Online Allocation Algorithms with Applications in Computational Advertising

by

Morteza Zadimoghaddam

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2014

Author .....
Department of Electrical Engineering and Computer Science
January 24, 2014

Certified by .....
Erik D. Demaine
Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

To Faezeh

Online Allocation Algorithms with Applications in Computational Advertising

by

Morteza Zadimoghaddam

Submitted to the Department of Electrical Engineering and Computer Science
on January 24, 2014, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

Over the last few decades, a wide variety of allocation markets emerged from the Internet and introduced interesting algorithmic challenges, e.g., ad auctions, online dating markets, matching skilled workers to jobs, etc. I focus on the use of allocation algorithms in computational advertising as it is the quintessential application of my research. I will also touch on the classic secretary problem with submodular utility functions, and show that how it is related to advertiser's optimization problem in computational advertising applications. In all these practical situations, we should focus on solving the allocation problems in an online setting since the input is being revealed during the course of the algorithm, and at the same time we should make irrevocable decisions. We can formalize these types of computational advertising problems as follows. We are given a set of online items, arriving one by one, and a set of advertisers where each advertiser specifies how much she wants to pay for each of the online items. The goal is to allocate online items to advertisers to maximize some objective function like the total revenue, or the total quality of the allocation. There are two main classes of extensively studied problems in this context: *budgeted allocation* (a.k.a. the *adwords* problem) and *display ad* problems. Each advertiser is constrained by an overall budget limit, the maximum total amount she can pay in the first class, and by some positive integer capacity, the maximum number of online items we can assign to her in the second class.

Thesis Supervisor: Erik D. Demaine
Title: Professor

Acknowledgments

As I am approaching the end of my doctorate studies at MIT, I vividly remember the people who had a great influence on my life. I am indebted to them for their help and support throughout this journey. At first I would like to thank my mother and father (Fatemeh and Abbas); it is because of their never ending support that I have had the chance to progress in life. Their dedication to my education provided the foundation for my studies.

I started studying extracurricular Mathematics and Computer Science books in high school with the hope of succeeding in the Iranian Olympiad in Informatics. For this, I am grateful to Professor Mohammad Ghodsi; for the past many years he has had a critical role in organizing Computer Science related competitions and training camps all around Iran. With no doubt, without his support and guidance I would not have had access to an amazing educational atmosphere.

Probably the main reason I got admitted to MIT was the excellent guidance of Professor MohammadTaghi Hajiaghayi on how to conduct research in Theoretical Computer Science. From my undergraduate years up to now, Prof. Hajiaghayi has been a great colleague to work with. I would also like to thank Dr. Vahab Mirrokni who has been a great mentor for me in my graduate studies and during my internship at Google research. I am grateful for his support, advice, and friendship.

I would like to express my deepest gratitude to my advisor, Professor Erik Demaine for providing me with an excellent environment for conducting research. Professor Demaine has always amazed me by his intuitive way of thinking and his great teaching skills. He has been a great support for me and a nice person to talk to.

I would like to thank my committee members, Professors Piotr Indyk and Costis Daskalakis who were more than generous with their expertise and precious time.

I would also like to thank all my friends during my studies. In particular, I want to thank Mohammad Norouzi, Mohammad Rashidian, Hamid Mahini, Amin Karbasi, Hossein Fariborzi, Dan Alistarh, and Martin Demaine.

Finally, and most importantly, I want to thank my wife, Faezeh, who has been

very supportive during all my study years. Without your encouragements, I could not come this far. You inspire me by the way you look at life. Thank you for making all these years so wonderful.

Contents

List of Figures

Chapter 1

Introduction

Over the last few decades, a wide variety of allocation markets have emerged. Some of the most important examples of these markets are a) online dating markets that span around 75% of single people in United States, b) matching skilled workers with jobs such as National Resident Matching Program (NRMP) (or the Match) that matches medical school students with residency programs, c) Ad Auction multi-billion dollars markets which I elaborate more on it later as the quintessential application of my research. The emergence of these markets introduced many interesting optimization and algorithmic challenges. We can formalize these problems with a set of resources present in advance that should be consumed with a set of online nodes arriving one at a time during the algorithm. The goal is to design an algorithm that allocates these resources to online nodes when they arrive without over exhausting resources. We note that the allocation decisions are irrevocable. There are several objective functions considered in these problems including a) the revenue achieved by these allocations, and b) the quality of the allocation which is the total satisfaction of online nodes based on the resources they have received. We have studied designing online allocation algorithms that incorporate these multiple objectives simultaneously. Another important aspect of these problems that contributes to their online nature is the arrival order of online nodes. The performance of an online algorithm heavily depends on which nodes are going to arrive and their arrival order. In the worst case approach, the algorithm's performance is measured assuming the arrival order

is determined by an adversary, and therefore the algorithm's worst performance is computed as its indicative performance. However in stochastic settings, the arrival order is assumed to be a random permutation of online items, and the expected value of algorithm's performance is considered as its indicative performance. How to combine these two settings, and coming up with a more realistic analysis framework is the other main topic of my research.

I would like to highlight the use of allocation algorithms in computational advertising which is the quintessential application of my research e.g., the multi-billion dollar ad auction markets of Google or Bing search engines. We can formalize these types of computational advertising problems as follows. We are given a set of online items that could represent keywords searched in a search engine during a day, pageviews of a news website, or users of any website that shows ads. These online items are arriving one by one, and the algorithm should serve these items by assigning them some advertisements. On the other hand, we have a set of advertisers where each advertiser specifies how much she wants to pay for each of the online items when they arrive. There are multiple important objectives in this context including maximizing the total revenue, the number of clicks on ads, or the number of served online items. In these advertising applications, advertisers are in fact the resources, and there are different types of constraints on these resources depending on the application. There are two main classes of extensively studied problems in this context: *budgeted allocation* (a.k.a. the *adwords* problem) and *display ad* problems. Each advertiser is constrained by an overall budget limit, the maximum total amount she can pay in the first class, and by some positive integer capacity, the maximum number of online items we can assign to her in the second class.

The main issues of combining different objectives, and considering different arrival orders are discussed above as part of the optimization problem the publisher is facing. The publisher (Google, Bing, etc) is the search engine that decides which ads to show for each online node that arrives and has to maintain some revenue and keep up the quality (relevance) of these ads to maintain the online users in long term. At the end of this thesis, we look at these settings from an advertisers perspective which has a

limited budget and tries to win (capture) the most relevant online nodes to publicize her own business. We will see later how this is related to the classic secretary problem, and how we should extend the classic secretary problem to capture complexities of the advertiser's problem in online ad auctions.

1.1 Budgeted Allocation Problem

The performance of online algorithms for the adwords (Budgeted Allocation) problem heavily depends on the order in which the online items will appear. Mehta et al.[65] provided a $1 - 1/e$ competitive algorithm which works for every input order (adversarial or worst case order). They presented a novel adaptive scoring function on the advertisers (based on their remaining budgets) and allocated each online item to an advertiser for whom the product of her bid for the online item and her score is maximized. In another independent direction, researchers assumed that the online items appear according to a random permutation. Devanur and Hayes [23] proposed a Primal-Dual $1 - \varepsilon$ competitive algorithm in this case which can be adapted to other similar stochastic settings as well. The adversarial case is too pessimistic to model reality. On the other hand, the random arrival (and other stochastic) setting is useful only if the incoming traffic patterns of online items (e.g., page-views) can be predicted with a reasonably good precision. In other words, such algorithms may rely heavily on a precise forecast of the online traffic patterns, and may not react quickly to sudden changes in these patterns. In fact, the slow reaction to such spikes in traffic patterns imposes a serious limitation on the real-world use of stochastic algorithms in practical applications. This is a common issue in applying stochastic optimization techniques to the online resource allocation problems, e.g., see [79]. In an effort to tackle this ad allocation problem, we [67] study algorithms that achieve good performance ratios in both adversarial and stochastic frameworks simultaneously, and are robust against different traffic patterns.

We present algorithms that achieve the best competitive ratios of both settings simultaneously, i.e. $1 - \varepsilon$ in the random arrival model and $1 - 1/e$ in the adversarial

case for unweighted graphs. However, for weighted graphs we prove that this is not possible; we show when the competitive ratio of an online algorithm tends to 1 in the random arrival model, its competitive ratio tends to 0 in the adversarial model. Formally, we prove that an algorithm with competitive ratio $1 - \varepsilon$ in the random order arrival setting cannot have a competitive ratio more than $4\sqrt{\varepsilon}$ in the adversarial setting. We also prove that no online algorithm that achieves an approximation factor of $1 - 1/e$ for the adversarial inputs can achieve an average approximation factor better than 97.6% for random arrival inputs. In light of this hardness result, we design algorithms with improved approximation ratios in the random arrival model while preserving the competitive ratio of $1 - 1/e$ in the worst case. To this end, we show the algorithm proposed by [65] achieves a competitive ratio of 0.76 for the random arrival model, while having a $1 - 1/e \approx 0.63$ competitive ratio in the worst case.

Main Techniques: To achieve the hardness result, we exploit the fact that the algorithm has no means of distinguishing between adversarial and stochastic inputs. It is then sufficient to construct instances in which the behavior of an optimum online algorithm in the stochastic setting differs drastically from an algorithm that works well in the adversarial setting. For the positive results, we propose a general three stage process which is useful in analyzing the performance of any greedy algorithm: (i) We first define an appropriate potential function as the sum of the indefnite integrals of the advertisers' scores, and interpret the online algorithm as a greedy approach acting to improve the potential function by optimizing the corresponding scoring functions. (ii) We then track the changes in the potential function by formulating a mathematical program. (iii) Finally, by discretization on two spectrums of time and budgets and applying the mean value theorem of calculus, we translate the mathematical program into a constant-size LP and solve it numerically. We prove that the solution of this LP lower bounds the competitive ratio of the algorithm.

1.2 Display Ad Problem

In contrast to the budgeted allocation problem, in the display advertising problem, advertisers are not constrained by budget caps. Instead, each advertiser has some integer capacity which is the maximum number of online items we can assign to her. This problem has been modeled as maximizing the weight of an online matching instance [35, 34, 25, 15, 24]. While weight is indeed important, this model ignores the fact that cardinality of the matching is also crucial in the display ad application. This fact illustrates that in many real applications of online allocation, one needs to optimize multiple objective functions, though most of the previous work in this area deals with only a single objective function. On the other hand, there is a large body of work exploring *offline* multi-objective optimization in the approximation algorithms literature. In this part of thesis, we focus on simultaneously maximizing *online* two objectives which have been studied extensively in matching problems: cardinality and weight.

In online display advertising, advertisers typically purchase bundles of millions of display ad impressions from web publishers. Display ad serving systems that assign ads to pages on behalf of publishers must satisfy the contracts with advertisers, respecting targeting criteria and delivery goals. Modulo this, publishers try to allocate ads intelligently to maximize overall quality (measured, for example, by clicks), and therefore a desirable property of an ad serving system is to maximize this quality while satisfying the contracts to deliver to each advertiser its purchased bundle of impressions. This motivates our model of the display ad problem as simultaneously maximizing weight and cardinality.

We study this multi-objective online allocation problem by providing bicriteria competitive algorithms and showing the tightness of our algorithms by proving hardness results. We [60] present a bicriteria $(p(1 - 1/e^{1/p}), (1 - p)(1 - 1/e^{1-p}))$-approximation algorithm for every p in range $(0, 1)$ where the two approximation factors are for the weight and cardinality objectives. We provide hardness results that show the exact tightness of our algorithm at three main points: the starting

point $(0, 1 - 1/e)$, the middle point $(0.43, 0.43)$, and the final point $(1 - 1/e, 0)$. Our hardness results also show that the gap between our algorithm approximation factors and the pareto optimal curve is at most 9% at other points.

To achieve efficient algorithms that focus on both objectives, we use two primal dual subroutines that are responsible for weight and cardinality of the allocation separately. We allow the subroutines to share and exhaust the total capacities of advertisers separately, and provide an analysis to handle the collisions. To get hardness results, we construct multi-phase instances with exponentially growing weights. We then formulate linear programs that upper bound the performance of any online algorithm on the synthesized instances. Approximating the linear programs with constant size factor-revealing LPs yields the appropriate hardness results.

1.3 Submodular Secretary Problem

Another interesting and well-studied online allocation problem is the classic secretary problem in which a company wants to hire a secretary with a sequence of applicants arriving one by one. The goal is to hire the maximum value applicant, and the main assumption that yields non-zero competitive algorithms in this setting is the random order arrival of applicants. One of the most important generalizations of the secretary problem is the multiple choice version in which the company wants to hire several skilled workers (applicants) with different constraints varying from a simple cap on the number of secretaries to much more complex cases, such as matroid constraints.

We consider the problem of hiring up to a fixed number of applicants to maximize the performance of the secretarial group [11]. The overlapping skills of applicants can be modeled by defining the performance function as a submodular function on the set of applicants. The class of submodular functions spans a variety of value functions in practical settings. Following we show how this generalized version of classic secretary problem relates to the ad auctions applications. Consider an advertiser that has limited budget to spend throughout the day. She wants to show her ads to a subset of online users (e.g. associated with searched keywords in a search engine) to

maximize her own value while respecting her own budget. The budget constraint of the advertiser could be modelled with a knapsack constraint when the price of winning each online node is estimated accurately. The value function of subsets of online nodes is submodular in many applications as it observes the diminishing marginal value property.

We present constant competitive algorithms for this submodular secretary problem and provide constant hardness results as well. We then generalize our result to achieve constant competitive algorithms when we have multiple knapsack constraints. In the case of multiple matroid constraints, we present poly-logarithmic competitive algorithms. Finally, we show that $\Theta(\sqrt{n})$ is the best achievable competitve ratio for the subadditive secretary problem which models almost all practical situations including this class of submodular secretary problems.

Chapter 2

Simultaneous Algorithms for Budgeted Allocation Problem

Online bipartite matching is a fundamental optimization problem with many applications in online resource allocation, especially the online allocation of ads on the Internet. In this problem, we are given a bipartite graph $G = (X, Y, E)$ with a set of fixed nodes (or bins) Y, a set of online nodes (or balls) X, and a set E of edges between them. Any fixed node (or bin) $y_j \in Y$ is associated with a total weighted capacity (or budget) c_j. Online nodes (or balls) $x_i \in X$ arrive online along with their incident edges $(x_i, y_j) \in E(G)$ and their weights $w_{i,j}$. Upon the arrival of a node $x_i \in X$, the algorithm can assign x_i to at most one bin $y_j \in Y$ where $(x_i, y_j) \in E(G)$ and the total weight of nodes assigned to y_j does not exceed c_j. The goal is to maximize the total weight of the allocation. This problem is known as the *AdWords* problem, and it has been studied under the assumption that $\frac{\max_{i,j} w_{i,j}}{\min_j c_j} \to 0$, in [65, 18, 23].

Under the most basic online model, known as the *adversarial model*, the online algorithm does not know anything about the x_i's or $E(G)$ beforehand. In this model, the seminal result of Karp, Vazirani and Vazirani [56] gives an optimal online $1 - \frac{1}{e}$-competitive algorithm to maximize the size of the matching for *unweighted graphs* where $w_{ij} = 1$ for each $(x_i, y_j) \in E(G)$. For weighted graphs, Mehta et al. [65, 18] presented the first $1 - \frac{1}{e}$-approximation algorithm to maximize the total weight of the allocation for the AdWords problem and this result has been generalized to more

general weighted allocation problems [18, 33].

Other than the adversarial model, motivated by applications in online advertising, various *stochastic online* models have been proposed for this problem. In such stochastic models, online nodes $x_i \in X$ arrive in a random order. In other words, given a random permutation $\sigma \in \mathbb{S}_n$, the ball $x_{\sigma(t)}$ arrives at time time t for $t = 1, 2, \ldots, n$. In this case, the seminal result of Devanur and Hayes [23] gives a $1-\varepsilon$-approximation for the problem if the number of balls n is a prior information to the algorithm, and $\frac{\text{OPT}}{w_{ij}} \geq O(\frac{m \log n}{\varepsilon^3})$, where $m := |Y|$. This result has also been generalized and improved in several followup work [34, 2, 76], and its related models like the iid models with known or unknown distributions [36, 10, 66, 25][1]. These stochastic models are particularly motivated in the context of online ad allocation. In this context, online nodes correspond to page-views, search queries, or online requests for ads. In these settings, the incoming traffic of page-views may be predicted with a reasonable precision using a vast amount of historical data.

All these stochastic models and their algorithms are useful only if the incoming traffic of online nodes (e.g. page-views) can be predicted with a reasonably good precision. In other words, such algorithms may rely heavily on a precise forecast of the online traffic patterns, and may not react quickly to sudden changes in the traffic. In fact, the slow reaction to such traffic spikes impose a serious limitation in the real-world use of stochastic algorithms in practical applications. This is a common issue in applying stochastic optimization techniques for online resource allocation problems (see e.g., [79]). Various methodologies such as robust or control-based stochastic optimization [13, 14, 79, 74] have been applied to alleviate this drawback. In this chapter, we study this problem from a more idealistic perspective and aim to design algorithms that simultaneously achieve optimal approximation ratios for both the adversarial and stochastic models. Our goal is to design algorithms that achieve good performance ratios both in the worst case and in the average case.

Our Contributions and Techniques. In this chapter, we study simultaneous approximation algorithms for the adversarial and stochastic models for the online

[1] In the iid stochastic models, online nodes are drawn iid from a known or an unknown distribution.

budgeted allocation problem. Our goal is to design algorithms that achieve a competitive ratio strictly better than $1-1/e$ on average, while preserving a nearly optimal worst case competitive ratio. Ideally, we want to achieve the best of both worlds, i.e, to design an algorithm with the optimal competitive ratio in both the adversarial and random arrival models. Toward this goal, we show that this can be achieved for unweighted graphs, but not for weighted graphs. Nevertheless, we present improved approximation algorithms for weighted graphs.

For weighted graphs we prove that no algorithm can simultaneously achieve nearly optimal competitive ratios on both the adversarial and random arrival models. In particular, we show that no online algorithm that achieve an approximation factor of $1-\frac{1}{e}$ for the worst-case inputs may achieve an average approximation factor better than 97.6% for the random inputs (See Corollary 2.5.3). More generally, we show that any algorithm achieving an approximation factor of $1-\varepsilon$ in the stochastic model may not achieve a competitive ratio better than $4\sqrt{\varepsilon}$ in the adversarial model (See Theorem 2.5.1). In light of this hardness result, we aim to design algorithms with improved approximation ratios in the random arrival model while preserving the competitive ratio of $1-\frac{1}{e}$ in the worst case. To this end, we show an almost tight analysis of the algorithm proposed in [65] in the random arrival model. In particular, we show its competitive ratio is at least 0.76, and is no more than 0.81 (See Theorem 2.3.1, and Lemma 2.5.5). Combining this with the worst-case ratio analysis of Mehta et al. [65] we obtain an algorithm with the competitive ratio of 0.76 for the random arrival model, while having a $1-\frac{1}{e}$ ratio in the worst case. It is worth noting that unlike the result of [23] we do not assume any prior knowledge of the number of balls is given to the algorithm.

On the other hand, for unweighted graphs, under some mild assumptions, we show a generalization an algorithm in [54] achieves a competitive ratio of $1-\varepsilon$ in the random arrival model (See Theorem 2.4.1). Combining this with the worst-case ratio analysis of [54, 65], we obtain an algorithm with the competitive ratio of $1-\varepsilon$ in the random arrival model, while preserving the optimal competitive ratio of $1-\frac{1}{e}$ in the adversarial model. Previously, a similar result was known for a more

restricted stochastic model where all bins have equal capacities [68]. For the case of small degrees, an upper bound of 0.82 is known for the approximation ratio of any algorithm for the online stochastic matching problem (even for the under the iid model with known distributions) [66].

Our proofs consist of three main steps: (i) the main technique is to define an appropriate potential function as an indefinite integral of a scoring function, and interpret the online algorithms as a greedy algorithm acting to improve these potential functions by optimizing the corresponding scoring functions (see Section 2.2); These potential functions may prove useful elsewhere; (ii) An important component of the proof is to write a factor-revealing mathematical program based on the potential function and its changes, and finally (iii) the last part of the proofs involve changing the factor-revealing programs to a constant-size LP and solve it using a solver (in the weighted case), or analyzing the mathematical program explicitly using an intermediary algorithm with an oracle access to the optimum (in the unweighted case). The third step of the proof in the weighted case is inspired by the technique employed by Mahdian and Yan [64] for unweighted graphs, however, the set of mathematical programs we used are quite different from theirs.

All of our results hold under two mild assumptions: (i) large capacities (i.e., $\frac{\max_{i,j} w_{i,j}}{\min_j c_j} \to 0$), and (ii) a mild lower bound on the value of OPT: the aggregate sum of the largest weight ball assigned to each bin by the optimum is much smaller than OPT, i.e., $\sum_j \max_{i:opt(i)=j} w_{i,j} \ll \text{OPT}$. Both of these assumptions are valid in real-world applications of this problem in online advertising. The first assumption also appears in the AdWords problem, and the second assumption aims to get rid of some degenerate cases in which the optimum solution is very small.

2.1 Notations

Let $G(X, Y, E)$ be a (weighted) bipartite graph, where $X := \{x_1, \ldots, x_n\}$ is the set of online nodes (or balls), and $Y := \{y_1, \ldots, y_m\}$ is the set of fixed nodes (or bins). For each pair of nodes x_i, y_j, $w_{i,j}$ represents the weight of edge (x_i, y_j). Each online node

y_j is associated with a weighted capacity (or budget) $c_j > 0$. The online matching problem is as follows: first a permutation $\sigma \in \mathbb{S}_n$ is chosen (the distribution may be chosen according to any unknown distribution): at times $t = 1, 2, \ldots, n$, the ball $x_{\sigma(t)}$ arrives and its incident edges are revealed to the algorithm. The algorithm can assign this ball to at most one of the bins that are adjacent to it. The total weight of balls assigned to each bin y_j may not exceed its weighted capacity c_j. The objective is to maximize the weight of the final matching.

Given the graph G, the optimum offline solution is the maximum weighted bipartite matching in G respecting the weighted capacities, i.e, the total weight of balls assigned to to a bin y_j may not exceed c_j. For each ball x_i, let $opt(i)$ denote the index of the bin that x_i is being matched to in the optimum solution, and $alg(i)$ be the index of the bin that x_i is matched to in the algorithm. Also for each node $y_j \in Y$, let o_j be the weighted degree of y_j in the optimum solution. Observe that for each j, we have $0 \leq o_j \leq c_j$. By definition, we have the size of the optimum solution is $\text{OPT} = \sum_j o_j$. Throughout the chapter, we use OPT as the total weight of the optimal solution, and ALG as the total weight of the output of the online algorithm.

Throughout this chapter, we make the assumption that the weights of the edges are small compared to the capacities, i.e., $\max_{i,j} w_{i,j}$ is small compared to $\min_i c_j$. Also we assume that the aggregate sum of the largest weight ball assigned to each bin by the optimum is much smaller than OPT i.e., $\sum_j \max_{i:opt(i)=j} w_{i,j} \ll \text{OPT}$. In particular, let

$$\gamma \geq \max \left\{ \max_{i,j} \frac{w_{i,j}}{c_j}, \sqrt{\frac{\sum_j \max_{i:opt(i)=j} w_{i,j}}{\text{OPT}}} \right\} \tag{2.1}$$

the guarantees of our algorithm are provided for the case when $\gamma \to 0$. For justifications behind these mild assumptions, see the discussion in the Introduction. Throughout this chapter, wlog we assume that the optimum matches all of the balls, otherwise we can throw out the unmatched ball and it can only make the competitive ratio of the algorithm worse.

2.2 Main Ideas

In this section, we describe the main ideas of the proof. We start by defining the algorithms as deterministic greedy algorithms optimizing specific scoring functions. We also define a concave potential function as an indefinite integral of the scoring function, and show that a "good" greedy algorithm must try to maximize the potential function. In section 2.2.1, we show that if σ is chosen uniformly at random, then we can lower-bound the increase of the potential in an ε fraction of process; finally in section 2.2.2 we write a factor-revealing mathematical program based on the potential function and its changes.

We consider a class of deterministic greedy algorithms that assign each incoming ball $x_{\sigma(t)}$ based on a "scoring function" defined over the bins. Roughly speaking, the scoring function characterizes the "quality" of a bin, and a larger score implies a better-quality bin. In this chapter, we assume that the score is independent of the particular labeling of the bins, and it is a non-negative, *non-increasing* function of the amount that is saturated so far (roughly speaking, these algorithms try to prevent over-saturating a bin when the rest are almost empty). Throughout this chapter, we assume that the scoring function and its derivative are bounded (i.e., $|f'(.)|, |f(.)| \leq 1$). However, all of our arguments in this section can also be applied to the more general scoring functions that may even depend on the overall capacity c_i of the bins. At a particular time t, let $r_j(t)$ represent the fraction of the capacity of the bin y_j that is saturated so far. Let $f(r_j(t))$ be the score of bin y_j at time t. When the ball $x_{\sigma(t+1)}$ arrives, the greedy algorithm simply computes the score of all of the bins and assigns $x_{\sigma(t+1)}$ to the bin y_j *maximizing the product of* $w_{\sigma(t+1),j}$ *and* $f(r_j(t))$.

Kalyanasundaram, and Pruhs [54] designed the algorithm Balance using the scoring function $f_u(r_j(t)) := 1 - r_j(t)$ (i.e., the algorithm simply assigns an in-coming ball to the neighbor with the smallest ratio if it is less than 1, and drops the ball otherwise). They show that for any unweighted graph G, it achieves a $1 - 1/e$ competitive ratio against any adversarially chosen permutation σ. Mehta et al. [65] generalized this al-

gorithm to weighted graphs by defining the scoring function $f_w(r_j(t)) = (1 - e^{1-r_j(t)})$. Their algorithm, denoted by Weighted-Balance , achieves a competitive ratio of $1-1/e$ for the AdWords problem in the adversarial model. We note that both of the algorithms never over-saturate bins (i.e., $0 \leq r_j(t) \leq 1$). Other scoring functions have also been considered for other variants of the problem (see e.g. [63, 33]). Intuitively, these scoring functions are chosen to ensure that the algorithm assigns the balls as close to $opt(x_{\sigma(t)})$ as possible. When the permutation is chosen adversarially, any scoring function would fail to perfectly monitor the optimum assignment (as discussed before, no online algorithm can achieve a competitive ratio better than $1 - 1/e$ in the adversarial model). However, we hope that when σ is chosen uniformly at random, for any adversarially chosen graph G, the algorithm can almost capture the optimum assignment. In the following we try to formalize this observation.

We measure the performance of the algorithm at time t by assigning a potential function that in some sense compares the quality of the overall decisions of the algorithm w.r.t. the optimum. Assuming the optimum solution saturates all of the bins (i.e., $c_j = o_j$), the potential function achieves its maximum at the end of the algorithm if the balls are assigned exactly according to the optimum. The closer the value of the potential function to the optimum means a better assignment of the balls. We define the potential function as the weighted sum of the indefinite integral of the scoring functions of the bins chosen by the algorithm:

$$\phi(t) := \sum_j c_j \int_{r=0}^{r_j(t)} f(r)dr = \sum_j c_j F(r_j(t)).$$

In particular, we use the following potential function for the Balance and the Weighted-Balance algorithms respectively:

$$\phi_u(t) : = -\frac{1}{2}\sum_j c_j(1 - r_j(t))^2 \tag{2.2}$$

$$\phi_w(t) : = \sum_j c_j(r_j - e^{r_j(t)-1}). \tag{2.3}$$

Observe that since the scoring function is a non-increasing function of the ratios, its

antiderivative $F(.)$ will be a concave function of the ratios. Moreover, since it is always non-negative the value of the potential function never decreases in the running time of the algorithm. By this definition the greedy algorithm can be seen as an online gradient descent algorithm which tries to maximize a concave potential function; for each arriving ball $x_{\sigma(t)}$, it assigns the ball to the bin that makes the largest local increase in the function.

To analyze the performance of the algorithm we lower-bound the increase in the value of the potential function based on the optimum matching. This allows us to show that the final value of the potential function achieved by the algorithm is close to its value in the optimum, thus bound the competitive ratio of the algorithm. In the next section, we use the fact that σ is chosen randomly to lower-bound the increase in εn steps. Finally, in section 2.2.2 we write a factor-revealing mathematical program to compute the competitive ratio of the greedy algorithm.

2.2.1 Lower bounding the Increase in the Potential Function

In this part, we use the randomness defined on the permutation σ to argue that with high probability the value of the potential function must have a significant increase during the run of the algorithm. We define a particular event $\mathcal{E}$ corresponding to event that the arrival process of the balls is approximately close to its expectation. To show that $\mathcal{E}$ occurs with high probability, we only consider the distribution of arriving balls at $1/\varepsilon$ equally distance times; as a result we can mainly monitor the amount of increase in the potential function at these time intervals. For a carefully chosen $0 < \varepsilon < 1$, we divide the process into $1/\varepsilon$ slabs such that the k^{th} slab includes the $[kn\varepsilon + 1, (k+1)n\varepsilon]$ balls. Assuming σ is chosen uniformly at random, we show a concentration bound on the weight of the balls arriving in the k^{th} slab. Using that we lower bound $\phi((k+1)n\varepsilon) - \phi(kn\varepsilon)$ in Lemma 2.2.2.

First we use the randomness to determine the weight of the balls arriving in the k^{th} slab. Let $I_{i,k}$ be the indicator random variable indicating that the i^{th} ball will arrive in the k^{th} slab. Observe that for any k, the indicators $I_{i,k}$ are *negatively correlated*: knowing that $I_{i,k} = 1$ can only decrease the probability of the occurrence of the other

balls in the k^{th} slab (i.e., $\mathbf{P}[I_{i,k}|I_{i',k}=1] \leq \mathbf{P}[I_{i,k}]$). Define $N_{j,k} := \sum_{i:opt(i)=j} w_{i,j} I_{i,k}$ as the sum of the weight of the balls that are matched to the j^{th} bin in the optimum and arrive in the k^{th} slab. It is easy to see that $\mathbf{E}_\sigma[N_{j,k}] = \varepsilon \cdot o_j$, moreover, since it is a linear combination of negatively correlated random variables it will be concentrated around its mean. Define $h(k) := \sum_j |N_{j,k} - \varepsilon o_j|$. The following Lemma shows that $h(k)$ is very close to zero for all time slabs k with high probability. Intuitively, this implies that with high probability the balls from each slab are assigned similarly in the optimum solution.

Lemma 2.2.1. *Let* $h(k) := \sum_j |N_{j,k} - \varepsilon o_j|$. *Then* $\mathbf{P}_\sigma\left[\forall k, h(k) \leq \frac{5\gamma}{\varepsilon\delta}\mathrm{OPT}\right] \geq 1-\delta$.

Proof. It suffices to upper-bound $\mathbf{P}\left[h(k) > \frac{5\gamma}{\varepsilon\delta}\mathrm{OPT}\right] \leq \delta\varepsilon$; the lemma can then be proved by a simple application of the union bound. First we use Azuma-Hoeffding concentration bound to compute $\mathbf{E}[|N_{j,k} - \varepsilon o_j|]$; then we simply apply the Markov inequality to upper-bound $h(k)$.

Let $W_j := \sqrt{2\sum_{i:opt(i)=j} w_{i,j}^2}$, for any j,k, we show $\mathbf{E}[|N_{j,k} - \varepsilon o_j|] \leq 3W_j$. Since $N_{j,k}$ is a linear combination of negatively correlated random variables $I_{i,k}$ for $opt(i) = j$, and $\mathbf{E}[N_{j,k}] = \varepsilon \cdot o_j$ by a generalization of the Azuma Hoeffding bound to negatively correlated random variables [70] we have

$$\mathbf{E}[|N_{j,k} - \varepsilon \cdot o_j|] \leq W_j \left\{ \sum_{l=0}^{\infty} \mathbf{P}[|N_{j,k} - \mathbf{E}[N_{j,k}]| \geq l \cdot W_j] \right\}$$

$$\leq W_j \left\{ 1 + 2\sum_{l=1}^{\infty} e^{-\frac{l^2 W_j^2}{2\sum_{i:opt(i)=j} w_{i,opt(i)}^2}} \right\} \leq W_j \left(1 + 2\sum_{l=1}^{\infty} e^{-l^2} \right) \leq 3W_j. \qquad (2.4)$$

Let $w_{\max}(j) := \max_{i:opt(i)=j} w_{i,j}$. Since W_j^2 as twice the sum of the square of the weights assigned to the j^{th} bin is a convex function we can write $W_j \leq \sqrt{2w_{\max}(j)o_j}$. Therefore, by the linearity of expectation we have

$$\begin{aligned} \mathbf{E}[h(k)] &\leq \sum_j 3\sqrt{2w_{\max}(j)o_j} \leq 5\sum_j \frac{w_{\max}(j)/\gamma + \gamma o_j}{2} \\ &\leq 5\{\frac{1}{2\gamma}\sum_j w_{\max}(j) + \frac{\gamma}{2}\mathrm{OPT}\} \leq 5\gamma\mathrm{OPT}, \end{aligned}$$

where the last inequality follows from assumption (2.1). Since $h(k)$ is a non-negative random variable, by the Markov inequality we get $\mathbf{P}\left[h(k) > \frac{5\gamma}{\varepsilon\delta}\mathrm{OPT}\right] \leq \delta\varepsilon$. The lemma simply follows by applying this inequality for all $k \in \{0, \ldots, 1/\varepsilon\}$ and using the union bound. □

Let $\mathcal{E}$ be the event that $\forall k, h(k) \leq \frac{5\gamma}{\varepsilon\delta}\mathrm{OPT}$. The next lemma shows that conditioned on $\mathcal{E}$, one can lower-bound the increase in the potential function in any slab (i.e., $\phi((k+1)n\varepsilon) - \phi(kn\varepsilon)$ for any $0 \leq k < 1/\varepsilon$):

Lemma 2.2.2. *Conditioned on $\mathcal{E}$, for any $0 \leq k < 1/\varepsilon$, $t_0 = kn\varepsilon$, and $t_1 = (k+1)n\varepsilon$ we have*

$$\phi(t_1) - \phi(t_0) \geq \varepsilon \sum_j \{o_j f(r_j(t_1))\} - \frac{6\sqrt{\gamma}}{\varepsilon\delta}\mathrm{OPT}.$$

Proof. First we simply compute the increase of the potential function at time $t+1$, for some $t_0 \leq t < t_1$. Then, we lower-bound the increase using the monotonicity of the scoring function $f(.)$. Finally, we condition on $\mathcal{E}$ and lower-bound the final expression in terms of OPT. Let $\sigma(t+1) = i$, and assume the algorithm assigns x_i to the j^{th} bin (i.e., $alg(i) = j$); since the algorithm maximizes $w_{i,j} f(r_j(t))$ we can lower-bound the increase of the potential function based on the optimum. First using the mean value theorem of the calculus we have:

$$c_j \left\{ F(r_j(t) + \frac{w_{i,j}}{c_j}) - F(r_j(t) \right\} = c_j \left\{ \frac{w_{i,j}}{c_j} f(r_j(t)) + \frac{1}{2} \left(\frac{w_{i,j}}{c_j} \right)^2 f'(r^*) \right\},$$

for some $r^* \in [r_j(t), r_j(t) + w_{i,j}/c_j]$. Since the derivative of $f(.)$ is bounded (i.e., $|f'(r)| \leq 1$ for all $r \in [0,1]$), we get

$$\begin{aligned} \phi(t+1) - \phi(t) &= c_j \left\{ F(r_j(t) + \frac{w_{i,j}}{c_j}) - F(r_j(t) \right\} \geq w_{i,opt(i)} f(r_{opt(i)}(t)) - w_{i,j} \frac{w_{i,j}}{c_j} \\ &\geq w_{i,opt(i)} f(r_{opt(i)}(t_1)) - \gamma w_{i,j}, \end{aligned} \tag{2.5}$$

where the first inequality follows by the greedy decision chosen by the algorithm $w_{i,j} f(r_j(t)) \geq w_{i,opt(i)} f(r_{opt(i)}(t))$, and the last inequality follows by assumption (2.1).

Consider a single run of the algorithm; wlog we assume that ALG $\leq$ OPT. We

can monitor the amount of increase in the potential function in the k^{th} slab as follows:

$$\begin{aligned}\phi(t_1) - \phi(t_0) &\geq \sum_{t=t_0}^{t_1-1} \left\{ w_{\sigma(t),opt(\sigma(t))} f(r_{opt(\sigma(t))}(t)) - \gamma w_{\sigma(t),alg(\sigma(t))} \right\} \\ &\geq \sum_{j} \sum_{t_0 \leq t < t_1, opt(\sigma(t))=j} f(r_j(t_1)) w_{\sigma(t),j} - \gamma \mathrm{OPT} \\ &= \sum_{j} f(r_j(t_1)) N_{j,k} - \gamma \mathrm{OPT}\end{aligned}$$

where the second inequality follows by the fact that $f(.)$ is a non-increasing function of the ratio, and $\sum_{t_0 \leq t < t_1} w_{\sigma(t),alg(\sigma(t))} \leq \mathrm{ALG} \leq \mathrm{OPT}$, and the equality follows from the definition of $N_{j,k}$. By lemma 2.2.1 we know $N_{j,k}$ is highly concentrated around $\varepsilon \cdot o_j$. Conditioned on $\mathcal{E}$, we have $h(k) \leq \frac{5\gamma}{\varepsilon\delta} OPT$, thus:

$$\begin{aligned}\phi(t_1) - \phi(t_0) &\geq \varepsilon \sum_{j} f(r_j(t_1)) o_j - \sum_{j} |N_{j,k} - \varepsilon o_j| - \gamma \mathrm{OPT} \\ &\geq \varepsilon \sum_{j} f(r_j(t_1)) o_j - h(k) - \gamma \mathrm{OPT} \geq \varepsilon \sum_{j} f(r_j(t_1)) o_j - \frac{6\gamma}{\varepsilon\delta} \mathrm{OPT}\end{aligned}$$

where the first inequality follows by the assumption $|f(.)| \leq 1$.

□

2.2.2 Description of the Factor-Revealing Mathematical Program

In this section we propose a factor-revealing mathematical program that lower-bounds the competitive ratio of the algorithms Balance and Weighted-Balance . In sections 2.3, and 2.4 we derive a relaxation of the program and analyze that relaxation. Interestingly, the main non-trivial constraints follows from the lower-bounds obtained for the amount of increase in the potential function in Lemma 2.2.2.

The details of the program is described in MP(1). It is worth noting that the last inequality in this program follows from the monotonicity property of the ratios. In other words, we assume the ratio function $r_j(t)$ is a monotonically increasing function w.r.t. to t.

MP(1)

$$
\begin{array}{lcll}
\text{minimize } \frac{1}{\text{OPT}} \sum_j \min\{r_j(n), 1\} c_j & & & \\
\sum_j c_j F(r_j(t)) & = & \phi(t) & \forall t \in [n], \\
\varepsilon \sum_j o_j f(r_j((k+1)n\varepsilon)) - \frac{6\gamma}{\varepsilon\delta}\text{OPT} & \leq & \phi((k+1)n\varepsilon) - \phi(kn\varepsilon) & \forall k \in [\frac{1}{\varepsilon} - 1], \\
o_j & \leq & c_j & \forall j \in [m], \\
\sum_j o_j & = & \text{OPT}, & \\
r_j(t) & \leq & r_j(t+1) & \forall j, t \in [n-1].
\end{array}
$$

The following proposition summarizes our arguments and shows that the program MP(1) is a relaxation for any deterministic greedy algorithm that works based on a scoring function. It is worth noting that the whole argument still follows when the scoring function is not necessarily non-negative; we state the proposition in this general form.

Proposition 2.2.3. *Let f be any non-increasing, scoring function of the ratios $r_j(t)$ of the bins such that $|f(r)|, |f'(r)| \leq 1$ for the range of ratios that may be encountered in the running time of the algorithm. For any (weighted) graph $G = (X, Y)$, and $\varepsilon > 0$, with probability at least $1-\delta$,* MP(1) *is a factor-revealing mathematical program for the greedy deterministic algorithm that uses scoring function $f(.)$.*

Since the potential function $F(.)$ is a concave function, this program may not be solvable in polynomial time. In section 2.4, we show that after adding a new constraint it is possible to analyze it analytically for the unweighted graphs. To deal with this issue for the weighted graphs, we write a constant-size LP relaxation of the program that lower-bounds the optimum solution (after losing a small error). Finally, we solve the constant-size LP by an LP solver, and thus obtain a nearly tight bound for the competitive ratio of the Weighted-Balance (see section 2.3 for more details).

In the rest of this section, we write a simpler mathematical program that will be used later in section 2.3 for analyzing the Weighted-Balance algorithm. In particular, we simplify the critical constraint that measures the increase in the potential function by further removing the term $-\frac{6\gamma}{\varepsilon\delta}\text{OPT}$. Moreover, since Weighted-Balance never over-saturates bins we can also add the constraint $r_j(n) \leq 1$ to both MP(1) and MP(2) and still have a relaxation of Weighted-Balance .

$$
\begin{array}{lllll}
\text{MP(2)} & \text{minimize } \sum_j r_j(n)c_j & & & \\
\text{s.t.} & \sum_j c_j(r_j(t) - e^{r_j(t)-1}) & = & \phi(t) & \forall t \in [n], \\
& \varepsilon \sum_j o_j(1 - e^{r_j((k+1)n\varepsilon)-1}) & \leq & \phi((k+1)n\varepsilon) - \phi(kn\varepsilon) & \forall k \in [\frac{1}{\varepsilon}], \\
& o_j & \leq & c_j & \forall j \in [m], \\
& \sum_j o_j & = & 1. & \\
& r_j(t) & \leq & r_j(t+1) & \forall j, t \in [n-1], \\
& r_j(n) & \leq & 1 & \forall j \in [m].
\end{array}
$$

In the next Lemma we show that the optimum value of MP(2) is at least $(1-\sqrt{\frac{12\gamma}{\varepsilon^2\delta}})$ of MP(1):

Lemma 2.2.4. *For any weighted graph G, we have* $\text{MP(1)} \geq (1-\alpha)\min\{1, \text{MP(2)}\}$, *where* $\alpha := \sqrt{\frac{12\gamma}{\varepsilon^2\delta}}$.

Proof. Wlog we can replace OPT $= 1$ in MP(1). Let $s_1 := \{r_j(t), c_j, o_j, \phi(t)\}$ be a feasible solution of the MP(1). If $\sum_j r_j(n)c_j \geq (1-\alpha)$ we are done; otherwise we construct a feasible solution s_2 of the MP(2) such that the value of s_1 is at least $(1-\alpha)$ of the value of s_2. Then the lemma simply follows from the fact that the cost of the value of the optimum solution of MP(1) is at least of $(1-\alpha)$ of the value of the optimum of MP(2).

Define $s_2 := \{r_j(t), c_j/(1-\alpha), o_j, \phi(t)/(1-\alpha)\}$. Trivially, s_2 satisfies all except (possibly) the second constraint of MP(2). Moreover, the value of s_1 is $(1-\alpha)$ times the value of s_2. It remains to prove the feasibility of the second constraint of MP(2), i.e.,

$$\varepsilon(1-\alpha)\sum_j o_j(1 - e^{r_j((k+1)n\varepsilon)-1}) \leq \phi((k+1)n\varepsilon) - \phi(kn\varepsilon),$$

for all $k \in [1/\varepsilon]$. Since s_1 is a feasible solution of MP(1) we have

$$\phi((k+1)n\varepsilon) - \phi(kn\varepsilon) \geq \varepsilon\sum_j o_j(1 - e^{r_j((k+1)n\varepsilon)-1}) - \frac{\varepsilon}{2}\alpha^2$$

$$\geq \varepsilon\sum_j o_j(1 - e^{r_j((k+1)n\varepsilon)-1})\left\{1 - \frac{\frac{\varepsilon}{2}\alpha^2}{\frac{\varepsilon}{2}\sum_j o_j(1 - r_j((k+1)n\varepsilon)}\right\}, \tag{2.6}$$

where the last inequality follows from the assumption that $0 \leq r_j(t) \leq 1$, and the fact that $1 - e^{x-1} \geq \frac{1}{2}(1-x)$ for $x \in [0,1]$. On the other hand, since $\sum_j r_j(n)c_j < 1-\alpha$,

we can write:

$$\sum_j o_j(1 - r_j((k+1)n\varepsilon)) \geq 1 - \sum_j c_j r_j(n) \geq \alpha$$

The lemma simply follows from putting the above inequality together with equation (2.6). □

2.3 The Competitive Ratio of Weighted-Balance

In this section, we lower-bound the competitive ratio of the WEIGHED-BALANCE algorithm in the random arrival model. More specifically, we prove the following theorem:

Theorem 2.3.1. *For any weighted graph $G = (X, Y)$, and $\delta > 0$, with probability $1 - \delta$, the competitive ratio of the* Weighted-Balance *algorithm in the random arrival model is at least* $0.76(1 - O(\sqrt{\gamma/\delta}))$.

To prove the bound in this theorem, we write a constant-size linear programming relaxation of the problem based on MP(2) and solve the problem by an LP solver. The main two difficulties with solving program MP(2) are as follows: first, as we discussed in section 2.2.2, MP(2) is not a convex program; second, the size of the program (i.e., the number of variables and constraints) is a function of the size of the graph G. The main idea follows from a simple observation that the main inequalities in MP(2), those lower-bounding the increase in the potential function, are indeed lower-bounding the increase in the potential function only at constant (i.e., $1/\varepsilon$) number of times. Hence, we do not need to keep track of the ratios and the potential function for all $t \in [n]$; instead it suffices to monitor these values at $1/\varepsilon$ critical times (i.e., at times $kn\varepsilon$ for $k \in [1/\varepsilon]$), for a constant ε. Even in those critical times it suffices to approximately monitor the ratios of the bins by discretizing the ratios into $1/\varepsilon$ slabs.

For any integers $0 \leq i < 1/\varepsilon, 0 \leq k \leq 1/\varepsilon$, let $c_{i,k}$ be the sum of the capacities of the bins of ratio $r_j(kn\varepsilon) \in [i\varepsilon, (i+1)\varepsilon)$, and $o_{i,k}$ be the sum of the weighted degree

of the bins of ratio $r_j(kn\varepsilon) \in [i\varepsilon, (i+1)\varepsilon)$ in the optimum solution, i.e.,

$$c_{i,k} := \sum_{j: r_j(kn\varepsilon) \in [i\varepsilon,(i+1)\varepsilon)} c_j, \quad o_{i,k} := \sum_{j: r_j(kn\varepsilon) \in [i\varepsilon,(i+1)\varepsilon)} o_j. \tag{2.7}$$

Now we are ready to describe the constant-size LP relaxation of MP(2). We write the LP relaxation in terms of the new variables $c_{i,k}, o_{i,k}$. In particular, instead of writing the constraints in terms of the actual ratios of the bins, we round down (or round up) the ratios to the nearest multiple of ε such that the constraint remains satisfied. The details are described in LP(1).

$$\begin{array}{lllll}
\text{LP(1)} & \text{minimize } \frac{1}{1-1/e}\left\{\phi(\frac{1}{\varepsilon}) - \sum_{i=0}^{1/\varepsilon-1} c_{i,k}(i\varepsilon/e - e^{i\varepsilon-1})\right\} & & & \\
\text{s.t.} & \sum_{i=0}^{1/\varepsilon-1} c_{i,k}(i\varepsilon - e^{i\varepsilon-1}) & \leq & \phi(k) & \forall k \in [\frac{1}{\varepsilon}] \\
& \sum_{i=0}^{1/\varepsilon-1} \varepsilon o_{i,k+1}(1 - e^{(i+1)\varepsilon-1}) & \geq & \phi(k+1) - \phi(k) & \forall k \in [\frac{1}{\varepsilon} - 1] \\
& c_{i,k} & \geq & o_{i,k} & \forall i \in [\frac{1}{\varepsilon} - 1], k \in [\frac{1}{\varepsilon}] \\
& \sum_{i=0}^{1/\varepsilon-1} o_{i,k} & = & 1 & \forall k \in [\frac{1}{\varepsilon}] : \\
& \sum_{l=i}^{1/\varepsilon-1} c_{l,k+1} & \geq & \sum_{l=i}^{1/\varepsilon-1} c_{l,k} & \forall i \in [\frac{1}{\varepsilon} - 1], k \in [\frac{1}{\varepsilon} - 1]
\end{array}$$

In the next Lemma, we show that the LP(1) is a linear programming relaxation of the program MP(2):

Lemma 2.3.2. *The optimum value of* LP(1) *lower-bounds the optimum value of* MP(2).

Proof. We show that for any feasible solution $s := \{r_j(t), c_j, o_j, \phi(t)\}$ of MP(2) we can construct a feasible solution $s' = \{c'_{i,k}, o'_{i,k}, \phi'(k)\}$ for LP(1) with a smaller objective value. In particular, we construct s' simply by using equation (2.7), and letting $\phi'(k) := \phi(kn\varepsilon)$. First we show that all constraints of LP(1) are satisfied by s', then we show that the value of LP(1) for s' is smaller than the value of MP(2) for s.

The first equation of LP(1) simply follows from rounding down the ratios in the first equation of MP(2) to the nearest multiple of ε. The equation remains satisfied by the fact that the potential function $\phi(.)$ is increasing in the ratios (i.e., $F_w(r) = r - e^{r-1}$ is increasing in $r \in [0, 1]$). Similarly, the second equation of LP(1) follows from rounding up the ratios in the second equation of MP(2), and noting that the scoring function is decreasing in the ratios (i.e., $f_w(r) = 1 - e^{r-1}$ is decreasing for $r \in [0, 1]$).

The third and fourth equations can be derived from the corresponding equations in MP(2). Finally, the last equation follows from the monotonicity property of the ratios (i.e., $r_j(t)$ is a non-decreasing function of t).

It remains to compare the values of the two solutions s, and s'. We have

$$\begin{aligned} \frac{1}{1-1/e}\left\{\phi'(\frac{1}{\varepsilon}) - \sum_{i=0}^{1/\varepsilon-1} c'_{i,1/\varepsilon}\left(i\varepsilon/e - e^{i\varepsilon-1}\right)\right\} &\leq \\ \frac{1}{1-1/e}\left\{\phi(n) - \sum_j c_j(\frac{r_j(n)}{e} - e^{r_j(n)-1})\right\} &= \sum_j c_j r_j(n), \end{aligned}$$

where the inequality follows from the fact that $r/e - e^{r-1}$ is a decreasing function for $r \in [0,1]$, and the last inequality simply follows from the definition of $\phi_w(.)$ (i.e., the first equation of MP(2)). □

Now we are ready to prove Theorem 2.3.1:

Proof of Theorem 2.3.1. By Proposition 2.2.3, for any $\varepsilon > 0$, with probability $1-\delta$ the competitive ratio of Weighted-Balance is lower-bounded by the optimum of MP(1). On the other hand, by Lemma 2.2.4 the optimum solution of MP(1) is at least $(1-\sqrt{\frac{12\gamma}{\varepsilon^2\delta}})$ of the optimum solution of MP(2). Finally, by Lemma 2.3.2 the optimum solution of MP(2) is at least the optimum of LP(1). Hence, with probability $1-\delta$ the competitive ratio of Weighted-Balance is at least $(1-\sqrt{\frac{12\gamma}{\varepsilon^2\delta}})$ of the optimum of LP(1).

The constant-size linear program LP(1) can be solved numerically for any value of $\varepsilon > 0$. By solving this LP using an LP solver, we can show that for $\varepsilon = 1/250$ the optimum solution is greater than 0.76. This implies that with probability $1-\delta$ the competitive ratio of Weighted-Balance is at least $0.76(1-O(\sqrt{\gamma/\delta}))$. □

Remark 2.3.3. *We also would like to remark that the optimum solution of* LP(1) *beats the* $1-1/e$ *factor even for* $\varepsilon = 1/8$*; roughly speaking this implies that even if the permutation* σ *is almost random, in the sense that each* $1/8$ *of the input almost has the same distribution, then* Weighted-Balance *beats the* $1-1/e$ *factor.*

2.4 The Competitive Ratio of Balance

In this section we show that for any unweighted graph G, under some mild assumptions, the competitive ratio of Balance approaches 1 in the random arrival model.

Theorem 2.4.1. *For any unweighted bipartite graph $G = (X, Y, E)$, and $\delta > 0$, with probability $1 - \delta$ the competitive ratio of* Balance *in the random arrival model is at least $1 - \frac{\beta \sum_j c_j}{\text{OPT}}$, where $\beta := 3(\gamma/\delta)^{1/6}$.*

First we assume that our instance is *all-saturated* meaning that the optimum solution saturates all of the bins (i.e., $c_j = o_j$ for all j), and show that the competitive ratio of the algorithm is at least $1 - 3(\gamma/\delta)^{1/6}$:

Lemma 2.4.2. *For any $\delta > 0$, with probability $1 - \delta$ the competitive ratio of* Balance *on all-saturated instances in the random arrival model is at least $1 - \beta$.*

Then we prove Theorem 2.4.1 via a simple reduction to the all-saturated case.

To prove Lemma 2.4.2, we analyze a slightly different algorithm Balance' that always assigns an arriving ball (possibly to an over-saturated bin); this will allow us to keep track of the number of assigned balls at each step of the process. In particular we have

$$\forall t \in [n] : \sum_j c_j r_j(t) = t, \tag{2.8}$$

where $r_j(t)$ does not necessarily belong to $[0, 1]$. The latter certainly violates some of our assumptions in Section 2.2. To avoid the violation, we provide some additional knowledge of the optimum solution to Balance' such that the required assumptions are satisfied, and it achieves exactly the same weight as Balance .

We start by describing Balance' , then we show that it still is a feasible algorithm for the potential function framework studied in Section 2.2; in particular we show it satisfies Proposition 2.2.3. When a ball x_i arrives at time $t + 1$ (i.e., $\sigma(t + 1) = i$), similar to Balance , Balance' assigns it to a bin maximizing $w_{i,j} f_u(r_j(t))$; let j be such a bin. Unlike Balance if $r_j(t) \geq 1$ (i.e., all neighbors of x_i are saturated), we do not drop x_i; instead Balance' assigns it to the bin $y_{opt(i)}$.

First note that although Balance' magically knows the optimum assignment of a ball once all of its neighbors are saturated, it achieves the same weight matching. This simply follows from the fact that over-saturating bins does not increase our gain, and does not alter any future decisions of the algorithm. Next we use Proposition 2.2.3 to show that MP(1) is indeed a mathematical programming relaxation for Balance' .

By Proposition 2.2.3, we just need to verify that $|f_u(.)|, |f'_u(.)| \leq 1$ for all the ratios we might encounter in the run of Balance' . Since $f_u(r) = (1 - r)$, and the ratios are always non-negative, it is sufficient to show that the ratios are always upper-bounded by 2. To prove this, we crucially use the fact that Balance' has access to the optimum assignment for the balls assigned to the over-saturated bins. Observe that the set of balls assigned to a bin after it is being saturated, is always a subset of the balls assigned to it in the optimum assignment. Since the ratio of all bins are at most 1 in the optimum, they will be upper-bounded by 2 in Balance' .

The following is a simple mathematical programming relaxation to analyze Balance' in the all-saturated instances:

MP(3)

minimize $\sum_j \min\{r_j(n), 1\}c_j$

$$\begin{array}{lcll} \sum_j c_j r_j(t) & = & t & t \in [n], \\ \varepsilon \sum_j c_j(1 - r_j((k+1)n\varepsilon)) - \frac{6\gamma}{\varepsilon\delta}\mathrm{OPT} & \leq & \phi_u((k+1)n\varepsilon) - \phi_u(kn\varepsilon) & \forall k \in [\frac{1}{\varepsilon} - 1], \end{array}$$

Note that the first constraint follows from (2.8), and the second constraint follows from the second constraint of MP(1), and the fact that $c_j = o_j$ in the all-saturated instances.

Now we are ready to prove Lemma 2.4.2:

Proof of Lemma 2.4.2. With probability $1-\delta$, MP(3) is a mathematical programming relaxation of Balance' . First we sum up all $1/\varepsilon$ second constraints of MP(3) to obtain a lower-bound on $\phi_u(n)$, and we get $\phi_u(n)$ is very close to zero (intuitively, the algorithm almost manages to optimize the potential function). Then, we simply apply the Cauchy-Schwarz inequality to $\phi_u(n)$ to bound the loss of Balance' .

We sum up the second constraint of MP(3) for all $k \in \{0, 1, \ldots, \frac{1}{\varepsilon} - 1\}$; the RHS

telescopes and we obtain:

$$\begin{aligned}\phi_u(n) - \phi_u(0) &\geq OPT(1 - \frac{6\gamma}{\varepsilon^2\delta}) - \varepsilon \sum_{k=0}^{1/\varepsilon-1} \sum_j c_j r_j((k+1)n\varepsilon) \\ &\geq n(1 - \frac{6\gamma}{\varepsilon^2\delta}) - \varepsilon^2 n \sum_{k=0}^{1/\varepsilon-1} (k+1) \geq n(\frac{1}{2} - \frac{\varepsilon}{2} - \frac{6\gamma}{\varepsilon^2\delta})\end{aligned}$$

where the first inequality follows by the assumption that the instance is all-saturated, and the second inequality follows from applying the first constraint of MP(3) for $t = (k+1)n\varepsilon$, and the fact that OPT $= n$. Since $\phi_u(0) = -\frac{1}{2}\sum_j (1 - r_j(0))^2 = -n/2$, we obtain $\phi_u(n) \geq -n(\frac{\varepsilon}{2} + \frac{6\gamma}{\varepsilon^2\delta})$.

Observe that only the non-saturated bins incur a loss to the algorithm, i.e.,

$$Loss(\text{Balance' }) = \sum_{r_j(n)<1} c_j(1 - r_j(n)).$$

Using the lower-bound on $\phi_u(n)$ we have

$$\begin{aligned}\sum_{r_j(n)<1} c_j(1 - r_j(n)) &\leq \sqrt{\sum_{r_j(n)<1} c_j(1 - r_j(n))^2 \cdot \sum_{r_j(n)<1} c_j} \\ &\leq \sqrt{-2\phi_u(n) \cdot n} \leq n\sqrt{\varepsilon + \frac{12\gamma}{\varepsilon^2\delta}},\end{aligned}$$

where the first inequality follows by the Cauchy-Schwarz inequality, and the second inequality follows from the definition of $\phi_u(n)$. The lemma simply follows from choosing $\varepsilon = 2(2\gamma/\delta)^{1/3}$ in the above inequality. □

Next we prove Theorem 2.4.1; we analyze the general instances by a reduction to all-saturated instances.

Proof of Theorem 2.4.1. Let $G = (X, Y)$ be an unweighted graph, similar to Lemma 2.4.2 it is sufficient to analyze Balance' on G. For every bin y_j we introduce $c_j - o_j$ dummy balls that are only adjacent to the j^{th} bin, and let $G' = (X', Y)$ be the new instance. First we show that the expected number of non-dummy balls matched by Balance' in G' is at most the expected size of the matching that Balance' achieves

in G. We analyze the performance of Balance' on G simply using Lemma 2.4.2, and eliminating the effect of dummies.

Fix a permutation $\sigma \in \mathbb{S}_{|X'|}$; let $W'(\sigma)$ be the number of non-dummy balls matched by Balance' on σ. Similarly, let $W(\sigma[X])$ be the size of the matching obtained on $\sigma[X]$ in G, where $\sigma[X]$ is the projection of σ on X. Using an argument similar to [17, Lemma 2] (e.g., the monotonicity property), one can show that $W'(\sigma) \leq W(\sigma[X])$ for all $\sigma \in \mathbb{S}_{|X'|}$. Hence, to compute the competitive ratio of Balance' on G, it is sufficient to upper-bound the expected number of non-dummy balls not-matched by Balance' on G'. The latter is certainly not more than the total loss of Balance' on G' which is no more than $\beta \sum_j c_j$ by Lemma 2.4.2. □

2.5 Hardness Results

In this section, we show that there exists a family of weighted graphs G such that for any $\varepsilon > 0$, any online algorithm that achieves a $1 - \varepsilon$ competitive ratio in the random arrival model, does not achieve an approximation ratio better than a function $g(\varepsilon)$ in the adversarial model, where $g(\varepsilon) \to 0$ as $\varepsilon \to 0$. More specifically, we prove something stronger:

Theorem 2.5.1. *For any constants $\delta, \varepsilon > 0$, there exists family of weighted bipartite graphs $G = (X, Y)$ such that for any (randomized) algorithm that achieves $1 - \varepsilon$ competitive ratio (in expectation) on at least δ fraction of the permutations $\sigma \in \mathbb{S}_{|X|}$, does not achieve more than $4\sqrt{\varepsilon}$ (in expectation) for a particularly chosen permutation in another graph G'.*

As a corollary, we can show that any algorithm that achieves the competitive ratio of $1 - 1/e$ in the adversarial model can not achieve an approximation factor better than 0.976 in the random arrival model. Moreover, at the end of this section, we show that for some family of graphs the Weighted-Balance algorithm does not achieve an approximation factor better than 0.81 in the random arrival model (see Lemma 2.5.5 for more details). This implies that our analysis of the competitive ratio

of this algorithm is tight up to an additive factor of 5%. We start by presenting the construction of the hard examples:

Example 2.5.2. *Fix a large enough integer $l > 0$, and let $\alpha := \sqrt{\varepsilon}$; let $Y := \{y_1, y_2\}$ with capacities $c_1 = c_2 = l$. Let C and D be two types of balls (or online nodes), and let the set of online nodes X correspond to a set of l copies of C and l/α copies of D. Each type C ball has a weight of 1 for y_1, and a weight of 0 for y_2, while a type D ball has a weight of 1 in y_1 and a weight of α in y_2.*

First of all, observe that the optimum solution achieves a matching of weight $2l$ simply by assigning all type C balls to y_1, and type D balls to y_2. On the other hand, any algorithm that achieves the competitive ratio of $1-\varepsilon$ in the random arrival model should match the balls "very similar" to this strategy. However, if the algorithm uses this strategy, then an adversary may construct an instance by preserving the first l balls of the input followed by l/α dummy balls. But in this new instance it is "much better" to assign all of the first l balls to y_1. In the following we formalize this observation.

Proof of Theorem 2.5.1. Let G be the graph constructed in Example 2.5.2, and let A be a (randomized) algorithm that achieves $1-\varepsilon$ competitive ratio (in expectation) on at least δ fraction of permutations $\sigma \in \mathbb{S}_n$, where $n = l + l/\alpha$, for some constant $\delta > 0$. First we show that there exists a particular permutation σ^* such that there are at most $l\alpha$ balls of type C among $\{\sigma^*(1), \ldots, \sigma^*(l)\}$, and algorithm A achieves at least $(1-\varepsilon)2l$ on σ^*. Then we show that the (expected) gain of A from the first l balls is at most $4l\sqrt{\varepsilon}$. Finally, we construct a new graph $G' = (X', Y)$ and a permutation σ' such that the first l balls in σ' is the same as the first l balls of σ^*. This will imply that A does not achieve a competitive ratio better than $4\sqrt{\varepsilon}$ on G'.

To find σ^* it is sufficient to show that with probability strictly more than $1-\delta$ the number of type C balls among the first l balls of a uniformly random chosen permutation σ is at most $l\alpha$. This can be proved simply using the Chernoff-Hoeffding bound. Let B_i be a Bernoulli random variable indicating that $x_{\sigma(i)}$ is of type C, for $1 \leq i \leq l$. Observe that $\mathbf{E}_\sigma[B_i] = \frac{\alpha}{1+\alpha}$, and these variables are negatively correlated.

By a generalization of Chernoff-Hoeffding bound [70] we have

$$\mathbf{P}\left[\sum_{i=1}^{l} B_i > \alpha l\right] \leq e^{-\frac{l\alpha^3}{6}} < \delta,$$

where the last inequality follows by choosing l large enough. Hence, there exists a permutation σ^* such that the number of type C balls among its first l balls is at most $l\alpha$, and A achieves $(1-\varepsilon)2l$ on σ^*.

Next we show that the (expected) gain of A from the first l balls of σ^* is at most $2l(\alpha + \varepsilon/\alpha) = 4l\sqrt{\varepsilon}$. This simply follows from the observation that any ball of type D that is assigned to y_1 incurs a loss of α. Since the expected loss of the algorithm is at most $2l\varepsilon$ on σ^*, the expected number of type D balls assigned to y_1 (in the whole process) is no more than $\frac{2l\varepsilon}{\alpha}$. We can upper-bound the (expected) gain of the algorithm from the first l balls by $l\alpha + \frac{2l\varepsilon}{\alpha} + l\alpha$, where the first term follows from the upper-bound on the number of C balls, and the last term follows from the number of D balls (that may possibly be) assigned to y_2.

It remains to construct the adversarial instance G' together with the permutation σ'. G' has the same set of bins, while X' is the union of the first l balls of σ^* with l/α dummy balls (a dummy ball has zero weight in both of the bins). We construct σ' by preserving the first l balls of σ^*, filling the rest with the dummy balls (i.e., $x_{\sigma'(i)} = x_{\sigma^*(i)}$ for $1 \leq i \leq l$). First, observe that the optimum solution in G' achieves a matching of weight l simply by assigning all of the first l balls to y_1. On the other hand, as we proved the (expected) gain of the algorithm A is no more than $4l\sqrt{\varepsilon}$ on G'. Therefore, the competitive ratio of A in this adversarial instance is no more than $4\sqrt{\varepsilon}$. □

The following corollary can be proved simply by choosing δ small enough in Theorem 2.5.1:

Corollary 2.5.3. *For any constant $\varepsilon > 0$, any algorithm that achieves a competitive ratio of $1-\varepsilon$ in the random arrival model does not achieve strictly better than $4\sqrt{\varepsilon}$ in the adversarial model. In particular, it implies that any algorithm that achieves a competitive ratio of $1 - \frac{1}{e}$ in the adversarial model does not achieve strictly better*

than 0.976 *in the random order model.*

It is also worth noting that Weighted-Balance achieves at least 0.89 competitive ratio in the random arrival model for Example 2.5.2, and the worst case happens for $\alpha \approx 0.48$. Next we present a family of examples where the Weighted-Balance does not achieve a factor better than 0.81 in the random arrival model.

Example 2.5.4. *Fix a large enough integer $n > 0$, and $\alpha < 1$; again let $Y := \{y_1, y_2\}$ with capacities $c_1 = n$, and $c_2 = n^2$. Let X be a union of n identical balls each of weight 1 for y_1 and α for y_2.*

Lemma 2.5.5. *For a sufficiently large n, and a particularly chosen $\alpha > 0$, the competitive ratio of the* Weighted-Balance *in the random arrival model for Example 2.5.4 is no more than* 0.81.

Proof. First observe that the optimum solution achieves a matching of weight n simply by assigning all balls to y_1. Intuitively, Weighted-Balance starts with the same strategy, but after partially saturating y_1, it sends the rest to y_2 (note that each ball that is sent to y_2 incurs a loss of $1 - \alpha$ to the algorithm). Recall that $r_1(n)$ is the ratio of y_1 at the end of the algorithm. The lemma essentially follows from upper-bounding $r_1(n)$ by $1 + 1/n + \ln(1 - \alpha(1 - e^{1/n-1}))$. Since the algorithm achieves a matching of weight exactly $r_1(n)n + (1 - r_1(n))n\alpha$, and OPT $= n$, the competitive ratio is $r_1(n) + (1 - r_1(n))\alpha$. By optimizing over α, one can show that the minimum competitive ratio is no more than 0.81, and it is achieved by choosing $\alpha \simeq 0.55$.

It remains to show that $r_1(n) \leq 1 + 1/n + \ln(1 - \alpha(1 - e^{1/n-1}))$. Let t be the last time where a ball is assigned to y_1 (i.e., $r_1(t-1) + 1/n = r_1(t) = r_1(n)$). Since the ball at time t is assigned to y_1, we have

$$1 \cdot f_w(r_1(t-1)) \geq \alpha \cdot f_w(r_2(t-1)) \geq \alpha \cdot f_w(\frac{1}{n}),$$

where the last inequality follows by the fact that the ratio of the second bin can not be more than $\alpha \cdot n/c_2 < 1/n$, and $f_w(.)$ is a non-increasing function of the

ratios. Using $f_w(r) = 1 - e^{r-1}$, and $r_1(t-1) + 1/n = r_1(n)$ we obtain that $r_1(n) \leq 1 + 1/n + \ln(1 - \alpha(1 - e^{1/n-1}))$. □

2.6 Related Work

In this section, we discuss the related works that are not mentioned at the beginning of Chapter 2. For unweighted graphs, it has been recently observed that the Karp-Vazirani-Vazirani $1 - \frac{1}{e}$-competitive algorithm for the adversarial model also achieves an improved approximation ratio of 0.70 in the random arrival model [55, 64]. This holds even without the assumption of large degrees. It is known that without this assumption, one cannot achieve an approximation factor better than 0.82 for this problem (even in the case of iid with known distributions) [66]. This is in contrast with our result for unweighted graphs with large degrees.

Online budgeted allocation and its generalizations appear in two main categories of online advertising: *AdWords (AW)* problem [65, 18, 23], and the *Display Ads Allocation (DA)* problem [33, 34, 2, 76]. In both of these problems, the publisher must assign online page-views (or impressions) to an inventory of ads, optimizing efficiency or revenue of the allocation while respecting pre-specified contracts. In the DA problem, given a set of m advertisers with a set S_j of eligible impressions and demand of at most $n(j)$ impressions, the publisher must allocate a set of n impressions that arrive online. Each impression i has value $w_{ij} \geq 0$ for advertiser j. The goal of the publisher is to assign each impression to one advertiser maximizing the value of all the assigned impressions. The adversarial online DA problem has been studied in [33] in which the authors present a $1 - \frac{1}{e}$-competitive algorithm for the DA problem under a free disposal assumption for graphs of large degrees. This result generalizes the $1 - \frac{1}{e}$-approximation algorithm by Mehta et al [65] and by Buchbinder et. al. [18]. Following a training-based dual algorithm by Devanur and Hayes [23] for the AW problem, training-based $(1 - \varepsilon)$-competitive algorithms have been developed for the DA problem and its generalization to packing linear programs [34, 76, 2] including the DA problem. These papers develop a $(1 - \varepsilon)$-competitive algorithm for online

stochastic packing problems in which $\frac{\text{OPT}}{w_{ij}} \geq O(\frac{m \log n}{\varepsilon^3})$ (or $\frac{\text{OPT}}{w_{ij}} \geq O(\frac{m \log n}{\varepsilon^2})$ applying the technique of [2]) and the demand of each advertiser is large, in the random-order and the i.i.d model. Although studying a similar set of problems, none of the above papers study the simultaneous approximations for adversarial and stochastic models, and the dual-based $1 - \varepsilon$-competitive algorithms for the stochastic variants do not provide a bounded competitive ratio in the adversarial model.

Dealing with traffic spikes and inaccuracy in forecasting the traffic patterns is a central issue in operations research and stochastic optimization. Various methodologies such as robust or control-based stochastic optimization [13, 14, 79, 74] have been proposed. These techniques either try to deal with a larger family of stochastic models at once [13, 14, 79], try to handle a large class of demand matrices at the same time [79, 4, 6], or aim to design asymptotically optimal algorithms that react more adaptively to traffic spikes [74]. These methods have been applied in particular for traffic engineering [79] and inter-domain routing [4, 6]. Although dealing with similar issues, our approach and results are quite different from the approaches taken in these papers. Finally, an interesting related model for combining stochastic and online solutions for the Adwords problem is considered in [63], however their approach does not give an improved approximation algorithm for the iid model.

Chapter 3

Bicriteria Online Matching: Maximizing Weight and Cardinality

In the past decade, there has been much progress in designing better algorithms for online matching problems. This line of research has been inspired by interesting combinatorial techniques that are applicable in this setting, and by online ad allocation problems. For example, the display advertising problem has been modeled as maximizing the weight of an online matching instance [35, 34, 25, 15, 24]. While weight is indeed important, this model ignores the fact that cardinality of the matching is also crucial in the display ad application. This example illustrates the fact that in many real applications of online allocation, one needs to optimize multiple objective functions, though most of the previous work in this area deals with only a single objective function. On the other hand, there is a large body of work exploring *offline* multi-objective optimization in the approximation algorithms literature. In this chapter, we focus on simultaneously maximizing *online* two objectives which have been studied extensively in matching problems: cardinality and weight. Besides being a natural mathematical problem, this is motivated by online display advertising applications.

Applications in Display Advertising. In online display advertising, advertisers typically purchase bundles of millions of display ad impressions from web publishers.

Display ad serving systems that assign ads to pages on behalf of publishers must satisfy the contracts with advertisers, respecting targeting criteria and delivery goals. Modulo this, publishers try to allocate ads intelligently to maximize overall quality (measured, for example, by clicks), and therefore a desirable property of an ad serving system is to maximize this quality while satisfying the contracts to deliver the purchased number $n(a)$ impressions to advertiser a. This has been modeled in the literature (e.g., [35, 2, 76, 25, 15, 24]) as an online allocation problem, where quality is represented by edge weights, and contracts are enforced by overall delivery goals: While trying to maximize the *weight* of the allocation, the ad serving systems should deliver $n(a)$ impressions to advertiser a. However, online algorithms with adversarial input cannot guarantee the delivery of $n(a)$ impressions, and hence the goals $n(a)$ were previously modeled as upper bounds. But maximizing the *cardinality* subject to these upper bounds is identical to delivering as close to the targets as possible. This motivates our model of the display ad problem as simultaneously maximizing weight and cardinality.

Problem Formulation. More specifically, we study the following bicriteria online matching problem: consider a set of bins (also referred to as fixed nodes, or ads) A with capacity constraints $n(a) > 0$, and a set of online items (referred to as online nodes, or impressions or pageviews) I arriving one by one. Upon arrival of an online item i, a set S_i of eligible bins (fixed node neighbors) for the item is revealed, together with a weight w_{ia} for eligible bin $a \in S_i$. The problem is to assign each item i to an eligible bin in S_i or discard it online, while respecting the capacity constraints, so bin a gets at most $n(a)$ online items. The goal is to maximize both the cardinality of the allocation (i.e. the total number of assigned items) and the sum of the weights of the allocated online items.

It was shown in [35] that achieving any positive approximation guarantee for the total weight of the allocation requires the *free disposal* assumption, i.e. that there is no penalty for assigning more online nodes to a bin than its capacity, though these extra nodes do not count towards the objective. In the advertising application, this means that in the presence of a contract for $n(a)$ impressions, advertisers are only pleased by

– or at least indifferent to – getting *more* than $n(a)$ impressions. More specifically, if a set I^a of online items are assigned to each bin a, and $I^a(k)$ denotes the set of k online nodes with maximum weight in I^a, the goal is to simultaneously maximize cardinality which is $\sum_{a\in A} \min(|I^a|, n(a))$, and total weight which is $\sum_{a\in A} \sum_{i\in I^a(n(a))} w_{ia}$.

Throughout this chapter, we use W_{opt} to denote the maximum weight matching, and overload this notation to also refer to the weight of this matching. Similarly, we use C_{opt} to denote both the maximum cardinality matching and its cardinality. Note that C_{opt} and W_{opt} may be distinct matchings. We aim to find (α, β)-approximations for the bicriteria online matching problem: These are matchings with weight at least αW_{opt} and cardinality at least βC_{opt}. Our approach is to study parametrized approximation algorithms that allow a smooth tradeoff curve between the two objectives, and prove both approximation and hardness results in this framework. As an offline problem, the above bicriteria problem can be solved optimally in polynomial time, i.e., one can check if there exists an assignment of cardinality c and weight w respecting capacity constraints. (One can verify this by observing that the integer linear programming formulation for the offline problem is totally unimodular, and therefore the problem can be solved by solving the corresponding LP relaxation.) However in the online competitive setting, even maximizing one of these two objectives does not admit better than a $1 - 1/e$ approximation [56]. A naive greedy algorithm gives a $\frac{1}{2}$-approximation for maximizing a single objective, either for cardinality or for total weight under the free disposal assumption.

3.1 Results and Techniques

The seminal result of Karp, Vazirani and Vazirani [56] gives a simple randomized $(1 - 1/e)$-competitive algorithm for maximizing cardinality. For the weight objective, no algorithm better than the greedy 1/2-approximation is known, but for the case of large capacities, a $1 - 1/e$-approximation has been developed [35] following the primal-dual analysis framework of Buchbinder *et al.* [18, 65]. Using these results, one can easily get a $(\frac{p}{2}, (1-p)(1-\frac{1}{e}))$-approximation for the bicriteria online matching problem with

small capacities, and a $\left(p(1-\frac{1}{e}),(1-p)(1-\frac{1}{e})\right)$-approximation for large capacities. These factors are achieved by applying the online algorithm for weight, WeightAlg, and the online algorithm for cardinality, CardinalityAlg, as subroutines as follows: When an online item arrives, pass it to WeightAlg with probability p, and CardinalityAlg with probability $1-p$. As for a hardness result, it is easy to show that an approximation factor better than $(\alpha, 1-\alpha)$ is not achievable for any $\alpha > 0$. There is a large gap between the above approximation factors and hardness results. For example, the naive algorithm gives a $(0.4(1-1/e), 0.6(1-1/e)) \approx (0.25, 0.38)$-approximation, but the hardness result does not preclude a $(0.4, 0.6)$-approximation. In this chapter, we tighten the gap between these lower and upper bounds, and present new tradeoff curves for both algorithms and hardness results. Our lower and upper bound results are summarized in Figure 3-1. For the case of large capacities, these upper and lower bound curves are always close (with a maximum vertical gap of 9%), and exactly coincide at the point $(0.43, 0.43)$.

We first describe our hardness results. In fact, we prove three separate inapproximability results which can be combined to yield a 'hardness curve' for the problem. The first result gives better upper bounds for large values of β; this is based on structural properties of matchings, proving some invariants for any online algorithm on a family of instances, and writing a factor-revealing mathematical program (see Section 3.2.1). The second main result is an improved upper bound for large values of α, and is based on a new family of instances for which achieving a large value for α implies very small values of β (see Section 3.2.2). Finally, we show that for any achievable (α, β), we have $\alpha + \beta \leq 1 - \frac{1}{e^2}$ (see Theorem 3.2.2).

These hardness results show the limit of what can be achieved in this model. We next turn to algorithms, to see how close we can come to these limits. The key to our new algorithmic results lies in the fact that though each subroutine WeightAlg and CardinalityAlg only receives a fraction of the online items, it can use the entire set of bins. This may result in both subroutines filling up a bin, but if WeightAlg places t items in a bin, we can discard t of the items placed there by CardinalityAlg and still get at least the cardinality obtained by CardinalityAlg and the weight obtained by

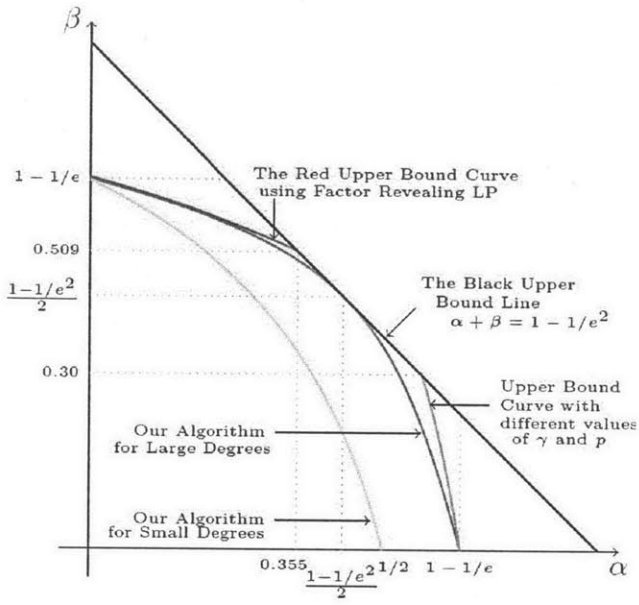

Figure 3-1: New curves for upper and lower bounds.

WeightAlg. Each subroutine therefore has access to the entire bin capacity, which is more than it 'needs' for those items passed to it. Thus, its competitive ratio can be made better than $1 - 1/e$. For large capacities, we prove the following theorem by extending the primal-dual analysis of Buchbinder *et al.* and Feldman *et al.* [35, 65, 18].

Theorem 3.1.1. *For all $0 < p < 1$, there is an algorithm for the bicriteria online matching problem with competitive ratios tending to $\left(p(1 - \frac{1}{e^{1/p}}), (1-p)(1 - \frac{1}{e^{1/(1-p)}})\right)$ as $\min_a\{n(a)\}$ tends to infinity.*

For small capacities, our result is more technical and is based on studying structural properties of matchings, proving invariants for our online algorithm over any instance, and solving a factor-revealing LP that combines these new invariants and previously known combinatorial techniques by Karp, Vazirani, Vazirani, and Birnbaum and Mathieu [56, 17]. Factor revealing LPs have been used in the context of online allocation problems [65, 64]. In our setting, we need to prove new variants and introduce new inequalities to take into account and analyze the tradeoff between the two objective functions. This result can also be parametrized by p, the fraction

of items sent to WeightAlg, but we do not have a closed form expression. Hence, we state the result for $p = 1/2$.

Theorem 3.1.2. *For all $0 \leq p \leq 1$, the approximation guarantee of our algorithm for the bicriteria online matching problem is lower bounded by the green curve of Figure 3-1. In particular, for $p = 1/2$, we have the point $(1/3, 0.3698)$.*

Related Work. Our work is related to online ad allocation problems, including the *Display Ads Allocation (DA)* problem [35, 34, 2, 76], and the *AdWords (AW)* problem [65, 23]. In both of these problems, the publisher must assign online impressions to an inventory of ads, optimizing efficiency or revenue of the allocation while respecting pre-specified contracts. The Display Ad (DA) problem is the online matching problem described above only considering the weight objective [35, 15, 24]. In the AdWords (AW) problem, the publisher allocates impressions resulting from search queries. Advertiser a has a budget $B(a)$ on the total spend instead of a bound $n(a)$ on the number of impressions. Assigning impression i to advertiser a consumes w_{ia} units of a's budget instead of 1 of the $n(a)$ slots, as in the DA problem. For both of these problems, $1 - \frac{1}{e}$-approximation algorithms have been designed under the assumption of large capacities [65, 18, 35]. None of the above papers for the adversarial model study multiple objectives at the same time.

Besides the adversarial model studied in this chapter, online ad allocations have been studied extensively in various *stochastic models*. In particular, the problem has been studied in the *random order model*, where impressions arrive in a random order [23, 34, 2, 76, 55, 64, 67]; and the *iid* model in which impressions arrive iid according to a known (or unknown) distribution [36, 66, 46, 25, 26]. In such stochastic settings, primal and dual techniques have been applied to getting improved approximation algorithms. These techniques are based on computing offline optimal primal or dual solutions of an expected instance, and using this solution online [36, 23]. It is not hard to generalize these techniques to the bicritera online matching problem. In this chapter, we focus on the adversarial model. Note that in order to deal with traffic spikes, adversarial competitive analysis is important from a practical perspective, as

discussed in [67].

Most previous work on online problems with multiple objectives has been in the domain of routing and scheduling, and with different models. Typically, goals are to maximize throughput and fairness; see the work of Goel *et al.* [43, 42], Buchbinder and Naor [19], and Wang *et al.* [78]. In this literature, different objectives often come from applying different functions on the same set of inputs, such as processing times or bandwidth allocations. In a model more similar to ours, Bilò *et al.* [16] consider scheduling where each job has two different and unrelated requirements, processing time and memory; the goal is to minimize makespan while also minimizing maximum memory requirements on each machine. In another problem with distinct metrics, Flammini and Nicosia [38] consider the k-server problem with a distance metric and time metric defined on the set of service locations. However, unlike our algorithms, theirs do not compete simultaneously against the best solution for each objective; instead, they compete against offline solutions that must simultaneously do well on both objectives. Further, the competitive ratio depends on the relative values of the two objectives. Such results are of limited use in advertising applications, for instance, where click-through rates per impression may vary by several orders of magnitude.

3.2 Hardness Instances

In this section for any $0 \leq \alpha \leq 1 - 1/e$, we prove upper bounds on β such that the bi-criteria online matching problem admits an (α, β)-approximation. Note that it is not possible to achieve α-approximation guarantee for the total weight of the allocation for any $\alpha > 1 - 1/e$. We have two types of techniques to achieve upper bounds: a) Factor-Revealing Linear Programs, b) Super Exponential Weights Instances, which are discussed in Subsections 3.2.1, and 3.2.2 respectively. Factor revealing LP hardness instances give us the red upper bound curve in Figure 3-1. The orange upper bound curve in Figure 3-1 is proved by Super Exponential Weights Instances presented in Subsection 3.2.2, and the black upper bound line in Figure 3-1 is proved in Theorem 3.2.2.

3.2.1 Better Upper Bounds via Factor-Revealing Linear Programs

We construct an instance, and a linear program $LP_{\alpha,\beta}$ based on the instance where α and β are two parameters in the linear program. We prove that if there exists an (α, β)-approximation for the bicriteria online matching problem, we can find a feasible solution for $LP_{\alpha,\beta}$ based on the algorithm's allocation for the generated instance. Finally we find out for which pairs (α, β) the linear program $LP_{\alpha,\beta}$ is infeasible. These pairs (α, β) are upper bounds for the bicriteria online matching problem.

For any two integers C, l, and some large weight $W \gg 4l^2$, we construct the instance as follows. We have l phases, and each phase consists of l sets of C identical items, i.e. l^2C items in total. For any $1 \leq t, i \leq l$, we define $O_{t,i}$ to be the set i in phase t that has C identical items. In each phase, we observe the sets of items in increasing order of i. There are two types of bins: a) l weight bins $b_1, b_2, \cdots, b_l$ which are shared between different phases, b) l^2 cardinality bins $\{b'_{t,i}\}_{1 \leq t,i \leq l}$. For each phase $1 \leq t \leq l$, we have l separate bins $\{b'_{t,i}\}_{1 \leq i \leq l}$. The capacity of all bins is C. We pick two permutations $\pi_t, \sigma_t \in \mathbb{S}_n$ uniformly at random at the beginning of each phase t to construct edges. We note that these permutations are private knowledge, and they are not revealed to the algorithm. For any $1 \leq i \leq j \leq l$, we put an edge between every item in set $O_{t,i}$ and bin $b'_{t,\sigma_t(j)}$ with weight 1 where $\sigma_t(j)$ is the jth number in permutation σ_t. We also put an edge between every item in set $O_{t,i}$ and bin $b_{\pi_t(j)}$ (for each $j \geq i$) with weight W^t.

Suppose there exists an (α, β)-approximation algorithm $A_{\alpha,\beta}$ for the bicriteria online matching problem. For any $1 \leq t, i \leq l$, let $x_{t,i}$ be the expected number of items in set $O_{t,i}$ that algorithm $A_{\alpha,\beta}$ assigns to weight bins $\{b_{\pi_t(j)}\}_{j=i}^{l}$. Similarly we define $y_{t,i}$ to be the expected number of items in set $O_{t,i}$ that algorithm $A_{\alpha,\beta}$ assigns to cardinality bins $\{b'_{t,\sigma_t(j)}\}_{j=i}^{l}$. We know that when set $O_{t,i}$ arrives, although the algorithm can distinguish between weight and cardinality bins, it sees no difference between the weight bins $\{b_{\pi_t(j)}\}_{j=i}^{l}$, and no difference between the cardinality bins $\{b'_{t,\sigma_t(j)}\}_{j=i}^{l}$. By uniform selection of π and σ, we ensure that in expectation the $x_{t,i}$ items are

allocated equally to weight bins $\{b_{\pi_t(j)}\}_{j=i}^{l}$, and the $y_{t,i}$ items are allocated equally to cardinality bins $\{b'_{t,\sigma_t(j)}\}_{j=i}^{l}$. In other words, for $1 \leq i \leq j \leq l$, in expectation $x_{t,i}/(l-i+1)$ and $y_{t,i}/(l-i+1)$ items of set $O_{t,i}$ is allocated to bins $b_{\pi_t(j)}$ and $b'_{t,\sigma_t(j)}$, respectively. It is worth noting that similar ideas have been used in previous papers on online matching [56, 17].

Since weights of all edges to cardinality bins are 1, we can assume that the items assigned to cardinality bins are kept until the end of the algorithm, and they will not be thrown away. We can similarly say that the weights of all items for weight bins is the same in a single phase, so we can assume that an item that has been assigned to some weight bin in a phase will not be thrown away at least until the end of the phase. However, the algorithm might use the free disposal assumption for weight bins in different phases. We have the following capacity constraints on bins $b_{\pi_t(j)}$ and $b'_{t,\sigma_t(j)}$:

$$\forall 1 \leq t, j \leq l: \quad \sum_{i=1}^{j} x_{t,i}/(l-i+1) \leq C \quad \& \sum_{i=1}^{j} y_{t,i}/(l-i+1) \leq C. \tag{3.1}$$

At any stage of phase t, the total weight assigned by the algorithm cannot be less than α times the optimal weight allocation up to that stage, or we would not have weight αW_{opt} if the input stopped at this point. After set $O_{t,i}$ arrives, the maximum weight allocation achieves at least total weight CiW^t which is achieved by assigning items in set $O_{t,i'}$ to weight bin $b_{\pi_t(i')}$ for each $1 \leq i' \leq i$. On the other hand, the expected weight in allocation of algorithm $A_{\alpha,\beta}$ is at most $C(tl + W^{t-1}l) + W^t \sum_{i'=1}^{i} x_{t,i'} \leq W^t(C/\sqrt{W} + \sum_{i'=1}^{i} x_{t,i'})$. Therefore we have the following inequality for any $1 \leq t, i \leq l$:

$$\sum_{i'=1}^{i} x_{t,i'}/C \geq \alpha i - 1/\sqrt{W}. \tag{3.2}$$

We prove in Lemma 3.2.1 that the linear program $LP_{\alpha,\beta}$ is feasible if there exists an algorithm $A_{\alpha,\beta}$ by defining $p_i = \sum_{t=1}^{l} x_{t,i}/lC$, and $q_i = \sum_{t=1}^{l} y_{t,i}/lC$. Now for any α, we can find the maximum β for which the $LP_{\alpha,\beta}$ has some feasible solution for

large values of l and W. These factor-revealing linear programs yield the red upper bound curve in Figure 3-1.

$$
\begin{array}{llcll}
LP_{\alpha,\beta} & & & & \\
\text{C1:} & \sum_{i'=1}^{i} p_{i'} & \geq & \alpha i - 1/\sqrt{W} & \forall 1 \leq i \leq l \\
\text{C2:} & \sum_{i=1}^{l} q_i & \geq & l\beta - 1 & \\
\text{C3:} & p_i + q_i & \leq & 1 & \forall 1 \leq i \leq l \\
\text{C4:} & \sum_{i=1}^{j} p_i/(l-i+1) & \leq & 1 & \forall 1 \leq j \leq l \\
\text{C5:} & \sum_{i=1}^{j} q_i/(l-i+1) & \leq & 1 & \forall 1 \leq j \leq l
\end{array}
$$

Lemma 3.2.1. *If there exists an (α, β)-approximation algorithm for the bicriteria online matching problem, there exists a feasible solution for $LP_{\alpha,\beta}$ as well.*

Proof. We claim that the values $p_i = \sum_{t=1}^{l} x_{t,i}/lC$, and $q_i = \sum_{t=1}^{l} y_{t,i}/lC$ form a feasible solution for $LP_{\alpha,\beta}$. Intuitively, p_i and q_i are the average values of $x_{t,i}/C$ and $y_{t,i}/C$ in the l different phases. Since we have inequality 3.2 for each value of t, their average values also admit the same type of inequlity which is constraint C1 in $LP_{\alpha,\beta}$. To prove that constraint C2 holds, we should look at the total cardinality of algorithm's allocation in all phases. The optimal cardinality allocation is to assign set $O_{t,i}$ to $b'_{t,\sigma_t(i)}$ for all $1 \leq t, i \leq l$ which assigns all items and achieves l^2C in cardinality. But algorithm $A_{\alpha,\beta}$ assigns at most lC items to weight bins at the end (after applying free disposals), and $\sum_{1 \leq t,i \leq l} y_{t,i}$ items to cardinality bins. Since it is a β-approximation for cardinality, we should have that $\sum_{1 \leq t,i \leq l} y_{t,i} + lC \geq l^2C\beta$. Based on definition of $\{q_i\}_{i=1}^{l}$, this inequality is equivalent to constraint $C2$. Constraint C3 holds because the expected total number of the items the algorithm assigns from each set to weight and cardinality bins can not be more than C, the number of items in the set. Constraints C4 and C5 are derived from inequalities of Equation 3.1. □

In addition to computational bounds for infeasibility of certain (α, β) pairs, we can theoretically prove in Theorem 3.2.2 that for any (α, β) with $\alpha + \beta > 1 - 1/e^2$, the $LP_{\alpha,\beta}$ is infeasible so there exists no (α, β) approximation for the problem. We note that Theorem 3.2.2 is a simple generalization of the $1 - 1/e$ hardness result for the classic online matching problem [56, 17].

Theorem 3.2.2. *For any small $\varepsilon > 0$, and $\alpha + \beta \geq 1 - 1/e^2 + \varepsilon$, there exists no (α, β)-approximation algorithm for the bicriteria matching problem.*

Proof. We just need to show that $LP_{\alpha,\beta}$ is infeasible. Given a solution of $LP_{\alpha,\beta}$, we find a feasible solution for LP'_ε defined below by setting $r_i = p_i + q_i$ for any $1 \leq i \leq l$.

$$\begin{array}{lllll} LP'_\varepsilon & \sum_{i=1}^{l} r_i & \geq & (1 - 1/e^2 + \varepsilon/2)l & \\ & r_i & \leq & 1 & \forall 1 \leq i \leq l \\ & \sum_{i=1}^{j} r_i/(l-i+1) & \leq & 2 & \forall 1 \leq j \leq l \end{array}$$

The first inequality in LP'_ε is implied by summing up the constraint C1 for $i = l$, and constraint C2 in $LP_{\alpha,\beta}$, and also using the fact that $\alpha+\beta \geq (1-1/e^2+\varepsilon/2)+\varepsilon/2$. We note that the $\varepsilon/2$ difference between the $\alpha + \beta$ and $1 - 1/e^2 + \varepsilon/2$ takes care of $-1/\sqrt{W}$ and -1 in the right hand sides of constraints C1 and C2 for large enough values of l and W. Now we prove that LP'_ε is infeasible for any $\varepsilon > 0$ and large enough l. Suppose there exists a feasible solution $r_1, r_2, \cdots, r_n$. For any pair $1 \leq i < j \leq n$, if we have $r_i < 1$ and $r_j > 0$, we update the values of r_i and r_j to $r_i^{new} = r_i + \min\{1 - r_i, r_j\}$, and $r_j^{new} = r_j - \min\{1 - r_i, r_j\}$. Since we are moving the same amount from r_j to r_i (for some $i < j$), all constraints still hold. If we do this operation iteratively until there is no pair r_i and r_j with the above properties, we reach a solution $\{r'_i\}_{i=1}^{l}$ of this form: $1, 1, \cdots, 1, x, 0, 0, \cdots, 0$ for some $0 \leq x \leq 1$. Let t be the maximum index for which r'_t is 1. Using the third inequality for $j = l$, we have that $\sum_{i=1}^{t} 1/(l-i+1) \leq 2$ which means that $\ln(l/(l-t+1)) \leq 2$. So t is not greater than $l(1-1/e^2)$, and consequently $\sum_{i=1}^{l} r'_i \leq t+1 \leq (1-1/e^2)l+1 < (1-1/e^2+\varepsilon/2)l$. This contradiction proves that LP'_ε is infeasible which completes the proof of theorem. □

3.2.2 Hardness Results for Large Values of Weight Approximation Factor

The factor-revealing linear program $LP_{\alpha,\beta}$ gives almost tight bounds for small values of α. In particular, the gap between the the upper and lower bounds for the cardinality approximation ratio β is less than 0.025 for $\alpha \leq (1 - 1/e^2)/2$. But for large values of α ($\alpha > (1 - 1/e^2)/2$), this approach does not give anything better than the $\alpha +$

$\beta \leq 1 - 1/e^2$ bound proved in Theorem 3.2.2 . This leaves a maximum gap of $1/e - 1/e^2 \approx 0.23$ between the upper and lower bounds at $\alpha = 1 - 1/e$. In order to close the gap at $\alpha = 1 - 1/e$, we present a different analysis based on a new set of instances, and reduce the maximum gap between lower and upper bounds from 0.23 to less than 0.09 for all values of $\alpha \geq (1 - 1/e^2)/2$.

The main idea is to construct a hardness instance I_γ for any $1/e \leq \gamma < 1$, and prove that for any $0 \leq p \leq 1-\gamma$, the pair $(1-1/e-f(p), p/(1-\gamma))$ is an upper bound on (α, β) where $f(p)$ is $\frac{p}{e(\gamma+p)}$. In other words, there exists no (α, β)-approximation algorithm for this problem with both $\alpha > 1 - 1/e - f(p)$ and $\beta > p/(1-\gamma)$. By enumerating different pairs of γ and p, we find the orange upper bound curve in Figure 3-1.

For any $\gamma \geq 1/e$, we construct instance I_γ as follows: The instance is identical to the hardness instance in Subsection 3.2.1, but we change some of the edge weights. To keep the description short, we only describe the edges with modified weights here. Let r be $\lfloor 0.5 \log_{1/\gamma} l \rfloor$. In each phase $1 \leq t \leq l$, we partition the l sets of items $\{O_{t,i}\}_{i=1}^{l}$ into r groups. The first $l(1-\gamma)$ sets are in the first group. From the remaining γl sets, we put the first $(1-\gamma)$ fraction in the second group and so on. Formally, we put set $O_{t,i}$ in group $1 \leq z < r$ for any $i \in [l - l\gamma^{z-1} + 1, l - l\gamma^z]$. Group r of phase t contains the last $l\gamma^{r-1}$ sets of items in phase t. The weight of all edges from sets of items in group z in phase t is $W^{(t-1)r+z}$ for any $1 \leq z \leq r$ and $1 \leq t \leq l$.

Given an (α, β)-approximation algorithm $A_{\alpha,\beta}$, we similarly define $x_{t,i}$ and $y_{t,i}$ to be the expected number of items from set $O_{t,i}$ assigned to weight and cardinality bins by algorithm $A_{\alpha,\beta}$ respectively. We show in the following lemma that in order to have a high α, the algorithm should allocate a large fraction of sets of items in each group to the weight bins.

Lemma 3.2.3. *For any phase $1 \leq t \leq l$, and group $1 \leq z < r$, if the expected number of items assigned to cardinality bins in group z of phase t is at least $plC\gamma^{z-1}$ (which is p times the number of all items in groups $z, z+1, \cdots, r$ of phase t), the weight approximation ratio cannot be greater than $1 - 1/e - f(p)$ where $f(p)$ is $\frac{p}{e(\gamma+p)}$.*

Proof. We define $w = W^{(t-1)r+z}$ which is the weight of edges from items in group z of phase t to weight bins. At the end of group z of phase t, we look at the expected number of items allocated to cardinality bins in this group. If it is more than $plC\gamma^{z-1}$, we construct a new instance I' by changing the weights of the edges of the next items as follows. Instance I' is identical to our instance I_γ up to the end of group z in phase t. After this group, everything is identical to I_γ except the edge weights that we change as follows. For every group $r \geq z' > z$ in phase t, instead of setting the weights of edges between items and bins to $W^{(t-1)r+z'}$, we set them to the weights of edges in group z to weight bins which is $w = W^{(t-1)r+z}$. After phase t, we set the weights of all edges from items in phases $t+1, t+2, \cdots, l$ to both weight and cardinality bins to zero. Since our instance I_γ, and the new instance I' are identical up to the end of group z of phase t, we know that the algorithm has the same expected allocation up to that point for both instances.

To simplify notation in the rest of the proof, we rename the items in groups $z, z+1, \cdots, r$ of phase t, and their associated weight bins. We define l' to be $l\gamma^{z-1}$ which is the number of sets of items in groups $z, z+1, \cdots, r$ of phase t. For any $1 \leq i \leq l'$, we let O'_i to be the set of items $O_{t,l-l'+i}$, and also let b''_i to be the weight bin $b_{\pi_t(l-l'+i)}$. In instance I', the optimum weight allocation is at least $wl'C$ which can be achieved by assigning set of items O'_i to weight bin b''_i for any $1 \leq i \leq l'$. We also know that the weight any algorithm achieves in instance I' is from assigning these l' items to the weight bins, because the total weights achieved before and after these l' sets of items is negligible by the choice of large W. We define r_i to be the fraction of items in set O'_i assigned to weight bins $\{b''_j\}_{j=i}^{l'}$ for any $1 \leq i \leq l'$. We want to prove that if $\sum_{i=1}^{(1-\gamma)l'}(1-r_i) \geq pl'$, the weight approximation ratio of the algorithm is at most $1 - 1/e - f(p)$. We note that group z is the first $1-\gamma$ fraction of these l' sets. We prove that even if the algorithm tries to allocate all items to weight bins $\{b''_i\}_{i=1}^{l'}$, it cannot saturate more than $1-1/e$ fraction of these l' weight bins in total. Define r'_j to be the fraction of weight bin b''_j which is filled with items $\{O'_i\}_{i=1}^{j}$ for any $1 \leq j \leq l'$. When set of items O'_i arrive, the algorithm do not see any difference between items $\{b''_j\}_{j=i}^{l'}$, and we choose their order to be a random permutation. So in

expectation, the algorithm cannot saturate more than $1/(l'-i+1)$ fraction of bin b''_j with set O'_i for any $1 \leq i \leq j \leq l'$. Therefore for any $1 \leq j \leq l'$, the fraction r'_j is at most $\min\{1, \sum_{i=1}^{j} 1/(l'-i+1)\}$. Summing up these upper bounds for all l' values of j gives us $(1-1/e)l'$ upper bound when l' goes to infinity. Applying the values of r_i's we get the following bound on the total filled fraction of weight bins $\{b''_j\}_{j=1}^{l'}$:

$$\sum_{j=1}^{l'} r'_j \leq \sum_{j=1}^{l'} \min\{1, \sum_{i=1}^{j} r_i/(l'-i+1)\} = \sum_{j=1}^{l'} \min\{1, \sum_{i=1}^{j} 1/(l'-i+1)\} -$$

$$\sum_{j=1}^{l'} \left(\min\{1, \sum_{i=1}^{j} 1/(l'-i+1)\} - \min\{1, \sum_{i=1}^{j} r_i/(l'-i+1)\} \right) =$$

$$(1-1/e)l' - \sum_{j=1}^{l'(1-1/e)} \sum_{i=1}^{j} (1-r_i)/(l'-i+1) -$$

$$\sum_{j=l'(1-1/e)+1}^{l'} \max \left\{ 0, \sum_{i=1}^{l'(1-1/e)} (1-r_i)/(l'-i+1) - \sum_{i'=l'(1-1/e)+1}^{j} r_{i'}/(l'-i'+1) \right\}$$

Now we want to upper bound the above expression by $(1-1/e-f(p))l'$ using the fact that $\sum_{i=1}^{l'(1-\gamma)}(1-r_i)$ is at least pl'. It is not hard to see that the above expression is maximized when $r_{i'} = 1$ for any $i' > l'(1-\gamma)$. We also observe that for any $1 \leq i \leq l'(1-\gamma)$, the fraction $1-r_i$ which is the lack of contribution of items in set O'_i to the weight bins is distributed evenly between bins $\{b_j\}_{j=i}^{l'}$. For the first $l'(1-1/e)$ bins, this lack will be counted for sure, but for other bins after the threshold $l'(1-1/e)$, there is some chance that sets of items $\{O'_{i'}\}_{i'=l'(1-1/e)+1}^{l'}$ cover up for the lack of previous items, and their lacks will not be counted in the sum. So the maximum of the above expression is achieved when the lacks are for bins with higher indices, because this way the lack will be distributed more on the bins after threshold $l'(1-1/e)$ and less before that. We conclude with the following upper bound on $\sum_{j=1}^{l'} r'_j$ inequality in which r_i is zero for $i \in [l'(1-\gamma-p), l'(1-\gamma)]$ and is

1 for other values of i:

$$(1-1/e)l' - \sum_{k=1}^{pl'} \frac{(k + l'(\gamma - 1/e))}{(k + l'\gamma)} - \sum_{k=1}^{l'/e} \left(\max \left\{ 0, \sum_{k'=1}^{pl'} \frac{1}{(k' + l'\gamma)} - \sum_{k''=1}^{k} \frac{1}{(l'/e - k'' + 1)} \right\} \right)$$

For large values of l and therefore l', we can write each sum in the above formula in an integral form by defining variable x to be the index of each sum divided by l'. We will have the above formula equal to:

$$l' \left(1 - 1/e - \int_{x=0}^{p} \frac{x + \gamma - 1/e}{x + \gamma} dx - \int_{x=0}^{p^*} \left(\ln(\frac{\gamma + p}{\gamma}) - \ln(\frac{1/e}{1/e - x}) \right) dx \right)$$

$$= l' \left(1 - 1/e - (p - \ln(\frac{p + \gamma}{\gamma})/e) - \frac{p}{e(\gamma + p)} \ln(\frac{\gamma + p}{\gamma}) + \frac{1}{e}(1 + \frac{\gamma \ln(\gamma/((\gamma + p)e))}{\gamma + p}) \right)$$

$$= 1 - 1/e - \frac{p}{e(\gamma + p)}$$

where p^* is some fraction such that for $x = p^*$ we have that $\frac{\gamma+p}{\gamma} = \frac{1/e}{1/e-x}$. In other words, we should take the integral until the point that the maximum of zero and the difference of the two summations becomes zero. Computing the integrals, and summing them up gives us a simple formula $1 - 1/e - p/e(\gamma + p)$ for the above expression.

□

We conclude this part with the main result of this subsection:

Theorem 3.2.4. *For any small $\varepsilon > 0$, $1/e \leq \gamma < 1$, and $0 \leq p \leq 1-\gamma$, any algorithm for bicriteria online matching problem with weight approximation guarantee, α, at least $1 - 1/e - f(p)$ cannot have cardinality approximation guarantee, β, greater than $p/(1 - \gamma) + \varepsilon$.*

Proof. Using Lemma 3.2.3, for any group $1 \leq z < r$ in any phase $1 \leq t \leq l$, we know that at most p fraction of items are assigned to cardinality bins, because $1 - 1/e - f(p)$ is a strictly increasing function in p. Since in each phase the number of items is decreasing with a factor of γ in consecutive groups, the total fraction of items assigned to cardinality bins is at most $p+p\gamma+p\gamma^2+\cdots+p\gamma^{r-2}$ plus the fraction of items assigned to cardinality in the last group r of phase t. Even if the algorithm assigns all of group r to cardinality, it does not achieve more than fraction γ^{r-1} from these items in each phase. Since the optimal cardinality algorithm can match all items, the cardinality approximation guarantee is at most $p(1+\gamma+\gamma^2+\cdots+\gamma^{r-2})+\gamma^{r-1}$. For large enough l (and consequently large enough r), this sum is not more than $p/(1-\gamma)+\varepsilon$. □

One way to compute the best values for p and γ corresponding to the best upper bound curve is to solve some corresponding complex equations explicitly. Instead, we compute these values numerically by trying different values of p and γ which, in turn, yield the orange upper bound curve in Figure 3-1.

3.3 Algorithm for Large Capacities

We now turn to algorithms, to see how close one can come to matching the upper bounds of the previous section. In this section, we assume that the capacity $n(a)$ of each bin $a \in A$ is "large", and give an algorithm with the guarantees in Theorem 3.1.1 as $\min_{a \in A} n(a) \to \infty$.

Recall that our algorithm `Alg` uses two subroutines WeightAlg and CardinalityAlg, each of which, if given an online item, suggests a bin to place it in. Each item i is independently passed to WeightAlg with probability p and CardinalityAlg with the remaining probability $1-p$. First note that CardinalityAlg and WeightAlg are independent and unaware of each other; each of them thinks that the only items which exist are those passed to it. This allows us to analyze the two subroutines separately.

We now describe how `Alg` uses the subroutines. If WeightAlg suggests matching

item i to a bin a, we match i to a. If a already has $n(a)$ items assigned to it in total, we remove any item assigned by CardinalityAlg arbitrarily; if all $n(a)$ were assigned by WeightAlg, we remove the item of lowest value for a. If CardinalityAlg suggests matching item i to a', we make this match unless a' has already had at least $n(a')$ total items assigned to it by both subroutines. In other words, the assignments of CardinalityAlg might be thrown away by some assignments of WeightAlg; however, the total number of items in a bin is always at least the the number assigned by CardinalityAlg. Items assigned by WeightAlg are never thrown away due to CardinalityAlg; they may only be replaced by later assignments of WeightAlg. Thus, we have proved the following proposition.

Proposition 3.3.1. *The weight and cardinality of the allocation of* `Alg` *are respectively at least as large as the weight of the allocation of* WeightAlg *and the cardinality of the allocation of* CardinalityAlg.

Note that the above proposition does not hold for any two arbitrary weight functions, and this is where we need one of the objectives to be cardinality. We now describe WeightAlg and CardinalityAlg, and prove Theorem 3.1.1. WeightAlg is essentially the exponentially-weighted primal-dual algorithm from [35], which was shown to achieve a $1 - \frac{1}{e}$ approximation for the weighted online matching problem with large degrees. For completeness, we present the primal and dual LP relaxations for weighted matching below, and then describe the algorithm. In the primal LP, for each item i and bin a, variable x_{ia} denotes whether impression i is one of the $n(a)$ most valuable items for bin a.

Primal			**Dual**		
max	$\sum_{i,a} w_{ia} x_{ia}$				
$\sum_a x_{ia} \leq$	1	$(\forall\ i)$	$\min \sum_a n(a)\beta_a +$	$\sum_i z_i$	
$\sum_i x_{ia} \leq$	$n(a)$	$(\forall\ a)$	$\beta_a + z_i \geq$	w_{ia}	$(\forall i, a)$
$x_{ia} \geq$	0	$(\forall\ i, a)$	$\beta_a, z_i \geq$	0	$(\forall i, a)$

Following the techniques of Buchbinder *et al.* [18], the algorithm of [35] simultaneously maintains feasible solutions to both the primal and dual LPs. Each dual

variable β_a is initialized to 0. When item i arrives online:

- Assign i to the bin $a' = \arg\max_a\{w_{ia} - \beta_a\}$. (If this quantity is negative for all a, discard i.)
- Set $x_{ia'} = 1$. If a' previously had $n(a')$ items assigned to it, set $x_{i'a'} = 0$ for the least valuable item i' previously assigned to a'.
- In the dual solution, set $z_i = w_{ia'} - \beta_{a'}$ and *update* dual variable $\beta_{a'}$ as described below.

Definition 3.3.2 (Exponential Weighting). *Let $w_1, w_2, \ldots w_{n(a)}$ be the weights of the $n(a)$ items currently assigned to bin a, sorted in non-increasing order, and padded with 0s if necessary.*
Set $\beta_a = \frac{1}{p \cdot n(a) \cdot \left((1+1/p \cdot n(a))^{n(a)} - 1\right)} \sum_{j=1}^{n(a)} w_j \left(1 + \frac{1}{p \cdot n(a)}\right)^{j-1}$.

Lemma 3.3.3. *If* WeightAlg *is the primal-dual algorithm, with dual variables β_a updated acccording to the exponential weighting rule defined above, the total weight of the allocation of* WeightAlg *is at least $p \cdot \left(1 - \frac{1}{k}\right)$ where $k = \left(1 + \frac{1}{p \cdot d}\right)^d$, and $d = \min_a\{n(a)\}$. Note that $\lim_{d \to \infty} k = e^{1/p}$.*

Before proving Lemma 3.3.3, we provide some brief intuition. If *all* items are passed to WeightAlg, it was proved in [35] that the algorithm has competitive ratio tending to $1 - 1/e$ as $d = \min_a\{n(a)\}$ tends to ∞; this is the statement of Lemma 3.3.3 when $p = 1$. Now, suppose each item is passed to WeightAlg with probability p. The expected value of the optimum matching induced by those items passed to WeightAlg is at least $p \cdot W_{\text{opt}}$, and this is *nearly* true (up to $o(1)$ terms) *even if we reduce the capacity of each bin a to $p \cdot n(a)$*. This follows since W_{opt} assigns at most $n(a)$ items to bin a, and as we are unlikely to sample more than $p \cdot n(a)$ of these items for the reduced instance, we do not lose much by reducing capacities. But note that WeightAlg can use the entire capacity $n(a)$, while there is a solution of value close to pW_{opt} even with capacities $p \cdot n(a)$. This extra capacity allows an improved competitive ratio of $1 - \frac{1}{e^{1/p}}$, proving the lemma.

Proof of Lemma 3.3.3. We construct a feasible dual solution as follows. Recall that the dual LP has a variable β_a for each bin a and z_i for each online item i. Though WeightAlg is unaware of items which are not passed to it, we maintain dual variables for these items as well, purely for the purpose of analysis. Let S denote the set of items passed to WeightAlg, and $I - S$ those passed to CardinalityAlg. For ease of notation, we use $e^{1/p}$ to represent $\left(1 + \frac{1}{p \cdot d}\right)^d$ where $d = \min_a\{n(a)\}$; as d tends to infinity, the latter expression tends to $e^{1/p}$.

For each item i, whether it is in S or not, we set $z_i = \max_a\{w_{ia} - \beta_a\}$, or $z_i = 0$ if $w_{ia} - \beta_a$ is negative for each bin a. For those items $i \in S$, if z_i is positive, we update β_a using the update rule of Definition 3.3.2. This gives a feasible solution to the dual LP defined on the entire set of items (including those in $I - S$).

In the previous analysis of weighted online matching in [35], following previous work of Buchbinder *et al.* [18], one shows a competitive ratio of $1 - 1/e$ by arguing that the change in the primal is at least $(1 - 1/e)$ times the change in the dual at each step. Since we end with a feasible primal and dual, the total value obtained by the algorithm (the value of the primal solution) is at least $(1 - 1/e) \cdot W_{\text{opt}}$.

Here, however, we do not compare the change in the primal to the change in the dual directly. This is because, when an item in $I - S$ arrives, there is a change in the dual, but perhaps no change in the primal as the item does not get passed to WeightAlg. To deal with this, we introduce a new 'reduced' objective function $\sum_{a \in A} p \cdot n(a)\beta_a + \sum_{i \in S} z_i$. We show that the change in primal is comparable to the change in this new reduced objective. Though new reduced objective is not an upper bound on the true optimal solution, it is not too difficult to see that it is at least $p \cdot W_{\text{opt}}$: For the terms involving variables β_a, we have simply scaled by p, but the reduced objective only includes the terms z_i for $i \in S$; this latter difference in the objectives requires some care.

First, though, we examine the relationship between the primal and the reduced objective. Consider how these change when an item i is sent to WeightAlg. If i is unassigned (if each $w_{ia} - \beta_a$ is non-positive), we have $z_i = 0$ and no change in either primal or dual. Otherwise, let a be the bin that item i is assigned to, and let v be

the value of the currently lowest-valued item assigned to bin a. The change in the primal is $w_{ia} - v$.

The change in the reduced objective is $z_i + p \cdot n(a)$ times the change in β_a. Let β_n, β_o represent the new and old values of β_a respectively. We assume that i now becomes the most valuable item for bin a.[1] Then, from the definition of our update rule for β_a, we have:

$$\beta_n = \left(1 + \frac{1}{p \cdot n(a)}\right)\beta_o + \frac{w_{ia}}{(p \cdot n(a))(e^{1/p} - 1)} - \frac{v \cdot e^{1/p}}{(p \cdot n(a))(e^{1/p} - 1)}$$

Overall, then, the change in the reduced objective is:

$$\begin{aligned} z_i + p \cdot n(a)\,(\beta_n - \beta_o) &= (w_{ia} - \beta_o) + \left(\beta_o + \frac{w_{ia}}{e^{1/p} - 1} - \frac{v \cdot e^{1/p}}{e^{1/p} - 1}\right) \\ &= w_{ia} + \frac{w_{ia}}{e^{1/p} - 1} - \frac{v \cdot e^{1/p}}{e^{1/p} - 1} \\ &= (w_{ia} - v) \cdot \left(\frac{e^{1/p}}{e^{1/p} - 1}\right) \\ &= (w_{ia} - v) \,/\, \left(1 - \frac{1}{e^{1/p}}\right) \end{aligned}$$

Since the change in the primal is $w_{ia} - v$, we have that the change in the primal is at least $1 - 1/e^{1/p}$ times the change in the reduced objective. It remains only to argue that the reduced objective is, in expectation, at least p times the original dual objective. Recall that the reduced objective is $\sum_{a \in A} p \cdot n(a)\beta_a + \sum_{i \in S} z_i$, while the original dual objective is $\sum_{a \in A} n(a)\beta_a + \sum_{i \in I} z_i$.

The terms involving β_a are simply scaled by p in the reduced objective, so for every run of the algorithm, the contribution of these terms to the reduced objective is exactly p times that to the original dual objective. However, of the terms involving z_is, the reduced objective only includes a subset which depends on which items are selected for S. As each item is selected for S with probability p independently, we would like to conclude that in expectation (though clearly not in every run), $\sum_{i \in S} z_i = p \sum_{i \in I} z_i$.

[1] It is easy to observe that this is the worst case for our algorithm, as this gives the maximum increase in the dual; see [35].

However, the value of z_i depends on the random allocation of items to S, since it depends on each β_a at the time item i arrives, and each β_a is affected only by those items in S. Still, it is possible to use linearity of expectation since z_i is not a function of whether i *itself* is assigned to S (in all cases, $z_i = w_{ia} - \beta_a$), only a function of which *previous* items were assigned to S. Thus, the expected contribution of each item to the reduced objective is exactly p times its expected contribution to the original dual objective; this can be verified by some straightforward algebra which we omit. □

Algorithm CardinalityAlg is identical to WeightAlg, except that it assumes all items have weight 1 for each bin. Since items are assigned to CardinalityAlg with probability $1 - p$, Lemma 3.3.3 implies the following corollary. This concludes the proof of Theorem 3.1.1.

Corollary 3.3.4. *The total cardinality of the allocation of* CardinalityAlg *is at least* $(1-p)\cdot\left(1 - \frac{1}{k}\right)$, *where* $k = \left(1 + \frac{1}{(1-p)\cdot d}\right)^d$, *and* $d = \min_a\{n(a)\}$. *Note that* $\lim_{d\to\infty} k = e^{1/(1-p)}$.

3.4 Algorithm for Small Capacities

We now consider algorithms for the case when the capacities of bins are not large. Without loss of generality, we assume that the capacity of each bin is one, because we can think about a bin with capacity c as c identical bins with capacity one. So we have a set A of bins each with capacity one, and a set of items I arriving online. As before, we use two subroutines WeightAlg and CardinalityAlg, but the algorithms are slightly different from those in the previous section.

In WeightAlg, we match item i (that has been passed to WeightAlg) to the bin that maximizes its marginal value. Formally we match i to bin $a = \arg\max_{a\in A}(w_{i,a} - w_{i',a})$ where i' is the last item assigned to a before item i.

In CardinalityAlg, we run the RANKING algorithm presented in [56]. So CardinalityAlg chooses a permutation π uniformly at random on the set of bins A, assigns an item i (that has been passed to it) to the bin a that is available, has the minimum rank in π, and there is also an edge between i and a.

3.4.1 Lower Bounding the Weight Approximation Ratio

Let $n = |I|$ be the number of items. We denote the ith arrived item by i. Let a_i be the bin that i is matched to in W_{opt} for any $1 \leq i \leq n$. One can assume that all unmatched items in the optimum weight allocation are matched with zero-weight edges to an imaginary bin. So W_{opt} is equal to $\sum_{i=1}^{n} w_{i,a_i}$. Let S be the set of items that have been passed to WeightAlg. If WeightAlg matches item i to bin a_j for some $j > i$, we call this a forwarding allocation (edge) because item j (the match of a_j in W_{opt}) has not arrived yet. We call it a selected forwarding edge if $j \in S$. We define the marginal value of assigning item i to bin a to be w_{ia} minus the value of any item previously assigned to a.

Lemma 3.4.1. *The weight of the allocation of* WeightAlg *is at least* $(p/(p+1))W_{opt}$.

Proof. Each forwarding edge will be a selected forwarding edge with probability p because $Pr[j \in S]$ is p for any $j \in I$. Let F be the total weight of forwarding edges of WeightAlg, where by weight of a forwarding edge, we mean its marginal value (not the actual weight of the edge). Similarly, we define F_s to be the sum of marginal values of selected forwarding edges. We have the simple equality that the expected value of F, $E(F)$, is $E(F_s)/p$. We define W' and W_s to be the total marginal values of allocation of WeightAlg, and the sum $\sum_{i \in S} w_{i,a_i}$. We know that $E(W_s)$ is pW_{opt} because $Pr[i \in S]$ is p. We prove that W' is at least $W_s - F_s$.

For every item i that has been selected to be matched by WeightAlg, we get at least marginal value w_{i,a_i} minus the sum of all marginal values of items that have been assigned to bin a by WeightAlg up to now. If we sum up all these lower bounds on our gains for all selected items, we get $W_s (= \sum_{i \in S} w_{i,a_i})$ minus the sum of all marginal values of items that has been assigned to a_i before item i arrives for all $i \in S$. The latter part is exactly the definition of F_s. Therefore W' is at least $W_s - F_s$. We also know that $W' \geq F$. Using $E[F] \geq E[F_s]/p$, we have that $E(W')$ is at least $E(W_s) - pE(W')$, and this yields the $p/(p+1)$ approximation factor. □

Corollary 3.4.2. *The weight and cardinality approximation guarantees of* `Alg` *are at least* $p/(p+1)$ *and* $(1-p)/(1-p+1)$ *respectively.*

3.4.2 Factor Revealing Linear Program for CardinalityAlg

Our goal in this subsection is to prove an approximation factor for CardinalityAlg better than the $(1-p)/(1-p+1)$ bound of Theorem 3.1.2 by formulating a factor-revealing LP that lower bounds it.

Proof of Theorem 3.1.2. We prove that the cardinality approximation ratio of CardinalityAlg is lower bounded by the solution of the linear program LP_k shown below for any positive integer k. We state the proof and the linear program LP_k for the simpler case of $p = 1/2$, and show the necessary changes for general $p \in [0,1]$ at the end of the proof. Before showing how this LP lower bounds the cardinality approximation factor, we note that the first three lines of constraints in LP_k hold for the weighted version and in fact give us the $1/3$ lower bound. The last inequality is specific to cardinality and makes an improvement on the lower bound from $1/3$ to almost 0.37.

$$
\begin{array}{llllll}
\text{Minimize:} & \beta & & & & \\
\forall 1 < i \leq k: & s_i \geq s_{i-1} & \& & sf_i \geq sf_{i-1} & \& & sb_i \geq sb_{i-1} \\
\forall 1 \leq i \leq k: & t_i \geq t_{i-1} & \& & t_i \geq sf_i & \& & s_i = sf_i + sb_i \\
 & \beta \geq s_k + t_k & \& & \beta \geq 1/2 - sf_k & & \\
\forall 1 < i \leq k: & s_i - s_{i-1} & \geq & 1/2k - (s_i + t_i)/k & &
\end{array}
$$

In CardinalityAlg, a uniformly random permutation π is selected on set A of bins. Let $A' \subseteq A$ be the set of bins matched in C_{opt}. We define A'' to be the set of bins matched in CardinalityAlg which depends on permutation π. We divide permutation π into k equal parts each with $|A|/k$ bins. For each $a \in A'$, we define $i(a)$ to be the match of a in C_{opt}. Let A^c be the set of bins like a such that $i(a)$ has been selected to be passed to CardinalityAlg. For each $1 \leq i \leq k$, we define sets S_i, and T_i as follows: $S_i = A' \cap A'' \cap A^c \cap \{\pi_j | 1 \leq j \leq (i|A|/k)\}$, and $T_i = A' \cap A'' \cap \{\pi_j | 1 \leq j \leq (i|A|/k)\} \setminus A^c$. In other words, S_i is the set of bins in the first i parts of π that are matched in both C_{opt} and our algorithm, and their matches in C_{opt} are selected to be passed to CardinalityAlg. The only difference for T_i is that their matches in C_{opt} are not selected for CardinalityAlg. We also partition S_i into two sets: SF_i the set of bins that has been matched in our algorithm with forwarding edges, and the rest in set SB_i. We

remind that a forwarding edge in our algorithm is an edge matching an item to a bin a such that $i(a)$ has not arrived yet. We prove that this is a feasible solution for LP_k: $s_i = E[|S_i|/|A'|]$, $sf_i = E[|SF_i|/|A'|]$, $sb_i = E[|SB_i|/|A'|]$, and $t_i = E[|T_i|/|A'|]$.

Now we prove the constraints of LP_k hold. The first 4 constraints are the monotonicity constraints which hold by definition, the same is true for the 6th constraint. The constraint $t_i \geq sf_i$ holds because every forwarding edge (incident to a bin in the first i parts of π) with probability $1/2$ will be counted in sf_i, and with the other $1/2$ will be counted in t_i. This way everything in sf_i will be counted, but there might be other uncounted bins in t_i. This is why we get an inequality instead of equality. We have β at least $1/2 - sf_k$ because in expectation half of the items matched in $C_{\text{opt}} = |A'|$ will be selected for cardinality, and the number of them that are not matched in CardinalityAlg is at most $sf_k|A'|$. The inequality $\beta \geq s_k + t_k$ holds by definition. These inequalities give us the $1/3$ lower bound on β. We add the following inequality as well to get a better approximation ratio for cardinality. We note that similar (and simpler) inequalities have been presented in the literature of online matching, e.g. Lemma 5 in [17].

$$\forall 1 \leq i \leq k, s_i - s_{i-1} \geq 1/2k - (s_i + t_i)/k$$

Let $s_0 = 0$ to make the math consistent. This inequality lower bounds the expected number of matched bins in $A' \cap A'' \cap A^c$ in part i of permutation π. For each $a \in A'$ in part i of permutation π, we prove that the probability that $a \in A^c$, and a is in A'' is at least $1/2 - (s_i + t_i)/k$. With probability $1/2$, a is in set A^c. In this case, we know that if no item is matched to a by CardinalityAlg, item $i(a)$ has been assigned by CardinalityAlg to some bin a' whose rank in π is smaller than a. This means that $i(a)$ is matched to one of the bins in S_i or T_i. Since π is selected uniformly at random, a is selected uniformly at random from $A' \cap A^c$. So the probability of this event ($i(a)$ being matched to some bin before a in π) given that $a \in A' \cap A^c$ is at most $(s_i + t_i)/|A' \cap A^c|$. Therefore with probability at least $(1 - (s_i + t_i)/|A' \cap A^c|)/2$, bin a is matched by CardinalityAlg. Summing up these lower bounds on probabilities for all

different choices of a, and applying the fact that in expectation there are $|A'|/k$ bins like $a \in A'$ in part i of π, we get the lower bound on $s_i - s_{i-1} \geq 1/2k - (s_i + t_i)/2k$. We note that $E(|A' \cap A^c|) = E(|A'|)/2$, and for large values of $|A'|$, the ratio $|A'|/|A' \cap A^c|$ is around 2 with high precision, i.e. it is at most $2+\varepsilon$ for any small constant $\varepsilon > 0$, and large enough $|A'|$. Since the solution for this linear program is greater than 0.3698 for large enough k, we know that the approximation ratio of CardinalityAlg and therefore our algorithm is at least 0.3698.

For any other value of p, we can change the LP_k as follows. We just need to change three constraints in LP_k. Instead of $t_i \geq sf_i$, we have $t_i/p \geq sf_i/(1-p)$ because with probability p we choose each item for weight and with the remaining probability $1-p$ for cardinality. With a similar reason, we have $\beta \geq (1-p) - sf_k$, and $s_i - s_{i-1} \geq (1-p)/k - (s_i + t_i)/k$ instead of their' simpler versions. We enumerate on different values of p, and solve the LP_k for each of these values to get the green lower bound curve in Figure 3-1. □

Chapter 4

Submodular Secretary Problem and its Extensions

Online auction is the essence of many modern markets, particularly networked markets, in which information about goods, agents, and outcomes is revealed over a period of time, and the agents must make irrevocable decisions without knowing future information. Optimal stopping theory is a powerful tool for analyzing such scenarios which generally require optimizing an objective function over the space of stopping rules for an allocation process under uncertainty. Combining optimal stopping theory with game theory allows us to model the actions of rational agents applying competing stopping rules in an online market. This first has been done by Hajiaghayi et al. [48] who considered the well-known *secretary problem* in online settings and initiated several follow-up papers (see e.g. [7, 8, 9, 47, 52, 59]).

Perhaps the most classic problem of stopping theory is the secretary problem. Imagine that you manage a company, and you want to hire a secretary from a pool of n applicants. You are very keen on hiring only the best and brightest. Unfortunately, you cannot tell how good a secretary is until you interview her, and you must make an irrevocable decision whether or not to make an offer at the time of the interview. The problem is to design a strategy which maximizes the probability of hiring the most qualified secretary. It is well-known since 1963 [27] that the optimal policy is to interview the first $t-1$ applicants, then hire the next one whose quality exceeds

that of the first $t-1$ applicants, where t is defined by $\sum_{j=t+1}^{n} \frac{1}{j-1} \leq 1 < \sum_{j=t}^{n} \frac{1}{j-1}$; as $n \to \infty$, the probability of hiring the best applicant approaches $1/e$, as does the ratio t/n. Note that a solution to the secretary problem immediately yields an algorithm for a slightly different objective function optimizing the expected value of the chosen element. Subsequent papers have extended the problem by varying the objective function, varying the information available to the decision-maker, and so on, see e.g., [3, 41, 75, 80].

An important generalization of the secretary problem with several applications (see e.g., a survey by Babaioff et al. [8]) is called the *multiple-choice secretary problem* in which the interviewer is allowed to hire up to $k \geq 1$ applicants in order to maximize performance of the secretarial group based on their overlapping skills (or the joint utility of selected items in a more general setting). More formally, assuming applicants of a set $S = \{a_1, a_2, \cdots, a_n\}$ (applicant pool) arriving in a uniformly random order, the goal is to select a set of at most k applicants in order to maximize a profit function $f : 2^S \mapsto \mathbb{R}$. We assume f is non-negative throughout this chapter. For example, when $f(T)$ is the maximum individual value [39, 40], or when $f(T)$ is the sum of the individual values in T [59], the problem has been considered thoroughly in the literature. Indeed, both of these cases are special monotone non-negative submodular functions that we consider in this chapter. A function $f : 2^S \mapsto \mathbb{R}$ is called *submodular* if and only if $\forall A, B \subseteq S : f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. An equivalent characterization is that the marginal profit of each item should be non-increasing, i.e., $f(A \cup \{a\}) - f(A) \leq f(B \cup \{a\}) - f(B)$ if $B \subseteq A \subseteq S$ and $a \in S \setminus B$. A function $f : 2^S \mapsto \mathbb{R}$ is *monotone* if and only if $f(A) \leq f(B)$ for $A \subseteq B \subseteq S$; it is *non-monotone* if it is not necessarily the case. Since the number of sets is exponential, we assume a value oracle access to the submodular function; i.e., for a given set T, an algorithm can query an oracle to find its value $f(T)$. As we discuss below, maximizing a (monotone or non-monotone) submodular function which demonstrates economy of scale is a central and very general problem in combinatorial optimization and has been subject of a thorough study in the literature.

The closest setting to our submodular multiple-choice secretary problem is the *ma-*

troid secretary problem considered by Babaioff et al. [9]. In this problem, we are given a matroid by a ground set $\mathcal{U}$ of elements and a collection of independent (feasible) subsets $\mathcal{I} \subseteq 2^{\mathcal{U}}$ describing the sets of elements which can be simultaneously accepted. We recall that a matroid has three properties: 1) the empty set is independent; 2) every subset of an independent set is independent (closed under containment)[1]; and finally 3) if A and B are two independent sets and A has more elements than B, then there exists an element in A which is not in B and when added to B still gives an independent set[2]. The goal is to design online algorithms in which the structure of $\mathcal{U}$ and $\mathcal{I}$ is known at the outset (assume we have an oracle to answer whether a subset of $\mathcal{U}$ belongs to $\mathcal{I}$ or not), while the elements and their values are revealed one at a time in random order. As each element is presented, the algorithm must make an irrevocable decision to select or reject it such that the set of selected elements belongs to $\mathcal{I}$ at all times. Babaioff et al. present an $O(\log r)$-competitive algorithm for general matroids, where r is the rank of the matroid (the size of the maximal independent set), and constant-competitive algorithms for several special cases arising in practical scenarios including graphic matroids, truncated partition matroids, and bounded degree transversal matroids. However, they leave as a main open question the existence of constant-competitive algorithms for general matroids. Our constant-competitive algorithms for the submodular secretary problem in this chapter can be considered in parallel with this open question. To generalize both results of Babaioff et al. and ours, we also consider the *submodular matroid secretary problem* in which we want to maximize a submodular function over all independent (feasible) subsets $\mathcal{I}$ of the given matroid. Moreover, we extend our approach to the case in which l matroids are given and the goal is to find the set of maximum value which is independent with respect to all the given matroids. We present an $O(l \log^2 r)$-competitive algorithm for the submodular matroid secretary problem generalizing previous results.

Prior to our work, there was no polynomial-time algorithm with a nontrivial guarantee for the case of l matroids—even in the offline setting—when l is not a fixed

[1]This is sometimes called the *hereditary property.*

[2]This is sometimes called the *augmentation property* or the *independent set exchange property.*

constant. Lee et al. [61] give a local-search procedure for the offline setting that runs in time $O(n^l)$ and achieves approximation ratio $l+\varepsilon$. Even the simpler case of having a linear function cannot be approximated to within a factor better than $\Omega(l/\log l)$ [51]. Our results imply an algorithm with guarantees $O(l \log r)$ and $O(l \log^2 r)$ for the offline and (online) secretary settings, respectively. Both these algorithms run in time polynomial in l. In case of the knapsack constraints, the only previous relevant work that we are aware of is that of Lee et al. [61] which gives a $(5+\varepsilon)$ approximation in the offline setting if the number of constraints is a constant. In contrast, our results work for arbitrary number of knapsack constraints, albeit with a loss in the guarantee; see Theorem 4.1.3.

Our competitive ratio for the submodular secretary problem is $\frac{7}{1-1/e}$. Though our algorithm is relatively simple, it has several phases and its analysis is relatively involved. As we point out below, we cannot obtain any approximation factor better than $1 - 1/e$ even for offline special cases of our setting unless $\mathbf{P} = \mathbf{NP}$. A natural generalization of a submodular function while still preserving economy of scale is a subadditive function $f : 2^S \mapsto \mathbb{R}$ in which $\forall A, B \subseteq S : f(A) + f(B) \geq f(A \cup B)$. In this chapter, we show that if we consider the subadditive secretary problem instead of the submodular secretary problem, there is no algorithm with competitive ratio $\tilde{o}(\sqrt{n})$. We complement this result by giving an $O(\sqrt{n})$-competitive algorithm for the subadditive secretary problem.

Background on submodular maximization Submodularity, a discrete analog of convexity, has played a central role in combinatorial optimization [62]. It appears in many important settings including cuts in graphs [53, 44, 71], plant location problems [22, 21], rank function of matroids [28], and set covering problems [30].

The problem of maximizing a submodular function is of essential importance, with special cases including Max Cut [44], Max Directed Cut [49], hypergraph cut problems, maximum facility location [1, 22, 21], and certain restricted satisfiability problems [50, 29]. While the Min Cut problem in graphs is a classical polynomial-time solvable problem, and more generally it has been shown that any submodular

function can be minimized in polynomial time [53, 72], maximization turns out to be more difficult and indeed all the aforementioned special cases are NP-hard.

Max-k-Cover, where the goal is to choose k sets whose union is as large as possible, is another related problem. It is shown that a greedy algorithm provides a $(1 - 1/e)$ approximation for Max-k-Cover [58] and this is optimal unless $\mathbf{P} = \mathbf{NP}$ [30]. More generally, we can view this problem as maximization of a monotone submodular function under a cardinality constraint, that is, we seek a set S of size k maximizing $f(S)$. The greedy algorithm again provides a $(1 - 1/e)$ approximation for this problem [69]. A 1/2 approximation has been developed for maximizing monotone submodular functions under a matroid constraint [37]. A $(1 - 1/e)$ approximation has been also obtained for a knapsack constraint [73], and for a special class of submodular functions under a matroid constraint [20].

Recently constant factor $(\frac{3}{4} + \varepsilon)$-approximation algorithms for maximizing nonnegative non-monotone submodular functions has also been obtained [32]. Typical examples of such a problem are max cut and max directed cut. Here, the best approximation factors are 0.878 for max cut [44] and 0.859 for max directed cut [29]. The approximation factor for max cut has been proved optimal, assuming the Unique Games Conjecture [57]. Generalizing these results, Vondrák very recently obtains a constant factor approximation algorithm for maximizing non-monotone submodular functions under a matroid constraint [77]. Subadditive maximization has been also considered recently (e.g. in the context of maximizing welfare [31]).

Submodular maximization also plays a role in maximizing the difference of a monotone submodular function and a modular function. A typical example of this type is the maximum facility location problem in which we want to open a subset of facilities and maximize the total profit from clients minus the opening cost of facilities. Approximation algorithms have been developed for a variant of this problem which is a special case of maximizing nonnegative submodular functions [1, 22, 21]. The current best approximation factor known for this problem is 0.828 [1]. Asadpour et al. [5] study the problem of maximizing a submodular function in a stochastic setting, and obtain constant-factor approximation algorithms.

4.1 Our Results and Techniques

The main theorem in this chapter is as follows.

Theorem 4.1.1. *There exists a $\frac{7}{1-1/e}$-competitive algorithm for the monotone submodular secretary problem. More generally there exists a $8e^2$-competitive algorithm for the non-monotone submodular secretary problem.*

We prove Theorem 4.1.1 in Section 4.2. We first present our simple algorithms for the problem. Since our algorithm for the general non-monotone case uses that of monotone case, we first present the analysis for the latter case and then extend it for the former case. We divide the input stream into equal-sized segments, and show that restricting the algorithm to pick only one item from each segment decreases the value of the optimum by at most a constant factor. Then in each segment, we use a standard secretary algorithm to pick the best item conditioned on our previous choices. We next prove that these local optimization steps lead to a global near-optimal solution.

The argument breaks for the non-monotone case since the algorithm actually approximates a set which is larger than the optimal solution. The trick is to invoke a new structural property of (non-monotone) submodular functions which allows us to divide the input into two equal portions, and randomly solve the problem on one.

Indeed Theorem 4.1.1 can be extended for the submodular matroid secretary problem as follows.

Theorem 4.1.2. *There exists an $O(l \log^2 r)$ competitive algorithm for the (non-monotone) matroid submodular secretary problem, where r is the maximum rank of the given l matroids.*

We prove theorem 4.1.2 in Section 4.3. We note that in the submodular matroid secretary problem, selecting (bad) elements early in the process might prevent us from selecting (good) elements later since there are matroid independence (feasibility) constraints. To overcome this issue, we only work with the first half of the input. This guarantees that at each point in expectation there is a large portion of the optimal solution that can be added to our current solution without violating the matroid

constraint. However, this set may not have a high value. As a remedy we prove there is a near-optimal solution all of whose large subsets have a high value. This novel argument may be of its own interest.

We shortly mention in Section 4.4 our results for maximizing a submodular secretary problem with respect to l knapsack constraints. In this setting, there are l knapsack capacities $C_i : 1 \leq i \leq l$, and each item j has different weights w_{ij} associated with each knapsack. A set T of items is feasible if and only if for *each* knapsack i, we have $\sum_{j \in T} w_{ij} \leq C_i$.

Theorem 4.1.3. *There exists an $O(l)$-competitive algorithm for the (non-monotone) multiple knapsack submodular secretary problem, where l denotes the number of given knapsack constraints.*

Lee et al. [61] gives a better $(5 + \varepsilon)$ approximation in the offline setting if l is a fixed constant.

We next show that indeed submodular secretary problems are the most general cases that we can hope for constant competitiveness.

Theorem 4.1.4. *For the subadditive secretary problem, there is no algorithm with competitive ratio in $\tilde{o}(\sqrt{n})$. However there is an algorithm with almost tight $O(\sqrt{n})$ competitive ratio in this case.*

We prove Theorem 4.1.4 in Section 4.5. The algorithm for the matching upper bound is very simple, however the lower bound uses clever ideas and indeed works in a more general setting. We construct a subadditive function, which interestingly is almost submodular, and has a "hidden good set". Roughly speaking, the value of any query to the oracle is proportional to the intersection of the query and the hidden good set. However, the oracle's response does not change unless the query has considerable intersection with the good set which is hidden. Hence, the oracle does not give much information about the hidden good set.

Finally in our concluding remarks in Section 4.6, we briefly discuss two other aggregate functions max and min, where the latter is not even submodular and models a bottle-neck situation in the secretary problem.

Remark Subsequent to our study of online submodular maximization [12], Gupta et al. [45] consider similar problems. By reducing the case of non-monotone submodular functions to several runs of the greedy algorithm for monotone submodular functions, they present $O(p)$-approximation algorithms for maximizing submodular functions (in the offline setting) subject to p-independence systems (which include the intersection of p matroids), and constant factor approximation algorithms when the maximization is subject to a knapsack constraint. In the online secretary setting, they provide $O(1)$-competitive results for maximizing a submodular function subject to cardinality or partition matroid constraints. They also obtain an $O(\log r)$ competitive ratio for maximization subject to a general matroid of rank r. The latter result improves our Theorem 4.1.2 when $l = 1$.

4.2 The Submodular Secretary Problem

4.2.1 Algorithms

In this section, we present the algorithms used to prove Theorem 4.1.1. In the classic secretary problem, the efficiency value of each secretary is known only after she arrives. In order to marry this with the value oracle model, we say that the oracle answers the query regarding the efficiency of a set $S' \subseteq S$ only if all the secretaries in S' have already arrived and been interviewed.

Our algorithm for the monotone submodular case is relatively simple though its analysis is relatively involved. First we assume that n is a multiple of k, since otherwise we could virtually insert $n - k\lfloor \frac{n}{k} \rfloor$ dummy secretaries in the input: for any subset A of dummy secretaries and a set $B \subseteq S$, we have that $f(A \cup B) = f(B)$. In other words, there is no profit in employing the dummy secretaries. To be more precise, we simulate the augmented input in such a way that these secretaries are arriving uniformly at random similarly to the real ones. Thus, we say that n is a multiple of k without loss of generality.

We partition the input stream into k equally-sized segments, and, roughly speak-

Algorithm 1 Monotone Submodular Secretary Algorithm

Input: A monotone submodular function $f : 2^S \mapsto \mathbb{R}$, and a randomly permuted stream of secretaries, denoted by $(a_1, a_2, \ldots, a_n)$, where n is an integer multiple of k.
Output: A subset of at most k secretaries.

Let $T_0 \leftarrow \emptyset$
Let $l \leftarrow n/k$
for $i \leftarrow 1$ **to** k **do** {phase i}
  Let $u_i \leftarrow (i-1)l + l/e$
  Let $\alpha_i \leftarrow \max_{(i-1)l \leq j < u_i} f(T_{i-1} \cup \{a_j\})$
  if $\alpha_i < f(T_{i-1})$ **then**
    $\alpha_i \leftarrow f(T_{i-1})$
  end if
  Pick an index $p_i : u_i \leq p_i < il$ such that $f(T_{i-1} \cup \{a_{p_i}\}) \geq \alpha_i$
  if such an index p_i exists **then**
    Let $T_i \leftarrow T_{i-1} \cup \{a_{p_i}\}$
  else
    Let $T_i \leftarrow T_{i-1}$
  end if
end for
Output T_k as the solution

ing, try to employ the *best* secretary in each segment. Let $l := \frac{n}{k}$ denote the length of each segment. Let $a_1, a_2, \cdots, a_n$ be the actual ordering in which the secretaries are interviewed. Break the input into k segments such that $S_j = \{a_{(j-1)l+1}, a_{(j-1)l+2}, \ldots, a_{jl}\}$ for $1 \leq j < k$, and $S_k = \{a_{(k-1)l+1}, a_{(k-1)l+2}, \ldots, a_n\}$. We employ at most one secretary from each segment S_i. Note that this way of having several phases of (almost) equal length for the secretary problem seems novel to this chapter, since in previous works there are usually only two phases (see e.g. [48]). The phase i of our algorithm corresponds to the time interval when the secretaries in S_i arrive. Let T_i be the set of secretaries that we have employed from $\bigcup_{j=1}^{i} S_j$. Define $T_0 := \emptyset$ for convenience. In phase i, we try to employ a secretary e from S_i that maximizes $f(T_{i-1} \cup \{e\}) - f(T_{i-1})$. For each $e \in S_i$, we define $g_i(e) = f(T_{i-1} \cup \{e\}) - f(T_{i-1})$. Then, we are trying to employ a secretary $x \in S_i$ that has the maximum value for $g_i(e)$. Using a classic algorithm for the *secretary problem* (see [27] for instance) for employing the single secretary, we can solve this problem with constant probability $1/e$. Hence, with constant probability, we pick the secretary that maximizes our local

profit in each phase. It leaves us to prove that this local optimization leads to a reasonable global guarantee.

The previous algorithm fails in the non-monotone case. Observe that the first **if** statement is never true for a monotone function, however, for a non-monotone function this guarantees the values of sets T_i are non-decreasing. Algorithm 2 first divides the input stream into two equal-sized parts: U_1 and U_2. Then, with probability $1/2$, it calls Algorithm 1 on U_1, whereas with the same probability, it skips over the first half of the input, and runs Algorithm 1 on U_2.

Algorithm 2 Submodular Secretary Algorithm

Input: A (possibly non-monotone) submodular function $f : 2^S \mapsto \mathbb{R}$, and a randomly permuted stream of secretaries, denoted by $(a_1, a_2, \ldots, a_n)$, where n is an integer multiple of $2k$.
Output: A subset of at most k secretaries.

Let $U_1 := \{a_1, a_2, \ldots, a_{n/2}\}$
Let $U_2 := \{a_{n/2} + 1, \ldots, a_{n-1}, a_n\}$
Let $0 \leq X \leq 1$ be a uniformly random value.
if $X \leq 1/2$ **then**
  Run Algorithm 1 on U_1 to get S_1
  Output S_1 as the solution
else
  Run Algorithm 1 on U_2 to get S_2
  Output S_2 as the solution
end if

4.2.2 Analysis

In this section, we prove Theorem 4.1.1. Since the algorithm for the non-monotone submodular secretary problem uses that for the monotone submodular secretary problem, first we start with the monotone case.

Monotone Submodular

We prove in this section that for Algorithm 1, the expected value of $f(T_k)$ is within a constant factor of the optimal solution. Let $R = \{a_{i_1}, a_{i_2}, \cdots, a_{i_k}\}$ be the optimal solution. Note that the set $\{i_1, i_2, \cdots, i_k\}$ is a uniformly random subset of $\{1, 2, \cdots, n\}$

with size k. It is also important to note that the permutation of the elements of the optimal solution on these k places is also uniformly random, and is independent from the set $\{i_1, i_2, \cdots, i_k\}$. For example, any of the k elements of the optimum can appear as a_{i_1}. These are two key facts used in the analysis.

Before starting the analysis, we present a simple property of submodular functions which will prove useful in the analysis.

Lemma 4.2.1. *If $f : 2^S \mapsto \mathbb{R}$ is a submodular function, we have $f(B) - f(A) \leq \sum_{a \in B \setminus A} [f(A \cup \{a\}) - f(A)]$ for any $A \subseteq B \subseteq S$.*

Proof. Let $k := |B| - |A|$. Then, define in an arbitrary manner sets $\{B_i\}_{i=0}^{k}$ such that

- $B_0 = A$,
- $|B_i \setminus B_{i-1}| = 1$ for $i : 1 \leq i \leq k$,
- and $B_k = B$.

Let $b_i := B_i \setminus B_{i-1}$ for $i : 1 \leq i \leq k$. We can write $f(B) - f(A)$ as follows

$$\begin{aligned} f(B) - f(A) &= \sum_{i=1}^{k} [f(B_i) - f(B_{i-1})] \\ &= \sum_{i=1}^{k} [f(B_{i-1} \cup \{b_i\}) - f(B_{i-1})] \\ &\leq \sum_{i=1}^{k} [f(A \cup b_i) - f(A)], \end{aligned}$$

where the last inequality follows from the non-increasing marginal profit property of submodular functions. Noticing that $b_i \in B \setminus A$ and they are distinct, namely $b_i \neq b_{i'}$ for $1 \leq i \neq i' \leq k$, finishes the argument. □

Define $\mathcal{X} := \{S_i : |S_i \cap R| \neq \emptyset\}$. For each $S_i \in \mathcal{X}$, we pick one element, say s_i, of $S_i \cap R$ randomly. These selected items form a set called $R' = \{s_1, s_2, \cdots, s_{|\mathcal{X}|}\} \subseteq R$ of size $|\mathcal{X}|$. Since our algorithm approximates such a set, we study the value of such random samples of R in the following lemmas. We first show that restricting ourselves

to picking at most one element from each segment does not prevent us from picking many elements from the optimal solution (i.e., R).

Lemma 4.2.2. *The expected value of the number of items in R' is at least $k(1-1/e)$.*

Proof. We know that $|R'| = |\mathcal{X}|$, and $|\mathcal{X}|$ is equal to k minus the number of sets S_i whose intersection with R is empty. So, we compute the expected number of these sets, and subtract this quantity from k to obtain the expected value of $|\mathcal{X}|$ and thus $|R'|$.

Consider a set S_q, $1 \leq q \leq k$, and the elements of $R = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$. Define $\mathcal{E}_j$ as the event that a_{i_j} is not in S_q. We have $\Pr(\mathcal{E}_1) = \frac{(k-1)l}{n} = 1 - \frac{1}{k}$, and for any $i : 1 < i \leq k$, we get

$$\Pr\left(\mathcal{E}_i \middle| \bigcap_{j=1}^{i-1} \mathcal{E}_j\right) = \frac{(k-1)l - (i-1)}{n - (i-1)} \leq \frac{(k-1)l}{n} = 1 - \frac{1}{k},$$

where the last inequality follows from a simple mathematical fact: $\frac{x-c}{y-c} \leq \frac{x}{y}$ if $c \geq 0$ and $x \leq y$. Now we conclude that the probability of the event $S_q \cap R = \emptyset$ is

$$\Pr(\cap_{i=1}^{k} \mathcal{E}_i) = \Pr(\mathcal{E}_1) \cdot \Pr(\mathcal{E}_2|\mathcal{E}_1) \cdots \Pr(\mathcal{E}_k| \cap_{j=1}^{k-1} \mathcal{E}_j) \leq \left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}.$$

Thus each of the sets $S_1, S_2, \ldots, S_k$ does not intersect with R with probability at most $1/e$. Hence, the expected number of such sets is at most k/e. Therefore, the expected value of $|\mathcal{X}| = |R'|$ is at least $k(1 - 1/e)$. □

The next lemma materializes the proof of an intuitive statement: if you randomly sample elements of the set R, you expect to obtain a profit proportional to the size of your sample. An analog of this is proved in [31] for the case when $|R|/|A|$ is an integer.

Lemma 4.2.3. *For a random subset A of R, the expected value of $f(A)$ is at least $\frac{|A|}{k} \cdot f(R)$.*

Proof. Let $(x_1, x_2, \ldots, x_k)$ be a random ordering of the elements of R. For $r = 1, 2, \ldots, k$, let F_r be the expectation of $f(\{x_1, \ldots, x_r\})$, and define $D_r := F_r - F_{r-1}$,

where F_0 is interpreted to be equal to zero. Letting $a := |A|$, note that $f(R) = F_k = D_1 + \cdots + D_k$, and that the expectation of $f(A)$ is equal to $F_a = D_1 + \cdots + D_a$. We claim that $D_1 \geq D_2 \geq \cdots \geq D_k$, from which the lemma follows easily. Let $(y_1, y_2, \ldots, y_k)$ be a cyclic permutation of $(x_1, x_2, \ldots, x_k)$, where $y_1 = x_k, y_2 = x_1, y_3 = x_2, \ldots, y_k = x_{k-1}$. Notice that for $i < k$, F_i is equal to the expectation of $f(\{y_2, \ldots, y_{i+1}\})$ since $\{y_2, \ldots, y_{i+1}\}$ is equal to $\{x_1, \ldots, x_i\}$.

F_i is also equal to the expectation of $f(\{y_1, \ldots, y_i\})$, since the sequence $(y_1, \ldots, y_i)$ has the same distribution as that of $(x_1, \cdots, x_i)$. Thus, D_{i+1} is the expectation of $f(\{y_1, \ldots, y_{i+1}\}) - f(\{y_2, \ldots, y_{i+1}\})$, whereas D_i is the expectation of $f(\{y_1, \ldots, y_i\}) - f(\{y_2, \ldots, y_i\})$. The submodularity of f implies that $f(\{y_1, \ldots, y_{i+1}\}) - f(\{y_2, \ldots, y_{i+1}\})$ is less than or equal to $f(\{y_1, \ldots, y_i\}) - f(\{y_2, \ldots, y_i\})$, hence $D_{i+1} \leq D_i$. □

Here comes the crux of our analysis where we prove that the local optimization steps (i.e., trying to make the best move in each segment) indeed lead to a globally approximate solution.

Lemma 4.2.4. *The expected value of $f(T_k)$ is at least $\frac{|R'|}{7k} \cdot f(R)$.*

Proof. Define $m := |R'|$ for the ease of reference. Recall that R' is a set of secretaries $\{s_1, s_2, \ldots, s_m\}$ such that $s_i \in S_{h_i} \cap R$ for $i : 1 \leq i \leq m$ and $h_i : 1 \leq h_i \leq k$. Also assume without loss of generality that $h_{i'} \leq h_i$ for $1 \leq i' < i \leq m$, for instance, s_1 is the first element of R' to appear. Define Δ_j for each $j : 1 \leq j \leq k$ as the gain of our algorithm while working on the segment S_j. It is formally defined as $\Delta_j := f(T_j) - f(T_{j-1})$. Note that due to the first **if** statement in the algorithm, $\Delta_j \geq 0$ and thus $\mathbf{E}[\Delta_j] \geq 0$. With probability $1/e$, we choose the element in S_j which maximizes the value of $f(T_j)$ (given that the set T_{j-1} is fixed). Notice that by definition of R' only one s_i appears in S_{h_i}. Since $s_i \in S_{h_i}$ is one of the options,

$$\mathbf{E}[\Delta_{h_i}] \geq \frac{\mathbf{E}[f(T_{h_i-1} \cup \{s_i\}) - f(T_{h_i-1})]}{e}. \tag{4.1}$$

To prove by contradiction, suppose $\mathbf{E}[f(T_k)] < \frac{m}{7k} \cdot f(R)$. Since f is monotone, $\mathbf{E}[f(T_j)] < \frac{m}{7k} \cdot f(R)$ for any $0 \leq j \leq k$. Define $B := \{s_i, s_{i+1}, \cdots, s_m\}$. By

Lemma 4.2.1 and monotonicity of f,

$$f(B) \leq f(B \cup T_{h_i-1}) \leq f(T_{h_i-1}) + \sum_{j=i}^{m} [f(T_{h_i-1} \cup \{s_j\}) - f(T_{h_i-1})],$$

which implies

$$\mathbf{E}[f(B)] \leq \mathbf{E}[f(T_{h_i-1})] + \sum_{j=i}^{m} \mathbf{E}[f(T_{h_i-1} \cup \{s_j\}) - f(T_{h_i-1})].$$

Since the items in B are distributed uniformly at random, and there is no difference between s_{i_1} and s_{i_2} for $i \leq i_1, i_2 \leq m$, we can say

$$\mathbf{E}[f(B)] \leq \mathbf{E}[f(T_{h_i-1})] + (m-i+1) \cdot \mathbf{E}[f(T_{h_i-1} \cup \{s_i\}) - f(T_{h_i-1})]. \tag{4.2}$$

We conclude from (4.1) and (4.2)

$$\mathbf{E}[\Delta_{h_i}] \geq \frac{\mathbf{E}[f(T_{h_i-1} \cup \{s_i\}) - f(T_{h_i-1})]}{e} \geq \frac{\mathbf{E}[f(B)] - \mathbf{E}[f(T_{h_i-1})]}{e(m-i+1)}.$$

Since B is a random sample of R, we can apply Lemma 4.2.3 to get $\mathbf{E}[f(B)] \geq \frac{|B|}{k} f(R) = f(R)(m-i+1)/k$. Since $\mathbf{E}[f(T_{h_i-1})] \leq \frac{m}{7k} \cdot f(R)$, we reach

$$\mathbf{E}[\Delta_{h_i}] \geq \frac{\mathbf{E}[f(B)] - \mathbf{E}[f(T_{h_i-1})]}{e(m-i+1)} \geq \frac{f(R)}{ek} - \frac{m}{7k} f(R) \cdot \frac{1}{e(m-i+1)}. \tag{4.3}$$

Adding up (4.3) for $i : 1 \leq i \leq \lceil m/2 \rceil$, we obtain

$$\sum_{i=1}^{\lceil m/2 \rceil} \mathbf{E}[\Delta_{h_i}] \geq \left\lceil \frac{m}{2} \right\rceil \cdot \frac{f(R)}{ek} - \frac{m}{7ek} \cdot f(R) \cdot \sum_{i=1}^{\lceil m/2 \rceil} \frac{1}{m-i+1}.$$

Since $\sum_{j=a}^{b} \frac{1}{j} \leq \ln \frac{b}{a+1}$ for any integer values of $a, b : 1 < a \leq b$, we conclude

$$\sum_{i=1}^{\lceil m/2 \rceil} \mathbf{E}[\Delta_{h_i}] \geq \left\lceil \frac{m}{2} \right\rceil \cdot \frac{f(R)}{ek} - \frac{m}{7ek} \cdot f(R) \cdot \ln \frac{m}{\lfloor \frac{m}{2} \rfloor}.$$

A similar argument for the range $1 \leq i \leq \lfloor m/2 \rfloor$ gives

$$\sum_{i=1}^{\lfloor \frac{m}{2} \rfloor} \mathbf{E}[\Delta_{h_i}] \geq \left\lfloor \frac{m}{2} \right\rfloor \cdot \frac{f(R)}{ek} - \frac{m}{7ek} \cdot f(R) \cdot \ln \frac{m}{\lceil \frac{m}{2} \rceil}.$$

We also know that both $\sum_{i=1}^{\lfloor m/2 \rfloor} \mathbf{E}[\Delta_{h_i}]$ and $\sum_{i=1}^{\lceil m/2 \rceil} \mathbf{E}[\Delta_{h_i}]$ are at most $\mathbf{E}[f(T_k)]$ because $f(T_k) \geq \sum_{i=1}^{m} \Delta_{h_i}$. We conclude with

$$\begin{aligned}
2\mathbf{E}[f(T_k)] &\geq \left\lceil \frac{m}{2} \right\rceil \frac{f(R)}{ek} - \frac{mf(R)}{7ek} \cdot \ln \frac{m}{\lfloor \frac{m}{2} \rfloor} + \left\lfloor \frac{m}{2} \right\rfloor \frac{f(R)}{ek} - \frac{mf(R)}{7ek} \cdot \ln \frac{m}{\lceil \frac{m}{2} \rceil} \\
&\geq \frac{mf(R)}{ek} - \frac{mf(R)}{7ek} \cdot \ln \frac{m^2}{\lfloor \frac{m}{2} \rfloor \lceil \frac{m}{2} \rceil}, \qquad \text{and since } \frac{m^2}{\lfloor m/2 \rfloor \lceil m/2 \rceil} < 4.5 \\
&\geq \frac{mf(R)}{ek} - \frac{mf(R)}{7ek} \cdot \ln 4.5 = \frac{mf(R)}{k} \cdot \left(\frac{1}{e} - \frac{\ln 4.5}{7e} \right) \geq \frac{mf(R)}{k} \cdot \frac{2}{7},
\end{aligned}$$

which contradicts $\mathbf{E}[f(T_k)] < \frac{mf(R)}{7k}$, hence proving the supposition false. □

The following theorem wraps up the analysis of the algorithm.

Theorem 4.2.5. *The expected value of the output of our algorithm is at least* $\frac{1-1/e}{7} f(R)$.

Proof. The expected value of $|R'| = m \geq (1 - 1/e)k$ from Lemma 4.2.2. In other words, we have $\sum_{m=1}^{k} \Pr[|R'| = m] \cdot m \geq \left(1 - \frac{1}{e}\right) k$. We know from Lemma 4.2.4 that if the size of R' is m, the expected value of $f(T_k)$ is at least $\frac{m}{7k} f(R)$, implying that $\sum_{v \in V} \Pr\left[f(T_k) = v \middle| |R'| = m\right] \cdot v \geq \frac{m}{7k} f(R)$, where V denotes the set of different values that $f(T_k)$ can get. We also know that

$$\begin{aligned}
\mathbf{E}[f(T_k)] &= \sum_{m=1}^{k} \mathbf{E}[f(T_k) | |R'| = m] \Pr[|R'| = m] \geq \sum_{m=1}^{k} \frac{m}{7k} f(R) \Pr[|R'| = m] \\
&= \frac{f(R)}{7k} \mathbf{E}[|R'|] \geq \frac{1 - 1/e}{7} f(R).
\end{aligned}$$

□

Non-monotone Submodular

Before starting the analysis of Algorithm 2 for non-monotone functions, we show an interesting property of Algorithm 1. Consistently with the notation of Section 4.2.2,

we use R to refer to some optimal solution. Recall that we partition the input stream into (almost) equal-sized segments $S_i : 1 \leq i \leq k$, and pick one item from each. Then T_i denotes the set of items we have picked at the completion of segment i. We show that $f(T_k) \geq \frac{1}{2e} f(R \cup T_i)$ for some integer i, even when f is not monotone. Roughly speaking, the proof mainly follows from the submodularity property and Lemma 4.2.1.

Lemma 4.2.6. *If we run the monotone algorithm on a (possibly non-monotone) submodular function f, we obtain $f(T_k) \geq \frac{1}{2e^2} f(R \cup T_i)$ for some i.*

Proof. Consider the stage $i+1$ in which we want to pick an item from S_{i+1}. Lemma 4.2.1 implies

$$f(R \cup T_i) \leq f(T_i) + \sum_{a \in R \setminus T_i} f(T_i \cup \{a\}) - f(T_i).$$

At least one of the two right-hand side terms has to be larger than $f(R \cup T_i)/2$. If this happens to be the first term for any i, we are done: $f(T_k) \geq f(T_i) \geq \frac{1}{2} f(R \cup T_i)$ since $f(T_k) \geq f(T_i)$ by the definition of the algorithm: the first **if** statement makes sure $f(T_i)$ values are non-decreasing. Otherwise assume that the lower bound occurs for the second terms for all values of i.

Consider the events that among the elements in $R \setminus T_i$ exactly one, say a, falls in S_{i+1}. Call this event $\mathcal{E}_a$. Conditioned on $\mathcal{E}_a$, $\Delta_{i+1} := f(T_{i+1}) - f(T_i)$ is at least $f(T_i \cup \{a\}) - f(T_i)$ with probability $1/e$: i.e., if the algorithm picks the best secretary in this interval. Each event $\mathcal{E}_a$ occurs with probability at least $\frac{1}{k} \cdot \frac{1}{e}$. Since these events are disjoint, we have

$$\begin{aligned} \mathbf{E}[\Delta_{i+1}] &\geq \sum_{a \in R \setminus T_i} \Pr[\mathcal{E}_a] \cdot \frac{1}{e} [f(T_{i+1}) - f(T_i)] \geq \frac{1}{e^2 k} \sum_{a \in R \setminus T_i} f(T_i \cup \{a\}) - f(T_i) \\ &\geq \frac{1}{2e^2 k} f(R \cup T_i), \end{aligned}$$

and by summing over all values of i, we obtain

$$\mathbf{E}[f(T_k)] = \sum_i \mathbf{E}[\Delta_i] \geq \sum_i \frac{1}{2e^2 k} f(R \cup T_i) \geq \frac{1}{2e^2} \min_i f(R \cup T_i). \qquad \square$$

Unlike the case of monotone functions, we cannot say that $f(R \cup T_i) \geq f(R)$, and conclude that our algorithm is constant-competitive. Instead, we need to use other techniques to cover the cases that $f(R \cup T_i) < f(R)$. The following lemma presents an upper bound on the value of the optimum.

Lemma 4.2.7. *For any pair of disjoint sets Z and Z', and a submodular function f, we have $f(R) \leq f(R \cup Z) + f(R \cup Z')$.*

Proof. The statement follows from the submodularity property, observing that $(R \cup Z) \cap (R \cup Z') = R$, and $f([R \cup Z] \cup [R \cup Z']) \geq 0$. $\square$

We are now at a position to prove the performance guarantee of our main algorithm.

Theorem 4.2.8. *Algorithm 2 has competitive ratio $8e^2$.*

Proof. Let the outputs of the two algorithms be sets Z and Z', respectively. The expected value of the solution is thus $[f(Z) + f(Z')]/2$.

We know that $\mathbf{E}[f(Z)] \geq c' f(R \cup X_1)$ for some constant c', and $X_1 \subseteq U_1$. The only difference in the proof is that each element of $R \setminus Z$ appears in the set S_i with probability $1/2k$ instead of $1/k$. But we can still prove the above lemma for $c' := 1/4e^2$. Same holds for Z': $\mathbf{E}[f(Z')] \geq \frac{1}{4e} f(R \cup X_2)$ for some $X_2 \subseteq U_2$.

Since U_1 and U_2 are disjoint, so are X_1 and X_2. Hence, the expected value of our solution is at least $\frac{1}{4e^2}[f(R \cup X_1) + f(R \cup X_2)]/2$, which via Lemma 4.2.7 is at least $\frac{1}{8e^2} f(R)$. $\square$

4.3 The Submodular Matroid Secretary Problem

In this section, we prove Theorem 4.1.2. We first design an $O(\log^2 r)$-competitive algorithm for maximizing a monotone submodular function, when there are matroid

constraints for the set of selected items. Here we are allowed to choose a subset of items only if it is an independent set in the given matroid.

The matroid $(\mathcal{U}, \mathcal{I})$ is given by an oracle access to $\mathcal{I}$. Let n denote the number of items, i.e., $n := |\mathcal{U}|$, and r denotes the rank of the matroid. Let $S \in \mathcal{I}$ denote an optimal solution that maximizes the function f. We focus our analysis on a refined set $S^* \subseteq S$ that has certain nice properties: 1) $f(S^*) \geq (1 - 1/e)f(S)$, and 2) $f(T) \geq f(S^*)/\log r$ for any $T \subseteq S^*$ such that $|T| = \lfloor |S^*|/2 \rfloor$. We cannot necessarily find S^*, but we prove that such a set exists.

Start by letting $S^* = S$. As long as there is a set T violating the second property above, remove T from S^*, and continue. The second property clearly holds at the termination of the procedure. In order to prove the first property, consider one iteration. By submodularity (subadditivity to be more precise) we have $f(S^* \setminus T) \geq f(S^*) - f(T) \geq (1 - 1/\log r)f(S^*)$. Since each iteration halves the set S^*, there are at most $\log r$ iterations. Therefore, $f(S^*) \geq (1 - 1/\log r)^{\log r} \cdot f(S) \geq (1 - 1/e)f(S)$.

We analyze the algorithm assuming the parameter $|S^*|$ is given, and achieve a competitive ratio $O(\log r)$. If $|S^*|$ is unknown, though, we can guess its value (from a pool of $\log r$ different choices) and continue with Lemma 4.3.1. This gives an $O(\log^2 r)$ competitive ratio.

Lemma 4.3.1. *Given $|S^*|$, Algorithm 3 picks an independent subset of items with size $|S^*|/2$ whose expected value is at least $f(S^*)/4e \log r$.*

Proof. Let $k := |S^*|$. We divide the input stream of n items into k segments of (almost) equal size. We only pick $k/2$ items, one from each of the first $k/2$ segments.

Similarly to Algorithm 1 for the submodular secretary problem, when we work on each segment, we try to pick an item that maximizes the marginal value of the function given the previous selection is fixed (see the **for** loop in Algorithm 1). We show that the expected gain in each of the first $k/2$ segments is at least a constant fraction of $f(S^*)/k \log r$.

Suppose we are working on segment $i \leq k/2$, and let Z be the set of items already picked; so $|Z| \leq i - 1$. Furthermore, assume $f(Z) \leq f(S^*)/2 \log r$ since otherwise, the

Algorithm 3 Monotone Submodular Secretary Algorithm with Matroid constraint

Input: A monotone submodular function $f : 2^{\mathcal{U}} \mapsto \mathbb{R}$, a matroid $(\mathcal{U}, \mathcal{I})$, and a randomly permuted stream of secretaries, denoted by $(a_1, a_2, \ldots, a_n)$.

Output: A subset of secretaries that are independent according to $\mathcal{I}$.

Let $U_1 := \{a_1, a_2, \ldots, a_{\lfloor n/2 \rfloor}\}$
Pick the parameter $k := |S^*|$ uniformly at random from the pool $\{2^0, 2^1, 2^{\log r}\}$
if $k = O(\log r)$ **then**
  Select the best item of the U_1 and output the singleton
else {run Algorithm 1 on U_1 and respect the matroid}
  Let $T_0 \leftarrow \emptyset$
  Let $l \leftarrow \lfloor n/k \rfloor$
  for $i \leftarrow 1$ **to** k **do** {phase i}
    Let $u_i \leftarrow (i-1)l + l/e$
    Let $\alpha_i \leftarrow \max_{\substack{(i-1)l \le j < u_i \\ T_{i-1} \cup \{a_j\} \in \mathcal{I}}} f(T_{i-1} \cup \{a_j\})$
    if $\alpha_i < f(T_{i-1})$ **then**
      $\alpha_i \leftarrow f(T_{i-1})$
    end if
    Pick an index $p_i : u_i \le p_i < il$ such that $f(T_{i-1} \cup \{a_{p_i}\}) \ge \alpha_i$ and $T_{i-1} \cup \{a_{p_i}\} \in \mathcal{I}$
    if such an index p_i exists **then**
      Let $T_i \leftarrow T_{i-1} \cup \{a_{p_i}\}$
    else
      Let $T_i \leftarrow T_{i-1}$
    end if
  end for
  Output T_k as the solution
end if

lemma is already proved. By matroid properties we know there is a set $T \subseteq S^* \setminus Z$ of size $\lfloor k/2 \rfloor$ such that $T \cup Z \in \mathcal{I}$. The second property of S^* gives $f(T) \geq f(S^*)/\log r$.

From Lemma 4.2.1 and monotonicity of f, we obtain

$$\sum_{s \in T} [f(Z \cup \{s\}) - f(Z)] \geq f(T \cup Z) - f(Z) \geq f(T) - f(Z) \geq f(S^*)/2 \log r.$$

Note that each item in T appears in this segment with probability $2/k$ because we divided the input stream into $k/2$ equal segments. Since in each segment we pick the item giving the maximum marginal value with probability $1/e$, the expected gain in this segment is at least

$$\sum_{s \in T} \frac{1}{e} \cdot \frac{2}{k} \cdot [f(Z \cup \{s\}) - f(Z)] \geq f(S^*)/ek \log r.$$

We have this for each of the first $k/2$ segments, so the expected value of our solution is at least $f(S^*)/2e \log r$. □

Finally, it is straightforward (and hence the details are omitted) to combine the algorithm in this section with Algorithm 2 for the non-monotone submodular secretary problem, to obtain an $O(\log^2 r)$-competitive algorithm for the non-monotone submodular secretary problem subject to a matroid constraint.

Here we show the same algorithm works when there are $l \geq 1$ matroid constraints and achieves a competitive ratio of $O(l \log^2 r)$. We just need to respect all matroid constraints in Algorithm 3. This finishes the proof of Theorem 4.1.2.

Lemma 4.3.2. *Given $|S^*|$, Algorithm 3 picks an independent subset of items (i.e., independent with respect to all matroids) with expected value at least $f(S^*)/4el \log r$.*

Proof. The proof is similar to the proof of Lemma 4.3.1. We show that the expected gain in each of the first $k/2l$ segments is at least a constant fraction of $f(S^*)/k \log r$.

Suppose we are working on segment $i \leq k/2l$, and let Z be the set of items already picked; so $|Z| \leq i - 1$. Furthermore, assume $f(Z) \leq f(S^*)/2 \log r$ since otherwise, the lemma is already proved. We claim that there is a set $T \subseteq S^* \setminus Z$ of

size $k - l \times \lfloor k/2l \rfloor \geq k/2$ such that $T \cup Z$ is an independent set in all matroids. The proof is as follows. We know that there exists a set $T_1 \subseteq S^*$ whose union with Z is an independent set of the first matroid, and the size of T_1 is at least $|S^*| - |Z|$. This can be proved by the exchange property of matroids, i.e., adding Z to the independent set S^* does not remove more than $|Z|$ items from S^*. Since T_1 is independent with respect to the second matroid (as it is a subset of S^*), we can prove that there exists a set $T_2 \subseteq T_1$ of size at least $|T_1| - |Z|$ such that $Z \cup T_2$ is an independent set in the second matroid. If we continue this process for all matroid constraints, we can prove that there is a set T_l which is an independent set in all matroids, and has size at least $|S^*| - l|Z| \geq k - l \times \lfloor k/2l \rfloor \geq k/2$ such that $Z \cup T_l$ is independent with respect to all the given matroids. The rest of the proof is similar to the proof of Lemma 4.3.1—we just need to use the set T_l instead of the set T in the proof.

Since we are gaining a constant times $f(S^*)/k \log r$ in each of the first $k/2l$ segments, the expected value of the final solution is at least a constant times $f(S^*)/l \log r$. □

4.4 Knapsack Constraints

In this section, we prove Theorem 4.1.3. We first outline how to reduce an instance with multiple knapsacks to an instance with only one knapsack, and then we show how to solve the single knapsack instance.

Without loss of generality, we can assume that all knapsack capacities are equal to one. Let I be the given instance with the value function f, and item weights w_{ij} for $1 \leq i \leq l$ and $1 \leq j \leq n$. Define a new instance I' with one knapsack of capacity one in which the weight of the item j is $w'_j := \max_i w_{ij}$. We first prove that this reduction loses no more than a factor $4l$ in the total value. Take note that both the scaling and the weight transformation can be carried in an online manner as the items arrive. Hence, the results of this section hold for the online as well as the offline setting.

Lemma 4.4.1. *With instance I' defined above, we have* $\frac{1}{4l}\mathrm{OPT}(I) \leq \mathrm{OPT}(I') \leq \mathrm{OPT}(I)$.

Proof. The latter inequality is very simple: Take the optimal solution to I'. This is also feasible in I since all the item weights in I are bounded by the weight in I'.

We next prove the other inequality. Let T be the optimal solution of I. An item j is called *fat* if $w'_j \geq 1/2$. Notice that there can be at most $2l$ fat items in T since $\sum_{j \in T} w'_j \leq \sum_{j \in T} \sum_i w_{ij} \leq l$. If there is any fat item with value at least $\mathrm{OPT}(I)/4l$, the statement of the lemma follows immediately, so we assume this is not the case. The total value of the fat items, say F, is at most $\mathrm{OPT}(I)/2$. Submodularity and non-negativity of f gives $f(T \setminus F) \geq f(T) - f(F) \geq \mathrm{OPT}(I)/2$. Sort the non-fat items in decreasing order of their value density (i.e., ratio of value to weight), and let T' be a maximal prefix of this ordering that is feasible with respect to I'. If $T' = T \setminus F$, we are done; otherwise, T' has weight at least $1/2$. Let x be the total weight of items in T' and let y indicate the total weight of items in $T \setminus (F \cup T')$. Let α_x and α_y denote the densities of the two corresponding subsets of the items, respectively. Clearly $x + y \leq l$ and $\alpha_x \geq \alpha_y$. Thus, $f(T \setminus F) = \alpha_x \cdot x + \alpha_y \cdot y \leq \alpha_x(x + y) \leq \alpha_x \cdot l$. Now $f(T') \geq \alpha_x \cdot \frac{1}{2} \geq \frac{1}{2l} f(T \setminus F) \geq \frac{1}{4l} f(T)$ finishes the proof. □

Here we show how to achieve a constant competitive ratio when there is only one knapsack constraint. Let w_j denote the weight of item $j : 1 \leq j \leq n$, and assume without loss of generality that the capacity of the knapsack is 1. Moreover, let f be the value function which is a non-monotone submodular function. Let T be the optimal solution, and define $\mathrm{OPT} := f(T)$. The value of the parameter $\lambda \geq 1$ will be fixed below. Define T_1 and T_2 as the subsets of T that appear in the first and second half of the input stream, respectively. We first show the this solution is broken into two *balanced* portions.

Lemma 4.4.2. *If the value of each item is at most* OPT/λ*, for sufficiently large* λ*, the random variable* $|f(T_1) - f(T_2)|$ *is bounded by* $\mathrm{OPT}/2$ *with a constant probability.*

Proof. Each item of T goes to either T_1 or T_2 with probability $1/2$. Let the random variable X_j^1 denote the increase of the value of $f(T_1)$ due to the possible addition of item j. Similarly X_j^2 is defined for the same effect on $f(T_2)$. The two variables X_j^1 and X_j^2 have the same probability distribution, and because of submodularity

and the fact that the value of item j is at most OPT/λ, the contribution of item j in $f(T_1) - f(T_2)$ can be seen as a random variable that always take values in range $[-\text{OPT}/\lambda, \text{OPT}/\lambda]$ with mean zero. (In fact, we also use the fact that in an optimal solution, the marginal value of any item is non-negative. Submodularity guarantees that this holds with respect to any of the subsets of T as well.) Azuma's inequality ensures that with constant probability the value of $|f(T_1) - f(T_2)|$ is not more than $\max\{f(T_1), f(T_2)\}/2$ for sufficiently large λ. Since both $f(T_1)$ and $f(T_2)$ are at most OPT, we can say that they are both at least $\text{OPT}/4$, with constant probability. □

The algorithm is as follows. Without loss of generality assume that all items are feasible, i.e., any one item fits into the knapsack. We flip a coin, and if it turns up "heads," we simply try to pick the one item with the maximum value. We do the following if the coin turns up "tails." We do not pick any item from the first half of the stream. Instead, we compute the maximum value set in the first half with respect to the knapsack constraint; Lee et al. give a constant factor approximation for this task. From the above argument, we know that $f(T_1)$ is at least $OPT/4$ since all the items have limited value in this case (i.e., at most OPT/λ). Therefore, we obtain a constant factor estimation of OPT by looking at the first half of the stream: i.e., if the estimate is $\hat{\text{OPT}}$, we get $\text{OPT}/c \leq \hat{\text{OPT}} \leq \text{OPT}$. After obtaining this estimate, we go over the second half of the input, and pick an item j if and only if it is feasible to pick this item, and moreover, the ratio of its marginal value to w_j is at least $\hat{\text{OPT}}/6$.

Lemma 4.4.3. *The above algorithm is a constant competitive algorithm for the non-monotone submodular secretary problem with one knapsack constraint.*

Proof. We give the proof for the monotone case. Extending it for the non-monotone requires the same idea as was used in the proof of Theorem 2. First suppose there is an item with value at least OPT/λ. With probability 1/2, we try to pick the best item, and we succeed with probability $1/e$. Thus, we get an $O(1)$ competitive ratio in this case.

In the other case, all the items have small contributions to the solution, i.e., less than OPT/λ. In this case, with constant probability, both $f(T_1)$ and $f(T_2)$ are at

least OPT /4. Hence, $\hat{\text{OPT}}$ is a constant estimate for OPT. Let T' be the set of items picked by the algorithm in this case. If the sum of the weights of the items in T' is at least 1/2, we are done, because all these items have (marginal) value density at least $\hat{\text{OPT}}/6$, so $f(T') \geq (1/2) \cdot (\hat{\text{OPT}}/6) = \hat{\text{OPT}}/12 \geq \text{OPT}/48$.

Otherwise, the total weight of T' is less than 1/2. Therefore, there are items in T_2 that are not picked. There might be two reasons for this. There was not enough room in the knapsack, which means that the weight of the items in T_2 is more than 1/2. However, there cannot be more than one such item in T_2, and the value of this item is not more than OPT/λ. Let z be this single big item, for future reference. Therefore, $f(T') \geq f(T_2) - \text{OPT}/\lambda$ in this case.

The other case is when the ratios of some items from T_2 are less than $\hat{\text{OPT}}/6$, and thus we do not pick them. Since they are all in T_2, their total weight is at most 1. Because of submodularity, the total loss due to these missed items is at most $\hat{\text{OPT}}/6$. Submodularity and non-negativity of f then gives $f(T') \geq f(T_2) - f(\{z\}) - \hat{\text{OPT}}/6 \geq \hat{\text{OPT}} - \text{OPT}\,\lambda - \hat{\text{OPT}}/6 = O(\text{OPT})$. □

4.5 The Subadditive Secretary Problem

In this section, we prove Theorem 4.1.4 by presenting first a hardness result for approximation subadditive functions in general. The result applies in particular to our online setting. Surprisingly, the monotone subadditive function that we use here is *almost submodular*; see Proposition 4.5.4 below. Hence, our constant competitive ratio for submodular functions is nearly the most general we can achieve.

Definition 4.5.1 (Subadditive function maximization). *Given a nonnegative subadditive function f on a ground set U, and a positive integer $k \leq |U|$, the goal is to find a subset S of U of size at most k so as to maximize $f(S)$. The function f is accessible through a value oracle.*

4.5.1 Hardness Result

In the following discussion, we assume that there is an upper bound of m on the size of sets given to the oracle. We believe this restriction can be lifted. If the function f is not required to be monotone, this is quite easy to have: simply let the value of the function f be zero for queries of size larger than m. Furthermore, depending on how we define the online setting, this may not be an *additional* restriction here. For example, we may not be able to query the oracle with secretaries that have already been rejected.

The main result of the section is the following theorem. It shows the subadditive function maximization is difficult to approximate, even in the offline setting.

Theorem 4.5.2. *There is no polynomial time algorithm to approximate an instance of subadditive function maximization within $\tilde{O}(\sqrt{n})$ of the optimum. Furthermore, no algorithm with exponential time 2^t can achieve an approximation ratio better than $\tilde{O}(\sqrt{n/t})$.*

First, we are going to define our *hard* function. Afterwards, we continue with proving certain properties of the function which finally lead to the proof of Theorem 4.5.2.

Let n denote the size of the universe, i.e., $n := |U|$. Pick a random subset $S^* \subseteq U$ by sampling each element of U with probability k/n. Thus, the expected size of S^* is k.

Define the function $g : U \mapsto \mathbb{N}$ as $g(S) := |S \cap S^*|$ for any $S \subseteq U$. One can easily verify that g is submodular. We have a positive r whose value will be fixed below. Define the final function $f : U \mapsto \mathbb{N}$ as

$$f(S) := \begin{cases} 1 & \text{if } g(S) = 0 \\ \lceil g(S)/r \rceil & \text{otherwise.} \end{cases}$$

It is not difficult to verify the subadditivity of f; it is also clearly monotone.

In order to prove the core of the hardness result in Lemma 4.5.3, we now let

$r := \lambda \cdot \frac{mk}{n}$, where $\lambda \geq 1 + \sqrt{\frac{3tn}{mk}}$ and $t = \Omega(\log n)$ will be determined later.

Lemma 4.5.3. *An algorithm making at most 2^t queries to the value oracle cannot solve the subadditive maximization problem to within k/r approximation factor.*

Proof. Note that for any $X \subseteq U$, $f(X)$ lies between 0 and $\lceil k/r \rceil$. In fact, the optimal solution is the set S^* whose value is at least k/r. We prove that with high probability the answer to all the queries of the algorithm is one. This implies that the algorithm cannot achieve an approximation ratio better than k/r.

Assume that X_i is the i-th query of the algorithm for $1 \leq i \leq 2^t$. Notice that X_i can be a function of our answers to the previous queries. Define $\mathcal{E}_i$ as the event $f(X_i) = 1$. This is equivalent to $g(X_i) \leq r$. We show that with high probability all events $\mathcal{E}_i$ occur.

For any $1 \leq i \leq 2^t$, we have

$$\Pr\left[\mathcal{E}_i | \bigcap_{j=1}^{i-1} \mathcal{E}_j\right] = \frac{\Pr[\bigcap_{j=1}^{i} \mathcal{E}_j]}{\Pr[\bigcap_{j=1}^{i-1} \mathcal{E}_j]} \geq \Pr\left[\bigcap_{j=1}^{i} \mathcal{E}_j\right] \geq 1 - \sum_{j=1}^{i} \overline{\mathcal{E}_j}.$$

Thus, we have $\Pr[\cap_{i=1}^{2^t} \mathcal{E}_i] \geq 1 - 2^t \sum_{i=1}^{2^t} \Pr[\overline{\mathcal{E}_i}]$ from union bound. Next we bound $\Pr[\mathcal{E}_i]$. Consider a subset $X \subseteq U$ such that $|X| \leq m$. Since the elements of S^* are picked randomly with probability k/n, the expected value of $X \cap S^*$ is at most mk/n. Standard application of Chernoff bounds gives

$$\begin{aligned}\Pr[f(X) \neq 1] = \Pr[g(X) > r] = \Pr\left[|X \cap S^*| > \lambda \cdot \frac{mk}{n}\right] &\leq \exp\left\{-(\lambda - 1)^2 \frac{mk}{n}\right\} \\ &\leq \exp\{-3t\} \leq \frac{2^{-2t}}{n},\end{aligned}$$

where the last inequality follows from $t \geq \log n$. Therefore, the probability of all $\mathcal{E}_i$ events occurring simultaneously is at least $1 - 1/n$. □

Now we can prove the main theorem of the section.

Proof of Theorem 4.5.2. We just need to set $k = m = \sqrt{n}$. Then, $\lambda = \sqrt{3t}$, and the inapproximability ratio is $\Omega(\sqrt{\frac{n}{t}})$. Restricting to polynomial algorithms, we obtain

$t := O(\log^{1+\varepsilon} n)$, and considering exponential algorithms with running time $O(2^{t'})$, we have $t = O(t')$, giving the desired results. □

In case the query size is not bounded, we can define $f(X) := 0$ for large sets X, and pull through the same result; however, the function f is no longer monotone in this case.

We now show that the function f is almost submodular. Recall that a function g is submodular if and only if $g(A) + g(B) \geq g(A \cup B) + g(A \cap B)$.

Proposition 4.5.4. *For the hard function f defined above, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B) - 2$ always holds; moreover, $f(X)$ is always positive and attains a maximum value of $\tilde{\Theta}(\sqrt{n})$ for the parameters fixed in the proof of Theorem 4.5.2.*

Proof. The function $h(X) := g(X)/r$ is clearly submodular, and we have $h(X) \leq f(X) \leq h(X) + 1$. We obtain $f(A) + f(B) \geq h(A) + h(B) \geq h(A \cup B) + h(A \cap B) \geq f(A \cup B) + f(A \cap B) - 2$. □

4.5.2 Algorithm

An algorithm that only picks the best item clearly gives a k competitive ratio. We now show how to achieve an $O(n/k)$ competitive ratio, and thus by combining the two, we obtain an $O(\sqrt{n})$-competitive algorithm for the monotone subadditive secretary problem. This result complements our negative result nicely.

Partition the input stream S into $\ell := n/k$ (almost) equal-sized segments, each of size at most k. Randomly pick all the elements in one of these segments. Let the segments be denoted by $S_1, S_2, \ldots, S_\ell$. Subadditivity of f implies $f(S) \leq \sum_i f(S_i)$. Hence, the expected value of our solution is $\sum_i \frac{1}{\ell} f(S_i) \geq \frac{1}{\ell} f(S) \geq \frac{1}{\ell} \mathrm{OPT}$, where the two inequalities follow from subadditivity and monotonicity, respectively.

4.6 Further Results

In this chapter, we consider the (non-monotone) submodular secretary problem for which we give a constant-competitive algorithm. The result can be generalized when

we have a matroid constraint on the set that we pick; in this case we obtain an $O(\log^2 r)$-competitive algorithm where r is the rank of the matroid. However, we show that it is very hard to compete with the optimum if we consider subadditive functions instead of submodular functions. This hardness holds even for "almost submodular" functions; see Proposition 4.5.4.

One may consider special non-submodular functions which enjoy certain structural results in order to find better guarantees. For example, let $f(T)$ be the minimum individual value in T which models a bottle-neck situation in the secretary problem, i.e., selecting a group of k secretaries to work together, and the speed (efficiency) of the group is limited to that of the slowest person in the group (note that unlike the submodular case here the condition for employing exactly k secretaries is enforced.) In this case, we present a simple $O(k)$ competitive ratio for the problem as follows. Interview the first $1/k$ fraction of the secretaries without employing anyone. Let α be the highest efficiency among those interviewed. Employ the first k secretaries whose efficiency surpasses α.

Theorem 4.6.1. *Following the prescribed approach, we employ the k best secretaries with probability at least $1/e^2k$.*

Indeed we believe that this $O(k)$ competitive ratio for this case should be almost tight. One can verify that provided individual secretary efficiencies are far from each other, say each two consecutive values are farther than a multiplicative factor n, the problem of maximizing the expected value of the minimum efficiency is no easier than being required to employ all the k best secretaries. The following theorem provides evidence that the latter problem is hard to approximate.

Theorem 4.6.2. *Any algorithm with a single threshold—i.e., interviewing applicants until some point (observation phase), and then employing any one who is better than all those in the observation phase—misses one of the k best secretaries with probability $1 - O(\log k/k)$.*

Another important aggregation function f is that of maximizing the performance of the secretaries we employ: think of picking k candidate secretaries and finally

hiring the best. We consider this function in Subsection 4.6.1 for which we present a near-optimal solution. In fact, the problem has been already studied, and an optimal strategy appears in [40]. However, we propose a simpler solution which features certain "robustness" properties (and thus is of its own interest): in particular, suppose we are given a vector $(\gamma_1, \gamma_2, \ldots, \gamma_k)$ such that $\gamma_i \geq \gamma_{i+1}$ for $1 \leq i < k$. Sort the elements in a set R of size k in a non-increasing order, say $a_1, a_2, \ldots, a_k$. The goal is to maximize the efficiency $\sum_i \gamma_i a_i$. The algorithm that we propose maximizes this more general objective obliviously; i.e., the algorithm runs irrespective of the vector γ, however, it can be shown the resulting solution approximates the objective for all vectors γ at the same time. We refer to Subsection 4.6.1 for more details. Following we provide proofs of theorems 4.6.1 and, 4.6.2.

Proof of Theorem 4.6.1. Let $R = \{a_1, a_2, \ldots, a_{|R|}\} \subseteq S$ denote the set of k best secretaries. Let S^* denote the first $1/k$ fraction of the stream of secretaries. Let $\mathcal{E}^1$ denote the event when $S^* \cap R = \emptyset$, that is, we do not lose the chance of employing the best secretaries (R) by being a mere observer in S^*. Let $\mathcal{E}^2$ denote the event that we finally pick the set R. Let us first bound $\Pr[\mathcal{E}^1]$. In order to do so, define $\mathcal{E}^1_j$ for $j : 1 \leq j \leq |R|$ as the event that $a_j \notin S_e$. We know that $\Pr[\mathcal{E}^1_1] \geq 1/k$. In general, we have for $j > 1$

$$\begin{aligned}
\Pr\left[\mathcal{E}^1_j \middle| \bigcap_{i<j} \mathcal{E}^1_i\right] &\geq \frac{n - \frac{n}{k} - j + 1}{n - j + 1} \\
&\geq \frac{n - \frac{n}{k} - k}{n - k} \\
&= 1 - \frac{n/k}{n - k} \\
&\geq 1 - \frac{2}{k} \qquad \text{assuming } k \leq \frac{n}{2}. \qquad (4.4)
\end{aligned}$$

Notice that the final assumption is justified because we can solve the problem of finding the $k' = n - k \leq n/2$ smallest numbers in case $k > n/2$. Using Equation (4.4)

we obtain

$$\begin{aligned}\Pr[\mathcal{E}^1] &= \Pr[\mathcal{E}_1^1]\Pr[\mathcal{E}_2^1|\mathcal{E}_1^1]\cdots\Pr[\mathcal{E}_{|R|}^1|\cup_{j<|R|}\mathcal{E}_j^1] \\ &\geq \left(1-\frac{2}{k}\right)^k \\ &\geq e^{-2}. \end{aligned} \tag{4.5}$$

The event $\mathcal{E}^2$ happens when $\mathcal{E}^1$ happens and the $(k+1)$th largest element appears in S^*. Thus, we have $\Pr[\mathcal{E}^2] = \Pr[\mathcal{E}^1]\Pr[\mathcal{E}^2|\mathcal{E}^1] \geq e^{-2}\cdot 1/k = \frac{1}{e^2 k}$. □

Proof of Theorem 4.6.2. We assume that we cannot find the actual efficiency of a secretary, but we only have an oracle that given two secretaries already interviewed, reports the better of the two. This model is justified if the range of efficiency values is large, and a suitable perturbation is introduced into the values.

Suppose the first secretary is hired after interviewing a β fraction of the secretaries. If $\beta > \log k/k$ then the probability that we miss at least one of the k best secretaries is at least $1-(1-\beta)^k = 1-1/k$. If on the other hand, β is small, say $\beta \leq \log k/k$, there is little chance that the right threshold can be picked. Notice that in the oracle model, the threshold has to be the efficiency of one prior secretary. Thus for the right threshold to be selected, we need to have the $(k+1)$th best secretary in the first β fraction—the probability of this even is no more than β. Therefore, the probability of success cannot be more than $\log k/k$. □

4.6.1 The Secretary Problem with the "Maximum" Function

We now turn to consider a different efficiency aggregation function, namely the maximum of the efficiency of the individual secretaries. Alternately, one can think of this function as a secretary function *with k choices*, that is, we select k secretaries and we are satisfied as long as one of them is the best secretary interviewed. We propose an algorithm that accomplishes this task with probability $1-O\left(\frac{\ln k}{k}\right)$ for $k>1$.

As we did before, we assume that n is a multiple of k, and we partition the input stream into k equally-sized segments, named $S_1, S_2, \ldots, S_k$. Let $f(s)$ denote

the efficiency of the secretary $s \in S$. For each set $i : 1 \leq i < k$, we compute

$$\alpha_i := \max_{s \in \bigcup_{j \leq i} S_i} f(s),$$

which is the efficiency of the best secretary in the first i segments. Clearly, α_i can be computed in an online manner after interviewing the first i segments. For each $i : 1 \leq i < k$, we try to employ the first secretary in $\bigcup_{j>i} S_j$ whose efficiency surpasses α_i. Let this choice, if at all present, be denoted s_i. The output of the algorithm consists of all such secretaries $\{s_i\}_i$. Notice that such an element may not exist for a particular i, or we may have $s_i = s_{i'}$ for $i \neq i'$. We employ at most $k - 1$ secretaries. The following theorem bounds the failure probability of the algorithm.

Theorem 4.6.3. *The probability of not employing the best secretary is* $O\left(\frac{\ln k}{k}\right)$.

Proof. Let $(a_1, a_2, \ldots, a_n)$ denote the stream of interviewed secretaries. Let a_m be the best secretary, and suppose $a_m \in S_i$, namely $(i-1)l < m \leq il$, where $l := n/k$. Our algorithm is successful if the second best secretary of the set $\{a_1, a_2, \ldots, a_{m-1}\}$ does not belong to S_i. The probability of this event is

$$\frac{(i-1)l}{m} \geq \frac{(i-1)l}{il} = \frac{i-1}{i}. \tag{4.6}$$

The probability of $a_m \in S_i$ is $1/k$ and conditioned on this event, the probability of failure is at most $1/i$. Hence, the total failure probability is no more than $\sum_{i=1}^{k} \frac{1}{k}\frac{1}{i} = O\left(\frac{\ln k}{k}\right)$ as claimed. □

This problem has been previously studied by Gilbert and Mosteller [40]. Our algorithm above is simpler and yet "robust" in the following sense. The primary goal is to select the best secretary, but we also guarantee that many of the "good" secretaries are also selected. In particular, we show that the better the rank of a secretary is in our evaluation, the higher is the guarantee we have for employing her.

Theorem 4.6.4. *The probability of not hiring a secretary of rank* y *is* $O\left(\sqrt{\frac{y}{k}}\right)$.

Proof. Let $(a_1, a_2, \ldots, a_n)$ denote the stream of interviewed secretaries. Let a_m be the secretary of rank y, and suppose $a_m \in S_i$, namely $(i-1)l < m \leq il$, where $l := n/k$. Below we define three bad events whose probabilities we bound, and we show that a_m is hired provided none of these events occur. In particular, we give an upper bound of $O(\sqrt{y/k}$ for each event. The claim then follows from the union bound.

Let $z := \sqrt{\frac{k}{y-1}} - 1$. The event $\mathcal{E}_1$ occurs if $i \leq z$. This event happens with probability z/k which is less than $\sqrt{\frac{1}{k(y-1)}} \leq \sqrt{\frac{y}{k}}$.

We say the event $\mathcal{E}_2$ happens if a_m is not the best secretary among those in sets S_i, S_{i-1}, ..., S_{i-z}. This happens when there is at least one of the $y-1$ secretaries better than a_m in these sets. Let W be a random variable denoting the number of these $y-1$ secretaries in any of the mentioned sets. Since any secretary is in one of these sets with probability $(z+1)/k$ (notice that $z+1$ is the number of these sets), we can say that the expected value of W is $(y-1)(z+1)/k$. Using the Markov Inequality, the probability that W is at least 1 is at most its expected value which is $(y-1)(z+1)/k$. Thus, using the definition of z, we get an upper bound of $O\left(\sqrt{\frac{y-1}{k}}\right)$ for $\mathcal{E}_2$.

Finally, we define $\mathcal{E}_3$ as the event when the best secretary among $\{a_{(i-z-1)l+1}, a_{(i-z-1)l+2}, \ldots, a_{j-1}\}$ (secretaries appearing before a_m in the above-mentioned sets) is in set S_i. This happens with probability at most $1/(z+1)$, because there are $z+1$ sets that the best secretary is equally likely to be in each. Thus, we get $\Pr[\mathcal{E}_3] = O\left(\sqrt{\frac{y}{k}}\right)$ by definition of z.

If non of the events $\mathcal{E}_1, \mathcal{E}_2$ and $\mathcal{E}_3$ happen, we claim a_m is employed. Because if the maximum item of items $\{a_{(i-z-1)l+1}, a_{(i-z-1)l+2}, \ldots, a_{j-1}\}$ is in the set $S_{i'}$, and $i - z \leq i' < i$, then we hire a_m for the set $S_{i'}$; refer to the algorithm when we consider the threshold $\alpha_{i'}$. □

The aforementioned algorithm of [40] misses a *good* secretary of rank y with probability roughly $1/y$. On the other hand, one can show that the algorithm of Kleinberg [59] (for maximizing the sum of the efficiencies of the secretaries) picks secretaries of high rank with probability about $1 - \Theta(1/\sqrt{k})$. However, the latter algorithm guarantees the selection of the best secretary with a probability no more than $O(1/\sqrt{k})$.

Therefore, our algorithm has the nice features of both these algorithms: the best secretary is hired with a very good probability, while other good secretaries also have a good chance of being employed.

Bibliography

[1] A. A. Ageev and M. I. Sviridenko. An 0.828-approximation algorithm for the uncapacitated facility location problem. *Discrete Appl. Math.*, 93(2-3):149–156, 1999.

[2] S. Agrawal, Z. Wang, and Y. Ye. A dynamic near-optimal algorithm for online linear programming. Working paper posted at http://www.stanford.edu/ yyye/.

[3] Miklos Ajtai, Nimrod Megiddo, and Orli Waarts. Improved algorithms and analysis for secretary problems and generalizations. *SIAM J. Discrete Math.*, 14(1):1–27, 2001.

[4] David Applegate and Edith Cohen. Making routing robust to changing traffic demands: algorithms and evaluation. *IEEE/ACM Trans. Netw.*, 16(6):1193–1206, 2006.

[5] Arash Asadpour, Hamid Nazerzadeh, and Amin Saberi. Stochastic submodular maximization. In *WINE*, pages 477–489, 2008.

[6] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Räcke. Optimal oblivious routing in polynomial time. *J. Comput. Syst. Sci.*, 69(3):383–394, 2004.

[7] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In *APPROX*, pages 16–28, 2007.

[8] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exch.*, 7(2):1–11, 2008.

[9] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA*, pages 434–443, 2007.

[10] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *ESA*, pages 170–181, 2010.

[11] MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. *ACM Transactions on Algorithms*, 9(4):32, 2013.

[12] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Morteza Zadimoghaddam. The submodular secretary problem. Technical Report TD-7UEP26, AT&T Labs–Research, July 2009.

[13] Aharon Ben-Tal and Arkadi Nemirovski. Robust optimization - methodology and applications. *Math. Program.*, 92(3):453–480, 2002.

[14] Dimitris Bertsimas, Dessislava Pachamanova, and Melvyn Sim. Robust linear optimization under general norms. *Oper. Res. Lett.*, 32(6):510–516, 2004.

[15] Anand Bhalgat, Jon Feldman, and Vahab S. Mirrokni. Online ad allocation with smooth delivery. In *ACM Conference on Knowledge Discovery (KDD)*, 2012.

[16] V. Bilò, M. Flammini, and L. Moscardelli. Pareto approximations for the bicriteria scheduling problem. *Journal of Parallel and Distributed Computing*, 66(3):393–402, 2006.

[17] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *SIGACT News*, 39(1):80–87, 2008.

[18] N. Buchbinder, K. Jain, and J.S. Naor. Online Primal-Dual Algorithms for Maximizing Ad-Auctions Revenue. In *Proc. ESA*, page 253. Springer, 2007.

[19] N. Buchbinder and J.S. Naor. Fair online load balancing. In *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 291–298. ACM, 2006.

[20] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *IPCO*, pages 182–196, 2007.

[21] Gerard Cornuejols, Marshall Fisher, and George L. Nemhauser. On the uncapacitated location problem. In *Studies in integer programming (Proc. Workshop, Bonn. 1975)*, pages 163–177. Ann. of Discrete Math., Vol. 1. North-Holland, Amsterdam, 1977.

[22] Gerard Cornuejols, Marshall L. Fisher, and George L. Nemhauser. Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Manage. Sci.*, 23(8):789–810, 1976/77.

[23] Nikhil Devanur and Thomas Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *ACM EC*, 2009.

[24] Nikhil R. Devanur, Zhiyi Huang, Nitish Korula, Vahab S. Mirrokni, and Qiqi Yan. Whole-page optimization and submodular welfare maximization with online bidders. In *ACM Conference on Electronic Commerce*, pages 305–322, 2013.

[25] Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *ACM Conference on Electronic Commerce*, pages 29–38, 2011.

[26] Nikhil R. Devanur, Balasubramanian Sivan, and Yossi Azar. Asymptotically optimal algorithm for stochastic adwords. In *ACM ee on Electronic Commerce*, pages 388–404, 2012.

[27] E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Sov. Math. Dokl.*, 4:627–629, 1963.

[28] Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Structures and their Applications (Proc. Calgary Internat. Conf., Calgary, Alta., 1969)*, pages 69–87. Gordon and Breach, New York, 1970.

[29] U. Feige and M. X. Goemans. Approximating the value of two power proof systems, with applications to MAX 2SAT and MAX DICUT. In *ISTCS*, page 182, 1995.

[30] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[31] Uriel Feige. On maximizing welfare when utility functions are subadditive. In *STOC*, pages 41–50, 2006.

[32] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. In *FOCS*, pages 461–471, 2007.

[33] J. Feldman, N. Korula, V. Mirrokni, S. Muthukrishnan, and M. Pal. Online ad assignment with free disposal. In *WINE*, 2009.

[34] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. Online stochastic packing applied to display ad allocation. In Mark de Berg and Ulrich Meyer, editors, *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part I*, volume 6346 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2010.

[35] Jon Feldman, Nitish Korula, Vahab S. Mirrokni, S. Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In Stefano Leonardi, editor, *Internet and Network Economics, 5th International Workshop, WINE 2009, Rome, Italy, December 14-18, 2009. Proceedings*, volume 5929 of *Lecture Notes in Computer Science*, pages 374–385. Springer, 2009.

[36] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1 - 1/e. In *FOCS*, 2009.

[37] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions. II. *Math. Prog. Stud.*, (8):73–87, 1978. Polyhedral combinatorics.

[38] M. Flammini and G. Nicosia. On multicriteria online problems. In *Proceedings of the 8th Annual European Symposium on Algorithms*, pages 191–201. Springer-Verlag, 2000.

[39] P. R. Freeman. The secretary problem and its extensions: a review. *Internat. Statist. Rev.*, 51(2):189–206, 1983.

[40] John P. Gilbert and Frederick Mosteller. Recognizing the maximum of a sequence. *J. Amer. Statist. Assoc.*, 61:35–73, 1966.

[41] Kenneth S. Glasser, Richard Holzsager, and Austin Barron. The d choice secretary problem. *Comm. Statist. C—Sequential Anal.*, 2(3):177–199, 1983.

[42] A. Goel, A. Meyerson, and S. Plotkin. Approximate majorization and fair online load balancing. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 384–390. Society for Industrial and Applied Mathematics, 2001.

[43] A. Goel, A. Meyerson, and S. Plotkin. Combining fairness with throughput: Online routing with multiple objectives. *Journal of Computer and System Sciences*, 63(1):62–79, 2001.

[44] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995.

[45] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *WINE*, pages 246–257, 2010.

[46] B. Haeupler, V. Mirrokni, and M. Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *WINE*, 2011.

[47] Mohammad T. Hajiaghayi, Robert Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *AAAI*, pages 58–65, 2007.

[48] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes. Adaptive limited-supply online auctions. In *EC*, pages 71–80, 2004.

[49] Eran Halperin and Uri Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *SODA*, pages 1–7, 2001.

[50] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[51] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-set packing. *Computational Complexity*, 15(1):20–39, 2006.

[52] Nicole Immorlica, Robert D. Kleinberg, and Mohammad Mahdian. Secretary problems with competing employers. In *WINE*, pages 389–400, 2006.

[53] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 48(4):761–777, 2001.

[54] Bala Kalyanasundaram and Kirk Pruhs. On-line network optimization problems. In *Developments from a June 1996 seminar on Online algorithms: the state of the art*, pages 268–280, London, UK, 1998. Springer-Verlag.

[55] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *STOC*, 2011.

[56] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358, 1990.

[57] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? In *FOCS*, pages 146–154, 2004.

[58] Samir Khuller, Anna Moss, and Joseph Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45, 1999.

[59] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA*, pages 630–631, 2005.

[60] Nitish Korula, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Bicriteria online matching: Maximizing weight and cardinality. In *to appear in Proceedings of the Ninth Conference on Web and Internet Economics*, WINE '13, 2013.

[61] Jon Lee, Vahab Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Maximizing non-monotone submodular functions under matroid and knapsack constraints. In *STOC*, pages 323–332, 2009.

[62] L. Lovász. Submodular functions and convexity. In *Mathematical programming: the state of the art (Bonn, 1982)*, pages 235–257. Springer, Berlin, 1983.

[63] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *ACM Conference on Electronic Commerce*, pages 288–294, 2007.

[64] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: A strongly factor revealing lp approach. In *STOC*, 2011.

[65] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007.

[66] H. Menshadi, S. OveisGharan, and A. Saberi. Offline optimization for online stochastic matching. In *SODA*, 2011.

[67] Vahab S. Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *SODA*, pages 1690–1701, 2012.

[68] Rajeev Motwani, Rina Panigrahy, and Ying Xu 0002. Fractional matching via balls-and-bins. In *APPROX-RANDOM*, pages 487–498, 2006.

[69] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. I. *Math. Program.*, 14(3):265–294, 1978.

[70] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *Siam Journal on Computing*, 26:350–368, 1997.

[71] Maurice Queyranne. A combinatorial algorithm for minimizing symmetric submodular functions. In *SODA*, pages 98–101, 1995.

[72] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Combin. Theory Ser. B*, 80(2):346–355, 2000.

[73] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004.

[74] Bo Tan and R. Srikant. Online advertisement, optimization and stochastic networks. *CoRR*, abs/1009.0870, 2010.

[75] R. J. Vanderbei. The optimal choice of a subset of a population. *Math. Oper. Res.*, 5(4):481–486, 1980.

[76] Erik Vee, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. Optimal online assignment with forecasts. In *ACM EC*, 2010.

[77] Jan Vondrák. Symmetry and approximability of submodular maximization problems. In *FOCS*, 2009.

[78] C.M. Wang, X.W. Huang, and C.C. Hsu. Bi-objective optimization: An online algorithm for job assignment. *Advances in Grid and Pervasive Computing*, pages 223–234, 2009.

[79] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. Cope: traffic engineering in dynamic networks. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 99–110, New York, NY, USA, 2006. ACM.

[80] John G. Wilson. Optimal choice and assignment of the best m of n randomly arriving items. *Stochastic Process. Appl.*, 39(2):325–343, 1991.