \leq \max_{\mathcal{P}} \frac{\mathbf{P}(\mathcal{P} \text{ collects the top candidate})}{p}.$$

The probability of collecting the highest candidate in **SP-UA** is shown by [161] to be $p^{1/(1-p)} + o(1)$, where $o(1) \to 0$ as $n \to \infty$. Thus, by Lemma 4.7, we have

$$\gamma_\infty^*(p) \leq \gamma_n^*(p) \leq p^{p/(1-p)} + o(1)/p.$$

Taking the limit $n \to \infty$, we conclude $\gamma_\infty^*(p) \leq p^{p/(1-p)}$.

For the second bound, we use the following technical result; its proof is deferred to Appendix 4.A.5, but we give a short explanation here. A γ-robust algorithm $\mathcal{A}$ for the **SP-UA**, in expectation, has pn candidates to choose from and $(1-p)n$ candidates from which the algorithm can learn about candidate quality. We give an algorithm $\mathcal{A}'$ that solves the i.i.d. prophet inequality for any $m \approx pn$ i.i.d. random variables $X_1, \ldots, X_m$, for m large. The algorithm $\mathcal{A}'$ runs a utility version of $\mathcal{A}$ in n values sampled from the distribution X_1 (see the discussion before Proposition 4.5), guaranteeing at least a factor γ of the maximum of $m \approx pn$ of these samples, which is the value of the prophet. $\mathcal{A}'$ is the capped utility version of $\mathcal{A}$, where no more than $m \approx pn$ offers can be made. Using concentration bounds, we show that the loss of these restrictions is minimal. [105] uses a similar argument, with the difference that their sampling is a fixed fraction of the input and is done in advance, while in our case the sampling is online and might deviate from the expectation, implying the need for concentration bounds.

Lemma 4.10. *Fix $p \in (0,1)$ and consider any algorithm $\mathcal{A}$ that is γ-robust for the **SP-UA** for any n. Then there is an algorithm $\mathcal{A}'$ for the i.i.d. prophet inequality problem that for*

any $\varepsilon, \delta > 0$ *satisfying*

$$(1+\varepsilon)p < 1, \qquad n \geq \frac{2}{p\varepsilon^2}\log\left(\frac{2}{\delta}\right), \qquad m = \lfloor (1+\varepsilon)pn \rfloor,$$

ensures

$$\mathbf{E}\left[\mathrm{Val}(\mathcal{A}')\right] + \delta \geq \gamma(1-4\varepsilon-\delta)\,\mathbf{E}\left[\max_{i\leq m} X_i\right],$$

for any $X_1, \ldots, X_m$ *sequence of i.i.d. random variables with support in* $[0,1]$*, where* $\mathrm{Val}(\mathcal{A}')$ *is the profit obtained by* $\mathcal{A}'$ *from the sequence of values* $X_1, \ldots, X_m$ *in the prophet problem.*

A combination of results by [98] and [107] shows that for any m and for $\varepsilon' > 0$ small enough, there is an i.i.d. instance $X_1, \ldots, X_m$ with support in $[0,1]$ such that

$$\mathbf{E}\left[\max_{i\leq m} X_i\right] \geq (a_m - \varepsilon')\sup\left\{\mathbf{E}[X_\tau] : \tau \in T_m\right\},$$

where T_m is the class of stopping times for $X_1, \ldots, X_m$, and $a_m \to \beta$. Thus, using Lemma 4.10, for any γ such that a γ-robust algorithm $\mathcal{A}$ exists for the **SP-UA**, we must have

$$\gamma(1-4\varepsilon-\delta) \leq \frac{1}{a_m - \varepsilon'} + \frac{\delta}{\mathbf{E}\left[\max_{i\leq m} X_i\right]}$$

for $m = \lfloor (1+\varepsilon)pn \rfloor$; here we set $\delta = 2e^{-pn^2/2}$. A slight reformulation of [98]'s result allows us to set $\varepsilon' = 1/m^3$ and $\mathbf{E}[\max_{i\leq m} X_i] \geq 1/m^3$ (see the discussion at the end of Appendix 4.A.5). Thus, as $n \to \infty$ we have $m \to \infty$ and so $\delta / \mathbf{E}[\max_{i\leq m} X_i] \to 0$. In the limit we obtain $\gamma(1-4\varepsilon) \leq 1/\beta$ for any $\varepsilon > 0$ such that $(1+\varepsilon)p < 1$. From here, the upper bound $\gamma \leq 1/\beta$ follows.

An algorithm that solves $(LP)_{n,p}$ and implements the policy given by the solution is γ^*_∞-robust (Theorem 4.2 and the fact that $\gamma^*_n \geq \gamma^*_\infty$) for any n. Thus, by the previous analysis and Lemma 4.7, we obtain $\gamma^*_\infty \leq 1/\beta$.

4.7 Lower Bounds for the Continuous LP

In this section we consider lower bounds for (CLP) and prove Theorem 4.4. We first give optimal solutions for large values of p. For $p \geq p^* \approx 0.594$, the optimal value of $(CLP)_p$ is $\gamma^*_\infty(p) = p^{p/(1-p)}$. We then show that for $p \leq p^*$, $\gamma^*_\infty(p) \geq (p^*)^{p^*/(1-p^*)} \approx 0.466$.

4.7.1 Exact Solution for Large p

We now show that for $p \geq p^*$, $\gamma^*_\infty(p) = p^{p/(1-p)}$, where $p^* \approx 0.594$ is the solution of $(1-p)^2 = p^{(2-p)/(1-p)}$. Thanks to the upper bound $\gamma^*_\infty(p) \leq p^{p/(1-p)}$ for any $p \in (0,1]$, it is enough to exhibit feasible solutions (α, γ) of the continuous LP $(CLP)_p$ with $\gamma \geq p^{p/(1-p)}$.

Let $t_1 = p^{1/(1-p)}$, $t_2 = t_3 = \cdots = 1$, and consider the function α defined by $t_1, t_2, \ldots$ in Proposition 4.9. That is, for $t \in [0, p^{1/(1-p)})$, $\alpha(t,s) = 0$ for any $s \geq 1$ and for $t \in [p^{1/(1-p)}, 1]$ we have

$$\alpha(t,s) = \begin{cases} p^{p/(1-p)}/t^{1+p} & s = 1 \\ 0 & s > 1. \end{cases}$$

Let $\gamma \doteq \inf_{k \geq 1} p\big(1-(1-p)^k\big)^{-1} \int_{p^{1/(1-p)}}^1 \frac{p^{p/(1-p)}}{t^{1+p}} \sum_{\ell=1}^k t(1-t)^{\ell-1} \mathrm{d}t$. Then (α, γ) is feasible for the continuous LP $(CLP)_p$, and we aim to show that $\gamma \geq p^{p/(1-p)}$ when $p \geq p^*$. The result follows by the following lemma.

Lemma 4.11. *For any $p \geq p^*$ and any $\ell \geq 0$, $\int_{p^{1/(1-p)}}^1 (1-t)^\ell t^{-p} \, \mathrm{d}t \geq (1-p)^\ell$.*

We defer the proof of this lemma to Appendix 4.A.6. Now, we have

$$\begin{aligned}\gamma &= \inf_{k\geq 1} \frac{p}{1-(1-p)^k} \int_{p^{1/1-p}}^{1} \frac{p^{p/(1-p)}}{t^{1+p}} \sum_{\ell=1}^{k} t(1-t)^{\ell-1} \mathrm{d}t \\ &= p^{p/(1-p)} \inf_{k\geq 1} \frac{\sum_{\ell=1}^{k} \int_{p^{1/(1-p)}}^{1} t^{-p}(1-t)^{\ell-1}\,\mathrm{d}t}{\sum_{\ell=1}^{k}(1-p)^{\ell-1}} \\ &\geq p^{p/(1-p)} \inf_{k\geq 1} \inf_{\ell\in[k]} \int_{0}^{1} \frac{1}{t^p} \frac{(1-t)^{\ell-1}}{(1-p)^{\ell-1}}\,\mathrm{d}t \geq p^{p/(1-p)},\end{aligned}$$

where we use the known inequality $\sum_{\ell=1}^{m} a_\ell / \sum_{\ell=1}^{m} b_\ell \geq \min_{\ell\in[m]} a_\ell/b_\ell$ for $a_\ell, b_\ell > 0$, for any ℓ, and the lemma. This shows that $\gamma_\infty^* \geq p^{p/(1-p)}$ for $p \geq p^*$.

Remark 0. Our analysis is tight. For $k = 2$, constraint

$$\frac{p}{1-(1-p)^2} \int_{p^{1/(1-p)}}^{1} \frac{p^{p/(1-p)}}{t^{1+p}} \sum_{\ell=1}^{k} t(1-t)^{\ell-1}\,\mathrm{d}t \geq p^{p/(1-p)}$$

holds if and only if $p \geq p^*$.

4.7.2 Lower Bound for Small p

We now argue that for $p \leq p^*$, $\gamma_\infty^*(p) \geq (p^*)^{p^*/(1-p^*)}$. Let $\varepsilon \in [0,1)$ satisfy $p = (1-\varepsilon)p^*$. For the argument, we take the solution α^* for $(CLP)_{p^*}$ that we obtained in the last subsection and we construct a feasible solution for $(CLP)_p$ with objective value at least $(p^*)^{p^*/(1-p^*)}$. For simplicity, we denote $\tau^* = (p^*)^{1/(1-p^*)}$.

From the previous subsection, we know that the optimal solution α^* of $(CLP)_{p^*}$ has the following form. For $t \in [0, \tau^*)$, $\alpha^*(t,s) = 0$ for any s, while for $t \in [\tau^*, 1]$ we have

$$\alpha^*(t,s) = \begin{cases} (p^*)^{p^*/(1-p^*)}/t^{p^*+1} & s = 1 \\ 0 & s > 1. \end{cases}$$

For $(CLP)_p$, we construct a solution α as follows. Let $\alpha(t,s) = \varepsilon^{s-1}\alpha^*(t,1)$ for any $t \in [0,1]$ and $s \geq 1$; for example, $\alpha(t,1) = \alpha^*(t,1)$. If we interpret α^* as a policy, it only makes offers to the highest candidate observed. By contrast, in $(CLP)_p$ the policy implied by α makes offers to more candidates (after time τ^*), with a probability geometrically decreasing according to the relative ranking of the candidate.

Claim 4.12. *The solution α satisfies constraints* (4.10)*,*

$$t\alpha(t,s) \leq 1 - p\int_0^t \sum_{\sigma\geq 1} \alpha(\tau,\sigma)\mathrm{d}\tau,$$

for any $t \in [0,1]$, $s \geq 1$.

Proof. Indeed,

$$\begin{aligned}
1 - p\int_0^t \sum_{\sigma\geq 1} \alpha(\tau,\sigma)\mathrm{d}\tau &= 1 - p^*(1-\varepsilon)\int_0^t \sum_{\sigma\geq 1} \varepsilon^{\sigma-1}\alpha^*(\tau,1)\mathrm{d}\tau \\
&= 1 - p^*\int_0^t \alpha^*(\tau,1)\mathrm{d}\tau \qquad \text{(Since } \textstyle\sum_{\sigma\geq 1}\varepsilon^{\sigma-1} = 1/(1-\varepsilon)\text{)} \\
&= 1 - p^*\int_0^t \sum_{\sigma\geq 1} \alpha^*(\tau,\sigma)\,\mathrm{d}\tau \qquad \text{(Since } \alpha^*(\tau,\sigma) = 0 \text{ for } \sigma > 1\text{)} \\
&\geq t\alpha^*(t,1). \qquad \text{(By feasibility of } \alpha^*\text{)}
\end{aligned}$$

Since $\alpha(t,s) = \varepsilon^{s-1}\alpha^*(t,1) \leq \alpha^*(t,1)$, we conclude that α satisfies (4.10) for any t and s. □

We now define $\gamma = \inf_{k\geq 1} p\big(1-(1-p)^k\big)^{-1}\int_0^1 \sum_{s\geq 1}\alpha(t,s)\sum_{\ell=s}^k \binom{\ell-1}{s-1}t^s(1-t)^{\ell-s}\mathrm{d}t$. Using the claim, we know that (α,γ) is feasible for $(CLP)_p$, and need to verify that $\gamma \geq (p^*)^{p^*/(1-p^*)}$. Similar to the analysis in the previous section, the result follows by the following claim.

Claim 4.13. *For any $\ell \geq 0$, $\int_{\tau^*}^1 (1-(1-\varepsilon)t)^\ell t^{-p^*}\,\mathrm{d}t \geq (1-p)^\ell$.*

Before proving the claim, we establish the bound:

$$\gamma = \inf_{k\geq 1} \frac{1}{\sum_{\ell=1}^k (1-p)^{\ell-1}} \int_0^1 \sum_{s=1}^k \varepsilon^{s-1}\alpha^*(s,1) \sum_{\ell=s}^k \binom{\ell-1}{s-1} t^s (1-t)^{\ell-s}\,\mathrm{d}t$$

(Using definition of α)

$$= (p^*)^{p^*/(1-p^*)} \inf_{k\geq 1} \frac{1}{\sum_{\ell=1}^k (1-p)^{\ell-1}} \int_{\tau^*}^1 \frac{1}{t^{p^*+1}} \sum_{\ell=1}^k \sum_{s=1}^\ell \varepsilon^{s-1} \binom{\ell-1}{s-1} t^s (1-t)^{\ell-s}\,\mathrm{d}t$$

(Using the definition of α^* and changing order of summmation)

$$= (p^*)^{p^*/(1-p^*)} \inf_{k\geq 1} \frac{1}{\sum_{\ell=1}^k (1-p)^{\ell-1}} \int_{\tau^*}^1 \frac{1}{t^{p^*+1}} \sum_{\ell=1}^k (1-(1-\varepsilon)t)^{\ell-1}\,\mathrm{d}t$$

(Using the binomial expansion)

$$= (p^*)^{p^*/(1-p^*)} \inf_{k\geq 1} \frac{\sum_{\ell=1}^k \int_{\tau^*}^1 t^{-p^*-1}(1-(1-\varepsilon)t)^{\ell-1}\,\mathrm{d}t}{\sum_{\ell=1}^k (1-p)^{\ell-1}} \geq (p^*)^{p^*/(1-p^*)}$$

We again used the inequality $\sum_{\ell=1}^m a_\ell / \sum_{\ell=1}^m b_\ell \geq \min_{\ell\in[m]} a_\ell/b_\ell$ for $a_\ell, b_\ell > 0$, for any ℓ, and the claim.

Proof of Claim 4.13. We have $1-(1-\varepsilon)t = (1-\varepsilon)(1-t)+\varepsilon$. Therefore,

$$\begin{aligned}
\int_{\tau^*}^{1} \frac{1}{t^{p^*}}(1-(1-\varepsilon)t)^{\ell}\mathrm{d}t &= \int_{\tau^*}^{1} \frac{1}{t^{p^*}} \sum_{j=0}^{\ell} \binom{\ell}{j} (1-\varepsilon)^{\ell-j}(1-t)^{\ell-j}\varepsilon^{\ell-j}\mathrm{d}t \\
&\qquad \text{(Binomial expansion)} \\
&= \sum_{j=0}^{\ell} \binom{\ell}{j} (1-\varepsilon)^{\ell-j}\varepsilon^{j} \int_{\tau^*}^{1} \frac{1}{t^{p^*}}(1-t)^{\ell-j}\,\mathrm{d}t \\
&\geq \sum_{j=0}^{\ell} \binom{\ell}{j} (1-\varepsilon)^{\ell-j}\varepsilon^{j}(1-p^*)^{\ell-j}\,\mathrm{d}t \\
&\qquad \text{(Using Lemma 4.11 for } p^*\text{)} \\
&= (\varepsilon + (1-\varepsilon)(1-p^*))^{\ell} \qquad \text{(Using binomial expansion)} \\
&= (1-(1-\varepsilon)p^*)^{\ell} = (1-p)^{\ell},
\end{aligned}$$

where we used $p = (1-\varepsilon)p^*$. From this inequality the claim follows. □

4.8 Computational Experiments

In this section we aim to empirically test our policy; to do so, we focus on utility models. Recall from Proposition 4.5 that a γ-robust policy ensures at least γ fraction of the optimal offline utility, for any utility functions that is related to the ranking, i.e., $U_j < U_i$ if and only if $i \prec j$. This is advantageous for practical scenarios, where a candidate's "value" may be unknown to the decision maker.

We evaluate the performance of two groups of solutions. The first group includes policies that are computed without the knowledge of any utility function:

- Robust policy (**Rob-Pol**(n,p)), corresponds to the optimal policy obtained by solving $(LP)_{n,p}$.
- Tamaki's policy (**Tama-Pol**(n,p)), that maximizes the probability of selecting success-

fully the best candidate willing to accept an offer. To be precise, [164] studies two models of availability: MODEL 1, where the availability of the candidate is known after an offer has been made; and MODEL 2, where the availability of the candidate is known upon the candidate's arrival. MODEL 2 has higher values and it is computationally less expensive to compute; we use this policy. Note that in **SP-UA**, the expected value obtained by learning the availability of the candidate after making an offer is the same value obtained in the model that learns the availability right upon arrival. Therefore, MODEL 2 is a better model to compare our solutions than MODEL 1.

In the other group, we have policies that are computed with knowledge of the utility function.

- The expected optimal offline value ($\mathbf{E}[U(\textsc{Opt}(U,n,p))]$), which knows the outcome of the offers and the utility function. It can be computed via $\sum_{i=1}^{n} U_i p(1-p)^{i-1}$. For simplicity, we write OPT when the parameters are clear from the context.

- The optimal rank-based policy if the utility function is known in advance, (**Util-Pol**(U,n,p)), computed by solving the optimality equation

$$v_{(t,s)} = \max\left\{ U_t(s) + \frac{1-p}{t+1}\sum_{\sigma=1}^{t+1} v_{(t+1,\sigma)}, \frac{1}{t+1}\sum_{\sigma=1}^{t+1} v_{(t+1,\sigma)}, \right\},$$

  with boundary condition $v_{(n+1,\sigma)} = 0$ for any σ. We write **Util-Pol**(n,p) when U is clear from the context. We use a rank-based policy as opposed to a value-based policy for computational efficiency.

Note that $\mathbf{E}[U(\textbf{Rob-Pol})], \mathbf{E}[U(\textbf{Tama-Pol})] \leq \mathbf{E}[U(\textbf{Util-Pol})] \leq \mathbf{E}[U(\textsc{Opt})]$ and by Proposition 4.5, $E[U(\textbf{Rob-Pol})] \geq \gamma_n^* \mathbf{E}[U(\mathcal{A})]$ for any $\mathcal{A}$ of the aforementioned policies.

We consider the following decreasing utility functions:

- **Top k candidates are valuable (top-k)**. For $k \in [n]$, we consider utility functions of the form $U_i = 1 + \varepsilon^i$ for $i \in [k]$ and $U_i = \varepsilon^i$ for $i > k$ with $\varepsilon = 1/n$. Intuitively, we aim to capture the notion of an elite set of candidates, where candidates outside the top k are not nearly as appealing to the decision maker. For instance, renowned brands like to target certain members of a population for their ads. We test $k = 1, 2, 3, 4$.

- **Power law population**. $U_i = i^{-1/(1+\delta)}$ for $i \geq 1$ and small $\delta > 0$. Empirical studies have shown that the distribution of individual performances in many areas follows a power law or Pareto distribution [43]. If we select a random person from $[n]$, the probability that this individual has a performance score of at least t is proportional to $t^{-(1+\delta)}$. We test $\delta \in \{10^{-2}, 10^{-1}, 2 \cdot 10^{-1}\}$.

We run experiments for $n = 200$ candidates and range the probability of acceptance p from $p = 10^{-2}$ to $p = 9 \cdot 10^{-1}$.

4.8.1 Results for Top-k Utility Function

In this subsection, we present the results for utility function that has largest values in the top k candidates, where $k = 1, 2, 3, 4$. In Figure 4.3, we plot the ratio between the value collected by $\mathcal{A}$ and $\mathbf{E}[U(\textsc{Opt})]$, for $\mathcal{A}$ being **Util-Pol**, **Rob-Pol** and **Tama-Pol**.

Naturally, of all sequential policies, **Util-Pol** attains the largest approximation factor of $\mathbf{E}[U(\textsc{Opt})]$. We observe empirically that **Rob-Pol** collects larger values than **Tama-Pol** for smaller values of k. Interestingly, we observe in the four experiments that the approximation factor for **Rob-Pol** is always better than **Tama-Pol** for small values of p. In other words, robustness helps online selection problems when the probability of acceptance is relatively low. In general, for this utility function, we observe in the experiments that **Rob-Pol** collects at least 50% of the optimal offline value, except for the case $k = 1$. As n increases (not shown in the figures), we observe that the approximation factors of all three policies decrease; this is consistent with the fact that γ_n^*, the optimal robust ratio, is

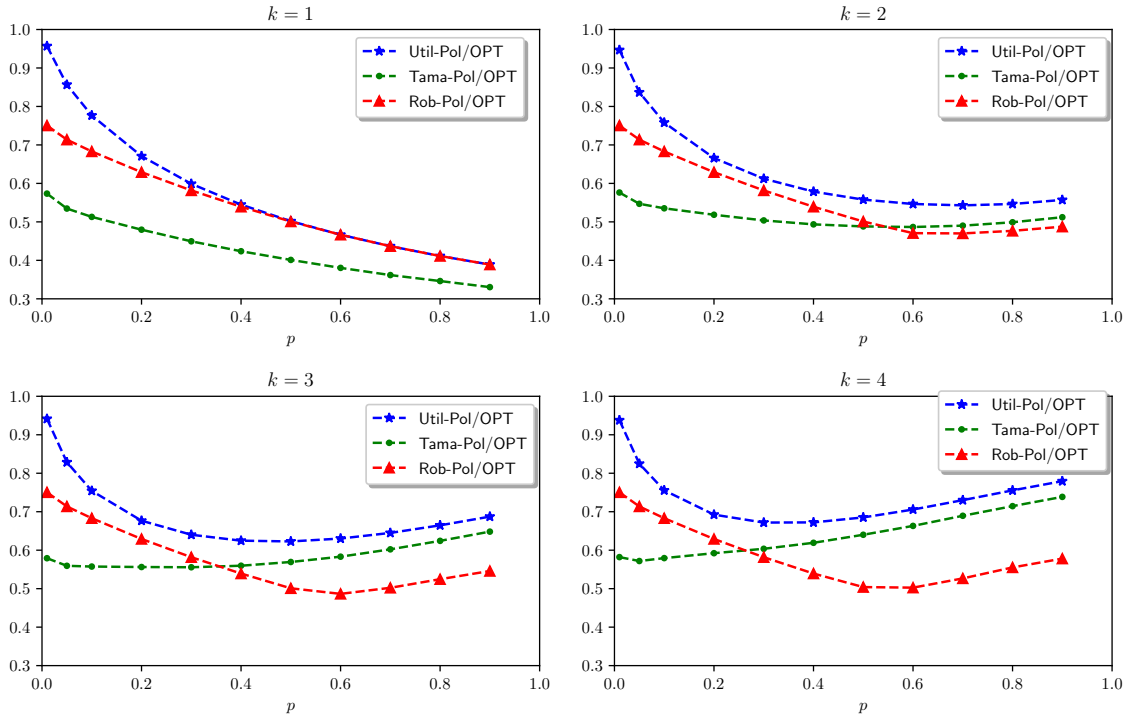

Figure 4.3: Approximation factors for the top k utility function, for $k = 1, 2, 3, 4$.

decreasing in n.

4.8.2 Results for Power Law Utility Function

In this subsection, we present the result of our experiments for the power law utility function $U_i = i^{-(1+\delta)}$ for $\delta = 10^{-2}, 10^{-1}$ and $2 \cdot 10^{-1}$. In Figure 4.4, we display the approximation factors of the three sequential policies.

Again, we note that **Util-Pol** collects the largest fraction of all sequential policies. We also observe a similar behavior as in the case of the top-k utility function. For small values of p, **Rob-Pol** empirically collects more value than **Tama-Pol**. As p increases, the largest valued candidate is more willing to accept an offer; hence, [164]'s policy is able to capture that candidate.

In general, our experiments suggests that **Rob-Pol** is better than **Tama-Pol** for smaller values of p. This may be of interest in applications where the probability of acceptance

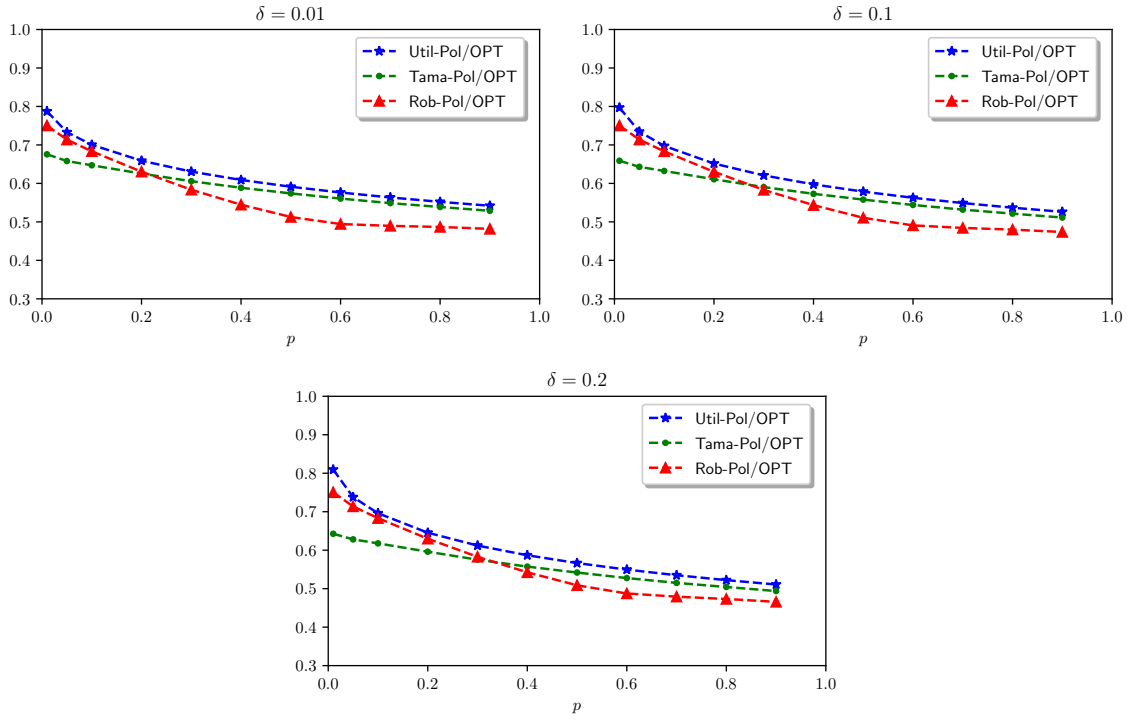

Figure 4.4: Approximation factor for the power law utility function. The function has the form $U_i = i^{-(1+\delta)}$. Experiments are run for $\delta \in \{10^{-2}, 10^{-1}, 2 \cdot 10^{-1}\}$.

is small, say 20% or less. For instance, some sources state that click-through rates (the fraction of time that an ad is clicked on) are typically less than 1% [71]. Therefore, ad display policies based on **Rob-Pol** may be more appropriate than other alternatives.

4.9 Concluding Remarks

We have studied the **SP-UA**, which models an online selection problem where candidates can reject an offer. We introduced the robust ratio as a metric that tries to simultaneously maximize the probability of successfully selecting one of the best k candidates given that at least one of these will accept an offer, for all values of k. This objective captures the worst-case scenario for an online policy against an offline adversary that knows in advance which candidates will accept an offer. We also demonstrated a connection between this robust ratio and online selection with utility functions. We presented a framework based on MDP theory to derive a linear program that computes the optimal robust ratio and its

optimal policy. This framework can be generalized and used in other secretary problems (Section 4.4.1), for instance, by augmenting the state space. Furthermore, using the MDP framework, we were able to show that the robust ratio γ_n^* is a decreasing function in n. This enabled us to make connections between early works in secretary problems and recent advances. To study our LP, we allow the number of candidates to go to infinity and obtain a continuous LP. We provide bounds for this continuous LP, and optimal solutions for large p.

We empirically observe that the robust ratio $\gamma_n^*(p)$ is convex and decreasing as a function of p, and thus we expect the same behavior from $\gamma_\infty(p)$, though this remains to be proved (see Figure 4.1). Based on numerical values obtained by solving $(LP)_{n,p}$, we conjecture that $\lim_{p\to 0} \gamma_\infty^*(p) = 1/\beta \approx 0.745$. This limit is also observed in a similar model [50], where a fraction of the input is given in advance to the decision maker as a sample. In our model, if we interpret the rejection from a candidate as a sample, then in the limit both models might behave similarly. Numerical comparisons between our policies and benchmarks suggest that our proposed policies perform especially well in situations where the probability of acceptance is small, say less than 20%, as in the case of online advertisement.

A natural extension is the value-based model, where candidates reveal numerical values instead of partial rankings. Our algorithms are rank-based and guarantee an expected value at least a fraction $\gamma_n^*(p)$ of the optimal offline expected value (Proposition 4.5). Nonetheless, algorithms based on numerical values may attain higher expected values than the ones guaranteed by our algorithm. In fact, a threshold algorithm based on sampling may perhaps be enough to guarantee better values, although this this requires an instance-dependent approach. The policies we consider are instance-agnostic, can be computed once and used for any input sequence of values. In this value-based model, we would like to consider other arrivals processes. A popular arrival model is the adversarial arrival, where an adversary constructs values and the arrival order in response to the DM's algorithm. Unfortunately,

a construction similar to the one in [128] for the online knapsack problem shows that it is impossible to attain a finite competitive ratio in an adversarial regime.

Customers belonging to different demographic groups may have different willingness to click on ads [39]. In this work, we considered a uniform probability of acceptance, and our techniques do not apply directly in the case of different probabilities. In ad display, one way to cope with different probabilities depending on customers' demographic group is the following. Upon observing a customer, a random variable (independent of the ranking of the candidate) signals the group of the customer. The probability of acceptance of a candidate depends on the candidate's group. Assuming independence between the rankings and the demographic group allows us to learn nothing about the global quality of the candidates beyond what we can learn from the partial rank. Using the framework presented in this work, with an augmented state space (time, partial rank, group type), we can write an LP that solves this problem exactly. Nevertheless, understanding the robust ratio in this new setting and providing a closed-form policy are still open questions.

Another interesting extension is the case of multiple selections. In practice, platforms can display the same ad to more than one user, and some job posts require more than one person for a position. In this setting, the robust ratio is less informative. If k is the number of possible selections, one possible objective is to maximize the number of top k candidates selected. We can apply the framework from this work to obtain an optimal LP. Although there is an optimal solution, no simple closed-form strategies have been found even for $p = 1$; see e.g. [34]).

4.A Appendix

4.A.1 Missing Proofs From Section 4.3

Proof of Proposition 4.5. Let ALG be a γ-robust algorithm. Fix any algorithm ALG and any $U_1 \geq \cdots \geq U_n \geq 0$. Let $\varepsilon > 0$ and let $\widehat{U}_i = U_i + \varepsilon^i$. Thus $\widehat{U}_i > \widehat{U}_{i+1}$ and so rank and

utility are in one-to-one correspondence. Then

$$\frac{\mathbf{E}[\widehat{U}(\textsc{Alg})]}{\mathbf{E}[\widehat{U}(\textsc{Opt})]} \geq \min_{x \in \{\widehat{U}_1,\ldots,\widehat{U}_n\}} \frac{\mathbf{P}(\widehat{U}(\textsc{Alg}) \geq x)}{\mathbf{P}(\widehat{U}(\textsc{Opt}) \geq x)} = \min_{k \leq n} \frac{\mathbf{P}(\textsc{Alg} \text{ collects a top } k \text{ candidate})}{\mathbf{P}(\text{A top } k \text{ candidate accepts})} \geq \gamma$$

where we used the fact that ALG is γ-robust. Notice that $\mathbf{E}[U(OPT)] \leq \mathbf{E}[\widehat{U}(OPT)]$, and also $\mathbf{E}[\widehat{U}(\textsc{Alg})] \leq \mathbf{E}[U(\textsc{Alg})] + \varepsilon$. Thus doing $\varepsilon \to 0$ we obtain

$$\frac{\mathbf{E}[U(\textsc{Alg})]}{\mathbf{E}[U(\textsc{Opt})]} \geq \gamma \tag{4.12}$$

for any nonzero vector U with $U_1 \geq U_2 \geq \cdots \geq U_n \geq 0$. This finishes the first part. For the second part, let

$$\overline{\gamma}_n = \min_{\textsc{Alg}} \max \left\{ \frac{\mathbf{E}[U(\textsc{Alg})]}{\mathbf{E}[U(\textsc{Opt})]} : U : [n] \to \mathbf{R}_+, U_1 \geq \cdots \geq U_n \right\}.$$

Note that the RHS of Inequality (4.12) is independent of U, thus minimizing in U in the LHS and then maximizing in ALG on both sides we obtain $\overline{\gamma}_n \geq \gamma_n^*$.

To show the reverse inequality, fix $k \in [n]$ and let $\widehat{U} : [n] \to \mathbf{R}_+$ given by $\widehat{U}_i = 1$ for $i \leq k$ and $\widehat{U}_i = 0$ for $i > k$. Then,

$$\frac{\mathbf{P}(\textsc{Alg} \text{ collects a top } k \text{ candidate})}{\mathbf{P}(\text{A top } k \text{ candidate accepts})} = \frac{\mathbf{E}[\widehat{U}(\textsc{Alg})]}{\mathbf{E}[\widehat{U}(\textsc{Opt})]} \geq \min_{\substack{U:[n]\to\mathbf{R}_+ \\ U_1 \geq \cdots \geq U_n}} \frac{\mathbf{E}[U(\textsc{Alg})]}{\mathbf{E}[U(\textsc{Opt})]}.$$

This bound holds for any k, thus minimizing over k and then maximizing over ALG on both sides, we obtain $\gamma_n^* \geq \overline{\gamma}_n$, which finishes the second part. □

4.A.2 Missing Proofs From Section 4.4

Here we present a detailed derivation of Theorem 4.1 and Theorem 4.2 by revisiting Section 4.4.

As stated in Section 4.4, we are going to proceed in two stages: (1) First, using a generic utility function, we uncover the space of policies POL. (2) Second, we show that our objective is a concave linear function of the variables of the space of policies that allows us to optimize it over POL.

Stage 1: The Space of Policies

Let $U : [n] \to \mathbf{R}_+$ be an arbitrary utility function and suppose that a DM makes decisions based on partial rankings and her goal is to maximize the utility obtained, where she gets U_i if she is able to collect a candidate of ranking i. Let $v^*_{(t,s)}$ be the optimal value obtained by a DM in $\{t, t+1, \ldots, n\}$ if she is currently observing the t-th candidate and this candidate has partial ranking $r_t = s$, i.e., the current state of the system is $s_t = (t, s)$. The function $v^*_{(t,s)}$ is called the *value function*. We define $v^*_{(n+1,\sigma)}$ for any $\sigma \in [n+1]$. Then, by optimality equations [146], we must have

$$v^*_{(t,s)} = \max \left\{ pU_t(s) + (1-p)\frac{1}{t+1}\sum_{\sigma=1}^{t+1} v^*_{(t+1,\sigma)}, \frac{1}{t+1}\sum_{\sigma=1}^{t+1} v^*_{(t+1,\sigma)} \right\} \tag{4.13}$$

where the first part in the max operator corresponds to the expected value obtained by selecting the current candidate, while the second part in the operator corresponds to the expected value collected by passing to the next candidate. Here $U_t(s) = \sum_{i=1}^{n} U_i \, \mathbf{P}(R_t = i \mid r_t = s)$ is the expected value collected by the DM given that the current candidate has partial ranking $r_t = s$ and accepts the offer. Although an optimal policy for this arbitrary utility function can be computed via the optimality equations, we are more interested in all the possible policies that can be obtained via these formulations. For this, we are going to use linear programming. This has been used in MDP theory [146, 14] to study the space of policies.

The following proposition shows that the solution of the optimality equations (4.13) solves the LP (D) in Figure 4.5. We denote by $v_{(D)}$ the value of the LP (D).

$$
\begin{array}{ll|ll}
(D) & \min\limits_{v\geq 0} \quad v_{(1,1)} & (P) & \max\limits_{\mathbf{x},\mathbf{y}\geq 0} \quad \sum\limits_{t=1}^{n}\sum\limits_{s=1}^{t} U_t(s)x_{t,s} \\
\text{s.t.} & & \text{s.t.} & \\
& v_{(t,s)} \geq pU_t(s) + \dfrac{1-p}{t+1}\sum\limits_{\sigma=1}^{t+1} v_{(t+1,\sigma)} & & x_{1,1} + y_{1,1} \leq 1 \qquad (4.16) \\
& \qquad \forall t \in [n], s \in [t] \qquad (4.14) & & x_{t,s} + y_{t,s} \leq \dfrac{1}{t}\left(\sum\limits_{\sigma=1}^{t-1} y_{t-1,\sigma} + (1-p)x_{t-1,\sigma}\right) \\
& v_{(t,s)} \geq \dfrac{1}{t+1}\sum\limits_{\sigma=1}^{t+1} v_{(t+1,\sigma)} & & \qquad \forall t \in [n], s \in [t] \qquad (4.17) \\
& \qquad \forall t \in [n], s \in [t] \qquad (4.15) & &
\end{array}
$$

Figure 4.5: Linear program that finds value function v^* for **SP-UA** and its dual.

Proposition 4.14. *Let $v^* = (v^*_{(t,s)})_{t,s}$ be a solution of* (4.13)*, then v^* is an optimal solution of the problem of (D) in Figure 4.5.*

Proof. Since v^* satisfies the optimality equation (4.13) then it clearly satisfies constraints (4.14) and (4.15). Thus, v^* is feasible and so $v^*_{(1,1)} \geq v_{(D)}$.

To show the optimality of v^*, we show that any solution $\overline{v}$ of the LP is an upper bound for the value function: $v^* \leq \overline{v}$. To show this, we proceed by backward induction in $t = n+1, n, \ldots, 1$ and we prove that $v^*_{(t,s)} \leq \overline{v}_{(t,s)}$ for any $s \in [t]$.

We start with the case $t = n+1$. In this case $v^*_{(n+1,s)} = 0$ for any s and since $\overline{v}(n+1,s) \geq 0$ for any s, then the result follows.

Suppose the result is true for $t = \tau+1, \ldots, n+1$ and let us show it for $t = \tau$. Using Constraints (4.14)-(4.15) we must have

$$
\begin{aligned}
\overline{v}_{(\tau,s)} &\geq \max\left\{ pU_\tau(s) + (1-p)\frac{1}{\tau+1}\sum_{\sigma=1}^{\tau+1}\overline{v}_{(\tau+1,\sigma)}, \frac{1}{\tau+1}\sum_{\sigma=1}^{\tau+1}\overline{v}_{(\tau+1,\sigma)} \right\} \\
&\geq \max\left\{ pU_\tau(s) + (1-p)\frac{1}{\tau+1}\sum_{\sigma=1}^{\tau+1} v^*_{(\tau+1,\sigma)}, \frac{1}{\tau+1}\sum_{\sigma=1}^{\tau+1} v^*_{(\tau+1,\sigma)} \right\} \\
&\qquad\qquad \text{(backward induction)} \\
&= v^*_{(\tau,s)}
\end{aligned}
$$

where the last line follows by the optimality equations (4.13). Thus, $v_{(D)} = \overline{v}_{(1,1)} \geq v^*_{(1,1)}$. □

The dual of the LP (D) is depicted in Figure 4.5 and named (P). The crucial fact to notice here is that the feasible region of (P) is oblivious of the utility function (or rewards) given initially to the MDP. This suggest that the region

$$\text{POL} = \left\{ (\mathbf{x}, \mathbf{y}) \geq 0 : x_{1,1} + y_{1,1} = 1, x_{t,s} + y_{t,s} = \frac{1}{t} \sum_{\sigma=1}^{t-1} \left(y_{t-1,\sigma} + (1-p) x_{t-1,\sigma} \right), \forall t \in [n], s \in [t] \right\}$$

codifies all possible policies. The following two propositions formalize this.

Proposition 4.15. *For any policy $\mathcal{P}$ for the **SP-UA**, consider*

$$x_{t,s} = \mathbf{P}(\mathcal{P} \textit{ reaches state } (t,s), \textit{ selects candidate})$$

and

$$y_{t,s} = \mathbf{P}(\mathcal{P} \textit{ reaches state } (t,s), \textit{ does not select candidate}).$$

Then $(\mathbf{x}, \mathbf{y})$ belongs to POL.

Proof. Consider the event $D_t = \{t\text{-th candidate turns down offer}\}$. Then $p = \mathbf{P}(D_t)$. Consider also the events

$$O_t = \{\mathcal{P} \text{ reaches } t\text{-th candidate and extends an offer}\}$$

and

$$\overline{O}_t = \{\mathcal{P} \text{ reaches } t\text{-th candidate and does not extend offer}\}.$$

Then O_t and $\overline{O}_t$ are disjoint events and $O_t \cup \overline{O}_t$ equals the event of $\mathcal{P}$ reaching stage t. Thus

$$\mathbf{1}_{O_t \cap \{S_t=(t,s)\}} + \mathbf{1}_{\overline{O}_t \cap \{S_t=(t,s)\}} = \mathbf{1}_{\{\mathcal{P} \text{ reaches state } S_t=(t,s)\}}. \tag{4.18}$$

Note that $x_{t,s} = \mathbf{P}(O_t \cap \{S_t = (t,s)\})$ and $y_{t,s} = \mathbf{P}(\overline{O}_t \cap \{S_t = (t,s)\})$. For $t = 1$, then $S_1 = (1,1)$ and $\{\mathcal{P}$ reaches state $S_1 = (1,1)\}$ occurs with probability 1. Thus

$$x_{1,1} + y_{1,1} = 1.$$

For $t > 1$, by the dynamics of the system, the only way that $\mathcal{P}$ reaches state t is by reaching stage $t-1$ and not extending an offer to the $t-1$ candidate or extending an offer but this was turned down. Thus,

$$\begin{aligned}
&\mathbf{1}_{\{\mathcal{P} \text{ reaches state } S_t=(t,s)\}} \\
&= \sum_{\sigma=1}^{t-1} \mathbf{1}_{\overline{O}_{t-1} \cap \{S_{t-1}=(t-1,\sigma)\} \cap \{S_t=(t,s)\}} + \mathbf{1}_{O_{t-1} \cap \{S_{t-1}=(t-1,\sigma)\} \cap \overline{D}_{t-1} \cap \{S_t=(t,s)\}}
\end{aligned} \tag{4.19}$$

Note that

$$\begin{aligned}
&\mathbf{P}(\overline{O}_{t-1} \cap \{S_{t-1} = (t-1,\sigma)\} \cap \{S_t = (t,s)\}) \\
&= \mathbf{P}(\overline{O}_{t-1} \cap \{S_{t-1} = (t-1,\sigma)\} \mid S_t = (t,s))\, \mathbf{P}(S_t = (t,s)) \\
&= \mathbf{P}(\overline{O}_{t-1} \cap \{S_{t-1} = (t-1,\sigma)\}) \frac{1}{t} \\
&= \frac{1}{t} y_{t-1,\sigma}.
\end{aligned}$$

Note that we use that $\mathcal{P}$'s action at stage $t-1$ only depends on S_{t-1} and not what is observed

in the future. Likewise we obtain

$$\begin{aligned}
&\mathbf{P}(O_{t-1} \cap \{S_{t-1} = (t-1,\sigma)\} \cap \overline{D}_{t-1} \cap \{S_t = (t,s)\}) \\
&= \mathbf{P}(\overline{D}_{t-1})\, \mathbf{P}(O_{t-1} \cap \{S_{t-1} = (t-1,\sigma)\} \cap \{S_t = (t,s)\}) \\
&= (1-p)\, \mathbf{P}(O_{t-1} \cap \{S_{t-1} = (t-1,\sigma)\} \cap \{S_t = (t,s)\}) \\
&= \frac{1-p}{t} x_{t-1,\sigma}.
\end{aligned}$$

Using the equality between (4.18) and (4.19) and taking expectation, we obtain

$$x_{t,s} + y_{t,s} = \sum_{\sigma=1}^{t-1} \frac{1}{t} y_{t-1,\sigma} + \frac{1-p}{t} x_{t-1,\sigma}$$

which shows that $(\mathbf{x}, \mathbf{y}) \in$ POL. □

Conversely

Proposition 4.16. *Let* $(\mathbf{x}, \mathbf{y})$ *be a point in* POL. *Consider the (randomized) policy* $\mathcal{P}$ *that in state* (t,s) *makes an offer to the candidate* t *with probability* $x_{1,1}$ *if* $t = 1$ *and*

$$\frac{t x_{t,s}}{\sum_{\sigma=1}^{t-1} y_{t-1,\sigma} + (1-p) x_{t-1,\sigma}},$$

if $t > 1$. *Then* $\mathcal{P}$ *is a policy for* ***SP-UA*** *such that* $x_{t,s} = \mathbf{P}(\mathcal{P}$ *reaches state* (t,s)*, selects candidate*$)$ *and* $y_{t,s} = \mathbf{P}(\mathcal{P}$ *reaches state* (t,s)*, does not select candidate*$)$ *for any* $t \in [n]$ *and* $s \in [t]$.

Proof. We use the same events $O_t, \overline{O}_t$ and D_t as defined in the previous proof. Thus, we need to show that $x_{t,s} = \mathbf{P}(O_t \cap \{S_t = (t,s)\})$ and $y_{t,s} = \mathbf{P}(\overline{O}_t \cap \{S_t = (t,s)\})$ are the right marginal probabilities. For this, its is enough to show that

$$\mathbf{P}(O_t \mid S_t = (t,s)) = t x_{t,s} \quad \text{and} \quad \mathbf{P}(\overline{O}_t \mid S_t = (t,s)) = t y_{t,s}$$

for any $t \in [n]$ and for any $s \in [t]$. We prove this by induction in $t \in [n]$. For $t = 1$, the result is true by definition of the acceptance probability and the fact that $x_{1,1} + y_{1,1} = 1$. Let us assume the result is true for $t-1$ and let us show it for t. First we have

$$\begin{aligned}
\mathbf{P}(O_t \mid S_t = (t,s)) &= \mathbf{P}(\text{Reach } t, \mathcal{P}(S_t) = \textbf{offer} \mid S_t = (t,s)) \\
&= \mathbf{P}(\text{Reach } t \mid S_t = (t,s))\, \mathbf{P}(\mathcal{P}(S_t) = \textbf{offer} \mid S_t = (t,s)) \\
&= \mathbf{P}(\text{Reach } t \mid S_t = (t,s)) \cdot \frac{t x_{t,s}}{\left(\sum_{\sigma=1}^{t-1} y_{t-1,\sigma} + (1-p) x_{t-1,s}\right)}
\end{aligned}$$

Now, we have

$$\begin{aligned}
&\mathbf{P}(\text{Reach } t \mid S_t = (t,s)) \\
&= \mathbf{P}((O_{t-1} \cap \overline{D}_{t-1}) \cup \overline{O}_{t-1} \mid S_t = (t,s)) \\
&= (1-p) \sum_{\sigma=1}^{t-1} \mathbf{P}(O_{t-1} \mid S_{t-1} = (t-1,\sigma), S_t = (t,s))\, \mathbf{P}(S_{t-1} = (t-1,\sigma) \mid S_t = (t,s)) \\
&\qquad + \sum_{\sigma=1}^{t-1} \mathbf{P}(\overline{O}_{t-1} \mid S_{t-1} = (t-1,\sigma), S_t = (t,s))\, \mathbf{P}(S_{t-1} = (t-1,\sigma) \mid S_t = (t,s)) \\
&= (1-p) \sum_{\sigma=1}^{t-1} \mathbf{P}(O_{t-1} \mid S_{t-1} = (t-1,\sigma)) \frac{1}{t-1} \\
&\qquad + \sum_{\sigma=1}^{t-1} \mathbf{P}(\overline{O}_{t-1} \mid S_{t-1} = (t-1,\sigma)) \frac{1}{t-1} \\
&\qquad\qquad (\mathcal{P} \text{ only makes decisions at stage } t-1 \text{ based on } S_{t-1}) \\
&= (1-p) \sum_{\sigma=1}^{t-1} x_{t-1,\sigma} + y_{t-1,\sigma} \qquad \text{(induction)}
\end{aligned}$$

Note that we used

$$\begin{aligned}
\mathbf{P}(S_{t-1} = (t-1,\sigma) \mid S_t = (t,s)) &= \frac{\mathbf{P}(S_t = (t,s) \mid S_{t-1} = (t-1,\sigma))\, \mathbf{P}(S_{t-1} = (t-1,\sigma))}{\mathbf{P}(S_t = (t,s))} \\
&= \frac{1}{t-1}.
\end{aligned}$$

Thus, the induction holds for $\mathbf{P}(O_t \mid S_t = (t,s)) = tx_{t,s}$. Similarly, for

$$\begin{aligned}
\mathbf{P}(\overline{O}_t \mid S_t = (t,s)) &= \mathbf{P}(\text{Reach } t, \mathcal{P}(S_t) = \textbf{pass} \mid S_t = (t,s)) \\
&= \mathbf{P}(\text{Reach } t \mid S_t = (t,s))\, \mathbf{P}(\mathcal{P}(S_t) = \textbf{pass} \mid S_t = (t,s)) \\
&= \left(\sum_{\sigma=1}^{t-1} y_{t-1,\sigma} + (1-p)x_{t-1,\sigma}\right)\left(1 - \frac{tx_{t,s}}{\left(\sum_{\sigma=1}^{t-1} y_{t-1,\sigma} + (1-p)x_{t-1,s}\right)}\right) \\
&= ty_{t,s}
\end{aligned}$$

where we used the fact that $(\mathbf{x}, \mathbf{y}) \in \text{POL}$. □

Stage 2: The robust objective

Proposition 4.17. *Let $\mathcal{P}$ be any policy for **SP-UA** and let $(\mathbf{x}, \mathbf{y}) \in \text{POL}$ be its corresponding vector as in Proposition 4.15. Then, for any $k \in [n]$,*

$$\mathbf{P}(\mathcal{P} \textit{ collects a top } k \textit{ candidate}) = \sum_{t=1}^{n} \sum_{s=1}^{t} px_{t,s}\, \mathbf{P}(R_t \leq k \mid r_t = s).$$

Proof. We use the same events as in the proof of Proposition 4.15. Then,

$$\begin{aligned}
\mathbf{P}(\mathcal{P} \text{ collects a top } k \text{ candidate}) &= \sum_{t=1}^{n} \sum_{s=1}^{t} \mathbf{P}(O_t \cap D_t \cap \{S_t = (t,s)\} \cap \{R_t \leq k\}) \\
&= \sum_{t=1}^{n} \sum_{s=1}^{t} px_{t,s}\, \mathbf{P}(R_t \leq k \mid O_t \cap D_t \cap \{S_t = (t,s)\}) \\
&= \sum_{t=1}^{n} \sum_{s=1}^{t} px_{t,s}\, \mathbf{P}(R_t \leq k \mid r_t = s)
\end{aligned}$$

Note that R_t only depends on S_t and $S_t = (t,s)$ is equivalent to $r_t = s$. □

We are ready to prove of Theorem 4.2.

Theorem 4.18 (Theorem 4.2 restated)**.** *The largest robust ratio γ_n^* corresponds to the op-*

timal value of the LP

$$
\begin{aligned}
&\max_{\mathbf{x}\geq 0} && \gamma \\
&\textit{s.t.} \\
(LP)_{n,p} \qquad & x_{t,s} \leq \frac{1}{t}\left(1 - p\sum_{\tau=1}^{t-1}\sum_{\sigma=1}^{\tau} x_{\tau,\sigma}\right) && \forall t \in [n], s \in [t] \\
& \gamma \leq \frac{p}{1-(1-p)^k}\sum_{t=1}^{n}\sum_{s=1}^{t} x_{t,s}\,\mathbf{P}(R_t \leq k \mid r_t = s) && \forall k \in [n],
\end{aligned}
$$

Moreover, given an optimal solution $(\mathbf{x}^*, \gamma_n^*)$ of $(LP)_{n,p}$, the (randomized) policy $\mathcal{P}^*$ that at state (t,s) makes an offer with probability $tx_{t,s}^*/\big(1 - p\sum_{\tau=1}^{t-1}\sum_{\sigma=1}^{\tau} x_{\tau,\sigma}^*\big)$ is γ_n^*-robust.

Proof. We have

$$
\begin{aligned}
\gamma_n^* &= \max_{\mathcal{P}} \min_{k\in[n]} \frac{\mathbf{P}(\mathcal{P} \text{ collects a candidate with rank} \leq k)}{1-(1-p)^k} \\
&= \max_{(x,y)\in \textsc{Pol}} \min_{k\in[n]} \frac{p\sum_{t=1}^{n}\sum_{s=1}^{t} x_{t,s}\,\mathbf{P}(R_t \leq k \mid r_t = s)}{1-(1-p)^k}
\end{aligned}
$$

(Propositions 4.15, 4.16 and 4.17)

Now note that the function $\gamma : (\mathbf{x}, \mathbf{y}) \mapsto \min_{k\in[n]} \frac{p\sum_{t=1}^{n}\sum_{s=1}^{t} x_{t,s}\,\mathbf{P}(R_t \leq k | r_t = s)}{1-(1-p)^k}$ is constant in $\mathbf{y}$. Thus any point $(\mathbf{x}, \mathbf{y})$ satisfying Constraints (4.16)-(4.17) has an equivalent point in $(\mathbf{x}', \mathbf{y}') \in \textsc{Pol}$ with $\mathbf{x}' = \mathbf{x}$, $\mathbf{y}' \geq \mathbf{y}$ so all constraints tighten and the objective of γ is the same for both points. Thus, γ_n^* equals the optimal value of the LP (P'):

$$
\begin{aligned}
\max_{\mathbf{x}\geq 0} \quad & \gamma \\
\text{s.t.} \quad & \\
(P') \qquad & x_{1,1} + y_{1,1} \leq 1 \\
& x_{t,s} + y_{t,s} \leq \frac{1}{t}\left(\sum_{\sigma=1}^{t-1} y_{t-1,\sigma} + (1-p)x_{t-1,\sigma}\right) && \forall t \in [n], s \in [t] \\
& \gamma \leq \frac{p\sum_{t=1}^{n}\sum_{s=1}^{t} x_{t,s}\,\mathbf{P}(R_t \leq k \mid r_t = s)}{1-(1-p)^k} && \forall k \in [n]
\end{aligned}
$$

where we linearized the objective with the variable γ. By projecting the feasible region of (P') onto the variables $(\mathbf{x}, \gamma)$ we obtain $(LP)_{n,p}$. This is a routine procedure that can be carried out using Fourier-Motzkin [154] but we skip it here for brevity.

For the second part, we can take an optimal solution $(\mathbf{x}^*, \gamma_n^*)$ and its corresponding point $(\mathbf{x}^*, \mathbf{y}^*) \in \text{POL}$. A routine calculation shows that $1 - p\sum_{\tau<t}\sum_{\sigma=1}^{\tau} x_{\tau,\sigma}^* = \sum_{\sigma=1}^{t-1} y_{t-1,\sigma}^* + (1-p)x_{t-1,\sigma}^*$. Thus by Proposition 4.16 we obtain the optimal policy. □

4.A.3 Missing Proofs From Section 4.4: γ_n^* is Decreasing in n

Proof of Lemma 4.7. We know that γ_n^* equals the value

$$
\begin{aligned}
\min_{\substack{u:[n]\to\mathbf{R}_+ \\ \sum_i u_i \geq 1}} \quad & v_{(1,1)} \\
\text{(DLP)} \qquad \text{s.t.} \quad & \\
& v_{(t,s)} = \max\left\{pU_t(s) + \frac{1-p}{t+1}\sum_{\sigma=1}^{t=1} v_{(t+1,\sigma)}, \frac{1}{t+1}\sum_{\sigma=1}^{t=1} v_{(t+1,\sigma)}\right\} && \forall t \in [n], s \in [t] \\
& v_{(n+1,s)} = 0 && \forall s \in [n+1]
\end{aligned}
$$

where $U_t(s) = \sum_{i=1}^{n} \frac{u_k}{1-(1-p)^k}\,\mathbf{P}(R_t \in [k] \mid r_t = s) = \sum_{j=1}^{n}\sum_{k\geq j}\left(\frac{u_k}{1-(1-p)^k}\right)\mathbf{P}(R_t = j \mid r_t = s)$. Thus the utility collected by the policy if it collects a candidate with rank i is $U_i = \sum_{k\geq i}\frac{u_k}{1-(1-p)^k}$. Let $u : [n] \to [0,1]$ such that $\sum_{i=1}^{n} u_i = 1$ and extend u to $\widehat{u} : [n+1] \to [0,1]$ by $\widehat{u}_{n+1} = 0$ and define $\widehat{U}_t(s)$ accordingly. Consider the optimal policy

that solves the program

$$\widehat{v}_{(t,s)} = \max\left\{ p\widehat{U}_t(s) + \frac{1-p}{t+1}\sum_{\sigma=1}^{t+1}\widehat{v}_{(t+1\sigma)}, \frac{1}{t+1}\sum_{\sigma=1}^{t+1}\widehat{v}_{(t+1,\sigma)} \right\}, \forall t \in [n+1], s \in [t]$$

with the boundary condition $\widehat{v}_{(n+2,s)} = 0$ for all $s \in [n+2]$. Call this policy $\widehat{\mathcal{P}}$. Note that when policy $\widehat{\mathcal{P}}$ collects a candidate with rank i, then it gets a utility of

$$\widehat{U}_i = \sum_{k\geq i}\frac{\widehat{u}_k}{1-(1-p)^k} = \sum_{k\geq i}\frac{u_k}{1-(1-p)^k} = U_i,$$

for $i \leq n$ and $\widehat{U}_{n+1} = 0$. By the choice of $\widehat{\mathcal{P}}$, the expected utility collected by $\mathcal{P}$ is $\mathrm{val}(\widehat{\mathcal{P}}) = \widehat{v}_{(1,1)}$. We can obtain a policy $\mathcal{P}$ for n elements out of $\widehat{\mathcal{P}}$ by simulating an entry of $n+1$ elements as follows. Policy $\mathcal{P}$ randomly selects a time $t^* \in [n+1]$ and its availability b: we set $b = 0$ (unavailable) with probability $1-p$ and $b = 1$ (available) with probability p. Now, on a input of length n, the policy $\mathcal{P}$ will squeeze an item of rank $n+1$ in position t^* and it will run the policy $\widehat{\mathcal{P}}$ in this input, simulating appropriately the new partial ranks. That is, before stage t^* policy $\mathcal{P}$ behaves exactly as $\widehat{\mathcal{P}}$ in the original input of $\mathcal{P}$. When the policy leaves the stage t^*-1 to transition to stage t^*, then the policy $\mathcal{P}$ simulates the simulated candidate t^* (with real rank $n+1$) that $\widehat{\mathcal{P}}$ would have received and does the following: ignores the candidate and moves to stage t^* if the simulated candidate is unavailable ($b = 0$) or if $\widehat{\mathcal{P}}((t^*,t^*)) =$ **pass**, while if $\widehat{\mathcal{P}}((t^*,t^*)) =$ **offer** and the simulated candidate accepts ($b = 1$) then the policy $\mathcal{P}$ accepts any candidate from that point on.

Coupling the input of length $n+1$ for $\widehat{\mathcal{P}}$ and the input of length n with the random stage t^* for $\mathcal{P}$, we can see that the utilities collected by $\mathcal{P}$ and $\widehat{\mathcal{P}}$ coincide, i.e., $U(\mathcal{P}) = \widehat{\widehat{\mathcal{P}}} = \widehat{v}_{(1,1)}$. Thus the optimal utility $v_{(1,1)}$ collected by a policy for n candidates and utilities given by $U : [n] \to \mathbf{R}_+$, holds $v_{(1,1)} \geq \widehat{v}_{(1,1)}$. Since $\widehat{v}_{(1,1)} \geq \gamma^*_{n+1}$, by minimizing over u we obtain $\gamma^*_n \geq \gamma^*_{n+1}$. □

4.A.4 Missing Proofs From Section 4.5

In this subsection we show that $|\gamma_n^* - \gamma_\infty^*| \leq \mathcal{O}((\log n)^2/\sqrt{n})$ for fixed p (Lemma 4.8). The proof is similar to other infinite approximation of finite models and we require some preliminary results before showing the result. First, we introduce two relaxations, one for (LP) and one for (CLP). We show that the relaxations have values close to their non-relaxed versions. After these preliminaries have been introduced we present the proof of Lemma 4.8.

Consider the relaxation of $(LP)_{n,p}$ to the top q candidate constraints:

$$\begin{aligned}
\gamma_{n,q}^* = \max_{x \geq 0} \quad & \gamma \\
(LP)_{n,p,q} \qquad & x_{t,s} \leq \frac{1}{t}\left(1 - p\sum_{\tau<t}\sum_{s'=1}^{\tau} x_{\tau,s'}\right) && \forall t, s \quad (4.20) \\
& \gamma \leq \frac{p}{(1-(1-p)^k)}\sum_{t=1}^{n}\sum_{s=1}^{t} x_{t,s}\,\mathbf{P}(R_t \in [k] \mid r_t = s) && \forall k \in [q] \\
& && (4.21)
\end{aligned}$$

Note that $\gamma_n^* \leq \gamma_{n,q}^*$ since $(LP)_{n,p,q}$ is a relaxation of $(LP)_{n,p}$. The following result gives a bound on γ_n^* compared to $\gamma_{n,q}^*$.

Proposition 4.19. *For any $q \in [n]$, $\gamma_n^* \geq (1-(1-p)^q)\,\gamma_{n,q}^*$.*

Proof. Let $(\mathbf{x}, \gamma_{n,q}^*)$ be an optimal solution of $(LP)_{n,p,q}$. Let

$$f_k = \frac{p}{1-(1-p)^k}\sum_{t=1}^{n}\sum_{s=1}^{t} x_{t,s}\,\mathbf{P}(R_t \in [k] \mid r_t = s).$$

Then $\gamma_{n,q}^* = \min_{k=1,\ldots,q} f_i$. It is enough to show that $f_i \geq \left(\frac{1-(1-p)^q}{1-(1-p)^n}\right) f_q$ for $i \geq q$ because $\gamma_n^* \geq \min_{i=1,\ldots,n} f_i \geq \left(\frac{1-(1-p)^q}{1-(1-p)^n}\right)\min_{i=1,\ldots,q} f_i \geq (1-(1-p)^q)\,\gamma_{n,q}^*$.

For any j we have

$$f_{j+1} = \frac{p}{1-(1-p)^{j+1}} \sum_{t=1}^{n} \sum_{s=1}^{t} x_{t,s}\, \mathbf{P}(R_t \in [j+1] \mid r_t = s) \geq \left(\frac{1-(1-p)^j}{1-(1-p)^{j+1}} \right) f_j.$$

Thus, iterating this for $j > q$ we get $f_j \geq \left(\frac{1-(1-p)^q}{1-(1-p)^j} \right) f_q$ and we obtain the desired result.

□

Likewise we consider the relaxation of $(CLP)_p$ to the top q candidates:

$$\gamma^*_{\infty,q} = \max_{\substack{\alpha \in [0,\infty)^{[0,1]\times \mathbf{N}} \\ \gamma \geq 0}} \quad \gamma$$

$$(CLP)_{p,q} \qquad \alpha(t,s) \leq \frac{1}{t} \left(1 - p \int_0^t \sum_{\sigma \geq 1} \alpha(\tau, \sigma)\, \mathrm{d}\tau \right) \qquad \forall t \in [0,1], s \geq 1 \tag{4.22}$$

$$\gamma \leq \frac{p}{(1-(1-p)^k)} \int_0^1 \sum_{s \geq 1} \alpha(t,s) \sum_{\ell=s}^{k} \binom{\ell-1}{s-1} t^s (1-t)^{\ell-s}\, \mathrm{d}t \quad \forall k \in [q] \tag{4.23}$$

We have $\gamma^*_{\infty,q} \geq \gamma^*_\infty$ and we have the approximate converse

Proposition 4.20. *For any $q \geq 1$, $\gamma^*_\infty \geq (1-(1-p)^q)\, \gamma^*_{\infty,q}$.*

Proof. Let $(\alpha, \gamma_{\infty,q})$ be a feasible solution of $(CLP)_{p,q}$. Let

$$f_k = \frac{p}{1-(1-p)^k} \int_0^1 \sum_{s \geq 1} \alpha(t,s) \sum_{\ell=s}^{k} \binom{\ell-1}{s-1} t^s (1-t)^{\ell-s}\, \mathrm{d}t$$

Assume that $\gamma_{\infty,q} \leq \min_{k \leq q} f_k$. As in the previous proof, we aim to show that $f_i \geq$

$(1-(1-p)^q)\, f_q$ for $i \geq q$, since this will imply $\gamma^*_\infty \geq (1-(1-p)^q)\gamma_{\infty,q}$ for any $(\alpha, \gamma_{\infty,q})$ feasible for $(CLP)_{p,q}$.

Now, for any j we have

$$\begin{aligned} f_{j+1} &= \frac{p}{1-(1-p)^{j+1}} \int_0^1 \sum_{s\geq 1} \alpha(t,s) \sum_{\ell=s}^{j+1} \binom{\ell-1}{s-1} t^s (1-t)^{\ell-s}\, \mathrm{d}t \\ &\geq \frac{p}{1-(1-p)^{j+1}} \int_0^1 \sum_{s\geq 1} \alpha(t,s) \sum_{\ell=s}^{j} \binom{\ell-1}{s-1} t^s (1-t)^{\ell-s}\, \mathrm{d}t \\ &= \left(\frac{1-(1-p)^j}{1-(1-p)^{j+1}}\right) f_j. \end{aligned}$$

Iterating the inequality until reaching q we deduce that for any $j \geq q$ we have $f_j \geq \left(\frac{1-(1-p)^q}{1-(1-p)^j}\right) f_q$. From here the result follows. □

Remark 0. If we set $q = (\log n)/p$, both results imply that $\gamma^*_n \geq (1-1/n)\, \gamma^*_{n,q}$ and $\gamma^*_\infty \geq (1-1/n)\, \gamma^*_{\infty,q}$. Thus we lose at most $1/n$ by restricting the analysis to the top q candidates.

Proposition 4.21. *There is n_0 such that for $n \geq n_0$, for any t such that $\sqrt{n}\log n \leq t \leq n - \sqrt{n}\log n$, $\ell \leq \log(n)/p$ and $\ell \geq s$ it holds that for any $\tau \in [t/n, (t+1)/n]$ we have*

$$1 - \frac{10}{p\sqrt{n}} \leq \frac{\binom{\ell-1}{s-1}\binom{n-\ell}{t-s}/\binom{n}{t}}{\binom{\ell-1}{s-1}\tau^s(1-\tau)^{\ell-s}} \leq 1 + \frac{10}{p\sqrt{n}}.$$

Proof. We only need to show that

$$1 - \frac{10}{p\sqrt{n}} \leq \frac{\binom{n-\ell}{t-s}/\binom{n}{t}}{\tau^s(1-\tau)^{\ell-s}} \leq 1 + \frac{10}{p\sqrt{n}}.$$

We have

$$
\begin{aligned}
&\frac{\binom{n-\ell}{t-s}}{\binom{n}{t}} \\
&= \frac{t!}{(t-s)!}\frac{1}{n!}\frac{(n-\ell)!(n-t)!}{(n-t-(\ell-s))!} \\
&= \left(\frac{t\cdot(t-1)\cdots(t-s+1)}{n\cdot(n-1)\cdots(n-s+1)}\right)\left(\frac{(n-t)(n-t-1)\cdots(n-t-(\ell-s)+1)}{(n-s)(n-s-1)\cdots(n-s-(\ell-s)+1)}\right) \\
&= \underbrace{\left(\frac{t}{n}\right)^s\left(1-\frac{t}{n}\right)^{\ell-s}}_{A}\underbrace{\left(\frac{1\cdot\left(1-\frac{1}{t}\right)\cdots\left(1-\frac{(s-1)}{t}\right)}{1\cdot\left(1-\frac{1}{n}\right)\cdots\left(1-\frac{(s-1)}{n}\right)}\right)}_{B}\underbrace{\left(\frac{1\cdot\left(1-\frac{1}{n-t}\right)\cdots\left(1-\frac{(\ell-s)-1}{n-t}\right)}{\left(1-\frac{s}{n}\right)\cdot\left(1-\frac{(s+1)}{n}\right)\cdots\left(1-\frac{(s+(\ell-s)-1)}{n}\right)}\right)}_{C}
\end{aligned}
$$

We bound terms A, B and C separately. Since $s \le \ell$ and we are assuming that $\ell \le (\log n)/p$ and $t \ge \sqrt{n}\log n$ for n large, then we will implicitly use that $s, \ell \le \min\{t/2, n/2\}$.

Claim 4.22. *We have*

$$\left(1-\frac{4}{(p\sqrt{n})}\right)\tau^s(1-\tau)^{\ell-s} \le A = \left(\frac{t}{n}\right)^s(1-t/n)^{\ell-s} \le \left(1+\frac{4}{(p\sqrt{n})}\right)\tau^s(1-\tau)^{\ell-s}.$$

Proof. For the upper bound we have

$$
\begin{aligned}
\left(\frac{t}{n}\right)^s\left(1-\frac{t}{n}\right)^{\ell-s} &\le \tau^s\left(1-\tau+\frac{1}{n}\right)^{\ell-s} && (\tau \in [t/n, (t+1)/n]) \\
&= \tau^s(1-\tau)^{\ell-s}\left(1+\frac{1}{(1-\tau)n}\right)^{\ell-s} \\
&\le \tau^s(1-\tau)^{\ell-s}e^{(\ell-s)/((1-\tau)n)} \\
&\le \tau^s(1-\tau)^{\ell-s}e^{\ell/(n-t-1)} \\
&\le \tau^s(1-\tau)^{\ell-s}\left(1+2\frac{\ell}{n-t-1}\right) \\
& && \text{(Using } e^x \le 1+2x \text{ for } x \in [0,1])
\end{aligned}
$$

The upper bound now follows by using the information over ℓ and t and that $(1+2\ell/(n-t-1)) \le 1+2(\log n)p^{-1}(\sqrt{n}\log n-1)^{-1} \le 1+4/(p\sqrt{n})$ for n large.

For the lower bound we have

$$\begin{aligned}\left(\frac{t}{n}\right)^s\left(1-\frac{t}{n}\right)^{\ell-s} &\geq \left(\tau-\frac{1}{n}\right)^s(1-\tau)^{\ell-s}\\ &= \tau^s(1-\tau)^{\ell-s}\left(1-\frac{1}{\tau n}\right)^s\\ &\geq \tau^s(1-\tau)^{\ell-s}e^{-\frac{s}{\tau n-1}}\\ &\geq \tau^s(1-\tau)^{\ell-s}\left(1-s/(\tau n-1)\right)\end{aligned}$$

Since $s/(\tau n-1) \leq \log(n)/(p(t-1)) \leq 2/(p\sqrt{n})$ for n large, the lower bound follows.

□

Claim 4.23. $1-2s^2/t \leq B \leq 1+2s^2/n$

Proof. For the upper bound we upper bound the denominator

$$\begin{aligned}\left(\frac{1\cdot\left(1-\frac{1}{t}\right)\cdots\left(1-\frac{(s-1)}{t}\right)}{1\cdot\left(1-\frac{1}{n}\right)\cdots\left(1-\frac{(s-1)}{n}\right)}\right) &\leq \frac{1}{1\cdot\left(1-\frac{1}{n}\right)\cdots\left(1-\frac{(s-1)}{n}\right)}\\ &\leq e^{\sum_{k=1}^{s-1}k/(n-k)}\\ &\leq e^{s^2/n} \qquad \text{(Function } x\mapsto x/(n-x) \text{ is increasing)}\\ &\leq 1+2\frac{s^2}{n}\end{aligned}$$

For the lower bound we lower bound the numerator:

$$\begin{aligned}\left(\frac{1\cdot\left(1-\frac{1}{t}\right)\cdots\left(1-\frac{(s-1)}{t}\right)}{1\cdot\left(1-\frac{1}{n}\right)\cdots\left(1-\frac{(s-1)}{n}\right)}\right) &\geq 1\cdot\left(1-\frac{1}{t}\right)\cdots\left(1-\frac{(s-1)}{t}\right)\\ &\geq e^{-\sum_{k=1}^{s-1}k/(t-k)}\\ &\text{(Using } 1-k/t=(1+k/(t-k))^{-1}\geq e^{-k/(t-k)})\\ &\geq 1-\frac{s^2}{t-s} \geq 1-2\frac{s^2}{t}.\end{aligned}$$

□

Claim 4.24. $1 - 2\ell^2/(n-t) \leq C \leq 1 + 2\ell^2/n$

Proof. Similar to the previous claim, we bound denominator for an upper bound and numerator for a lower bound.

$$\left(\frac{1 \cdot \left(1 - \frac{1}{n-t}\right) \cdots \left(1 - \frac{(\ell-s)-1}{n-t}\right)}{\left(1 - \frac{s}{n}\right) \cdot \left(1 - \frac{(s+1)}{n}\right) \cdots \left(1 - \frac{(s+(\ell-s)-1)}{n}\right)}\right) \leq \frac{1}{\left(1 - \frac{s}{n}\right) \cdot \left(1 - \frac{(s+1)}{n}\right) \cdots \left(1 - \frac{(s+(\ell-s)-1)}{n}\right)}$$
$$\leq e^{\sum_{k=0}^{\ell-s-1} (k+s)/(n-k)} \leq 1 + 2\frac{\ell^2}{n},$$

and

$$\left(\frac{1 \cdot \left(1 - \frac{1}{n-t}\right) \cdots \left(1 - \frac{(\ell-s)-1}{n-t}\right)}{\left(1 - \frac{s}{n}\right) \cdot \left(1 - \frac{(s+1)}{n}\right) \cdots \left(1 - \frac{(s+(\ell-s)-1)}{n}\right)}\right) \geq 1 \cdot \left(1 - \frac{1}{n-t}\right) \cdots \left(1 - \frac{(\ell-s)-1}{n-t}\right)$$
$$\geq e^{-\sum_{k=0}^{\ell-s-1} k/(n-t-k)} \geq 1 - 2\frac{\ell^2}{n-t}.$$

□

We can now upper bound ABC as

$$\begin{aligned}
ABC &\leq \tau^s (1-\tau)^{\ell-s} \left(1 + \frac{4}{p\sqrt{n}}\right)\left(1 + 2\frac{s^2}{n}\right)\left(1 + 2\frac{\ell^2}{n}\right) \\
&\leq \tau^s (1-\tau)^{\ell-s} \left(1 + \frac{4}{p\sqrt{n}}\right)\left(1 + 2\frac{(\log n)^2}{p^2 n}\right)^2 \\
&\qquad \text{(Using } t \leq n - \sqrt{n}\log n \text{ and } s \leq \ell \leq (\log n)/p\text{)} \\
&\leq \tau^s (1-\tau)^{\ell-s} \left(1 + \frac{4}{p\sqrt{n}}\right)\left(1 + 6\frac{(\log n)^2}{p^2 n}\right) \\
&\qquad \text{(Using } (1+x)^2 \leq 1 + 3x \text{ if } x \in [0,1]\text{)} \\
&\leq \tau^s (1-\tau)^{\ell-s} \left(1 + \frac{10}{p\sqrt{n}}\right).
\end{aligned}$$

Recall that we are assuming p constant and n large, thus the dominating term is $1/\sqrt{n}$. Similarly, we can lower bound ABC as

$$\begin{aligned} ABC &\geq \tau^s(1-\tau)^{\ell-s}\left(1-\frac{4}{p\sqrt{n}}\right)\left(1-2\frac{s^2}{t}\right)\left(1-2\frac{\ell^2}{n-t}\right) \\ &\geq \tau^s(1-\tau)^{\ell-s}\left(1-\frac{4}{p\sqrt{n}}\right)\left(1-2\frac{(\log n)^2}{p^2 t}\right)\left(1-2\frac{(\log n)^2}{p^2(n-t)}\right) \\ &\geq \tau^s(1-\tau)^{\ell-s}\left(1-\frac{4}{p\sqrt{n}}\right)\left(1-2\frac{(\log n)}{p^2\sqrt{n}}\right)^2 \geq \tau^s(1-\tau)^{\ell-s}\left(1-\frac{10}{p\sqrt{n}}\right). \end{aligned}$$

□

Proof of Lemma 4.8. We are going to show $|\gamma_n^* - \gamma_\infty^*| \leq \mathcal{O}((\log n)^2/\sqrt{n})$. Since we can only guarantee good approximation of the binomial terms in Proposition 4.21 for $\ell \leq (\log n)/p$, we need to restrict our analysis to $\gamma_{n,q}^*$ and $\gamma_{\infty,q}^*$ for $q = (\log n)/p$. This is enough since these values are withing $1/n$ of γ_n^* and γ_∞^* respectively due to Propositions 4.19 and 4.20 (see Remark 2).

Before proceeding, we give two technical results that allow us to control an error for values of t not considered by Proposition 4.21. The deduction is a routine calculation and it is skipped for brevity.

Claim 4.25. *For any x feasible for Constraints* (4.20) *and such that $x_{t,s} = 0$ for $s > q$, we have for $k \leq q$*

- $\sum_{t=1}^{\sqrt{n}\log n}\sum_{s=1}^{t} x_{t,s}\,\mathbf{P}(R_t \in [k] \mid r_t = s) \leq 10(\log n)^2/(p\sqrt{n})$.
- $\sum_{t=n-\sqrt{n}\log n}^{n}\sum_{s=1}^{t} x_{t,s}\,\mathbf{P}(R_t \in [k] \mid r_t = s) \leq 10(\log n)^2/(p\sqrt{n})$.

Claim 4.26. *For any α feasible for Constraints* (4.22)*, we have for $k \leq q$*

- $\int_{1-(\log n)/\sqrt{n}}^{1}\sum_{s=1}^{k}\alpha(\tau,s)\sum_{\ell=s}^{k}\binom{\ell-1}{s-1}\tau^s(1-\tau)^{\ell-s}\mathrm{d}\tau \leq (\log n)^2/(p\sqrt{n})$.

- $\int_0^{(\log n)/\sqrt{n}} \sum_{s=1}^{k} \alpha(\tau, s) \sum_{\ell=s}^{k} \binom{\ell-1}{s-1} \tau^s (1-\tau)^{\ell-s} \mathrm{d}\tau \leq (\log n)^2/(p\sqrt{n})$.

First, we show that $\gamma^*_{n,q} \geq \gamma^*_\infty - 40(\log n)^2/(p\sqrt{n})$. Let (α, γ) be a feasible solution of the continuous LP $(CLP)_p$. We construct a solution of the $(LP)_{n,p,q}$ as follows. Define

$$x_{t,s} = \frac{t-(1-p)}{t} \int_{(t-1)/n}^{t/n} \alpha(\tau, s)\, \mathrm{d}\tau \qquad\qquad \forall t \in [n], \forall s \in [t],$$

and $\gamma_{n,q} = \min_{k \leq q} \frac{p}{1-(1-p)^k} \sum_{t=1}^{n} \sum_{s=1}^{t} x_{t,s}\, \mathbf{P}(R_t \in [k] \mid r_t = s)$. Let us show that $(\mathbf{x}, \gamma_{n,q})$ is feasible for $(LP)_{n,q}$, i.e., it satisfies Constraints (4.20)-(4.21). First, for $\tau \in [(t-1)/n, t/n]$ we have

$$\begin{aligned}
\tau\alpha(\tau, s) + p \int_{(t-1)/n}^{\tau} \alpha(\tau', s)\, \mathrm{d}\tau' &\leq 1 - p \int_0^{\tau} \sum_{\sigma \geq 1} \alpha(\tau', \sigma)\, \mathrm{d}\tau' + p \int_{(t-1)/n}^{\tau} \alpha(\tau', s)\, \mathrm{d}\tau' \\
&\leq 1 - p \int_0^{(t-1)/n} \sum_{\sigma \geq 1} \alpha(\tau', \sigma)\, \mathrm{d}\tau' \leq 1 - p \sum_{\tau'=1}^{t-1} \sum_{\sigma=1}^{\tau'} x_{\tau',\sigma}.
\end{aligned}$$

We now integrate in $[(t-1)/n, t/n]$ on both sides of the inequality. After integration, the RHS equals $\left(1 - p \sum_{\tau=1}^{t} \sum_{\sigma=1}^{\tau} x_{\tau,\sigma}\right)/n$. On the LHS we obtain,

$$\begin{aligned}
\int_{(t-1)/n}^{t/n} \left(\tau\alpha(\tau, s) + p \int_{(t-1)/n}^{\tau} \alpha(\tau', s) \mathrm{d}\tau' \right) \mathrm{d}\tau &= \frac{t}{n} \int_{(t-1)/n}^{t/n} \alpha(\tau, s) \mathrm{d}\tau \\
&\quad - (1-p) \int_{(t-1)/n}^{t/n} \left(\frac{t}{n} - \tau \right) \alpha(\tau, s) \mathrm{d}\tau \\
&\geq \frac{(t-(1-p))}{n} \int_{(t-1)/n}^{t/n} \alpha(\tau, s) \mathrm{d}\tau \\
&\qquad\qquad \text{(Using } t/n - \tau \leq 1/n\text{)} \\
&= \frac{t}{n} x_{t,s}.
\end{aligned}$$

Thus Constraints (4.20) hold. By definition of $\gamma_{n,q}$, Constraints (4.21) also hold.

Now, note that for $t \geq \sqrt{n}\log n$ we have

$$x_{t,s} \geq \left(1 - \frac{1}{\sqrt{n}\log n}\right) \int_{(t-1)/n}^{t/n} \alpha(\tau, s)\mathrm{d}\tau.$$

Then,

$$\begin{aligned}
\gamma_{n,q} &= \min_{k\leq q} \frac{p}{1-(1-p)^k} \sum_{t=1}^{n}\sum_{s=1}^{t} x_{t,s} \sum_{\ell=s}^{k\wedge(n-t+s)} \frac{\binom{\ell-1}{s-1}\binom{n-\ell}{t-s}}{\binom{n}{t}} \quad \text{(Definition of } \mathbf{P}(R_t \in [k] \mid r_t = s)\text{)} \\
&\geq \min_{k\leq q} \frac{p}{1-(1-p)^k} \sum_{t=\sqrt{n}\log n}^{n-\sqrt{n}\log n}\sum_{s=1}^{t} \int_{(t-1)/n}^{t/n} \alpha(\tau, s)\,\mathrm{d}\tau \sum_{\ell=s}^{k\wedge(n-t+s)} \frac{\binom{\ell-1}{s-1}\binom{n-\ell}{t-s}}{\binom{n}{t}} \left(1 - \frac{1}{\sqrt{n}\log n}\right) \\
&\geq \min_{k\leq q} \frac{p}{1-(1-p)^k} \sum_{t=\sqrt{n}\log n}^{n-\sqrt{n}\log n}\sum_{s=1}^{k} \int_{(t-1)/n}^{t/n} \alpha(\tau, s) \sum_{\ell=s}^{k} \binom{\ell-1}{s-1}\tau^s(1-\tau)^{\ell-s}\,\mathrm{d}\tau \left(1 - \frac{20}{p\sqrt{n}}\right) \\
&\qquad \text{(Since } n - t + s \geq \sqrt{n}\log n \geq k \text{ and } t \leq k \text{ and using Proposition 4.21)} \\
&= \min_{k\leq q} \frac{p}{1-(1-p)^k} \sum_{t=\sqrt{n}\log n}^{n-\sqrt{n}\log n} \int_{(t-1)/n}^{t/n} \sum_{\ell=1}^{k}\sum_{s=1}^{\ell} \alpha(\tau, s)\binom{\ell-1}{s-1}\tau^s(1-\tau)^{\ell-s}\,\mathrm{d}\tau \left(1 - \frac{20}{p\sqrt{n}}\right) \\
&\geq \min_{k\leq q} \left(\frac{p}{1-(1-p)^k} \int_0^1 \sum_{\ell=1}^{k}\sum_{s=1}^{\ell} \alpha(\tau, s)\binom{\ell-1}{s-1}\tau^s(1-\tau)^{\ell-s}\,\mathrm{d}\tau - 2\frac{(\log n)^2}{p\sqrt{n}}\right)\left(1 - \frac{20}{p\sqrt{n}}\right) \\
&\qquad \text{(Claim 4.26)} \\
&\geq \left(\gamma - 2\frac{(\log n)^2}{p\sqrt{n}}\right)\left(1 - \frac{10}{p\sqrt{n}}\right) \geq \gamma - \frac{20}{p\sqrt{n}} - 2\frac{\log n}{\sqrt{n}}.
\end{aligned}$$

Optimizing over γ, $\gamma_{n,q}$ and using Proposition 4.19 gives us $\gamma^*_{n,q} \geq \gamma^*_\infty - 40\log(n)/(p\sqrt{n})$ for n large.

Now we show that $\gamma^*_{\infty,q} \geq \gamma^*_{n,q} - 40(\log n)^2/(p\sqrt{n})$. Let $(x, \gamma_{n,q})$ be a solution of $(LP)_{n,p,q}$. Let us construct a solution of $(CLP)_{p,q}$. Note that we can assume $x_{t,s} = 0$ for $s > q$ as $(LP)_{n,p,q}$ does not improve its objective function by allocating any mass to these variables. Consider α defined as follows: for $\tau \in [0, 1]$ let

$$\alpha(\tau, s) = \begin{cases} nx_{t,s}\,(1 - \log(n)/\sqrt{n}) & t = \lceil \tau n \rceil \geq \sqrt{n}, s \leq \min\{t, \log n/p\} \\ 0 & t = \lceil \tau n \rceil < \sqrt{n} \text{ or } s > \min\{t, \log n/p\} \end{cases}$$

Let $\gamma_{\infty,q} = \min_{k\le q} \frac{p}{1-(1-p)^k} \int_0^1 \sum_{s\ge 1} \alpha(\tau,s) \sum_{\ell=s}^{k} \binom{\ell-1}{s-1} \tau^s (1-\tau)^{\ell-s}\, \mathrm{d}\tau$. We show first that $(\alpha, \gamma_{\infty,q})$ is feasible for $(CLP)_{p,q}$, and for this it is enough to show that α holds Constraints (4.20). For $\tau < 1/\sqrt{n}$ we have $\alpha(\tau,s) = 0$ for any s, thus Constraint (4.22) is satisfied in this case. Let us verify that for $\tau \ge 1/\sqrt{n}$ the constraint also holds. Let $t = \lceil \tau n \rceil$ and $s \ge 1$. Then

$$
\begin{aligned}
\tau\alpha(\tau,s) + p\int_0^\tau \sum_{\sigma\ge 1} \alpha(\tau',\sigma)\,\mathrm{d}\tau' &\le \tau\alpha(\tau,s) + p\sum_{t'=1}^{t-1} \int_{\frac{t'-1}{n}}^{\frac{t'}{n}} \sum_{s=1}^{t'} \alpha(\tau',s)\,\mathrm{d}\tau' \\
&\quad + p\int_{\frac{t-1}{n}}^{\tau} \sum_{\sigma=1}^{\frac{\log n}{p}} \alpha(\tau',\sigma)\,\mathrm{d}\tau' \\
&\le \left(1-\frac{\log n}{\sqrt{n}}\right)\left(t x_{t,s} + p\sum_{t'=1}^{t-1}\sum_{s=1}^{t'} x_{t',s} + p\frac{\log n}{pt}\right) \\
&\qquad\qquad \text{(Since } x_{t,s} \le \tfrac{1}{t} \text{ always)} \\
&\le \left(1-\frac{\log n}{\sqrt{n}}\right)\left(1+\frac{\log n}{t}\right) \\
&\le \left(1-\frac{\log n}{\sqrt{n}}\right)\left(1+\frac{\log n}{\sqrt{n}}\right). \qquad (t \ge \sqrt{n})
\end{aligned}
$$

The last term is < 1. Thus $(\alpha, \gamma_{\infty,q})$ is feasible for $(CLP)_{p,q}$. Now,

$$\begin{aligned}
\gamma_{\infty,q} &= \min_{k\le q} \frac{p}{1-(1-p)^k} \int_0^1 \sum_{s\ge 1} \alpha(\tau,s) \sum_{\ell=s}^{k} \binom{\ell-1}{s-1} \tau^s (1-\tau)^{\ell-s}\, \mathrm{d}\tau \\
&\ge \min_{k\le q} \frac{p}{1-(1-p)^k} \sum_{t=\sqrt{n}\log n}^{n-\sqrt{n}\log n} \int_{\frac{t-1}{n}}^{\frac{t}{n}} \sum_{s\ge 1} \alpha(\tau,s) \sum_{\ell=s}^{k} \binom{\ell-1}{s-1} \tau^s (1-\tau)^{\ell-s}\, \mathrm{d}\tau \\
&\ge \min_{k\le q} \frac{p}{1-(1-p)^k} \sum_{t=\sqrt{n}\log n}^{n-\sqrt{n}\log n} \int_{\frac{t-1}{n}}^{\frac{t}{n}} \sum_{s\ge 1} \alpha(\tau,s) \sum_{\ell=s}^{k} \frac{\binom{\ell-1}{s-1}\binom{n-\ell}{t-s}}{\binom{n}{t}}\, \mathrm{d}\tau \left(1-\frac{10}{p\sqrt{n}}\right) \\
&\qquad\qquad \text{(Proposition 4.21)} \\
&\ge \min_{k\le q} \frac{p}{1-(1-p)^k} \sum_{t=\sqrt{n}\log n}^{n-\sqrt{n}\log n} \sum_{s=1}^{t} x_{t,s} \sum_{\ell=s}^{k} \frac{\binom{\ell-1}{s-1}\binom{n-\ell}{t-s}}{\binom{n}{t}}\, \mathrm{d}\tau \left(1-\frac{\log n}{\sqrt{n}}\right)\left(1-\frac{10}{p\sqrt{n}}\right) \\
&\ge \left(\min_{k\le q} \frac{p}{1-(1-p)^k} \sum_{t=1}^{n}\sum_{s=1}^{t} x_{t,s} \sum_{\ell=s}^{k} \frac{\binom{\ell-1}{s-1}\binom{n-\ell}{t-s}}{\binom{n}{t}}\, \mathrm{d}\tau - 20\frac{(\log n)^2}{p\sqrt{n}}\right)\left(1-\frac{\log n}{\sqrt{n}}\right)\left(1-\frac{10}{p\sqrt{n}}\right) \\
&\qquad\qquad \text{(Claim 4.25)} \\
&\ge \gamma_{n,q} - 40\frac{(\log n)^2}{p\sqrt{n}}.
\end{aligned}$$

Thus optimizing over $\gamma^*_{\infty,q}, \gamma^*_{n,q}$ we obtain the desired bound. Using Propositions 4.19 and 4.20 we can conclude that, for n large, $\gamma^*_\infty - 50(\log n)^2/(p\sqrt{n}) \le \gamma^*_n \le \gamma^*_\infty + 50(\log n)^2/(p\sqrt{n})$. □

4.A.5 Missing Proofs From Section 4.6

Proof of Lemma 4.10. The input of the i.i.d. prophet inequality problem corresponds to a known distribution $\mathcal{D}$ with support in $[0,1]$. The DM sequentially accesses at most m samples from $\mathcal{D}$ and upon observing one of these values, she has to decide irrevocably if to take it and stop the process or continue. We are going to use $\mathcal{A}$ to design a strategy for the prophet problem. We assume that the samples from $\mathcal{D}$ are all distinct. Indeed, we can add some small Gaussian noise to the distribution and consider a continuous distribution

$\mathcal{D}'$ instead.

Note that $\mathcal{A}$ runs on an input of size n where a fraction p of the candidates accept an offer. We interpret $pn \approx m$ as the set of samples for the prophet inequality problem, while the remaining $(1-p)n$ items are used as additional information for the algorithm. By concentration bounds, we are going to argue that we only need to run $\mathcal{A}$ in at most $(1+\varepsilon)pn$ positive samples.

Formally, we proceed as follows. Fix n and $\varepsilon > 0$ and consider the algorithm $\mathcal{B}$ that receives an online input of n numbers $x_1, \ldots, x_n$. The algorithm flips n coins with probability of heads p and marks item i as available if the corresponding coins that turn out heads. Algorithm $\mathcal{B}$ feeds algorithm $\mathcal{A}$ with the partial rankings given by the ordering given by $x_1, \ldots, x_n$. If $\mathcal{A}$ selects a candidate but the candidate is mark as unavailable, then $\mathcal{B}$ moves to the next item. If $\mathcal{A}$ selects a candidate i and it is marked as available, then the process ends with $\mathcal{B}$ collecting the value x_i. Let us denote by $\mathrm{Val}(\mathcal{B}, x_1, \ldots, x_n)$ the value collected by $\mathcal{B}$ in the online input $x_1, \ldots, x_n$. Then we have the following claim.

Claim 4.27. $\mathbf{E}_{\substack{X_1,\ldots,X_n \\ S}} [\mathrm{Val}(\mathcal{B}, X_1, \ldots, X_n)] \geq \gamma \, \mathbf{E}_{\substack{X_1,\ldots,X_n \\ S}} [\max_{i \in S} X_i]$*, where S is the random set of items marked as available and $X_1, \ldots, X_n$ are n i.i.d. random variables with common distribution $\mathcal{D}$.*

Proof. Fix $x_1, \ldots, x_n$ points in the support of $\mathcal{D}$. Then, a simple application of Proposition 4.5 shows

$$\frac{\mathbf{E}_{S,\pi} \left[\mathrm{Val}(\mathcal{B}, x_{\pi(1)}, \ldots, x_{\pi(n)}) \right]}{\mathbf{E}_S \left[\max_{i \in S} x_i \right]} \geq \gamma$$

Note that we need to feed $\mathcal{B}$ with all permutations of $x_1, \ldots, x_n$ in order to obtain the guarantee of $\mathcal{A}$. From here, we obtain $\mathbf{E}_{\mathcal{S},\pi} [\mathrm{Val}(\mathcal{B}, x_\pi(1), \ldots, x_\pi(n))] \geq \gamma \, \mathbf{E}_{\mathcal{S}} [\max_{i \in S} x_i]$ and the conclusion follows by taking expectation in the variables $X_1 = x_1, \ldots, X_n =$

x_n. □

For ease of notation, we will refer by $\mathrm{Val}(\cdot)$ to $\mathrm{Val}(\cdot, X_1, \ldots, X_n)$. We modify slightly $\mathcal{B}$. Consider $\mathcal{B}'$ that runs normally $\mathcal{B}$ if $|S| \leq (1+\varepsilon)pn$ or simply return 0 value if $|S| > (1+\varepsilon)pn$. Then, we have

Claim 4.28. *Let $\varepsilon, \delta > 0$. For $n \geq 2\log(2/\delta)/(p\varepsilon^2)$ we have $\mathbf{E}_{\substack{X_1,\ldots,X_n \\ \mathcal{S}}}[\max_{i\notin\mathcal{S}} X_i] \geq (1-\delta)\,\mathbf{E}\left[\max_{i\leq(1-\varepsilon)pn} X_i\right]$ and $\mathbf{E}[\mathrm{Val}(\mathcal{B}')] + \delta \geq \mathbf{E}[\mathrm{Val}(\mathcal{B})]$.*

Proof. Using standard Chernoff concentration bounds (see for instance [30]) we get the inequality $\mathbf{P}_{\mathcal{S}}(||\mathcal{S}| - pn| \geq \varepsilon pn) \leq 2e^{-pn\varepsilon^2/2} = \delta$. Hence, for a number $n \geq 2\log(2/\delta)/(p\varepsilon^2)$, we can guarantee that

$$\mathop{\mathbf{E}}_{\substack{X_1,\ldots,X_n \\ \mathcal{S}}}\left[\max_{i\in\mathcal{S}} X_i\right] \geq (1-\delta)\,\mathbf{E}\left[\max_{i\leq(1-\varepsilon)pn} X_i\right].$$

For the second part we have $\mathbf{E}[\mathrm{Val}(\mathcal{B})] \leq \delta + \mathbf{E}[\mathrm{Val}(\mathcal{B}) \mid |S| \leq (1+\varepsilon)pn] = \delta + \mathbf{E}[\mathrm{Val}(\mathcal{B}')]$. □

Claim 4.29. *For any $\varepsilon > 0$ we have $\mathbf{E}\left[\max_{i\leq(1-\varepsilon)pn} X_i\right] \geq (1-\varepsilon)^2\,\mathbf{E}\left[\max_{i\leq(1+\varepsilon)pn} X_i\right]$.*

Proof. Since $\mathbf{P}(\max_{i\leq k} X_i \leq x) = \mathbf{P}(X_1 \leq x)^k$, then we have

$$\frac{\mathbf{E}\left[\max_{i\leq(1-\varepsilon)pn} X_i\right]}{\mathbf{E}\left[\max_{i\leq(1+\varepsilon)pn} X_i\right]} = \frac{\int_0^\infty \left(1-\mathbf{P}(X_1\leq x)^{pn(1-\varepsilon)}\right)\,\mathrm{d}x}{\int_0^\infty \left(1-\mathbf{P}(X_1\leq x)^{pn(1+\varepsilon)}\right)\,\mathrm{d}x} \geq \inf_{x\geq 0}\frac{1-\mathbf{P}(X_1\leq x)^{pn(1-\varepsilon)}}{1-\mathbf{P}(X_1\leq x)^{pn(1+\varepsilon)}}$$

$$\geq \inf_{v\in[0,1)} f(v)$$

where $f(v) = (1-v^{1-\varepsilon})/(1-v^{1+\varepsilon})$. Now the conclusion follows by using the fact that the function f is nonincreasing and that $\inf_{v\in[0,1)} f(v) = \lim_{v\to 1} f(v) = (1-\varepsilon)/(1+\varepsilon) \geq$

$(1-\varepsilon)^2$. □

Putting together these two claims, we obtain an algorithm that checks at most $(1+\varepsilon)pn$ items and guarantees

$$\gamma(1-\varepsilon)^2(1-\delta)\,\mathbf{E}\left[\max_{i\leq(1+\varepsilon)pn} X_i\right] \leq \mathbf{E}\left[\mathrm{Val}(\mathcal{B}')\right] + \delta.$$

Now, fix $\varepsilon > 0$ small enough such that $(1+\varepsilon)p < 1$. We know that the set $\{\lfloor(1+\varepsilon)pn\rfloor\}_{n\geq 1}$ contains all non-negative integers. Thus, for $n \geq 2\log(2/\delta)/(p\varepsilon^2)$ algorithm $\mathcal{B}'$ in an input of length $m = \lfloor(1+\varepsilon)pn\rfloor$ guarantees

$$\gamma(1-\varepsilon)^2(1-\delta)\,\mathbf{E}\left[\max_{i\leq m} X_i\right] \leq \mathbf{E}\left[\mathrm{Val}(\mathcal{B}')\right] + \delta.$$

for any distribution $\mathcal{D}$ with support in $[0,1]$. This finishes the proof. □

The next result uses notation from the work by Hill & Kertz. For the details we refer the reader to the work [98]. The result states that there is a hard instance for the i.i.d. prophet inequality problem where $\mathbf{E}[\max_{i\leq m} X_i]$ is away from 0 by a quantity at least $1/m^3$. The importance of this reformulation of the result by Hill & Kertz is that $e^{-\Theta(n)}/\,\mathbf{E}[\max_{i\leq m} X_i] \to 0$ which is what we needed to show that $\gamma \leq 1/\beta$. Recall that $\beta \approx 1.341$ is the unique solution of the integral equation $\int_0^1 (y(1-\log y)+\beta-1)^{-1}\mathrm{d}y = 1$ [107].

Proposition 4.30 (Reformulation of Proposition 4.4 by [98])**.** *Let a_m be the sequence constructed by Hill & Kertz, i.e, such that $a_m \to \beta$ and for any sequence of i.i.d. random variables $X_1,\ldots,X_m$ with support in $[0,1]$ we have*

$$\mathbf{E}\left[\max_{i\leq m} X_i\right] \leq a_m \sup\left\{\mathbf{E}[X_t] : t \in T_m\right\},$$

where T_m is the set of stopping rules for $\widehat{X}_1, \ldots, \widehat{X}_m$. Then, for m large enough, there is a sequence of random variables $\widehat{X}_1, \ldots, \widehat{X}_m$ such that

- $\mathbf{E}\left[\max_{i \le m} \widehat{X}_i\right] \ge 1/m^3$, *and*
- $\mathbf{E}\left[\max_{i \le m} \widehat{X}_i\right] \ge (a_m - 1/m^3) \sup\left\{\mathbf{E}[\widehat{X}_t] : t \in T_m\right\}$.

Proof. In Proposition 4.4 [98], it is shown that that for any ε' sufficiently small, there is a random variable $\widehat{X}$ with $\widehat{p}_0 = \mathbf{P}(\widehat{X} = 0)$, $\widehat{p}_j = \mathbf{P}(\widehat{X} = V_j(\widehat{X}))$ for $j = 0, \ldots, m-2$, $\mathbf{P}(\widehat{X} = V_{m-1}(\widehat{X})) = \widehat{p}_{m-1} - \varepsilon'$ and $\mathbf{P}(\widehat{X} = 1) = \varepsilon'$ such that $\mathbf{E}[\max_{i \le m} \widehat{X}_i] \ge (a_m - \varepsilon') \sup\left\{\mathbf{E}[\widehat{X}_t] : t \in \widehat{T}_m\right\}$, where $\widehat{X}_1, \ldots, \widehat{X}_m$ are m independent copies of $\widehat{X}$. Here $V_j(\widehat{X}) = \mathbf{E}[\widehat{X} \wedge \mathbf{E}[V_{j-1}(\widehat{X})]]$ corresponds to the optimal value computed via dynamic programming and one can show that $\sup\left\{\mathbf{E}[\widehat{X}_t] : t \in \widehat{T}_m\right\} = V_m(\widehat{X})$ (see Lemma 2.1 in [98]). We only need to show that we can choose $\varepsilon' = 1/m^3$. The probabilities $\widehat{p}_0, \ldots, \widehat{p}_{m-1}$ are computed as follows: Let $\widehat{s}_j = (\eta_{j,m}(\alpha_m))^{1/m}$ for $j = 1, \ldots, n-2$ where $\alpha_m \in (0,1)$ is the (unique) solution of $\eta_{m-1,m}(\alpha_m) = 1$, then $\widehat{p}_0 = \widehat{s}_0$, $\widehat{p}_j = \widehat{s}_j - \widehat{s}_{j-1}$ for $j = 1, \ldots, n-2$ and $\widehat{p}_{n-1} = 1 - \widehat{s}_{n-2}$. One can show that $\widehat{s}_{m-2} = (1 - 1/m)^{1/(m-1)} (1 - \alpha_m/m)^{1/(m-1)}$ and α_m holds $1/(3e) \le \alpha_m \le 1/(e-1)$ (see Proposition 3.6 in [98]). For m large we have

$$e^{-1/(m-1)} \le \widehat{s}_{m-2} \le e^{-1/(3em^2)}$$

then $\widehat{p}_{m-1} = 1 - \widehat{s}_{m-2} \ge 1 - e^{-1/(m-1)} \ge 1/m^2$ for m large. Thus we can set $\varepsilon' = 1/m^3$ and $\widehat{p}_{m-1} - \varepsilon' > 0$ and the rest of the proof follows. Furthermore, $\mathbf{E}[\max_{i \le m} X_i] \ge \varepsilon' \cdot 1 \ge 1/m^3$. □

4.A.6 Missing Proofs From Section 4.7

Proof of Lemma 4.11. For $p \ge p^*$ and $\ell = 0, 1, \ldots, 4$, we calculate tight lower bounds for the expression in the letf-hand side of the inequality in the claim, and we show that these lower bounds are at least one, with the lower bound attaining equality with 1 for $\ell = 1, 2$.

For $\ell \geq 5$ we can generalize the previous bounds and show a universal lower bound of at least 1.

- For $\ell = 0$, we have

$$\int_{p^{1/(1-p)}}^{1} \frac{1}{t^p}\,\mathrm{d}t = \frac{1}{1-p}\,(1-p) = 1 = (1-p)^0.$$

- For $\ell = 1$, we have

$$\int_{p^{1/(1-p)}}^{1} \frac{(1-t)}{t^p}\,\mathrm{d}t = 1 - \int_{p^{1/(1-p)}}^{1} t^{1-p}\,\mathrm{d}t = 1 - \frac{1}{2-p}\left(1 - p^{(2-p)/(1-p)}\right).$$

  The last value is at least $1-p$ if an only if $p(2-p) \geq 1-p^{(2-p)/(1-p)}$ iff $p^{(2-p)/(1-p)} \geq (1-p)^2$. The last inequality holds iff $p \geq p^* \approx 0.594134$ where p^* is computed numerically by solving $(1-p)^2 = p^{(2-p)/(1-p)}$.

- For $\ell = 2$, we use the approximation $p^{1/(1-p)} \leq (1+p)/(2e)$ that follows from the concavity of the function $p^{1/(1-p)}$ and the first-order approximation of the function at $p = 1$. With this we can lower bound the integral

$$\begin{aligned}
\int_{p^{1/(1-p)}}^{1} \frac{(1-t)^2}{t^p}\,\mathrm{d}t &\geq \int_{(1+p)/(2e)}^{1} \frac{(1-t)^2}{t^p}\,\mathrm{d}t \\
&= \int_{0}^{1-(1+p)/(2e)} u^2(1-u)^{-p}\,\mathrm{d}u \qquad \text{(change of variable } u = 1-t\text{)} \\
&\geq \int_{0}^{1-(1+p)/(2e)} u^2\left(1 + pu + p(p+1)\frac{u^2}{2}\right)\,\mathrm{d}u \\
&\qquad\qquad \text{(Using the series } (1-u)^{-p} = \textstyle\sum_{k\geq 0}\binom{-p}{k}(-u)^k\text{)} \\
&= \frac{1}{3}\left(1 - \frac{1+p}{2e}\right)^3 + \frac{p}{4}\left(1 - \frac{1+p}{2e}\right)^4 + \frac{p(p+1)}{10}\left(1 - \frac{1+p}{2e}\right)^5.
\end{aligned}$$

  By solving the polynomial we see that the last expression is $\geq (1-p)^2$ if and only if $p \geq 0.585395$, thus the inequality holds for $p \geq p^*$.

- For $\ell = 3, 4$ we can use a similar approach to get

$$\int_{p^{1/(1-p)}}^{1} \frac{(1-p)^\ell}{t^p}\,\mathrm{d}t \geq \frac{1}{\ell+1}\left(1-\frac{1+p}{2e}\right)^{\ell+1} + \frac{p}{\ell+2}\left(1-\frac{1+p}{2e}\right)^{\ell+2}.$$

  The last expression is $\geq (1-p)^\ell$ for $\ell = 3, 4$ if and only if $p \geq 0.559826$.

- For $\ell \geq 5$, we have

$$\int_{p^{1/(1-p)}}^{1} \frac{(1-t)^\ell}{t^p}\,\mathrm{d}t \geq \frac{(1-(1+p)/(2e))^{\ell+1}}{\ell+1}.$$

  We aim to show that $(1-(1+p)/(2e))^{\ell+1}/(\ell+1) \geq (1-p)^\ell$. This is equivalent to show that $\left(\frac{1-(1+p)/(2e)}{1-p}\right)^\ell \left(1-\frac{1+p}{2e}\right) \geq \ell+1$.

  Note that the function $f(p) = (1-(1+p)/(2e))/(1-p)$ is increasing since $f'(p) = (1-1/e)/(1-p)^2 > 0$. For $\overline{p} = (2e-1)/(4e-3) \approx 0.56351$ we have $f(\overline{p}) = 2-1/e$. Thus for $p \geq p^* > \overline{p}$ and $\ell = 5$ we have $f(p)^5\,(1-(1+p)/(2e)) \geq (2-1/e)^5(1-1/e) \geq 7.32 \geq 6$. By an inductive argument, we can show that $f(p)^\ell\,(1-(1+p)/(2e)) \geq \ell+1$ for any $\ell \geq 5$ and this finishes the proof.

□

CHAPTER 5

CONCLUSIONS AND FUTURE DIRECTIONS

Resource allocation has become highly prominent in the new era of e-commerce. This thesis explored multiple models that aim to address sources of uncertainty faced by practitioners/DMs. We explored unstudied tensions observed in practical applications and provided new models. We hope these models will give managerial insight and become the starting point for more complex and realistic models.

Each chapter summarizes model-specific questions, and we will not repeat those questions here. However, on a broader level, there are further open questions:

- We understand better how to benchmark algorithms in sequential models. We learned that this task depends strongly on the model and what we want to communicate from it. In this work, most of the objectives were either linear (regret in Chapter 2 and cost in Chapter 3), or completely multi-objective (robust ratio in Chapter 4). We can draw a parallel between ℓ_1 norm for the linear costs and ℓ_∞ for the multi-objective metric. Many possibilities appear between these two extremes. One of them is the ℓ_2 norm, which can be interpreted as a deviation from a path. Exploring better benchmarks is a neverending task, and it goes hand-by-hand with the model's design.

- The models explored in this thesis tend to be either adversarial/robust or stochastic. Unfortunately, nature is possibly less inclined to any of these extremes. So then, the natural question is how to capture new, more realistic sources of uncertainty? An approach found in the literature is distributionally robust optimization, which combines both extremes. However, this approach poses other burdens, for example,

the choice of a distance between distributions.

- Besides the model explored in Chapter 2, most of the models studied in this thesis assume some mild knowledge from the input data. If enough historical data is available, this is potentially a realistic assumption. However, exploring the integration of learning from the data and decision-making would be ideal. In particular, in robust settings with multiple competing objectives, such as the model from Chapter 2.

REFERENCES

[1] https://www.reveliolabs.com/news/business/what-are-the-brainiest-companies-alternative-careers-for-phds/.

[2] https://www.statista.com/statistics/233725/development-of-amazon-web-services-revenue/.

[3] https://www.wpoven.com/blog/microsoft-azure-market-share/.

[4] https://ubuntu.com/blog/cloud-optimized-linux-kernels.

[5] https://www.nytimes.com/2020/02/27/technology/cloud-computing-energy-usage.html.

[6] https://www.docker.com/.

[7] https://www.statista.com/statistics/271258/facebooks-advertising-revenue-worldwide/.

[8] J. Abernethy, P. L. Bartlett, A. Rakhlin, and A. Tewari, "Optimal strategies and minimax lower bounds for online convex games," 2008.

[9] D. Adelman, "Dynamic bid prices in revenue management," *Operations Research*, vol. 55, no. 4, pp. 647–661, 2007.

[10] S. Agrawal, Z. Wang, and Y. Ye, "A dynamic near-optimal algorithm for online linear programming," *Operations Research*, vol. 62, no. 4, pp. 876–890, 2014.

[11] S. Alaei, M. Hajiaghayi, and V. Liaghat, "Online prophet-inequality matching with applications to ad allocation," in *Proceedings of the 13th ACM Conference on Electronic Commerce*, 2012, pp. 18–35.

[12] ——, "The online stochastic generalized assignment problem," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Springer, 2013, pp. 11–25.

[13] S. Albers, "Online algorithms: A survey," *Mathematical Programming*, vol. 97, no. 1, pp. 3–26, 2003.

[14] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[15] R. Anand, D. Aggarwal, and V. Kumar, "A comparative analysis of optimization solvers," *Journal of Statistics and Management Systems*, vol. 20, no. 4, pp. 623–635, 2017.

[16] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: A meta-algorithm and applications.," *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.

[17] M. Babaioff, J. Hartline, and R. Kleinberg, "Selling banner ads: Online algorithms with buyback," in *Fourth Workshop on Ad Auctions*, 2008.

[18] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, "A knapsack secretary problem with applications," in *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, Springer, 2007, pp. 16–28.

[19] ——, "Online auctions and generalized secretary problems," *ACM SIGecom Exchanges*, vol. 7, no. 2, pp. 1–11, 2008.

[20] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin, "A new and improved algorithm for online bin packing," in *26th Annual European Symposium on Algorithms (ESA 2018)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[21] J. Balogh, J. Békési, and G. Galambos, "New lower bounds for certain classes of bin packing algorithms," *Theoretical Computer Science*, vol. 440, pp. 1–13, 2012.

[22] S. Balseiro and D. Brown, "Approximations to stochastic dynamic programs via information relaxation duality," *Operations Research*, vol. 67, pp. 577–597, 2019.

[23] M. Bateni, M. Hajiaghayi, and M. Zadimoghaddam, "Submodular secretary problem and extensions," *ACM Transactions on Algorithms (TALG)*, vol. 9, no. 4, pp. 1–23, 2013.

[24] A. Ben-Tal and A. Nemirovski, *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Siam, 2001, vol. 2.

[25] A. Bhalgat, A. Goel, and S. Khanna, "Improved approximation results for stochastic knapsack problems," in *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, SIAM, 2011, pp. 1647–1665.

[26] D. Blado, W. Hu, and A. Toriello, "Semi-Infinite Relaxations for the Dynamic Knapsack Problem with Stochastic Item Sizes," *SIAM Journal on Optimization*, vol. 26, pp. 1625–1648, 2016.

[27] D. Blado and A. Toriello, "Relaxation Analysis for the Dynamic Knapsack Problem with Stochastic Item Sizes," *SIAM Journal on Optimization*, vol. 29, pp. 1–30, 2019.

[28] ——, " A Column and Constraint Generation Algorithm for the Dynamic Knapsack Problem with Stochastic Item Sizes," *Mathematical Programming Computation*, 2020, Forthcoming.

[29] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. cambridge university press, 2005.

[30] S. Boucheron, G. Lugosi, and P. Massart, *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

[31] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[32] F. T. Bruss, "On an optimal selection problem of cowan and zabczyk," *Journal of Applied Probability*, pp. 918–928, 1987.

[33] S. Bubeck, N. Cesa-Bianchi, *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[34] N. Buchbinder, K. Jain, and M. Singh, "Secretary problems via linear programming," *Mathematics of Operations Research*, vol. 39, no. 1, pp. 190–206, 2014.

[35] N. Buchbinder, J. S. Naor, *et al.*, "The design of competitive online algorithms via a primal–dual approach," *Foundations and Trends® in Theoretical Computer Science*, vol. 3, no. 2–3, pp. 93–263, 2009.

[36] G. Campbell and S. M. Samuels, "Choosing the best of the current crop," *Advances in Applied Probability*, vol. 13, no. 3, pp. 510–532, 1981.

[37] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

[38] T. H. Chan, F. Chen, and S. H.-C. Jiang, "Revealing optimal thresholds for generalized secretary problem via continuous lp: Impacts on online k-item auction and bipartite k-matching with random arrival order," in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2014, pp. 1169–1188.

[39] H. Cheng and E. Cantú-Paz, "Personalized click prediction in sponsored search," in *Proceedings of the third ACM international conference on Web search and data mining*, 2010, pp. 351–360.

[40] C.-H. Cho and H. J. Cheon, "Why do people avoid advertising on the internet?" *Journal of advertising*, vol. 33, no. 4, pp. 89–97, 2004.

[41] H. Choi, C. F. Mela, S. R. Balseiro, and A. Leary, "Online display advertising markets: A literature review and future directions," *Information Systems Research*, vol. 31, no. 2, pp. 556–575, 2020.

[42] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali, *Multidimensional bin packing and other related problems: A survey*, 2016.

[43] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.

[44] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson, "An application of bin-packing to multiprocessor scheduling," *SIAM Journal on Computing*, vol. 7, no. 1, pp. 1–17, 1978.

[45] E. G. Coffman Jr, C. Courcoubetis, M. R. Garey, D. S. Johnson, P. W. Shor, R. R. Weber, and M. Yannakakis, "Bin packing with discrete item sizes, part i: Perfect packing theorems and the average case behavior of optimal packings," *SIAM Journal on Discrete Mathematics*, vol. 13, no. 3, pp. 384–402, 2000.

[46] E. G. Coffman Jr and G. S. Lueker, "Approximation algorithms for extensible bin packing," in *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, 2001, pp. 586–588.

[47] E. G. Coffman Jr, K. So, M. Hofri, and A. Yao, "A stochastic model of bin-packing," *Information and Control*, vol. 44, no. 2, pp. 105–115, 1980.

[48] E. Coffman Jr, M. Garey, and D. Johnson, "Approximation algorithms for bin packing: A survey," *Approximation algorithms for NP-hard problems*, pp. 46–93, 1996.

[49] M. C. Cohen, P. W. Keller, V. Mirrokni, and M. Zadimoghaddam, "Overcommitment in cloud services: Bin packing with chance constraints," *Management Science*, 2019.

[50] J. Correa, A. Cristi, B. Epstein, and J. Soto, "Sample-driven optimal stopping: From the secretary problem to the iid prophet inequality," *arXiv preprint arXiv:2011.06516*, 2020.

[51] J. Correa, A. Cristi, L. Feuilloley, T. Oosterwijk, and A. Tsigonias-Dimitriadis, "The secretary problem with independent sampling," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, SIAM, 2021, pp. 2047–2058.

[52] R. Cowan and J. Zabczyk, "An optimal selection problem associated with the poisson process," *Theory of Probability & Its Applications*, vol. 23, no. 3, pp. 584–592, 1979.

[53] J. Csirik, D. S. Johnson, C. Kenyon, J. B. Orlin, P. W. Shor, and R. R. Weber, "On the sum-of-squares algorithm for bin packing," *Journal of the ACM (JACM)*, vol. 53, no. 1, pp. 1–65, 2006.

[54] C. Curino, D. E. Difallah, C. Douglas, S. Krishnan, R. Ramakrishnan, and S. Rao, "Reservation-based scheduling: If you're late don't blame us!" In *Proceedings of the ACM Symposium on Cloud Computing*, ACM, 2014, pp. 1–14.

[55] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Efficient datacenter resource utilization through cloud resource overcommitment," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2015, pp. 330–335.

[56] D. P. De Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations research*, vol. 51, no. 6, pp. 850–865, 2003.

[57] W. F. De La Vega and G. S. Lueker, "Bin packing can be solved within 1+ ε in linear time," *Combinatorica*, vol. 1, no. 4, pp. 349–355, 1981.

[58] B. C. Dean, M. X. Goemans, and J. Vondrák, "Adaptivity and approximation for stochastic packing problems," in *SODA*, vol. 5, 2005, pp. 395–404.

[59] ——, "Approximating the stochastic knapsack problem: The benefit of adaptivity," *Mathematics of Operations Research*, vol. 33, no. 4, pp. 945–964, 2008.

[60] P. Dell'Olmo, H. Kellerer, M. G. Speranza, and Z. Tuza, "A 1312 approximation algorithm for bin packing with extendable bins," *Information Processing Letters*, vol. 65, no. 5, pp. 229–233, 1998.

[61] M. L. Della Vedova, D. Tessera, and M. C. Calzarossa, "Probabilistic provisioning and scheduling in uncertain cloud environments," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, IEEE, 2016, pp. 797–803.

[62] B. T. Denton, A. J. Miller, H. J. Balasubramanian, and T. R. Huschka, "Optimal allocation of surgery blocks to operating rooms under uncertainty," *Operations research*, vol. 58, no. 4-part-1, pp. 802–816, 2010.

[63] C. Derman, G. J. Lieberman, and S. M. Ross, "A renewal decision problem," *Management Science*, vol. 24, no. 5, pp. 554–561, 1978.

[64] N. R. Devanur and T. P. Hayes, "The adwords problem: Online keyword matching with budgeted bidders under random permutations," in *Proceedings of the 10th ACM conference on Electronic commerce*, 2009, pp. 71–78.

[65] N. R. Devanur and S. M. Kakade, "The price of truthfulness for pay-per-click auctions," in *Proceedings of the 10th ACM conference on Electronic commerce*, 2009, pp. 99–106.

[66] F. Dexter, A. Macario, and R. D. Traub, "Which algorithm for scheduling add-on elective cases maximizes operating room utilization? use of bin packing algorithms and fuzzy constraints in operating room management," *Anesthesiology: The Journal of the American Society of Anesthesiologists*, vol. 91, no. 5, pp. 1491–1491, 1999.

[67] X. Drèze and F.-X. Hussherr, "Internet advertising: Is anybody watching?" *Journal of interactive marketing*, vol. 17, no. 4, pp. 8–23, 2003.

[68] P. Dütting, S. Lattanzi, R. Paes Leme, and S. Vassilvitskii, "Secretaries with advice," in *Proceedings of the 22nd ACM Conference on Economics and Computation*, 2021, pp. 409–429.

[69] E. B. Dynkin, "The optimum choice of the instant for stopping a markov process," *Soviet Mathematics*, vol. 4, pp. 627–629, 1963.

[70] B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords," *American economic review*, vol. 97, no. 1, pp. 242–259, 2007.

[71] A. Farahat and M. C. Bailey, "How effective is targeted advertising?" In *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 111–120.

[72] M. Feldman, O. Svensson, and R. Zenklusen, "A simple o (log log (rank))-competitive algorithm for the matroid secretary problem," in *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, SIAM, 2014, pp. 1189–1201.

[73] W. Feller, "An introduction to probability theory and its applications," 1957.

[74] T. S. Ferguson *et al.*, "Who solved the secretary problem?" *Statistical science*, vol. 4, no. 3, pp. 282–289, 1989.

[75] P. Freeman, "The secretary problem and its extensions: A review," *International Statistical Review/Revue Internationale de Statistique*, pp. 189–206, 1983.

[76] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[77] ——, "Adaptive game playing using multiplicative weights," *Games and Economic Behavior*, vol. 29, no. 1-2, pp. 79–103, 1999.

[78] K. Fridgeirsdottir and S. Najafi-Asadolahi, "Cost-per-impression pricing for display advertising," *Operations Research*, vol. 66, no. 3, pp. 653–672, 2018.

[79] H. Fu, J. Li, and P. Xu, "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, 2018.

[80] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C.-C. Yao, "Resource constrained scheduling as generalized bin packing," *Journal of Combinatorial Theory, Series A*, vol. 21, no. 3, pp. 257–298, 1976.

[81] A. Gera and C. H. Xia, "Learning curves and stochastic models for pricing and provisioning cloud computing services," *Service Science*, vol. 3, no. 1, pp. 99–109, 2011.

[82] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types.," in *Nsdi*, vol. 11, 2011, pp. 24–24.

[83] J. P. Gilbert and F. Mosteller, "Recognizing the maximum of a sequence," *Journal of the American Statistical Association*, pp. 35–73, 1966.

[84] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations research*, vol. 9, no. 6, pp. 849–859, 1961.

[85] A. Goel and P. Indyk, "Stochastic load balancing and related problems," in *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, IEEE, 1999, pp. 579–586.

[86] A. Gordon, M. Hines, D. Da Silva, M. Ben-Yehuda, M. Silva, and G. Lizarraga, "Ginkgo: Automated, application-driven memory overcommitment for cloud computing," *Proc. RESoLVE*, 2011.

[87] V. Goyal and R. Udwani, "Online matching with stochastic rewards: Optimal competitive ratio via path based formulation," in *Proceedings of the 21st ACM Conference on Economics and Computation*, 2020, pp. 791–791.

[88] R. Grandl, M. Chowdhury, A. Akella, and G. Ananthanarayanan, "Altruistic scheduling in multi-resource clusters.," in *OSDI*, 2016, pp. 65–80.

[89] A. Gupta, R. Krishnaswamy, M. Molinaro, and R. Ravi, "Approximation algorithms for correlated knapsacks and non-martingale bandits," in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, IEEE, 2011, pp. 827–836.

[90] V. Gupta and A. Radovanovic, "Lagrangian-based online stochastic bin packing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1, pp. 467–468, 2015.

[91] V. Gupta and A. Radovanović, "Interior-point-based online stochastic bin packing," *Operations Research*, vol. 68, no. 5, pp. 1474–1492, 2020.

[92] S. Gusein-Zade, "The problem of choice and the sptimal stopping rule for a sequence of independent trials," *Theory of Probability & Its Applications*, vol. 11, no. 3, pp. 472–476, 1966.

[93] E. C. Hall and R. M. Willett, "Online convex optimization in dynamic environments," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 647–662, 2015.

[94] W. B. Haskell and R. Jain, "Stochastic dominance-constrained markov decision processes," *SIAM Journal on Control and Optimization*, vol. 51, no. 1, pp. 273–303, 2013.

[95] ——, "A convex analytic approach to risk-aware markov decision processes," *SIAM Journal on Control and Optimization*, vol. 53, no. 3, pp. 1569–1598, 2015.

[96] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[97] E. Hazan, "Introduction to online convex optimization," *arXiv preprint arXiv:1909.05207*, 2019.

[98] T. P. Hill and R. P. Kertz, "Comparisons of stop rule and supremum expectations of iid random variables," *The Annals of Probability*, pp. 336–345, 1982.

[99] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center.," in *NSDI*, vol. 11, 2011, pp. 22–22.

[100] M. Hlynka and J. Sheahan, "The secretary problem for a random walk," *Stochastic Processes and their applications*, vol. 28, no. 2, pp. 317–325, 1988.

[101] R. Hoberg and T. Rothvoss, "A logarithmic additive integrality gap for bin packing," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2017, pp. 2616–2625.

[102] J. Jiang, W. Ma, and J. Zhang, "Tight guarantees for multi-unit prophet inequalities and online stochastic knapsack," *arXiv preprint arXiv:2107.02058*, 2021.

[103] D. S. Johnson, "Fast algorithms for bin packing," *Journal of Computer and System Sciences*, vol. 8, no. 3, pp. 272–314, 1974.

[104] S. A. Jyothi, C. Curino, I. Menache, S. M. Narayanamurthy, A. Tumanov, J. Yaniv, R. Mavlyutov, I. Goiri, S. Krishnan, J. Kulkarni, *et al.*, "Morpheus: Towards automated slos for enterprise clusters.," in *OSDI*, 2016, pp. 117–134.

[105] H. Kaplan, D. Naori, and D. Raz, "Competitive analysis with a sample and the secretary problem," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2020, pp. 2082–2095.

[106] N. Karmarkar and R. M. Karp, "An efficient approximation scheme for the one-dimensional bin-packing problem," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, IEEE, 1982, pp. 312–320.

[107] R. P. Kertz, "Stop rule and supremum expectations of iid random variables: A complete comparison by conjugate duality," *Journal of multivariate analysis*, vol. 19, no. 1, pp. 88–112, 1986.

[108] T. Kesselheim, A. Tönnis, K. Radke, and B. Vöcking, "Primal beats dual on online packing lps in the random-order model," in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, 2014, pp. 303–312.

[109] J. Kleinberg, Y. Rabani, and É. Tardos, "Allocating bandwidth for bursty connections," *SIAM Journal on Computing*, vol. 30, no. 1, pp. 191–217, 2000.

[110] R. Kleinberg, "A multiple-choice secretary algorithm with applications to online auctions," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, Citeseer, 2005, pp. 630–631.

[111] A. Kleywegt and J. Papastavrou, "The Dynamic and Stochastic Knapsack Problem," *Operations Research*, vol. 46, pp. 17–35, 1998.

[112] ——, "The Dynamic and Stochastic Knapsack Problem with Random Sized Items," *Operations Research*, vol. 49, pp. 26–41, 2001.

[113] A. J. Kleywegt and J. D. Papastavrou, "The dynamic and stochastic knapsack problem," *Operations research*, vol. 46, no. 1, pp. 17–35, 1998.

[114] N. Korula and M. Pál, "Algorithms for secretary problems on graphs and hypergraphs," in *International Colloquium on Automata, Languages, and Programming*, Springer, 2009, pp. 508–520.

[115] O. Lachish, "O (log log rank) competitive ratio for the matroid secretary problem," in *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, IEEE, 2014, pp. 326–335.

[116] A. Levin, "Approximation schemes for the generalized extensible bin packing problem," *arXiv preprint arXiv:1905.09750*, 2019.

[117] A. Levin and A. Vainer, "Adaptivity in the stochastic blackjack knapsack problem," *Theoretical Computer Science*, vol. 516, pp. 121–126, 2014.

[118] C.-p. Li and M. J. Neely, "Energy-optimal scheduling with dynamic channel acquisition in wireless downlinks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 4, pp. 527–539, 2009.

[119] J. Li and W. Yuan, "Stochastic combinatorial optimization via poisson approximation," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, ACM, 2013, pp. 971–980.

[120] D. V. Lindley, "Dynamic programming and decision theory," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 10, no. 1, pp. 39–51, 1961.

[121] B. Lucier, "An economic view of prophet inequalities," *ACM SIGecom Exchanges*, vol. 16, no. 1, pp. 24–47, 2017.

[122] W. Ma, "Improvements and generalizations of stochastic knapsack and markovian bandits approximation algorithms," *Mathematics of Operations Research*, vol. 43, no. 3, pp. 789–812, 2018.

[123] M. Macías and J. Guitart, "A genetic model for pricing in cloud computing markets," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ACM, 2011, pp. 113–118.

[124] S. T. Maguluri and R. Srikant, "Scheduling jobs with unknown duration in clouds," *IEEE/ACM Transactions On Networking*, vol. 22, no. 6, pp. 1938–1951, 2014.

[125] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *2012 Proceedings IEEE Infocom*, IEEE, 2012, pp. 702–710.

[126] ——, "Heavy traffic optimal resource allocation algorithms for cloud computing clusters," *Performance Evaluation*, vol. 81, pp. 20–39, 2014.

[127] A. Manne, "Linear Programming and Sequential Decisions," *Management Science*, vol. 6, pp. 259–267, 1960.

[128] A. Marchetti-Spaccamela and C. Vercellis, "Stochastic on-line knapsack problems," *Mathematical Programming*, vol. 68, no. 1, pp. 73–104, 1995.

[129] A. Mehta, B. Waggoner, and M. Zadimoghaddam, "Online stochastic matching with unequal probabilities," in *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, SIAM, 2014, pp. 1388–1404.

[130] I. Menache and M. Singh, "Online caching with convex costs," in *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures*, ACM, 2015, pp. 46–54.

[131] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[132] A. Mokhtari, S. Shahrampour, A. Jadbabaie, and A. Ribeiro, "Online optimization in dynamic environments: Improved regret rates for strongly convex problems," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*, IEEE, 2016, pp. 7195–7201.

[133] V. Narasayya, S. Das, M. Syamala, B. Chandramouli, and S. Chaudhuri, "SQLVM: Performance isolation in multi-tenant relational database-as-a-service," 2013.

[134] V. Narasayya, I. Menache, M. Singh, F. Li, M. Syamala, and S. Chaudhuri, "Sharing buffer pool memory in multi-tenant relational database-as-a-service," *Proceedings of the VLDB Endowment*, vol. 8, no. 7, pp. 726–737, 2015.

[135] M. J. Neely, "Optimal energy and delay tradeoffs for multiuser wireless downlinks," *IEEE Transactions on Information Theory*, vol. 53, no. 9, pp. 3095–3113, 2007.

[136] ——, "Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 5, pp. 1188–1199, 2008.

[137] J. Papastavrou, S. Rajagopalan, and A. Kleywegt, "The Dynamic and Stochastic Knapsack Problem with Deadlines," *Management Science*, vol. 42, pp. 1706–1718, 1996.

[138] M. Passacantando, D. Ardagna, and A. Savi, "Service provisioning problem in cloud and multi-cloud systems," *INFORMS Journal on Computing*, vol. 28, no. 2, pp. 265–277, 2016.

[139] S. Perez-Salazar, I. Menache, M. Singh, and A. Toriello, "Dynamic resource allocation in the cloud with near-optimal efficiency," *Operations Research*, 2021.

[140] S. Perez-Salazar, M. Singh, and A. Toriello, "Robust online selection with uncertain offer acceptance," *arXiv preprint arXiv:2112.00842*, 2021.

[141] ——, "Adaptive bin packing with overflow," *Mathematics of Operations Research*, 2022.

[142] S. A. Plotkin, D. B. Shmoys, and É. Tardos, "Fast approximation algorithms for fractional packing and covering problems," *Mathematics of Operations Research*, vol. 20, no. 2, pp. 257–301, 1995.

[143] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[144] E. L. Presman and I. M. Sonin, "The best choice problem for a random number of objects," *Theory of Probability & Its Applications*, vol. 17, no. 4, pp. 657–668, 1973.

[145] E. Pujol, O. Hohlfeld, and A. Feldmann, "Annoyed users: Ads and ad-block usage in the wild," in *Proceedings of the 2015 Internet Measurement Conference*, 2015, pp. 93–106.

[146] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[147] M. Raghavan, S. Barocas, J. M. Kleinberg, and K. Levy, "Mitigating bias in algorithmic employment screening: Evaluating claims and practices.," 2019.

[148] J. Rasley, K. Karanasos, S. Kandula, R. Fonseca, M. Vojnovic, and S. Rao, "Efficient queue management for cluster scheduling," in *Proceedings of the Eleventh European Conference on Computer Systems*, ACM, 2016, p. 36.

[149] W. T. Rhee, "Optimal bin packing with items of random sizes," *Mathematics of Operations Research*, vol. 13, no. 1, pp. 140–151, 1988.

[150] S. M. Ross, *Stochastic processes*. Wiley New York, 1996, vol. 2.

[151] T. Rothvoß, "Approximating bin packing within o(log opt* log log opt) bins," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, IEEE, 2013, pp. 20–29.

[152] G. Sagnol, D. S. genannt Waldschmidt, and A. Tesch, "The price of fixed assignments in stochastic extensible bin packing," in *International Workshop on Approximation and Online Algorithms*, Springer, 2018, pp. 327–347.

[153] J. Salem and S. Gupta, "Closing the gap: Group-aware parallelization for the secretary problem with biased evaluations," *Available at SSRN 3444283*, 2019.

[154] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[155] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[156] B. Sharma, R. K. Thulasiram, P. Thulasiraman, S. K. Garg, and R. Buyya, "Pricing cloud compute commodities: A novel financial economic model," in *Proceedings of the 2012 12th IEEE/ACM international symposium on cluster, cloud and grid computing (ccgrid 2012)*, IEEE Computer Society, 2012, pp. 451–457.

[157] H. Shirani-Mehr, G. Caire, and M. J. Neely, "Mimo downlink scheduling with non-perfect channel state knowledge," *IEEE Transactions on Communications*, vol. 58, no. 7, pp. 2055–2066, 2010.

[158] P. W. Shor, "The average-case analysis of some on-line algorithms for bin packing," *Combinatorica*, vol. 6, no. 2, pp. 179–200, 1986.

[159] ——, "How to pack better than best fit: Tight bounds for average-case online bin packing," in *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, IEEE, 1991, pp. 752–759.

[160] M. Sipser, *Introduction to the theory of computation*. PWS Publishing Company, 1997, ISBN: 978-0-534-94728-6.

[161] M. Smith, "A secretary problem with uncertain employment," *Journal of applied probability*, pp. 620–624, 1975.

[162] J. A. Soto, "Matroid secretary problem in the random-assignment model," *SIAM Journal on Computing*, vol. 42, no. 1, pp. 178–211, 2013.

[163] R. Srikant and L. Ying, *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press, 2013.

[164] M. Tamaki, "A secretary problem with uncertain employment and best choice of available candidates," *Operations Research*, vol. 39, no. 2, pp. 274–284, 1991.

[165] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *29th IEEE Conference on Decision and Control*, IEEE, 1990, pp. 2130–2132.

[166] ——, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Transactions on Information Theory*, vol. 39, no. 2, pp. 466–478, 1993.

[167] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979.

[168] R. J. Vanderbei, "The optimal choice of a subset of a population," *Mathematics of Operations Research*, vol. 5, no. 4, pp. 481–486, 1980.

[169] ——, "The postdoc variant of the secretary problem," Technical report, Princeton University, Tech. Rep., 2012.

[170] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, ACM, 2013, p. 5.

[171] B. Vijayakumar, P. J. Parikh, R. Scott, A. Barnes, and J. Gallimore, "A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital," *European Journal of Operational Research*, vol. 224, no. 3, pp. 583–591, 2013.

[172] A. van Vliet, "An improved lower bound for on-line bin packing algorithms," *Information processing letters*, vol. 43, no. 5, pp. 277–284, 1992.

[173] D. Wilcox, A. McNabb, and K. Seppi, "Solving virtual machine packing with a reordering grouping genetic algorithm," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, IEEE, 2011, pp. 362–369.

[174] I. D. Wilson and P. A. Roach, "Principles of combinatorial optimization applied to container-ship stowage planning," *Journal of Heuristics*, vol. 5, no. 4, pp. 403–418, 1999.

[175] A. C.-C. Yao, "New algorithms for bin packing," *Journal of the ACM (JACM)*, vol. 27, no. 2, pp. 207–227, 1980.

[176] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in *Proceedings of the 5th European conference on Computer systems*, 2010, pp. 265–278.

[177] L. Zhang, T. Yang, J. Yi, J. Rong, and Z.-H. Zhou, "Improved dynamic regret for non-degenerate functions," in *Advances in Neural Information Processing Systems*, 2017, pp. 732–741.

[178] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 928–936.

VITA

Sebastian Walter Perez Salazar was born on August 20, 1990, in Santiago, Chile. He is the son of Arturo Perez and Maria Salazar. In the classrooms of Universidad de Chile, Sebastian discovered his interest in mathematics. He enjoyed solving problems and unraveling the mysteries of math while sharing ideas with others. His introduction to the academic world was thanks to his former advisor Iván Rapaport while working on communication complexity problems in distributed systems. Sebastian graduated from Universidad de Chile with a B.E. and M.E. in Applied Math (Ingeniería Matemática). Upon arriving at Georgia Tech, Sebastian discovered his passion for optimization and its applications to resource allocation problems. He obtained his Ph.D. in the Algorithms, Combinatorics, & Optimization (ACO) program at Georgia Tech in 2022. He was extremely fortunate to be advised by Mohit Singh and Alejandro Toriello.

Sebastian became an Assistant Professor in the Department of Computational Applied Mathematics and Operations Research at Rice University in August 2022.