ABSTRACT

Title of dissertation: ALLOCATIONS IN LARGE MARKETS

Hossein Esfandiari, Doctor of Philosophy, 2017

Dissertation directed by: Professor MohammadTaghi Hajiaghayi
Department of Computer Science

Rapid growth and popularity of internet based services such as online markets and online advertisement systems provide a lot of new algorithmic challenges. One of the main challenges is the limited access to the input. There are two main reasons that algorithms have limited data accessibility.

- The input is extremely large, and hence having access to the whole data at once is not practical.
- The nature of the system forces us to make decisions before observing the whole input.

Internet-enabled marketplaces such as Amazon and eBay deal with huge datasets registering transaction of merchandises between lots of buyers and sellers. It is important that algorithms become more time and space efficient as the size of datasets increase. An algorithm that runs in polynomial time may not have a reasonable running time for such large datasets. In the first part of this dissertation, we study the development of allocation algorithms that are appropriate for use with massive datasets. We especially focus on the streaming setting which is a common model for big data analysis. In the graph streaming, the algorithm has access to a sequence of edges, called a stream. The algorithm reads edges in the order in which they appear in the stream. The goal is to design an algorithm that maintains a large allocation, using as little space as possible. We achieve our results by developing powerful sampling techniques. Indeed, one can implement our sampling techniques in the streaming setting as well as other distributed settings such as MapReduce.

Giant online advertisement markets such as Google, Bing and Facebook raised up several interesting allocation problems. Usually, in these applications we need to make the decision before obtaining the full information of the input graph. This enforces an uncertainty on our belief about the input, and thus makes the classical algorithms inapplicable. To address this shortcoming online algorithms have been developed. In online algorithms again the input is a sequence of items. Here the algorithm needs to make an irrevocable decision upon arrival of each item. In the second part of this dissertation, we aim to achieve two main goals for each allocation problem in the market. Our first goal is to design models to capture the uncertainty of the input based on the properties of problems and the accessible data in real applications. Our second goal is to design algorithms and develop new techniques for these market allocation problems.

ALLOCATIONS IN LARGE MARKETS

by

Hossein Esfandiari

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2017

Dedicated to my parents for their love and support.

Acknowledgments

I would like to thank my advisor Prof. MohammadTaghi Hajiaghayi for his time and support. He helped me to work on several interesting projects, and taught me how to start, continue working, and wrap up a challenging project. It was a great pleasure to work with him. I would also like to thank my internship mentors at Google, Dr. Nitish Korula, Dr. MohammadHossein Bateni, and Dr. Vahab Mirrokni. I have learned very useful techniques and ideas from them. Many thanks to Prof. Lawrence M. Ausubel, Prof. William Gasarch, Prof. David Mount, and, Prof. VS Subrahmanian for serving on my dissertation committee. Thanks to my family for their love and support.

Table of Contents

List of Figures

Chapter 1: Introduction

Rapid growth and popularity of internet based services such as online markets and online advertisement systems provide a lot of new algorithmic challenges. One of the main challenges is the limited access to the input. There are two main reasons that algorithms have limited data accessibility.

- The input is extremely large, and hence having access to the whole data at once is not practical.
- The nature of the system forces us to make decisions before observing the whole input.

Internet-enabled marketplaces such as Amazon and eBay deal with huge datasets registering transaction of merchandises between lots of buyers and sellers. It is important that algorithms become more time and space efficient as the size of datasets increase. An algorithm that runs in polynomial time may not have a reasonable running time for such large datasets. In the first part of this dissertation, we study the development of allocation algorithms that are appropriate for use with massive datasets. We especially focus on the streaming setting which is a common model for big data analysis. In the graph streaming, the algorithm has access to a sequence of edges, called a stream. The algorithm reads edges in the order in which they appear in the stream. The goal is to design an algorithm that maintains a large allocation, using as little space as possible. We achieve our results by developing powerful sampling techniques. Indeed, one can implement our sampling techniques in the streaming setting as well as other distributed settings such as MapReduce. In Section 1.1 we briefly mention our results in the streaming setting.

Giant online advertisement markets such as Google, Bing and Facebook raised up several interesting allocation problems. Usually, in these applications we need to make the decision before obtaining the full information of the input graph. This enforces an uncertainty on our belief about the input, and thus makes the classical algorithms inapplicable. To address this shortcoming online algorithms have been developed. In online algorithms again the input is a sequence of items. Here the algorithm needs to make an irrevocable decision upon arrival of each item. In the second part of this dissertation, we aim to achieve two main goals for each allocation problem in the market. Our first goal is to design models to capture the uncertainty of the input based on the properties of problems and the accessible data in real applications. Our second goal is to design algorithms and develop new techniques for these market allocation problems. In Section 1.2 we briefly mention our results on online allocations.

1.1 Streaming Algorithms

1.1.1 Matching in Streaming Setting

In Chapter 2, we consider the problem of estimating the size of a maximum matching when the edges are revealed in a streaming fashion. When the input graph is planar, we present a simple and elegant streaming algorithm that with high probability estimates the size of a maximum matching within a constant factor using $\tilde{\mathcal{O}}(n^{2/3})$ space, where n is the number of vertices. The approach generalizes to the family of graphs that have bounded arboricity, which include graphs with an excluded constant-size minor. To the best of our knowledge, this is the first result for estimating the size of a maximum matching in the *adversarial-order* streaming model (as opposed to the random-order streaming model) in $o(n)$ space. We circumvent the barriers inherent in the adversarial-order model by exploiting several structural properties of planar graphs, and more generally, graphs with bounded arboricity. We further reduce the required memory size to $\tilde{\mathcal{O}}(\sqrt{n})$ for three restricted settings: (i) when the input graph is a forest; (ii) when we have 2-passes and the input graph has bounded arboricity; and (iii) when the edges arrive in random order and the input graph has bounded arboricity.

Finally, we design a reduction from the Boolean Hidden Matching Problem to show that there is no randomized streaming algorithm that estimates the size of the maximum matching to within a factor better than $3/2$ and uses only $o(n^{1/2})$ bits of space. Using the same reduction, we show that there is no deterministic algorithm that computes this kind of estimate in $o(n)$ bits of space. The lower bounds hold even for graphs that are collections of paths of constant length.

A paper containing these results appeared in the proceeding of SODA 2015.

1.1.2 Matching in Dynamic Streams

In Chapter 3 we present a simple but powerful subgraph sampling primitive that is applicable in a variety of computational models including dynamic graph streams (where the input graph is defined by a sequence of edge/hyperedge insertions and deletions) and distributed systems such as MapReduce. In the case of dynamic graph streams, we use this primitive to prove the following results:

- *Matching:* Our main result for matchings is that there exists an $\tilde{O}(k^2)$ space algorithm that returns the edges of a maximum matching on the assumption the cardinality is at most k. The best previous algorithm used $\tilde{O}(kn)$ space where n is the number of vertices in the graph and we prove our result is optimal up to logarithmic factors. Our algorithm has $\tilde{O}(1)$ update time. We also show that there exists an $\tilde{O}(n^2/\alpha^3)$ space algorithm that returns an α-approximation for matchings of arbitrary size. We generalize our exact and approximate algorithms to weighted matching. While there has been a substantial amount of work on approximate matching in insert-only graph streams, these are the first non-trivial results in the dynamic setting.

- *Vertex Cover and Hitting Set:* There exists an $\tilde{O}(k^d)$ space algorithm that solves the minimum hitting set problem where d is the cardinality of the input sets and k is an upper bound on the size of the minimum hitting set. We prove this is optimal up to logarithmic factors. Our algorithm has $\tilde{O}(1)$ update time. The case $d = 2$ corresponds to minimum vertex cover.

Finally, we consider a larger family of parameterized problems (including b-matching, disjoint paths, vertex coloring among others) for which our subgraph sampling primitive yields fast, small-space dynamic graph stream algorithms. We then show lower bounds for natural problems outside this family.

A paper containing these results appeared in the proceeding of SODA 2016.

1.1.3 Market Pricing over Streams

Internet-enabled marketplaces such as Amazon deal with huge datasets registering transaction of merchandises between lots of buyers and sellers. It is important that algorithms become more time and space efficient as the size of datasets increase. An algorithm that runs in polynomial time may not have a reasonable running time for such large datasets. Here, we study the development of pricing algorithms that are appropriate for use with massive datasets. We especially focus on the streaming setting, the common model for big data analysis.

In Chapter 4 we present an envy-free mechanism for social welfare maximization problem in the streaming setting using $O(k^2 l)$ space, where k is the number of different goods and l is the number of available items of each good. We also provide an α-approximation mechanism for revenue maximization in this setting given an α-approximation mechanism for the corresponding offline problem exists.

Moreover, we provide mechanisms to approximate the optimum social welfare (or revenue) within $1-\epsilon$ factor, in space independent of l which would be favorable in case l is large compared to k. Finally, we present hardness results showing approximation of *optimal prices* that maximize social welfare (or revenue) in the streaming setting needs $\Omega(l)$ space.

We achieve our results by developing a powerful sampling technique for bipartite networks. The simplicity of our sampling technique empowers us to maintain the sample over the input sequence. Indeed, one can construct this sample in the distributed setting (a.k.a, MapReduce) and get the same results in two rounds of computations, or one may simply apply this sampling technique to provide faster offline algorithms.

A paper containing these results appeared in the proceeding of AAAI 2017.

1.2 Online Allocations

1.2.1 Prophet Secretary

Optimal stopping theory is a powerful tool for analyzing scenarios such as online auctions in which we generally require optimizing an objective function over the space of stopping rules for an allocation process under uncertainty. Perhaps the most classic problems of stopping theory are the prophet inequality problem and the secretary problem. The classical prophet inequality states that by choosing the same threshold $OPT/2$ for every step, one can achieve the tight competitive ratio of 0.5. On the other hand, for the basic secretary problem, the optimal strategy is to ignore the first $\frac{n}{e}$ elements of the sequence while using the maximum of the first $\frac{n}{e}$ elements as the threshold for the rest of sequence. This strategy achieves the optimal competitive ratio of $1/e = 0.36$.

In Chapter 5, we introduce *prophet secretary*, a natural combination of the prophet inequality problem and the secretary problem. An example of motivations for our problem is as follows. Consider a seller that has an item to sell on the market to a set of arriving customers. The seller knows the types of customers that may be interested in the item and he has a price distribution for each type: the price offered by a customer of a type is anticipated to be drawn from the corresponding distribution. However, the customers arrive in a random order. Upon the arrival of a customer, the seller makes an irrevocable decision to whether sell the item at the offered price. We address the question of finding a strategy for selling the item at a high price.

We show that by using a single uniform threshold one cannot break the 0.5 barrier of the prophet inequality for the prophet secretary problem. However, we show that

- using n distinct non-adaptive thresholds one can indeed obtain the competitive ratio of $(1 - 1/e \approx 0.63)$; and
- no online algorithm can achieve a competitive ratio better than 0.75.

We also consider the minimization variants of the prophet inequality problem. In particular, we show that, even for the simple case in which numbers are drawn from identical and independent distributions (i.i.d.), there is no constant competitive online algorithm for the minimization variants of the prophet inequality and prophet secretary problems.

A paper containing these results appeared in the proceeding of ESA 2015.

1.2.2 IID Prophet Inequalities

Hill and Kertz studied the *prophet inequality* on iid distributions [*The Annals of Probability 1982*]. They proved a theoretical bound of $1-\frac{1}{e}$ on the approximation factor of their algorithm. They conjectured that the best approximation factor for arbitrarily large n is $\frac{1}{1+1/e} \simeq 0.731$. This conjecture remained open prior to this work for over 30 years.

In Chapter 6 we present a threshold-based algorithm for the prophet inequality with n iid distributions. Using a nontrivial and novel approach we show that our algorithm is a 0.738-approximation algorithm. By beating the bound of $\frac{1}{1+1/e}$, this refutes the conjecture of Hill and Kertz.

Moreover, we generalize our results to non-iid distributions and discuss its applications in mechanism design.

A paper containing these results appeared in the proceeding of STOC 2017.

1.2.3 Bi-Objective Online Matching

Online allocation problems have been widely studied due to their numerous practical applications (particularly to Internet advertising), as well as considerable theoretical interest. The main challenge in such problems is making assignment decisions in the face of uncertainty about future input; effective algorithms need to predict which constraints are most likely to bind, and learn the balance between short-term gain and the value of long-term resource availability.

In many important applications, the algorithm designer is faced with multiple objectives to optimize. While there has been considerable work on multi-objective offline optimization (when the entire input is known in advance), very little is known about the online case, particularly in the case of adversarial input. In Chapter 7, we give the first results for bi-objective online *submodular* optimization, providing almost matching upper and lower bounds for allocating items to agents with two submodular value functions. We also study practically relevant special cases of this problem related to Internet advertising, and obtain improved results. All our algorithms are nearly best possible, as well as being efficient and easy to implement in practice.

A paper containing these results appeared in the proceeding of NIPS 2016.

1.2.4 Online Allocation with Traffic Spikes

Motivated by Internet advertising applications, online allocation problems have been studied extensively in various adversarial and stochastic models. While the adversarial arrival models are too pessimistic, many of the stochastic (such as i.i.d or random-order) arrival models do not realistically capture uncertainty in predictions. A significant cause for such uncertainty is the presence of unpredictable traffic spikes, often due to breaking news or similar events. To address this issue, a simultaneous approximation framework has been proposed to develop algorithms that work well both in the adversarial and stochastic models; however, this framework does not enable algorithms that make good use of partially accurate forecasts when making online decisions. In Chapter 8, we propose a robust online stochastic model that captures the nature of traffic spikes in online advertising. In our model, in addition to the stochastic input for which we have good forecasting, an unknown number of impressions arrive that are adversarially chosen. We design algorithms that combine an stochastic algorithm with an online algorithm that adaptively reacts to inaccurate predictions. We provide provable bounds for our new algorithms in this framework. We accompany our positive results with a set of hardness results showing that that our algorithms are not far from optimal in this framework. As a byproduct of our results, we also present improved online algorithms for a slight variant of the simultaneous approximation framework.

A paper containing these results appeared in the proceeding of EC 2015.

Chapter 2: Matching in Streaming Setting

2.1 Introduction

As noted by Lovasz and Plummer in their classic book [1],"*Matching Theory* is a central part of graph theory, not only because of its applications, but also because it is the source of important ideas developed during the rapid growth of combinatorics during the last several decades." In the classical offline model, where we assume we have enough space to store all vertices and edges of a graph $G = (V, E)$, the problem of computing the maximum matching of G has been extensively studied. The best result in this model is the 30-years-old algorithm due to Micali and Vazirani [2] with running time $\mathcal{O}(m\sqrt{n})$, where $n = |V|$ and $m = |E|$.

In contrast, in the *streaming* model, the algorithm has access to a sequence of edges, called a *stream*. The algorithm reads edges in the order in which they appear in the stream. The main goal is to design an algorithm that solves a given problem, using as little space as possible. In particular, we wish that the amount of space be sublinear in the size of the input.

Note that the size of a maximum matching in a graph can be as large as $\Omega(n)$. Hence there is little hope to solve the problem exactly or even with a relatively good approximation in $o(n)$ space, since the algorithm has to remember the labels of endpoints for each edge in the matching. This and similar constraints with other graph problems were the main motivation behind the *semi-streaming* model introduced by Feigenbaum, Kannan, McGregor, Suri, and Zhang [3]. In this model, the streaming algorithm is allowed to use $\tilde{\mathcal{O}}(n)$ space (the $\tilde{\mathcal{O}}$ notation hides logarithmic dependencies). The semi-streaming model allows for finding a *maximal* matching (a 2-approximation for the maximum matching) using $\tilde{\mathcal{O}}(n)$ space in a greedy manner. For every new edge (u, v), we add it to the current matching if both u and v are unmatched; otherwise we discard it.

Unfortunately, in many real-world applications where the data is modeled by a massive graph, we may not be able to store all vertices in main memory. Hence we investigate the following natural question: *Given the stream of edges of a massive graph G, how well can the maximum matching size be approximated in $o(n)$ space?* We stress again that we focus on approximating the *size* of the maximum matching, not computing a large matching.

The problem of estimating the size of a maximum matching of a graph has been studied relatively well in the case of *sublinear-time algorithms* [4–8]. Surprisingly, very little is known about this problem in the streaming model which is one of the most fundamental models of the area of algorithms for big data. The only known result is a recent algorithm by Kapralov, Khanna, and Sudan [9], which computes an estimate within a factor of $\mathcal{O}(\text{polylog}(n))$ in the *random-order* streaming model using $\mathcal{O}(\text{polylog}(n))$ space. In the random-order model, the input stream is assumed to be chosen uniformly at random from the set of all possible permutations of the edges. However, to the best of our knowledge, if the algorithm is required to provide a good approximation with high constant probability for any ordering of edges (we refer to this setting as the *adversarial-order* model), nothing is known for the problem of estimating the size of a maximum matching using $o(n)$ space.

2.1.1 Our Results.

In this work, we mainly focus on the family of graphs with bounded arboricity. A graph $G = (V, E)$ has arboricity c if

$$c = \max_{U \subseteq V} \left\lceil \frac{|E(U)|}{|U| - 1} \right\rceil ,$$

where $E(U)$ is the subset of edges with both endpoints in U.[1] Several important families of graphs have constant arboricity. Examples include planar graphs, bounded genus graphs, bounded treewidth graphs, and more generally, graphs that exclude a fixed minor.[2] Our main result is a simple and elegant streaming algorithm with a constant approximation factor in the adversarial-model for estimating the size of a maximum matching in graphs with bounded arboricity using $\tilde{\mathcal{O}}(n^{2/3})$ space (see Theorem 2.1). The problem is non-trivial even when the underlying graph is a tree. For the case of trees (or forests), we give a $2(1 + \epsilon)$-approximation algorithm that uses $\tilde{\mathcal{O}}(\sqrt{n})$ space, where ϵ is an arbitrarily small constant.

We complement our upper bounds with polynomial space lower bounds. We show that no algorithm that uses $o(n^{1/2})$ bits of space can estimate the maximum

[1]Equivalently, the arboricity of a graph can be defined as the minimum number of forests into which its edges can be partitioned.

[2]It can be shown that for an H-minor-free graph, the arboricity number is $\mathcal{O}(h\sqrt{h})$ where h is the number of vertices of H. [10]

matching size to within a factor of less than 3/2 with probability at least 3/4 on every input. A deterministic algorithm that is required to never err and always provide an estimate within a factor better than 3/2 cannot use only $o(n^{1/2})$ bits of space.

2.1.2 Unsuccessful Approaches.

As mentioned before, little is known about the adversarial-order version of the problem. Before sketching our main ideas, we briefly mention main difficulties in applying successful techniques from related areas of algorithms. In the area of sublinear-time algorithms [4–8], the technique of choice has been *local exploration*. A number of dynamic algorithms for matchings [11–15] have applied various *partitioning* techniques. The polylog(n)-approximation algorithm for random-order streams [9] employs a combination of both partitioning and exploration.

Local exploration is applied by sublinear-time algorithms for the maximum matching size. They locally apply greedy techniques for constructing a good matching. Given random access to a graph $G = (V, E)$ with degree bounded by d, they sample a small number of vertices and explore their local neighborhoods using breadth-first search to bounded depth. Using the information gathered from the exploration, they estimate the fraction of vertices in a locally constructed large matching. By exploring this approach one can obtain an algorithm that approximates the maximum matching size within an additive error of $\epsilon \cdot n$ in time $d^{\mathcal{O}(1/\epsilon^2)}$ [6].

The partitioning approach used by some dynamic algorithms for maintaining a large matching [12, 13] is based on a hierarchical decomposition of the vertex set into a logarithmic number of layers. Depending on a specific approach taken, the decomposition can be a complicated function of the graph structure and the history of operations on the graph. In general, high degree vertices tend to be included in higher layers and low degree vertices tend to be included in lower layers. The exact placement of a vertex depends, however, on its connections to other vertices. Equipped with additional information, such a decomposition allows for easy access to a large matching, which is a result of combining together large matchings involving each layer. Kapralov, Khanna, and Sudan [9] show that one can get $\mathcal{O}(\text{polylog}(n))$-approximation to the size of a maximum matching in the *random-order* model, using $\mathcal{O}(\text{polylog}(n))$ space. The intuition behind their approach can be interpreted as a combination of the local exploration and partitioning approaches. In the random-order variant, for any arbitrary edge e, we may assume that a large fraction of edges adjacent[3] to e comes after e. Because of that one can get samples from neighborhoods of vertices and edges in the graph. This turns out to be enough to construct a recursive exploration procedure for estimating the number of vertices in a specific layer of the decomposition.

These approaches do not seem to be effective in the adversarial-order model. Local exploration may be made difficult by placing edges in order unsuitable for following sequences of edges. This idea is used in works [16, 17] to show lower bounds for the amount of space necessary for building a BFS tree from a specific node and for verifying if two nodes are at small distance, even with a few passes

[3]Two edges are adjacent if they share an endpoint.

over the stream.

The partitioning approach is also difficult to implement due to the relations between vertices. Identifying them require some amount of exploration. For instance, the algorithm of Kapralov et al. [9] for random-order streams uses a recursive procedure for determining which layer a given vertex belongs to. Executing the procedure requires exploration from low-degree to high-degree vertices and sampling random neighbors of each visited vertex. In a random-order stream, after reaching a high-degree vertex, the algorithm is still likely to see sufficiently many incident edges. This is not guaranteed for adversarial-order streams.

2.1.3 Our Techniques.

Our approach consists of two main parts. As discussed before, it is inherently hard to probe a certain neighborhood of the graph in the adversarial-order streaming model. Therefore in the first part, we present combinatorial parameters of bounded-arboricity graphs (including graphs that exclude a fixed-minor) that (i) provide a constant approximation for the size of a maximum matching; and more crucially (ii) computing the parameters do not involve a neighborhood search. In the second part, we design estimators for these parameters in the streaming model by sampling subgraphs with certain properties and carefully bounding the positive correlation between the samples. We hope that this approach together with the structural properties provided in this chapter may help in estimating other graph properties in the streaming model as well. For a vertex $v \in V$, let $\deg(v)$ denote the degree of vertex v in G. Let β_G denote an upper bound on the *average* degree of *every* subgraph of G. Recall that for planar graphs $\beta_G \leq 6$. In fact, if the arboricity of a graph is ν, it is easy to verify that $\beta_G \leq 2\nu$. Thus for the family of graphs with constant arboricity (including graphs that exclude a fixed minor), we may assume that β_G is constant. We use the following intuition behind the structural properties. Suppose we sample a small subset of vertices $V' \subseteq V$. Let $G[V']$ denote the subgraph induced by V'. If the maximum degree of V' is constant, then every edge is adjacent to at most a constant number of other edges. Thus the size of a maximum matching in $G[V']$ can be in fact approximated by simply the number of edges in $G[V']$. On the other hand, if the degrees of vertices are large, most of the edges fall between V' and $\overline{V'}$, since the subgraph $G[V']$ is sparse. Therefore in this case one may hope to find a large matching between V' and $\overline{V'}$ that covers most of V'. Hence the size of such a matching can be approximated by $|V'|$. This intuition can be formalized as follows.

Let $\mu = \lceil \beta \rceil + 3$ denote a constant threshold. A vertex v is *light*, if $\deg(v) \leq \mu$; otherwise, the vertex is *heavy*. An edge is *shallow*, if both of its endpoints are light. Throughout this chapter, let h_G and s_G denote the number of heavy vertices and the number of shallow edges in G, respectively. We may drop the index G when it is clear from the context. The crux of our algorithm is the following two-fold structural property of graphs.

Lemma 2.1. *Let G be a graph with maximum matching size M^*. We have the following bounds:*

- *upper bound:* $M^* \leq h_G + s_G$,
- *lower bound:* $M^* \geq \frac{\max\{h_G, s_G\}}{\eta}$,

where $\eta = 1.25\mu + 0.75$. *For graphs with bounded arboricity* ν*, the constant* μ *is at most* $\lceil 2\nu + 3 \rceil$.

It is not hard to verify the upper bound of Lemma 3.10. Observe that every edge of the maximum matching is either shallow, or saturates at least one heavy vertex. Thus the size of a matching cannot be more than the combined number of shallow edges and heavy vertices. On the other hand, proving the lower bound turns out to be quite non-trivial. In Section 2.3, we use the sparsity of graphs with bounded arboricity together with an extension of Hall's Theorem to prove the lower bound on M^*.

By Lemma 3.10, estimating h_G and s_G leads to a constant-factor approximation for the size of a maximum matching. Thus, for the second part of our work, we use random sampling to estimate these parameters. For estimating h_G, we sample a set of vertices and we find the number of heavy vertices among them. We show that a sample of size $\mathcal{O}(\sqrt{n})$ is enough for estimating h_G. For estimating s_G, a major difficulty is that in the adversarial-order setting one does not hope to maintain information about independent (or negatively correlated) samples of edges: when an edge arrives, we may have already seen all the edges adjacent to it! Therefore instead of sampling edges, we sample a set of vertices V' and maintain the *shallow-subgraph* induced by V'. We then show that although the probability of sampling edges is positively correlated, the degree of dependency is *constant* and thus the variance can be bounded; showing that the output of the estimator is highly concentrated. However, to obtain a a good estimator for s_G, we need to maintain a shallow-subgraph of size $\mathcal{O}(n^{2/3})$, which is indeed the space bottleneck.

Theorem 2.1. *Let G be a graph with arboricity ν and $n = \omega(\nu^2)$ vertices. Let $\epsilon, \delta \in (0, 1]$ be two arbitrary positive values less than one. With probability at least $(1 - \delta)$, our algorithm estimates the size of the maximum matching in G within a $((5\nu + 9)(1 + \epsilon)^2)$-factor in the streaming model using $\tilde{\mathcal{O}}(\nu\epsilon^{-2} \log(\delta^{-1})\, n^{2/3})$ space. Both the update time and final processing time are $\mathcal{O}(\log(\delta^{-1}))$.*

In particular, for planar graphs, by choosing $\delta = n^{-1}$ and ϵ as a small constant, the output of our algorithm is within 25-approximation of the size of the maximum matching with probability at least $1 - \frac{1}{n}$ using at most $\tilde{\mathcal{O}}(n^{2/3})$ space.

We further study the limits of the approximation factor by considering the simple but still non-trivial case when the graph is a tree, or more generally, a forest with no isolated vertices. We improve the approximation factor of our general result to (roughly) 2 for the special case of trees while reducing the required memory size to $\tilde{\mathcal{O}}(\sqrt{n})$. Furthermore, in Section 2.6 we further significant improvements may not be possible: using only $o(\sqrt{n})$ bits of space one cannot estimate the size of a maximum matching in forests within a factor better than 1.5.

Theorem 2.2. *Let F be a forest with no isolated vertices. Let $\delta \in (0, 1]$ and $\epsilon \in (0, 1/5]$ be arbitrary. With probability at least $(1 - \delta)$, our algorithm estimates the maximum matching size in F within a factor of $2(1+3\epsilon)$ in the streaming model using $\tilde{\mathcal{O}}(\epsilon^{-2} \log(\delta^{-1}) \sqrt{n})$ space. Both the update time and final processing time are*

$\mathcal{O}(\log(\delta^{-1}))$.

Improving to $\tilde{\mathcal{O}}(\sqrt{n})$ space. We further show that by relaxing the streaming model, one can improve the required space of our algorithm. In particular, given either (i) a *random-order stream* ; or (ii) *two passes over an adversarial-order stream*, we can $(1+\epsilon)$-approximate the number of shallow edges *using only* $\tilde{\mathcal{O}}(\sqrt{n})$ space. This in turn gives $\tilde{\mathcal{O}}(\sqrt{n})$-space algorithm that approximates the size of the maximum matching within $\mathcal{O}(\beta)$ in these models.

In the 2-pass streaming model, we first sample $\tilde{\mathcal{O}}(\sqrt{n})$ edges uniformly at random. In the second pass, we extract the set of shallow edges out of this sample set, leading to an estimation for s_G. If the number of shallow edges is at least $\tilde{\Omega}(\sqrt{n})$, this estimator gives a $(1+\epsilon)$-approximation factor. Otherwise if the number of shallow edges is small, one can argue that maintaining a small maximal matching and estimating h_G is enough to obtain a constant factor approximation factor for the size of the maximum matching. See Section 2.5 for a detailed discussion. For a (one-pass) random-order stream, we can indeed use a similar technique. The idea is that the first $\tilde{\mathcal{O}}(\sqrt{n})$ edges of the stream is in fact a random sample set. Therefore we maintain the first $\tilde{\mathcal{O}}(\sqrt{n})$ edges and we use the rest of the stream to find out how many of them are shallow. This again gives an estimation for the number of shallow edges which leads to an $\mathcal{O}(\beta)$-approximation factor (see Section 2.5 for more details).

Table 2.1 summarizes the known results for estimating the size of a maximum matching.

Reference	Graph class	Stream ordering	Approximation	Space
Folklore	General	Adversarial	$\sqrt{n}$	$\mathcal{O}(\sqrt{n})$
Greedy	General	Adversarial	2	$\mathcal{O}(n)$
[9]	General	Random	$\mathcal{O}(\text{polylog}(n))$	$\mathcal{O}(\text{polylog}(n))$
This work	Planar	Adversarial	25	$\tilde{\mathcal{O}}(n^{2/3})$
This work	Forests	Adversarial	2	$\tilde{\mathcal{O}}(\sqrt{n})$
This work	Bounded Arboricity	Adversarial	$\mathcal{O}(1)$	$\tilde{\mathcal{O}}(n^{2/3})$
This work	Bounded Arboricity	Random	$\mathcal{O}(1)$	$\tilde{\mathcal{O}}(\sqrt{n})$
This work	Bounded Arboricity	2-Pass Adver.	$\mathcal{O}(1)$	$\tilde{\mathcal{O}}(\sqrt{n})$

Table 2.1: Known results for estimating the size of a maximum matching in data streams.

2.1.4 Further Related Streaming Work.

The question of approximating the maximum cardinality matching has been extensively studied in the streaming model. An $\mathcal{O}(n)$-space greedy algorithm trivially obtains a maximal matching, which is a 2-approximation for the maximum cardinality matching [3]. A natural question is whether one can beat the approximation factor of the greedy algorithm with $\tilde{\mathcal{O}}(n)$ space. Very recently, it was shown that obtaining an approximation factor better than $\frac{e}{e-1} \simeq 1.58$ in one pass requires $n^{1+\Omega(1/\log\log n)}$ space [18,19], even in bipartite graphs and in the *vertex-arrival* model.[4]

Closing the gap between the upper bound of 2 and the lower bound of $\frac{e}{e-1}$ remains one of the most appealing open problems in the graph streaming area (see [21]). The factor of 2 can be improved on if one either considers the random-order model or allows for two passes [22]. By allowing even more passes, the approximation factor can be improved to multiplicative $(1-\epsilon)$-approximation via finding and applying augmenting paths with successive passes [23–27].

Another line of research [3,23,28,29] has explored the question of approximating the maximum-weight matching in one pass and $\tilde{\mathcal{O}}(n)$ space. Currently, the best known approximation factor equals $4.9108 + \epsilon$ (for any positive constant ϵ) [29].

2.1.5 Preliminaries.

The Streaming Model. Let S be a stream of edges of an underlying graph $G = (V, E)$. We assume that the vertex set V is fixed and given, and that $|V| = n$. We assume that there is a unique numbering for the vertices in V so that we can treat $v \in V$ as a unique number v for $1 \leq v \leq n$. We denote an undirected edge in E with two endpoints $u, v \in V$ by (u, v). The graph G can have at most $\binom{n}{2} = n(n-1)/2$ edges. Thus, each edge can also be thought of as referring to a unique number

[4]In the vertex-arrival model, the vertices arrive in the stream together with all their incident edges. This setting has also been studied in the context of *online algorithms*, where each arriving vertex has to be either matched or discarded irrevocably upon arrival. Seminal work due to Karp, Vazirani and Vazirani [20] gives an online algorithm with $\frac{e}{e-1}$ approximation factor in the online model.

between 1 and $\binom{n}{2}$.

Adversarial Order. We work in the most-popular *adversarial-order model*. In this model, a streaming algorithm has to compute a satisfying solution with satisfying probability for *every* ordering of items in the input stream. This should be contrasted with the easier *random-order model*, where an algorithm has to provide a satisfying solution with satisfying probability for an input stream permuted uniformly at random.

Approximation Factor. Throughout this chapter, let M^* denote the *size* of a maximum matching in G. Our task is to output M, an estimate of M^*, after receiving all the edges. A (randomized) algorithm has *approximation factor* α with probability $(1-\delta)$ if for every permutation of input edges, with probability at least $(1-\delta)$, we have $M \leq M^* \leq \alpha M$.

Concentration Bounds. We use an extension of the Chernoff-Hoeffding bound for negatively correlated Boolean variables. It was first proved by Panconesi and Srinivasan [30].

Theorem 2.3 ([30]). *If the Boolean random variables $X_1, X_2, \ldots, X_r$ are negatively correlated then for $X = \sum_{i=1}^{r} X_i$, $\mu = E[X]$, and $0 < \delta \leq 1$, we have* $\Pr\left[X < (1-\delta)\mu\right] < e^{-\mu\delta^2/2}$.

We also use Chebyshev's inequality, which we state here for completeness.

Theorem 2.4 (Chebyshev's Inequality)**.** *Let X be a random variable with finite expected value μ and finite non-zero variance σ^2, we have* $\Pr[|X-\mu| \geq k\sigma] \leq \frac{1}{k^2}$.

2.2 Algorithm

To estimate the size of a maximum matching, by Lemma 3.10, it is sufficient to estimate the two parameters h and s, if one is willing to loose a constant (2η) in the approximation factor. Our algorithm consists of three parallel subroutines. Each subroutine reports an estimated lower bound for the size of a maximum matching. The joined result is the maximum of the three estimated values. Since the probabilities of success for the second and the third subroutines are small, we boost the success probabilities by independently running a logarithmic number of copies of these subroutines. The final output is the median of the values reported by all copies. In what follows we describe these subroutines (see Algorithm 1).

Since we are restricted to sublinear space, it is natural to use sampling for estimating h and s. However, the first obstacle occurs in the instances where both h and s are small. Thus we do not hope to capture a heavy vertex (or a shallow edge) using sublinear samples. We overcome these instances by maintaining a *maximal* matching of the graph, up to a sublinear size. Therefore for instances where $M^* = \tilde{\mathcal{O}}(n^{2/3})$, we can estimate M^* up to a factor of two (see Subroutine 1).

We now focus on the instances where M^* is relatively large, and thus we cannot keep a maximal matching in the memory. By the upper bound of Lemma 3.10, at least one of h or s is large. If the number of heavy vertices is large, we can estimate

h using a *vertex-set sample*. We sample $\tilde{\mathcal{O}}(n^{2/3})$ random vertices and we maintain their degrees throughout the execution of the algorithm. We note that in a vertex-set sample, we hit roughly $\Omega(\frac{h}{n^{1/3}})$ heavy vertices. Therefore h would be proportional to the fraction of sampled vertices that are heavy (see Subroutine 2).

Now we are left with the most subtle scenario that h is small, but the number of shallow edges is large. In this case we face a new obstacle inherent of the adversarial setting; it is often not possible to *independently* sample edges and maintain information about their neighborhood. At the time that an edge arrives, we may have already seen all the adjacent edges and thus we will not be able to distinguish whether an edge has a desired property. Recall that an edge is shallow if both its endpoints are light. Therefore instead of sampling edges, we maintain a *shallow-subgraph* sample.

We first sample a set V' of $\mathcal{O}(n^{2/3})$ vertices. We maintain the degrees of these vertices throughout the execution. At an iteration i (i.e., after receiving the i-th edge), let $L' \subseteq V'$ denote the set of vertices of V' with degree at most μ. We maintain all the edges induced by L'. Observe that vertices may become heavy and leave L', in which case we will ignore their adjacent edges (see Subroutine 3). This guarantees that we only need $\tilde{\mathcal{O}}(\mu n^{2/3})$ space for keeping all the edges in our shallow-subgraph sample. After processing the entire input stream, the maintained edges are exactly the shallow edges induced by V'. Every vertex is sampled with probability (roughly) $n^{-1/3}$, however, the probabilities are not independent. Fortunately since we only have negative correlation, we can still say that a shallow edge falls in the induced subgraph with probability roughly $n^{-2/3}$. However as mentioned before,

Algorithm 1 Maximum Matching Estimator

```
Input: A stream of edges of the graph $e_1, \ldots, e_m$.

Output: An estimation $M$ of the size of a maximum matching $M^*$.

Initialization Process:

1: Set $\alpha_1 = 3$, $\alpha_2 = 3\epsilon^{-2}$, $\alpha_3 = 4\epsilon^{-1}$ and $Q = 26\lceil \ln(\frac{1}{\delta}) \rceil + 1$.
2: Set $c_1 = c_2 = c_3 = 2/3$.
3: Initialize Subroutine 1 with factor $\alpha_1$ and $c_1$.
4: Initialize $Q$ independent executions of Subroutines 2 and 3, with the corresponding $\alpha$ and $c$ factors.

Update Process, upon the arrival of $e_i = (u, v)$:

1: Update each subroutine with the new arrival.

Termination Process:

1: Let $R$ denote the return value of Subroutine 1.
2: For $i \in [Q]$, let $\hat{h}_i$ denote the value returned by the $i$-th execution of Subroutine 2. Similarly, let $\hat{s}_i$ denote that of Subroutine 3.
3: For $i \in [Q]$, let $M_i = \max\{\alpha_1 n^{2/3}, \hat{h}_i, \hat{s}_i\}$.
4: if $R < \alpha_1 n^{2/3}$ then
5:     Output $M = R$.
6: else
7:     Let $M_{\text{med}}$ denote the median value of $M_i$'s.
8:     Output $M = \frac{M_{\text{med}}}{(1+\epsilon)\eta}$.
9: end if
```

we do not hope to have an efficient sublinear sampling that independently (or even with negative correlation) samples the edges. Our algorithm is not an exception either. For two edges that share an endpoint, the probabilities of hitting them in a shallow-subgraph sample is *positively* correlated. In general, one cannot hope for strong concentration bounds at the presence of positive correlation. However, the crux of our analysis is that since the light vertices have bounded degree, the degree of dependency is constant. Therefore we are able to bound the variance of our estimator and show that with a constant probability the estimated value is highly concentrated around its expectation. We believe this approach may be of its own interest for testing properties of (sub-)graphs with bounded degree. Finally to boost the probability of success, we maintain independent estimators of subroutines in our algorithm (see Algorithm 1).

2.2.1 Main Subroutines.

The subroutines of Algorithm 1 are as follows.

Subroutine 1. In this subroutine, we greedily construct a maximal matching M''. However, we stop the process if $|M''|$ exceeds the limit $\alpha_1 n^{c_1}$, where the constants α_1 and c_1 are tuned by the algorithm.

Subroutine 2. In this subroutine, we choose a subset of size $\alpha_2 n^{c_2}$ from the set of vertices uniformly at random. The parameters α_2 and c_2 are tuned by the algorithm. During the execution, we maintain the degrees of sampled vertices.

Process 1. [Bounded Maximal Matching]

Input: A stream of edges of the graph $e_1, \dots, e_m$.

Return Value: $R := \min\{|M'|, \alpha_1 n^{c_1}\}$ where M' is a maximal matching.

Initialization Process:

```
1: Initialize $M''$ to an empty set.
```

Update Process, upon the arrival of $e_i = (u, v)$:

```
1: if $|M''| < \alpha_1 n^{c_1}$ and $M'' \cup \{e_i\}$ is a valid matching then
2:     $M'' \leftarrow M'' \cup \{e_i\}$.
3: else
4:     Discard $e_i$.
5: end if
```

Termination Process:

```
1: Return $|M''|$.
```

Subroutine 3. In this subroutine, we again choose a subset of size $\alpha_3 n^{c_3}$ from the set of vertices uniformly at random. However, in addition to the degrees of vertices, we also maintain the edges that the degrees of their endpoints are at most μ. Thus we require at most $\tilde{\mathcal{O}}(\mu \alpha_3 n^{c_3})$ memory space.

2.2.2 Analysis.

In this section we analyze the approximation ratio and the success probability of our randomized algorithm. We first distinguish between two primary cases, de-

Process 2. [Heavy Vertices]

Input: A stream of edges of the graph $e_1, \ldots, e_m$.

Return Value: $\hat{h}$, an estimation for the number of heavy vertices in the graph.

Initialization Process:

```
1: Initialize $V' \subset V$ as a random subset of size $\alpha_2 n^{c_2}$.
2: For every $v \in V'$, initialize $\deg(v)$ to zero.
```

Update Process, upon the arrival of $e_i = (u, v)$:

```
1: if $u \in V'$ then
2:     $\deg(u) \leftarrow \deg(u) + 1$.
3: end if
4: if $v \in V'$ then
5:     $\deg(v) \leftarrow \deg(v) + 1$.
6: end if
```

Termination Process:

```
1: Let $h' = |\{v \in V' | \deg(v) > \mu\}|$ denote the number of heavy vertices in $V'$.
2: Return $\hat{h} = h' \times \frac{n^{1-c_2}}{\alpha_2}$.
```

Process 3. [Shallow Edges]

Input: A stream of edges of the graph $e_1, \ldots, e_m$.

Return Value: $\hat{s}$, an estimation for the number of shallow edges in the graph.

Initialization Process:

```
1: Initialize $V' \subset V$ as a random subset of size $\alpha_3 n^{c_3}$.
2: For every $v \in V'$, initialize $\deg(v)$ to zero and $E'$ to an empty set.
```

Update Process, upon the arrival of $e_i = (u, v)$:

```
1: if both $u, v \in V'$ then
2:     $E' \leftarrow E' \cup \{e_i\}$.
3: end if
4: if $u \in V'$ then
5:     $\deg(u) \leftarrow \deg(u) + 1$.
6:     If $\deg(u) > \mu$, remove all the edges adjacent to $u$ from $E'$.
7: end if
8: if $v \in V'$ then
9:     $\deg(v) \leftarrow \deg(v) + 1$.
10:    If $\deg(v) > \mu$, remove all the edges adjacent to $v$ from $E'$.
11: end if
```

Termination Process:

```
1: Let $s' = |E'|$ denote the number of shallow edges induced by $V'$.
2: Return $\hat{s} = s' \times \frac{n(n-1)}{(|V'|)(|V'|-1)}$.
```

pending on the size of a maximum matching M^*. Recall that the size of a maximal matching is always withing a factor 2 of a maximum matching. The first subroutine of our algorithm, maintains the size R of a maximal matching up to $\alpha_1 n^{2/3}$. Observe that in Algorithm 1, we completely ignore the second and third subroutines if $R < \alpha_1 n^{2/3}$. Therefore for the input scenarios that $R < \alpha_1 n^{2/3}$, we indeed have a maximal matching of size R and thus our algorithm *deterministically* estimates M^* within a two factor. In the rest of this section we focus on the case where $\alpha_1 n^{2/3} \leq R \leq M^*$.

By Lemma 3.10, we know that either h or s is large (respectively, the number of heavy vertices and the number of shallow edges). Consider a single execution of Subroutine 2 (resp. Subroutine 3), that returns an estimation for $\hat{h}$ (resp. $\hat{s}$). In our algorithm, we maintain $\log(\frac{1}{\delta})$ executions of the subroutines to boost the success probability. Therefore the goal is to show that for a single execution, with a constant probability the output of our algorithm is within a constant factor of M^*. Since we output the *maximum* of the two estimations, we need to show two concentration bounds for each estimator: (i) if the target value is large, with constant probability we get a constant approximation; and (ii) if the target value is small, with constant probability the estimation is small too. We prove this for both estimators in the following lemmas.

Lemma 2.2. *For Subroutine 2, we have:*

- *If* $h \geq n^{2/3}$, $\Pr\left[\left|\hat{h} - h\right| \geq \epsilon h\right] \leq 2\exp(-n^{1/3})$*; and*
- *If* $h < n^{2/3}$, $\Pr\left[\hat{h} > \alpha_1 n^{2/3}\right] \leq \exp(-n^{1/3})$.

Proof. Recall in the subroutine we select a random sample V' of size $n' = \alpha_2 n^{2/3}$. Let $V' = \{v'_1, \ldots, v'_{n'}\}$. For $i \in [n']$, let X_i denote the random variable where $X_i = 1$ if v'_i is a heavy vertex, and $X_i = 0$ otherwise. Since we have h heavy vertices, we have that $\Pr[X_i = 1] = \frac{h}{n}$ for every $i \in [n']$. Let $h' = \sum_i X_i$. Clearly $\mathrm{E}[h'] = \frac{n' \cdot h}{n}$. Recall that the return value of Subroutine 2 is $\hat{h} = h' \times \frac{n}{n'}$. Thus in expectation, we return the value of h.

Now since all X_i's are negatively correlated, we use the extension of Chernoff bound 2.3 to prove the first bound of the lemma for $h \geq n^{2/3}$:

$$\begin{aligned}
\Pr\left[\left|\hat{h} - h\right| \geq \epsilon h\right] &= \Pr\left[\left|\hat{h} - \mathrm{E}\left[\hat{h}\right]\right| \geq \epsilon\, \mathrm{E}\left[\hat{h}\right]\right] \\
&= \Pr\left[\left|h' - \mathrm{E}[h']\right| \geq \epsilon\, \mathrm{E}[h']\right] \\
&\leq 2\exp\left(-\frac{\epsilon^2\, \mathrm{E}[h']}{3}\right) \\
&\leq 2\exp(-n^{1/3}).
\end{aligned}$$

We again use the Chernoff bound to prove the lemma for $h < n^{2/3}$:

$$\begin{aligned}
\Pr\left[\hat{h} > \alpha_1 n^{2/3}\right] &\leq \Pr\left[\hat{h} > \left(\frac{\alpha_1 n^{2/3}}{h}\right) \mathrm{E}\left[\hat{h}\right]\right] \\
&= \Pr\left[h' > \left(\frac{\alpha_1 n^{2/3}}{h}\right) \mathrm{E}[h']\right] \\
&\leq \Pr\left[h' > \left(1 + \frac{\alpha_1 n^{2/3}}{2h}\right) \mathrm{E}[h']\right] \\
&\leq \exp\left(-\frac{\alpha_1^2 \cdot n^{4/3} \cdot \mathrm{E}[h']}{12h^2}\right) \\
&\leq \exp\left(-\left(\frac{\alpha_1}{2}\right)^2 n^{1/3}\right).
\end{aligned}$$

□

Indeed for the shallow-edge estimator, the problem is more sophisticated. For a shallow edge $e = (u, v)$, the probability of hitting e in a shallow subgraph is positively correlated to that of all edges adjacent to u and v. Thus a normal application of (generalized variants of) the Chernoff bound fails to satisfy the required concentration bounds. Indeed in the next lemma, we use the small dependency degree between the variables to prove the concentration bounds.

Lemma 2.3. *For Subroutine 3, we have:*

- *If* $s \geq n^{2/3}$, $\Pr[|\hat{s} - s| \geq \epsilon s] \leq 1/4$*; and*
- *If* $s < n^{2/3}$, $\Pr\left[\hat{s} > \alpha_1 n^{2/3}\right] \leq \alpha_1^{-2} = 1/9$.

Proof. Recall that in Subroutine 3, we sample a shallow subgraph (V', E') where $|V'| = n' = \alpha_3 n^{2/3}$. Let $e_1, \ldots, e_s$ denote the shallow edges in the original graph. For a shallow edge e_i, let X_i denote the random variable where $X_i = 1$ if $e_i \in E'$ and $X_i = 0$ otherwise. Let $p = \Pr[X_i = 1]$. We have

$$\mathrm{E}[X_i] = \Pr[X_i = 1] = p = \frac{\binom{n'}{2}}{\binom{n}{2}} = \frac{n'(n'-1)}{n(n-1)}.$$

We note that this probability is indeed very small $p \in [\frac{\alpha_3^2 n^{-2/3}}{2}, \alpha_3^2 n^{-2/3}]$. Let $s' = \sum_i X_i$; note that $\mathrm{E}[s'] = sp$. In order to get a concentration bound for s', we would need to bound the variance of our estimator. We know that

$$\mathrm{Var}(s') = \sum_i \mathrm{Var}(X_i) + \sum_{i \neq j} \mathrm{Cov}(X_i, X_j).$$

Thus we need to calculate $\mathrm{Var}\left(X_i\right)$ and $\mathrm{Cov}\left(X_i, X_j\right)$ for which we have

$$\begin{aligned}
&\mathrm{Var}\left(X_i\right) = \mathrm{E}\left[(X_i - \mathrm{E}\left[X_i\right])^2\right] \\
&= \Pr\left[X_i = 1\right](1 - \mathrm{E}\left[X_i\right])^2 + \Pr\left[X_i = 0\right](\mathrm{E}\left[X_i\right])^2 \\
&= p(1-p)^2 + (1-p)(p)^2 \\
&= p(1-p) \leq p.
\end{aligned}$$

For the covariance, we have

$$\begin{aligned}
\mathrm{Cov}\left(X_i, X_j\right) &= \mathrm{E}\left[X_i X_j\right] - \mathrm{E}\left[X_i\right]\mathrm{E}\left[X_j\right] \\
&= \Pr\left[X_i = X_j = 1\right] - p^2.
\end{aligned}$$

Thus we have two cases based on whether the edges corresponding to X_i and X_j share an endpoint:

(i) if e_i and e_j share an endpoint, $\Pr\left[X_i = X_j = 1\right]$ is the probability of having 3 fixed vertices in V'; and

(ii) if e_j and e_j do not share an endpoint, $\Pr\left[X_i = X_j = 1\right]$ is the probability of having 4 fixed vertices in V'.

For case (i) we have

$$\begin{aligned}
\mathrm{Cov}\left(X_i, X_j\right) &= \frac{\binom{n-3}{n'-3}}{\binom{n}{n'}} - p^2 \\
&= \frac{n'(n'-1)(n'-2)}{n(n-1)(n-2)} - p^2 < p^{3/2} - p^2 \leq p^{3/2}.
\end{aligned}$$

For case (ii) we have

$$\begin{aligned}
\mathrm{Cov}\left(X_i, X_j\right) &= \frac{\binom{n-4}{n'-4}}{\binom{n}{n'}} - p^2 = \frac{n' \cdots (n'-3)}{n \cdots (n-3)} - p^2 \\
&< p^2 - p^2 = 0.
\end{aligned}$$

This implies that we can completely ignore the terms for Case (ii) since we are interested in an upper bound on the variance. On the other hand, the degree of a light vertex is at most μ. Hence, every shallow edge is adjacent to at most 2μ other edges. Leading to the following upper bound for the variance of s' for $p = \Theta(n^{-2/3})$ and $n \geq (2\mu)^3$

$$\begin{aligned}
\operatorname{Var}(s') &= \sum_{i=1}^{s} \operatorname{Var}(X_i) + \sum_{i \neq j} \operatorname{Cov}(X_i, X_j) \\
&\leq sp + \sum_{i \neq j} \operatorname{Cov}(X_i, X_j) \\
&\leq sp + \sum_{i \neq j : e_i \text{ and } e_j \text{ share an endpoint}} p^{3/2} \\
&\leq sp + s(2\mu)(p^{3/2}) \leq 2sp.
\end{aligned}$$

Now we can indeed use the Chebyshev's inequality 2.4 to prove the desired concentration bounds. For $s \geq n^{2/3}$, we have

$$\begin{aligned}
&\Pr\left[|\hat{s} - s| \geq \epsilon s\right] \\
&= \Pr\left[|\hat{s} - \mathrm{E}[\hat{s}]| \geq \epsilon \, \mathrm{E}[\hat{s}]\right] = \Pr\left[|s' - \mathrm{E}[s']| \geq \epsilon \, \mathrm{E}[s']\right] \\
&\leq \frac{\operatorname{Var}(s')}{\epsilon^2 (\mathrm{E}[s'])^2} \leq \frac{2}{\epsilon^2 \cdot n^{2/3} \cdot p} \leq \frac{4n^{2/3}}{\epsilon^2 n^{2/3} \alpha_3^2} \leq 1/4.
\end{aligned}$$

One can use a similar argument to prove the concentration bound for $s < n^{2/3}$:

$$\begin{aligned}
\Pr\left[\hat{s} > \alpha_1 n^{2/3}\right] &= \Pr\left[\hat{s} > \left(\frac{\alpha_1 n^{2/3}}{\mathrm{E}[\hat{s}]}\right) \mathrm{E}[\hat{s}]\right] \\
&= \Pr\left[s' > \left(\frac{\alpha_1 n^{2/3}}{s}\right) \mathrm{E}[s']\right]
\end{aligned}$$

$$\leq \frac{\operatorname{Var}(s')}{\left(\mathrm{E}[s']\left(\frac{\alpha_1 n^{2/3}}{s} - 1\right)\right)^2}$$

$$\leq \frac{4s^2 \operatorname{Var}(s')}{s^2 p^2 \alpha_1^2 n^{4/3}} \leq \alpha_1^{-2}.$$

□

We are now ready to prove the main theorem.

Proof of Theorem 2.1: Consider the parameters in Algorithm 1. As discussed before, if $R < \alpha_1 n^{2/3}$, we deterministically output the size of a maximal matching, which is within a 2-approximation of M^*. Thus we focus on the instances where $M^* \geq \alpha_1 n^{2/3}$. In this scenario, the output is proportional to the median of the M_i values where $M_i = \max\{\alpha_1 n^{2/3}, \hat{h}_i, \hat{s}_i\}$. Let $\tau = \max\{\alpha_1 n^{2/3}, h_G, s_G\} = \max\{h_G, s_G\}$. By Lemma 3.10, we know that at least one of h_G and s_G is at least $\frac{M^*}{2} > n^{2/3}$. We say the i^{th} execution of the sampling is *successful*, if $|M_i - \tau| \leq \epsilon\tau$. Let X_i denote the random variable where $X_i = 1$ if the i^{th} sample is successful and $X_i = 0$ otherwise. Let us consider the event $X_i = 1$. We have two cases based on whether $h_G \geq s_g$. One can use the union bound to get a lower bound on the probability of success.

For $\tau = h_G$, we have

$$\Pr[X_i = 1]$$

$$\geq 1 - \Pr\left[\left|\hat{h} - h_G\right| \geq \epsilon h_G\right] - \Pr\left[\hat{s} > \alpha_1 n^{2/3}\right].$$

For $\tau = s_G s$, we have

$$\begin{aligned}&\Pr\left[X_i = 1\right]\\ &\geq 1 - \Pr\left[|\hat{s} - s_G| \geq \epsilon s_G\right] - \Pr\left[\hat{h} > \alpha_1 n^{2/3}\right].\end{aligned}$$

Now we can use the concentration bounds given by Lemmas 2.2 and 2.3, to get a constant lower bound for $\Pr\left[X_i = 1\right]$. Note that we can assume $n \geq 125$ since otherwise we can keep the (constant-size) input in memory. Thus, we have $n^{1/3} \geq 5$.

$$\begin{aligned}&\Pr\left[X_i = 1\right]\\ &\geq \min\{1 - \Pr\left[\left|\hat{h} - h_G\right| \geq \epsilon h_G\right] - \Pr\left[\hat{s} > \alpha_1 n^{2/3}\right]\\ &\qquad , 1 - \Pr\left[|\hat{s} - s_G| \geq \epsilon s_G\right] - \Pr\left[\hat{h} > \alpha_1 n^{2/3}\right]\}\\ &\geq \min\{1 - 2\exp(-n^{1/3}) - \frac{1}{9}\\ &\qquad , 1 - \frac{1}{4} - \exp(-n^{1/3})\}\\ &\geq \min\{8/9 - 2\exp(-5), 3/4 - \exp(-5)\} \geq 0.743.\end{aligned}$$

Observe if for some $i \in [Q]$, $X_i = 1$, then by Lemma 3.10, $\frac{M_i}{\eta(1+\epsilon)} \leq M^* \leq 2M_i(1+\epsilon)$. However, the output of the algorithm is proportional to median value of all M_i's. In the event that more than half of X_i's are successful, the median is indeed successful. Let $X = \sum_{i \in [Q]} X$ denote the number of successful M_i's. With probability at least $\Pr\left[X \geq 0.501Q\right]$, the output of our algorithm is within $2\eta(1+\epsilon)^2$-approximation of M^*. Since the executions are independent, we can use a Chernoff bound to get the success probability of our algorithm.

$$\begin{aligned}
&\Pr[X \geq 0.501Q] \\
&\geq 1 - \exp\left(-\frac{(1 - \frac{0.501Q}{\mathrm{E}[X]})^2 \,\mathrm{E}[X]}{2}\right) \\
&\geq 1 - \exp\left(-\frac{(1 - 0.501/0.743)^2 \,\mathrm{E}[X]}{2}\right) \\
&\geq 1 - \exp(-0.053 \,\mathrm{E}[X]) \geq 1 - \exp(-0.039Q) \\
&\geq 1 - \exp(-\ln(\delta^{-1})) = 1 - \delta,
\end{aligned}$$

where $\mathrm{E}[X] = \Pr[X_i = 1]\, Q \geq 0.743Q$ and $Q \geq 26\ln(\delta^{-1})$.

Therefore, with probability at least $1-\delta$, the output of our algorithm is within $(2\eta)(1+\epsilon)^2$-approximation of the size of the maximum matching. It only remains to analyze the running time of our algorithm. Observe that the update process and the termination process of each single subroutine can be implemented to run in $O(1)$ time. Therefore the update time of our algorithm is linear to the number of subroutines $\mathcal{O}(Q) = \mathcal{O}(\ln(\delta^{-1}))$. Furthermore, at the end of the execution of the algorithm, we need to find the median of Q values which can also be implemented in $\mathcal{O}(\ln(\delta^-1))$ time. ■

2.3 Proof of Lemma 3.10

In a maximum matching, the number of edges that are incident to at least one heavy vertex cannot be more than the number of heavy vertices h_G: two matching edges cannot share a (heavy) vertex. On the other hand, the edges that are not adjacent to a heavy vertex are induced by light vertices, i.e., such an edge is a shallow edge. The number of these edges is at most s_G, and thus, the maximum matching size is at most $h_G + s_G$.

In order to prove the lower bound, we first show that there is a matching of size $\frac{4s_G}{5\mu+3}$ in the induced subgraph of light vertices. This implies $M^* \geq \frac{4s_G}{5\mu+3}$. We then show a matching of size $\frac{\mu-\beta+1}{2\mu}h_G$ in G, which means that $M^* \geq \frac{\mu-\beta+1}{2\mu}h_G$. Overall, this implies that $M^* \geq \max\{\frac{\mu-\beta+1}{2\mu}h_G, \frac{4}{5\mu+3}s_G\}$. If we set $\mu = \beta + 3$, we have

$$\frac{\mu-\beta+1}{2\mu} = \frac{(\beta+3)-\beta+1}{2(\beta+3)} = \frac{4}{\beta+3},$$
$$\frac{4}{5\mu+3} = \frac{4}{5(\beta+3)+3} = \frac{1}{1.25\beta+4.5}.$$

By comparing the above functions we can see $\frac{4}{5\mu+3} \leq \frac{\mu-\beta+1}{2\mu}$. Thus, we have

$$\begin{aligned} M^* &\geq \max\{\frac{\mu-\beta+1}{2\mu}h_G, \frac{4}{5\mu+3}s_G\} \\ &\geq \max\{\frac{4}{5\mu+3}h_G, \frac{4}{5\mu+3}s_G\} \\ &= \frac{\max\{h_G, s_G\}}{1.25\mu+0.75} \end{aligned}$$

as desired.

In this section, we write L to denote the set of light vertices.

Claim 2.1. *The subgraph induced by L contains a matching of size $\frac{4s_G}{5\mu+3}$.*

Proof. It is known that the maximum matching size in a graph H is at least $\frac{4m}{5\Delta+3}$, where m in the number of edges in H and Δ is the maximum degree of H [31]. By definition, each vertex of L has degree at most μ. Hence, we have a matching of size $\frac{4s_g}{5\mu+3}$ in the graph induced by L. □

In order to prove the next claim, we use the following generalization of Hall's theorem.

Lemma 2.4 ([32]). *Let $G_{(X,Y)}$ be a bipartite graph with bipartition (X, Y). The number of edges in a maximum matching of $G_{(X,Y)}$ is*

$$|X| - \max_{R \subseteq X}(|R| - |N(R)|),$$

where $N(R)$ is the set of all neighbors of vertices in R.

Claim 2.2. *There is a matching of size $\frac{\mu-\beta+1}{2\mu}h_G$ in H, where H is G excluding the edges with both endpoints in L.*

Proof. Let M_h be the set of vertices covered by a maximal matching in the graph induced by heavy vertices and assume $|M_h| = 2\lambda$. Let U be the set of unmatched heavy vertices. Since, M_h is maximal, there is no edge between vertices in U, i.e., U is an independent set. Let $G_{(U,L)}$ denote the bipartite graph consisting of edges connecting vertices in L and U. In the remainder, we use Lemma 2.4 to show the size of a maximum matching in $G_{(U,L)}$ plus $\frac{|M_h|}{2} = \lambda$ is at least $h_G(\frac{1}{2} - \frac{2\beta-2}{\mu-1})$.

Consider the bipartite graph $G_{(U,L)}$ and let R be an arbitrary subset of U. We bound $|N(R)|$ by double counting the number of edges between R and $N(R)$, namely $E[R, N(R)]$. On the one hand, the degree of each vertex in $N(R)$ is at most μ. Hence, we have $E[R, N(R)] \leq \mu N(R)$. On the other hand, vertices in R are heavy, which means they have a degree of at least μ in graph G. However, vertices in R may have some neighbors in M_h that do not exist in $G_{(U,L)}$. Recall that the average degree of vertices in $R \cup M_h$ is at most β and there is at least λ edges between vertices in M_h. Thus, there is at most $\frac{(|R|+2\lambda)\beta}{2} - \lambda$ edges between R and M_h. Therefore, we have $\mu|R| - \left(\frac{(|R|+2\lambda)\beta}{2} - \lambda\right) \leq E[R, N(R)]$. Thus, we have

$$\mu|R| - \left(\frac{|R|\beta}{2} + \lambda\beta - \lambda\right) \leq \mu N(R),$$

which gives us $|R| - \left(\frac{|R|\beta}{2\mu} + \frac{\lambda\beta-\lambda}{\mu}\right) \leq N(R)$. If we apply Lemma 2.4, size of the maximum matching in $G_{(U,L)}$ is at least

$$\begin{aligned}
&|U| - \max_{R\subseteq U}(|R| - |N(R)|) \\
&\geq h_G - 2\lambda - \max_{R\subseteq U}\left(|R| - |R| + \left(\frac{|R|\beta}{2\mu} + \frac{\lambda\beta - \lambda}{\mu}\right)\right) \\
&= h_G - 2\lambda - \frac{\lambda\beta - \lambda}{\mu} - \max_{R\subseteq U}\frac{|R|\beta}{2\mu} \\
&= h_G - 2\lambda - \frac{\lambda\beta - \lambda}{\mu} - \frac{(h_G - 2\lambda)\beta}{2\mu} \\
&= h_G - \frac{h_G\beta}{2\mu} - 2\lambda + \frac{\lambda}{\mu}.
\end{aligned}$$

Therefore, the total size of the matching is at least

$$\lambda + h_G - \frac{h_G\beta}{2\mu} - 2\lambda + \frac{\lambda}{\mu} = h_G - \frac{h_G\beta}{2\mu} - \lambda + \frac{\lambda}{\mu}.$$

This quantity is decreases as a function of λ. Thus, it is minimized for $\lambda = \frac{h_G}{2}$. In this case, the size of the matching is at least

$$h_G - \frac{h_G\beta}{2\mu} - \frac{h_G}{2} + \frac{h_G}{2\mu} = \frac{\mu - \beta + 1}{2\mu}h_G,$$

which completes the proof. □

2.4 Maximum Matching in Forests

In this section we approximate the maximum matching size in a tree within a $2(1+3\epsilon)$ factor. We use some structural property of trees to improve the lower and upper bounds of the maximum matching in Lemma 2.5. Later, we generalize this lemma to forests with no isolated vertices and approximate the maximum matching size of these forests by a $2(1 + 3\epsilon)$ factor. During this section we set $\mu = 1$ i.e. vetices with degree 1 are light and vertices with degree 2 or more are heavy.

Lemma 2.5. *Let T be a tree with maximum matching size M^*. We have the following bounds:*

- *upper bound: $M^* \leq h_T + 1$,*
- *lower bound: $M^* \geq \frac{h_T+1}{2}$.*

Proof. In a tree with more than two vertices, leaves can not be adjacent. Thus, each matching edge shares at least one vertex with the heavy vertices. Thus, maximum matching size in a tree T with more than two vertices is at most h_T. On the other hand, the maximum matching size in a tree with two vertices is 1. Therefore, for every tree we have $M^* \leq \max(h_T, 1) \leq h_T + 1$.

In order to show the lower bound we show that every tree has a matching with the following two property.

- All of the heavy vertices are matched.
- At least one light vertex (leaf) is matched.

This immediately gives us $M^* \geq \frac{h_T+1}{2}$.

Let M be a maximum matching that matches the maximum possible number of heavy vertices. Assume a heavy vertex v is not matched in M. let P_v be a maximal alternative path starting from v. Since, there is no cycle in the graph, the other end of P_v is a leaf. If we replace the unmatched edges and matched edges in P_v, size of the matching remain the same, but the number of matched heavy vertices increases by one. This contradicts the selection of M. Thus, all of the heavy vertices in M are matched.

Let P be a maximal alternative path in T. We know that both ends of P are leaves. On the other hand, since M is a maximum matching, at least one end of P is matched. This means that, at least one leaf is matched in M which completes the proof. □

Lemma 2.5 shows that $\frac{h_F+1}{2}$ approximate the maximum matching size with a factor of 2. Thus, if we estimate the number of heavy vertices with a factor of $1+\epsilon$ we can estimate the maximum matching size with a factor of $2(1+\epsilon)$. Later, we use subroutine 2 to estimate the number of heavy vertices.

Lemma 2.6 generalize Lemma 2.5 to the forests with no isolated vertices.

Lemma 2.6. *Let F be a forest with no isolated vertex and with maximum matching size M^*. We have the following bounds:*

- *upper bound: $M^* \leq h_F + c_F$,*
- *lower bound: $M^* \geq \frac{h_F+c_F}{2}$,*
- *lower bound: $M^* \geq c_F$,*

where c_F is the number of connected components in F.

Proof. Let $T_1, T_2, \ldots, T_{c_F}$ be the connected components of F. Let M_i^* be the size of the maximum of T_i. For all $1 \leq i \leq c_F$ we know that $\frac{h_{T_i}+1}{2} \leq M_i^* \leq h_{T_i} + 1$. By summing it up over all i we have $\frac{h_F+c_F}{2} \leq M^* \leq h_F + c_F$. In addition, since we do not have any isolated vertex, each connected component contains at least one edge. This give us $M^* \geq c_F$ which completes the proof. □

The number of connected components in a forest is exactly $n - m$ where n is the number of vertices and m is the number of edges [32]. Thus, similar to the trees, if we estimate the number of heavy vertices within a $1+\epsilon$ factor we can estimate the maximum matching size within a $2(1+\epsilon)$ factor. In the following algorithm we reuse Subroutine 2 and estimate the maximum matching size of a forest with no isolated vertex.

Lemma 2.7. *For Subroutine 2 with parameters $c_2 = 0.5$, $\alpha_2 = 6\epsilon^{-2}$ and $\alpha_3 = 2$, we have:*

- *If $h \geq \sqrt{n}$, $\Pr\left[\left|\hat{h} - h\right| \geq \epsilon h\right] \leq \frac{1}{e}$; and*
- *If $h < \sqrt{n}$, $\Pr\left[\hat{h} > \alpha_3\sqrt{n}\right] \leq \frac{1}{e}$.*

Proof. Recall in the subroutine we select a random sample V' of size $n' = \alpha_2\sqrt{n}$. Let $V' = \{v'_1, \ldots, v'_n\}$. For $i \in [n']$, let X_i denote the random variable where $X_i = 1$ if v'_i is a heavy vertex, and $X_i = 0$ otherwise. Since we have h heavy vertices, we have that $\Pr[X_i = 1] = \frac{h}{n}$ for every $i \in [n']$. Let $h' = \sum_i X_i$. Clearly $\mathrm{E}[h'] = \frac{n' \cdot h}{n}$. Recall that the return value of Subroutine 2 is $\hat{h} = h' \times \frac{n}{n'}$. Thus in expectation, we return the value of h. Now since all X_i's are negatively correlated, we use the

Algorithm 2 Maximum Matching Estimator For Forest

```
Input: A stream of edges of the forest $e_1, \dots, e_m$.

Output: An estimation $M$ of the size of maximum matching $M^*$.

Initialization Process:

1: Set $\alpha_1 = 5$, $\alpha_2 = 6\epsilon^{-2}$, $\alpha_3 = 2$ and $Q = 21\lceil \ln(\frac{1}{\delta}) \rceil + 1$.
2: Set $c_1 = c_2 = 0.5$.
3: Initialize Subroutine 1 with $\alpha_1$ and $c_1$ factors.
4: Initialize $Q$ independent executions of Subroutines 2, with $\alpha_2$ and $c_2$ factors.

Update Process, upon the arrival of $e_i = (u, v)$:

1: Update each subroutine with the new arrival.

Termination Process:

1: Let $R$ denote the return value of Subroutine 1.
2: For $i \in [Q]$, let $\hat{h}_i$ denote the value returned by the $i$-th execution of Subroutine 2.
3: Let $h_{\text{med}}$ denote the median value of $\hat{h}_i$'s.
4: Let $n$ donate the number of vertices and $m$ donate the number of edges
5: if $R < \alpha_1\sqrt{n}$ then
6:     Output $M = R$.
7: else if $h_{\text{med}} \leq \alpha_3\sqrt{n}$ then
8:     Output $M = n - m$.
9: else
10:    Output $M = \frac{h_{\text{med}}+n-m}{2(1+\epsilon)}$.
11: end if
```

extension of Chernoff bound 2.3 to prove the first bound of the lemma for $h \geq \sqrt{n}$:

$$\begin{aligned}
&\Pr\left[\left|\hat{h} - h\right| \geq \epsilon h\right] = \Pr\left[\left|\hat{h} - \mathrm{E}\left[\hat{h}\right]\right| \geq \epsilon\, \mathrm{E}\left[\hat{h}\right]\right] \\
&= \Pr\left[\left|h' - \mathrm{E}\left[h'\right]\right| \geq \epsilon\, \mathrm{E}\left[h'\right]\right] \\
&\leq 2\exp\left(-\frac{\epsilon^2\, \mathrm{E}\left[h'\right]}{3}\right) \leq \frac{2}{e^2} \leq \frac{1}{e}.
\end{aligned}$$

We again use the Chernoff bound to prove the lemma for $h < \sqrt{n}$:

$$\begin{aligned}
&\Pr\left[\hat{h} > \alpha_3\sqrt{n}\right] \leq \Pr\left[\hat{h} > \left(\frac{\alpha_3\sqrt{n}}{h}\right) \mathrm{E}\left[\hat{h}\right]\right] \\
&= \Pr\left[h' > \left(\frac{\alpha_3\sqrt{n}}{h}\right) \mathrm{E}\left[h'\right]\right] \\
&\leq \Pr\left[h' > \left(1 + \frac{\alpha_3\sqrt{n}}{2h}\right) \mathrm{E}\left[h'\right]\right] \\
&\leq \exp\left(-\frac{\alpha_3^2 \cdot n \cdot \mathrm{E}\left[h'\right]}{12h^2}\right) \leq \exp\left(-\left(\frac{\alpha_3}{2}\right)^2\right) = \frac{1}{e}.
\end{aligned}$$

□

Proof of Theorem 2.2: When the maximum matching size is less than $\alpha_1\sqrt{n}$, we report the size of a maximal matching of F. This is clearly estimates the maximum matching size within a factor of 2. Thus, we can assume the maximum matching size is at least $\alpha_1\sqrt{n}$. First, we assume the following two claims and prove the theorem in three cases. Later, we provide the proofs of the claims.

Claim 2.3. *If $h < \sqrt{n}$, $\Pr\left[h_{\text{med}} > \alpha_3\sqrt{n}\right] \leq \delta$.*

Claim 2.4. *If $h \geq \sqrt{n}$, $\Pr\left[\left|h_{\text{med}} - h\right| \geq \epsilon h\right] \leq \delta$.*

Case 1: $\mathbf{h < \sqrt{n}}$. In this case using Claim 2.3, with probability $1 - \delta$ we have $h_{\text{med}} \leq \alpha_3\sqrt{n}$ and the output is $c_F = n - m$. Thus, with probability $1 - \delta$, we have

$$M^* \leq h_F + c_F \leq \sqrt{n} + c_F \leq \frac{M^*}{\alpha_1} + c_F.$$

This combined with $M^* \geq c_F$, says that the output approximate the maximum matching size within a $\frac{\alpha_1}{\alpha_1 - 1} = \frac{5}{4}$ factor.

Case 2: $\mathbf{h \geq \sqrt{n}}$ and $\mathbf{h_{med} < \alpha_3 \sqrt{n}}$. In this case using Claim 2.4, with probability $1 - \delta$ we have $h_F - \epsilon h_F \leq h_{\text{med}}$. This together with $h_{\text{med}} < \alpha_3 \sqrt{n}$ gives us $h_F - \epsilon h_F < \alpha_3 \sqrt{n}$. Therefore, with probability $1 - \delta$ we have

$$M^* \leq h_F + c_F \leq \frac{\alpha_3}{1 - \epsilon} \sqrt{n} + c_F \leq \frac{\alpha_3 M^*}{\alpha_1 (1 - \epsilon)} + c_F.$$

This combined with $M^* \geq c_F$, says that the output approximate the maximum matching size within a $\frac{\alpha_1(1-\epsilon)}{\alpha_1(1-\epsilon) - \alpha_3} \leq 2$ factor, assuming $\epsilon \leq 1/5$.

Case 3: $\mathbf{h \geq \sqrt{n}}$ and $\mathbf{h_{med} \geq \alpha_3 \sqrt{n}}$. In this case using Claim 2.4, with probability $1 - \delta$ we have $h_F - \epsilon h_F \leq h_{\text{med}}$. Therefore, with probability $1 - \delta$ we have

$$\begin{aligned} M^* \leq h_F + c_F &\leq \frac{h_{\text{med}}}{1 - \epsilon} + c_F \leq \frac{h_{\text{med}} + c_f}{1 - \epsilon} \\ &= \frac{h_{\text{med}} + c_f}{2(1 + \epsilon)} \frac{2(1 + \epsilon)}{1 - \epsilon} \leq \frac{h_{\text{med}} + c_f}{2(1 + \epsilon)} 2(1 + 3\epsilon). \end{aligned}$$

At the same time we have

$$M^* \geq \frac{h_F + c_F}{2} \geq \frac{h_{\text{med}}/(1 + \epsilon) + c_F}{2} \geq \frac{h_{\text{med}} + c_f}{2(1 + \epsilon)}.$$

This means that the output approximate the maximum matching size within a $2(1 + 3\epsilon)$ factor, assuming $\epsilon \leq 1/5$.

Proof of Claim 2.3: Let X_i donate the random variable where $X_i = 1$, if $\hat{h}_i > \alpha_3\sqrt{n}$ and $X_i = 0$ otherwise. Form Lemma 2.7 we know that the probability of $X_i = 1$ is $1/e$. On the other hand, if $h_{\text{med}} > \alpha_3\sqrt{n}$ at least half of the h_i's are greater than $\alpha_3\sqrt{n}$. Thus, we have $\Pr\left[h_{\text{med}} > \alpha_3\sqrt{n}\right] \leq \Pr\left[\sum_{i\in Q} X_i \geq 0.5|Q|\right]$. Using Chernoff bound we can bound this probability as follows.

$$\Pr\left[h_{\text{med}} > \alpha_3\sqrt{n}\right] \leq \exp\left(-\frac{(1-\frac{e}{2})^2\frac{1}{e}21\ln(\frac{1}{\delta})}{3}\right) \leq \delta.$$

■

Proof of Claim 2.3: Let X_i donate the random variable where $X_i = 1$, if $|h_i - h| \geq \epsilon h$ and $X_i = 0$ otherwise. Form Lemma 2.7 we know that the probability of $X_i = 1$ is $1/e$. Again here, if $|h_{\text{med}} - h| \geq \epsilon h$ at least half of the h_i's are greater than $\alpha_3\sqrt{n}$. Hence, again we have

$$\begin{aligned}\Pr\left[|h_{\text{med}} - h| \geq \epsilon h\right] &\leq \Pr\left[\sum_{i\in Q} X_i \geq 0.5|Q|\right]\\ &\leq \exp\left(-\frac{(1-\frac{e}{2})^2\frac{1}{e}21\ln(\frac{1}{\delta})}{3}\right) \leq \delta.\end{aligned}$$

■

■

2.5 2-Pass and Random Order Streaming Algorithms

Here we approximate the number of shallow edges within $(1 \pm \epsilon)$ factor using $\tilde{O}(\sqrt{n})$ space. Observe that in Lemma 2.2 we can replace $h \geq n^{2/3}$ by $h \geq n^{1/2}$ and the lemma works with constant probability. Therefore, we can approximate h using $\tilde{O}(\sqrt{n})$ space. The goal is to estimate s using $\tilde{O}(\sqrt{n})$ space as well.

Process 4. [Shallow Edges using 2 Passes]

Input: A stream of edges of the graph $e_1, \dots, e_m$.

Return Value: $\hat{s}$, an estimation for the number of shallow edges in the graph.

First Pass:

```
1: Let $S$ be a random sample set of edges chosen with probability $\frac{3\log(4/\delta)}{\epsilon^2\sqrt{n}}$.
```

Second Pass:

```
1: for each edge $(u, v) \in S$ do
2:     if $(u, v)$ is a shallow edge then
3:         Add $(u, v)$ to set $E'$ which is initialized to zero in the beginning of this pass.
4:     end if
5: end for
```

Termination Process:

```
1: Let $s' = |E'|$ denote the number of shallow edges sampled using our algorithm.
2: Return $\hat{s} = s' \times \frac{\epsilon^2\sqrt{n}}{3\log(4/\delta)}$.
```

2-Pass Streaming Algorithm When we are allowed to have two passes over the input, a natural approach for estimating s is to get a random sample of edges in the first round and then distinguish between shallow and heavy edges in the second round (see Subroutine 4).

Lemma 2.8. *For Subroutine 4, we have* $\Pr\left[(1-\epsilon)\cdot|S| \le \hat{s} \le (1+\epsilon)\cdot|S|\right] \ge 1 - \delta/2$.

Proof. Let $\mathcal{S}$ be the set of shallow edges. Suppose that $|S| \ge \sqrt{n}$. If $|S| \le \sqrt{n}$, we either maintain a maximal matching if the size of matching is $\tilde{O}(n^{1/2})$ or charge the number of shallow edges to the number of heavy vertices if the number of them is $\tilde{\Omega}(n^{1/2})$. Thus, we do not elaborate on them here. For $i \in [|S|]$, let X_i denote the random variable where $X_i = 1$ if $e_i \in S$, and $X_i = 0$ otherwise. We then have, $\Pr[X_i] = \frac{3\log(4/\delta)}{\epsilon^2\sqrt{n}}$. Let $s' = \sum_{i\in[|S|]} X_i$. Thus, $\mathrm{E}[s'] = \mathrm{E}\left[\sum_{i\in[|S|]} X_i\right] = \frac{3\log(4/\delta)\cdot|S|}{\epsilon^2\sqrt{n}} \ge \frac{3\log(4/\delta)}{\epsilon^2}$. We then use the Chernoff bound to prove that

$$\Pr\left[|s' - \mathrm{E}[s']| \ge \epsilon\cdot\mathrm{E}[s']\right]$$
$$\le 2\exp\left(-\frac{\epsilon^2\,\mathrm{E}[s']}{3}\right) \le \delta/2.$$

Let us condition on the event that $|s' - \mathrm{E}[s']| \le \epsilon\cdot\mathrm{E}[s']$ which happens with probability at least $1-\delta/2$. Since

$$(1-\epsilon)\cdot\mathrm{E}[s'] \times \frac{\epsilon^2\sqrt{n}}{3\log(4/\delta)}$$
$$\le \hat{s} \le (1+\epsilon)\cdot\mathrm{E}[s'] \times \frac{\epsilon^2\sqrt{n}}{3\log(4/\delta)},$$

we obtain $(1-\epsilon)\cdot|S| \le \hat{s} \le (1+\epsilon)\cdot|S|$. □

Process 5. [Shallow Edges in Random-Order Streams]

Input: A stream of edges of the graph $e_1, \ldots, e_m$.

Return Value: $\hat{s}$, an estimation for the number of shallow edges in the graph.

```
1: Let $S$ be the first $\frac{3\log(4/\delta)}{\epsilon^2\sqrt{n}}$ edges of the stream.
2: for each edge $(u, v) \in S$ do
3:     if $(u, v)$ is a shallow edge then
4:         Add $(u, v)$ to set $E'$ which is initialized to zero in the beginning of this pass.
5:     end if
6: end for
```

Termination Process:

```
1: Let $s' = |E'|$ denote the number of shallow edges sampled using our algorithm.
2: Return $\hat{s} = s' \times \frac{\epsilon^2\sqrt{n}}{3\log(4/\delta)}$.
```

Random Order Streams. Recall that in the 2-pass algorithm, we use the first pass simply to take a random sampling of the edges. Although in the random order model we cannot have two passes over the input, the first few edges of the input is in fact a random sample (see Subroutine 5). Thus the following lemma can be proved in a manner similar to Lemma 2.8.

Lemma 2.9. *For Subroutine 5, we have* $\Pr[(1-\epsilon)\cdot|S| \le \hat{s} \le (1+\epsilon)\cdot|S|] \ge 1 - \delta/2$.

2.6 Hardness Results

In this section, we show streaming lower bounds by reducing from the Boolean Hidden Matching Problem [33–35], which we refer to as BHM.

> BHM_n ($n \in \mathbb{Z}_+$): Alice is given a binary string $x \in \{0,1\}^{4n}$. Bob is given a *perfect matching* $Y = \{(y_1, y_1'), (y_2, y_2'), \ldots, (y_{2n}, y_{2n,2}')\}$ between numbers in $[4n]$ and a vector $z \in \{0,1\}^{2n}$. It is guaranteed that there is a $\theta \in \{0,1\}$ such that for for all $i \in [2n]$, it holds $x_{y_i} \oplus x_{y_i'} \oplus z_i = \theta$. Alice sends a single message to Bob and Bob's task is to output θ, the *parity of the matching.*

We use the following two communication lower bounds for BHM_n.

Fact 2.1 ([35, Theorem 2.1])**.** *Any randomized protocol for* BHM_n *with Alice's message of length* $o(n^{1/2})$ *errs with probability greater than* $1/4$ *on some input.*

Fact 2.2 ([33, Theorem 8])**.** *Any randomized protocol for* BHM_n *with Alice's message of length* $o(n)$ *errs with non-zero probability on some input.*

Fact 2.1 is used to show a weaker lower bound for randomized streaming algorithms and Fact 2.2 is used to show a stronger lower bound for deterministic streaming algorithms. In the proof, we turn a streaming algorithm into a communication protocol. In particular a small-space streaming algorithm would imply an efficient communication protocol. Hence, a communication lower bound for protocols implies a space lower bound for streaming algorithms.

Theorem 2.5. *Let $\mathcal{A}$ be a streaming algorithm whose goal is to estimate the maximum matching size to within a factor better than $3/2$. Let n be the number of vertices in the input graph.*

- *If $\mathcal{A}$ errs with probability at most $1/4$ on any input, then it cannot use $o(n^{1/2})$ bits of space.*
- *If $\mathcal{A}$ is deterministic and never errs, then it cannot use $o(n)$ bits of space.*

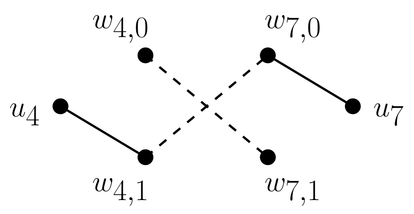
The lower bounds hold even for graphs consisting of paths of length up to 3.

Proof. Let $x \in \{0,1\}^{4n}$ be Alice's input in an instance of BHM_n. Let $Y = \{(y_1, y'_1), (y_2, y'_2), \ldots, (y_{2n}, y'_{2n,2})\}$ be a perfect matching between numbers in $[4n]$. Y and $z \in \{0,1\}^{2n}$ are Bob's input in the same instance of BHM_n.

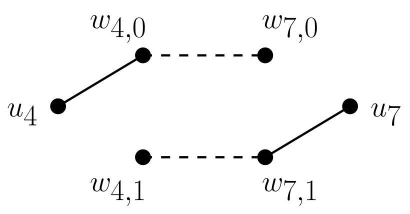
We construct a graph G on $4n$ groups of vertices. Each group consists of u_i, $w_{i,0}$, and $w_{i,1}$, where $i \in [4n]$. For each $i \in [4n]$, Alice holds only one edge of G: (u_i, w_{i,x_i}), which connects two vertices in the same group. For each edge (y_i, y'_i) of the matching Y, where $i \in [2n]$, Bob holds two edges of G: $(w_{y_i,0}, w_{y'_i,0\oplus z_i})$ and $(w_{y_i,1}, w_{y'_i,1\oplus z_i})$, which connect vertices in two groups. Note that both Alice and Bob can compute the edges they hold by only looking at their respective inputs.

We now show that the size of the matching significantly depends on the parity of the matching in the instance of BHM_n. We consider each pair of connected groups. It is easy to show that if the parity of the matching equals 0 (see Figure 2.1a for an example), then such pair consists of two disjoint paths of length 3 and 1, respectively. The size of the maximum matching in the pair of connected groups is then equal to 3, which implies that the maximum matching in the entire graph is of size $2n \cdot 3 = 6n$. If the parity of the matching equals 1 (as in Figure 2.1b), then the pair of connected groups consists of two edges of length 2 and has a maximum matching of size 2. In this case the maximum matching size in the entire graph G equals $2n \cdot 2 = 4n$. The multiplicative gap between these two cases equals $3/2$. Hence to distinguish them and to solve the BHM_n instance (i.e., to compute the parity of the matching), it suffices to estimate the maximum matching size in G to withing a factor better than $3/2$.

Suppose now that there is a streaming algorithm that uses $f(n)$ space for some function f and on any input estimates the maximum matching size within a factor better than $3/2$ with probability at least p on all inputs. Then BHM_n can be solved with probability at least p by sending a message consisting of $f(n)$ bits. First, Alice simulates the streaming algorithm on the set of her edges and passes the state of the algorithm to Bob. Bob continues the simulation on his set of edges and outputs his prediction of the parity of the matching bases on the estimate to the maximum matching size produced by the streaming algorithm as described above. More precisely, if the streaming algorithm produces an estimate within a factor better than $3/2$, the protocol correctly computes the parity of the matching in the



(a) The case of $x_4 = 1$, $x_7 = 0$, and the corresponding $z_i = 1$. The parity of the matching equals $x_4 \oplus x_7 \oplus z_i = 0$.



(b) The case of $x_4 = 0$, $x_7 = 1$, and the corresponding $z_i = 0$. The parity of the matching equals $x_4 \oplus x_7 \oplus z_i = 1$.

Figure 2.1: Examples of connected groups for different parities in the lower bound construction. Alice's edges and Bob's edges are drawn using solid and dashed lines, respectively.

BHM_n instance. By applying Fact 2.1, we obtain that there is no $o(n^{1/2})$-bits-of-space randomized streaming algorithm that provides this kind of guarantee with probability at least 3/4 for all inputs. Similarly, via Fact 2.2, we learn that there is no $o(n)$-bits-of-space deterministic streaming algorithm that always manages to compute a multiplicative 3/2-estimate to the maximum matching size. □

Chapter 3: Matching in Dynamic Streams

3.1 Introduction

Over the last decade, a growing body of work has considered solving graph problems in the data stream model. Most of the early work considered the insert-only variant of the model where the stream consists of edges being added to the graph and the goal is to compute properties of the graph using limited memory. Recently, however, there has been a significant amount of interest in being able to process dynamic graph streams where edges are both added and deleted from the graph [36–45]. These algorithms are all based on the surprising efficacy of using random linear projections, aka linear sketching, for solving combinatorial problems. Results include testing edge connectivity [36] and vertex connectivity [45], constructing sparsifiers [37–39], approximating the densest subgraph [43, 46, 47], correlation clustering [44], and estimating the number of triangles [42]. For a recent survey of the area, see [48].

The concept of *parameterized stream algorithms* was explored by Chitnis et al. [49] and Fafianie and Kratsch [50]. Their work investigated a natural connection between data streams and parameterized complexity. In parameterized complexity, the time cost of a problem is analyzed in terms of not only the input size but also other parameters of the input. For example, while the classic vertex cover problem is NP complete, it can be solved via a simple branching algorithm in time $2^k \cdot \text{poly}(n)$ where k is the size of the optimal vertex cover. An important concept in parameterized complexity is *kernelization* in which the goal is to efficiently transform an instance of a problem into a smaller instance such that the smaller instance is a "yes" instance (e.g., has a solution of at least a certain size) iff the original instance was also a "yes" instance. For more background on parameterized complexity and kernelization, see [51,52]. Parameterizing the *space* complexity of a problem in terms of the size of the output is a particularly appealing notion in the context of data stream computation. In particular, the space used by any algorithm that returns an actual solution (as opposed to an estimate of the size of the solution) is necessarily at least the size of the solution.

Our Results and Related Work. In this chapter we present a simple but powerful subgraph sampling primitive that is applicable in a variety of computational models including dynamic graph streams (where the input graph is defined by a sequence of edge/hyperedge insertions and deletions) and distributed systems such as MapReduce. This primitive will be useful for both those parameterized problems whose output has bounded size and for those where the optimal solution need not be bounded. In the case where the output has bounded size, our results can be thought of as *kernelization via sampling*, i.e., we sample a relatively small set of edges according to a simple (but not uniform) sampling procedure and can show that the resulting graph has a solution of size at most k iff the original graph has an optimal solution of size at most k. We present the subgraph sampling primitive and implementation details in Section 3.2.

Graph Matchings. Finding a large matching is the most well-studied graph problem in the data stream model [18, 22, 53–62]. However, all of the existing single-pass stream algorithms are restricted to the insert-only case, i.e., edges may be inserted but will never be deleted. This restriction is significant: for example, the simple greedy algorithm using $\tilde{O}(n)$ space returns a 2-approximation if there are no deletions. In contrast, prior to this work no $o(n)$-approximation was known in the dynamic case when there are both insertions and deletions. Finding an algorithm for the dynamic case of this fundamental graph problem was posed as an open problem in the Bertinoro Data Streams Open Problem List [63].

We prove the following results for computing a matching in the dynamic model. Our first result is an $\tilde{O}(k^2)$ space algorithm that returns the edges of a maximum matching on the assumption that its cardinality is at most k. Our algorithm has $\tilde{O}(1)$ update time. The best previous algorithm [49] collects $\max(\deg(u), 2k)$ edges incident to each vertex u and finds the optimal matching amongst these edges. This algorithm can be implemented in $\tilde{O}(kn)$ space where n is the number of vertices in the graph. Indeed obtaining an algorithm with $f(k)$ space, for any function f, in the dynamic graph stream case was left as an important open problem [49]. We can also extend our approach to maximum weighted matching. Our second result is an optimal $\tilde{O}(n^2/\alpha^3)$ space algorithm that returns an α-approximation for matchings of arbitrary size. For example, this implies an $n^{1/3}$ approximation using $\tilde{O}(n)$ space, commonly known as the *semi-streaming* space restriction [59, 64]. We present our second result and an algorithm for graphs with bounded arboricity, along with a discussion of very recent related work [65–67], in Section 3.5.

Vertex Cover and Hitting Set. We next consider the problem of finding the *minimum vertex cover* and its generalization, *minimum hitting set*. The hitting set problem can be defined in terms of hypergraphs: given a set of hyperedges, select the minimum set of vertices such that every hyperedge contains at least one of the selected vertices. If all hyperedges have cardinality two, this is the vertex cover problem.

There is a growing body of work analyzing hypergraphs in the data stream model [45, 68–72]. For example, Emek and Rosén [69] studied the following set-cover problem which is closely related to the hitting set problem: given a stream of hyperedges (without deletions), find the minimum subset of these hyperedges such that every vertex is included in at least one of the hyperedges. They present an $O(\sqrt{n})$ approximation streaming algorithm using $\tilde{O}(n)$ space along with results for covering all but a small fraction of the vertices. Another related problem is independent set since the minimum vertex cover is the complement of the maximum independent set. Halldórsson et al. [73] presented streaming algorithms for finding large independent sets but these do not imply a result for vertex cover in either the insert-only or dynamic setting.

In Section 3.3.2, we present a $\tilde{O}(k^d)$ space algorithm that finds the minimum hitting set where d is the cardinality of the input sets and k is an upper bound on the cardinality of the minimum hitting set. We prove the space use is optimal and matches the space used by previous algorithms in the insert-only model [49,50]. Our algorithms can be implemented with $\tilde{O}(1)$ update time. The only previous results in the dynamic model were by Chitnis et al. [49] and included a $\tilde{O}(kn)$ space algorithm and a $\tilde{O}(k^2)$ space algorithm under a much stronger "promise" that the vertex cover of the graph defined by any prefix of the stream may never exceed k. Relaxing this promise remained as the main open problem of Chitnis et al. [49]. In Section 3.3.2, we also generalize our exact matching result to hypergraphs. In Section 3.7, we show our result is also optimal.

General Family of Results. We consider a larger family of parameterized problems for which our subgraph sampling primitive yields fast, small-space dynamic graph stream algorithms. This result is presented in Section 3.4, while lower bounds for various problems outside this family are proved in Section 3.7.

3.2 Basic Subgraph Sampling Technique

Basic Approach and Intuition. The inspiration for our subgraph sampling primitive is the following simple procedure for *edge* sampling. Given a graph $G = (V, E)$ and probability $p \in [0, 1]$, let $\mu_{G,p}$ be the distribution $E \cup \{\perp\}$ defined by the following process:

1. Sample each vertex independently with probability p and let V' denote the set of sampled vertices.
2. Return an edge chosen uniformly at random from the edges in the induced

graph on V'. If no such edge exists, return $\perp$.

The distribution $\mu_{G,p}$ has some surprisingly useful properties. For example, suppose that the optimal matching in a graph G has size at most k. It is possible to show that this matching has the same size as the optimal matching in the graph formed by taking $O(k^2)$ independent samples from $\mu_{G,1/k}$. It is not hard to show that such a result would not hold if the edges were sampled uniformly at random.[1] The intuition is that when we sample from $\mu_{G,p}$ we are less likely to sample an edge incident to a high degree vertex then if we sampled uniformly at random from the edge set. For a large family of problems including matching, it will be advantageous to avoid bias towards edges whose endpoints have high degree.

Our subgraph sampling primitive essentially parallelizes the process of sampling from $\mu_{G,p}$. This will lead to more efficient algorithms in the dynamic graph stream model. The basic idea is rather than select a subset of vertices V', we randomly partition V into $V_1 \cup V_2 \cup \ldots \cup V_{1/p}$. Selecting a random edge from the induced graph on any V_i results in an edge distributed as in $\mu_{G,p}$. Sampling an edge on each V_i results in $1/p$ samples from $\mu_{G,p}$ although note that the samples are no longer independent. This lack of independence will not be an issue and will sometimes

[1]To see this, consider a layered graph on vertices $L_1 \cup L_2 \cup L_3 \cup L_4$ with edges forming a complete bipartite graph on $L_1 \times L_2$, a complete bipartite matching on $L_2 \times L_3$, and a perfect matching on $L_3 \times L_4$. If $|L_1| = n \gg k$ and $|L_2| = |L_3| = |L_4| = k/2$ then the maximum matching has size k and every matching includes all edges in the perfect matching on $L_3 \times L_4$. Since there are $\Omega(nk)$ edges in this graph we would need $\Omega(nk)$ edges sampled uniformly before we find the matching on $L_3 \times L_4$.

be to our advantage. In many applications it will make sense to parallelize the sampling further and select a random edge between each pair, V_i and V_j, of vertex subsets. For applications involving hypergraphs we select random edges between larger subsets of $\{V_1, V_2, \ldots, V_{1/p}\}$.

Sampling Data Structure: We now present the subgraph sampling primitive formally. Given an unweighted (hyper)graph $G = (V, E)$, consider a "coloring" defined by a function $c : V \to [b]$. It will be convenient to introduce the notation: for each $i \in [b]$

$$V_i = \{v \in V : c(v) = i\}$$

and say that every vertex in V_i has color i. For a (hyper)edge $e \in E$, we define $c(e) = \{i \ : \ v \in e, c(v) = i\}$, i.e., $c(e)$ is exactly the set of colors seen on the vertices of e. For $S \subseteq [b]$, we say that an (hyper)edge e of G is *S-colored* if $c(e) = S$, i.e., each color from S is used to color the vertices in e and no other colors are used. Given a constant $q \geq 1$ which denotes the "size restriction", for each $S \subseteq [b]$ of cardinality at most q, E_S contains a single edge chosen uniformly at random from the set of all S-colored edges. If there are no S-colored edges, then $E_S = \emptyset$. The union of these sets defines the random graph $G' = (V, E')$, i.e.,

$$E' = \bigcup_{S \subseteq [b], |S| \leq q} E_S \ .$$

For example, given a simple graph, if we have $q = 1$ then for each color $i \in [b]$ we choose an edge whose endpoints are both colored i. If $q = 2$, then for every $1 \leq i \leq j \leq b$ we choose an edge whose one endpoint has color i and the other

Algorithm 3 Algorithm for Sampling Subgraphs According to $\mathsf{Sample}_{b,q,r}$

Input: A (hyper)graph $G = (V, E)$ and natural numbers b, q, r.

Output: A subgraph $G' = (V, E')$ where $E' \subseteq E$

```
1: Choose $c_1, \ldots, c_r$ u.a.r. from a family of pairwise independent hash functions
   mapping $V$ to $[b]$
2: Set $E' = \emptyset$
3: for $1 \leq j \leq r$ do
4:     for each $S \subseteq [b]$ such that $|S| \leq q$ do
5:         Select an edge $E_S^j$ u.a.r. from the set of $S$-colored edges $\{e \in E : \cup_{v \in e} c_j(v) = S\}$
6:         $E' \leftarrow E' \cup E_S^j$
7:     end for
8: end for
9: Report the graph $G' = (V, E')$.
```

endpoint has color j: note that this includes the possibility that $i = j$. In the case of a weighted graph, for each distinct weight w we choose a single edge $E_{S,w}$ uniformly at random from the set of S-colored edges with weight w.

Definition 3.1. *We define* $\mathsf{Sample}_{b,q,1}$ *to be the distribution over subgraphs generated as above where c is chosen uniformly at random from a family of pairwise independent hash functions.* $\mathsf{Sample}_{b,q,r}$ *is the distribution over graphs formed by taking the union of r independent graphs sampled from* $\mathsf{Sample}_{b,q,1}$*. Algorithm 3 gives pseudocode for sampling from* $\mathsf{Sample}_{b,q,r}$*.*

Motivating Application. As a first application to motivate the subgraph sampling primitive we again consider the problem of estimating matchings. We will use the following simple lemma that will also be useful in subsequent sections.

Lemma 3.1. *Let $U \subseteq V$ be an arbitrary subset of $|U| = r$ vertices and let $c : V \to [4r\epsilon^{-1}]$ be a pairwise independent hash function. Then with probability at least $3/4$, at least $(1-\epsilon)r$ of the vertices in U are hashed to distinct values. Setting $\epsilon < 1/r$ ensures all vertices are hashed to distinct values with this probability.*

Proof. Let $b = 4\epsilon r$. For a vertex $u \in U$, let I_u be the indicator random variable that equals one if there exists $u' \in U \setminus \{u\}$ such that $c(u) = c(u')$. Since c is pairwise independent,

$$\mathbb{P}\left[I_u\right] \leq \sum_{u' \in U \setminus \{u\}} \mathbb{P}\left[c(u) = c(u')\right] = \sum_{u' \in U \setminus \{u\}} 1/b < r/b = \epsilon/4 \ .$$

Let $I = \sum_{u \in U} I_u$ and note that $\mathbb{E}\left[I\right] \leq \epsilon r/4$. Then Markov's inequality implies $\mathbb{P}\left[I \geq \epsilon r\right] \leq 1/4$. □

Suppose G is a graph with a matching $M = \{e_1, \ldots, e_k\}$ of size k. Let $G' \sim \mathsf{Sample}_{b,2,1}$. By the above lemma, there exists $b = O(k^2)$, such that all the $2k$ endpoints of edges in M are colored differently with constant probability. Suppose the endpoints of edge e_i received the colors a_i and b_i. Then G' contains an edge in $E_{\{a_i,b_i\}}$ for each $i \in [k]$. Assuming all endpoints receive different colors, no edge in $E_{\{a_i,b_i\}}$ shares an endpoint with an edge in $E_{\{a_j,b_j\}}$ for $j \neq i$. Hence, we can conclude that G' also has a matching of size k. In Section 3.4, we show that a similar approach can be generalized to a range of problems. Using a similar argument there exists

$b = O(k)$ such that G' contains a constant approximation to the optimum matching. However, in Section 7.3, we show that there exists $b = O(k)$ such that with high probability graphs sampled from $\mathsf{Sample}_{b,2,O(\log k)}$ preserve the size of the optimal matching exactly.

3.2.1 Application to Dynamic Data Streams and MapReduce

We now describe how the subgraph sampling primitive can be implemented in various computational models.

Dynamic Graph Streams. Let S be a stream of insertions and deletions of edges of an underlying graph $G(V, E)$. We assume that vertex set $V = \{1, 2, \ldots, n\}$. We assume that the length of stream is polynomially related to n and hence $O(\log |S|) = O(\log n)$. We denote an undirected edge in E with two endpoints $u, v \in V$ by uv. For weighted graphs, we assume that the weight of an edge is specified when the edge is inserted and deleted and that the weight never changes. The following theorem establishes that the sampling primitive can be efficiently implemented in dynamic graph streams.

Theorem 3.1. *Suppose G is a graph with w_0 distinct weights. It is possible to sample from $\mathsf{Sample}_{b,q,r}$ with probability at least $1 - \delta$ in the dynamic graph stream model using $\tilde{O}(b^q r w_0)$ space and $\tilde{O}(r)$ update time.*

Proof. To sample a graph from $\mathsf{Sample}_{b,q,r}$ we simply sample r graphs from $\mathsf{Sample}_{b,q,1}$ in parallel. To draw a sample from $\mathsf{Sample}_{b,q,1}$, we employ one instance of an ℓ_0-sampling primitive for each of the $O(b^q)$ edge colorings [74, 75]: Given a dynamic graph stream, the behavior of an ℓ_0-sampler algorithm is defined as follows: It returns FAIL with probability at most δ and otherwise, it returns an edge chosen uniformly at random amongst the edges that have been inserted and not deleted. If there are no such edges, the ℓ_0-sampler returns NULL. The ℓ_0-sampling primitive can be implemented using $O(\log^2 n \log \delta^{-1})$ bits of space and $O(\operatorname{polylog} n)$ update time. In some cases, we can make use of simpler deterministic data structures. For Theorem 3.2, we can replace the ℓ_0 sampler with a counter and the exclusive-or of all the edge identifiers, since we only require to recover edges when they are unique within their color class. For Theorem 3.5, we only require a counter. In both cases, the space cost is reduced to $O(\log n)$.

At the start of the stream we choose a pairwise independent hash function $c : V \to [b]$. For each weight w and subset $S \subseteq [b]$ of size q, this hash function defines a sub-stream corresponding to the S-colored edges of weight w. We then use ℓ_0-sampling on each sub-stream to select a random edge from E_S. $\square$

MapReduce and Distributed Models. The sampling distribution is naturally parallel, making it straightforward to implement in a variety of popular models. In MapReduce, the r hash functions can be shared state among all machines, allowing Map function to output each edge keyed by its color under each hash function. Then, these can be sampled from on the Reduce side to generate the graph G'. Optimizations can do some data reduction on the Map side, so that only one edge per color class is emitted, reducing the communication cost. A similar outline holds for other parallel graph models such as Pregel.

3.3 Parameterized Matching, Vertex Cover, and Hitting Set

3.3.1 Finding Maximum Matchings and Minimum Vertex Covers Exactly

In this section, we present results on finding edges in a maximum matching and the vertices in a minimum vertex cover of a graph G. We use $\mathrm{match}(G)$ to denote the size of the maximum (weighted or unweighted as appropriate) matching in G and use $\mathrm{vc}(G)$ to denote the size of minimum vertex cover. The main theorem we prove in this section is that a set of edges is a maximum matching in the sampled graph iff it is a maximum matching in the original graph:

Theorem 3.2 (Finding Exact Solutions). *Suppose* $\mathrm{match}(G) \leq k$. *Then, with probability* $1 - 1/\mathrm{poly}(k)$,

$$\mathrm{match}(G') = \mathrm{match}(G) \quad \textit{and} \quad \mathrm{vc}(G') = \mathrm{vc}(G) \ ,$$

where $G' = (V, E') \sim \mathsf{Sample}_{1000k,2,O(\log k)}$.

Intuition and Preliminaries. To argue that G' has a matching of the optimal size, it suffices to show that for every edge $uv \in G$ that is not in G', there are a large number of edges incident to one or both of u and v that is in G'. If this is the case, then it will still be possible to match at least one of these vertices in G'.

To make this precise, let U be the subset of vertices with degree at least $10k$. Let F be the set of edges in the induced subgraph on $V \setminus U$, i.e., the set of edges whose endpoints both have small degree. We will prove that with high probability,

$$(F \subseteq E') \qquad \text{and} \qquad (\forall u \in U\ ,\ \ \deg_{G'}(u) \geq 5k)\ , \tag{3.1}$$

where E' is the set of edges in G'. Note that any sampled graph G' that satisfies (3.1) has the property that for all edges $uv \in G$ that are not in G' we have $\deg_{G'}(u) \geq 5k$ or $\deg_{G'}(v) \geq 5k$.

Analysis. The first lemma establishes that it is sufficient to prove that (3.1) holds with high probability.

Lemma 3.2. *If* $\text{match}(G) \leq k$ *then* (3.1) *implies* $\text{match}(G') = \text{match}(G)$ *and* $\text{vc}(G') = \text{vc}(G)$*.*

Proof. We first argue that $\text{vc}(G') = \text{vc}(G)$. Since the vertex cover of G is of size at most $2k$, every vertex in U must be in the vertex cover of both G and G' since the degrees of such vertices in both graphs are strictly greater than $2k$. This follows because if a vertex in U was not in the minimum vertex cover then all its neighbors need to be in the vertex cover.

We next argue that $\text{match}(G') = \text{match}(G)$. If property (3.1) is satisfied then G' contains a matching of size $\text{match}(F) + |U| \geq \text{match}(G)$ since we may choose the optimum matching in F and then still be able to match every vertex in U. This follows because the optimum matching in F "consumes" at most $2k$ potential endpoints, since $\text{match}(G) \leq k$. Hence, each of the (at most $2k$) vertices in U can still be matched to $3k$ possible vertices. □

The next lemma establishes that (3.1) holds with the required probability.

Lemma 3.3. *Property* (3.1) *holds with probability at least* $1 - 1/\operatorname{poly}(k)$.

Proof. Let $\mathrm{VC}(G)$ be a minimum vertex cover of G. Note that $\mathrm{match}(G) \leq k$ implies that $\mathrm{vc}(G) = |\mathrm{VC}(G)| \leq 2k$ because the endpoints of the edges in a maximum matching form a vertex cover. Next consider $H \sim \mathsf{Sample}_{1000k,2,1}$. We will show that for any $e \in F$ and $u \in U$,

$$\mathbb{P}\left[e \in H\right] > 1/2 \quad \text{and} \quad \mathbb{P}\left[\deg_H(u) \geq 5k\right] \geq 1/2 \ .$$

It follows that if $r = O(\log k)$ and $G' \sim \mathsf{Sample}_{1000k,2,r}$ then

$$\mathbb{P}\left[e \in G' \text{ and } \deg_{G'}(u) \geq 5k\right] \geq 1 - 1/\operatorname{poly}(k) \ .$$

We then take the union bound over the $O(k^2)$ edges in F and the $O(k)$ vertices in U. The fact that $|F| = O(k^2)$ and $|U| = O(k)$ follows from the promises $\mathrm{match}(G) \leq k$ and $\mathrm{vc}(G) \leq 2k$. In particular, the induced graph on $V \setminus U$ has a matching of size $\Omega(|F|/k)$ since the maximum degree is $O(k)$ and this size is at most k. Since all vertices in U must be in the minimum vertex cover, $|U| \leq 2k$.

To prove $\mathbb{P}\left[e \in H\right] \geq 1/2$. Let the endpoints of e be x and y. We define a set of vertices A such that e is the unique edge that remains if all vertices in A are removed from the graph: $A = (\mathrm{VC}(G) \cup \Gamma(x) \cup \Gamma(y)) \setminus \{x, y\}$, where $\Gamma(\cdot)$ denotes the set of neighbors of a vertex. The removal of $\mathrm{VC}(G) \setminus \{x, y\}$ ensures all remaining edges are incident to either x or y. The subsequent removal of $(\Gamma(x) \cup \Gamma(y)) \setminus \{x, y\}$ ensures the unique remaining edge is xy as claimed.

Consider the hash function $c : [n] \to [b]$ that defined H where $b = 1000k$. Observe that if all the vertices in A receive colors that are different than $c(x)$ and

$c(y)$, then xy is the unique $\{c(x), c(y)\}$-colored edge and hence is definitely sampled. Since $b = 1000k$ and $|A| \leq 2k + 10k + 10k = 22k$,

$$\begin{aligned}\mathbb{P}[xy \in H] &\geq 1 - \mathbb{P}[\exists a \in A : c(a) = c(x)] - \mathbb{P}[\exists a \in A : c(a) = c(y)] \\ &\geq 1 - 2|A|/b > 1/2 \ .\end{aligned}$$

To prove $\mathbb{P}[\deg_H(u) \geq 5k] \geq 1/2$. Let N_u be an arbitrary set of $10k$ neighbors of u and $A = \text{VC}(G) \setminus \{u\}$. If $c(u) \notin c(A)$ and there exist different colors $c_1, \ldots, c_{5k}$ such that each $c_i \in c(N_u) \setminus c(A)$ then the algorithm returns at least $5k$ edges incident to u in H. This follows since every edge not incident to u has at least one vertex in A. Hence, every $\{c_i, c(u)\}$-colored edge is incident to u and is distinct from every $\{c_j, c(u)\}$-colored edge.

Observe that $\mathbb{P}[c(u) \in c(A)] \leq 2k/b$. By appealing to Lemma 3.1, with probability at least $3/4$, there are at least $6k$ colors used to color the vertices N_u. Of these colors, at least $5k$ are colored differently from vertices in A. Hence we find $5k$ edges incident to u with probability at least $3/4 - 2k/b \geq 1/2$. $\square$

Extension to Weighted Matching. We now extend the result of the previous section to the weighted case. The following lemma shows that it is possible to remove an edge uv from a graph without changing the weight of the maximum weighted matching, if u and v satisfy certain properties.

Lemma 3.4. *Let $G = (V, E)$ be a weighted graph and let $G' = (V, E')$ be a subgraph with the property:*

$$\forall uv \in E \setminus E' \ , \ \ \deg_{G'}^{w(uv)}(u) \geq 5k \quad or \quad \deg_{G'}^{w(uv)}(v) \geq 5k \ ,$$

where $\deg_G^w(u)$ *is the number of edges incident to* u *in* G *with weight* w*. Then,* $\mathrm{match}(G) = \mathrm{match}(G')$.

Proof. Let $E \setminus E' = \{e_1, e_2, \ldots e_t\}$ and let G'_i be the graph formed by removing $\{e_1, \ldots, e_i\}$ from G. So $G'_0 = G$ and $G'_t = G'$. For the sake of contradiction, suppose $\mathrm{match}(G) > \mathrm{match}(G')$ and let r be the minimal value such that $\mathrm{match}(G) > \mathrm{match}(G'_r)$.

By the minimality of r, $\mathrm{match}(G) = \mathrm{match}(G'_{r-1})$. Consider the maximum weight matching M in G'_{r-1}. If $e_r \notin M$ then $\mathrm{match}(G) = \mathrm{match}(G'_{r-1}) = \mathrm{match}(G'_r)$ and we have a contradiction. If $e_r \in M$, let u, v be the endpoints of e_r and the weight of e_r be w. Without loss of generality $\deg^w_{G'_r}(u) \geq d^w_{G'}(u) \geq 5k$. Hence, there exists edge ux of weight w in G'_r where x is not an endpoint in M. Therefore, the matching $(M \setminus \{e_r\}) \cup \{ux\}$ is contained in G'_r and has the same weight as M. Hence, $\mathrm{match}(G) = \mathrm{match}(G'_{r-1}) = \mathrm{match}(G'_r)$ and we again have a contradiction.

□

Proof Proof of Lemma 3.7: Consider the size of minimum hitting set of $M_{C,D}$. If $\mathrm{hs}(M_{C,D}) > s_G(C)d$, then $M_{C,D}$ has a matching of size greater than $s_G(C)$. This matching together with the set C forms a sunflower with core C and over $s_G(C)$ petals, which contradicts the assumption. Therefore, $\mathrm{hs}(M_{C,D}) \leq s_G(C)d$ as claimed. □

Consider a weighted graph G and let $G' \sim \mathsf{Sample}_{1000k,2,O(\log k)}$. For each weight w, let G_w and G'_w denote the subgraphs consisting of edges with weight exactly w. By applying the analysis of the previous section to G_w and G'_w we may conclude that G' satisfies the properties of the above lemma. Hence, $\text{match}(G) = \text{match}(G')$. To reduce the dependence on the number of distinct weights in Theorem 3.1, we may first round each weight to the nearest power of $(1+\epsilon)$ at the cost of incurring a $(1+\epsilon)$ factor error. If W is the ratio of the max weight to min weight, there are $O(\epsilon^{-1}\log W)$ distinct weights after the rounding.

3.3.2 Finding Minimum Hitting Set Exactly

In this section we present exact results for hitting set and hypergraph matching. Let $\text{hs}(G)$ denote the cardinality of the minimum hitting set of G. Throughout the section, let G be a hypergraph where each edge has size exactly d and $\text{hs}(G) \leq k$. Throughout this section we assume d is a constant.

Intuition and Preliminaries. Given that the hitting set problem is a generalization of the vertex cover problem, naturally some of the ideas in this section build upon ideas from the previous section. However, the combinatorial structure we need to analyze for our sampling result goes beyond what is typically needed when extending vertex cover kernalization results to hitting sets. We first need to review a basic definition and result about "sunflower" set systems.

Lemma 3.5 (Sunflower Lemma [76])**.** *Let $\mathcal{F}$ be a collection of subsets of $[n]$. Then $A_1, \ldots, A_s \in \mathcal{F}$ is an s-*sunflower *if $A_i \cap A_j = C$ for all $1 \leq i < j \leq s$. We refer to C as the* core *of the sunflower and $A_i \setminus C$ as the* petals*. If each set in $\mathcal{F}$ has size at most d and $|\mathcal{F}| > d!k^d$, then $\mathcal{F}$ contains a $(k+1)$-sunflower.*

Let $s_G(C)$ denote the number of petals in a maximum sunflower in the graph

G with core C. We say a core is *large* if $s_G(C) > ak$ for some large constant a and *significant* if $s_G(C) > k$. Define the sets:

- $U = \{C \subseteq V \mid s_G(C) > ak\}$ is the set of large cores.
- $F = \{D \in E \mid \forall C \in U, C \nsubseteq D\}$ is the set of edges that do not include a large core.
- $U' = \{C \in U \mid \forall C' \subset C, s_G(C') \leq k\}$ is the set of large cores that do not contain significant cores.

The sets U and F play a similar role to the sets of the same name in the previous section. For example, if $d = 2$, then a large core corresponds to a high degree vertex. However, the set U' has no corresponding notion when $d = 2$ because a high degree vertex cannot contain another high degree vertex.

Lemma 3.6. *$|F| = O(k^d)$ and $|U'| = O(k^{d-1})$*

Proof. For the sake of contradiction assume $|F| > d!(ak)^d$. Then, by the Sunflower Lemma, F contains a $(ak + 1)$-sunflower. If the core of this sunflower is empty, F has a matching of size $(ak + 1)$ and therefore cannot have a hitting set of size at most k. If the sunflower has a non-empty core C, then some edge $D \in F$ contains C, which contradicts the definition of F. Therefore, $|F| \leq d!(ak)^d$.

To prove $|U'| \leq (d - 1)!k^{d-1}$, first note that $|C'| \leq d - 1$ for all $C' \in U'$. For the sake of contradiction assume that $|U'| > (d - 1)!k^{d-1}$. Then, by the Sunflower Lemma again, U' contains a $(k + 1)$-sunflower. Note that it is a sunflower of cores, not hypergraph edges. Let $C_1, C_2, \ldots, C_{k+1}$ be the sets in the sunflower. Each of these sets has to contain at least one vertex of the minimum hitting set. Therefore, if $C_1, C_2, \ldots, C_{k+1}$ are disjoint (i.e., the core of the sunflower is empty), U' has a matching of size $(k+1)$ and cannot have a hitting set of size at most k. If the sunflower has a non-empty core C^*, we will show that union of the maximum sunflowers with cores $C_1, C_2, \ldots, C_{k+1}$ contains a sunflower with $k+1$ edges with core $C^* \subset C_1 \in U'$. This contradicts the definition of U' and therefore $|U'| \leq (d-1)!k^{d-1} = O(k^{d-1})$. To construct the sunflower on C^*, for $i = 1, \ldots, k+1$, we pick an edge D_i in the maximum sunflower with core C_i such that $D_i \cap C_j = C^*$ for $j \neq i$ and $D_i \cap D_j = C^*$ for $j < i$. This is possible if a is sufficiently large.

□

The following (rather technical) lemma will play a crucial role when dealing with cores that are subsets of other cores in U' or of edges in F. It shows that if a core C is contained in a set D, then the set of edges that intersect D at C has a hitting set that a) does not include vertices in C and b) has small size assuming $s_G(C)$ is small.

Lemma 3.7. *For any two sets of vertices C, D, where $C \subseteq D$, define*

$$M_{C,D} = \{D' \setminus C \mid D' \in E, D \cap D' = C\} .$$

Then $\text{hs}(M_{C,D}) \leq s_G(C)d$. See Figure 3.1 for an example.

Hitting Set. For the rest of this section we let $G' = (V, E') \sim \mathsf{Sample}_{b,d,r}(G)$ where $b = O(k)$, d is the cardinality of the hyperedges, and $r = O(\log k)$. It will also be convenient to use the notation $\text{HS}(\mathcal{S})$ to denote a minimum hitting set of a collection

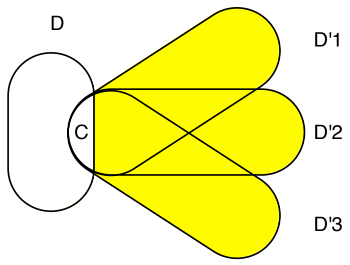


Figure 3.1: Given sets D'_1, D'_2, D'_3 that intersect set D exactly at C then $M_{C,D}$ consists of the shaded subsets of D'_1, D'_2, and D'_3. Assuming C is non-empty, $\{D'_1, D'_2, D'_3\}$ has a hitting set of size 1 since any vertex in C hits all sets. Lemma 3.7 bounds the size of the minimum hitting set of $\{D'_1, D'_2, D'_3\}$ that may not include any vertices in C.

of sets $\mathcal{S}$, i.e., $\text{hs}(\mathcal{S}) = |\operatorname{HS}(\mathcal{S})|$.

Theorem 3.3. *Suppose* $\text{hs}(G) \leq k$. *With probability* $1 - 1/\operatorname{poly}(k)$, $\text{hs}(G') = \text{hs}(G)$.

Proof. For each significant core C there has to be at least one vertex from the hitting set in C. Since all large cores are significant, $\text{hs}(G) = \text{hs}(U \cup F)$. If $C \in U$ has a subset C' such that $s_G(C') > k$, then there is at least one vertex from the hitting set in C' and this vertex also hits C. Thus, $\text{hs}(G) = \text{hs}(U' \cup F)$. By Lemma 3.8, the set of significant cores in G' is a superset of U' with high probability. By Lemma 3.9, every edge in F is in G' with high probability. □

Lemma 3.8. $\mathbb{P}\left[s_{G'}(C) > k \textit{ for all } C \in U'\right] \geq 1 - 1/\operatorname{poly}(k)$.

Proof. Fix an arbitrary core $C \in U'$. Consider $H \sim \mathsf{Sample}_{b,d,1}$ and let $c : [n] \to [b]$ be the coloring that defined H. We need to identify $k+1$ sets of colors

$S_1, S_2, \ldots S_{k+1} \subset [b]$ each of size d, such that any set of edges $D_1, D_2, \ldots, D_{k+1}$ where D_i is S_i-colored forms a sunflower of size $k+1$ on core C. In order for this to hold, the color sets have to satisfy the following three properties:

1. All edges that are S_i-colored contain C.

2. There is at least one S_i-colored edge.

3. If D is S_i-colored and D' is S_j-colored then $(D \setminus C) \cap (D' \setminus C) = \emptyset$.

In what follows, we first define a set $\mathcal{F} = \{S_1, S_2, \ldots\}$ that satisfies the above properties. We then argue that $|\mathcal{F}| \geq k+1$ with probability at least 1/2. By repeating the process $O(\log k)$ times will ensure that such a family exists with high probability. The lemma follows by taking the union bound over all $C \in U'$ since $|U'| = O(k^{d-1})$ by Lemma 3.6.

Property 1. We first define a set of vertices A such that any edge that does not intersect A must include C. Then, for any $S \subset [b]$ that is disjoint from $c(A)$, we may infer that all S-colored edges contain C. This follows since if $S = c(D)$ for some edge D, then $c(D) \cap c(A) = \emptyset$ which implies that $D \cap A = \emptyset$, and so $C \subseteq D$. Let

$$A = (\mathrm{HS}(G) \setminus C) \cup \left(\bigcup_{C' \subset C} \mathrm{HS}(M_{C',C}) \right) .$$

All edges that do not intersect $\mathrm{HS}(G) \setminus C$ must intersect with C. But all edges that intersect with only a subset of C, say C', must intersect with $\mathrm{HS}(M_{C',C})$. Hence A has the claimed property. We will say that C is a *good core* if $c(C) \cap c(A) = \emptyset$ and $|c(C)| = |C|$.

Property 2. Next, let $\mathcal{P}$ be a set of petals in a sunflower with core C that do not intersect with A. We may choose a set of $|\mathcal{P}| = ak - |A|$ such petals. For each $P \in \mathcal{P}$, define the set:

$$A_P = A \cup C \cup \left(\bigcup_{Q \in \mathcal{P} \setminus P} Q \right) .$$

If C is a good core, let $\mathcal{P}'$ contain all $P \in \mathcal{P}$ such that $c(P) \cap c(A_P) = \emptyset$ and $|c(P)| = |P|$. If C is not a good core, let $\mathcal{P}' = \emptyset$. Then the family $\mathcal{F} = \{c(P \cup C)\}_{P \in \mathcal{P}'}$ satisfies Properties 1 and 2. Note that no two petals in $\mathcal{P}'$ share the same color and hence $|\mathcal{F}| = |\mathcal{P}'|$ assuming C is a good core.

Property 3. Assume C is a good core since otherwise $\mathcal{F} = \emptyset$ and Property 3 is trivially satisified. Let $S_1, S_2 \in \mathcal{F}$ and suppose $S_1 = c(C \cup P_1)$ and $S_2 = c(C \cup P_2)$ for some $P_1, P_2 \in \mathcal{P}'$. Suppse edges $C \cup Q_1$ and $C \cup Q_2$ are S_1-colored and S_2-colored respectively. Then $c(Q_1) = c(P_1)$ and $c(Q_2) = c(P_2)$ because $|c(C)| = |C|$, $|c(P_1)| = |P_1|$, $|c(P_2)| = |P_2|$, and all edges have the same cardinality. But $c(P_1) \cap c(P_2) = \emptyset$ implies $c(Q_1) \cap c(Q_2) = \emptyset$ and so $Q_1 \cap Q_2 = \emptyset$ as required.

Size of $\mathcal{F}$. We need to show that $|\mathcal{P}'| \geq (k+1)$ with probability 1/2. Recall that $c : V \to [b]$ is chosen randomly from a family of pairwise independent hash functions and suppose $b = 8\max(d|A| + d^2, d|A_P| + d^2)$. Note that $b = O(k)$ since, by appealing to Lemma 3.7,

$$|A| \leq |A_P| \leq |A| + |C| + d|\mathcal{P}| \leq \text{hs}(G) + \sum_{C' \subset C} \text{hs}(M_{C,C'}) + |C| + d|\mathcal{P}|$$

$$\leq k + 2^d dk + d + dak = O(k).$$

Then, $\mathbb{P}[C \text{ is not a good core}] = \mathbb{P}[c(C) \cap c(A) \neq \emptyset \text{ or } |c(C)| \neq |C|] \leq (d|A| + d^2)/b \leq 1/8$. For each $P \in \mathcal{P}$, let $X_P = 1$ if $P \notin \mathcal{P}'$ or C is not a good core. Let $X_P = 0$ otherwise. Then

$$\mathbb{E}\left[\sum X_P\right] \leq |\mathcal{P}| \left(d|A_P| + d^2)/b + 1/8\right) \leq |\mathcal{P}|/4 ,$$

and so $\mathbb{P}[\sum X_P \geq |\mathcal{P}|/2] \leq 1/2$ by the Markov inequality. Hence, $|\mathcal{P}'| = |\mathcal{P}| - \sum X_P \geq |\mathcal{P}|/2 = ak/2 - |A|/2 \geq k+1$ for sufficiently large a with probability at least $1/2$.

$\square$

Lemma 3.9. *$\mathbb{P}[F \subseteq E'] \geq 1 - 1/\operatorname{poly}(k)$.*

Proof. Pick an arbitrary edge $D \in F$. Consider $H \sim \mathsf{Sample}_{b,d,1}$ and let $c : [n] \to [b]$ be the coloring that defined H. It suffices to show that there is a unique edge that is $c(D)$-colored since then D is necessarily an edge in H. It suffices to show that this is the case with probability at least $1/2$ because repeating the process $O(\log k)$ times will ensure that such a family exists with high probability. The result then follows by taking the union bound over all $D \in F$ since $|F| = O(k^d)$ by Lemma 3.6.

Let $S = c(D)$. We first define a set A of vertices such that the only edge that is disjoint from A is D. It follows that D is the unique S-colored edge if $S \cap c(A) = \emptyset$, since every other edge intersects A and hence must share a color with it. We define A as follows:

$$A = (\mathrm{HS}(G) \setminus D) \cup \left(\bigcup_{C \subset D} \mathrm{HS}(M_{C,D})\right) .$$

Note D itself is disjoint from A since each $\mathrm{HS}(M_{C,D})$ does not include vertices from D. If an edge is disjoint from $(\mathrm{HS}(G) \setminus D)$ then it must intersect D. Suppose there exists an edge D' such that $D \cap D' = C \neq D$, then D' intersects $\mathrm{HS}(M_{C,D})$. Hence, the only edge that is disjoint from A includes the vertices in D and so is equal to D on the assumption that all edges have the same number of vertices.

It remains to show that $S \cap c(A) = \emptyset$ with probability at least $1/2$. If $b \geq 2d|A|$ then we have

$$\mathbb{P}\left[S \cap c(A) = \emptyset\right] \geq 1 - d|A|/b \geq 1/2 \ .$$

Finally, note that $b = O(k)$ since $|A| \leq \mathrm{hs}(G) + \sum_{C \subset D} \mathrm{hs}(M_{C,D}) \leq k + 2^d akd = O(k)$ by appealing to Lemma 3.7 and using the fact that $s_G(C) \leq ak$ for all $C \subset D$ since $D \in F$. □

A result for hypergraph matching follows along similar lines.

Theorem 3.4. *Suppose* $\mathrm{match}(G) \leq k' = k/d$. *With probability* $1 - 1/\mathrm{poly}(k)$, $\mathrm{match}(G') = \mathrm{match}(G)$.

Proof. $\mathrm{hs}(G) \leq dk' = k$. Let M be the matching. $F \cap M$ is preserved in G'. Consider an edge $D \in M$ such that $C \subseteq D$ for some $C \in U$. Then in G' we can find (by Lemma 3.8) at least $k+1$ petals in a sunflower with core either C itself or some $C' \subset C$. At most k of those intersect $M \setminus \{D\}$. Therefore, there is still at least one edge we can pick for the matching. □

3.4 Sampling Kernels for Subgraph Search Problems

We extend our parameterized results to a class of problems where the objective is to search for a subgraph H of $G(V, E)$ which satisfies some property $\mathcal{P}$. In the parametrized setting, we typically search for the largest H which satisfies this property, subject to the promise that the size of any H satisfying $\mathcal{P}$ is at most k. For concreteness, we assume the size is captured by the number of vertices in H, and our objective is to find a maximum cardinality satisfying subgraph. The sampling primitive $\mathsf{Sample}_{b,2,1}$ can be used here when $\mathcal{P}$ is preserved under vertex contraction: if G' is a vertex contraction of G, then any subgraph H of G' satisfying $\mathcal{P}$ also satisfies $\mathcal{P}$ for G (with vertices suitably remapped). Here, the vertex contraction of vertices u and v creates a new vertex whose neighbors are $\Gamma(u) \cup \Gamma(v)$. Many well-studied problems posess the required structure, including:

— b-matching, to find a (maximum cardinality) subgraph H of G such that the degree of each vertex in H is at most b. Hence, the standard notion of matching in Section 3.2 is equivalent to 1-matching.

— k-colorable subgraph, to find a subgraph H that is k-colorable. The maximum cardinality 2-colorable subgraph forms a max-cut, and more generally the maximum cardinality k-colorable subgraph is a max k-cut.

— other maximum subgraph problems, such as to find the largest subgraph that is a forest, has at least c connected components, or is a collection of vertex disjoint paths.

Theorem 3.5. *Let $\mathcal{P}$ be a graph property preserved under vertex contraction. Suppose that the number of vertices in some optimum solution* $\mathrm{opt}(G)$ *is at most k. Let $G' \sim \mathsf{Sample}_{4k^2,2,1}(G)$. With constant probability, we can compute a solution H for $\mathcal{P}$ from G' that achieves $|H| = |opt(G)|$.*

Proof. We construct a *contracted graph* G'' from G' based on the color classes used in the Sample operator: we contract all vertices that are assigned the same color by the hash function $c()$. Fix an optimum solution $opt(G)$ with at most k vertices. Lemma 3.1 shows that for $b = 4k^2$, all vertices involved in $opt(G)$ are hashed into distinct color values. Hence, the subgraph $opt(G)$ is a subgraph of G'': for any edge $e = (u, v) \in opt(G)$, the edge itself was sampled from the data structure, or else a different edge with the same color values was sampled, and so can be used interchangeably in G''. Hence, (the remapped form of) $opt(G)$ persists in G''. By the vertex contraction property of $\mathcal{P}$, this means that a maximum cardinality solution for $\mathcal{P}$ in G'' is a maximum cardinality solution in G.

Note that for this application of the subgraph sampling primitive, it suffices to implement the sampling data structure with a counter for each pair of colors: any non-zero count corresponds to an edge in G''. $\square$

Note that the generality of the result comes at the cost of increasing the number of colors, and hence the space of the stream algorithms. To generalize the result to the weighted case (e.g., where the objective is to find the subgraph satisfying $\mathcal{P}$ with the greatest total weight), we take the approach used in Section 3.3.1. We perform the sampling in parallel for each distinct weight value, and then round each edge weight to the closest power of $(1 + \epsilon)$ to reduce the number of weight classes to $O(\epsilon^{-1} \log W)$, with a loss factor of $(1 + \epsilon)$.

3.5 Approximating Large Matchings

The problem of *approximating* large matchings in the dynamic graph stream model has seen a flurry of recent activity [65–67, 77]. Assadi et al. [65] present a different α-approximation algorithm for maximum matching that uses the same space as our algorithm (which they show is optimal). Konrad [66] proves slightly weaker bounds. Bury and Schwiegelshohn [67] present an algorithm for estimating the *size* of the maximum matching in graphs of bounded arboricity.

Intuition and Preliminaries. Given a hash function $c : V \to [b]$, we say an edge uv is colored i if $c(u) = c(v) = i$. If the endpoints have different colors, we say the edge is *uncolored*. The basic idea behind our algorithm is to repeatedly sample a set of colored edges with distinct colors. Note that a set of edges colored with different colors is a matching. We use the edges in this matching to augment the matching already constructed from previous rounds. In this section we require the hash functions to be $O(k)$-wise independent and, in the context of dynamic data streams, this will increase the update time by a $O(k)$ factor.

Theorem 3.6. *Suppose* $\operatorname{match}(G) \geq k$. *For any* $1 \leq \alpha \leq \sqrt{k}$ *and* $0 < \epsilon \leq 1$, *with probability* $1 - 1/\operatorname{poly}(k)$,

$$\operatorname{match}(G') \geq \left(\frac{1-\epsilon}{2\alpha}\right) \cdot k \ ,$$

where $G' \sim \mathsf{Sample}_{2k/\alpha,1,r}$ *where* $r = O(k\alpha^{-2}\epsilon^{-2}\log k)$.

Proof. Let $H_1, \ldots, H_r \sim \mathsf{Sample}_{2k/\alpha,1,1}$ and let G' be the union of these graphs.

Consider the greedy matching M_r where $M_0 = \emptyset$ and for $t \geq 1$, M_t is the union of M_{t-1} and additional edges from H_t. We will show that if M_{t-1} is small, then we can find many edges in H_t that can be used to augment M_{t-1}.

Consider H_t and suppose $|M_{t-1}| < \frac{1-\epsilon}{2\alpha} \cdot k$. Let $c : V \to [b]$ be the hash-function used to define H_t where $b = \frac{2k}{\alpha}$. Let U be the set of colors that are not used to color the endpoints of M_{t-1}, i.e.,

$$U = \{c \in [b] : \text{ there does not exist a matched vertex } u \text{ in } M_{t-1} \text{ with } c(u) = c\} \ .$$

and note that $|U| \geq b - 2|M_{t-1}| \geq \frac{k}{\alpha}$. For each $c \in U$, define the indicator variable X_c where $X_c = 1$ if there exists an edge uv with $c(u) = c(v) = c$. We will find $X = \sum_{c \in U} X_c$ edges to add to the matching.

Since $\text{match}(G) \geq k$, there exists a set $k - 2|M_{t-1}| > k\epsilon$ vertex disjoint edges that can be added to M_{t-1}. Let $p = \frac{\alpha}{2k}$ and observe that $\mathbb{E}[X_c] \geq k\epsilon p^2 - \binom{k\epsilon}{2} p^4 > k\epsilon p^2/2 = \epsilon \cdot \frac{\alpha^2}{8k}$. Therefore, $\mathbb{E}[X] \geq \left(\frac{k}{\alpha}\right) \cdot \epsilon \cdot \frac{\alpha^2}{8k} = \frac{\epsilon\alpha}{8}$. Since X_c and $X_{c'}$ are negative correlated, $\mathbb{P}[X \geq E[X]/2] \geq 1 - \exp(-\Omega(\epsilon\alpha)) \geq \Omega(\epsilon)$. Hence, with each repetition we may increase the size of the matching by at least $\epsilon\alpha/2$ with probability $\Omega(\epsilon)$. After $O(k\alpha^{-2}\epsilon^{-2} \log k)$ repetitions the matching has size at least $\frac{1-\epsilon}{2\alpha} \cdot k$. $\square$

By applying Theorem 3.6 for all $k \in \{1, 2, 4, 8, 16, \ldots\}$ and appealing to Theorem 3.1, we establish:

Corollary 3.1. *There exists a $O(n \operatorname{polylog} n)$-space algorithm that returns an $O(n^{1/3})$-approximation to the size of the maximum matching in the dynamic graph stream model.*

Proof. For $1 \leq i \leq \log n$, let $G_i' \sim \mathsf{Sample}_{b,1,r}$ where $r = O(2^i \alpha^{-2} \log k)$ and

$b = 2^{i+1}/\alpha$. These graphs can be generated in $\tilde{O}(n^2\alpha^{-3})$ space. For some i, $2^i \leq \text{match}(G) < 2^{i+1}$ and hence $\text{match}(G'_i) = \Omega(\text{match}(G)/\alpha)$. □

This result generalizes to the weighted case using the Crouch-Stubbs technique [58]. They showed that if we can find a β-approximation to the maximum *cardinality* matching amongst all edges of weight greater than $(1+\epsilon)^i$ for each i, then we can find a $2(1+\epsilon)\beta$-approximation to the maximum weighted matching.

Matchings in Planar and Bounded-Arboricity Graphs. We also provide an algorithm for estimating the size of the matching in a graph of bounded arboricity. Recall that a graph has *arboricity* ν if its edges can be partitioned into at most ν forests. See Section 3.6 for details.

Theorem 3.7. *There exists a $\tilde{O}(\nu\epsilon^{-2}n^{4/5}\log\delta^{-1})$-space dynamic graph stream algorithm that returns a $(5\nu+9)(1+\epsilon)^2$ approximation of* $\text{match}(G)$ *with probability at least $1-\delta$ where ν is the arboricity of G.*

3.6 Matchings in Planar and Bounded-Arboricity Graphs

In this section, we present an algorithm for estimating the size of the matching in a graph of bounded arboricity. Recall that a graph has *arboricity* ν if its edges can be partitioned into at most ν forests. In particular, it can be shown that a planar graph has arboricity at most 3. We will make repeated use of the fact that the average degree of every subgraph of a graph with arboricity ν is at most 2ν.

Our algorithm is based on an insertion-only streaming algorithm due to Esfandiari et al. [78]. They first proved upper and lower bounds on the size of the maximum matching in a graph of arboricity ν.

Lemma 3.10 (Esfandiari et al. [78])**.** *For any graph G with arboricity ν, define a vertex to be* heavy *if its degree is at least $2\nu + 3$ and define an edge to be* shallow *if it is not incident to a heavy vertex. Then,*

$$\frac{\max\{h, s\}}{2.5\nu + 4.5} \leq \text{match}(G) \leq 2\max\{h, s\} \ .$$

where h is the number of heavy vertices and s is the number of shallow edges.

To estimate $\max\{h, s\}$, Esfandiari et al. sampled a set of vertices Z and (a) computed the exact degree of these vertices, then (b) found the set of all edges in the induced subgraph on these vertices. The fraction of heavy vertices in Z and shallow edges in the induced graph are then used to estimate h and s. By choosing the size of Z appropriately, they showed that the resulting estimate was sufficiently accurate on the assumption that $\max\{h, s\}$ is large. In the case where $\max\{h, s\}$ is small, the maximum matching is also small and hence a maximal matching could be constructed in small space using a greedy algorithm.

Algorithm for Dynamic Graph Streams. In the dynamic graph stream model, it is not possible to construct a maximal matching. However, we may instead use the algorithm of Theorem 3.2 to find the exact size of the maximum matching. Furthermore we can still recover the induced subgraph on sampled vertices Z via a sparse recovery sketch [79]. This can be done space-efficiently because the number of edges is at most $2\nu|Z|$. Lastly, rather than fixing the size of Z, we consider sampling each vertex independently with a fixed probability as this simplifies the analysis significantly. The resulting algorithm is as follows:

1. Invoke algorithm of Theorem 3.2 for $k = 2n^{2/5}$ and let r be the reported matching size.

2. In parallel, sample vertices with probability $p = 8\epsilon^{-2}n^{-1/5}$ and let Z be the set of sampled vertices. Find the degrees of vertices in Z in G and maintain a $2\nu|Z|$-sparse recovery sketch of the edges in the induced graph on Z. Let s_Z be the number of shallow edges in the induced graph on Z and let s_Z be the number of heavy vertices in Z. Return $\max\{r, h_Z/p, s_Z/p^2\}$.

Analysis. Our analysis relies on the following lemma that shows that $\max\{h_Z/p, s_Z/p^2\}$ is a $1+\epsilon$ approximation for $\max\{s, h\}$ on the assumption that $\max\{s, h\} \geq n^{2/5}$.

Lemma 3.11. $\mathbb{P}\left[|\max\{h_Z/p, s_Z/p^2\} - \max\{s, h\}| \leq \epsilon \cdot \max\{n^{2/5}, s, h\}\right] \geq 4/5$.

Proof. First we show s_Z/p^2 is a sufficiently good estimate for s. Let S be the set of shallow edges in G and let E_Z be the set of edges in the induced graph on Z. For each shallow edge $e \in S$, define an indicator random variable X_e where $X_e = 1$ iff $e \in E_Z$ and note that $s_Z = \sum_{e\in S} X_e$. Then,

$$\mathbb{E}\left[s_Z\right] = sp^2 \quad \text{and} \quad \mathbb{V}\left[s_Z\right] = \sum_{e\in S}\sum_{e'\in S} \mathbb{E}\left[X_e X_{e'}\right] - \mathbb{E}\left[X_e\right]\mathbb{E}\left[X_{e'}\right] \ .$$

Note that

$$\sum_{e' \in S} \mathbb{E}\left[X_e X_{e'}\right] - \mathbb{E}\left[X_e\right]\mathbb{E}\left[X_{e'}\right] = \begin{cases} p^2 - p^4 & \text{if } e = e' \\ p^3 - p^4 & \text{if } e \text{ and } e' \text{ share exactly one endpoint} \\ 0 & \text{if } e \text{ and } e' \text{ share no endpoints} \end{cases} .$$

and since there are at most $2\nu + 3$ edges that share an endpoint with a shallow edge,

$$\mathbb{V}\left[s_Z\right] \leq s(p^2 - p^4 + (2\nu + 3)p^3 - p^4) \leq 2sp^2$$

on the assumption that $(2\nu + 3) \leq 1/p$. We then use Chebyshev's inequality to obtain

$$\mathbb{P}\left[|s_Z - sp^2| \leq p^2 \epsilon \cdot \max\{n^{2/5}, s\}\right] \leq \frac{2sp^2}{(p^2 \epsilon \cdot \max\{n^{2/5}, s\})^2} \leq 9/10 \ . \tag{3.2}$$

Next we show that h_Z/p is a sufficiently good estimate for h. Let H denote the set of h heavy vertices in G and define an indicator random variable Y_v for each $v \in H$, where $Y_v = 1$ iff $v \in Z$. Note that $h_Z = \sum_{v \in H} Y_v$ and $\mathbb{E}\left[h_Z\right] = hp$. Then, by an application of the Chernoff-Hoeffding bound,

$$\mathbb{P}\left[|h_Z - hp| \geq \epsilon p \max\{h, n^{2/5}\}\right] \leq \exp(-\epsilon^2 p n^{2/5}/3) \leq 9/10 \ . \tag{3.3}$$

Therefore, it follows from Eq. 3.2 and 3.3 that

$$\mathbb{P}\left[|\max\{h_Z/p, s_Z/p^2\} - \max\{s, h\}| \leq \epsilon \cdot \max\{n^{2/5}, s, h\}\right] \geq 4/5 \ .$$

□

Theorem 3.8. *There exists a $\tilde{O}(\nu\epsilon^{-2}n^{4/5}\log\delta^{-1})$-space dynamic graph stream algorithm that returns a $(5\nu+9)(1+\epsilon)^2$ approximation of* $\operatorname{match}(G)$ *with probability at least $1-\delta$ where ν is the arboricity of G.*

Proof. To argue the approximation factor, first suppose $\operatorname{match}(G) \le 2n^{2/5}$. In this case $r = \operatorname{match}(G)$ and $\max\{s,h\} \le (2.5\nu+4.5)\operatorname{match}(G)$ by appealing to Lemma 3.10. Hence,

$$\operatorname{match}(G) \le \max\{r, h_Z/p, s_Z/p^2\} \le (2.5\nu+4.5)\operatorname{match}(G)$$

Next suppose $\operatorname{match}(G) \ge 2n^{2/5}$. In this case, $\max\{s,h\} \ge n^{2/5}$ by Lemma 3.10. Therefore, by Lemma 3.11, $\max\{h_Z/p, s_Z/p^2\} = (1\pm\epsilon)\max\{s,h\}$, and so

$$\frac{\operatorname{match}(G)}{2(1+\epsilon)} \le \max\{r, h_Z/p, s_Z/p^2\} \le (1+\epsilon)\max\{s,h\} \le (1+\epsilon)(2.5\nu+4.5)\operatorname{match}(G)$$

To argue the space bound, recall that the algorithm used in Theorem 3.2 requires $\tilde{O}(n^{4/5})$ space. Note that $|Z| \le 2np = \tilde{O}(\epsilon^{-2}n^{4/5})$ with high probability. Hence, to sample the vertices Z and maintain a $2\nu|Z|$-sparse recovery data structure requires $\tilde{O}(n^{4/5}\nu)$ space. □

3.7 Lower Bounds

3.7.1 Matching and Hitting Set Lower Bounds

The following theorem establishes that the space-use of our matching, vertex cover, hitting set, and hyper matching algorithms is optimal up to logarithmic factors.

Theorem 3.9. *Any (randomized) parametrized streaming algorithm for the minimum d-hitting set or maximum (hyper)matching problem with parameter k requires $\Omega(k^d)$ space.*

Proof. We reduce from the MEMBERSHIP communication problem:

> MEMBERSHIP
>
> *Input*: Alice has a set $X \subseteq [n]$, and Bob has an element $1 \leq x \leq n$.
>
> *Question*: Bob wants to check whether $x \in X$.

There is a lower bound of $\Omega(n)$ bits of communication from Alice to Bob, even allowing randomization [80].

Let $S = s_1 s_2 ... s_n$ be the characteristic string of X, i.e. a binary string such that $s_i = 1$ iff $i \in X$. Let $k = \sqrt[d]{n}$. Fix a canonical mapping $h : [n] \to [k]^d$. This way we can view an n bit string as an adjacency matrix of a d-partite graph. Construct the following graph G with d vertex partitions $V_1, V_2, ..., V_d$:

- Each partition V_i has dk vertices: for each $j \in [k]$ create vertices $v^*_{i,j}$, $v^1_{i,j}$, $v^2_{i,j}$,..., $v^{d-1}_{i,j}$.
- Alice inserts a hyperedge $(v^*_{1,j_1}, v^*_{2,j_2}, ..., v^*_{d,j_d})$ iff the corresponding bit in the string S is 1, i.e., $s_a = 1$ where $h(a) = (j_1, j_2, ..., j_d)$.
- Let $h(x) = (J_1, J_2, ..., J_d)$. Bob inserts edge $(v^*_{i,j}, v^1_{i,j}, v^2_{i,j}, ..., v^{d-1}_{i,j})$ iff $j \neq J_i$.

Alice runs the hitting set algorithm on the edges she is inserting using space $f(k)$. Then she sends the memory contents of the algorithm to Bob, who finishes running the algorithm on his edges.

The minimum hitting set should include vertices $v^*_{i,j}$ such that $j \neq J_i$. If edge $(v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d})$ is in the graph, we also need to include one of its vertices. Therefore,

$$x \in X \iff s_x = 1 \iff (v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d}) \text{ is in } G \iff \text{hs}(G) = dk - d + 1$$

On the other hand,

$$x \notin X \iff s_x = 0 \iff (v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d}) \text{ is not in } G \iff \text{hs}(G) = dk - d$$

Alice only sends $f(k)$ bits to Bob. Therefore, $f(k) = \Omega(n) = \Omega(k^d)$.

For the lower bound on matching we use the same construction. For each vertex $v^*_{i,j}$ such that $j \neq J_i$ maximum matching should include $(v^*_{i,j}, v^1_{i,j}, v^2_{i,j}, ..., v^{d-1}_{i,j})$. If edge $(v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d})$ is in the graph, we include it in the matching as well. Therefore,

$$x \in X \iff s_x = 1 \iff (v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d}) \text{ is in } G \iff \text{match}(G) = dk - d + 1$$

And

$$x \notin X \iff s_x = 0 \iff (v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d}) \text{ is not in } G \iff \text{match}(G) = dk - d$$

□

3.7.2 Lower Bounds for Problems considered by Fafianie and Kratsch [50]

Comparison with Lower Bounds for Streaming Kernels: Fafianie and Kratsch [50] introduced the notion of kernelization in the streaming setting as follows:

Definition 3.2. *A 1-pass streaming kernelization algorithm is receives an input (x, k) and returns a kernel, with the restriction that the space usage of the algorithm is bounded by $p(k) \cdot \log |x|$ for some polynomial p.*

Fafianie and Kratsch [50] gave deterministic lower bounds for several parameterized problems. In particular, they showed that:

- Any 1-pass kernel for Edge Dominating Set(k) requires $\Omega(m)$ bits, where m is the number of edges. However, there is a 2-pass kernel which uses $O(k^3 \cdot \log n)$ bits of local memory and $O(k^2)$ time in each step and returns an equivalent instance of size $O(k^3 \cdot \log k)$.
- The lower bound of $\Omega(m)$ bits for any 1-pass kernel also holds for several other problems such as Cluster Editing(k), Cluster Deletion(k), Cluster Vertex Deletion(k), Cograph Vertex Deletion(k), Minimum Fill-In(k), Edge Bipartization(k), Feedback Vertex Set(k), Odd Cycle Transversal(k), Triangle Edge Deletion(k), Triangle Vertex Deletion(k), Triangle Packing(k), s-Star Packing(k), Bipartite Colorful Neighborhood(k).
- Any t-pass kernel for Cluster Editing(k) and Minimum Fill-In(k) requires $\Omega(n/t)$ space.

In this section, we give $\Omega(n)$ randomized lower bounds for the space complexity of all the problems considered by Fafianie and Kratsch. In addition, we also consider some other problems such as Path(k) which were not considered by Fafianie and Kratsch. A simple observation shows that any lower bound for parameterized streaming kernels also transfers for the parameterized streaming algorithms. Thus the results of Fafiane and Kratsch [50] also give lower bounds for the parameterized streaming algorithms for these problems. However, our lower bounds have the following advantage over the results of [50]:

- All our lower bounds also hold for *randomized algorithms*, whereas the kernel lower bounds were for deterministic algorithms.
- With the exception of Edge Dominating Set(k), all our lower bounds also hold for *constant number of passes*.

3.7.2.1 Lower Bound for Edge Dominating Set

We now show a lower bound for the Edge Dominating Set(k) problem.

Definition 3.3. *Given a graph $G = (V, E)$ we say that a set of edges $X \subseteq E$ is an edge dominating set if every edge in $E \setminus X$ is incident on some edge of X.*

Edge Dominating Set(k)	*Parameter*: k
Input: An undirected graphs G and an integer k	
Question: Does there exist an edge dominating set $X \subseteq E$ of size at most k?	

Theorem 3.10. *For the* Edge Dominating Set(k) *problem, any (randomized) streaming algorithm needs $\Omega(n)$ space .*

Proof. Given an instance of Membership, we create a graph G on $n + 2$ vertices as follows. For each $i \in [n]$ we create a vertex v_i. Also add two special vertices a and b. For every $y \in X$, add the edge (a, y). Finally add the edge (b, x).

Now we will show that G has an edge dominating set of size 1 iff Membership answers YES. In the first direction suppose that G has an edge dominating set of size 1. Then it must be the case that $x \in X$: otherwise for a minimum edge dominating set we need one extra edge to dominate the star incident on a, in addition to the edge (b, x) dominating itself. Hence Membership answers YES. In reverse direction, suppose that Membership answers YES. Then the edge (a, x) is clearly an edge dominating set of size 1.

Therefore, any (randomized) streaming algorithm that can determine whether a graph has an edge dominating set of size at most $k = 1$ gives a communication protocol for Membership, and hence requires $\Omega(n)$ space. □

3.7.2.2 Lower Bound for $\mathcal{G}$-Free Deletion

Definition 3.4. *A set of connected graphs $\mathcal{G}$ is* bad *if there is a minimal (under operation of taking subgraphs) graph $H \in \mathcal{G}$ such that H contains at least two distinct edges.*

For any bad set of graphs $\mathcal{G}$, we now show a lower bound for the following general problem:

$\mathcal{G}$-Free Deletion(k)	*Parameter*: k

Input: A bad set of graphs $\mathcal{G}$, an undirected graph $G = (V, E)$ and an integer k

Question: Does there exist a set $X \subseteq V$ such that $G \setminus X$ contains no graph from $\mathcal{G}$?

The reduction from the Disjointness problem in communication complexity.

Disjointness

Input: Alice has a string $x \in \{0,1\}^n$ given by $x_1x_2\dots x_n$. Bob has a string $y \in \{0,1\}^n$ given by $y_1y_2\dots y_n$.

Question: Bob wants to check if $\exists\ i \in [n]$ such that $x_i = y_i = 1$.

There is a lower bound of $\Omega(n/p)$ bits of communication between Alice and Bob, allowing p-rounds and randomization [81].

Theorem 3.11. *For a* bad *set of graphs $\mathcal{G}$, any p-pass (randomized) streaming algorithm for the $\mathcal{G}$-*Free Deletion *problem needs $\Omega(n/p)$ space .*

Proof.

Since $\mathcal{G}$ is a *bad* set of graphs, there is a minimal graph $H \in \mathcal{G}$ which has at least two distinct edges, say e_1 and e_2. Let $H' := H \setminus \{e_1, e_2\}$. Given an instance of Disjointness, we create a graph G which consists of n disjoint copies say $G_1, G_2, \dots, G_n$ of H'. For each $i \in [n]$, to the copy G_i of H' we add the edge e_1 iff $x_i = 1$ and the edge e_2 iff $y_i = 1$. We now show that the resulting graph G contains a copy of H if and only if it is a YES instance of Disjointness.

Suppose that it is a YES instance of Disjointness. So there is a $j \in [n]$ such that $x_j = 1 = y_j$. Therefore, to the copy G_j of H' we would have added the edges e_1 and e_2 which would complete it into H. So G contains a copy of H. In other direction, suppose that G contains a copy of H. Note that since we add n disjoint copies of H' and add at most two edges (e_1 and e_2) to each copy, it follows that each connected component of G is in fact a subgraph of $H = H' \cup (e_1 + e_2)$. Since H is connected and G contains a copy of H, some connected component of G must exactly be the graph H, i.e, to some copy G_i of H' we must have added both the edges e_1 and e_2. This implies $x_i = 1 = y_i$, and so DISJOINTNESS answers YES.

Since each connected component of G is a subgraph of H, the minimality of H implies that $\mathcal{G}$ contains a graph from G iff G contains a copy of H, which in turn is true iff DISJOINTNESS answers YES. Therefore, any p-pass (randomized) streaming algorithm that can determine whether a graph is $\mathcal{G}$-free (i.e., answers the question with $k = 0$) gives a communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space. □

This implies lower bounds for the following set of problems:

Theorem 3.12. *For each of the following problems, any p-pass (randomized) algorithm requires* $\Omega(n/p)$ *space:* FEEDBACK VERTEX SET(k)*,* ODD CYCLE TRANSVERSAL(k)*,* EVEN CYCLE TRANSVERSAL(k) *and* TRIANGLE DELETION(k)*.*

Proof. We first define the problems below:

FEEDBACK VERTEX SET(k)	*Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k	
Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ has no cycles?	

ODD CYCLE TRANSVERSAL(k)	*Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k	
Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ has no odd cycles?	

EVEN CYCLE TRANSVERSAL(k) *Parameter*: k

Input: An undirected graph $G = (V, E)$ and an integer k

Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ has no even cycles?

TRIANGLE DELETION(k) *Parameter*: k

Input: An undirected graph $G = (V, E)$ and an integer k

Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ has no triangles?

Now we show how each of these problems can be viewed as a $\mathcal{G}$-FREE DELETION problem for an appropriate choice of *bad* $\mathcal{G}$.

- FEEDBACK VERTEX SET(k): Take $\mathcal{G} = \{C_3, C_4, C_5, \ldots\}$ and $H = C_3$
- ODD CYCLE TRANSVERSAL(k): Take $\mathcal{G} = \{C_3, C_5, C_7, \ldots\}$ and $H = C_3$
- EVEN CYCLE TRANSVERSAL(k): Take $\mathcal{G} = \{C_4, C_6, C_8, \ldots\}$ and $H = C_4$
- TRIANGLE DELETION(k): Take $\mathcal{G} = \{C_3\}$ and $H = C_3$

We verify the conditions for FEEDBACK VERTEX SET(k); the proofs for other problems are similar. Note that the choice of $\mathcal{G} = \{C_3, C_4, C_5, \ldots\}$ and $H = C_3$ implies that $\mathcal{G}$ is *bad* since each graph in $\mathcal{G}$ is connected, the graph H belongs to $\mathcal{G}$, has at least two distinct edges and is a minimal element of $\mathcal{G}$ (under operation of taking subgraphs). Finally, finding a set X such that the graph $G \setminus X$ is $\mathcal{G}$-free implies that it has no cycles, i.e., X is a feedback vertex set for G. $\square$

It is easy to see that the same proofs also work for the edge deletion versions of the ODD CYCLE TRANSVERSAL(k), EVEN CYCLE TRANSVERSAL(k) and the TRIANGLE DELETION(k) problems.

3.7.2.3 $\mathcal{G}$-EDITING

Definition 3.5. *A set of graphs $\mathcal{G}$ is* good *if there is a minimal (under operation of taking subgraphs) connected graph $H \in \mathcal{G}$ such that H has at least two distinct edges.*

Definition 3.5 looks very similar to Definition 3.4: however there is a subtle difference. Each graph in a *bad* set of graphs must be connected while only a minimal graph in a *good* set of graphs is required to be connected. This difference is used crucially in the proofs of Theorem 3.11 and Theorem 3.13.

For any *good* set of graphs $\mathcal{G}$, we now show a lower bound for the following general problem:

$\mathcal{G}$-EDITING(k)	*Parameter*: k
Input: A graph class $\mathcal{G}$, an undirected graph $G = (V, E)$ and an integer k	
Question: Does there exist a set X of k edges such that $(V, E \cup X)$ contains a graph from $\mathcal{G}$?	

Theorem 3.13. *For a* good *set of graphs $\mathcal{G}$, any p-pass (randomized) streaming algorithm for the $\mathcal{G}$-EDITING(k) problem needs $\Omega(n/p)$ space .*

Proof. Since $\mathcal{G}$ is a *good* set of graphs, there is a minimal graph $H \in \mathcal{G}$ such that H is connected and has at least two distinct edges, say e_1 and e_2. Let $H' := H \setminus \{e_1, e_2\}$.

Given an instance of DISJOINTNESS, we create a graph G which consists of n disjoint copies say $G_1, G_2, \ldots, G_n$ of H'. By minimality of H, it follows that $H' \notin \mathcal{G}$. For each $i \in [n]$ we add to G_i the edge e_1 iff $x_i = 1$ and the edge e_2 iff $y_i = 1$. Let the resulting graph be G.

We now show that G contains a copy of H if and only if DISJOINTNESS answers YES. Suppose that G contains a copy of H. Note that since we add n disjoint copies of H' and add at most two edges (e_1 and e_2) to each copy, it follows that each connected component of G is in fact a subgraph of $H = H' \cup (e_1 + e_2)$. Since H is connected and G contains a copy of H, some connected component of G must exactly be the graph H, i.e, to some copy G_i of H' we must have added both the edges e_1 and e_2. This implies $x_i = 1 = y_i$, and so DISJOINTNESS answers YES. Now suppose that DISJOINTNESS answers YES, i.e., there exists $j \in [n]$ such that $x_j = 1 = y_j$. Therefore, to the copy G_j of H' we would have added the edges e_1 and e_2 which would complete it into H. So G contains a copy of H.

Otherwise due to minimality of H, the graph G does not contain any graph from $\mathcal{G}$. Therefore, any p-pass (randomized) streaming algorithm that can determine whether a graph G contains a graph from $\mathcal{G}$ (i.e., answers the question with $k = 0$) gives a communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space.

□

This implies lower bounds for the following set of problems:

Theorem 3.14. *For each of the following problems, any p-pass (randomized) algorithm requires $\Omega(n/p)$ space:* TRIANGLE PACKING(k), s-STAR PACKING(k) *and*

PATH(k).

Proof. We first define the problems below:

TRIANGLE PACKING(k) *Parameter*: k

Input: An undirected graph $G = (V, E)$ and an integer k

Question: Do there exist at least k vertex disjoint triangles in G?

s-STAR PACKING(k) *Parameter*: k

Input: An undirected graph $G = (V, E)$ and an integer k

Question: Do there exist at least k vertex disjoint instances of $K_{1,s}$ in G (where $s \geq 3$)?

PATH(k) *Parameter*: k

Input: An undirected graph $G = (V, E)$ and an integer k

Question: Does there exist a path in G of length $\geq k$?

Now we show how each of these problems can be viewed as a $\mathcal{G}$-EDITING problem for an appropriate choice of *good* $\mathcal{G}$.

- TRIANGLE PACKING(k) with $k = 1$: Take $\mathcal{G} = \{C_3\}$ and $H = C_3$
- s-STAR PACKING(k) with $k = 1$: Take $\mathcal{G} = \{K_{1,s}\}$ and $H = K_{1,s}$
- PATH(k) with $k = 3$: Take $\mathcal{G} = \{P_3, P_4, P_5, \ldots\}$ and $H = P_3$

We verify the conditions for TRIANGLE PACKING(k) with $k = 1$; the proofs for other problems are similar. Note that the choice of $\mathcal{G} = \{C_3\}$ and $H = C_3$ implies that $\mathcal{G}$ is *good* since $\mathcal{G}$ only contains one graph H which is connected and has at least two distinct edges. Finally, finding a set of edges X such that the graph $(V, E \cup X)$

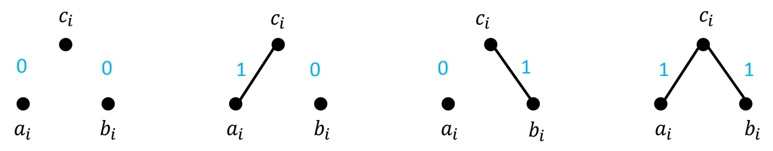


Figure 3.2: Gadget for reduction from DISJOINTNESS to CLUSTER VERTEX DELETION

contains a graph from $\mathcal{G}$ implies that it has at least one C_3, i.e., X is a solution for TRIANGLE PACKING(k) with $k = 1$. □

3.7.2.4 Lower Bound for CLUSTER VERTEX DELETION

We now show a lower bound for the CLUSTER VERTEX DELETION(k) problem.

Definition 3.6. *We say that G is a* cluster *graph if each connected component of G is a clique.*

CLUSTER VERTEX DELETION(k)	*Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k	
Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ is a cluster graph?	

Theorem 3.15. *For the* CLUSTER VERTEX DELETION(k) *problem, any p-pass (randomized) streaming algorithm needs $\Omega(n/p)$ space .*

Proof. Given an instance of DISJOINTNESS, we create a graph G on $3n$ vertices as follows. For each $i \in [n]$ we create three vertices a_i, b_i, c_i. Insert the edge (a_i, c_i) iff

$x_i = 1$ and the edge (b_i, c_i) iff $y_i = 1$ This is illustrated in Figure 3.2.

Now we will show that each connected component of G is a clique iff DISJOINTNESS answers NO. In the first direction suppose that each connected component of G is a clique. Then there cannot exist $i \in [n]$ such that $x_i = 1 = y_i$ because then the vertices a_i, b_i, c_i will form a connected component which is a P_3; this contradicts the assumption that each connected component of G is a clique. In reverse direction, suppose that DISJOINTNESS answers NO. Then it is easy to see that each connected component of G is either P_1 or P_2, both of which are cliques.

Therefore, any p-pass (randomized) streaming algorithm that can determine whether a graph is a cluster graph (i.e., answers the question with $k = 0$) gives a communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space. [2] □

3.7.2.5 Lower Bound for MINIMUM FILL-IN

We now show a lower bound for the MINIMUM FILL-IN(k) problem.

Definition 3.7. *We say that G is a* chordal *graph if it does not contain an induced cycle of length ≥ 4.*

MINIMUM FILL-IN(k)	*Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k	
Question: Does there exist a set X of at most k edges such that $(V, E \cup X)$ is a chordal graph?	

[2] It is easy to see that the same proof also works for the problems of CLUSTER EDGE DELETION(k) where we can delete at most k edges and CLUSTER EDITING(k) where we can delete/add at most k edges

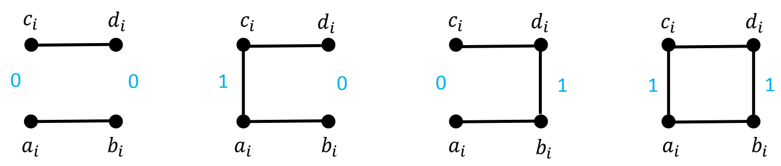


Figure 3.3: Gadget for reduction from Disjointness to Minimum Fill-In

Theorem 3.16. *For the* Minimum Fill-In(k) *problem, any p-pass (randomized) streaming algorithm needs $\Omega(n/p)$ space .*

Proof. We reduce from the Disjointness problem in communication complexity. Given an instance of Disjointness, we create a graph G on $4n$ vertices as follows. For each $i \in [n]$ we create vertices a_i, b_i, c_i, d_i and insert edges (a_i, b_i) and (c_i, d_i). Insert the edge (a_i, c_i) iff $x_i = 1$ and the edge (b_i, c_i) iff $y_i = 1$. This is illustrated in Figure 3.3.

Now we will show that G is chordal iff Disjointness answers NO. In the first direction suppose that G is chordal. Then there cannot exist $i \in [n]$ such that $x_i = 1 = y_i$ because then the vertices a_i, b_i, c_i, d_i will form an induced C_4; contradicting the assumption that G is chordal. In reverse direction, suppose that Disjointness answers NO. Then it is easy to see that each connected component of G is either P_2 or P_3. Hence, G cannot have an induced cycle of length ≥ 4, i.e., G is chordal.

Therefore, any p-pass (randomized) streaming algorithm that can determine whether a graph is a chordal graph (i.e., answers the question with $k = 0$) gives a

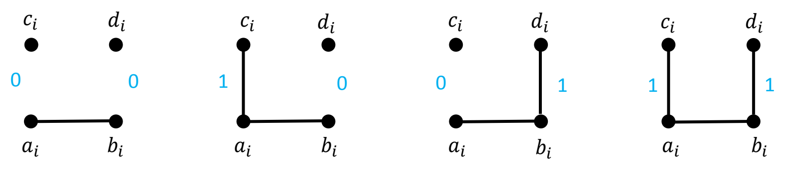


Figure 3.4: Gadget for reduction from DISJOINTNESS to COGRAPH VERTEX DELETION

communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space. □

3.7.2.6 Lower Bound for COGRAPH VERTEX DELETION

We now show a lower bound for the COGRAPH VERTEX DELETION(k) problem.

Definition 3.8. *We say that G is a* cograph *if it does not contain an induced P_4.*

COGRAPH VERTEX DELETION(k)	*Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k	
Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ is a cograph?	

Theorem 3.17. *For the* COGRAPH VERTEX DELETION(k) *problem, any p-pass (randomized) streaming algorithm needs $\Omega(n/p)$ space .*

Proof. We reduce from the DISJOINTNESS problem in communication complexity. Given an instance of DISJOINTNESS, we create a graph G on $4n$ vertices as follows.

For each $i \in [n]$ we create vertices a_i, b_i, c_i, d_i and insert edges (a_i, b_i). Insert the edge (a_i, c_i) iff $x_i = 1$ and the edge (b_i, c_i) iff $y_i = 1$. This is illustrated in Figure 3.4.

Now we will show that G has an induced P_4 if and only if DISJOINTNESS answers YES. In the first direction suppose that G has an induced P_4. Since each connected component of G can have at most 4 vertices, it follows that the P_4 is indeed given by the path $c_i - a_i - b_i - d_i$ for some $i \in [n]$. By construction of G, this implies that $x_i = 1 = y_i$, i.e., DISJOINTNESS answers YES. In reverse direction, suppose that DISJOINTNESS answers YES. Then there exists $j \in [n]$ such that the edges (a_i, c_i) and (b_i, d_i) belong to G. Then G has the following induced P_4 given by $c_j - a_j - b_j - d_j$.

Therefore, any p-pass (randomized) streaming algorithm that can determine whether a graph is a cograph (i.e., answers the question with $k = 0$) gives a communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space. □

3.7.2.7 Bipartite Colorful Neighborhood

We now show a lower bound for the BIPARTITE COLORFUL NEIGHBORHOOD(k) problem.

BIPARTITE COLORFUL NEIGHBORHOOD(k) *Parameter*: k

Input: A bipartite graph $G = (A, B, E)$ and an integer k

Question: Is there a 2-coloring of B such that there exists a set $S \subseteq A$ of size at least k such that each element of S has at least one neighbor in B of either color?

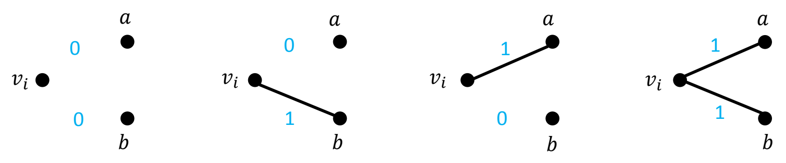


Figure 3.5: Gadget for reduction from Disjointness to Bipartite Colorful Neighborhood

Theorem 3.18. *For the* Bipartite Colorful Neighborhood(k) *problem, any p-pass (randomized) streaming algorithm needs $\Omega(n/p)$ space .*

Proof. We reduce from the Disjointness problem in communication complexity. Given an instance of Disjointness, we create a graph G on $n+2$ vertices as follows. For each $i \in [n]$ we create a vertex v_i. In addition, we have two special vertices a and b. For each $i \in [n]$, insert the edge (a, v_i) iff $x_i = 1$ and the edge (b, v_i) iff $y_i = 1$. Let $A = \{v_1, v_2, \ldots, v_n\}$ and $B = \{a, b\}$. This is illustrated in Figure 3.5.

Now we will show that G answers YES for Bipartite Colorful Neighborhood(k) with $k = 1$ iff Disjointness answers YES. In the first direction suppose that G answers YES for Bipartite Colorful Neighborhood(k) with $k = 1$. Let v_i be the element in A which has at least one neighbor in B of either color. Since $|B| = 2$, this means that v_i is adjacent to both a and b, i.e., $x_i = 1 = y_i$ and hence Disjointness answers YES. In reverse direction, suppose that Disjointness answers YES. Hence, there exists $j \in [n]$ such that $x_j = 1 = y_j$. This implies that v_j is adjacent to both a and b. Consider the 2-coloring of B by giving different colors to

a and b. Then $S = \{v_j\}$ satisfies the condition of having a neighbor of each color in B, and hence G answers YES for Bipartite Colorful Neighborhood(k) with $k = 1$.

Therefore, any p-pass (randomized) streaming algorithm that can solve Bipartite Colorful Neighborhood(k) with $k = 1$ gives a communication protocol for Disjointness, and hence requires $\Omega(n/p)$ space.

$\square$

Chapter 4: Market Pricing over Streams

4.1 Introduction

Modern, Internet-enabled marketplaces have the potential to serve an extremely large volume of transactions. Giant online markets such as eBay and Amazon work with massive datasets that record the exchange of goods between many different buyers and sellers. Such datasets present an opportunity: it is natural to analyze the history of transactions to estimate demand and solve central pricing problems. Indeed, a recent and exciting line of literature in the algorithmic game theory community has set out to understand how the availability of data, in the form of samples from a transaction history, can be employed to tune prices and design mechanisms [82–86]. Big-data environments are a boon for such tasks. However, as datasets grow ever larger, data-analysis algorithms must become ever more efficient. An algorithm that runs in polynomial (or even linear) time may not have a reasonable running time in practice.

In this chapter we study the development of pricing algorithms that are appropriate for use with massive datasets. We will adopt the model of streaming algorithms, a standard model for massive dataset analysis. In the streaming algorithm model, a stream of data arrives sequentially and must be analyzed by an algorithm with limited memory. These streams can only be read once (or a limited number of times), and hence streaming algorithms must be frugal in the amount and nature of data that they choose to store.

Streaming algorithms were first theoretically introduced in fields such as data mining and machine learning over 20 years ago in order to model problems in which the data cannot be accessed all at once. Over the past decade, there has been a significant demand for algorithms to process and handle dynamic data coming from huge and growing graphs such as social networks, webpages and their links and citations of academic work. These algorithmic techniques are also relevant to market design problems. For instance, one might imagine that there is a set of items (e.g., goods for sale) and a set of potential buyers (e.g., individual consumers) to which they should be matched. Such markets are essentially bipartite matching problems, which have themselves been the subject of study in the context of streaming algorithms. This begs the question: to what extent can market design and pricing problems be resolved adequately in the streaming model?

We have in mind two main applications of solving this pricing problem on a massive collection of static data, especially on the data of previous sales. First, by computing optimal prices on past transactions, one can subsequently employ those prices as a guideline for setting future prices; this is a key step in many recent pricing methodologies based on statistical learning theory. Also, the optimal welfare or revenue in hindsight is a useful benchmark for the online pricing mechanism being employed by the platform, and can therefore be used to evaluate and tune.

4.1.1 Our Results

In this chapter, we instantiate our high-level question by focusing on the *envy-free pricing problem* with big data. Our model is as follows. Suppose there is a bipartite graph G with a set of n unit-demand buyers $b_1, \ldots, b_n$ on one side, and a set of k distinct types of items $v_1, \ldots, v_k$, with l copies of each on the other side. The utility of buyer b_j for item v_i is denoted by u_{v_i,b_j}, and is shown by a weighted edge between the corresponding two vertices. The price assigned to item i is denoted by p_i. The goal is to assign prices to items, and then items to buyers, such that the assignment is *envy-free*[1], i.e., each buyer prefers the item assigned to her rather than an item assigned to another buyer. Subject to the envy-freeness condition, the designer wishes to maximize either the social welfare or revenue of the corresponding allocation. This is precisely the envy-free pricing problem introduced by Guruswami et al. [87]. We ask: how well can envy-free prices be computed in the *streaming setting*?

We note that there are many possible representations of the input as a data stream. We will perform our analysis under a model in which the utility values u_{v_i,b_j} arrive in a data stream in an arbitrary order. We note that there are other potential options, such as assuming that all values associated with a certain agent arrive simultaneously, or that the values u_{v_i,b_j} are not provided directly but rather the input contains only the revealed preference of a buyer in response to prices. We

[1]If we assign item v_{i_1} to buyer b_{j_1} and item v_{i_2} to buyer b_{j_2}, then we have $u_{v_{i_1},b_{j_1}} - p(i_1) \geq u_{v_{i_2},b_{j_1}} - p(i_2)$.

leave the exploration of these alternative models as an avenue for future work.

We consider both social-welfare maximization and revenue maximization versions of the envy-free pricing problem. First, we provide streaming mechanisms that compute both allocation and prices of the items using $O(k^2 l)$ space. Later, we present streaming mechanisms that only compute the prices using space $\tilde{O}(k^3)$, approximating social welfare (or revenue) within a factor of $1 - \epsilon$, where polylogarithmic factors are hidden in the notation of $\tilde{O}$. At the end, we present lower bounds on the required space of any mechanism that computes optimum prices for either social-welfare maximization or revenue maximization.

In Theorem 4.1 we provide an envy-free streaming mechanism for the social-welfare maximization problem using $O(k^2 l)$ space.

Theorem 4.1. *There exists an envy-free mechanism for the social-welfare maximization problem in the streaming setting using $O(k^2 l)$ space. This mechanism remembers the allocation as well as the prices.*

Indeed, finding the maximum matching in a bipartite graph with $O(k)$ vertices in the streaming setting requires $\Omega(k^2)$ space [88]. Thus, for $l = 1$, the space required by our mechanisms in Theorem 4.2 and Theorem 4.1 are tight.

The following theorem extends our result to the revenue maximization problem. Even in the static (i.e., non-streaming) environment, this problem has resisted constant-factor approximation factors for simple versions, including the case of unit-demand bidders studied here. We frame our result as a reduction: given an algorithm for computing envy-free prices in the static setting, we show how to construct a streaming algorithm with the same approximation guarantee. As with the welfare maximization problem, the required space is $O(k^2l)$.

Theorem 4.2. *Given an envy-free α-approximation mechanism for the revenue maximization problem, there exists an envy-free α-approximation mechanism for the revenue maximization problem in the streaming setting using $O(k^2l)$ space. This mechanism remembers the allocation as well as the prices.*

Each of the above results are with respect to algorithms that return not only a profile of envy-free prices, but also the corresponding allocation. We note that the size of the allocation is $O(kl)$, and thus any mechanism that remembers the allocation requires at least $\Omega(kl)$ space. This space may be quite large when l is large, which is not desirable. What if we are only interested in determining the envy-free prices, and just being within an approximation of the maximum social welfare(or revenue)? As it turns out, this variation of the problem allows significant improvement when l is large. We provide an almost optimal streaming mechanism using $\tilde{O}(k^3)$ space that computes prices. That is, the required space here is poly-logarithmic in the number of buyers and the number of copies of each item type. The following theorem states our result for the social-welfare maximization problem.

Theorem 4.3. *Let ϵ be an arbitrary small constant. There exists a streaming mechanism for the social-welfare maximization problem which with high probability gives an envy-free $(1-\epsilon)$-approximate solution using $\tilde{O}(k^3)$ space. This mechanism only remembers the prices.*

The following theorem extends our results to the revenue maximization problem as well. Note that, again here the required space is poly-logarithmic in the number of buyers and the number of copies of each item type.

Theorem 4.4. *Given an envy-free α-approximation mechanism for the revenue maximization problem, and any small constant ϵ, there exists a streaming mechanism for the revenue maximization problem which with high probability gives an envy-free $(1-\epsilon)\alpha$-approximate solution using $\tilde{O}(k^3)$ space. This mechanism only remembers the prices.*

To show that the approximation in Theorem 4.3 is necessary, we prove there is no streaming mechanism to find the prices that maintain the optimal social-welfare using space sublinear in l.

Theorem 4.5. *There is no envy-free streaming mechanism that finds the welfare-optimal envy-free prices using space $o(l)$. This bound holds even for $k = 2$.*

As with welfare maximization, any algorithm that computes revenue-optimal envy-free prices would require space that is at least linear in l.

Theorem 4.6. *There is no envy-free streaming mechanism which finds the set of prices that maximize the revenue using a space sublinear in l. This bound holds even for $k = 2$.*

4.1.2 Related Work

In this chapter we focus our attention on the problem of finding envy-free prices for unit-demand bidders in the streaming setting, a problem that has received much attention in the static setting. The revenue-maximizing envy-free pricing problem was introduced by Guruswami et al. [87]. There has since been a significant line of work attacking variants of this problem [89–91], and mounting evidence suggests that it is computationally hard to obtain better than a polylogarithmic approximation for general unit-demand bidders [92, 93]. For welfare maximization, it is well-known that a Walrasian equilibrium corresponds to a set of envy-free prices that optimizes welfare, and such an equilibrium always exists for unit-demand bidders. Moreover, in the static setting such prices can be found in polynomial time [94, 95]. Our focus is on developing streaming algorithms for these problems.

Our motivation of determining prices from sampled data relates to a recent line of literature on the sample complexity of pricing problems and applications of statistical learning theory. Much of this work has focused on the problem of learning an approximately revenue-optimal reserve price in a single-item auction [83, 84, 96, 97]. More generally, statistical learning methods have been used to quantify the sampling complexity of learning approximately optimal auctions, in the prior-free context by Balcan et al. [82] and in a prior-independent setting by Morgenstern and Roughgarden [86].

Hsu et al. [85] study the genericity of market-clearing prices learned from sampled data, and demonstrate that under some conditions on buyer preferences (including the unit-demand case studied here) prices computed from a large dataset will approximately clear a "similar" market; that is, one where buyer preferences are drawn from a the same underlying distribution.

Our technical results build upon recent work in the streaming algorithms literature on maximum matching. Chitnis et al. [88] consider the matching problem in the streaming setting and provide optimum solutions to both vertex cover and matching in $\tilde{O}(k^2)$ space, where k is the size of the solution. In addition, they show that any streaming algorithm for the maximum matching problem requires $\Omega(k^2)$ space. Later, they extend this result to dynamic streams in which we have both addition and deletion of edges [98].

McGregor [99] considered the matching problem in the streaming setting with several passes. He provides a $(1-\epsilon)$-approximation algorithm for unweighted graphs and a $(0.5-\epsilon)$-approximation algorithm for weighted graphs, both with constant number of passes and using $\tilde{O}(n)$ space.

Kapralov et al. [100] provide a streaming algorithm that estimates the size of a maximum matching in the random order setting (i.e., the graph is chosen by an adversary, but the order of arrival of edges is chosen uniformly at random among all permutations). They provide a ploylogarithmic approximation of the size of maximum matching, using a polylogarithmic space.

Later, Esfandiari et al. [101] consider the maximum matching problem in planer graphs and bounded arboricity graphs. They provide a constant approximation of the size of a maximum matching in these graphs using $\tilde{O}(n^{2/3})$ space in the streaming setting. Later, simultaneously Bury et al. [102] and Chitnis et al. [98] extend this algorithm to work for both addition and deletion of edges using a larger space of $\tilde{O}(n^{4/5})$.

4.2 Pricing problem: Maximizing Social Welfare

In this section, we consider the problem of assigning prices to items, and items to buyers in a streaming setting such that the assignment would be envy-free, and the *social welfare* is maximized. The social welfare would be sum of the weights (or utilities) of the assigned edges. In Subsection 4.2.1, we propose the optimum mechanism with $O(k^2l)$ space, and in Subsection 4.2.2 we approximate the optimum mechanism with improved memory.

4.2.1 Envy-Free Mechanism with $O(k^2l)$ Space

In this subsection, we propose a pricing mechanism to maximizing the social welfare in our setting. As we explained earlier, we only use $O(k^2l)$ memory for storage of the stream of edges. Our approach is to store the $kl + 1$ edges with maximum weight for each item, and to run the optimum algorithm to find the social welfare maximizing envy-free assignment in offline setting when the stream ends. We call the optimum streaming algorithm of this subsection SWM to use it in Subsection 4.2.2.

Let G be a weighted bipartite graph with k vertices corresponding to the k item types on one side, and n vertices corresponding to the buyers on the other side. The weight of the edges denote the utilities of buyers for item types. Each item type, can be sold to at most l buyers. In other words, there are l copies of each item type available for sale. Let G' be the graph constructed from graph G, such that for each item we only keep kl edges with maximum weight (breaking ties arbitrarily), and remove the rest of edges from the graph. Note that a feasible solution matches each item to at most l buyers. Here, we slightly abuse the notation and simply call this structure a matching.

The next lemma shows that removing these edges does not affect the weight of the maximum matching.

Lemma 4.1. *The value of the maximum weighted matching in G is equal to the value of the maximum weighted matching in G'.*

Proof. Let M' be a maximum weighted matching in G', and M be the maximum weighted matching in G which has the maximum number of intersecting edges with G'. Since removal of edges from a graph does not increase the weight of the maximum matching, it is clear that the weight of M is not less than the weight of M'. It remains to show that the reverse is also true. If M only contains edges present in G', then clearly the weight of M and M' would be equal. Suppose in contrast that M contains an edge $e = (v_i, b_j)$ (between buyer b_j and item v_i) which does not belong to G'. Note that M has exactly kl edges and therefore, one of the neighbors of item v_i in G' must be unassigned by matching M. Let $b_{j'}$ be this neighbor. Remove edge e from M and replace it with edge $e' = (v_i, b_{j'})$. We know that the weight of edge e' is bigger than or equal to the weight of e. In case the weight of e is bigger than weight of e', the new matching is bigger than M. On the other hand, if the weights are equal, we have another maximum weighted matching in G which shares more edges with the edges of G'. Both cases lead to a contradiction with the assumption about M. Therefore, the weight of maximum matchings in both graphs should be equal. □

In the next step, we build another graph H from graphs G and G' as follows. Starting from graph G', we build H by adding a dummy buyer vertex d_i for each vertex v_i corresponding to an item type in G. The weight of the edge between d_i and v_i denoted by u_{v_i,d_i} would be equal to the weight of $(kl+1)$-th maximum weight edge connected to item v_i in graph G.

Lemma 4.2. *There is a maximum weighted matching in H which does not include any dummy buyer vertices.*

Proof. Note that G' can be viewed as a subgraph of graph H in which for each item type we only keep kl edges connected to it with maximum weight. The rest follows from Lemma 4.1. □

Finally, we present our pricing and allocation rule. Recall that we keep graph G' and graph H using $O(kl^2)$ available memory while the edges are being streamed. By Lemma 4.2, we know that there exists a maximum weighted matching M in graph H which is a subgraph of graph G' (does not contain any of dummy vertices in graph H). Consider a minimum weight vertex cover C corresponding to this matching such that the values assigned to all dummy vertices would be zero (Since dummy vertices are not matched by M, such a vertex cover can be found). For a vertex z, let c_z denote the value assigned to z by vertex cover C. We set the price of item v_i to its value in our vertex cover c_{v_i}, and assign it to its matched buyer in M. Since the social welfare in this case would be the weight of M which is a maximum weighted matching, it is clear that our algorithm maximizes social welfare. It is left to show that our algorithm produces an envy-free assignment.

Lemma 4.3. *Our assignment algorithm is envy-free.*

Proof. Suppose for contradiction our assignment algorithm is not envy-free, and there exists a buyer b_j who prefers item v_q over item v_i that our algorithm assigned to her. Consider the dummy buyer vertex d_q that we added and connected to item v_q in graph H. As we argued, our chosen vertex cover C, sets $c_{d_k} = 0$ for every dummy vertex d_k. Therefore, the value assigned to item v_q by our vertex cover must be at least the weight of the edge between v_q and d_q, i.e, $c_{v_q} \geq u_{v_q,d_q}$. Also, recall that our algorithm sets the price of item v_q to c_{v_q}. Hence, since buyer b_j prefers item v_q, her profit for this item must be non-negative, i.e, $u_{v_q,b_j} - c_{v_q} \geq 0$. From the above two inequalities we have $u_{v_q,b_j} \geq u_{v_q,d_q}$ which means the weight of the edge between buyer b_j and item v_q is bigger than the weight of the $kl+1$-th biggest edge connected to item v_q. From this statement we can argue that graph G' contains the edge (b_j, v_q), and thus $c_{b_j} + c_{v_q} \geq u_{v_q,b_j}$. Furthermore, since the edge (b_j, v_i) belongs to the maximum weighted matching, we must have

$$c_{b_j} + c_{v_i} = u_{v_i,b_j} \Rightarrow u_{v_i,b_j} - c_{v_i} = c_{b_j} \geq u_{v_q,b_j} - c_{v_q}$$

This is a contradiction to the fact that buyer b_j prefers item v_q to item v_i assigned to her by our algorithm. □ The following theorem is the main result of this subsection.

Theorem 4.7. *Our streaming assignment algorithm which assigns prices to items and items to buyers in the aforementioned market is an envy-free social welfare maximizing assignment, and it uses $O(kl^2)$ memory.*

Proof. As we discussed above, we only keep graphs G' and H which have $O(k^2l)$ edges. As we explained in this subsection, our algorithm suggest an assignment that maximizes social welfare. Furthermore, by Lemma 4.3 this assignment is also envy-free. □

4.2.2 Improving Space Efficiency While Approximating Social Welfare

In this subsection, we try to improve space efficiency in the problem solved in Subsection 4.2.1, when we relax the goal of achieving maximum social welfare to obtaining an approximation of it. More specifically, suppose we have k item types, l available items of each type, and n buyers, and the utilities of buyers for item types are revealed in a streaming fashion. Recall that we can find the social welfare maximizing prices for the items and an assignment of items to buyers in $O(k^2l)$ available memory. In this subsection, our goal is to find prices for item types when the amount of available memory is independent of l (the number of available items of each item type). We prove when each buyer picks the most profitable item based on the prices that our algorithm suggests and his own utilities, there would be no more than l requests for any of the item types with high probability, and the social welfare would be a good approximation of the optimum social welfare. Thus, we can conclude this self selection of items by customers is envy-free and valid with high probability, and we do not have to deal with item to buyer allocations after setting the prices. Our approach here is to collect a sample of buyers while the data is being streamed, decide the prices based on this sample, and prove that these prices would yield a good approximation of the social welfare and an envy-free assignment of items to buyers in the original graph while the assignment of items is done by the buyers themselves and not by us. This algorithm is especially favorable over previous ones when the number of different item types is relatively small compared to the total number of items. In other words, when k is small compared to l.

Let B be the set of our n buyers, and V the set of k item types. Assume we have l available items of each type. Let G be the weighted bipartite graph of buyers and item types showing utilities of buyers for items. For arbitrary constants $\delta, \epsilon > 0$, our algorithm finds prices of items in V such that the greedy item picking strategy by buyers would yield a valid envy-free assignment and achieves a social welfare that is $(1-2\epsilon)$-approximation of the maximum possible social welfare with probability $1-\delta$. We define a new parameter $t = 3\ \frac{-log(\delta)+log(2k)+klog(n)}{\epsilon^2}$, and sample every buyer in B with probability $\frac{t}{l}$. Let B' be the set of buyers chosen in our sampling, and G' be the induced subgraph of G when we remove all the vertices that are not in B'. We assume there are $(1-\epsilon)t$ available copies from each item type in graph G' which can be sold to the buyers in B'. As we discussed in previous sections, we can find the optimal prices for items in B' to achieve maximum Social welfare in graph G' in $O(k^2t)$ available memory. After this step, we use the same prices for the general case, and prove that these prices along with the greedy item selection by buyers satisfies the aforementioned criteria.

Once the prices are determined by our algorithm, we let each buyer pick his own profit maximizing item to accomplish an envy-free assignment. We use a com-

Algorithm 4

Input: Weighted bipartite graph G with set of vertices $B \cup V$, l number of available items from each item type, and constants $\epsilon, \delta > 0$.

Output: Price vector $\vec{p}$ which yields a $(1 - 2\epsilon)$-approximation of opt SW and an envy-free assignment with probability $1 - \delta$.

```
 1: $t \leftarrow 3\ \frac{-log(\delta)+log(2k)+klog(n)}{\epsilon^2}$
 2: $B' \leftarrow \emptyset$
 3: for $b \in B$ do
 4:     Add $b$ to $B'$ with probability $\frac{t}{l}$
 5: end for
 6: Let $G'$ be the subgraph of $G$ induced by $B' \cup V$
 7: $l' \leftarrow t(1 - \epsilon)$ be the number of available items of each item type in $G'$
 8: Upon stream of edges in $G$, ignore any edge $e \notin G'$
 9: Find optimal $\vec{p}$ using SWM Algorithm (Subsection 4.2) on edges of $G'$
10: Return $\vec{p}$
```

bination of Chernoff and union bounds in the following two lemmas to prove the probability that this assignment is invalid is small. More formally, we show that when buyers choose items greedily, the probability that the number of requests for each item is within $(1-2\epsilon)l)$ and l is at least $1-\delta$.

Lemma 4.4. *Fix the prices of all available item types in V. Sample the buyers by choosing each buyer with probability q. Let x_i be the number of buyers in our sample who prefer item i over any other item. Let y_i be the number of buyers who prefer item i in the original setting with all buyers. Then assuming $x_i \leq ql$ we have*

$$Pr(|y_i - \frac{x_i}{q}| > \epsilon l) \leq 2exp(-\frac{\epsilon^2 ql}{3})$$

Proof.

$$Pr(|y_i - \frac{x_i}{q}| > \epsilon l) = Pr(|y_i q - x_i| > \epsilon ql)$$

We need to apply Chernoff bound to find an upperbound on the above probability. Here $y_i q$ can be thought of as a guess for x_i. Therefore, by letting $\zeta = \frac{\epsilon ql}{x_i}$ and $\mu = x_i$ in the Chernoff bound given by the inequality $Pr(\hat{x} - \mu) \leq 2exp(-\frac{\zeta^2}{3}\mu)$ we have

$$Pr(|y_i q - x_i| > \zeta\mu) \leq 2exp(-\mu\frac{\zeta^2}{3}) = 2exp(\frac{-\epsilon^2 q^2 l^2}{3x_i}) \leq 2exp(-\frac{\epsilon^2 ql}{3}).$$

Where the last inequality follows from the assumption that $x_i \leq ql$. $\square$

Lemma 4.5. *Let y_i be the number of buyers who prefer item i over any other item in the original setting with prices set by Algorithm 4. Then we have*

$$Pr(|y_i - (1-\epsilon)l| > \epsilon l) \leq 2exp(-\frac{\epsilon^2 t}{3})$$

Proof. Algorithm 4 chooses each item with probability $\frac{t}{l}$ and sells $(1-\epsilon)t$ of each item to the sampled buyers for an envy-free assignment in G'. Therefore, for a fixed pricing and a fixed item $i \in [k]$, we can use Lemma 4.4 with parameters $q = \frac{t}{l}$ and $x_i = (1-\epsilon)t$.

$$Pr(|y_i - \frac{(1-\epsilon)t}{\frac{t}{l}}| > \epsilon l) \leq 2exp(-\frac{\epsilon^2 \frac{t}{l} l}{3}) \Rightarrow Pr(|y_i - (1-\epsilon)l| > \epsilon l) \leq 2exp(-\frac{\epsilon^2 t}{3})$$

$\square$

Lemma 4.6. *When the prices are set by Algorithm 4 and the buyers greedily pick the best item for themselves, the probability that the number of requests for each item is between $(1-2\epsilon)l$ and l is at least $1-\delta$. Note that this also means the greedy selection of items by the buyers would yield a valid assignment with probability at least $1-\delta$.*

Proof.

From Lemma 4.5 we have

$$Pr(y_i > l) + Pr(y_i < (1-2\epsilon)l) \leq 2exp(-\frac{\epsilon^2 t}{3})$$

Let $\mathcal{P}$ denote the set of all possible price vectors for the item types in V that Algorithm 4 might return. Since there are k item types and n reasonable price values for each item type $|\mathcal{P}| < n^k$. Let $A_{\vec{p},i}$ be the event that with price vector $\vec{p}$, the number of requests for item i is more than l or less than $(1-2\epsilon)l$. Then

$$Pr(A_{\vec{p},i}) = Pr(y_i > l) + Pr(y_i < (1-2\epsilon)l) \leq 2exp(-\frac{\epsilon^2 t}{3})$$

Furthermore, using Union Bound we show that the probability of any of the events $A_{\vec{p},i}$ happening for all possible price vectors $\vec{p} \in \mathcal{P}$ and all item types $i \in [k]$ is

$$\bigcup_{\vec{p}\in\mathcal{P}} \bigcup_{i\in[k]} A_{\vec{p},i} \leq 2\ k\ n^k\ exp(-\frac{\epsilon^2 t}{3}) \leq \delta$$

Where the last inequality follows from the choice of t. □

Lemma 4.7. *Consider the price vector $\vec{p}$ returned by Algorithm 4 combined with the greedy selection of items by the individual buyers. If every item type is requested by the buyers between $(1-2\epsilon)l$ and l times then the social welfare of this assignment is within $(1-2\epsilon)$-approximation of the maximum possible social welfare.*

Proof. Let H be a bipartite graph with the set of buyers B on the left side and l vertices for each item type in V on the right side. The weight of the edges between a buyer vertex and an item would be the utility of the buyer for that item type. Let H' be a similar bipartite graph with set of buyers on the left side and $(1-2\epsilon)l$ vertices for each item type in V on the right side. Then the weight of the maximum weighted matching in H' is at least $(1-2\epsilon)$ of the weight of the maximum weighted matching in H.

Consider the greedy selection of items by the buyers after we run Algorithm 4 and set prices for item types. Suppose every item type is sold between $(1-2\epsilon)l$ and l times. Remove some buyers so that every item is picked by exactly $(1-2\epsilon)l$

buyers. The remaining edges form a matching in graph H'. It is left to show that this matching is a maximum weighted matching in H'. For this purpose, we build a vertex cover C of same weight for the matching.

For every buyer remaining in the graph, let its vertex cover value be the profit (utility minus price) that he makes from his selected item. For each item, let its vertex cover value be the suggested price by Algorithm 4 for its item type. These values in the vertex cover would clearly cover every edge in the matching. Let b_i be a buyer in B. Let v_j be the item type that b_i chooses and v_h be another item type. We want to show that C covers the weight of the edge between b_i and a vertex of the item type v_h. Note that $u_{b_i,v_j} - p_{v_j} > u_{b_i,v_h} - p_{v_h}$. The value of C for buyer b_i is $u_{b_i,v_j} - p_{v_j}$ and the value of C for any vertex of item type v_j is p_{v_j}. Therefore, $u_{b_i,v_j} - p_{v_j} + p_{v_h} > u_{b_i,v_h}$. Thus, our algorithm produces a maximum weighted matching in H'. □

Theorem 4.8. *The pricing suggested by Algorithm 4 along with the greedy selection of items by the buyers yields a valid envy-free assignment and a* $(1-2\epsilon)$*-approximation of the maximum possible social welfare with probability* $1-\delta$*. The space needed by Algorithm 4 is independent of* l*, the number of available items of each item type.*

Proof. Since Algorithm 4 runs the social welfare maximizing algorithm of Subsection 4.2 on a sample of buyers assuming there are $(1-\epsilon)t$ available item of each type, only $O(k^2 t)$ memory is needed for finding the optimal price vector $\vec{p}$ of graph G'. Here t is a function of k, n, δ and ϵ and does not depend on l. Lemma 4.6

guarantees the number of requests for each item type does not exceed the number of available items of that type with probability at least $1-\delta$ and Lemma 4.7 proves the social welfare in this case would be within $(1-2\epsilon)$-approximation of the maximum possible social welfare. □

4.3 Pricing problem: Maximizing Revenue

Just like the previous section, we try to find a pricing for items, and an envy-free assignment of items to buyers in our described market when the input is revealed in a streaming fashion. However, in this section, we aim to maximize *revenue* instead of social welfare. In Subsection 4.3.1, we propose the optimum mechanism with $O(k^2 l)$ space, and in Subsection 4.3.2 we approximate the optimum mechanism with improved memory.

4.3.1 Envy-Free Mechanism with $O(k^2 l)$ Space

In this subsection, we show that if we are given an envy-free α-approximation mechanism for the revenue maximization problem then we can have an envy-free α-approximation mechanism for the revenue maximization problem in the streaming setting with $O(k^2 l)$ available memory. We call this mechanism designed for the streaming setting RM for use in the later subsection.

First, we construct graph G' from graph G by keeping only the $kl+1$ largest edges connected to each item while the edges are being streamed.

Let M be the envy-free assignment in G which yields the maximum revenue and has the maximum number of intersecting edges with the edges of G'. If we prove that G' includes all the edges in M, then we can say M is a feasible envy-free assignment in G' and conclude that the optimum solution in G' is at least as good as the optimum solution in G. The following lemma shows this property.

Lemma 4.8. *G' includes all the edges of M.*

Proof. Suppose for contradiction M has at least one edge $e = (v_i, b_j)$ which is not present in G'. Since item v_i has $kl + 1$ neighbors in G', and we have k item types with l available copies of each item, according to the pigeonhole principle v_i has at least one neighbour like buyer b_q who has not bought any items. Let e' be the edge between b_q and v_i. Since G' has the $kl + 1$ largest edges of G for each item type, the weight of e is no more than the weight of e'. If the weight of e is equal to the weight of e', buyer b_j can be replaced by buyer b_q in M for an envy free revenue maximizing assignment in G that has more intersecting edges with G' than M resulting in a contradiction with the choice of M. On the other hand, if the weight of e is less than the weight of e', buyer b_q is envious of the outcome for buyer b_j which is a contradiction to envy-freeness of M. Therefore, G' includes all the edges in M. □

Now, we know that the optimum solution in G is a feasible solution in G'. It is left to show that the optimum solution in G' is feasible in G, to be able to conclude that the optimum solution in G' is also optimum in G. Let M' be the optimum (revenue maximizing) envy-free assignment in G'. We prove that M' is an envy-free assignment in G.

Lemma 4.9. *The optimum envy-free assignment in G' denoted by M' is an envy-free assignment in G.*

Proof. Suppose by contradiction that M' is not an envy-free assignment in G. Then, G has a buyer like b_j who is envious of item v_i bought by buyer b_q. Since M' is an envy-free assignment in G', the edge $e = (v_i, b_j)$ does not belong to G'. Item v_i has $kl + 1$ neighbors in G', and there are kl available items for purchase. Hence, according to the pigeonhole principle, v_i has at least one neighbor like b_p who has not bought any items in M'. Since G' has the $kl + 1$ largest edges for each item node, the weight of the edge between v_i and b_p is not less than the weight of e which means b_p would be envious of the item that b_j has bought which is a contradiction to the envy-freeness of M' in G'. Thus, M' should also be an envy-free assignment in G. $\square$

The following theorem is a corollary of the above lemmas and is the main result of this subsection.

Theorem 4.9. *Given an envy-free α-approximation mechanism for the offline revenue maximization problem, there exists an envy-free α-approximation mechanism for the revenue maximization problem in the streaming setting using $O(k^2 l)$ space.*

Proof. We construct graph G' in the $O(k^2 l)$ available memory while the edges of graph G are being streamed. According to Lemma 4.9, the optimum assignment in G' is a feasible assignment in G. Furthermore, due to lemma 4.8 we know that the optimum solution in G' is at least as good as the optimum solution in G. Hence, keeping the edges of graph G' while the edges of G are being streamed is enough for finding the optimum assignment in G. It is clear that given an envy-free mechanism that approximates maximum revenue in offline graph G' within a factor α we can use the exact mechanism as an α-approximation for the revenue maximization problem in graph G. $\square$

4.3.2 Improving Space Efficiency While Approximating Optimum Revenue

In Subsection 4.3.1, we introduced a simple streaming mechanism (RM) that finds the price vector and an envy-free assignment of items to buyers to α-approximate maximum *revenue* given a mechanism that α-approximate the maximum revenue in the offline case. In this subsection, we are concerned with reducing the amount of space used by our streaming algorithm. As we mentioned earlier, $O(k^2 l)$ available space is needed for any streaming algorithm that finds a revenue maximizing assignment in our setting. Just like the previous section, we are interested in a streaming algorithm for which the amount of space used is independent of l, the number of copies of each available item type. Algorithm 5 is our algorithm for this purpose. As a result of reduction in the required memory, the revenue of the assignment given by our algorithm loses another $(1 - 2\epsilon)$ approximation factor compared to the maximum possible revenue. When l is small compared to k, this algorithm would be beneficial since it dramatically improves the amount of space used. The algorithm and some of the proofs are similar to the ones in the previous section.

Algorithm 5

Input: Weighted bipartite graph G with set of vertices $B \cup V$, l number of available items from each item type, and constants $\epsilon, \delta > 0$.

Output: Price vector $\vec{p}$ which yields a $(1-2\epsilon)$-approximation of opt revenue and an envy-free assignment with probability $1-\delta$.

```
1: $t \leftarrow 3\ \frac{-log(\delta)+log(2k)+klog(n)}{\epsilon^2}$
2: $B' \leftarrow \emptyset$
3: **for** $b \in B$ **do**
4:     Add $b$ to $B'$ with probability $\frac{t}{l}$
5: **end for**
6: Let $G'$ be the subgraph of $G$ induced by $B' \cup V$
7: $l' \leftarrow t(1-\epsilon)$ be the number of available items of each item type in $G'$
8: Upon stream of edges in $G$, ignore any edge $e \notin G'$
9: Find optimal $\vec{p}$ using RM Algorithm (Subsection 4.3.1) on edges of $G'$
10: Return $\vec{p}$
```

Lemma 4.10. *Let y_i be the number of buyers who prefer item i over any other item type in the original setting with prices set by Algorithm 5. Then we have*

$$Pr(|y_i - (1-\epsilon)l| > \epsilon l) \leq 2exp(-\frac{\epsilon^2 t}{3})$$

Proof. This lemma is exactly the same as Lemma 4.5 that we proved in the previous section. □

Lemma 4.11. *When the prices are set by Algorithm 5 and the buyers greedily pick the best item for themselves, the probability that the number of requests for each item type is between $(1-2\epsilon)l$ and l is at least $1-\delta$. Note that this also means the greedy selection of items by the buyers would yield a valid assignment with probability at least $1-\delta$.*

Proof. Again this lemma is the same as Lemma 4.6 proved in the previous section and since our parameters in sampling is the same in both algorithms the proofs are exactly the same. □

So far in this section, we used exactly similar lemmas as the previous section to show our sampling approach results in an envy-free valid assignment with high probability. Next we want to show why our algorithm also yields a revenue that is a good approximation of the optimum revenue. Note that Algorithm 5 uses the revenue maximizing algorithm introduced in Section 4.2 as opposed to the social welfare maximizing Algorithm of Subsection 4.3.1 that Algorithm 4 uses, and our goal here is to maximize the revenue. Therefore, showing that our algorithm's revenue is $(1-2\epsilon)$-approximation of the maximum possible revenue is different from the previous section.

Lemma 4.12. *Fix a price vector $\vec{p}$ for all available item types in V. If $\vec{p}$ produces revenue R in G, it produces a revenue bigger than $(1-\epsilon)\frac{t}{l}R$ in G' with probability at least $1-\delta$.*

Proof. Let x_i be the number of requests for item type i in graph G' and y_i be the number of requests for the same item type in graph G when the price vector is fixed. Lemma 4.4 with parameters $q = \frac{t}{l}$ and $y_i = l$ shows $Pr(x_i < (1-\epsilon)t) < 2exp(-\frac{\epsilon^2 t}{3})$. Again, using Union Bound the same way we used in Lemma 4.5, we can show the probability that any of the x_i's is less than $(1-\epsilon)t$ is less than δ. Thus, with probability at least $1-\delta$ each item type is sold to at least $(1-\epsilon)t$ buyers in G'. Since each item type is sold to at most l buyers in G, our revenue in G' must be at least $(1-\epsilon)\frac{t}{l}$ of the revenue in G with probability $1-\delta$. □

Lemma 4.13. *Consider the price vector $\vec{p}$ returned by Algorithm 5 along with the greedy selection of items by the individual buyers. If the number of requests for every item type is at least $(1-2\epsilon)l$ and at most l, then the revenue of this assignment is within $(1-2\epsilon)$-approximation of the maximum possible revenue with probability at least $1-\delta$.*

Proof. Let $Rev_A(G)$ denote the revenue of the price vector given by algorithm A on graph G and $Rev_A(G')$ be the revenue that the same price vector produces on the sampled graph G'. Note that in case every item type is requested by the buyers at least $(1-2\epsilon)l$ times, we have

$$Rev_{Alg5}(G') \leq \frac{l'}{(1-2\epsilon)l} Rev_{Alg5}(G) = \frac{(1-\epsilon)t}{(1-2\epsilon)l} Rev_{Alg5}(G)$$

. Let OPT be the optimum offline revenue maximizing assignment for the specific graph G. Recall that the price vector chosen by Algorithm 5 maximizes the revenue in G' and thus

$$Rev_{OPT}(G') \leq Rev_{Alg5}(G')$$

. Furthermore, due to Lemma 4.12, we have

$$\frac{(1-\epsilon)t}{l} Rev_{OPT}(G) \leq Rev_{OPT}(G')$$

with probability at least $1-\delta$. Combining the above three inequalities proves $(1-2\epsilon)Rev_{OPT}(G) \leq Rev_{Alg5}(G)$ with probability at least $1-\delta$. □

Theorem 4.10. *The pricing suggested by Algorithm 5 along with the greedy selection of items by the buyers yields a valid envy-free assignment and a $(1-2\epsilon)\alpha$-approximation of the maximum possible revenue with probability at least $1-2\delta$ given an envy-free mechanism that α-approximates maximum revenue in the offline case. The space needed by Algorithm 5 is independent of l, the number of available items of each item type.*

Proof. Lemma 4.11 shows the greedy selection of items by buyers after running Algorithm 5 results in a valid and envy-free assignment with probability at least $1-\delta$. Lemma 4.13 shows in case the assignment is valid, the revenue is within $(1-2\epsilon)$-approximation of the maximum revenue with probability $1-\delta$. Thus, with probability $1-2\delta$ the Algorithm 5 results in a valid envy-free assignment and approximates the maximum revenue within a $(1-2\epsilon)\alpha$ factor. Here α is added to the approximation factor since RM gives an α-approximation for maximum social welfare in the streaming setting, given an α-approximation mechanism for the offline case. $\square$

4.4 Hardness Results

In Sections 4.2 and 4.3, we provided optimum as well as approximation algorithms for envy-free pricing problems to maximize social welfare or revenue in the streaming setting. In this section, we provide lower bounds on the required spaces to solve these problems optimally. To provide these hardness results we use the communication hardness of set disjointness problem.

4.4.1 Hardness of Social Welfare Maximization

In Section 4.2, we presented a streaming algorithm which finds an envy-free social welfare maximizing assignment of prices to items and items to buyers using $O(k^2 l)$ memory, where k is the number of item types and l is the number of available items of each type. This result raises the following interesting question. Is it necessary to have $\Omega(l)$ available memory for solving this problem? In other words, if the number of item types is small compared to the total number of items (or k is small compared to l, can we solve the problem in space independent of l? In this section we prove for any constant $\epsilon > 0$, no streaming algorithm can ϵ-approximate the envy-free social welfare maximizing *prices* in $o(l)$ space. The proof is done via a reduction from *Disjointness*, a well-known communication complexity problem .

Definition 4.1. Disjointness Problem *is a communication complexity problem in* *which Alice is given a string $x \in \{0,1\}^n$ and Bob is given a string $y \in \{0,1\}^n$. Their goal is to decide whether there is an index i, such that $x_i = y_i = 1$. Index i in this case is called an* intersection*. It is known that the minimum number of bits required to be exchanged between Alice and Bob to find an intersection is $\Omega(n)$ bits even with multi-passes allowed.*

Theorem 4.11. *For any arbitrary small constant $\epsilon > 0$, there is no streaming algorithm which uses $o(l)$ space and ϵ-approximates all the item prices of the social welfare maximizing price vector.*

Proof. For an arbitrary ϵ, assume for the sake of contradiction there exists an algorithm A which can find an ϵ-approximation of an optimal pricing in $o(l)$ space. We show a reduction from any instance of Disjointness problem to an instance of our market design problem such that if Algorithm A exists, Disjointness problem can be solved using $o(l)$ space.

Let $\mathcal{I}_1$ be an instance of Disjointness problem with $x \in \{0,1\}^l$ as Alice's string and $y \in \{0,1\}^l$ as Bob's string. A corresponding instance of our market design problem $\mathcal{I}_2$ can be built as follows. Consider two item types in $\mathcal{I}_2$, one corresponding to Alice and one corresponding to Bob. Suppose each of these two item types have $2l$ copies available. Let $G = (V_1, V_2, E)$ be the bipartite graph of item types and buyers in instance $\mathcal{I}_2$, with $V_1 = \{v_{Alice}, v_{Bob}\}$ as the item type vertices. We start with $2l$ buyer vertices $V_a = \{a_1, a_2, \dots, a_l\}$ and $V_b = \{b_1, b_2, \dots, b_l\}$ in V_2. For any index i, if $x_i = 1$, we connect v_{Alice} to both vertices a_i and b_i. Similarly, for any index i such that $y_i = 1$, we connect v_{Bob} to both vertices a_i and b_i. Let I_x be the set of

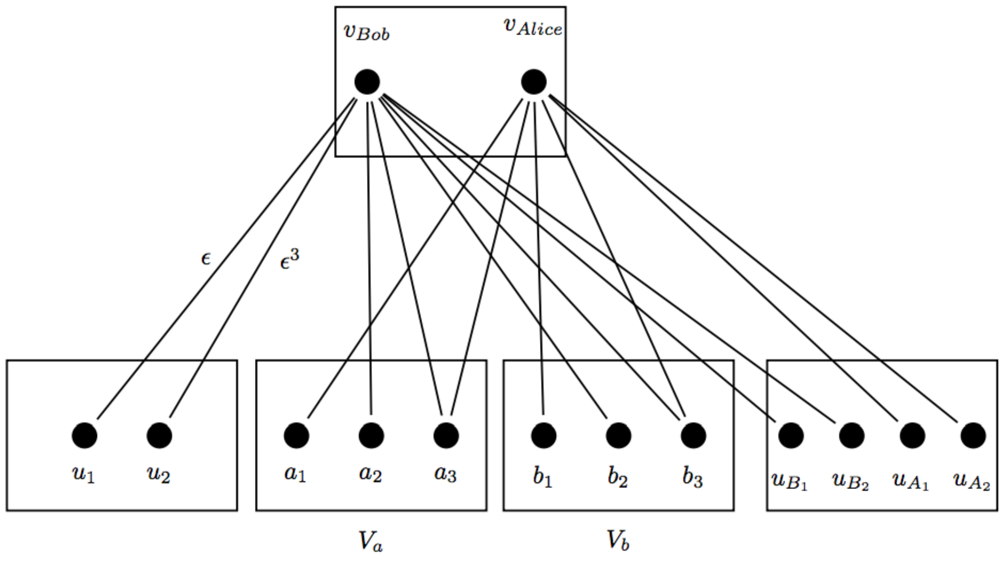


Figure 4.1: Here, $x = \{1, 0, 1\}$ and $y = \{0, 1, 1\}$. The weight of the first and the second edges are ϵ and ϵ^3 as they are labeled, and the weight of all other edges are one.

all indices j such that $x_j = 1$ in string x. We add a set U_{Alice} with $2l - |I_x|$ buyer vertices to V_2, and connect v_{Alice} to all vertices in U_{Alice} so that the degree of v_{Alice} would be exactly $2l$. Similarly, we add a set of vertices U_{Bob} with $2l - |I_y|$ buyer vertices to V_2 and connect v_{Bob} to all vertices in U_{Bob}. Finally, we add two buyer vertices u_1, u_2 to V_2 and connect v_{Bob} to both of them. Note that the set of buyer vertices V_2 would be $\{u1, u2\} \cup V_a \cup V_b \cup U_{Alice} \cup U_{Bob}$. The utility of buyers u_1 and u_2 for Bob's item are ϵ and ϵ^3 respectively. The utility of any other buyer for any other item connected to it (v_{Alice} or v_{Bob}) would be 1. In other words, the weight of the edge (v_{Bob}, u_1) is ϵ, the weight of the edge (v_{Bob}, u_2) is ϵ^3, and the weight of all the other edges in E is 1. (See Figure 4.1 as an example.)

Now suppose both Alice and Bob know about algorithm A. Let E_{Alice} be the set of edges connected to v_{Alice}, and E_{Bob} the set of edges connected to v_{Bob} in graph G. Note that Alice only knows about E_{Alice} and buyer vertices $V_2 \backslash (\{u_1, u_2\} \cup U_{Bob})$. Similarly, Bob only knows about E_{Bob} and buyer vertices $V_2 \backslash U_{Alice}$. Alice starts streaming her edges and running algorithm A on it. She would send the information that algorithm A stores in $o(l)$ available space to Bob. Bob at the other end receives all the information stored by algorithm A and sent by Alice, and continues running algorithm A by streaming his own edges. Algorithm A can find social welfare maximizing prices for both Alice and Bob items in $o(l)$ space. The algorithm finishes at Bob's end after he streams all of his edges. At this point we claim that Bob can decide whether the strings have intersection or not based on the following two case. If the price suggested by algorithm A for his item type is less than ϵ, Bob should declare no intersections exist and if the price is above ϵ^2 he should declare existence of at least one intersection. Furthermore, algorithm A would never set a price between ϵ and ϵ^2 for v_{Bob}. Next we prove why this claim is valid.

Suppose Alice and Bob's strings have no intersections. Then in graph G, no buyer is connected to both Alice and Bob. That is v_{Bob} is connected to $2l$ buyer vertices with utility 1 for his item and none of his buyers want Alice's item. Alice's item is also connected to $2l$ buyer vertices that do not want Bob's item. The optimal prices for both Alice and Bob's items to maximize social welfare would be 1 in this case, and no item would be sold to buyers u_1 and u_2.

On the other hand, if the strings have at least one intersection, say at index i, both v_{Alice} and v_{Bob} would be connected to a_i and b_i. To maximize social welfare in this case, Alice would sell all of her items to the $2l$ buyers who want her item at price 1, and Bob can sell at most $2l-2$ items to those who want his item at price 1 and has to sell two items to buyers u_1 and u_2 who want to pay ϵ and ϵ^3 for his item respectively. Therefore, the price for Bob's item should be ϵ^3. Since the goal is to maximize social welfare, Bob cannot decide to leave out u_1 and u_2 and sell $2l-2$ items at price 1 to the buyers whose utility for his item is 1.

Due to our assumption, algorithm A can ϵ-approximate all optimal prices for social welfare maximization while Alice and Bob stream the edges using only $o(l)$ available space. Specifically, if the optimal price for v_{Bob} is 1, i.e, there is no intersection in the two strings, algorithm A would set a price higher than ϵ for v_{Bob}. Otherwise, in case the optimal price for v_{Bob} is ϵ^3, algorithm A would set a price lower than ϵ^2 for Bob's item. These two prices are the only optimal prices for Bob's item and thus, only these two cases exist. Hence, Bob can distinguish between these two cases by checking the price set for his item once the algorithm ends. Any price less than ϵ^2 corresponds to an intersection, and any price higher than ϵ signals no intersection between the strings. □

4.4.2 Hardness of Revenue Maximization

In Subsection 4.4.1, we presented a hardness proof to show no streaming algorithm exists to approximate the optimal prices using $o(l)$ available space in our market design problem with the goal of *social welfare maximization.* In this subsection, we establish a hardness result for the case that our goal is to maximize the revenue, however, the result of this subsection does not involve any approximation. That is we only guarantee there exists no algorithm which finds the exact optimal prices for revenue maximization market design problem in $o(l)$ space. Just like previous subsection, our approach is based on a reduction from *Disjointness problem*.

Theorem 4.12. *There is no streaming algorithm which uses $o(l)$ memory, and finds the revenue maximizing price vector for an envy free assignment.*

Proof.

Suppose for contradiction, we have an algorithm A which finds an envy-free assignment and the revenue maximizing price vector using $o(l)$ available space. Let $x \in \{0,1\}^l$ be Alice's string and $y \in \{0,1\}^l$ be Bob's string. Consider two item types one corresponding to Alice and one corresponding to Bob. Suppose each of these two item types have l copies available. Let $G = (V_1, V_2, E)$ be the bipartite graph of item types and buyers, with $V_1 = \{v_{Alice}, v_{Bob}\}$ as the item type vertices. Consider three sets of buyers $V_a = a_1, a_2, \ldots, a_l$, $V_b = b_1, b_2, \ldots b_{l-|I_x|}$ and $V_c = c_1, c_2, \ldots, c_{l-|I_y|}$. For any index i, if $x_i = 1$, buyer a_i has utility one for Alice's item. Similarly, for each index i such that $y_i = 1$, buyer a_i has utility one for Bob's item. Moreover, suppose all buyers in V_b have utility 1 for Alice's item and all buyers in V_c have utility 1 for Bob's item. Finally, we add buyer v_1 with utility $1 - \frac{1}{2l}$ for Bob's item.

Alice knows all the buyers connected to her item with their utilities but she is not aware of the utilities of edges connected to Bob's item. She starts running algorithm A while streaming her edges. Once the stream of Alice's edges ends, she sends the information that algorithm A stores in $o(l)$ available space to Bob. Bob at the other end only knows about the buyers and the utility of edges connected to his own item. He also receives all the information stored by algorithm A and sent by Alice. Bob continues running algorithm A by streaming his own edges as A's input. Algorithm A finds the revenue maximizing prices for both Alice and Bob's items in $o(l)$ space. The algorithm finishes at Bob's end after he streams all of his edges. At this point we claim that Bob can decide whether the strings have an intersection or not based on the following two cases. If the price suggested by algorithm A for his item type is 1, Bob should declare no intersections exists and if the price is $1 - \frac{1}{2l}$ he should declare existence of at least one intersection. Next we prove validity of this claim.

In case x and y have no intersections, both Alice and Bob's items are connected to exactly l disjoint vertices. The optimal price for both item types is 1 and the optimal revenue would be $2l$. On the other hand, if Alice and Bob's strings have an intersection, then Bob can either sell at most $l - 1$ items at price 1 which with Alice's revenue would result in a total revenue of $2l - 1$, or he can sell all his l items at price $1 - \frac{1}{2l}$ which would give an overall revenue of $2l - \frac{1}{2}$ after adding Alice's revenue. Since the second strategy would result in a higher revenue, the optimal revenue maximizing revenue for Bob's item would be $1 - \frac{1}{2l}$. Thus, the price given by Algorithm A for Bob's item can help distinguish between the two cases in

Disjointness problem using only $o(l)$ available space, which is a contradiction. $\square$

Chapter 5: Prophet Secretary

5.1 Introduction

Optimal stopping theory is a powerful tool for analyzing scenarios in which we generally require optimizing an objective function over the space of stopping rules for an allocation process under uncertainty. One such a scenario is the online auction which is the essence of many modern markets, particularly networked markets where information about goods, agents, and outcomes is revealed over a period of time and the agents must make irrevocable decisions without knowing future information. Combining optimal stopping theory with game theory allows us to model the actions of rational agents applying competing stopping rules in an online market.

Perhaps the most classic problems of stopping theory are the *prophet inequality* and the *secretary problem*. Research investigating the relation between online auction mechanisms and prophet inequalities was initiated by Hajiaghayi, Kleinberg, and Sandholm [103]. They observed that algorithms used in the derivation of prophet inequalities, owing to their monotonicity properties, could be interpreted as truthful online auction mechanisms and that the prophet inequality in turn could be interpreted as the mechanism's approximation guarantee. The connection between the secretary problem and online auction mechanisms has been explored by

Hajiaghayi, Kleinberg and Parkes [104] and initiated several follow-up papers (see e.g. [105–109]).

Prophet Inequality. The classical prophet inequality has been studied in the optimal stopping theory since the 1970s when introduced by Krengel and Sucheston [110–112] and more recently in computer science Hajiaghayi, Kleinberg and Sandholm [103]. In the prophet inequality setting, given (not necessarily identical) distributions $\{D_1, \ldots, D_n\}$, an online sequence of values $X_1, \cdots, X_n$ where X_i is drawn from D_i, an onlooker has to choose one item from the succession of the values, where X_k is revealed at step k. The onlooker can choose a value only at the time of arrival. The onlooker's goal is to maximize her revenue. The inequality has been interpreted as meaning that a prophet with complete foresight has only a bounded advantage over an onlooker who observes the random variables one by one, and this explains the name *prophet inequality*.

An algorithm for the prophet inequality problem can be described by setting a threshold for every step: we stop at the first step that the arriving value is higher than the threshold of that step. The classical prophet inequality states that by choosing the same threshold $OPT/2$ for every step, one achieves the competitive ratio of $1/2$. Here the optimal solution OPT is defined as $\mathrm{E}[\max X_i]$. Naturally, the first question is whether one can beat $1/2$. Unfortunately, this is not possible: let $q = \frac{1}{\epsilon}$, and $q' = 0$. The first value X_1 is always 1. The second value X_2 is either q with probability ϵ or q' with probability $1-\epsilon$. Observe that the expected revenue of any (randomized) online algorithm is $\max(1, \epsilon(\frac{1}{\epsilon})) = 1$. However the prophet, i.e., the optimal offline solution would choose q' if it arrives, and he would choose the first value otherwise. Hence, the optimal offline revenue is $(1-\epsilon) \times 1 + \epsilon(\frac{1}{\epsilon}) \approx 2$. Therefore we cannot hope to break the $1/2$ barrier using any online algorithm.

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 him, 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 by Dynkin in [113] 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 $1/e \approx 0.36$, 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., [114–117].

5.1.1 Our Contributions and Results

In this work, we introduce a natural combination of the prophet inequality problem and the secretary problem that we call *prophet secretary*. Consider a seller that has an item to sell on the market to a set of arriving customers. The seller knows the types of customers that may be interested in the item and he has a price distribution for each type: the price offered by a customer of a type is anticipated to be drawn from the corresponding distribution. However, the customers arrive in a random order. Upon the arrival of a customer, the seller makes an irrevocable decision to whether sell the item at the offered price. We address the question of maximizing the seller's gain.

More formally, in the prophet secretary problem we are given a set $\{D_1, \dots, D_n\}$ of (not necessarily identical) distributions. A number X_i is drawn from each distribution D_i and then, after applying a random permutation $\pi_1, \dots, \pi_n$, the numbers are given to us in an online fashion, i.e., at step k, π_k and X_{π_k} are revealed. We are allowed to choose only one number, which can be done only upon receiving that number. The goal is to maximize the expectation of the chosen value, compared to the expectation of the optimum offline solution that knows the drawn values in advance (i.e., $OPT = \mathrm{E}\left[\max_i X_i\right]$). For the ease of notation, in what follows the index i iterates over the distributions while the index k iterates over the arrival steps.

An algorithm for the prophet secretary problem can be described by a sequence of (possibly adaptive) thresholds $\langle \tau_1, \dots, \tau_n \rangle$: we stop at the first step k that $X_{\pi_k} \geq \tau_k$. In particular, if the thresholds are non-adaptive, meaning that they are decided in advance, the following is a generic description of an algorithm. The competitive ratio of the following algorithm is defined as $\frac{\mathrm{E}[Y]}{OPT}$.

Algorithm Prophet Secretary

Input: A set of distributions $\{D_1, \ldots, D_n\}$; a randomly permuted stream of numbers $(X_{\pi_1}, \ldots, X_{\pi_n})$ drawn from the corresponding distributions.

Output: A number Y.

```
1. Let $\langle \tau_1, \ldots, \tau_n \rangle$ be a sequence of thresholds.
2. For $k \leftarrow 1$ to $n$
    (a) If $X_{\pi_k} \geq \tau_k$ then let $Y = X_{\pi_k}$ and exit the For loop.
3. Output $Y$ as the solution.
```

Recall that when the arrival order is adversarial, the classical prophet inequality states that by choosing the same threshold $OPT/2$ for every step, one achieves the tight competitive ratio of 1/2. On the other hand, for the basic secretary problem where the distributions are not known, the optimal strategy is to let $\tau_1 = \cdots = \tau_{\frac{n}{e}} = \infty$ and $\tau_{\frac{n}{e}+1} = \cdots = \tau_n = \max(X_{\pi_1}, \ldots, X_{\pi_{\frac{n}{e}}})$. This leads to the optimal competitive ratio of $\frac{1}{e} \simeq 0.36$. Hence, our goal in the prophet secretary problem is to beat the 1/2 barrier.

We first show that unlike the prophet inequality setting, one cannot obtain the optimal competitive ratio by using a single uniform threshold. Indeed, Theorem 5.4 in Section5.2 shows that 1/2 is the best competitive ratio one can achieve with uniform thresholds. To beat the $\frac{1}{2}$ barrier, as a warm up we first show that by using two thresholds one can achieve the competitive ratio of $5/9 \simeq 0.55$. This can be achieved by choosing the threshold $\frac{5}{9} \cdot OPT$ for the first half of the steps and then decreasing the threshold to $\frac{OPT}{3}$ for the second half of the steps. Later in Section 5.4, we show that by setting n distinct thresholds one can obtain almost-tight $(1 - 1/e \approx 0.63)$-competitive ratio for the prophet secretary problem.

Theorem 5.1. *Let $\langle \tau_1, \ldots, \tau_n \rangle$ be a non-increasing sequence of n thresholds, such that (i) $\tau_k = \alpha_k \cdot OPT$ for every $k \in [n]$; (ii) $\alpha_n = \frac{1}{n+1}$; and (iii) $\alpha_k = \frac{n\alpha_{k+1}+1}{n+1}$ for $k \in [n-1]$. The competitive ratio of Algorithm* Prophet Secretary *invoked by thresholds τ_k's is at least α_1. When n goes to infinity, α_1 quickly converges to* $1 - 1/e \approx 0.63$.

The crux of the analysis of our algorithm is to compute the probability of picking a value x at a step of the algorithm with respect to the threshold factors α_k's. Indeed one source of difficulty arises from the fundamental dependency between the steps: for any step k, the fact that the algorithm has not stopped in the previous steps leads to various restrictions on what we expect to see at the step k. For example, consider the scenario that D_1 is 1 with probability one and D_2 is either 2 or 0 with equal probabilities. Now if the algorithm chooses $\tau_1 = 1$, then it would never happen that the algorithm reaches step two and receives a number drawn from D_2! That would mean we have received a value from D_1 at the first step which is a contradiction since we would have picked that number. In fact, the optimal strategy for this example is to shoot for D_2! We set $\tau_1 = 2$ so that we can ignore the first value in the event that it is drawn from D_1. Then we set $\tau_2 = 1$ so that we can always pick the second value. Therefore in expectation we get $5/4$ which is slightly less than $OPT = 6/4$.

To handle the dependencies between the steps, we first distinguish between the events for $k \in [n]$ that we pick a value between τ_{k+1} and τ_k. We show that the expected value we pick at such events is indeed highly dependent on $\theta(k)$, the probability of passing the first k elements. We then use this observation to analyze competitive ratio with respect to $\theta(k)$'s and the thresholds factors α_k's. We finally show that the competitive ratio is indeed maximized by choosing the threshold factors described in Theorem 5.1. In Section 5.3, we first prove the theorem for the simple case of $n = 2$. This enables us to demonstrate our techniques without going into the more complicated dependencies for general n. We then present the full proof of Theorem 5.1 in Section 5.4.

On the other hand, from the negative side we prove in Section 5.5 that no online algorithm for the prophet secretary problem can achieve a competitive ratio 0.75.

Theorem 5.2. *For any arbitrary small positive number ϵ, there is no online algorithm for the prophet secretary problem with competitive ratio $0.75 + \epsilon$.*

We also consider the minimization variants of the prophet inequality problem. In particular, in Section 5.5.2 we show that, even for the simple case in which numbers are drawn from *identical and independent distributions (i.i.d.)*, there is no constant competitive online algorithm for the minimization variants of the prophet inequality and prophet secretary problems.

Theorem 5.3. *The competitive ratio of any online algorithm for the minimization prophet inequality with n identical and independent distribution is bounded by $\frac{(1.11)^n}{6}$.*

5.1.2 Related Work

Prophet Inequality. The first generalization of the basic prophet inequality introduced by Krengel and Sucheston [110–112] is *the multiple-choices* prophet inequality [118] in which both the onlooker and the prophet have $k > 1$ choices. Currently, the best algorithm for this setting is due to Alaei [119], who gave an online algorithm with $(1 - \frac{1}{\sqrt{k+3}})$-competitive ratio for k-choice optimal stopping. Besides this, we have two generalizations for the (multiple-choices) prophet inequality that are *matroid* prophet inequality [120] and *matching* prophet inequality [121].

In the matroid prophet inequality, we are given a matroid whose elements have random weights sampled independently from (not necessarily identical) probability distributions on $\mathcal{R}^+$. We then run an online algorithm with knowledge of the matroid structure and of the distribution of each element's weight. The online algorithm must then choose irrevocably an independent subset of the matroid by observing the sampled value of each element (in a fixed, prespecified order). The online algorithm's payoff is defined to be the sum of the weights of the selected elements. Kleinberg and Weinberg [120] show that for every matroid, there is an online algorithm whose expected payoff is at least half of the expected weight of the maximum-weight basis. Observe that the original prophet inequality introduced by Krengel and Sucheston [110–112] corresponds to the special case of rank-one matroids.

The matching prophet inequality is due to Alaei, Hajiaghayi, and Liaghat [121]. Indeed, they study the problem of online prophet-inequality matching in bipartite graphs. There is a static set of bidders and an online stream of items. The interest of bidders in items is represented by a weighted bipartite graph. Each bidder has a capacity, i.e., an upper bound on the number of items that can be allocated to him. The weight of a matching is the total weight of edges matched to the bidders. Upon the arrival of an item, the online algorithm should either allocate it to a bidder or discard it. The objective is to maximize the weight of the resulting matching. Here we assume we know the distribution of the incoming items in advance and we may assume that the t^{th} item is drawn from distribution $\mathcal{D}_t$. They generalize the $\frac{1}{2}$-competitive ratio of Krengel and Sucheston [111] by presenting an algorithm with an approximation ratio of $1 - \frac{1}{\sqrt{k+3}}$ where k is the minimum capacity. Oberve that the classical prophet inequality is a special case of this model where we have only one bidder with capacity one, i.e., $k = 1$ for which they get the same $\frac{1}{2}$-competitive ratio.

Secretary Problem. The first generalization of the basic secretary problem [113] is the *multiple-choice secretary problem* [106] (see a survey by Babaioff et al. [106])) 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 non-negative profit function $f : 2^S \mapsto \mathcal{R}^{\geq 0}$. For example, when $f(T)$ is the maximum individual value [122, 123], or when $f(T)$ is the sum of the individual values in T [109], the problem has been considered thoroughly in the literature. Beside this, two generalizations for the (multiple-choices) secretary problem are *submodular secretary* [124] and *matroid secretary* [107].

The submodular secretary problem is introduced by Bateni, Hajiaghayi, and Zadimoghaddam [124]. Indeed, both of the maximum individual value [122, 123] and the sum of the individual values [109] aforementioned are special monotone non-negative submodular functions. Bateni, Hajiaghayi, and Zadimoghaddam [124] give an online algorithm with $(\frac{7}{1-1/e})$-competitive ratio for the submodular secretary problem. We should mention that there are more recent results with better constant competitive ratio (See for example the references in [124]).

In the matroid secretary problem considered by Babaioff et al. [107], we are given a matroid with 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. 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 a 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. [107] present an $O(\log r)$-competitive algorithm for general matroids, where r is the rank of the matroid. However, they leave as a main open question the existence of constant-competitive algorithms for general matroids.

On the other hand, there are several follow-up works including the recent FOCS 2014 paper due to Lachish [125] which gives $O(\log\log \text{rank})$-competitive algorithm.

5.1.3 Preliminaries

We first define a few notations. For every $k \in [n]$, let z_k denote the random variable that shows the value we pick at the k^{th} step. Observe that for a fixed sequence of drawn values and a fixed permutation, at most one of z_k's is non-zero since we only pick one number. Let z denote the value chosen by the algorithm. By definition, $z = \sum_{k=1}^{n} z_k$. In fact, since all but one of z_k's are zero, we have the following proposition. We note that since the thresholds are deterministic, the randomness comes from the permutation π and the distributions.

Proposition 5.1. $\mathbb{P}[z \geq x] = \sum_{k\in[n]} \mathbb{P}[z_k \geq x]$.

For every $k \in [n]$, let $\theta(k)$ denote the probability that Algorithm Prophet Secretary does not choose a value from the first k steps. For every $i \in n$ and $k \in [n-1]$, let $q_{-i}(k)$ denote the probability that the following two events concurrently happen:

1. Algorithm Prophet Secretary does not choose a value from the first k elements.
2. None of the first k values are drawn from D_i.

Proposition 5.2. *If the thresholds of Algorithm* Prophet Secretary *are non-increasing, then for every $i \in [n]$ and $k \in [n-1]$, we have $\theta(k+1) \leq q_{-i}(k)$.*

Proof. In what fallows, let $i \in [n]$ be a fixed value. The claim is in fact very intuitive: $q_{-i}(k)$ is the probability of the event that the algorithm passes k values chosen from all distributions but D_i. On the other hand, $\theta(k+1)$ corresponds to the event that the algorithm passes $k+1$ values chosen from all distributions. Intuitively, in the latter we have surely passed k values chosen from all but D_i. Therefore $\theta(k+1)$ cannot be more than $q_{-i}(k)$.

Formalizing the intuition above, however, requires an exact formulation of the probabilities. For a permutation s of size k of $[n]$, let $s(j)$, for $j \in [k]$, denote the number at position j of s. For $k \in [n]$, let $S(k)$ denote the set of permutations of size k of $[n]$. Similarly, let $S_{-i}(k)$ denote the set of permutations of size k of $[n] \backslash \{i\}$. Observe that

$$|S(k)| = \frac{n!}{(n-k)!} \qquad\qquad |S_{-i}(k)| = \frac{(n-1)!}{(n-1-k)!}$$

In particular, we note that $|S(k+1)| = n|S_{-i}(k)|$. We can now write down the exact formula for $q_{-i}(k)$ and $\theta(k+1)$.

$$\theta(k+1) = \frac{1}{|S(k+1)|} \sum_{s \in S(k+1)} \prod_{j \in [k+1]} \mathbb{P}\left[X_{s(j)} < \tau_j\right] \tag{5.1}$$

$$q_{-i}(k) = \frac{1}{|S_{-i}(k)|} \sum_{s \in S_{-i}(k)} \prod_{j \in [k]} \mathbb{P}\left[X_{s(j)} < \tau_j\right] \tag{5.2}$$

We now show that $\theta(k+1)$ can be written down as a convex combination of terms that are less than $q_{-i}(k)$. For every $\ell \in [k]$, let $S(k+1,\ell) = \{s \in S(k+1) | s(\ell) = i\}$.

We have:

$$
\begin{aligned}
&\sum_{s:S(k+1,\ell)} \prod_{j\in[k+1]} \mathbb{P}\left[X_{s(j)} \le \tau_j\right] \\
&\le \sum_{s:S(k+1,\ell)} \prod_{j\in[k+1]\setminus\{\ell\}} \mathbb{P}\left[X_{s(j)} \le \tau_j\right] && \mathbb{P}\left[X_i < \tau_\ell\right] \le 1 \\
&\le \sum_{s:S(k+1,\ell)} \prod_{j\in[k]\setminus\{\ell\}} \mathbb{P}\left[X_{s(j)} \le \tau_j\right] \times \mathbb{P}\left[X_{s(k+1)} \le \tau_\ell\right] && \textit{since } \tau_{k+1} \le \tau_\ell \\
&= \sum_{s:S_{-i}(k)} \prod_{j\in[k]} \mathbb{P}\left[X_{s(j)} \le \tau_j\right] \\
&= |S_{-i}(k)| q_{-i}(k) && \textit{By Eq. 5.2}
\end{aligned}
\tag{5.3}
$$

Eq. 5.3 establishes the relation to $q_{-i}(k)$ for members of $S(k+1)$ that contain i in one of the first k positions.

Let $\overline{S(k+1)} = S(k+1)\setminus\bigcup_{\ell\in[k]} S(k+1,\ell)$.

$$
\begin{aligned}
&\sum_{s:\overline{S(k+1)}} \prod_{j\in[k+1]\setminus\{\ell\}} \mathbb{P}\left[X_{s(j)} \le \tau_j\right] \\
&= \sum_{s:\overline{S(k+1)}} \prod_{j\in[k]\setminus\{\ell\}} \mathbb{P}\left[X_{s(j)} \le \tau_j\right] \times \mathbb{P}\left[X_{s(k+1)} \le \tau_{k+1}\right] \\
&\le (n-k) \sum_{s:S_{-i}(k+1)} \prod_{j\in[k]\setminus\{\ell\}} \mathbb{P}\left[X_{s(j)} \le \tau_j\right] && \mathbb{P}\left[X_{s(k+1)} \le \tau_{k+1}\right] \le 1 \\
&= (n-k)|S_{-i}(k)| q_{-i}(k)
\end{aligned}
\tag{5.4}
$$

Finally by Eq. 5.1 we have

$$\theta(k+1) =$$
$$\frac{1}{|S(k+1)|}\sum_{s\in S(k+1)}\prod_{j\in[k+1]}\mathbb{P}\left[X_{s(j)}<\tau_j\right] =$$
$$\frac{1}{|S(k+1)|}\left(\sum_{\ell\in[k]}\sum_{s\in S(k+1,\ell)}\prod_{j\in[k+1]}\mathbb{P}\left[X_{s(j)}<\tau_j\right]+\sum_{s\in\overline{S(k+1)}}\prod_{j\in[k+1]}\mathbb{P}\left[X_{s(j)}<\tau_j\right]\right)\le$$
$$\frac{1}{|S(k+1)|}\left(k|S_{-i}(k)|q_{-i}(k)+(n-k)|S_{-i}(k)|q_{-i}(k)\right) =$$
$$q_{-i}(k) \qquad\qquad |S(k+1)| = n|S_{-i}(k)|$$

where the inequality is by Equations 5.3 and 5.4. □

5.2 One Threshold Can not Break $\frac{1}{2}$ Barrier

To illustrate that considering at least 2 thresholds is necessary to beat $\frac{1}{2}$ barrier for the prophet secretary problem, we first give an example that shows achieving better than $\frac{1}{2}$-competitive ratio for any online algorithm that uses only one threshold for the prophet secretary problem is not possible.

Theorem 5.4. *There is no online algorithm for the prophet secretary problem that uses one threshold and can achieve competitive ratio better than* $0.5+\frac{1}{2n}$.

Proof. Suppose we have $n+1$ distributions where the first n distributions always gives $\frac{1}{1-1/n}$ and the $(n+1)^{th}$ distribution gives n with probability $\frac{1}{n}$ and gives 0 with probability $1-\frac{1}{n}$. Therefore, with probability $\frac{1}{n}$, the maximum is n, and with probability $1-\frac{1}{n}$, the maximum is $\frac{1}{1-1/n}$. Thus, the expected outcome of the offline optimum algorithm is $\frac{1}{n}\times n+(1-\frac{1}{n})\times\frac{1}{1-1/n}=2$.

Now, suppose we have an online algorithm that uses one threshold, say T for a number T, that is the online algorithm accepts the first number greater or equal to a threshold T. We consider two cases for T. The first case is if $T > \frac{1}{1-1/n}$ for which the algorithm does not accept $\frac{1}{1-1/n}$ and thus, the expected outcome of such an algorithm is $\frac{1}{n} \times n = 1$.

The second case is if $T \leq \frac{1}{1-1/n}$. Observe that, with probability $\frac{n}{n+1}$, the first number is $\frac{1}{1-1/n}$ and the online algorithm accepts it. And, with probability $\frac{1}{n+1}$, the distribution that gives n with probability $\frac{1}{n}$ will be the first and the outcome of the algorithm is 1. Thus, the expected outcome of the online algorithm is

$$\frac{n}{n+1} \times \frac{1}{1-1/n} + \frac{1}{n+1} \times 1 = \frac{n^2}{n^2-1} + \frac{1}{n+1} \leq 1 + \frac{1}{n}.$$

Therefore, the competitive ratio of the online algorithms that uses only one threshold is bounded by $\frac{1+1/n}{2} = 0.5 + \frac{1}{2n}$. $\square$

5.3 Two Thresholds Breaks $\frac{1}{2}$ Barrier

Since using one threshold is hopeless, we now try using two thresholds. More formally, for the first half of steps, we use a certain threshold, and then we use a different threshold for the rest of steps. We note that similar to the one-threshold algorithm, both thresholds should be proportional to OPT. Furthermore, at the beginning we should be optimistic and try to have a higher threshold, but if we cannot pick a value in the first half, we may need to lower the bar! We show that by using two thresholds one can indeed achieve the competitive ratio of $\frac{5}{9} \simeq 0.55$. In fact, this improvement beyond $1/2$ happens even at $n = 2$. Thus as a warm up before analyzing the main algorithm with n thresholds, we focus on the case of $n = 2$.

Let $\tau_1 = \alpha_1 OPT$ and $\tau_2 = \alpha_2 OPT$ for some $1 \geq \alpha_1 \geq \alpha_2 \geq 0$ to be optimized later. Recall that z_1 and z_2 are the random variables showing the values picked up by the algorithm at step one and two, respectively. We are interested in comparing $\mathrm{E}[z]$ with OPT. By Proposition 5.1 we have

$$\mathrm{E}[z] = \int_0^\infty \mathbb{P}[z \geq x]\, dx = \int_0^\infty \mathbb{P}[z_1 \geq x]\, dx + \int_0^\infty \mathbb{P}[z_2 \geq x]\, dx$$

Observe that z_1 (resp. z_2) is either zero or has a value more than τ_1 (resp. τ_2). In fact, since $\tau_1 \geq \tau_2$, z is either zero or has a value more than τ_2. Recall that $\theta(1)$ is the probability of $z_1 = 0$ while $\theta(2)$ is the probability of $z_1 = z_2 = 0$. This observation leads to the following simplification:

$$\begin{aligned}
\mathrm{E}[z] &= \int_0^{\tau_2} \mathbb{P}[z_1 \geq x]\, dx + \int_{\tau_2}^{\tau_1} \mathbb{P}[z_1 \geq x]\, dx + \int_{\tau_1}^{\infty} \mathbb{P}[z_1 \geq x]\, dx \\
&\quad + \int_0^{\tau_2} \mathbb{P}[z_2 \geq x]\, dx + \int_{\tau_2}^{\infty} \mathbb{P}[z_2 \geq x]\, dx \\
&= \int_0^{\tau_2} \mathbb{P}[z \geq x]\, dx + \int_{\tau_2}^{\tau_1} \mathbb{P}[z_1 \geq x]\, dx + \int_{\tau_1}^{\infty} \mathbb{P}[z_1 \geq x]\, dx + \int_{\tau_2}^{\infty} \mathbb{P}[z_2 \geq x]\, dx \\
&= \tau_2(1 - \theta(2)) + (\tau_1 - \tau_2)(1 - \theta(1)) + \int_{\tau_1}^{\infty} \mathbb{P}[z_1 \geq x]\, dx + \int_{\tau_2}^{\infty} \mathbb{P}[z_2 \geq x]\, dx
\end{aligned}$$

Let us first focus on $\mathbb{P}[z_1 \geq x]$. The first value may come from any of the two distributions, thus we have

$$\mathbb{P}[z_1 \geq x] = \frac{1}{2}\mathbb{P}[X_1 \geq x] + \frac{1}{2}\mathbb{P}[X_2 \geq x]$$

On the other hand, z_2 is non-zero only if we do not pick anything at the first step. For $i \in \{1, 2\}$, we pick a value of at least x drawn from D_i at step two, if and only if: (i) the value drawn from D_i is at least x; and (ii) our algorithm does not pick a value from the previous step which is drawn from the other distribution. By definitions, the former happens with probability $\mathbb{P}[X_i \geq x]$, while the latter happens with probability $q_{-i}(1)$. Since these two events are independent we have

$$\mathbb{P}[z_2 \geq x] = \frac{1}{2} \sum_{i \in \{1,2\}} q_{-i}(1) \mathbb{P}[X_i \geq x] \geq \frac{\theta(2)}{2} \sum_i \mathbb{P}[X_i \geq x]$$

where the last inequality follows from Proposition 5.2, although the proposition is trivial for $n = 2$. We can now continue analyzing $\mathrm{E}[z]$ from before:

$$\begin{aligned}
\mathrm{E}[z] &= \tau_2(1-\theta(2)) + (\tau_1 - \tau_2)(1-\theta(1)) + \int_{\tau_1}^{\infty} \mathbb{P}[z_1 \geq x]\, dx + \int_{\tau_2}^{\infty} \mathbb{P}[z_2 \geq x]\, dx \\
&\geq \tau_2(1-\theta(2)) + (\tau_1 - \tau_2)(1-\theta(1)) \\
&+ \frac{\theta(1)}{2} \int_{\tau_1}^{\infty} \sum_i \mathbb{P}[X_i \geq x]\, dx + \frac{\theta(2)}{2} \int_{\tau_2}^{\infty} \sum_i \mathbb{P}[X_i \geq x]\, dx
\end{aligned}$$

We note that although the $\theta(1)$ factor is not required in the third term of the last inequality, we include it so that the formulas can have the same formation as in the general formula of the next sections.

It remains to bound $\int_{\tau_k}^{\infty} \sum_i \mathbb{P}[X_i \geq x]$ for $k \in \{1,2\}$. Recall that $OPT = \mathrm{E}[\max_i X_i]$. Hence for every $k \in \{1,2\}$ we have

$$\begin{aligned}
OPT &= \int_0^{\tau_k} \mathbb{P}[\max X_i \geq x]\, dx + \int_{\tau_k}^{\infty} \mathbb{P}[\max X_i \geq x] \\
&\leq \tau_k + \int_{\tau_k}^{\infty} \mathbb{P}[\max X_i \geq x] && \mathbb{P}[\max X_i \geq x] \leq 1 \\
(1-\alpha_k) OPT &\leq \int_{\tau_k}^{\infty} \mathbb{P}[\max X_i \geq x] && \tau_k = \alpha_k OPT \\
&\leq \int_{\tau_k}^{\infty} \sum_i \mathbb{P}[X_i \geq x]\, dx
\end{aligned}$$

where the last inequality is by $\mathbb{P}[\max X_i \geq x] \leq \sum_i \mathbb{P}[X_i \geq x]\, dx$. Therefore we get

$$\begin{aligned}
\mathrm{E}[z] &\geq \tau_2(1-\theta(2)) + (\tau_1-\tau_2)(1-\theta(1)) \\
&+ \frac{\theta(1)}{2}\int_{\tau_1}^{\infty}\sum_i \mathbb{P}[X_i \geq x]\, dx + \frac{\theta(2)}{2}\int_{\tau_2}^{\infty}\sum_i \mathbb{P}[X_i \geq x]\, dx \\
&\geq (\alpha_2 OPT)(1-\theta(2)) + (\alpha_1-\alpha_2)OPT(1-\theta(1)) \\
&+ \frac{\theta(1)}{2}(1-\alpha_1)OPT + \frac{\theta(2)}{2}(1-\alpha_2)OPT \\
&= OPT\left(\alpha_1 + \theta_1(\frac{1+2\alpha_2-3\alpha_1}{2}) + \theta_2(\frac{1-3\alpha_2}{2})\right)
\end{aligned}$$

Therefore by choosing $\alpha_2 = 1/3$ and $\alpha_1 = 5/9$, the coefficients of θ_1 and θ_2 become zero, leading to the competitive ratio of $5/9 \simeq 0.55$. In the next section, we show how one can generalize the arguments to the case of n thresholds for arbitrary n.

5.4 $(1-\frac{1}{e})$-Competitive Ratio Using n Thresholds

In this section we prove our main theorem. In particular, we invoke Algorithm Prophet Secretary with n distinct thresholds $\tau_1,\ldots,\tau_n$. The thresholds $\tau_1,\ldots,\tau_n$ that we consider are *non-adaptive* (i.e., Algorithm Prophet Secretary is oblivious to the history) and *non-increasing*. Intuitively, this is because as we move to the end of stream we should be more pessimistic and use lower thresholds to catch remaining higher values.

Formally, for every $k \in [n]$, we consider threshold $\tau_k = \alpha_k \cdot OPT$ where the sequence $\alpha_1,\ldots,\alpha_n$ is non-increasing that is, $\alpha_1 \geq \alpha_1 \geq \ldots \geq \alpha_n$. We invoke

Algorithm Prophet Secretary with these thresholds and analyze the competitive ratio of Algorithm Prophet Secretary with respect to coefficients α_k. Theorem 5.1 shows that there exists a sequence of coefficients α_k that leads to the competitive ratio of $(1 - 1/e) \approx 0.63$.

Proof of Theorem 5.1. We prove the theorem in two steps: First, we find a lower bound on $\mathrm{E}[z]$ in terms of OPT and coefficients α_i. Second, we set coefficients α_k so that (1) α_1 becomes the competitive ratio of Algorithm Prophet Secretary and (2) α_1 converges to $1 - 1/e$, when n goes to infinity.

Next we give the proof of the theorem in details. Here we first prove few auxiliary lemmas.

In the first lemma, we find a lower bound for $\int_{\tau_k}^{\infty} \mathbb{P}[\max X_i \geq x]\, dx$ based on $OPT = \mathrm{E}[\max_i X_i]$.

Lemma 5.1. *$\int_{\tau_k}^{\infty} \mathbb{P}[\max X_i \geq x]\, dx \geq (1 - \alpha_k) OPT$.*

Proof. For an arbitrary $k \in [n]$ and since $\mathbb{P}[\max X_i \geq x] \leq 1$, we have

$$\begin{aligned} OPT = \mathrm{E}\left[\max_i X_i\right] &= \int_0^{\infty} \mathbb{P}[\max X_i \geq x]\, dx \\ &= \int_0^{\tau_k} \mathbb{P}[\max X_i \geq x]\, dx + \int_{\tau_k}^{\infty} \mathbb{P}[\max X_i \geq x]\, dx \\ &\leq \tau_k + \int_{\tau_k}^{\infty} \mathbb{P}[\max X_i \geq x]\, dx. \end{aligned}$$

Since, by definition $\tau_k = \alpha_k \cdot OPT$, we have $(1 - \alpha_k) \cdot OPT \leq \int_{\tau_k}^{\infty} \mathbb{P}[\max X_i \geq x]\, dx$. □

In the next lemma we split $\mathrm{E}[z]$ into two terms. Later, we find lower bounds for each one of these terms based on $OPT = \mathrm{E}[\max_i X_i]$ separately.

Lemma 5.2. *Let $z = \sum_{k=1}^{n} z_k$ denote the value chosen by Algorithm* Prophet Secretary. *For z we have*

$$\mathrm{E}[z] = \sum_{k=1}^{n} \int_{0}^{\tau_k} \mathbb{P}[z_k \geq x]\, dx + \sum_{k=1}^{n} \int_{\tau_k}^{\infty} \mathbb{P}[z_k \geq x]\, dx.$$

Proof. By Proposition 5.1 we have

$$\begin{aligned}\mathrm{E}[z] &= \int_{0}^{\infty} \mathbb{P}[z \geq x]\, dx = \sum_{k=1}^{n} \int_{0}^{\infty} \mathbb{P}[z_k \geq x]\, dx \\ &= \sum_{k=1}^{n} \int_{0}^{\tau_k} \mathbb{P}[z_k \geq x]\, dx + \sum_{k=1}^{n} \int_{\tau_k}^{\infty} \mathbb{P}[z_k \geq x]\, dx,\end{aligned}$$

where we use this fact that $z = \sum_{k=1}^{n} z_k$ because we only pick one number for a fixed sequence of drawn values and a fixed permutation and therefore, at most one of z_k's is non-zero. □

Now, we find a lower bound for the first term of $\mathrm{E}[z]$ in Lemma 5.2.

Lemma 5.3. $\sum_{k=1}^{n} \int_{0}^{\tau_k} \mathbb{P}[z_k \geq x]\, dx \geq OPT \sum_{k=1}^{n} (1 - \theta(k))(\alpha_k - \alpha_{k+1})$.

Proof. Suppose $x \leq \tau_k$. Observe that event $z_k \geq x$ occurs when Algorithm Prophet Secretary chooses a value at step k. In fact, since the thresholds are non-increasing, whatever we pick at the first k steps would be at least x. Recall that for every $k \in [n]$, $\theta(k)$ is the probability that Algorithm Prophet Secretary does not choose a value from the first k steps. Hence, for every $k \in [n]$ and $x \leq \tau_k$ we have

$$\sum_{j \leq k} \mathbb{P}[z_j \geq x] = 1 - \theta(k). \tag{5.5}$$

To simplify the notation, we assume that $\alpha_0 = \infty$ which means $\tau_0 = \infty$ and we let $\alpha_{n+1} = 0$ which means $\tau_{n+1} = 0$. Therefore we have

$$\sum_{k=1}^{n} \int_{0}^{\tau_k} \mathbb{P}[z_k \geq x]\, dx = \sum_{k=1}^{n} \int_{\tau_{n+1}}^{\tau_k} \mathbb{P}[z_k \geq x]\, dx.$$

Next, we use Equation (5.5) to prove the lemma as follows.

$$\begin{aligned}
\sum_{k=1}^{n}\int_{0}^{\tau_k}\mathbb{P}\left[z_k \geq x\right]dx &= \sum_{k=1}^{n}\int_{\tau_{n+1}}^{\tau_k}\mathbb{P}\left[z_k \geq x\right]dx = \sum_{k=1}^{n}\sum_{r=k}^{n}\int_{\tau_{r+1}}^{\tau_r}\mathbb{P}\left[z_k \geq x\right]dx \\
&= \sum_{r=1}^{n}\int_{\tau_{r+1}}^{\tau_r}\sum_{k=1}^{r}\mathbb{P}\left[z_k \geq x\right]dx \geq \sum_{r=1}^{n}\int_{\tau_{r+1}}^{\tau_r}(1-\theta(r))dx \\
&= \sum_{r=1}^{n}(1-\theta(r))(\tau_r - \tau_{r+1}) = OPT \cdot \sum_{k=1}^{n}(1-\theta(k))(\alpha_k - \alpha_{k+1})
\end{aligned}$$

□

Then, we find a lower bound for the second term of $\mathrm{E}\left[z\right]$ in Lemma 5.2.

Lemma 5.4. $\sum_{k=1}^{n}\int_{\tau_k}^{\infty}\mathbb{P}\left[z_k \geq x\right]dx \geq OPT\sum_{k}\frac{\theta(k)}{n}(1-\alpha_k)$.

Proof. Recall that for every distribution D_i we draw a number X_i. Later, we randomly permute numbers $X_1, \cdots, X_n$. Let the sequence of indices after random permutation be $\pi_1, \ldots, \pi_n$ that is, at step k, number X_{π_k} for $\pi_k \in [n]$ is revealed.

Suppose $x \geq \tau_k$. We break the event $z_k > 0$ to n different scenarios depending on which index of the distributions $D_1, \cdots, D_n$ is mapped to index π_k in the random permutation. Let us consider the scenario in which Algorithm Prophet Secretary chooses the value drawn from a distribution i at step k. Such a scenario happens if (i) Algorithm Prophet Secretary does not choose a value from the first $k-1$ steps which are not drawn from i, and (ii) $X_i \geq \tau_k$. Observe that the two events are independent. Therefore, we have $\mathbb{P}\left[z_k \geq x\right] = \sum_i \mathbb{P}\left[\pi_k = i\right] \cdot \mathbb{P}\left[X_i \geq x\right] \cdot q_{-i}(k-1)$, where $q_{-i}(k)$ for every $i \in n$ and $k \in [n-1]$ is the probability that the following two events concurrently happen: (i) Algorithm Prophet Secretary does not choose a value from the first k elements, and (ii) none of the first k values are drawn from D_i. Since π_k is an index in the random permutation we obtain

$$\mathbb{P}\left[z_k \geq x\right] = \sum_i \mathbb{P}\left[\pi_k = i\right] \cdot \mathbb{P}\left[X_i \geq x\right] \cdot q_{-i}(k-1) = \frac{1}{n} \cdot \sum_i \mathbb{P}\left[X_i \geq x\right] \cdot q_{-i}(k-1).$$

Using Proposition 5.2 and an application of the union bound we then have

$$\begin{aligned}\mathbb{P}\left[z_k \geq x\right] &= \sum_i \mathbb{P}\left[\pi_k = i\right] \cdot \mathbb{P}\left[X_i \geq x\right] \cdot q_{-i}(k-1) = \frac{1}{n} \sum_i \mathbb{P}\left[X_i \geq x\right] \cdot q_{-i}(k-1)\\ &\geq \frac{\theta(k)}{n} \cdot \sum_i \mathbb{P}\left[X_i \geq x\right] \geq \frac{\theta(k)}{n} \cdot \mathbb{P}\left[\max_i X_i \geq x\right]\end{aligned}$$

Therefore, we obtain the following lower bound on $\sum_{k=1}^n \int_{\tau_k}^\infty \mathbb{P}\left[z_k \geq x\right] dx$.

$$\begin{aligned}\sum_{k=1}^n \int_{\tau_k}^\infty \mathbb{P}\left[z_k \geq x\right] dx &\geq \sum_k \int_{\tau_k}^\infty \frac{\theta(k)}{n} \mathbb{P}\left[\max X_i \geq x\right] dx\\ &= \sum_k \frac{\theta(k)}{n} \int_{\tau_k}^\infty \mathbb{P}\left[\max X_i \geq x\right] dx\end{aligned}$$

Finally, we use the lower bound of Lemma 5.1 for $\int_{\tau_k}^\infty \mathbb{P}\left[\max X_i \geq x\right] dx$ to prove the lemma.

$$\begin{aligned}\sum_{k=1}^n \int_{\tau_k}^\infty \mathbb{P}\left[z_k \geq x\right] dx \geq \sum_k \frac{\theta(k)}{n} \int_{\tau_k}^\infty \mathbb{P}\left[\max X_i \geq x\right] dx &\geq \sum_k \frac{\theta(k)}{n} \cdot (1-\alpha_k) \cdot OPT\\ &= OPT \cdot \sum_k \frac{\theta(k)}{n} \cdot (1-\alpha_k)\end{aligned}$$

□

We use the lower bounds of Lemmas 5.3 and 5.4 in Lemma 5.2 to obtain a lower bound for $\mathrm{E}\left[z\right]$.

Lemma 5.5. *Let $z = \sum_{k=1}^{n} z_k$ denote the value chosen by Algorithm* Prophet Secretary. *For z we have*

$$\mathrm{E}\left[z\right] \geq OPT \cdot (\alpha_1 + \sum_{k=1}^{n} \theta(k)(\frac{1}{n} - \frac{\alpha_k}{n} - \alpha_k + \alpha_{k+1})).$$

Proof. By using the lower bounds of Lemmas 5.3 and 5.4 for the terms of Lemma 5.2 we have

$$\begin{aligned}\mathrm{E}\left[z\right] &\geq OPT \sum_{k=1}^{n}((1-\theta(k))(\alpha_k - \alpha_{k+1}) + \frac{\theta(k)}{n}(1-\alpha_k)) \\ &= OPT(\alpha_1 + \sum_{k=1}^{n} \theta(k)(\frac{1}{n} - \frac{\alpha_k}{n} - \alpha_k + \alpha_{k+1})).\end{aligned}$$

□

We finish the theorem by proving the following claim.

Lemma 5.6. *The competitive ratio of Algorithm* Prophet Secretary *is α_1 which quickly converges to $1 - 1/e \approx 0.63$ when n goes to infinity.*

Proof. Using Lemma 5.5, for z we have $\mathrm{E}\left[z\right] \geq OPT\left(\alpha_1 + \sum_{k=1}^{n} \theta(k)\left(\frac{1}{n} - \frac{\alpha_k}{n} - \alpha_k + \alpha_{k+1}\right)\right)$ which means that the competitive ratio depends on the probabilities $\theta(k)$'s. However, we can easily get rid of the probabilities $\theta(k)$'s by choosing α_k's such that for every k, $\left(\frac{1}{n} - \frac{\alpha_k}{n} - \alpha_k + \alpha_{k+1}\right) = 0$.

More formally, by starting from $\alpha_{n+1} = 0$ and choosing $\alpha_k = \frac{1+n\alpha_{k+1}}{1+n}$ for $k \leq n$, the competitive ratio of the algorithm would be α_1. In below, we show that when n goes to infinity, α_1 quickly goes to $1 - 1/e$ which means that the competitive ratio of Algorithm Prophet Secretary converges to $1 - 1/e \approx 0.63$.

First, by the induction we show that $\alpha_k = \sum_{i=0}^{n+1-k} \frac{n^i}{(1+n)^{i+1}}$. For the base case we have

$$\alpha_n = \frac{1 + n\alpha_{n+1}}{1+n} = \frac{1 + n \times 0}{1+n} = \frac{n^0}{(1+n)^1}.$$

Given $\alpha_{k+1} = \sum_{i=0}^{n+1-(k+1)} \frac{n^i}{(1+n)^{i+1}}$ we show the equality for α_k as follows.

$$\begin{aligned}\alpha_k &= \frac{1 + n\alpha_{k+1}}{1+n} = \frac{1 + n \times \alpha_{k+1}}{1+n} = \frac{1 + n(\sum_{i=0}^{n+1-(k+1)} \frac{n^i}{(1+n)^{i+1}})}{1+n} \\ &= \frac{n^0}{(1+n)^1} + \sum_{i=0}^{n+1-(k+1)} \frac{n^{i+1}}{(1+n)^{i+2}} = \sum_{i=0}^{n+1-k} \frac{n^i}{(1+n)^{i+1}}.\end{aligned}$$

Now we are ready to show $\alpha_0 \geq 1 - 1/e$ when n goes to infinity.

$$\begin{aligned}\lim_{n\to\infty} \alpha_0 &= \lim_{n\to\infty} \sum_{i=0}^{n+1} \frac{n^i}{(n+1)^{i+1}} = \lim_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n+1} (1 - \frac{1}{n+1})^i \\ &\approx \lim_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n+1} e^{-i/n} \approx \int_0^1 e^{-x} dx = 1 - 1/e.\end{aligned}$$

□

5.5 Lower Bounds

In this section we give our lower bounds for the prophet secretary problem and the minimization variants of the prophet inequality and prophet secretary problems. First, in Section 5.5.1 we show that no online algorithm for the prophet secretary problem can achieve a competitive ratio of 0.75. Later, in Section 5.5.2 we consider the minimization variant of the prophet inequality problem. We show that, even for the simple case in which numbers are drawn from *identical and independent distributions (i.i.d.)*, there is no constant competitive online algorithm for the minimization variants of the prophet inequality and prophet secretary problems.

5.5.1 0.75-Lower Bound of Prophet Secretary Problem

Here we give a simple example for which no online algorithm for the prophet secretary problem can achieve a competitive ratio 0.75.

Lemma 5.7. *There is no online algorithm for the prophet secretary problem with competitive ratio* $0.75 + \epsilon$*, where* ϵ *is an arbitrary small positive number less than* 1.

Proof. It is known that no algorithm can guarantee a competitive ratio better than $0.5 + \epsilon$, where ϵ is an arbitrary small positive number less than 1. A hard example that shows this upper bound is as follow. We have two distributions. The first distribution always gives 1, and the second distribution gives $\frac{1}{\epsilon}$ with probability ϵ and 0 with probability $1 - \epsilon$. Observe that if we either accept the first number or reject it our expected outcome is 1. On the other hand, the offline optimum algorithm takes $\frac{1}{\epsilon}$ with probability ϵ and 1 with probability $1 - \epsilon$. Therefore, the expected outcome of the offline optimum algorithm is $\epsilon \cdot \frac{1}{\epsilon} + 1 \cdot (1 - \epsilon) = 2 - \epsilon$, and the competitive ratio is at most $\frac{1}{2-\epsilon} \leq 0.5 + \epsilon$.

The above example contains exactly two distributions. Thus, if the drawn numbers from these distributions arrive in a random order, with probability 0.5 the arrival order is the worst case order. This means that in the prophet secretary, with probability 0.5 the expected outcome of any algorithm on this example is at most 1, while the offline optimum algorithm is always $2 - \epsilon$. Therefore, there is no algorithm for the prophet secretary problem with competitive ratio better than $\frac{0.5 \times 1 + 0.5 \times (2-\epsilon)}{2-\epsilon} \leq 0.75 + \epsilon$. $\square$

5.5.2 Minimization Variants of Classical Problems

In this section, we consider the minimization variant of the prophet inequality setting. First, we consider the simple case that numbers are drawn from identical and independent distributions. In particular, we prove Theorem 5.3 that shows, even for the simple case of identical and independent distributions, there is no constant competitive online algorithm for the minimization variant of the prophet inequality problem. Later, we give more power to the online algorithm and let it to change its decision once; we call this model *online algorithm with one exchange*. We show in Lemma 5.5 that there is no constant competitive online algorithm with one exchange for the minimization variants of the prophet inequality and prophet secretary problems.

Proof of Theorem 5.3. Suppose we have n identical distributions, each gives 0 with probability $\frac{1}{3}$, 1 with probability $\frac{1}{3}$ and 2^n with probability $\frac{1}{3}$. One can see that with probability $(\frac{1}{3})^n$, all of the numbers are 2^n and thus the minimum number is 2^n. Also, with probability $(\frac{2}{3})^n - (\frac{1}{3})^n$, there is no 0 and there is at least one 1, and thus, the minimum is 1. In all the other cases, the minimum is 0. Therefore, the expected outcome of the offline optimum algorithm is $(\frac{1}{3})^n \times 2^n + (\frac{2}{3})^n - (\frac{1}{3})^n < \frac{2^{n+1}}{3^n}$.

For this example, without loss of generality we can assume that any online algorithm accept 0 as soon as it appears, and also it does not accept 2^n except for the last item. Assume $i+1$ is the first time that the algorithm is willing to accept 1. The probability that we arrive in this point is $(\frac{2}{3})^i$ and the probability that we see 1 at that point is $\frac{1}{3}$. On the other hand, the probability that such an algorithm does not accept anything up to the last number and sees that the last number is 2^n is at least $(\frac{2}{3})^i(\frac{1}{3})^{n-i}$. Therefore, the expected outcome of the online algorithm is at least $(\frac{2}{3})^i\frac{1}{3} \times 1 + (\frac{2}{3})^i(\frac{1}{3})^{n-i} \times 2^n = \frac{2^i 3^{n-i-1}+3^i}{3^n}$. Thus, the competitive ratio is at least

$$\frac{2^i 3^{n-i-1} + 3^i}{2^{n+1}} = \frac{1}{6}(\frac{3^{n-i}}{2^{n-i}} + \frac{3^{i+1}}{2^n}).$$

If $i \leq 0.73n$, the left term is at least $\frac{(1.11)^n}{6}$; otherwise, the right term is at least $\frac{(1.11)^n}{6}$.

Theorem 5.5. *For any large positive number C, there is no C-competitive algorithm for minimization prophet inequality with one exchange.*

Proof. Suppose we have three distributions as follows. The first distribution always gives 1. The second distribution gives $\frac{1}{\epsilon}$ with probability ϵ and gives $\frac{\epsilon}{1-\epsilon}$ with probability $1-\epsilon$. The third distribution gives $\frac{1}{\epsilon}$ with probability ϵ and gives 0 with probability $1-\epsilon$. We set ϵ later.

We observe that the minimum number is 0 with probability $1-\epsilon$, is $\frac{\epsilon}{1-\epsilon}$ with probability $\epsilon(1-\epsilon)$ and is 1 with probability ϵ^2. Thus, the expected outcome of the optimum algorithm is

$$(1-\epsilon) \times 0 + \epsilon(1-\epsilon) \times \frac{\epsilon}{1-\epsilon} + \epsilon^2 \times 1 = 2\epsilon^2.$$

Now, we show that the outcome of any online algorithm with one exchange for this input is at least ϵ. In fact, this means that, the competitive ratio can not be less than $\frac{\epsilon}{2\epsilon^2} = \frac{1}{2\epsilon}$. If we set $\epsilon = \frac{1}{2C}$, this means that there is no C-competitive algorithm for the minimization prophet inequality with one exchange.

Recall that there is no uncertainty in the first number. If an algorithm withdraw the first number, it can take the outcome of either the second distribution or the third distribution. However, if we do this, with probability ϵ^2 the outcome is $\frac{1}{\epsilon}$. Thus, the expected outcome is at least ϵ as desired.

Now, we just need to show that if an algorithm does not select the first number, its expected outcome is at least ϵ. Observe that if this happens, then the algorithm has only the option of choosing either the second and the third distribution. We consider two cases. The first case is if the algorithm does not select the second number, then it must select the third number. Therefore, with probability ϵ, the outcome of the algorithm is 1 and thus, the expected outcome is at least ϵ. The second case occurs if the algorithm selects the second number and that is $\frac{\epsilon}{1-\epsilon}$. Therefore, with probability $1-\epsilon$, the outcome of the algorithm is $\frac{\epsilon}{1-\epsilon}$ and again the expected outcome is $\frac{\epsilon}{1-\epsilon} \times (1-\epsilon) = \epsilon$. □

Finally, we show that even for the minimization variant of the prophet secretary there is no hope to get a constant competitive ratio.

Corollary 5.1. *For any large number C, there is no C-competitive algorithm for minimization prophet secretary with one exchange.*

Proof. In Theorem 5.5, we have three distributions. Thus, in the prophet secretary model the worst case order happen with probability $\frac{1}{6}$. Thus, the competitive ratio can not be more than $\frac{\frac{1}{6}\cdot\epsilon}{2\epsilon^2} = \frac{1}{12\epsilon}$. If we set $\epsilon = \frac{1}{12C}$, this essentially means that there is no C-competitive algorithm for the minimization prophet secretary with one exchange. □

Corollary 5.2. *For any large number C there is no C-competitive algorithm for the minimization secretary problem with one exchange.*

Proof. Suppose for the sake of contradiction that there exists an algorithm Alg for the minimization secretary problem which is C-competitive. Consider all possible realizations of the example in Theorem 5.5. Algorithm Alg is C-competitive when each of the these realizations comes in a random order. Therefore, Algorithm Alg is C-competitive when the input is a distribution over these realizations. This says that Alg is a C-competitive algorithm for the minimization prophet secretary, which contradicts Corollary 5.1 and completes the proof. □

Chapter 6: IID Prophet Inequalities

6.1 Introduction

Online auctions play a major role in modern markets. In online markets, information about customers and goods is revealed over time. Irrevocable decisions are made at certain discrete times, such as when a customer arrives to the market. One of the fundamental and basic tools to model this scenario is the *prophet inequality* and its variants.

In a prophet inequality instance we are given a sequence of distributions. Iteratively, we draw a value from one of the distributions, based on a predefined order. In each step we face two choices, either we accept the value and stop, or we reject the value and move to the next distribution. The goal in this problem is to maximize the expected value of the item selected. We say an algorithm for a prophet inequality instance is an α-approximation, for $\alpha \leq 1$, if the expectation of the value picked by the algorithm is at least α times that of an optimum solution which knows all of the values in advance.

Prophet inequalities were first studied in the 1970's by Krengel and Sucheston [111, 112, 126]. Hajiaghayi, Kleinberg and Sandholm [103] studied the relation between online auctions and prophet inequalities. In particular they showed that algorithms used in the derivation of prophet inequalities can be reinterpreted as truthful mechanisms for online auctions. Later Chawla, Hartline, Malec, and Sivan [127] used prophet inequalities to design sequential posted price mechanisms whose revenue approximates that of the Bayesian optimal mechanism.

In the classical definition of the prophet inequality, the values can be drawn from their distributions in an arbitrary (a.k.a. *adversarial* or *worst*) order. Assuming an adversarial order, the problem has a 0.5 approximation algorithm which is tight. Recently, Yan [128] considered a relaxed version of this problem in which the algorithm designer is allowed to pick the order of distributions (a.k.a. *best order*), and provided a $1 - \frac{1}{e}$ approximation algorithm for this problem. Later, Esfandiari, Hajiaghayi, Liaghat and Monemizadeh [129] showed that there exists a $1 - \frac{1}{e}$ approximation algorithm even when the distributions arrive in a *random order*. Both results provided by Yan and Esfandiari et al. are not tight.

In this work we consider prophet inequalities in both best order and random order settings and take steps towards providing tight approximation algorithms for these problems. Particularly, we consider this problem assuming a large market assumption (i.e. we have several copies of each distribution). Indeed, the large market assumption is well-motivated in this context [130–134].

6.1.1 Our Contribution

First we consider the prophet inequality on a set of *identical and independent distributions* (*iid*). The prophet inequality on iid distributions has been previously studied by Hill and Kertz [135] in the 1980's. Hill and Kertz provided an algorithm based on complicated recursive functions. They proved a theoretical bound of $1 - \frac{1}{e}$ on the approximation factor of their algorithm, and used a computer program to show that their algorithm is a 0.745-approximation when the number of distributions is $n = 10000$. They conjectured that the best approximation factor for arbitrarily large n is $\frac{1}{1+1/e} \simeq 0.731$. This conjecture remained open for more than three decades.

In this chapter we present a simple threshold-based algorithm for the prophet inequality with n iid distributions. Using a nontrivial and novel approach we show that our algorithm is a 0.738-approximation algorithm for large enough n, beating the bound of $\frac{1}{1+1/e}$ conjectured by Hill and Kertz. This is the first algorithm which is theoretically proved to have an approximation factor better than $1 - \frac{1}{e}$ for this problem. Indeed, beating the $1-\frac{1}{e}$ barrier is a substantial work in this area [136,137]. The following theorem states our claim formally.

Theorem 6.1. *There exists a constant number n_0 such that for every $n \geq n_0$, there exists a* 0.738*-approximation algorithm for any prophet inequality instance with n iid distributions.*

Next, we extend our results to support different distributions. However, we assume that we have several copies of each distribution. This can be reinterpreted as a large market assumption. We say a multiset of independent distributions $\{F_1, \ldots, F_n\}$ is *m-frequent* if for each distribution F_i in this multiset there are at least m copies of this distribution in the multiset. We show that by allowing the algorithm to pick the order of the distributions, there exists a 0.738-approximation algorithm for any prophet inequality instance on a set of m-frequent distributions, for large enough m. The following theorem states this fact formally.

Theorem 6.2. *There exists a constant number m_0 such that there exits a* 0.738-*approximation best order algorithm for any prophet inequality instance on a set of m_0-frequent distributions.*

Our next theorem shows that even in the random order setting one can achieve a 0.738-approximation algorithm on m-frequent distributions, for large enough m.

Theorem 6.3. *There exists a constant number c_0 such that there exits a* 0.738-*approximation random order algorithm for any prophet inequality instance on a set of $(c_0 \log(n))$-frequent distributions.*

To conclude the presentation of our results we show that it is not possible to extend our results to the worst order setting. The following theorem states this fact formally.

Theorem 6.4. *For any arbitrary m, there is a prophet inequality instance on a set of m-frequent distributions such that the instance does not admit any $0.5 + \epsilon$-approximation worst order algorithm.*

6.1.2 Applications in Mechanism Design

The prophet inequality has numerous applications in mechanism design and optimal search theory, so our improved prophet inequality for m-frequent distributions has applications in those areas as well. By way of illustration, we present here an application to optimal search theory. In Weitzman's [138] "box problem", there are n boxes containing indepedent random prizes, $v_1, \ldots, v_n$, whose distributions are not necessarily identical. The cost of opening box i is $c_i \geq 0$. A decision maker is allowed to open any number of boxes, after which she is allowed to claim the largest prize among the open boxes. The costs of the boxes, and the distributions of the prizes inside, are initially known to the decision maker, but the value v_i itself is only revealed when box i is opened. A search policy is a (potentially adaptive) rule for deciding which box to open next—or whether to stop—given the set of boxes that have already been opened and the values of the prizes inside. Weitzman [138] derived the structure of the optimal search policy, which turns out to be wonderfully simple: one computes an "option value" σ_i for each box i, satisfying the equation $E[\max\{0, v_i - \sigma_i\}] = c_i$. Boxes are opened in order of decreasing σ_i until there is some open box i such that $v_i > \sigma_j$ for every remaining closed box j, then the policy stops. Kleinberg, Waggoner, and Weyl [139] presented an alternative proof of this result which works by relating any instance of the box problem to a modified instance in which opening boxes is cost-free, but the prize in box i is $\min\{v_i, \sigma_i\}$ rather than v_i. The proof shows that when we run any policy on the modified instance, its net value (prize minus combined cost) weakly improves, and that the net value is preserved if the policy is *non-exposed*, meaning that whenever it opens a box with $v_i > \sigma_i$, it always claims the prize inside.

An interesting variant of the box problem arises if one constrains the decision maker, upon stopping, to choose the prize in the most recently opened box, rather than the maximum prize observed thus far. In other words, upon opening box i the decision maker must irrevocably decide whether to end the search and claim prize v_i, or continue the search and relinquish v_i. Let us call this variant the *impatient box problem.* It could be interpreted as modeling, for example, the decision problem that an employer faces when scheduling a sequence of costly job interviews in a labor market where hiring decisions must be made immediately after the interview. The factor $1 - \frac{1}{e}$ prophet inequality of Yan and Esfandiari et al. implies that if the decision maker is allowed to choose the order in which to inspect boxes (or even if a random order is used), the net value of the optimal impatient box problem policy is at least $1-\frac{1}{e}$ times the net value of the optimal policy for the corresponding instance of the original (non-impatient) box problem; for the proof of this implication, see Corollary 3 and Remark 1 in [139]. A consequence of Theorem 6.2 above is that this ratio improves to 0.738 if the instance of the impatient box problem contains sufficiently many copies of each type of box.

Our results also have applications to a recent line of work that employs prophet inequalities to design posted-price mechanisms. In the standard posted-price setup, a seller has a collection of resources to distribute among n buyers. The buyers' values are drawn independently from distributions that are known in advance to the seller. The seller can use this distributional knowledge to set a (possibly adaptive) price on the goods for sale. Buyers then arrive sequentially and make utility-maximizing purchases. Hajiaghayi et al. [103] noted the close connection between this problem and the prophet inequality, with the price corresponding to a choice of threshold. This has immediate implications for designing prices for welfare maximization, and one can additionally obtain bounds for revenue by applying the prophet inequality to virtual welfare [127, 140]. There has subsequently been a significant line of work extending this connection to derive posted-price mechanisms for broader classes of allocation problems, such as matroid constraints [141], multi-item auctions [119, 127] and combinatorial auctions [142]. The result of Yan and Esfandiari et al. [129] implies that for the original case of a single item for sale, if the seller is allowed to choose the order in which the buyers arrive (or if they can be assumed to arrive in random order), then a posted-price mechanism can obtain expected welfare that is at least $1 - \frac{1}{e}$ times the expected welfare of the optimal assignment. Theorem 6.2 implies that this ratio improves to 0.738 if the pool of buyers contains sufficiently many individuals whose values are drawn from the same distribution.

6.1.3 Other Related Work

The first generalization of the prophet inequality is the *multiple-choice prophet inequality* [126,143,144]. In the multiple-choice prophet inequality we are allowed to pick k values, and the goal is to maximize the total sum of picked values. Alaei [119] gives an almost tight $(1 - \frac{1}{\sqrt{k+3}})$-approximation algorithm for the k-choice prophet inequality (the lower bound is proved in Hajiaghayi, Kleinberg, and Sandholm [103]).

Prophet inequalities have been studied under complicated combinatorial structures such as matroid, polymatroid, and matching. Kleinberg and Weinberg [141] consider matroid prophet inequalities, in which the set of selected values should be an independent set of a predefined matroid. They give a tight 0.5-approximation worst order algorithm for this problem. Later, Dütting and Kleinberg extended this result to polymatroids [145].

Alaei, Hajiaghayi, and Liaghat study matching prophet inequalities [121, 146, 147]. They extend the multiple-choice prophet inequality and give an almost tight $(1-\frac{1}{\sqrt{k+3}})$-approximation worst order algorithm for any matching prophet inequality instance, where k is the minimum capacity of a vertex.

Rubinstein considers the prophet inequalities restricted to an arbitrary downward-closed set system [148]. He provides an $O(\log n \log r)$-approximation algorithm for this problem, where n is the number of distributions and r is the size of the largest feasible set. Babaioff, Immorlica and Kleinberg show a lower bound of $\Omega(\frac{\log n}{\log \log n)})$ for this problem [149]. Prophet inequalities has also been studied restricted to independent set in graphs [150].

6.2 IID Distributions

In this section we give a 0.738-approximation algorithm for prophet inequality with iid items. Let us begin with some definitions. Assume that $X_1, \ldots, X_n$ are iid random variables with common distribution function F. For simplicity, assume that F is continuous and strictly increasing on a subinterval of $\mathcal{R}^{\geq 0}$. An algorithm based on a sequence of thresholds $\theta_1, \ldots, \theta_n$ is the one that selects the first item k such that $X_k \geq \theta_k$.

Let τ denote the stopping time of this algorithm, where τ is $n+1$ when the algorithm selects no item. For simplicity suppose X_{n+1} is a zero random variable. The approximation factor of an algorithm based on $\theta_1 \ldots, \theta_n$ is defined as

$E[X_\tau]/E[\max X_i]$. This factor captures the ratio between what a player achieves in expectation by acting based on these thresholds and what a prophet achieves in expectation by knowing all X_i's in advance and taking the maximum of them.

In Algorithm 0 we presents a simple oblivious algorithm for every n and distribution function F. Theorem 6.5 proves that this algorithm is at least 0.738-approximation for large enough number of items.

Algorithm 6

Input: n iid items with distribution function F.

```
1: Set $a$ to 1.306 (root of $\cos(a) - \sin(a)/a - 1$).
2: Set $\theta_i = F^{-1}(\cos(ai/n)/\cos(a(i-1)/n))$.
3: Pick the first item $i$ for which $X_i \geq \theta_i$.
```

Theorem 6.5. *For every $\epsilon > 0$ there exists a number n_ϵ (a function of ϵ and independent of n) such that for every $n \geq n_\epsilon$ Algorithm 0 for n items is at least $(1-\epsilon)\alpha$-approximation where $\alpha = 1 - \cos(a) \approx 0.7388$.*

In the following we walk you through the steps of the design of Algorithm 0 and provide a proof for Theorem 6.5. For a given sequence of thresholds let $q_0, q_1, \ldots, q_n$ denote the probability of the algorithm not choosing any of the first items. More specifically, let $q_i = Pr[\tau > i]$ for every $0 \leq i \leq n$. Knowing the thresholds $\theta_1, \ldots, \theta_n$ one can find this sequence by starting from $q_0 = 1$ and computing the rest using $q_i = q_{i-1}F(\theta_i)$. Inversely, one can simply find the thresholds from $q_1, \ldots, q_n$ using $\theta_i = F^{-1}(q_i/q_{i-1})$. Hence, we focus the design of our algorithm on finding the sequence $q_1, \ldots, q_n$. To this end, we aim to find a continuous function $h : [0,1] \rightarrow$

$[0,1]$ with $h(0) = 1$ such that by setting $q_i = h(i/n)$ we can achieve our desired set of thresholds.

Note that such a function h has to meet certain requirements. For instance, it has to be strictly decreasing, because at every step the algorithm picks an item with some positive probability, therefore $h(i/n) = q_i = Pr[\tau > i]$ is smaller for larger i. In the following we define a class of functions which has two additional properties. We prove that these properties can be useful in designing a useful threshold algorithm.

Definition 6.1. *A continuous and strictly decreasing function $h : [0,1] \to [0,1]$ with $h(0) = 1$ is a threshold function if it has the following two properties:*

i. h is a strictly concave function.

ii. For every $\epsilon > 0$ there exists some $\delta_0 \leq \epsilon$ such that for every $\delta \leq \delta_0$, $\epsilon + \delta \leq s \leq 1$:

$$h'(s-\delta)/h(s-\delta) \leq (1-\epsilon)h'(s)/h(s) \ .$$

As shown in the following lemma, the first property leads to a decreasing sequence of thresholds. Also, we exploit the second property to show that the approximation factor of h improves by increasing the number of items.

Lemma 6.1. *If h is a threshold function, then the sequence of thresholds $\theta_1, \dots, \theta_n$ achieved from h is decreasing.*

Proof. For every $1 \leq i \leq n$ we have $\theta_i = F^{-1}(q_i/q_{i-1})$. Since every $q_i = h(i/n)$, we have $\theta_i = F^{-1}(\frac{h(i/n)}{h((i-1)/n)})$. Note that F is a strictly increasing function, therefore having $\theta_i > \theta_{i+1}$ requires $\frac{h(i/n)}{h((i-1)/n)} > \frac{h((i+1)/n)}{h(i/n)}$. For simplicity let $x = i/n$ and

$\delta = 1/n$. From the first property of threshold functions we have:

$$h(x) > \frac{h(x+\delta)+h(x-\delta)}{2} \ .$$

By raising both sides to the power of 2, and subtracting $(h(x+\delta)/2 - h(x-\delta)/2)^2$ from each side we have:

$$\begin{aligned}&h(x)^2 - \left(\frac{h(x+\delta)-h(x-\delta)}{2}\right)^2 > \\ &\left(\frac{h(x+\delta)+h(x-\delta)}{2}\right)^2 - \left(\frac{h(x+\delta)-h(x-\delta)}{2}\right)^2 = \\ &h(x+\delta)h(x-\delta) \ .\end{aligned}$$

Therefore $h(x)^2 > h(x+\delta)h(x-\delta)$, which means $h(x)/h(x-\delta) > h(x+\delta)/h(x)$ and the proof is complete. □

Next, we define a class of functions and prove for every function of this class that its approximation factor approaches α for a large enough n. This enables us to narrow down our search for a useful function h.

Definition 6.2. *A threshold function h is α-strong if it has the following properties:*

i. $h(1) \leq 1 - \alpha$.

ii. $\int_0^1 h(r)dr \geq \alpha$.

iii. $\forall\ 0 \leq s \leq 1 : 1 - h(s) - h'(s)/h(s) \int_s^1 h(r)dr \geq \alpha(1 - \exp(h'(s)/h(s)))$.

The following theorem formally states the mentioned claim for α-strong functions.

Theorem 6.6. *If h is an α-strong function, then for every $\epsilon > 0$ there exists an n_ϵ such that for every $n \geq n_\epsilon$ the threshold algorithm that acts based on h is at least $(1-\epsilon)\alpha$-approximation on n iid items.*

Proof. Let OPT be a random variable that denotes the optimum solution and ALG be a random variable that denotes the value picked by the algorithm. We can write the expectation of OPT as

$$\mathcal{E}[OPT] = \int_0^\infty Pr[\max X_i \geq x]dx \ . \tag{6.1}$$

Similarly the expectation of ALG is

$$\mathcal{E}[ALG] = \int_0^\infty Pr[X_\tau \geq x]dx \ . \tag{6.2}$$

The main idea behind the proof is to show for α-strong functions that the integrand in (6.2) is an approximation of the integrand in (6.1) for every non-negative value of x. In particular, for every ϵ there exists some n_ϵ such that for every $n \geq n_\epsilon$ the second integrand is at least $(1-\epsilon)\alpha$ times the first integrand and this proves the theorem.

Let us begin with finding an upper bound for the integrand in (6.1). Let $G(x) = 1 - F(x)$ for every $x \in \mathcal{R}^{\geq 0}$. The following lemma gives an upper bound for $Pr[\max X_i \geq x]$ based on $G(x)$ and n.

Lemma 6.2. *For every $\epsilon > 0$ there exists an n_ϵ such that for every $n \geq n_\epsilon$ the following inequality holds :*

$$Pr[\max X_i \geq x] \leq \frac{1 - \exp(-nG(x))}{1-\epsilon} \ .$$

Proof. For every $x \in \mathcal{R}^{\geq 0}$ we have:

$$\begin{aligned} Pr[\max X_i \geq x] &= 1 - F(x)^n \\ &= 1 - (1 - G(x))^n \\ &= 1 - \left(\frac{1}{1 - G(x)}\right)^{-n} \\ &= 1 - \left(1 + \frac{G(x)}{1 - G(x)}\right)^{-n} \\ &\leq 1 - \exp\left(\frac{-nG(x)}{1 - G(x)}\right) \quad . \end{aligned} \tag{6.3}$$

We complete the proof by proving for every $\epsilon > 0$ that there exists an n_ϵ such that for every $n \geq n_\epsilon$ and $0 \leq z \leq 1$, the ratio between $A(n, z) = 1 - \exp(-nz/(1 - z))$ and $B(n, z) = 1 - \exp(-nz)$ is no more than $1/(1 - \epsilon)$.

For every n and z there are two cases:

- If $ln(n)/n \leq z \leq 1$ then we have:

$$\frac{A(n,z)}{B(n,z)} \leq \frac{1}{B(n,z)} \leq \frac{1}{1 - \exp(-ln(n))} = \frac{1}{1 - 1/n} \quad . \tag{6.4}$$

- If $0 \leq z \leq ln(n)/n$, we use partial derivatives of the functions to find an upper bound of their ratio. In the following the derivative of a function is with respect to variable z.

$$\begin{aligned}
\frac{A(n,z)}{B(n,z)} &= \frac{\int_0^z A'(n,w)dw}{\int_0^z B'(n,w)dw} \\
&= \frac{\int_0^z B'(n,w)\frac{A'(n,w)}{B'(n,w)}dw}{\int_0^z B'(n,w)dw} \\
&\leq \frac{\int_0^z B'(n,w)dw}{\int_0^z B'(n,w)dw} \cdot \max_{0\leq w\leq z}\left\{\frac{A'(n,w)}{B'(n,w)}\right\} \\
&= \max_{0\leq w\leq z}\left\{\frac{A'(n,w)}{B'(n,w)}\right\} \\
&= \max_{0\leq w\leq z}\left\{\frac{n\exp(-nz/(1-z))/(1-z)^2}{n\exp(-nz)}\right\} \\
&= \max_{0\leq w\leq z}\left\{\frac{\exp(-nz^2/(1-z))}{(1-z)^2}\right\} \\
&\leq \frac{1}{(1-z)^2} \\
&\leq \frac{1}{1-2\ln(n)/n+ln^2(n)/n^2} \quad . \qquad (6.5)
\end{aligned}$$

Note that the denominators of both (6.4) and (6.5) become greater than $1-\epsilon$ as n becomes greater than some n_ϵ, thus the proof of the lemma follows. □

Lemma 6.2 gives us an upper bound on $Pr[\max X_i \geq x]$. Now we aim to find a lower bound for $Pr[X_\tau \geq x]$. Through these two bounds we are able to find a lower bound on the approximation factor of the algorithm.

In Lemma 6.1 we showed that the thresholds are decreasing. Hence for an $x \in \mathcal{R}^{\geq 0}$, if $x < \theta_n$ then $Pr[X_\tau \geq x]$ is equal to $Pr[X_\tau \geq \theta_n]$ because the algorithm never selects an item below that value. Moreover, $Pr[X_\tau \geq \theta_n]$ is equal to $Pr[\tau \leq n]$ which is equal to $1 - Pr[\tau > n] = 1 - q_n = 1 - h(1)$. The first property of α-strong functions ensures that this number is at least α. Since $Pr[\max X_i \geq x]$ is no more than 1, therefore, for every $x < \theta_n$ we have $Pr[X_\tau \geq x] \geq \alpha Pr[\max X_i \geq x]$.

Now suppose $x \in \mathcal{R}^{\geq 0}$ and $x \geq \theta_n$. For $Pr[X_\tau \geq x]$ we have,

$$\begin{aligned} Pr[X_\tau \geq x] &= \sum_{i=1}^{n} Pr[X_\tau \geq x | \tau = i] Pr[\tau = i] \\ &= \sum_{i=1}^{n} q_{i-1}(1 - F(\max\{\theta_i, x\})) \ . \end{aligned} \tag{6.6}$$

Since the thresholds are decreasing, there exists a unique index $j(x)$ for which $\theta_{j(x)} > x \geq \theta_{j(x)+1}$. For the sake of simplicity we assume there is an imaginary item X_0 for which $\theta_0 = \infty$. In this way $j(x)$ is an integer number from 0 to $n-1$. By expanding (6.6) we have:

$$\begin{aligned} Pr[X_\tau \geq x] &= \sum_{i=1}^{n} q_{i-1}(1 - F(\max\{\theta_i, x\})) \\ &= \sum_{i=1}^{n} q_{i-1} G(\max\{\theta_i, x\}) \\ &= \sum_{i=1}^{j(x)} q_{i-1} G(\theta_i) + \sum_{i=j(x)+1}^{n} q_{i-1} G(x) \ . \end{aligned} \tag{6.7}$$

The first sum in (6.7) is indeed the probability of selecting one of the first $j(x)$ items, therefore we can rewrite it as $1 - q_{j(x)}$. Hence,

$$\begin{aligned} \Pr[X_\tau \geq x] &= 1 - q_{j(x)} + \sum_{i=j(x)+1}^{n} q_{i-1} G(x) \\ &= 1 - q_{j(x)} + nG(x) \sum_{i=j(x)+1}^{n} q_{i-1} \frac{1}{n} \\ &= 1 - q_{j(x)} + nG(x) \sum_{i=j(x)+1}^{n} h((i-1)/n) \frac{1}{n} \\ &\geq 1 - q_{j(x)} + nG(x) \int_{j(x)/n}^{1} h(r) dr \ . \end{aligned} \tag{6.8}$$

The integral in (6.8) comes from the fact that h is a decreasing function and for such functions the Riemann sum of an interval is an upper bound of the integral of the function in that interval. For simplicity let $s(x) = j(x)/n$. Inequality (6.8) can be written as follows:

$$Pr[X_\tau \geq x] \geq 1 - h(s(x)) + nG(x) \int_{s(x)}^{1} h(r)dr \ . \tag{6.9}$$

In order to complete the proof of the theorem, we need to show that the right hand side of Inequality (6.9) is an approximation of $Pr[\max X_i \geq x]$. To this end, we use the following lemma.

Lemma 6.3. *For every $\epsilon > 0$ there exists an n_ϵ such that for every integer $n \geq n_\epsilon$ the following inequality holds for every $x \geq \theta_n$:*

$$1 - h(s(x)) + nG(x) \int_{s(x)}^{1} h(r)dr \geq (1-\epsilon)\alpha(1 - \exp(-nG(x))) \ .$$

Proof. Recall that $s(x) = j(x)/n$ is a number from 0 to 1. We prove the correctness of the lemma by analyzing it for two different ranges of $s(x)$. For simplicity we may refer to $s(x)$ as s in different parts of the proof. Suppose $s_0 = min(0.5, \alpha)\epsilon$. For $0 \leq s \leq s_0$ we have:

$$\begin{aligned} 1 - h(s) + nG(x) \int_{s}^{1} h(r)dr &\geq nG(x) \int_{s}^{1} h(r)dr \\ &\geq nG(x) \int_{s_0}^{1} h(r)dr \\ &= nG(x)(\int_{0}^{1} h(r)dr - \int_{0}^{s_0} h(r)dr) \\ &\geq nG(x)(\int_{0}^{1} h(r)dr - s_0) \ . \end{aligned} \tag{6.10}$$

From the second property of α-strong functions we have $\int_0^1 h(r)dr \geq \alpha$. Also, for every $z \in \mathcal{R}^{\geq 0}$ it holds that $z \geq 1 - \exp(-z)$. By using these two inequalities in

Inequality (6.10) the lemma is proved for this case:

$$\begin{aligned} 1 - h(s) + nG(x)\int_s^1 h(r)dr &\geq nG(x)(\int_0^1 h(r)dr - s_0) \\ &\geq (1 - \exp(-nG(x)))(\alpha - s_0) \\ &\geq (1 - \exp(-nG(x)))\alpha(1 - \frac{s_0}{\alpha}) \\ &\geq (1 - \exp(-nG(x)))\alpha(1 - \epsilon) \qquad s_0 \leq \alpha\epsilon \end{aligned}$$

Now what remains is the case that $s_0 < s \leq 1$. Again, for this case we want the following inequality to hold:

$$\frac{1 - h(s) + nG(x)\int_s^1 h(r)dr}{1 - \exp(-nG(x))} \geq (1 - \epsilon)\alpha \ . \tag{6.11}$$

Recall that for every $x \geq \theta_n$, $s(x) = j(x)/n$ where $j(x)$ is the greatest index for which $\theta_j > x \geq \theta_{j+1}$. Since G is a strictly decreasing function, we have $G(\theta_j) < G(x) \leq G(\theta_{j+1})$. Recall that for every $1 \leq i \leq n$ we have $q_i = q_{i-1}(1 - G(\theta_i))$, or equivalently $G(\theta_i) = 1 - q_i/q_{i-1}$. Therefore we can bound $G(x)$ as follows:

$$1 - q_j/q_{j-1} < G(x) \leq 1 - q_{j+1}/q_j \ . \tag{6.12}$$

Now, finding a lower bound for $1 - q_j/q_{j-1}$ and an upper bound for $1 - q_{j+1}/q_j$ in Inequality (6.12) gives us a lower bound and an upper bound for $G(x)$. For the lower bound we have

$$G(\theta_j) = 1 - q_j/q_{j-1} = \frac{q_{j-1} - q_j}{q_{j-1}} = \frac{-(q_j - q_{j-1})}{q_{j-1}} = \frac{-(h(s) - h(s - 1/n))}{h(s - 1/n)} \ .$$

By multiplying this fraction by n/n we get:

$$G(\theta_j) = \frac{-(\frac{h(s) - h(s-1/n)}{1/n})}{n \ \ h(s - 1/n)} \geq \frac{-h'(s - 1/n)}{n \ \ h(s - 1/n)} \ , \tag{6.13}$$

where the last inequality in (6.13) comes from the concavity of h. From the second property of threshold functions there exists some $\delta_0 \leq s_0$ such that for every $n \geq 1/\delta_0, s_0 + 1/n \leq s \leq 1$ the following inequality holds:

$$\frac{-h'(s-1/n)}{h(s-1/n)} \geq (1-s_0)\frac{-h'(s)}{h(s)} \quad . \tag{6.14}$$

By using Inequality (6.14) in Inequality (6.13), and by using that inequality in Inequality (6.12), we get:

$$G(x) > (1-s_0)\frac{-h'(s)}{n \;\, h(s)} \quad . \tag{6.15}$$

Similarly one can show the following upper bound on $G(x)$:

$$\begin{aligned}
G(x) &\leq G(\theta_{j+1}) \\
&= 1 - \frac{q_{j+1}}{q_j} \\
&= \frac{-(h(s+1/n) - h(s))}{h(s)} \\
&= \frac{-\frac{h(s+1/n)-h(s)}{1/n}}{n \;\, h(s)} && \text{multiplying by } \frac{n}{n} \\
&\leq \frac{-\frac{h(s+1/n)-h(s)}{1/n}}{n \;\, h(s+1/n)} && \text{since } h \text{ is decreasing} \\
&\leq \frac{-h'(s+1/n)}{n \;\, h(s+1/n)} && \text{concavity of } h \qquad (6.16) \\
&\leq \frac{1}{1-s_0} . \frac{-h'(s)}{n \;\, h(s)} \quad . && \text{second property of threshold functions} \qquad (6.17)
\end{aligned}$$

Using these bounds and the following auxiliary lemma we prove the correctness of Inequality (6.11).

Lemma 6.4. *For every $z < 0$ and $t \geq 1$ we have: $(1 - \exp(zt))/(1 - \exp(z)) \leq t$.*

Proof. Let $A(z) = 1 - \exp(zt)$ and $B(z) = 1 - \exp(z)$. In the following the derivatives are with respect to z. For the ratio of these functions we have:

$$\begin{aligned}
\frac{A(z)}{B(z)} &= \frac{\int_0^z A'(w)dw}{\int_0^z B'(w)dw} \\
&= \frac{\int_0^z B'(w)\frac{A'(w)}{B'(w)}dw}{\int_0^z B'(w)dw} \\
&\leq \max_{z\leq w\leq 0}\left\{\frac{A'(w)}{B'(w)}\right\}\frac{\int_0^z B'(w)dw}{\int_0^z B'(w)dw} \\
&\leq \max_{z\leq w\leq 0}\left\{\frac{-t\exp(tw)}{-\exp(w)}\right\} \\
&= \max_{z\leq w\leq 0}\left\{t\exp(w(t-1))\right\} .
\end{aligned}$$

Since $w \leq 0$ and $t \geq 1$ the $\exp(w(t-1)) \leq 1$, and the proof follows. □ By applying the bounds of Inequalities (6.15) and (6.17) to the left hand side of Inequality (6.11) we have:

$$\begin{aligned}
&\frac{1 - h(s) + nG(x)\int_s^1 h(r)dr}{1 - \exp(-nG(x))} \\
&\geq \frac{1 - h(s) - (1-s_0)h'(s)/h(s)\int_s^1 h(r)dr}{1 - \exp(-nG(x))} && \text{Inequality (6.15)} \\
&\geq \frac{(1-s_0)(1 - h(s) - h'(s)/h(s)\int_s^1 h(r)dr)}{1 - \exp(-nG(x))} && 1 - h(s) \geq 0 \\
&\geq \frac{(1-s_0)(1 - h(s) - h'(s)/h(s)\int_s^1 h(r)dr)}{1 - \exp(\frac{h'(s)}{h(s)(1-s_0)})} && \text{Inequality (6.17)} .
\end{aligned}$$

By applying Lemma 6.4 to the denominator we have:

$$\frac{1 - h(s) + nG(x)\int_s^1 h(r)dr}{1 - \exp(-nG(x))} \geq (1-s_0)^2 . \frac{1 - h(s) - h'(s)/h(s)\int_s^1 h(r)dr}{1 - \exp(h'(s)/h(s))} .$$

From the third property of α-strong functions, the fraction at the right hand side of the above inequality is at least α. Moreover, since $s_0 \leq 0.5\epsilon$ it holds that $(1-s_0)^2 \geq (1-\epsilon)$, and thus Inequality (6.11) holds and the proof of the lemma is complete. □

To wrap up the proof of the theorem we combine the results of the previous lemmas. Suppose n_1 and n_2 are the lower bounds of Lemma 6.2 and Lemma 6.3 for n, respectively, such that their inequalities hold for $\epsilon/2$. For every $n \geq n_\epsilon = \max\{n_1, n_2\}$ we have:

$$\begin{aligned}
Pr[X_\tau \geq x] &\geq 1 - h(s(x)) + nG(x)\int_{s(x)}^{1} h(r)dr && \text{Inequality (6.9)} \quad (6.18)\\
&\geq (1-\epsilon/2)\alpha(1-\exp(-nG(x))) && \text{Lemma 6.3} \quad (6.19)\\
&\geq (1-\epsilon/2)^2\alpha \;\; Pr[\max X_i \geq x] && \text{Lemma 6.2} \quad (6.20)\\
&\geq (1-\epsilon)\alpha \;\; Pr[\max X_i \geq x] \;\; .
\end{aligned}$$

This shows that for every non-negative value of x the chance of the algorithm in selecting an item with value at least x is an approximation of the corresponding probability for the optimum solution. More specifically, we showed that for every $n \geq n_\epsilon$ and for every $x \geq 0$ the integrand of (6.2) is a $(1-\epsilon)\alpha$-approximation of the integrand of (6.1), hence the theorem is proved. $\square$

Now we have all the materials needed to prove Theorem 6.5. In order to prove the theorem, we show that the function $h(s) = \cos(as)$ is an α-strong function, where $a \approx 1.306$ is a root of $\cos(a) + \sin(a)/a - 1$ and $\alpha = 1 - \cos(a) \approx 0.7388$. To this end, we first need to show that this function is a threshold function:

i. To show the concavity of h it suffices to show that its second derivative is negative for every $0 < s \leq 1$. Note that $h'(s) = -a\sin(as)$ and $h"(s) = -a^2\cos(as)$.

ii. The ratio of $h'(s)/h(s)$ for every s is equal to $-a\tan(as)$. For every ϵ we need to show that there exists some $\delta_0 \leq \epsilon$ such that for every $\delta \leq \delta_0$ and $\epsilon + \delta \leq s \leq 1$ the following holds:

$$-a\tan(a(s-\delta)) \leq -(1-\epsilon)a\tan(as)$$

or equivalently, by dividing both sides to $-a$ and changing the direction of the inequality we want to have:

$$\tan(as - a\delta)) \geq (1-\epsilon)\tan(as) \ .$$

Note that $\tan(as)$ is a convex function because $\tan"(as) = 2\tan(as)\sec^2(as) \geq 0$ for $0 \leq s \leq 1$. For every $0 \leq \delta \leq s$ in such functions we have:

$$\frac{\tan(as) - \tan(as - a\delta)}{a\delta} \leq \tan'(as) = \sec^2(as) \leq sec^2(a) \ .$$

Therefore,

$$\tan(as) \leq \tan(as - a\delta) + a\delta\sec^2(a) \ .$$

By multiplying both sides by $(1-\epsilon)$ and assuming that $\delta \leq \delta_0 = \frac{\epsilon\tan(a\epsilon)}{a(1-\epsilon)\sec^2(a)}$ we have:

$$\begin{aligned}(1-\epsilon)\tan(as) &\leq (1-\epsilon)(\tan(as-a\delta) + a\delta\sec^2(a))\\ &\leq \tan(as-a\delta) - \epsilon\tan(as-a\delta) + (1-\epsilon)a\delta\sec^2(a)\\ &\leq \tan(as-a\delta) - \epsilon\tan(as-a\delta) + \epsilon\tan(a\epsilon)\\ &= \tan(as-a\delta) - \epsilon(\tan(a(s-\delta)) - \tan(a\epsilon)) \qquad (6.21)\end{aligned}$$

Note that $\tan(x)$ is an increasing function, therefore for every $s \geq \epsilon+\delta$ Inequality (6.21) is less than or equal to $\tan(as - a\delta)$, thus the second property holds as well.

We showed that $h(s) = \cos(as)$ is a threshold function. Now we prove that this threshold function is also an α-strong function. Due to definition $\alpha = 1 - cos(a) = 1 - h(1)$, thus the first property holds. Moreover, $\int_0^1 h(r)dr = sin(a)/a$. Again, due to definition a is a root of $cos(a) + sin(a)/a - 1$, and thus $sin(a)/a = 1 - cos(a) = \alpha$. Now we only need to show that the third property of α-strong functions holds. To do so, we need to show that:

$$1 - \cos(as) + a\tan(as)[\sin(a)/a - \sin(as)/a] \geq \alpha(1 - \exp(-a\tan(as))) \ . \quad (6.22)$$

By subtracting $\alpha(1 - \exp(-a\tan(as)))$ from both sides and multiplying them by $\cos(as)$ we have:

$$\cos(as) - \cos(as)^2 + \sin(a)sin(as) - \sin(as)^2 - \alpha\cos(as)$$
$$+ \alpha\cos(as)\exp(-a\tan(as))) \geq 0.$$

Note that $\cos^2(as) + \sin^2(as) = 1$, therefore the above inequality is equivalent to:

$$(1 - \alpha)\cos(as) + \sin(a)\sin(as) + \alpha\cos(as)\exp(-a\tan(as)) \geq 1 \ .$$

Since $\sin(a)/a = 1 - \cos(a) = \alpha$ we can replace $\sin(a)$ with αa. Also, from the relation between trigonometric functions we have $\cos(x) = 1/\sqrt{1 + \tan^2(x)}$ and $\sin(x) = \tan(x)/\sqrt{1 + \tan^2(x)}$. By considering these equalities and assuming that $w = \tan(as)$ the above inequality becomes simplified as follows:

$$\frac{1 - \alpha}{\sqrt{1 + w^2}} + \frac{\alpha a w}{\sqrt{1 + w^2}} + \frac{\alpha\exp(-aw)}{\sqrt{1 + w^2}} \geq 1 \ .$$

By multiplying both sides by $\sqrt{1 + w^2}$ and raising them to the power of two, and subtracting $1 + w^2$ from both sides we have:

$$(1 - \alpha + \alpha a w + \alpha\exp(-aw))^2 - 1 - w^2 \geq 0 \ .$$

Now we use the following lemma to finish the proof.

Lemma 6.5. *Suppose* $A(w) = (1-\alpha+\alpha a w+\alpha \exp(-aw))^2-1-w^2$ *where* $a \approx 1.306$ *is a root of* $\cos(a) + \sin(a)/a - 1$ *and* $\alpha = 1 - \cos(a) \approx 0.7388$. *Then for every* $0 \leq w \leq \tan(a)$ *we have* $A(w) \geq 0$.

Proof. Let us first take a look at the first three derivatives of $A(w)$ which are all continuous and bounded in range $[0, \tan(a)]$:

$$A'(w) = 2\alpha a(1 - \exp(-aw))(1 - \alpha + \alpha a w + \alpha \exp(-aw)) - 2w,$$

$$A''(w) = 2\alpha a^2 \exp(-aw)(1 - 3\alpha + \alpha a w + 2\alpha \exp(-aw)) + 2(\alpha^2 a^2 - 1),$$

$$A'''(w) = -2\alpha a^3 \exp(-aw)(1 - 4\alpha + \alpha a w + 4\alpha \exp(-aw)) \ .$$

In this part of the proof we frequently use one of the implications of intermediate value theorem: if $f(x)$ and $f'(x)$ are two continuous and bounded functions, then there exists a root of $f'(x)$ between every two roots of $f(x)$. This also implies that the number of the roots of $f(x)$ is at most one plus the number of the roots of $f'(x)$.

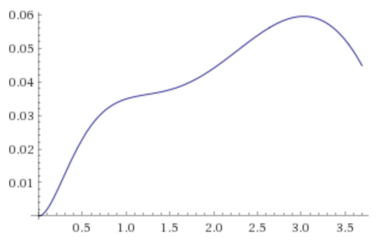


Figure 6.1: The plot shows function $A(w)$ for values of w from 0 to $\tan(a) \approx 3.7$.

We claim that $A'''(w)$ has at most two roots. The reason for this is because $-2\alpha a^3 \exp(-aw)$ is always non-zero, and $1 - 4\alpha + \alpha a w + 4\alpha \exp(-aw)$ has at most two roots, because its derivative, $\alpha a(1-4\exp(-aw))$ has exactly one root, which is $\ln(4)/a$.

The fact that $A'''(w)$ has at most two roots implies that $A''(w)$ has at most three roots, which are $w_1 \approx 0.28157$, $w_2 \approx 1.24251$, and $w_3 \approx 2.27082$. We note that $A'(w)$ is positive at all these points. Therefore $A'(w)$ has at most two roots, because otherwise there would be a point in which $A'(w) \leq 0$ and $A''(w) = 0$ which is impossible.

Note that $A'(0) = 0$, therefore $A'(w)$ has at most one root in $\mathcal{R}^+$. Now we note that $A(0^+) > 0$ because $A'(0) = 0$ and $A''(0) = 2(\alpha a^2(1-\alpha) + \alpha^2 a^2 - 1) > 0$. Also $A(\tan(a)) > 0$. Now if $A(w) < 0$ for some $0 < w < \tan(a)$, then $A(w)$ would have at least two roots in range $(0, \tan(a))$ which results in $A'(w)$ having two roots in $\mathcal{R}^+$. Since this is not true, we have $A(w) \geq 0$ for every $0 \leq w \leq \tan(a)$. $\square$

Lemma 6.5 shows that this inequality holds for every $0 \leq w \leq \tan(a)$. Consequently, Inequality (6.22) holds for every $0 \leq s \leq 1$. This completes the proof that $h(s) = \cos(as)$ is an α-strong function for $\alpha \approx 0.7388$, since it has all the three properties.

6.3 Non IID Distributions

In this section we study more generalized cases of the prophet inequalities problem. Suppose $X_1, \ldots, X_n$ are random variables from distribution functions $F_1, \ldots, F_n$. Similar to Section 6.2 we assume, for the sake of simplicity, that all distribution functions are continuous and strictly increasing on a subinterval of $\mathcal{R}^+$.

The goal of this section is to show improving results for the best order and a random order of *large market* instances. We use the term large market as a general term to refer to instances with repeated distributions. The following definition formally captures this concept.

Definition 6.3. *A set of n items with distribution functions $F_1, \ldots, F_n$ is m-frequent if for every item in this set there are at least $m-1$ other items with the same distribution function.*

In the remainder of this section we show for the best order and a random order of a large market instance that one can find a sequence of thresholds which in expectation performs as good as our algorithm for iid items. Roughly speaking, we design algorithms that are α-approximation for large enough m-frequent instances, where $\alpha \approx 0.7388$. The following two theorems formally state our results for the best order and a random order, respectively.

Theorem 6.7. *For every $\epsilon > 0$ and set $\mathbb{X}$ of n items, there exists a number m_ϵ (a function of ϵ and independent of n) such that if $\mathbb{X}$ is m-frequent for $m \geq m_\epsilon$ then there exits an algorithm which is $(1-\epsilon)\alpha$-approximation on a permutation of $\mathbb{X}$.*

Theorem 6.8. *For every $\epsilon > 0$ and set $\mathbb{X}$ of n items there exists a number c_ϵ (a function of ϵ and independent of n) such that if $\mathbb{X}$ is m-frequent for $m \geq c_\epsilon \log(n)$ then there exists an algorithm which in expectation is $(1-\epsilon)\alpha$-approximation on a random permutation of $\mathbb{X}$.*

To prove the theorems we first provide an algorithm for a specific class of large market instances, namely partitioned sequences. Lemma 6.6 states that this algorithm is α-approximation when the number of partitions is large. We later show how to apply this algorithm on the best order and a random order of large market instances to achieve a similar approximation factor. Following is a formal definition of partitioned sequences.

Definition 6.4. *A sequence of items with distribution functions $F_1, \ldots, F_n$ is m-partitioned if $n = mk$ and the sequence of functions $F_{ik+1}, \ldots, F_{ik+k}$ is a permutation of $F_1, \ldots, F_k$ for every $0 \le i < m$.*

The following algorithm exploits Algorithm 0 for iid items in order to find thresholds for a partitioned large market instance.

Algorithm 7

Input: An m-partitioned sequence of items with distribution functions $F_1, \ldots, F_n$.

```
1: Let $k = n/m$.
2: Let $F(x) = \prod_{i=1}^{k} F_i(x)$.
3: Let $\theta_1 \ldots, \theta_m$ be the thresholds by Algorithm 0 for $m$ iid items with distribution
   function $F$.
4: Pick the first item $i$ if $X_i \ge \theta_{\lceil i/k \rceil}$.
```

Lemma 6.6. *For every $\epsilon > 0$ there exists a number m_ϵ (a function of ϵ and independent of the number of items) such that for every $m \ge m_\epsilon$ Algorithm 0 is $(1-\epsilon)\alpha$-approximation on an m-partitioned input.*

Proof. Let $X_1, \ldots, X_n$ be random variables representing the items, and let $Y_1, \ldots, Y_m$ be iid random variables with distribution function $F(x) = \prod_{i=1}^{k} F_i(x)$.

For the expectation of the maximum of these variables we have:

$$\begin{aligned}
\mathcal{E}[\max_{i=1}^{k} Y_i] &= \int_0^\infty Pr[\max_{i=1}^{k} Y_i \geq x]dx \\
&= \int_0^\infty \left(1 - \prod_{i=1}^{m} F(x)\right)dx \\
&= \int_0^\infty \left(1 - \prod_{i=1}^{m}\prod_{j=1}^{k} F_j(x)\right)dx \\
&= \int_0^\infty \left(1 - \prod_{i=1}^{n} F_i(x)\right)dx \\
&= \int_0^\infty Pr[\max_{i=1}^{n} X_i \geq x]dx \\
&= \mathcal{E}[\max_{i=1}^{n} X_i] \ . \qquad (6.23)
\end{aligned}$$

This shows that the optimum solution is the same for both sets of items. Let τ_Y and τ_X be random variables that denotes the index of the picked items in $Y_1, \ldots, Y_m$ and $X_1 \ldots, X_n$ respectively. Theorem 6.5 states that there exists some s such for every $m \geq s$, we have $E[Y_{\tau_Y}] \geq (1-\epsilon/2)\alpha E[\max_{i=1}^{m} Y_i]$. In the following we show that there exist some m_2 such that for every $m \geq m_2$, $E[X_{\tau_X}] \geq (1-\epsilon/2)E[Y_{\tau_Y}]$. This proves the lemma for every $m \geq m_\epsilon = \max\{s, m_2\}$, i.e.

$$\begin{aligned}
\mathcal{E}[X_{\tau_X}] \geq (1-\epsilon/2)\mathcal{E}[Y_{\tau_Y}] &\geq (1-\epsilon/2)^2\alpha\mathcal{E}[\max_{i=1}^{m} Y_i] \\
&\geq (1-\epsilon)\alpha\mathcal{E}[\max_{i=1}^{m} Y_i] = (1-\epsilon)\alpha\mathcal{E}[\max_{i=1}^{n} X_i].
\end{aligned}$$

Since for every non-negative random variable Z, $\mathcal{E}[Z] = \int_0^\infty Pr[Z \geq z]dz$, therefore, in order to show $\mathcal{E}[X_{\tau_X}] \geq (1-\epsilon/2)\mathcal{E}[Y_{\tau_Y}]$ we show $Pr[X_{\tau_X}] \geq (1-\epsilon/2)Pr[Y_{\tau_Y}]$ for every $x \geq 0$.

In the following, we use $G_i(x)$ to denote $1 - F_i(x)$. For every non-negative x

we have:

$$\begin{aligned}Pr[X_{\tau_X} \geq x] &= \sum_{i=1}^{n} Pr[X_{\tau_X} \geq x | \tau_X = i] Pr[\tau_X = i] \\ &= \sum_{i=1}^{n} \prod_{j=1}^{i-1} F_j(\theta_{\lceil j/k \rceil}) G_i(\max\{x, \theta_{\lceil i/k \rceil}\}) \ .\end{aligned}$$

By rewriting the above sum with respect to the m partitions we have:

$$\begin{aligned}Pr[X_{\tau_X} \geq x] &= \sum_{i=0}^{m-1} \sum_{j=1}^{k} \prod_{l=1}^{ik+j-1} F_l(\theta_{\lceil l/k \rceil}) G_{ik+j}(\max\{x, \theta_{i+1}\}) \\ &= \sum_{i=0}^{m-1} \sum_{j=1}^{k} \left(\prod_{l=1}^{ik} F_l(\theta_{\lceil l/k \rceil}) \prod_{p=1}^{j-1} F_{ik+p}(\theta_{i+1}) G_{ik+j}(\max\{x, \theta_{i+1}\}) \right) \\ &= \sum_{i=0}^{m-1} \left(\prod_{l=1}^{ik} F_l(\theta_{\lceil l/k \rceil}) \sum_{j=1}^{k} \prod_{p=1}^{j-1} F_{ik+p}(\theta_{i+1}) G_{ik+j}(\max\{x, \theta_{i+1}\}) \right) \ .\end{aligned} \tag{6.24}$$

Note that $X_1, \ldots, X_n$ are m-partitioned, hence for every partition $0 \leq i < m$ and $x \geq 0$ we have $\prod_{l=ik+1}^{ik+k} F_l(x) = F(x)$. Therefore $\prod_{l=1}^{ik} F_l(\theta_{\lceil l/k \rceil}) = \prod_{l=1}^{i} F(\theta_l)$. By this replacement, Inequality (6.24) can be written as follows:

$$Pr[X_{\tau_X} \geq x] = \sum_{i=0}^{m-1} \left(\prod_{l=1}^{i} F(\theta_l) \sum_{j=1}^{k} \prod_{p=1}^{j-1} F_{ik+p}(\theta_{i+1}) G_{ik+j}(\max\{x, \theta_{i+1}\}) \right) \ . \tag{6.25}$$

Moreover, for every $0 \leq i < m$ and $1 \leq j \leq k$ we have:

$$\begin{aligned}\prod_{p=1}^{j-1} F_{ik+p}(\theta_{i+1}) &\geq \prod_{p=1}^{k} F_{ik+p}(\theta_{i+1}) && \textit{every } F_t(x) \textit{ is at most } 1 \\ &= F(\theta_{i+1}) \\ &= 1 - G(\theta_{i+1}) \\ &\geq 1 - \frac{a\tan(a)}{m} && \textit{inequality 6.16 for } h(s) = \cos(as) \\ &\geq 1 - \epsilon/2 && \textit{for every } m \geq m_2 = \frac{2a\tan(a)}{\epsilon} \ .\end{aligned} \tag{6.26}$$

Inequality (6.26) shows that for a large enough m, the left hand side of the inequality becomes close enough to 1. By using this inequality in Inequality (6.25) we have:

$$\begin{aligned} Pr[X_{\tau_X} \geq x] &\geq \sum_{i=0}^{m-1} \left(\prod_{l=1}^{i} F(\theta_l) \sum_{j=1}^{k} (1-\epsilon/2) G_{ik+j}(\max\{x, \theta_{i+1}\}) \right) \\ &= (1-\epsilon/2) \sum_{i=0}^{m-1} \left(\prod_{l=1}^{i} F(\theta_l) \sum_{j=1}^{k} G_{ik+j}(\max\{x, \theta_{i+1}\}) \right) \ . \end{aligned} \tag{6.27}$$

Let $r = \max\{x, \theta_{j+1}\}$. By multiplying every term in Inequality (6.27) by $\prod_{p=1}^{j-1} F_{ik+p}(r)$, which is less than or equal to 1, we have:

$$\begin{aligned} Pr[X_{\tau_X} \geq x] &\geq (1-\epsilon/2) \sum_{i=0}^{m-1} \left(\prod_{l=1}^{i} F(\theta_l) \sum_{j=1}^{k} G_{ik+j}(r) \right) \\ &\geq (1-\epsilon/2) \sum_{i=0}^{m-1} \left(\prod_{l=1}^{i} F(\theta_l) \sum_{j=1}^{k} \prod_{p=1}^{j-1} F_{ik+p}(r) G_{ik+j}(r) \right) \\ &= (1-\epsilon/2) \sum_{i=0}^{m-1} \left(\prod_{l=1}^{i} F(\theta_l) \sum_{j=1}^{k} \prod_{p=1}^{j-1} F_{ik+p}(r)(1 - F_{ik+j}(r)) \right) \\ &= (1-\epsilon/2) \sum_{i=0}^{m-1} \left(\prod_{l=1}^{i} F(\theta_l) \sum_{j=1}^{k} \left(\prod_{p=1}^{j-1} F_{ik+p}(r) - \prod_{p=1}^{j} F_{ik+p}(r) \right) \right) \\ &= (1-\epsilon/2) \sum_{i=0}^{m-1} \left(\prod_{l=1}^{i} F(\theta_l) \left(1 - \prod_{p=1}^{k} F_{ik+p}(r) \right) \right) \\ &= (1-\epsilon/2) \sum_{i=0}^{m-1} \left(\prod_{l=1}^{i} F(\theta_l) \Big(1 - F(r) \Big) \right) \\ &= (1-\epsilon/2) \sum_{i=0}^{m-1} \prod_{l=1}^{i} F(\theta_l) G(r) \ . \end{aligned} \tag{6.28}$$

Note that $Pr[Y_{\tau_Y} \geq x] = \sum_{i=0}^{m-1} \prod_{l=1}^{i} F(\theta_l) G(\max\{x, \theta_{i+1}\})$. Using this in Inequality (6.28) results that $Pr[X_{\tau_X} \geq x] \geq (1-\epsilon/2) Pr[Y_{\tau_Y} \geq x]$, hence the proof is complete. □

Now we are ready to prove Theorem 6.7 and Theorem 6.8.

Proof of Theorem 6.7: Let s be the lower bound on the number of partitions in Lemma 6.6 for $\epsilon/2$, and let $m_\epsilon = 2(s-1)/\epsilon$. The outline of the proof is as follows. Let $\mathbb{X}$ be an m-frequent set of items for $m \geq m_\epsilon$. We uniformly group the items into s parts with $\lfloor m/s \rfloor$ items of each type in every group. Let $\mathbb{Y}$ denote the set of partitioned items. In order to make all parts similar, we may need to discard some of the items, however, we show this does not hurt the approximation factor significantly. Finally, by applying Algorithm 0 to $\mathbb{Y}$ we achieve the desired approximation factor.

The following lemma shows that discarding a fraction of items influences the approximation factor proportionally.

Lemma 6.7. *Let $\{X_1, \dots, X_n\}$ be a k-frequent set of items. Suppose for some $S \subseteq \{1, \dots, n\}$ that the set $\{X_{S_1}, \dots, X_{S_r}\}$ is p-frequent and contains every X_i for $1 \leq i \leq n$. Then we have*

$$\mathcal{E}[\max_{i \in S} X_i] \geq \frac{p}{k} \mathcal{E}[\max_{1 \leq i \leq n} X_i] \ .$$

Proof. Let ρ and ρ' be random variables that denote the index of the maximum with the smallest index amongst $X_1, \dots, X_n$ and $X_{S_1}, \dots, X_{S_r}$, respectively. Then,

$$\begin{aligned}
\mathcal{E}[\max_{i \in S} X_i] &= \sum_{i \in S} \mathcal{E}[X_i | i = \rho'] Pr[i = \rho'] \\
&\geq \sum_{i \in S} \mathcal{E}[X_i | i = \rho] Pr[i = \rho] \\
&\geq \frac{p}{k} \sum_{i=1}^{n} \mathcal{E}[X_i | i = \rho] Pr[i = \rho] \\
&= \frac{p}{k} \mathcal{E}[\max_{1 \leq i \leq n} X_i] \ .
\end{aligned}$$

Therefore, the proof is complete. $\square$

Note that in partitioning $\mathbb{X}$ to s groups there might be at most $s-1$ items of each type being discarded in $\mathbb{Y}$, therefore $\mathbb{Y}$ is $(m-s+1)$-frequent. Let ALG be a random variable that denotes the value of the item picked by our algorithm. We have:

$$\begin{aligned}
\mathcal{E}[ALG] &\geq (1-\epsilon/2)\alpha\mathcal{E}[\max_{Y\in\mathbb{Y}} Y] && \text{Lemma 6.6}\\
&\geq (1-\epsilon/2)\alpha\frac{m-s+1}{m}\mathcal{E}[\max_{X\in\mathbb{X}} X] && \text{Lemma 6.7}\\
&\geq (1-\epsilon/2)^2\alpha\mathcal{E}[\max_{X\in\mathbb{X}} X]\\
&\geq (1-\epsilon)\alpha\mathcal{E}[\max_{X\in\mathbb{X}} X] \enspace .
\end{aligned}$$

Therefore, for every m-frequent set $\mathbb{X}$ there exists an ordering of its items on which our algorithm is $(1-\epsilon)\alpha$-approximation. $\square$

Proof of Theorem 6.8: Let π be a random permutation of the items. Consider s different partitions for the items, i.e. one from X_{π_1} to $X_{\pi_{n/s}}$, one from $X_{\pi_{n/s+1}}$ to $X_{\pi_{2n/s}}$, so on so forth. We show that when the number of similar items is large enough then a random permutation is very likely to uniformly distribute similar items into these parts. Therefore, by discarding a small fraction of the items $X_{\pi_1}, \ldots, X_{\pi_n}$ can be assumed as an s-partitioned sequence, hence Algorithm 0 can be applied to it.

Note that $\mathbb{X}$ is m-frequent, which means that for every item i there are at least $m-1$ other items with the same distribution functions as F_i. We refer to a set of similar items as a type. Therefore, there are at least m items of every type in $\mathbb{X}$.

We use the following lemma to show for every type that with a high probability the number of items of that type in every partition is almost m/s.

Lemma 6.8 ([151]). *Let $x_1, \ldots, x_m$ be a sequence of negatively correlated boolean (i.e. 0 or 1) random variables, and let $X = \sum_{i=1}^{m} x_i$. We have:*

$$Pr[|X - \mathcal{E}[X]| \geq \delta\mathcal{E}[X]] \leq 3\exp(-\delta^2\mathcal{E}[X]/3) \ .$$

Since π is a random permutation, the expected number of these items in a fixed partition is m/s. Using Lemma 6.8, with probability at most $3\exp(\frac{-\delta^2 m}{3s})$ there are less than $(1-\delta)m/s$ of these items in a fixed partition. Using Union Bound on all the s partitions and all types of items (note that there at at most n/m types), with probability at most $3s\frac{n}{m}\exp(\frac{-\delta^2 m}{3s})$ there is a type of item which has less than $(1-\delta)m/s$ items in a partitions. If we choose $\delta = \epsilon/3$ then for every $m \geq \frac{3s}{\delta^2}(\log(n) + \log(\frac{9}{\epsilon}))$ this probability becomes less than $\epsilon/3$.

Now we are ready to wrap up the proof. If we choose $s = m_{\epsilon/3}$ using Lemma 6.6, $\delta = \epsilon/3$, and $c_\epsilon = \frac{3s\log(9/\epsilon)}{\delta^2}$ then for every $m \geq c_\epsilon \log(n)$ with probability at least $(1-\epsilon/3)$ there are at least $(1-\epsilon/3)m/s$ items of each type in every partitions. In such cases by discarding at most $\epsilon/3$ fraction of the items of each type we have exactly $(1-\epsilon/3)m/s$ of them in each partition. Lemma 6.7 states that removing this fraction of items changes the approximation factor by at most $(1-\epsilon/3)$. This means that for a random permutation of the items, with probability at least $(1-\epsilon/3)$ we can loose on the approximation factor by no worse than $(1-\epsilon/3)$ and have an s-partitioned sequence. Due to Lemma 6.6, Algorithm 0 is $(1-\epsilon/3)\alpha$-approximation on this number of partitions. Therefore, the approximation factor of our method is

$(1-\epsilon/3)^3\alpha$ which is more than $(1-\epsilon)\alpha$. □

6.4 Hardness

Proof of Theorem 6.4: Pick arbitrary numbers m and ϵ. Suppose we have $2m$ distribution. Each of the first m distributions gives 1 with probability 1. Each of the last m distributions gives 0 with probability $(1-\epsilon)^{1/m}$ and $\frac{1}{\epsilon}$ otherwise. Notice that with probability $\left((1-\epsilon)^{1/m}\right)^m = 1-\epsilon$ all of the last m items are 0 and with probability ϵ at least one of the last m items is $\frac{1}{\epsilon}$. Hence in expectation the optimum takes

$$1 \times (1-\epsilon) + \epsilon \times \frac{1}{\epsilon} = 2 - \epsilon.$$

While any online algorithm takes at most $\max(1, \epsilon \times \frac{1}{\epsilon}) = 1$. Therefore, the approximation factor of any online algorithm is upper-bounded by

$$\frac{1}{2-\epsilon} = \frac{2+\epsilon}{4-\epsilon^2} \leq \frac{2+\epsilon}{4} \leq 0.5 + \epsilon.$$

□

Chapter 7: Bi-Objective Online Matching

7.1 Introduction

As a central optimization problem with a wide variety of applications, online resource allocation problems have attracted a large body of research in networking, distributed computing, and electronic commerce. Here, items arrive one at a time (i.e. *online*), and when each item arrives, the algorithm must irrevocably assign it to an agent; each agent has a limited resource budget / capacity for items assigned to him. A big challenge in developing good algorithms for these problems is to *predict* future binding constraints or *learn* future capacity availability, and allocate items one by one to agents who are unlikely to hit their capacity in the future. Various stochastic and adversarial models have been proposed to study such online allocation problems, and many techniques have been developed for these problems. For stochastic input, a natural approach is to build a predicted instance (for instance, via sampling, or using historical data), and some of these techniques solve a dual linear program to learn dual variables that are used by the online algorithm moving forward [131,152–157]. However, stochastic approaches may provide poor results on some input (for example, when there are unexpected spikes in supply / demand), and hence such problems have been extensively studied in adversarial models as well.

Here, the algorithm typically maintains a careful balance between greedily *exploiting* the current item by assigning it to agents with high value for it, and assigning the item to a lower-value agent for whom the value is further from the distribution of 'typical' items they have received. Again, primal-dual techniques have been applied to learn the dual variables used by the algorithm in an online manner [130,133,158].

A central practical application of such online algorithms is the online allocation of impressions or page-views to ads on the Internet [153, 154, 158–161]. Such problems are present both in the context of sponsored search advertising where advertisers have global budget constraints [130,131,133], or in display advertising where each ad campaign has a desired goal or a delivery constraint [152–154, 158–161]. Many of these online optimization techniques apply to general optimization problems including the online submodular welfare maximization problem (SWM) [162, 163].

For many real-world optimization problems, the goal is to optimize multiple objective functions [164, 165]. For instance, in Internet advertising, such objectives might include revenue, clicks, or conversions. A variety of techniques have been developed for multi-objective optimization problems; however, in most cases, these techniques are only applicable for offline multi-objective optimization problems [166, 167], and they do not apply to online settings, especially for online competitive algorithms that work against an adversarial input [133, 158] or in the presence of traffic spikes [157, 168] or hard-to-predict traffic patterns [159, 169, 170].

Our contributions. Motivated by the above applications and the increasing need to satisfy multiple objectives, we study a wide class of multi-objective online optimization problems, and present both hardness results and (almost tight) bi-objective approximation algorithms for them. In particular, we study resource allocation problems in which a sequence of items (also referred to as impressions) i from an unknown set I arrive one by one, and we have to allocate each item to one agent (for example, one advertiser) a in a given set of agents A. Each agent a has two monotone submodular set functions $f_a, g_a \colon 2^I \to R$ associated with it. Let $\mathcal{S}_a$ be the set of items assigned to bin a as a result of online allocation decisions. The goal of the online allocation algorithm is to maximize two social welfare functions based on f_a's and g_a, i.e., $\sum_{a \in A} f_a(\mathcal{S}_a)$ and $\sum_{a \in A} g_a(\mathcal{S}_a)$. We first present almost tight online approximation algorithms for the general online bi-objective submodular welfare maximization problem (see Theorems 7.2 and 7.3, and Fig. 7.1). We show that a simple random selection rule along with the greedy algorithm results in almost optimal algorithms. Our allocation rule is thus both very fast to run and easy to implement. The main technical result of this part is the hardness result showing that the achieved approximation factor is almost tight unless P=NP. Furthermore, we consider special cases of this problem motivated by online ad allocation. In particular, for the special cases of online budgeted allocation and online weighted matching, motivated by sponsored search and display advertising (respectively), we present improved primal-dual-based algorithms along with improved hardness results for these problems (see, for example, the tight Theorem 7.4).

Related Work. It is known that the greedy algorithm leads to a 1/2-approximation for the submodular social welfare maximization problem (SWM) [171], and this problem admits a $1 - 1/e$-approximation in the offline setting [172], which is tight [173]. However, for the online setting, the problem does not admit a better than 1/2-approximation algorithm unless P= NP [174]. Bi-objective online allocation problems have been studied in two previous papers [164, 165]. The first paper presents [164] an online bi-objective algorithm for the problem of maximizing a general weight function and the cardinality function, and the second paper [165] presents results for the combined budgeted allocation and cardinality constraints. Our results in this chapter improve and generalize those results for more general settings.

Our work is related to online ad allocation problems, including the *Display Ads Allocation (DA)* problem [152–154, 158], and the *Budgeted Allocation (AdWords)* problem [131, 133]. 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 with a single weight objective [158, 160, 161]. In the Budgeted Allocation 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 [130, 133, 158]. None of the above papers for adversarial models 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 [131, 152–157], and the *iid* model in which impressions arrive iid according to a known (or unknown) distribution [159, 175–178]. In such stochastic settings, primal and dual techniques have been applied to get improved approximation algorithms. These techniques are based on computing offline solutions to a predicted instance (based on learning from what has been observed so far), and using this solution online [131, 175]. It is not hard to generalize these techniques to the bi-objective online matching problem. In this extended abstract, 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 [157].

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.* [179, 180], and Buchbinder and Naor [181] In this literature, the different objectives often come from applying different functions (e.g. L_1 and L_∞ norms) on the same inputs to each objective, such as processing times or bandwidth allocations.

7.2 Bi-Objective Online Submodular Welfare Maximization

7.2.1 Model and Overview

For any allocation $\mathcal{S}$, let $\mathcal{S}_a$ denote the set of items assigned to agent $a \in A$ by this allocation. In the classic Submodular Welfare Maximization problem (SWM) for which there is a single monotone submodular objective, each agent $a \in A$ is associated with a submodular function f_a defined on the set of items I. The welfare of allocation $\mathcal{S}$ is defined as $\sum_a f_a(\mathcal{S}_a)$, and the goal of SWM is to maximize this welfare. In the classic SWM, the natural greedy algorithm is to assign each item (when it arrives) to the agent whose gain increases the most. This greedy algorithm (note that it is an online algorithm) is $(1/2 + 1/n)$-competitive, and this is the best possible [182].

In this section, we consider the extension of online SWM to *two* monotone submodular functions. Formally, each agent $a \in A$ is associated with two submodular functions - f_a and g_a - defined on I. The goal is to find an allocation S that does well on both objectives $\sum_a f_a(S_a)$ and $\sum_a g_a(S_a)$. We measure the performance of the algorithm by comparison to the offline optimum for each objective: Let $\mathcal{S}^{*f} = \arg\max_{\text{allocations } \mathcal{S}} \sum_a f_a(S_a)$ and $\mathcal{S}^{*g} = \arg\max_{\text{allocations } \mathcal{S}} \sum_a g_a(S_a)$. An algorithm A is (α, β)-competitive if, for every input, it produces an allocation $\mathcal{S}$ such that $\sum_a f_a(\mathcal{S}_a) \geq \alpha \sum_a f_a(\mathcal{S}_a^{*f})$ and $\sum_a g_a(\mathcal{S}_a) \geq \beta \sum_a g_a(S_a^{*g})$.

A $(1, 1)$-competitive algorithm would be one that finds an allocation which is simultaneously optimal in both objectives, but since the objectives are distinct, no single allocation may maximize both, even ignoring computational difficulties or lack of knowledge of the future. One could attempt to maximize a linear combination of the two submodular objectives, but since the linear combination is itself submodular, this is no harder than the classic online SWM. Instead, we provide algorithms with the stronger guarantee that they are *simultaneously* competitive with the optimal solution for each objective separately. Further, our algorithms are parametrized, so the user can balance the importance of the two objectives.

Similar to previous approaches for bi-objective online allocation problems [164], we run two simultaneous greedy algorithms, each based on one of the objective functions. Upon arrival of each online item, with probability p we pass the item to the greedy algorithm based on the objective function f, and with probability $1-p$ we pass the item to the greedy algorithm based on g.

First, as a warmup, we provide a charging argument to show that the greedy algorithm for (single-objective) SWM is 1/2-competitive. This charging argument is similar to the usual primal-dual analysis for allocation problems. However, since the objective functions are not linear, it may not be possible to interpret the proof using a primal-dual technique. Later, we modify our charging argument and show that if we run the greedy algorithm for SWM but only consider items for allocation with probability p, the competitive ratio is $\frac{p}{1+p}$. (Note that a naive analysis would yield a competitive ratio of $p/2$, since we lose a factor of p in the sampling and a factor of 1/2 due to the greedy algorithm.)

Since our algorithm for bi-objective online SWM passes items to the 'first' greedy algorithm with probability p and passes items to the second greedy algorithm with probability $1-p$, the modified charging argument immediately implies that our algorithm is $(\frac{p}{1+p}, \frac{1-p}{2-p})$ competitive, as we state in Theorem 7.2 below. Also, using a factor-revealing framework, assuming $NP \neq RP$, we provide an almost tight hardness result, which holds even if the objective functions have the simpler 'coverage' structure. Both our competitive ratio and the associated hardness result is presented in Figure 7.1.

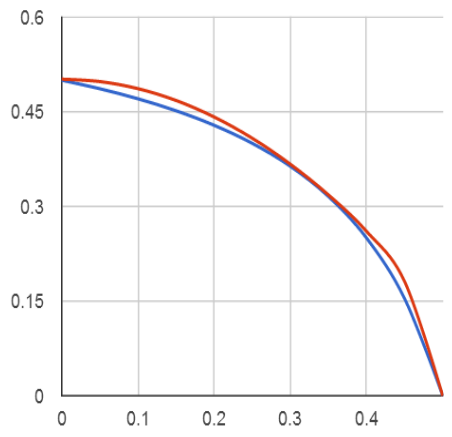


Figure 7.1: The lower (blue) curve is the competitive ratio of our algorithm, and the red curve is the upper bound on the competitive ratio of any algorithm.

7.2.2 Algorithm for Bi-Objective online SWM

We define some notation and ideas that we use to bound the competitive ratio of our algorithm. Let Gr be the greedy algorithm and let opt be a fixed optimum allocation. For an agent j, and an algorithm Alg, let Alg_j be the set of online items allocated to the agent j by Alg; opt_j denotes the set of online items allocated to j in opt. Trivially, for any two agents j and k, we have $Alg_j \cap Alg_k = \emptyset$.

For each online item i we define a variable α_i, and for each agent j we define a variable β_j. In order to bound the competitive ratio of the algorithm Alg by c, it suffices to set the values of α_is and β_js such that 1) the value of Alg is at least $c\left(\sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{m} \beta_j\right)$ and 2) the value of opt is at most $\sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{m} \beta_j$.

Theorem 7.1. *(Warmup) The greedy algorithm is* 0.5*-competitive for online SWM.*

Proof. For each online item i, let α_i be the marginal gain by Gr from allocating item i upon its arrival. It is easy to see that $\sum_{i=1}^{n} \alpha_i$ is equal to the value of *Gr*. For each agent j, let β_j be the total value of the allocation to j at the end fo the algorithm. By definition, we know that $\sum_{j=1}^{m} \beta_j$ is equal to the value of *Gr*. Thus, the value of *Gr* is clearly $0.5 \left(\sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{m} \beta_j \right)$.

Recall that $f_j(.)$ denotes the valuation function of agent j. Below, we show that $f_j(opt_j)$ is upper-bounded by $\beta_j + \sum_{i \in opt_j} \alpha_i$. Note that for distinct agents j and k, opt_j and opt_k are disjoint. Thus, by summing over all agents, we can upper-bound the value of *opt* by $\sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{m} \beta_j$. This means that the competitive ratio of *Gr* is 0.5.

Now, we just need to show that for any agent j we have $f_j(opt_j) \leq \beta_j + \sum_{i \in opt_j} \alpha_i$. Note that for any item $i \in opt_j$, the value of α_i is at least the marginal gain that would have been obtained from assigning i to j when it arrives. Applying submodularity of f_j, we have $\alpha_i \geq f_j(Gr_j \cup i) - f_j(Gr_j)$. Moreover, by definition we have $\beta_j = f_j(Gr_j)$. Thus, we have:

$$\begin{aligned}
\beta_j + \sum_{i \in opt_j} \alpha_i &\geq f_j(Gr_j) + \sum_{i \in opt_j} \left(f_j(Gr_j \cup i) - f_j(Gr_j) \right) \\
&\geq f_j(Gr_j) + \left(f_j(Gr_j \cup opt_j) - f_j(Gr_j) \right) \\
&= f_j(Gr_j \cup opt_j) \geq f_j(opt_j),
\end{aligned}$$

where the second inequality follows by submodularity, and the last inequality by monotonicity. This completes the proof. □

Lemma 7.1. *Let Gr_p be an algorithm that with probability p passes each online item to Gr for allocation, and leaves it unmatched otherwise. Gr_p is $\frac{p}{1+p}$-competitive for* *online SWM.*

Proof. The proof here is fairly similar to Theorem 7.1. For each online item i, set $\alpha_i to$ be the marginal gain that would have been achieved from allocating item i upon its arrival (assuming i is passed to Gr), given the current allocation of items. Note that α_i is a random variable (depending on the outcome of previous decisions to pass items to Gr or not), but it is independent of the coin toss that determines whether it is passed to Gr, and so the *expected* marginal gain of allocating item i, (given all previous allocations) is $pE[\alpha_i]$. Thus, by linearity of expectation, the expected value of Gr_p is $pE[\sum_{i=1}^{n} \alpha_i]$. On the other hand, for each agent j, set β_j to be the value of the *actual* allocations to j at the end of the algorithm. Again, we have $\sum_{j=1}^{m} \beta_j$ equal to the value of Gr_p. Combining these two, we conclude that the expected value of Gr_p is equal to $\frac{1}{1+1/p}\left(\sum_{i=1}^{n} E[\alpha_i] + \sum_{j=1}^{m} E[\beta_j]\right)$.

As before, we show that $f_j(opt_j)$ is upper-bounded by $\beta_j + \sum_{i \in opt_j} \alpha_i$. Therefore, we can conclude that the competitive ratio of Gr_p is $\frac{1}{1+1/p} = \frac{p}{1+p}$.

It remains only to show that for any agent j, we have $f_j(opt_j) \leq \beta_j + \sum_{i \in opt_j} \alpha_i$. This is exactly the same as our proof for Theorem 7.1: By submodularity of f_j we have, $\alpha_i \geq f_j(Gr_p(j) \cup i) - f_j(Gr_p(j))$, and by definition we have $\beta_j = f_j(Gr_p(j))$. See To complete the proof, we need to show that for any agent j, we have $f_j(opt_j) \leq$

$\beta_j + \sum_{i \in opt_j} \alpha_i$. We have:

$$\beta_j + \sum_{i \in opt_j} \alpha_i \geq f_j(Gr_p(j)) + \sum_{i \in opt_j} (f_j(Gr_p(j) \cup i) - f_j(Gr_p(j)))$$
$$\geq f_j(Gr_p(j)) + \big(f_j(Gr_p(j) \cup opt_j) - f_j(Gr_p(j))\big)$$
$$= f_j(Gr_p(j) \cup opt_j) \geq f_j(opt_j),$$

where the second inequality follows by submodularity and the last inequality by monotonicity. □

The main theorem of this section follows immediately.

Theorem 7.2. *For any $0 < p < 1$, there is a $(\frac{p}{1+p}, \frac{1-p}{2-p})$-competitive algorithm for bi-objective online SWM.*

7.2.3 Hardness of Bi-Objective online SWM

We now prove that Theorem 7.2 is almost tight, by describing a hard instance for bi-objective online SWM. To describe this instance, we define notions of *super nodes* and *super edges*, which capture the hardness of maximizing a submodular function even in the offline setting. Using the properties of super nodes and edges, we construct and analyze a hard example for bi-objective online SWM.

Our construction generalizes that of Kapralov *et al.* ([174]), who prove the upper bound corresponding to the two points $(0.5, 0)$ and $(0, 0.5)$ in the curve shown in Figure 7.1. They use the following result: For any fixed c_0 and ϵ' it is NP-hard to distinguish between the following two cases for offline SWM with n agents and $m = kn$ items. Indeed, this holds even for submodular functions with 'coverage' valuations.

- There is an allocation with value n.
- For any $l \leq c_0$ there is no allocation that allocates kl items and gets a value more than $1 - e^{-l} + \epsilon'$.

Intuitively, in the former case, we can assign k items to each agent and obtain value 1 per agent. In the latter case, even if we assign $2k$ items (however they are split across agents), we can obtain total value at most 0.865. It also follows that there exist 'hard' instances such that there is an optimal solution of value n, but for any $l < 1$, any assigment of ml edges obtains value at most $(1 - e^{-l} + \epsilon')n$.

We now define a *super edge* to be a hard instance of offline SWM as defined above. We refer to the set of agents in a super edge as the agent super node, and the set of items in the super edge as the item super node. If two super edges share a super node, it means that they share the agents / items corresponding to that super node in the same order. If (in expectation) we allocate ml items of a super edge, we say the load of that super edge is l. Similarly, if (in expectation) we allocate ml items to an agent super node, we say the load of that super node is l. Using the definition of super edge and super node, the hardness result of Kapralov *et al.* ([174]) gives us the following lemma:

Lemma 7.2. *Assume $RP \neq NP$ and let ϵ be an arbitrary small constant. If the (expected) load of a randomized polynomial algorithm on an agent super node is l, the expected welfare of all agents is at most $(1 - e^{-l} + \epsilon)n$.*

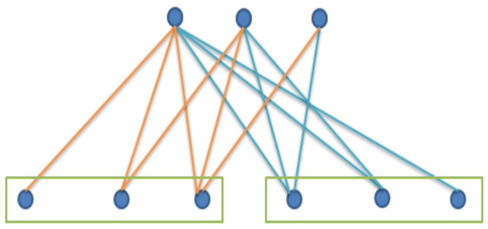


Figure 7.2: Blue edges have weight $(f(.), 0)$ and orange edges have weight $(0, f(.))$.

Now with Lemma 7.2 in hand, we are ready to present an upper bound for bi-objective online SWM.

Theorem 7.3. *Assume $RP \neq NP$. The competitive ratio (α, β) of any algorithm for bi-objective online SWM is upper bounded by the red curve in Figure 7.1. More precisely (assuming w.l.o.g. that $\alpha \geq \beta$), for any $\gamma \in [0, 1]$, there is no algorithm with $\alpha > \frac{0.5+\gamma^2/6}{1+\gamma^2}$ and $\beta > \gamma\alpha$.*

Proof. Consider the following example (see Figure 7.2). There are $2n$ agent super nodes partitioned into two sets $V = \{v_1, v_2, ..., v_n\}$ and $V' = \{v'_1, v'_2, ..., v'_n\}$. The super nodes in V contribute only to the first objective function (i.e. for any agent a in these super nodes, $g_a(\cdot) = 0$.) Similarly, the super nodes in V' contribute only to the second objective function.

There are n item super nodes $U = \{u_1, u_2, .., u_n\}$ where u_i is the i-th item super node that appears online. The first item super node has a super-edge to all agent super nodes. The $(i+)1$-th online vertex is identical to the i-th online super node but we drop one of its super edges to V and one of its edges to V': The two super edges that we drop at each stage are the ones connected to the agent super nodes with the minimum current loads. Without loss of generality, we assume that the super edges corresponding to agent super nodes v_i and v'_i are dropped at stage $i+1$ (that is, the agents in v_i and v'_i have no value for any items in the super node u_{i+1}).

It is easy to see that there is a matching using super edges of size n that covers all vertices in V, and there is another matching of size n that covers all vertices in V'. Thus, the optimal solution on each objective has value n^2.

Let x_j and y_j be an algorithm's final load on agent super nodes v_j and v'_j respectively. Note that after step j, the sum of the loads on all agent super nodes is j. Moreover, by definition, the load of v_1 to v_{j-1} and v'_1 to v'_{j-1} is $\sum_{k=1}^{j-1}(x_k + y_k)$. Thus (since v_j and v'_j are the least loaded remaining super nodes from V and V' respectively), we have:

$$x_j + y_j \le \frac{j - \sum_{k=1}^{j-1}(x_k + y_k)}{n - j + 1}. \tag{7.1}$$

Recall that the welfare from a super node with load x is bounded by $(1 - e^{-x} + \epsilon)n$.

Assume that the algorithm is (α, β)-competitive. By Lemma 7.2 we have the following inequalities:

$$\sum_{j=1}^{n} 1 - e^{-x_j} \ge (\alpha - \epsilon)n \tag{7.2}$$

$$\sum_{j=1}^{n} 1 - e^{-y_j} \ge (\beta - \epsilon)n. \tag{7.3}$$

For simplicity of notation, we drop ϵ from the above inequalities. Without loss of generality (by symmetry), we assume that $\alpha \ge \beta$. For a fixed $\gamma = \frac{\beta}{\alpha} \le 1$, we can

Maximize: α

Subject to:

$$x_j + y_j \leq \frac{j - \sum_{k=1}^{j-1}(x_k + y_k)}{n - j + 1} \quad \forall 1 \leq j \leq n \tag{7.6}$$

$$\sum_{j=1}^{n} \left(e^{-x_j} + \gamma e^{-y_j}\right) \leq (1+\gamma)n - (1+\gamma^2)\alpha n \tag{7.7}$$

Figure 7.3: Configuration Mathematical Program 3 to bound the competitive ratio of bi-objective online SWM.

rewrite Inequality 7.3 as

$$\gamma \sum_{j=1}^{n} 1 - e^{-y_j} \geq \gamma^2 \alpha n. \tag{7.4}$$

By summing up both sides of Inequalities 7.2 and 7.4, and rearranging the terms we have

$$\sum_{j=1}^{n} \left(e^{-x_j} + \gamma e^{-y_j}\right) \leq (1+\gamma)n - (1+\gamma^2)\alpha n. \tag{7.5}$$

Therefore, for any fixed γ, α is upper-bounded by the Mathematical Program 7.3. It remains only to bound the value of this mathematical program for any fixed

γ. Note that

$$\begin{aligned}\frac{j-\sum_{k=1}^{j-1}(x_k+y_k)}{n-j+1} &\geq \frac{j-\sum_{k=1}^{j-2}(x_k+y_k)-\frac{j-1-\sum_{k=1}^{j-2}(x_k+y_k)}{n-j+2}}{n-j+1}\\ &= \frac{(j-1-\sum_{k=1}^{j-2}(x_k+y_k))(1-\frac{1}{n-j+2})+1}{n-j+1}\\ &> \frac{(j-1-\sum_{k=1}^{j-2}(x_k+y_k))(1-\frac{1}{n-j+2})}{n-j+1}\\ &= \frac{(j-1-\sum_{k=1}^{j-2}(x_k+y_k))(\frac{n-j+1}{n-j+2})}{n-j+1} = \\ &\frac{j-1-\sum_{k=1}^{j-2}(x_k+y_k)}{n-j+2}.\end{aligned}$$

This means that the upper-bound enforced by Constraint 7.6 is increasing in j. On the other hand the left term in Constraint 7.7 is a convex function. Thus, for any j and k such that $x_j \leq x_k$, if we increase x_j by some small ϵ and increase x_k by ϵ the constraint remains feasible. These two together say that for any k and j such that $k > j$ and Constraint 7.6 is not tight for j, we can increase x_j by some small ϵ and decrease x_k by ϵ and keep all the constraints feasible. By applying this procedure iteratively we can make Constraint 7.6 tight for all js. Therefore, without loss of generality, we assume that Constraint 7.6 is tight for all j. From this tightness, one conclude that for all $1 \leq j \leq n$ we have

$$x_j + y_j = \sum_{k=1}^{j} \frac{1}{n-k+1} \leq \int_0^j \frac{1}{n-z}dz = ln\frac{n}{n-j}.$$

We rewrite the j term of Constraint 7.7 as

$$e^{-x_j} + \gamma e^{-y_j} = e^{-x_j} + \gamma e^{-(ln\frac{n}{n-j}-x_j)} = e^{-x_j} + \gamma\frac{n-j}{n}e^{x_j}.$$

This maximizes at $-e^{-x_j} + \gamma\frac{n-j}{n}e^{x_j} = 0$, which means $x_j = \frac{-ln(\gamma\frac{n-j}{n})}{2}$ when $\frac{-ln(\gamma\frac{n-j}{n})}{2} \leq ln\frac{n}{n-j}$, and $x_j = ln\frac{n}{n-j}$ otherwise. Therefore, for $\gamma \leq \frac{n-j}{n}$ we have

$$
\begin{aligned}
e^{-x_j} + \gamma e^{-y_j} &\leq e^{ln(\gamma\frac{n-j}{n})/2} + \gamma e^{-(ln\frac{n}{n-j} + ln(\gamma\frac{n-j}{n})/2)} \\
&= \sqrt{\gamma\frac{n-j}{n}} + \gamma\frac{n-j}{n}\sqrt{\frac{1}{\gamma}\frac{n}{n-j}} \\
&= 2\sqrt{\gamma\frac{n-j}{n}},
\end{aligned}
$$

and for $\gamma > \frac{n-j}{n}$ we have

$$
e^{-x_j} + \gamma e^{-y_j} \leq e^{-ln\frac{n}{n-j}} = \frac{n-j}{n} + \gamma.
$$

Therefore, we have

$$
\begin{aligned}
\sum_{j=1}^{n}\left(e^{-x_j} + \gamma e^{-y_j}\right) &\leq \sum_{j=1}^{(1-\gamma)n}(\frac{n-j}{n} + \gamma) + \sum_{j=(1-\gamma)n+1}^{n} 2\sqrt{\gamma\frac{n-j}{n}} \\
&\leq \int_{0}^{(1-\gamma)n}(\frac{n-z}{n} + \gamma)dz + \int_{(1-\gamma)n}^{n} 2\sqrt{\gamma\frac{n-z}{n}}dz \\
&= n\left((0.5 + \gamma - \frac{3}{2}\gamma^2) + (\frac{4}{3}\gamma^2)\right) \\
&= n(0.5 + \gamma - \frac{1}{6}\gamma^2).
\end{aligned}
$$

Applying this to Constraint 7.7 upper-bounds α by

$$
\frac{1 + l - (0.5 + \gamma - \frac{1}{6}\gamma^2)}{1 + \gamma^2} = \frac{0.5 + \frac{1}{6}\gamma^2}{1 + \gamma^2},
$$

and thus, bounds β by

$$
\gamma\frac{0.5 + \frac{1}{6}\gamma^2}{1 + \gamma^2}.
$$

This is the curve in Figure 7.1. □

7.3 Bi-Objective Online Weighted Matching

In this section, we consider two special cases of bi-objective online SWM, each of which generalizes the (single objective) online weighted matching problem (with free disposal). Here, each item i has two weights w_{ij}^f and w_{ij}^g for agent j, and each agent j has (large) capacity C_j. The weights of item i are revealed when it arrives, and the algorithm must allocate it to some agent immediately.

In the *first model*, after the algorithm terminates, and each agent j has received items $\mathcal{S}_j$, it chooses a subset $\mathcal{S}'_j \subseteq \mathcal{S}_j$ of at most C_j items. The total value in the first objective is then $\sum_j \sum_{i \in \mathcal{S}'_j} w_{ij}^f$, and in the second objective $\sum_j \sum_{i \in \mathcal{S}'_j} w_{ij}^g$. Intuitively, each agent must pick a subset of its items, and it gets paid its (additive) value for these items. In the (single-objective) case where each agent can only be *allocated* C_j items, this is the online weighted b-matching problem, where vertices are arriving online, and we have edge weights in the bipartite (item, agent) graph. This problem is completely intractable in the online setting, while the free disposal variant [158] in which additional items can be assigned, but at most C_j items count towards the objective, is of theoretical and practical interest.

In the *second model*, after the algorithm terminates and agent j has received items $\mathcal{S}_j$, it chooses two (not necessarily disjoint) subsets $S_j'^f$ and $\mathcal{S}_j'^g$; items in $\mathcal{S}_j'^f$ are counted towards the first objective, and those in $\mathcal{S}_j'^g$ are counted towards the second objective.

Theorem 7.4. *For any (α, β) such that $\alpha + \beta \leq 1 - \frac{1}{e}$, there is an (α, β)-competitive algorithm for the first model of the bi-objective online weighted matching. For any*

```
**Exponential Weight Algorithm.**

Set $\beta_j$ to 0 for each agent $j$.

Upon arrival of each item $i$:

  1. If there is agent $j$ with $w_{ij} - \beta_j > 0$
      (a) Let $j$ be the agent that maximizes $w_{ij} - \beta_j$
      (b) Assign $i$ to $j$ .
      (c) Set $\alpha_i$ to $w_{ij} - \beta_j$.
      (d) Let $w_1, w_2, \ldots, w_{C_j}$ be the weights of the $C_j$ highest weight items,
          matched to $j$ in a non-increasing order.
      (e) Set $\beta_j$ to $\frac{\sum_{j=1}^{C_j} w_j \left(1+\frac{1}{C_j}\right)^{j-1}}{C_j\left((1+1/C_j)^{C_j}-1\right)}$.
  2. Else: Leave $i$ unassigned.
```

Figure 7.4: Exponential weight algorithm for online matching with free disposal.

constant $\epsilon > 0$, there is no such algorithm when $\alpha + \beta > 1 - \frac{1}{e} + \epsilon$.

Proof. The following algorithm yields the positive result: With probability p, run the exponential weight algorithm for the first objective (for all items), and with probability $1 - p$ run the exponential weight algorithm for the second objective for all items. Feldman *et al.* ([158]) show that the exponential weight algorithm is $1 - \frac{1}{e}$ competitive as $\min_j\{C_j\}$ tends to infinity, which clearly means that this algorithm is $(p(1 - \frac{1}{e}), (1 - p)(1 - \frac{1}{e}))$-competitive. Choosing $p \in [0, 1]$ gives us any desired (α, β) s.t. $\alpha + \beta \le 1 - \frac{1}{e}$.

It remains only to prove the hardness result: For simplicity of explanation, in the hard instance we describe below, we set the capacity of each agent to one and instead allow the online algorithm to select a fractional assignment. Indeed, for any number C one can set the capacity of each agent to C, replace each item with C copies, and interpret the integral solution of the new instance as a fractional solution of the original instance.

We have $2n$ agents and $2n$ items (see figure 7.5). The first item is connected to all of the agents. The weights of edges are $(1, 0)$ and $(0, 1)$ alternatively i.e. weight of edges to the odd-index agents are $(1, 0)$ and edge to the even-index agents are $(0, 1)$. The second item is also connected to all of the agents. However, the weights of the edges from the second item to the odd-index agents are $(0, 1)$ and weights of edges to the even-index vertices are $(1, 0)$.

The edge set of the $2i + 1$-th and $2i + 2$-th items are the same as $2i - 1$-th and $2i$-th items, respectively, except that we drop the edges to one even-index agent and one odd-index agent. Later, we explain how to choose these two dropped agents.

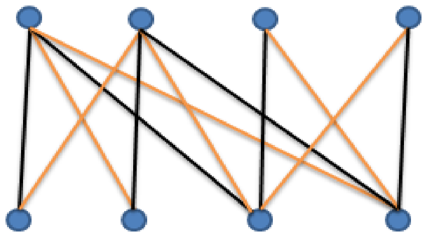


Figure 7.5: Black edges have weight $(1, 0)$ and orange edges have weight $(0, 1)$.

Without loss of generality, we assume that the offline vertices that we drop at step i are $2i + 1$ and $2i + 2$.

First, we show that there is a matching with weight $2n$ using the first objective function, and also, there is a matching with weight $2n$ using the second objective function. Later, we show that no online algorithm can guarantee the sum of both objective functions to be more than $2(1 - \frac{1}{e})n$. Together, these claims prove the theorem.

To see that there exist good optimal matchings, consider the matching that, for all $1 \leq i \leq n$, matches item $2i - 1$ to agent $2i - 1$ and item $2i$ to agent $2i$. This matching has weight $2n$ based on the first objective. Moreover, the matching that, for all $1 \leq i \leq n$, matches item $2i - 1$ to agent $2i$ and item $2i$ to agent $2i - 1$ has weight $2n$ based on the second objective.

Next we upper bound the sum of the two objective function values for any online algorithm. Consider an arbitrary online algorithm. Let y_j^i be the probability that vertex j is matched at step i, and let y_j be the probability that offline vertex j is matched at the end of the algorithm. Remark that we have $y_{2j}^{2j} = y_{2j}$ and $y_{2j-1}^{2j} = y_{2j}$. Lets define $y_j^* = y_{2j-1} + y_{2j}$. Now, it is easy to explain which vertices we drop at step $2j$. Indeed, at step $2j$,we drop an even-index and an odd-index vertex to minimize y_j^*.

Note that the sum of both weights on each edge is exactly 1. Thus, the sum the value of both objective functions is exactly $\sum_{j=1}^{n} y_j^*$. In the rest, we show that $\sum_{j=1}^{n} y_j^* \leq 2(1 - \frac{1}{e} + \epsilon)n$. This immediately says that the sum of the competitive ratios do not exceed $1 - \frac{1}{e} + \epsilon$, as desired.

Note that, after $2j$ step the number of matched vertices is at most $2j$. On the other hand, the expected number of matched vertices 1 to $2j-2$ is exactly $\sum_{k=1}^{j-1} y_k^*$. Thus, we have

$$y_j^* \leq \frac{2j - \sum_{k=1}^{j-1} y_k^*}{n-j+1}.$$

Therefore, the expected number of matched vertices is bounded by the solution of Linear Program 7.6. Consider an optimum solution to this LP and assume that one of the constraints in Line 7.8 is not tight. Assume that for some j, the upper bound of y_j^* is not tight and for some $k \geq j$, $y_k^* \neq 0$. Then, for some small number ϵ we can increase y_j^* by ϵ and decrease y_k^* by ϵ while keeping all constraints feasible. Indeed, we can repeat this procedure until the upper bound of y_n^* is not tight. Then, we can increase y_n^* by some small number ϵ while keeping all constraints feasible. This increases the value of the objective function, which is a contradiction. Thus, in the optimum solution all the constraints are tight. It is easy to see that this solution corresponds to the algorithm that match each vertex to all its neighbors uniformly and make the objective function $2(1 - \frac{1}{e})n$.

$\square$

$$\begin{aligned} &\text{Maximize:} && \sum_{j=1}^{n} y_j^* && \\ &\text{Subject to:} && y_j^* \le \frac{2j - \sum_{k=1}^{j-1} y_k^*}{n-j+1} && \forall 1 \le j \le n \qquad (7.8) \\ & && 0 \le y_j^* \le 2 && \forall 1 \le j \le n \end{aligned}$$

Figure 7.6: Configuration LP to bound the competitive ratio of bi-objective online weighted matching, in the first model.

Having given matching upper and lower bounds for the first model, we now consider the second model, where if we assign a set $\mathcal{S}_j$ of items / edges to an agent j we can select two subsets $\mathcal{S}_j'^f, \mathcal{S}_j'^g \subseteq \mathcal{S}_j$ and use them for the first and second objective functions respectively.

Theorem 7.5. *There is a* $(p(1-\frac{1}{e^{1/p}}), (1-p)(1-\frac{1}{e^{1/(1-p)}}))$*-competitive algorithm for the bi-objective online weighted matching problem in the second model as* $\min_j\{C_j\}$ *tends to infinity.*

Proof. We run 2 copies of the exponential weight algorithm 'in parallel': For each item, with probability p, pass it to the exponential weight algorithm based on the first weight and with probability $1-p$, run the exponential weight algorithm based on the second weight. Note that, here, for each weight, we can use the whole capacity C_j of each offline vertex. Korula *et al.* ([164]) show that the expected weight of the output assignment for a single objective when each item is passed with probability p is at least $p(1-\frac{1}{e^{1/p}})$ that of the optimum. The same argument holds for the second

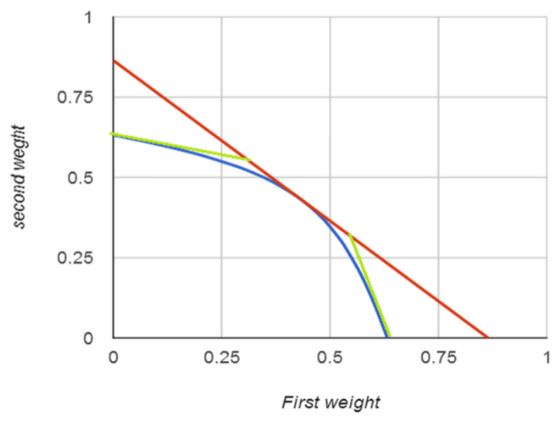


Figure 7.7: The blue curve is the competitive ratio of our algorithm, while the red line and the green curves are the upper bounds on the competitive ratio of any algorithm.

objective, which gives us a competitive ratio of $(1-p)(1-\frac{1}{e^{1/(1-p)}})$ and completes the proof. $\square$

Theorem 7.6. *The competitive ratio of any algorithm for bi-objective online weighted matching in the second model is upper bounded by the curve in figure 7.7.*

Proof. In fact, this problem is a generalized version of bi-objective online matching with one weight function and one cardinality function. Thus, the same hardness results from [164] holds for the bi-objective online weighted matching as well. In addition to this, given an (α, β) hardness, the symmetry of the objectives immediately gives us a (β, α) hardness as well. This fact, together with the hardness results provided in [164] gives us the curve in Figure 7.7. $\square$

7.4 Bi-Objective Online Budgeted Allocation

In this section, we consider the bi-objective online allocation problem where one of the objectives is a budgeted allocation problem and the other objective function is weighted matching. Here, each item i has a weight w_{ij} and a bid b_{ij} for agent j. Each agent j has a capacity C_j and a budget B_j. If an agent is allocated items $\mathcal{S}_j$, for the first objective (weighted matching), it chooses a subset $\mathcal{S}'_j$ of at most C_j items; its score is $\sum_{i \in \mathcal{S}'_j} w_{ij}$. For the second objective, its score is $\min\{\sum_{i \in \mathcal{S}_j} b_{ij}, B_j\}$. Note that in the second objective, the agent does not need to choose a subset; it obtains the sum of the bids of all items assigned to it, capped at its budget B_j.

Clearly, if we set all bids b_{ij} to 1, the goal of the budgeted allocation part will be maximizing the cardinality. Thus, this is a clear generalization of the bi-objective online allocation to maximized weight and cardinality, and the same hardness results hold here.

As is standard, throughout this section we assume that the bid of each agent for each item is vanishingly small compared to the budget of each bidder. Interestingly, again here, we provide a $(p(1 - \frac{1}{e^{1/p}}), (1-p)(1 - \frac{1}{e^{1/(1-p)}}))$-competitive algorithm, which is almost tight. At the end, as a corollary of our results, we provide a a $(p(1 - \frac{1}{e^{1/p}}), (1-p)(1 - \frac{1}{e^{1/(1-p)}}))$-competitive algorithm, for the case that both objectives are budgeted allocation problems (with separate budgets).

Our algorithm here is roughly the same as for two weight objectives. For each item, with probability $1-p$, we pass it to the Exponential Weight algorithm for matching, and allocate it based on its weight. With the remaining probability

p, we assign the algorithm based on its bids and count it towards the Budgeted Allocation objective. However, the algorithm we use for Budgeted Allocation is slightly different: We *virtually* run the Balance algorithm of Mehta *et al.* ([133]) for Budgeted Allocation (Fig. 7.8), as though we were assigning *all* items (not just those passed to this algorithm), but with each item's bids scaled down by a factor of p. For those p fraction of items to be assigned by the Budgeted Allocation algorithm, assign them according to the recommendation of the virtual Balance algorithm.

Theorem 7.5 from the previous section shows that our algorithm is $(1-p)(1-\frac{1}{e^{1/(1-p)}})$-competitive against the optimum weighted matching objective. Thus, in the rest of this section, we only need to show that this algorithm is $p(1-\frac{1}{e^{1/p}})$-competitive against the optimum Budgeted Allocation solution. First, using a primal dual approach, we show that the outcome of the virtual Balance algorithm (that runs on p fraction of the value of each item) is $p(1-\frac{1}{e^{1/p}})$ against the optimum with the actual weights. Then, using the Hoeffding inequality, we show that the expected value of our allocation for the budgeted allocation objective is fairly close to the virtual algorithm's value, i.e. the difference between the competitive ratio of our allocation and the virtual allocation is $o(1)$.

Lemma 7.3. *When $\frac{\max_{i,j} b_{ij}}{B_j} \to 0$, the virtual balance algorithm that runs on p fraction of the value of each bid is $p(1-\frac{1}{e^{1/p}})$-competitive against the optimum with the actual values.*

Proof. In order to prove this lemma, we show that Algorithm 7.8 provides feasible

```
Virtual Balance algorithm on p fraction of values.

Set $\beta_j$ and $y_j$ to 0 for each agent $j$.

Upon arrival of each item $i$:

  1. If $i$ has a neighbor with $b_{ij}(1-\beta_j) > 0$
     (a) Let $j$ be the agent that maximizes $b_{ij}(1-\beta_j)$
     (b) Assign $i$ to $j$ i.e. set $x_{ij}$ to 1.
     (c) Set $\alpha_i$ to $b_{ij}(1-\beta_j)$.
     (d) Increase $y_j$ by $\frac{b_{ij}}{B_j}$
     (e) Increase $\beta_j$ by $\frac{e^{y_j-1/p}}{1-e^{-1/p}}\frac{b_{ij}}{B_j}$
  2. Else: Leave $i$ unassigned.
```

Figure 7.8: Maintaining solution to primal and dual LPs.

$$\begin{aligned}
\text{Maximize:} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} p b_{ij} x_{ij} & \\
\text{Subject to:} \quad & \sum_{j=1}^{m} x_{ij} \le 1 & \forall 1 \le i \le n \\
& \sum_{i=1}^{n} p b_{ij} x_{ij} \le B_j & \forall 1 \le j \le m
\end{aligned}$$

Figure 7.9: Primal LP using p fraction of weights

$$\text{Minimize:} \quad \sum_{j=1}^{m} B_j\beta_j + \sum_{i=1}^{n} \alpha_i$$

Subject to:

$$b_{ij}\beta_j + \alpha_i \geq b_{ij} \qquad \forall (i,j) \in E$$

$$\beta_j, \alpha_i \geq 0 \qquad \forall 1 \leq i \leq n, 1 \leq j \leq m$$

Figure 7.10: Dual LP for Budgeted Allocation using all of the budgets

solution to the LPs 7.9 and 7.10 above. In addition, we show that whenever we increase the dual LP 7.10 by $\frac{1}{1-e^{-1/p}}$ we increase the primal LP 7.9 by pb_{ij}. These two together say that the Algorithm 7.8 is $p(1-e^{-1/p})$-competitive as desired.

Consider that for each bidder j we have

$$\begin{aligned}
\beta_j &= \sum_{j:x_{ij}=1} \frac{e^{y_j-1/p}}{1-e^{-1/p}} \frac{b_{ij}}{B_j} \\
&= \int_0^{y_j} \frac{e^{y_j-1/p}}{1-e^{-1/p}} \delta y_j && \text{Since } \frac{b_{ij}}{B_j} \to 0 \\
&= \frac{e^{y_j-1/p} - e^{-1/p}}{1-e^{-1/p}}.
\end{aligned}$$

Let j be the bidder that maximize $\frac{1-e^{y_j-1/p}}{1-e^{-1/p}} b_{ij}$ (and is matched to i) and let k be an arbitrary other neighbor of i. We have

$$\begin{aligned}
\alpha_i + \beta_k b_{i,k} &= \frac{1-e^{y_j-1/p}}{1-e^{-1/p}} b_{ij} + \frac{e^{y_k-1/p} - e^{-1/p}}{1-e^{-1/p}} b_{i,k} \\
&\leq \frac{1-e^{y_k-1/p}}{1-e^{-1/p}} b_{i,k} + \frac{e^{y_k-1/p} - e^{-1/p}}{1-e^{-1/p}} b_{i,k} \\
&= \frac{1-e^{-1/p}}{1-e^{-1/p}} b_{i,k} = b_{i,k},
\end{aligned}$$

where the inequality is by the definition of j. This means that the dual LP 7.10 is feasible.

On the other hand, for each bidder j (by ignoring one bid if $y_j > 1/p$), we have $\frac{1-e^{y_j-1/p}}{1-e^{-1/p}} b_{ij} \geq 0$ which means that $y_j \leq 1/p$. By replacing y_j with $\sum_{i:x_{ij}=1} \frac{b_{ij}}{B_j}$ and rearranging the terms we have, $\sum_i p x_{ij} b_{ij} \leq B_j$. This means that primal LP 7.9 is feasible.

Finally, consider that each time we match an edge (i,j), we increase the objective of the primal LP by pb_{ij}. On the other hand, we increase the objective function of the dual LP by $\frac{1-e^{y_j-1/p}}{1-e^{-1/p}} b_{ij} + \frac{e^{y_j-1/p}}{1-e^{-1/p}} \frac{b_{ij}}{B_j} B_j = \frac{1}{1-e^{-1/p}} b_{ij}$. This means that our primal solution is at least $p(1-e^{-1/p})$ the optimum of the dual solution and completes the proof. □

Lemma 7.4 (Hoeffding inequality). *Let $x_1, x_2, \ldots, x_t$ be t random variable such that, for all $1 \leq s \leq t$ we have $0 \leq x_s \leq b_s$ and let $X = \sum_{s=1}^{t} x_s$. We have*

$$Pr(X - E[X] \geq \lambda) \leq \exp\left(-\frac{2\lambda^2}{\sum_s b_s^2}\right).$$

Assume that for all $1 \leq s \leq t$, with probability p, $x_s = b_s$ and with probability $1-p$, $x_s = 0$. Moreover, let $\lambda = \epsilon B$, for some $B \geq E[X]$. One can rewrite the

Hoeffding inequality as follow.

$$\begin{aligned}
Pr(X - E[X] \geq \epsilon B) &\leq \exp\left(-\frac{2\epsilon^2 B^2}{\sum_s b_s^2}\right) \\
&\leq \exp\left(-\frac{2\epsilon^2 B^2}{\max_s(b_s)\sum_s b_s}\right) \\
&= \exp\left(-\frac{2\epsilon^2 B^2}{\max_s(b_s)\frac{1}{p}E[X]}\right) \\
&\leq \exp\left(-\frac{2p\epsilon^2 B}{\max_s(b_s)}\right)
\end{aligned}$$

Lemma 7.5. *For any constant p, assuming $\frac{\max_{i,j} b_{ij}}{B_j} \to 0$, the budgeted allocation value of our algorithm tends to the value of balance with p fraction of each bid.*

Proof. We use the version of the Hoeffding inequality immediately above to bound the budgeted allocation value of our algorithm.

Let $L = \max_{i,j} b_{ij}$, let j be an arbitrary bidder and let $b_1^j, b_2^j, \ldots, b_t^j$ be the set of bids of online items assigned to the bidder j by the virtual balance. Let x_s^j be a random variable which is b_s^j if we assign b_s^j in our budgeted allocation algorithm and 0 otherwise. Let $X^j = \sum_s x_s^j$. Note that, the budget that the virtual balance gets from bidder j is $\sum_s pb_s^j = E[X^j] \leq B_j$, and the expected budget that our algorithm gets from this bidder is $E[\min(X^j, B_j)]$.

Remark that $E[X^j] = E[\min(X^j, B_j)] + E[\max(0, X^j - B_j)]$. Moreover, for any positive number ϵ, we have $E[\max(0, X^j - B_j)] \leq \epsilon E[X^j] + E[\max(0, X^j - B_j - \epsilon E[X^j])]$. In the rest we show that for any arbitrary small positive constant ϵ we have $\frac{E[\max(0, X^j - B_j - \epsilon E[X^j])]}{E[X^j]} \to 0$, which immediately gives us $\frac{E[\min(X^j, B_j)]}{E[X^j]} \to 1$, as desired.

The probability that X^j is greater than $B_j + \epsilon E[X^j]$ is

$$\begin{aligned}
Pr(X^j - B_j \geq \epsilon E[X^j]) &= Pr(X^j - E[X^j] \geq \epsilon E[X^j] + B_j - E[X^j]) \\
&= Pr(X^j - E[X^j] \geq (1-\epsilon)(B_j - E[X^j]) + \epsilon B_j) \\
&\leq Pr(X^j - E[X^j] \geq \epsilon B_j) \\
&\leq \exp\left(-\frac{2p\epsilon^2 B_j}{L}\right),
\end{aligned}$$

where the last inequality is by Hoeffding inequality. Note that X^j is upper-bounded by $\sum_s b_s^j = \frac{1}{p}\sum_s p b_s^j = \frac{1}{p}E[X^j]$. Thus, we have

$$\begin{aligned}
E[\max(0, X^j - B_j - \epsilon E[X^j])] &= Pr(X^j - B_j \geq \epsilon E[X^j]) \times (X^j - B_j - \epsilon E[X^j]) \\
&\leq Pr(X^j - B_j \geq \epsilon E[X^j]) \times (X^j) \\
&\leq Pr(X^j - B_j \geq \epsilon E[X^j]) \times \frac{1}{p}E[X^j] \\
&\leq \exp\left(-\frac{2p\epsilon^2 B_i}{L}\right)\frac{1}{p}E[X^j].
\end{aligned}$$

Note that $\exp\left(-\frac{2p\epsilon^2 B_i}{L}\right)\frac{1}{p} \to 0$ when $\frac{L}{B_i} \to 0$. Therefore, for any small positive constant ϵ, $\frac{E[\max(0, X^j - B_j - \epsilon E[X^j])]}{E[X^j]} \to 0$, which completes the proof. □

The following lemma is an immediate result of combining Lemma 7.3 and Lemma 7.5.

Lemma 7.6. *For any constant p, assuming $\frac{max_{i,j} b_{ij}}{B_j} \to 0$, our algorithm is $p(1 - \frac{1}{e^{1/p}})$-competitive against the optimum budgeted allocation solution.*

Lemma 7.6 immediately gives us the following theorem.

Theorem 7.7. *For any constant p, assuming $\frac{max_{i,j} b_{ij}}{B_j} \to 0$, there is a $(p(1 - \frac{1}{e^{1/p}}), (1-p)(1 - \frac{1}{e^{1/(1-p)}}))$-competitive algorithm for the bi-objective online allocation with two budgeted allocation objectives.*

Moreover, if we pass each item to the exponential weight algorithm with probability p the expected size of the output matching is at least $p(1 - \frac{1}{e^{1/p}})$ that of the optimum [164]. Together with Lemma 7.6, this gives us the following theorem.

Theorem 7.8. *For any constant p, assuming $\frac{max_{i,j} b_{ij}}{B_j} \to 0$, there is a $(p(1 - \frac{1}{e^{1/p}}), (1-p)(1 - \frac{1}{e^{1/(1-p)}}))$-competitive algorithm for the bi-objective online allocation with a budgeted allocation objective and a weighted matching objective.*

7.5 Conclusions

In this chapter, we gave the first algorithms for several bi-objective online allocation problems. Though these are nearly tight, it would be interesting to consider other models for bi-objective online allocation, special cases where one may be able to go beyond our hardness results, and other objectives such as fairness to agents.

Chapter 8: Online Allocation with Traffic Spikes

8.1 Introduction

In the past decade, online budgeted allocation problems have been studied extensively due to their important applications in Internet Advertising. In such problems, we are given a bipartite graph $G = (X, Y, E)$ with a set of fixed nodes (also known as *agents*, or *advertisers*) Y, a set of online nodes (corresponding to *items* or *impressions*) X, and a set E of edges between them. Each agent / fixed node $j \in Y$ is associated with a total weighted capacity (or budget) c_j; in the context of Internet advertising, each agent corresponds to an advertiser with a fixed budget to spend on showing their ad to users. The items / online nodes $i \in X$ arrive one at a time, along with their incident edges $(i, j) \in E(G)$ and the weights $w_{i,j}$ on these edges. These online nodes correspond to search queries, page-views, or in general, impressions of ads by users. Upon the arrival of an item $i \in X$, the algorithm can assign i to at most one agent $j \in Y$ where $(i, j) \in E(G)$ and the total weight of nodes assigned to j does not exceed c_j. The goal is to maximize the total weight of the allocation.

This problem is known as the *Budgeted Allocation* or *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 [130, 131, 133] (called the *large-degree* assumption). Many variants of this problem such as *the display ads* problem [158] have been studied, and techniques to solve the budgeted allocation problem have been generalized to solve those problems.

Traditionally, results have been developed for a worst-case arrival model in which the algorithm does not have any prior on the arrival model of online nodes. Under this most basic online model, known as the *adversarial model*, the online algorithm does not know anything about the items or $E(G)$ beforehand. In this model, the seminal result of Karp, Vazirani and Vazirani [183] 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 $(i, j) \in E(G)$. For weighted graphs, Mehta et al. [130, 133] presented the first $1 - \frac{1}{e}$-approximation algorithm to maximize the total weight of the allocation for the AdWords problem.

In practical settings motivated by placement of Internet ads, the incoming traffic of page-views may be predicted with a reasonable precision using a vast amount of historical data. Motivated by this ability to forecast traffic patterns, various stochastic online arrival models have been introduced. Such models include (i) the i.i.d. stochastic arrival model in which there is a (known or unknown) distribution over the types of items, and each item that arrives is drawn i.i.d. from this distribution [154, 175], or (ii) the random order model [131, 152, 153], which makes the weaker assumption that individual items and edge weights can be selected by an adversary, but that they arrive in a random order. Several techniques have been developed to design asymptotically optimal online allocation algorithms for these stochastic arrival models (For example, these algorithms include a set of dual-based algorithms [131, 152], and a set of primal-based algorithms discussed later).

These algorithms for the stochastic models are useful mainly if the incoming traffic of items (i.e. online impressions) can be predicted with high precision. In other words, such algorithms tend to rely heavily on a precise forecast of the online traffic patterns (or if the forecast is not explicitly provided in advance, that the pattern 'learnt' by the algorithm is accurate), and hence these algorithms may not react quickly to sudden changes in traffic. In fact, the slow reaction to such traffic spikes imposes a serious limitation in the real-world use of stochastic algorithms for online advertising, and more generally, this is a common issue in applying stochastic optimization techniques to online resource allocation problems (see e.g., [184]). To the best of our knowledge, no large Internet advertising systems deploy such stochastic allocation algorithms 'as-is' without modifications to deal with situations where the forecasts are inaccurate. Various techniques such as robust or control-based stochastic optimization have been described in the literature [170, 184–186] to deal with this shortcoming, but they do not provide theoretical guarantees when the input is near-adversarial.

One recent theoretical result in this direction is the simultaneous adversarial and stochastic framework [187]. The main question of this recent work is whether there exists an algorithm which simultaneously achieves optimal approximation ratios in both the adversarial and random-order settings. More specifically, does there exist an algorithm achieving a $1-\epsilon$ approximation for the random-order model, and at the same time achieving a $1-\frac{1}{e}$-approximation for the adversarial model? [187] showed that the answer to this question is positive for unweighted bipartite graphs, but it is negative for the general budgeted allocation problem. Further, they show that the best $1 - \frac{1}{e}$-competitive algorithm for the adversarial model achieves a 0.76-approximation in the random-order model. Though this shows that the adversarial algorithm has an improved competitive ratio in stochastic settings, it does not use forecast information explicitly, and hence it can be quite far from optimal even when the forecast is perfectly accurate. Moreover, the simultaneous approximation framework is still trying to design an algorithm that is guaranteed to work well in extreme situations (where the input follows the forecast perfectly, or is completely adversarial). What if the forecast is mostly, but not entirely accurate? For instance, suppose traffic to a website largely follows the prediction, but there is a sudden spike due to a breaking news event? Treating this as entirely adversarial input is too pessimistic.

Our Model and Results In this chapter, we propose a model of online stochastic budgeted allocation with traffic spikes, referred to as *Robust Budgeted Allocation*, that goes beyond the worst-case analysis in the adversarial model, and develop algorithms that explicitly use the stochastic information available for arrival pattern. In our model, in addition to the stochastic input for which we have good forecasting, an unknown number of impressions arrive that are adversarially chosen. This model is motivated by the patterns of traffic spikes in online advertising in which part of the incoming traffic of users may be the result of a new event that we did not predict, corresponding to a new traffic pattern. We design an algorithm that adaptively checks if the traffic forecast is accurate, and reacts to flaws in traffic forecasting due to traffic spikes. We measure the accuracy of the forecast in terms of a parameter λ which, roughly speaking, captures the fraction of the value of an optimal solution that can be obtained from the stochastic input (as opposed to the adversarially chosen impressions). In general, the competitive ratio of the algorithm will naturally increase with λ. Furthermore, we accompany our results with a set of hardness results showing that our provable approximation guarantees are not far from the optimal achievable bounds. Interestingly, our techniques *also* result in new approaches for the simultaneous approximation framework of [187]; though the models are slightly different (i.i.d. vs. random order, as well as the fact that we require a possibly inaccurate forecast of traffic), our algorithm gives improved performance for the weighted case under uncertain input compared to what was achieved in that paper. Section 8.2 describes the model precisely, allowing us to formally state our results.

Our technique is based on defining a notion of ϵ-closeness in distributions, and then understanding the behaviour of an online algorithm over sequences that are ϵ-close to a given distribution. Most notably, we can show how to modify any online stochastic algorithm to work for online adversarial input sequences that are ϵ-close to a known distribution. This technique is summarized in the next section. We then combine such a modified stochastic algorithm with an adversarial algorithm to guarantee robustness. Converting this idea to provable algorithms for the robust online allocation problem requires applying several combinatorial lemmas and proving invariants that can be converted to a factor-revealing mathematical program, which can then be analyzed numerically and analytically to prove desirable competitive ratios.

Other Related Work **Online Stochastic Allocation.** Two general techniques have been applied to get improved approximation algorithms for online stochastic allocation problems: *primal-based* and *dual-based* techniques. The dual-based technique is based on solving a dual linear program on a sample instance, and using this dual solution in the online decisions. This method was pioneered by Devanur and Hayes [131] for the AdWords problem and extended to more general problems [152–154]. It gives a $1-\epsilon$-approximations for the random order model if the number of items m is known to the algorithm in advance, and $\frac{opt}{w_{ij}} \geq O(\frac{m \log n}{\epsilon^3})$, where $n := |Y|$ is the number of agents. The primal-based technique is based on solving an offline primal instance, and applying this solution in an online manner. This method applies the idea of power-of-two choices, and gives improved approximation algorithms for the iid model with known distributions. This technique was initiated by [175] for the online (unweighted) matching problem and has been improved [176, 177, 188, 189]. All the above algorithms heavily rely on an accurate forecast of the traffic. An alternative technique that has been applied to online stochastic allocation problems is based on optimizing a potential function at each stage of the algorithm [178, 190]. This technique has been analyzed and proved to produce asymptotically optimal results under the i.i.d. model with unknown distributions. Although this technique does not rely on the accurate predictions as much, it does not combine stochastic and adversarial models, and the analysis techniques used are not applicable to our robust online allocation model. 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 [155,156]. 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 i.i.d. draws from a known distribution) [176]. All the above results rely on stochastic assumptions and apply only to the random-order or the iid stochastic models.

Robust stochastic optimization. Dealing with traffic spikes and inaccuracy in forecasting the traffic patterns is a central issue in operations research and stochastic optimization. Methods including robust or control-based stochastic optimization [170, 184–186] have been proposed. These techniques either try to deal with a larger family of stochastic models at once [184–186], try to handle a large class of demand matrices at the same time [184, 191, 192], or aim to design asymptotically optimal algorithms that react more adaptively to traffic spikes [170]. These methods have been applied in particular for traffic engineering [184] and inter-domain routing [191, 192]. Although dealing with similar issues, our approach and results are quite different from the approaches taken in these papers. For example, none of these previous models give theoretical guarantees in the adversarial model while preserving an improved approximation ratio for the stochastic model. Finally, an interesting related model for combining stochastic and online solutions for the Adwords problem is considered in [193], however their approach does not give an improved approximation algorithm for the i.i.d. model.

8.2 Preliminaries and Techniques

8.2.1 Model

Let $\mathcal{I}$ denote a set of item 'types'; in the Internet advertising applications, these represent queries / ad impressions with different properties that are relevant to advertiser targeting and bidding. A *forecast* $F = (D, f)$ has two components: A distribution D over $\mathcal{I}$, together with a number f; this is interpreted as a prediction that f items will arrive, each of which is drawn independently from D.

In the *Stochastic Budgeted Allocation problem*, the input known to the algorithm in advance is a forecast $F = (D, f)$, and a set of agents Y, with a capacity c_j for agent j. A sequence of f items is drawn from the distribution D and sent to the algorithm one at a time; the algorithm must allocate these items as they appear online. The total weight of items allocated to agent j must not exceed c_j, and the objective is to maximize the weight of the allocation. As discussed above, there has been considerable work on near-optimal algorithms for Stochastic Budgeted Allocation [131, 152–154, 194].

In this chapter, we define the new *Robust Budgeted Allocation problem*, for which our input model is the following: The adversary can create in advance an arbitrary forecast $F = (D, f)$, and a collection of agents Y. Further, at each time step, the adversary can either create a new arbitrary item (together with its incident edges and weights) and send it to the algorithm, or choose to send an item drawn from D. After at least f items have been drawn from D, the adversary can either send additional (arbitrary) items, or choose to terminate the input. The online algorithm knows in advance only the forecast F and the agents Y, and so it knows that it will receive f items corresponding to i.i.d. draws from D; it does not know anything about the items created by the adversary, where in the sequence they arrive, or the total number m of items that will arrive. As usual, the competitive ratio of the algorithm is measured by the worst-case ratio (over all inputs) of the value of its allocation to the value of the optimal allocation on the sequence that arrives.

With the preceding description of the model, no algorithm can have a competitive ratio better than $1-1/e$, for the simple reason that we could set $f = 0$, allowing the adversary to control the entire input. (Or even for larger f, the edge weights for the adversarial items could be considerably larger than the weights for the forecast items in $\mathcal{I}$.) We have not quantified the accuracy of the forecast, or meaningfully limited the power of the adversary. Our goal is to design algorithms with a competitive ratio that improves with the accuracy of the forecast. We quantify this accuracy as follows:

Definition 8.1. *For an instance I of the Robust Budgeted Allocation problem with forecast (D, f), let $S(I)$ denote the set of f 'stochastic' items drawn from distribution D. Let $A(I)$ denote the set of $n - f$ 'adversarial' items. When I is clear from context, we simply use S and A to denote the stochastic and adversarial items respectively. We mildly abuse notation and, when clear from context, also use I to refer to the sequence of items in an instance. For a solution Sol to an instance I, let*

$Val_S(Sol)$ denote the value obtained by Sol from allocating the items of S to agents, and $Val_A(Sol)$ denote the value obtained by Sol from allocating the items of A.

Definition 8.2. *An instance I of the Robust Budgeted Allocation problem is said to be λ-stochastic if $\lambda = \max_{Sol \in OPT(I)} \{ \frac{E[Val_S(Sol)]}{E[Val_S(Sol) + Val_A(Sol)]} \}$, where $OPT(I)$ is the set of all optimal solutions of I.*

Note that when the forecast is completely accurate (there are no adversarial items), the instance is 1-stochastic, and when $f = 0$ (all items are adversarial), the input is 0-stochastic. Though λ is unknown to the algorithm, our goal is to design algorithms which, when restricted to λ-stochastic instances, have good competitive ratio (ideally, increasing in λ).

8.2.2 Algorithms for Working with Forecasts

In this section, we consider how to solve the Stochastic Budgeted Allocation problem. Similar problems have been applied before (see e.g. [194]), but we describe a specific approach below that will be a useful tool for the Robust Budgeted Allocation problem. Further, our argument implies that this approach performs well even for an *adversarial* input sequence if it is sufficiently 'close' to the forecast.

Roughly speaking, given a forecast $F = (D, f)$, if the number of items f is sufficiently large, then we can work with an 'expected instance'. In the expected instance, the set of items is created by assuming each type $t \in \mathcal{I}$ arrives in proportion to $P_{D=t}$. We then run any (offline) algorithm *Alg* on the expected instance; when an item arrives in the real online instance, we assign it according to the allocation given by *Alg* on the expected instance. If the number of items f is sufficiently large, then only a small error is induced because we assign according to the expected instance instead of the random realization of the forecast.

We begin by defining the notion of a sequence being *ϵ-close* to a distribution. Indeed, we show that with high probability a 'long' sequence of draws from a distribution is ϵ-close to that distribution, where ϵ is an arbitrary small constant. We next prove that if we ignore inputs which are not ϵ-close to the input distribution, we lose only $\frac{1}{m}$ in the competitive ratio. Finally, we show that we can modify any online stochastic algorithm, or more strongly any offline algorithm for Budgeted Allocation to work for online adversarial input sequences which are guaranteed to be ϵ-close to a known distribution. Interestingly, this reduction loses only 4ϵ on the competitive ratio.

Definition 8.3. *Let $S = \langle s_1, s_2, ..., s_m \rangle$ be a sequence of items and let D be a distribution over a set of item types $\mathcal{I}$. For a type $t \in \mathcal{I}$, let $P_{D=t}$ be the probability that a draw from D is t. We say S is ϵ-close to distribution D, if for any continuous sub-sequence $S_{i,k} = s_i, s_{i+1}, \ldots, s_k \subset S$ and any type t, the number of items of type t in $S_{i,k}$ is within the range $(k - i + 1 \pm \epsilon m)P_{D=t}$. If S is not ϵ-close to distribution D we say it is ϵ-far from the distribution.*

Consider that $(k - i + 1)P_{D=t}$ is the expected number of items of type t in a set of $k - i + 1$ draws from D. When $k - i + 1$ is large, using the Chernoff bound we can show that the number of items of type t is close to the expectation. On the other hand, when $k - i + 1$ is small, the term ϵm dominates $k - i + 1$, and thus the number of items of type t is in the range $(k - i + 1 \pm \epsilon m)P_{D=t}$. Lemma 8.1 formalizes this intuition, showing that with high probability a sequence of drawn items from a distribution D is ϵ-close to D.

Definition 8.4. *Given a distribution D, a sequence of f items is said to satisfy the* long input condition *for D if $\frac{f}{\log(f)} \geq \frac{15}{\epsilon^2 min_{t \in D}(P_{D=t})}$.*

We use the following version of the Chernoff bound in Lemma 8.1.

Proposition 8.1. *Let $x_1, x_2, \ldots, x_m$ be a set of independent boolean random variables. For $X = \sum_{i=1}^{m} x_i$ and $\mu = E[X]$ we have: $Pr(|X - \mu| \geq \delta) \leq 2\exp(\frac{-\delta^2}{3\mu})$*

Lemma 8.1. *Let S be a sequence of m items drawn from a distribution D. Assuming the long input condition, the probability that S is ϵ-far from D is at most $\frac{1}{m^2}$.*

Proof. S contains $\frac{m(m-1)}{2}$ subsequences. In addition, the long input condition gives us $P_{D=t} \geq \frac{15\log(m)}{\epsilon^2 m}$. This means that there are at most $\frac{\epsilon^2 m}{15\log(m)}$ different types. Thus, we have fewer than $\frac{m^3}{2}$ combinations of a type and a subsequence. We next argue that the probability that the number of items of a fixed type t in a fixed sub sequence $S_{i,k}$ is out of the range $(k - i + 1 \pm \epsilon m)P_{D=t}$ is at most $\frac{2}{m^5}$. Applying the union bound, the probability that S is ϵ-far from D is at most $\frac{2}{m^5}\frac{m^3}{2} = \frac{1}{m^2}$, as desired.

For a type t and an index $1 \leq \ell \leq m$, let x_ℓ^t be a random variable which is 1 if the ℓ-th item in sequence S is of type t and is 0 otherwise. The variables x_k^t are independent for a fixed type t and different indices. Let $X_{i,k}^t$ be $\sum_{\ell=i}^{k} x_\ell^t$, which is the number of items of type t in the sub sequence $S_{i,k}$. The expected value of $X_{i,k}^t$

is $(k-i+1)P_{D=t}$, and by applying the Chernoff bound we have:

$$\begin{aligned}
Pr(|X^t_{i,j} - (j-i+1)P_{D=t}| \geq \epsilon m) &\leq 2\exp(\frac{-\epsilon^2 m^2}{3(j-i+1)P_{t=D}}) \\
&\leq 2\exp(\frac{-\epsilon^2 m^2}{3n\frac{m\epsilon^2}{15\log(m)}}) \\
&= 2\exp(-5\log(m)) = \frac{2}{m^5}.
\end{aligned}$$

This completes the proof of the lemma. □

In the rest of this section, we use the monotonicity and subadditivity properties of Budgeted Allocation, stated in Lemma 8.2 and Lemma 8.3 respectively.

Lemma 8.2 (Monotonicity). *Budgeted Allocation is monotone: Fixing the set of agents and their capacities, for any sequence of items S and any sequence $T \subseteq S$, we have $Opt(T) \leq Opt(S)$ where $Opt(S)$ and $Opt(T)$ are the values of the optimum solutions when the items that arrive are S and T respectively.*

Proof. Any feasible allocation of items of T is a feasible allocation of items of S as well. This immediately means $Opt(T) \leq Opt(S)$ as desired. □

Lemma 8.3 (subadditivity). *Budgeted Allocation is subadditive: Fixing the set of agents and their capacities, for any sequence of items S and any sequence of items T, we have $Opt(S \cup T) \leq Opt(S) + Opt(T)$ where $Opt(X)$ indicates the size of the optimum solution when the sequence of items that arrive is X.*

Proof. Fix an optimum solution $Opt(S \cup T)$. The allocation of items of S in $Opt(S \cup T)$ is a feasible allocation for S. Similarly the allocation of items of T in $Opt(S \cup T)$ is a feasible allocation for T. Therefore we have $Opt(S \cup T) \leq Opt(S) + Opt(T)$. □

Lemma 8.1 says that w.h.p, a sequence of items drawn from a distribution D is ϵ-close to D. That is, inputs which are ϵ-far from the input distribution are rare, but this does not immediately imply that the total value of such rare ϵ-far inputs is small as well. Lemma 8.4 says that we may ignore all inputs which are ϵ-far from the input distribution and only lose a small fraction in the competitive ratio.

Lemma 8.4. *Let S be a sequence of m items drawn from a distribution D, satisfying the long input condition. Let Alg be an α-competitive algorithm for Stochastic Budgeted Allocation with forecast (D, m). Let Alg_{close} be an algorithm that has the same outcome as Alg when the input is ϵ-close to D and 0 otherwise. Alg_{close} is $(\alpha - \frac{1}{m})$-competitive.*

Proof. Let Alg_{far} be an algorithm that has the same outcome as Alg when the input is ϵ-far from D and 0 otherwise. We slightly abuse notation and use $Alg, Alg_{\text{close}}, Alg_{\text{far}}$ to refer both to the algorithms and the expected values of their outcomes. By definition we have $Alg = Alg_{\text{close}} + Alg_{\text{far}}$. Let Opt denote the expected value of an optimal solution on the sequence drawn from the distribution. We bound the expected outcome of Alg_{far} to compare the competitive ratio of Alg and Alg_{close}.

Let S_k be an input that contains k items of each type. The monotonicity of Budgeted Allocation implies that for any sequence S of size m, $Opt(S_m)$ is greater than $Opt(S)$. Moreover, by subadditivity, $Opt(S_m)$ is at most $\frac{m}{2}$ times $Opt(S_2)$. Together, these imply that, for any sequence S of size m, we have $Opt(S) \leq \frac{m}{2} Opt(S_2)$. On the other hand, any sequence S of size m which is ϵ-close to D contains at least 2 of each item in D and thus by monotonicity we have

$\frac{1}{Pr(S \text{ is } \epsilon \text{ close to } D)} Alg_{\text{close}} \geq Opt(S_2)$. Therefore, for *any* sequence S of size m we have

$$Opt(S) \leq \frac{m}{2} Opt(S_2) \leq \frac{m}{2} \frac{1}{(1 - \frac{1}{m^2})} Alg_{\text{close}} \leq m Alg_{\text{close}} \leq m Opt$$

This together with Lemma 8.1 gives us

$$Alg_{\text{close}} = Alg - Alg_{\text{far}} \geq Alg - m Opt \frac{1}{m^2} \geq \alpha Opt - \frac{1}{m} Opt = (\alpha - \frac{1}{m}) Opt$$

as desired. □

Note that any algorithm *Alg* for Budgeted Allocation has a random outcome when the items are drawn from a forecast, simply due to the randomness in the sequence of items. We now define a derandomization of such algorithms: Given an algorithm *Alg*, a forecast $F = (D, f)$ and a constant ϵ, algorithm $DeRand_{\epsilon}^{F}(Alg)$ is defined as follows:

Let S' be a sequence of $(1-\epsilon)f$ impressions, with $(1-\epsilon)fP_{D=t}$ impressions of type t, for each type t. Run algorithm *Alg* on sequence S'. Let $Alg(S', t, i)$ be the agent to which *Alg* assigns the ith impression of type t in S'. Note that any sequence of f items which is ϵ-close to D contains at least $(1-\epsilon)fP_{D=t}$ impressions of each type t. We can now describe how $DeRand_{\epsilon}^{F}(Alg)$ allocates items of a sequence S. For each type t, Algorithm $DeRand_{\epsilon}^{F}(Alg)$ allocates the first $(1-\epsilon)f \cdot P_{D=t}$ impressions of type t in S in the same manner as *Alg* allocated S'. That is, we assign the ith impression of type t in S to $Alg(S', t, i)$. After the first $(1-\epsilon)fP_{D=t}$ impressions of type t have been allocated, we do not allocate any more items of this type. Finally, if at any time during the algorithm, we observe that the input sequence (so far) is not ϵ-close to distribution D, the algorithm stops and returns false. Otherwise, it returns true.

When it is clear from the context, we drop F and ϵ from the notation of $DeRand_{\epsilon}^{F}(Alg)$.

Remark Note that for any forecast $F = (D, f)$ and constant ϵ, the outcome of $DeRand_{\epsilon}^{F}(Alg)$ on any item sequence of length f that is ϵ-close to D is a function purely of D and ϵ, but not the actual impressions in the sequence.

Theorem 8.1. *Let $F = (D, f)$ be a distribution, and let A be an adversarial input with length f such that A satisfies the long input condition and A is ϵ-close to D. Let Alg be an α-competitive algorithm for Stochastic Budgeted Allocation. Though A is not explicitly drawn from D, $DeRand_{\epsilon}^{F}(Alg)$ is $\alpha - 2\epsilon$ competitive on sequence A.*

Proof. Since A is ϵ-close to D, $DeRand(Alg)$ will not return false on S, and thus, the allocation of $DeRand(Alg)$ on S is identical to that of Alg on S'. Let S'' be a sequence of $(1+\epsilon)d$ impressions, $(1+\epsilon)fP_{D=t}$ from each type t. Using the subadditivity of Budgeted Allocation, we have $Opt(S') \geq \frac{1-\epsilon}{1+\epsilon}Opt(S'') \geq (1-2\epsilon)Opt(S'')$.

On the other hand, using the monotonicity of Budgeted Allocation, $Opt(A) \leq Opt(S'')$. Together, these inequalities imply that $Opt(S') \geq (1-2\epsilon)Opt(A)$, which means that $DeRand_{\epsilon}^{F}(Alg)$ is $(\alpha - 2\epsilon)$-competitive. □

Consider an α-competitive online algorithm (or α-approximate offline algorithm), Alg, for Budgeted Allocation with stochastic input. By Theorem 8.1, for inputs which are ϵ-close to D, $DeRand_{\epsilon}^{F}(Alg)$ has a competitive ratio that is only

2ϵ worse than that of *Alg*. Moreover, Lemma 8.4 says that if we ignore all inputs which are ϵ-far from the input distribution, we lose only $\frac{1}{m}$ on the competitive ratio. Together with the assumption that $\epsilon \geq \frac{1}{m}$, we obtain the following corollary.

Corollary 8.1. *Let Alg be an α-competitive algorithm for Stochastic Budgeted Allocation (or α-approximate offline algorithm for Budgeted Allocation). For any small constant ϵ and any forecast F, DeRand$_\epsilon^F$(Alg) is an $(\alpha - 3\epsilon)$-competitive algorithm for Stochastic Budgeted Allocation.*

8.3 Robust Online Budgeted Allocation

In this section, we consider the Robust Budgeted Allocation problem. As described above, the algorithm knows in advance the set of agents Y, together with a capacity c_j for each agent $j \in Y$. Further, it knows the forecast $F = (D, f)$, but not how many additional items will be sent by the adversary, nor how they are intermixed with the f items drawn from D.

Recall that a λ-stochastic instance is one where the 'stochastic items' (those drawn from D) provide λ-fraction of the value of an optimal solution. (Also, note that λ is not known to the algorithm.) As λ increases (corresponding to an increase in the accuracy of our forecast, or a smaller traffic spike), we expect the performance of our algorithms to improve. However, we wish our algorithms to be robust, obtaining good performance compared to an optimal offline solution even when λ is close to 0 (corresponding to a very large traffic spike, when the typical 'forecast' traffic is only a small fraction of the total).

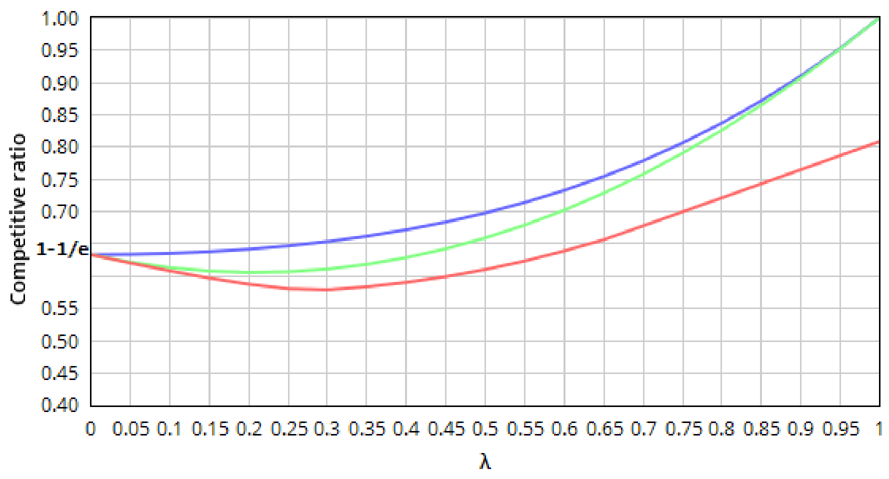


Figure 8.1: Summary of results, parametrized by λ. The blue curve denotes the upper bound on the competitive ratio of any algorithm, the green curve is our algorithm for the unweighted case, and the red curve is our algorithm for the weighted case.

First, in Section 8.3.1, we consider the unweighted Robust Budgeted Allocation problem, in which w_{ij} is the same for all $(i, j) \in E$. As desired, we obtain an algorithm with competitive ratio tending to 1 as λ tends to 1, and $1 - 1/e$ when λ tends to 0. Then, in Section 8.3.2, we consider the general weighted Robust Budgeted Allocation problem, and finally, in Section 8.3.3, we give upper bounds on the competitive ratio of any algorithm. All our competitive ratios are parametrized by λ, and they are summarized in Figure 8.1.

For simplicity, throughout this section, we assume that the capacity of all the agents are the same (normalized to 1). This assumption can be removed by dividing agents with large capacities into multiple dummy agents. Moreover, as usual we assume $\max_{i,j} w_{i,j} \to 0$ (a.k.a., large degree assumption).

8.3.1 Unweighted Robust Budgeted Allocation

Our general approach will be to *simulate* the following idea: Suppose we first receive the f stochastic items drawn from D. We could allocate these items optimally (assuming they are the entire input). If this uses up most of the budgets of the agents, we automatically have a good solution, even if we do not allocate any of the adversarial items. If some fraction of the budgets remain unused, use this remaining capacity to allocate any adversarial items that arrive. If there is a way to allocate these adversarial items using the remaining capacities, we obtain $1 - 1/e$ fraction of this value.

Unfortunately, in the real Robust Budgeted Allocation model, we do not know which items are stochastic and which are adversarial, so we cannot perform this clean separation perfectly. Still, we can approximate this separation as follows: Let A be an algorithm for Stochastic Budgeted Allocation. We have two algorithms running simultaneously. The first is a slight variant of $DeRand_{\epsilon}^{F}(A)$, and the second is Balance [195]. More precisely, we send each item to $DeRand_{\epsilon}^{F}(A)$; recall that this algorithm first checks if the input seen so far is ϵ-close to D. If it is, it allocates this item according to A; otherwise, it returns false. Now, instead of returning false, we assume that this item must have been sent by the adversary. As such, we erase this item from the history of $DeRand_{\epsilon}^{F}(A)$, and try to allocate it using Balance. That is, we 'guess' that all items that are matched by $DeRand_{\epsilon}^{F}(A)$ are stochastic items and all other items are adversarial.

Note that $DeRand_{\epsilon}^{F}(A)$ does not match more than $(1-\epsilon)f$ items, $(1-\epsilon)fP_{D=t}$

from each type t. From Lemmas 8.1 and 8.4, we know that w.h.p., the sequence of stochastic items is ϵ-close to D; by ignoring the case when it is ϵ-far from D, we lose at most $\frac{1}{m}$ in the competitive ratio. Given the long input assumption, $1/m < \epsilon$, and hence by losing ϵ in the competitive ratio, we assume that the stochastic items are always ϵ-close to D. Since the sequence of stochastic items is ϵ-close to D, we have at least $(1-\epsilon)fP_{D=t}$ items of type t. Therefore, the items that $DeRand_{\epsilon}^{F}(A)$ leaves unallocated are a superset of all the adversarial items. (More precisely, there may be an adversarial item of type t that we guess is stochastic, but there must be a corresponding stochastic item of type t that we treat as adverarial instead.)

We now complete the description of our combined algorithm: Given an allocation, let x_j denote the fraction of agent j's budget used in this allocation. Let Alg_S be a $(1-\epsilon)$-competitive algorithm for Stochastic Budgeted Allocation with the further property that it minimizes $\sum_{j=1}^{n} x_j^2$. (In other words, Alg_S reproduces an optimal offline solution such that each item is allocated to an eligible agent with the lowest x_j.) We run $DeRand_{\epsilon}^{F}(Alg_S)$ as the first of our two algorithms. Recall from Remark 8.2.2 we know the exact allocation of $DeRand_{\epsilon}^{F}(Alg_S)$, which is independent of the random draws. If x_j denotes the fraction of agent j's capacity used by this allocation, for items unallocated by $DeRand_{\epsilon}^{F}(Alg_S)$, we run the Balance algorithm on the instance of Budgeted Allocation with the adversarial items, and in which the capacity of agent j is $1-x_j$.

How does this combined algorithm perform? We use S to denote the stochastic items, and A the adversarial items. From Corollary 8.1 we know that $DeRand_{\epsilon}^{F}(Alg_S)$ is $1-O(\epsilon)$ competitive against $Opt(S)$. We will prove in Lemma

8.8 that the optimum solution on the items of A using the remaining capacities $1 - x_j$ is at least $(1 - \lambda - O(\epsilon))(Opt(A \cup S) - Opt(S))$. Since Balance is a $(1 - 1/e)$-competitive algorithm, the value we derive from Balance is at least $(1 - \frac{1}{e})(1 - \lambda - O(\epsilon)) \cdot (Opt(A \cup S) - Opt(S))$. Therefore, the competitive ratio of the combined algorithm is at least:

$$\frac{(1 - O(\epsilon))Opt(S) + (1 - \lambda - O(\epsilon))(1 - \frac{1}{e})(Opt(A \cup S) - Opt(S))}{Opt(S \cup A)}$$
$$= \frac{\lambda Opt(S \cup A) + (1 - \lambda)^2(1 - \frac{1}{e})Opt(S \cup A) - O(\epsilon)}{Opt(S \cup A)}$$
$$= \lambda + (1 - \lambda)^2(1 - \frac{1}{e}) - O(\epsilon)$$

where the first equality uses the definition of λ-stochastic to replace $Opt(S)$ with $\lambda Opt(S \cup A)$. This proves the following theorem.

Theorem 8.2. *Assuming the long input condition, there exists a $\lambda + (1 - \lambda)^2(1 - \frac{1}{e}) - O(\epsilon)$ competitive algorithm for Robust Budgeted Allocation when the input is λ-stochastic.*

In order to prove the key Lemma 8.8, we first write Mathematical Program 8.2, and show in Lemma 8.6 that this lower bounds the ratio of the optimum allocation of the adversarial items on the remaining capacities to the overall optimum. Next, in Lemma 8.7, we show that the solution of this mathematical program is at least $1 - \lambda$. Lemma 8.8 is an immediate result of combining these two lemmas.

We now show that there exists an optimum allocation of all items such that the contribution of stochastic items in this allocation is exactly $Opt(S)$. We will need this for Lemma 8.6.

$$
\begin{array}{lll}
\text{Minimize:} & \dfrac{\sum_{j=1}^{n}(y_j - z_j)}{\sum_{j=1}^{n} y_j} & \\
\text{Subject to:} & z_j \le \max(0, x_j + y_j - 1) & \forall 1 \le j \le n \\
& \displaystyle\sum_{k=j+1}^{n} (x_k + y_k) \le n - j & \forall 1 \le j \le n-1 \text{ s.t. } x_j < x_{j+1} \\
& x_j \le x_{j+1} & \forall 1 \le j \le n-1 \\
& \lambda(\displaystyle\sum_{j=1}^{n} x_j + \sum_{j=1}^{n} y_j) \ge \sum_{j=1}^{n} x_j & \\
& 0 \le x_j, y_j, z_j \le 1 & \forall 1 \le j \le n
\end{array}
$$

Figure 8.2: Mathematical Program 2 to bound the optimum allocation of adversarial items on the remaining capacities in uniform weight case.

Lemma 8.5. *For any disjoint sets of items A and S, there exist an optimum allocation of $S \cup A$ such that the contribution of items of S in this allocation is exactly $Opt(S)$.*

Proof. We prove this lemma by contradiction. Assume that in all optimum allocations of $S \cup A$, the contribution of items of S is less than $Opt(S)$. Consider an optimum allocation $Opt'(S \cup A)$ of $S \cup A$ such that the items of S have the maximum contribution to the allocation. Denote the allocation of items of S in $Opt'(S \cup A)$ by $A'(S)$. By definition, $A'(S)$ is not an optimum allocation of S. Thus, there exists an augmenting path in $A'(S)$ which can increase the number of assigned items to one of the agents by exactly one, and keep the number of assigned items to all other agents the same. This change increases the number of assigned items in $A'(S)$ by one, and may decrease the number of assigned items from A by at most one. Therefore, it is an optimal solution of $S \cup A$ in which items from S have more contribution, and hence gives a contradiction. $\square$

Lemma 8.6. *The optimum allocation of the adversarial items on the remaining capacities is at least $Opt(S \cup A) - Opt(S)$ times the solution of Mathematical Program 8.2.*

Proof. Let x_j be the fraction of the capacity of the agent j which is filled by Alg_S. Without loss of generality, assume that x_js are in increasing order. That is, for all $1 \leq j \leq n-1$, we have $x_j \leq x_{j+1}$, which is the third constraint in Mathematical Program 8.2.

Consider the optimum solution of $S \cup A$ from Lemma 8.5. Let y_j be the fraction of the capacity of the agent j that is filled by the adversarial items in this solution. One can see that the fourth constraint says that the contribution of the stochastic items is at most λ fraction of the total value. (From Lemma 8.5, we could have equality in the fourth constraint, but for simplicity of analysis, we maintain the inequality.)

Note that we want to compare the optimum solution of the adversarial items on the remaining capacities $1 - x_j$ with the total value $\sum_{j=1}^{n} y_j$ of the adversarial items in the optimum solution of $S \cup A$. For some agent j, if we have $y_j + x_j \leq 1$, we can assign the same adversarial items to the agent j as in the optimum allocation. On the other hand, if $y_j + x_j \geq 1$, we can only use $1 - x_j$ fraction of the capacity of agent j for the adversarial items. Thus, we can always assign $y_j - max(0, x_j + y_j - 1)$ fraction of the capacity of agent i to the adversarial items. This quantity is denoted by z_j in the first constraint.

By assigning adversarial items as in the optimum solution of $S \cup A$ using the remaining capacities, we can obtain at least $\sum_{j=1}^{n} y_j - z_j$. The objective function of Mathematical Program 8.2 compares this to what the optimum solution of $S \cup A$ can get from adversarial items.

Finally, fix an index j such that $x_j < x_{j+1}$ and look at all agents with index greater than j. All stochastic items that we match to agents with index greater than j have no edge to the agents with index less than or equal to j. (By definition of Alg_S, which assigns items to the eligible agent with lowest value of x_j.) Thus, in any optimum solution of S, they cover at least $\sum_{k=j+1}^{n} x_k$ of the agents with index greater than j. Thus, this optimum solution of $S \cup A$ covers $\sum_{k=j+1}^{n} (x_k + y_k)$ of the agents with index greater than j. Consider that we have $n-j$ such agents, we get $\sum_{k=j+1}^{n}(x_k + y_k) \leq n - j$. This is the second constraint in Mathematical program 8.2. □

Lemma 8.7. *For any small number δ, the solution of the mathematical program 8.2 is at least $1 - \lambda - O(\delta)$.*

Proof. First, we show that given any solution to the Mathematical Program 8.2, we can construct a related solution with a very restricted structure that only increases the objective function by $O(\delta)$. We will lower bound the objective value of this new solution, which will imply a lower bound on the original competitive ratio.

Our restricted solution will satisfy Properties 8.3, 8.4, 8.5, 8.6 ,8.7, and 8.8, which we state below. Finally, we use these properties to bound the value of the objective function by $1 - \lambda$. This means that the solution of MP 8.2 is bounded by $1 - \lambda - O(\delta)$.

Property 8.3. *The sum of y_js is much larger than 1, that is, for any positive number δ, we have, $\frac{1}{\sum_{j=1}^{n} y_j} \leq \delta$.*

Consider a solution $Sol^n = <(x_1, \ldots, x_n), (y_1, \ldots, y_n), (z_1, \ldots, z_n)>$, to the MP 8.2 with n agents. For any integer C, one can construct an equivalent feasible solution, $Sol^{Cn} = <(x'_1, \ldots, x'_{Cn}), (y'_1, \ldots, y'_{Cn}), (z'_1, \ldots, z'_{Cn})>$ to an extended Mathematical Program equivalent to 8.2 with Cn agents as follow. For all $1 \leq j \leq n$ and all $1 \leq k \leq C$, set

$$x'_{C(j-1)+k} = x_j, \quad y'_{C(j-1)+k} = y_j, \quad z'_{C(j-1)+k} = z_j.$$

On can easily check that if Sol^n is feasible, Sol^{Cn} is feasible as well. The value of the objective functions in both Sol^n and Sol^{Cn} are the same. Thus, if we bound the value of the objective function in Sol^{Cn}, this bound holds for Sol^n as well. Moreover, since we have C copies of each variable here, we have $\sum_{j=1}^{Cn} y'_j = C\sum_{j=1}^{n} y_j$. Hence, if we set $C = \frac{1}{\delta \sum_{j=1}^{n} y_j}$, it satisfies Property 8.3. In the rest of the proof, we assume that the solution that we are considering has Property 8.3, and for simplicity we refer to it as a solution with n agents and variables x_js, y_js and z_js.

Note that Property 8.3 means that if we decrease the value of a constant number of z_js to 0, we only increase the objective value by $O(\delta)$. We later use this to provide Property 8.4.

Definition 8.5. *We say an agent j is* harmful *if we have $z_j > 0$. Otherwise, we say the agent is a sink.*

If an agent j is harmful, z_j is positive and thus it contributes to reducing the objective value below 1.

Property 8.4. *All y_js are either 0 or 1.*

Let j be the largest index such that the agent j is harmful and y_j is neither 1 nor 0. If this is not the only harmful agent with this property, there is some other index k such that agent k is harmful and y_k is neither 1 nor 0. We decrease the value of y_j and z_j by some ϵ and increase y_j and z_k by ϵ. This change has no effect on the objective function, or the third and fourth constraints. It increases both sides of the first constraint by ϵ for agent k and decreases both sides by ϵ for agent j, keeping them both feasible. One can verify that this change may only decrease the left hand side of the second constraint for indices between k and j.

We can repeat this procedure until there is just one harmful agent j with a fractional y_j. We decrease y_j and z_j of this agent to 0, and by property 8.3, only increase the objective value by $O(\delta)$. Note that harmful agents with $y_j = 0$ now become sink agents.

We can use the same procedure to make the y_js of all sink agents either 0 or 1.

Property 8.5. *If agent j is harmful, $z_j = x_j$.*

For each harmful agent j, the value of z_j can be as much as x_j (since y_j for a harmful agent is now 1). Increasing z_js of harmful agents up to their x_js may only decrease the objective function, and keeps all the constraints feasible.

From now on, for each harmful agent j, we keep x_j and z_j the same, and if we change one of these two, we change the other one as well. This implicit change of z_j keeps the constraints feasible. However, each time we decrease x_j of a harmful agent, we need to guarantee that the objective function does not increase. This guarantee is trivial when we are increasing the value of some other x_k to match.

Property 8.6. *If an agent $j > 1$ is a sink, we have $x_{j-1} = x_j$.*

Let j and h be two consecutive harmful agents with $j < h$. For all agents k in the range $[j, h)$ we replace the value of x_k with the average value of xs in this range i.e. we set $x_k = \frac{\sum_{k=j}^{h-1} x_k}{h-j}$. This keeps the sum of all x_ks the same, and keeps all of the constraints feasible. Moreover, since x_j has the minimum value in the range $[j, h)$, and agent j is the only harmful agent in this range, we are only increasing the x valuess of harmful agents. This only decreases the objective function.

Let k be the number of different x_j values that we have, and let α_h be the h-th smallest value among distinct x_js. For simplicity, we assume we always have some $x_j = 0$. Let β_h be the number of x_js which are at most α_h, and set $\beta_0 = 0$. Thus, $\beta_{h-1} \leq j < \beta_h$ means that $x_j = \alpha_h$.

Property 8.7. *For any $1 \leq h \leq k$, all harmful agents in the range $(\beta_{-h1}, \beta h]$, are the first agents in the range.*

Let i be a sink and let $i+1$ be a harmful agent, such that $x_i = x_{i+1}$. We can replace the value of the variables for these two agents and make i harmful, instead of $i+1$. It is easy to see that this keeps the mathematical program feasible. By repeating this, for each range $(\beta_{j-1}, \beta j]$, we can move all harmful agents to be the first agents in the range.

Property 8.8. *All x_js are either 0 or the same fixed number x^*.*

We iteratively decrease the number of distinct x_js as follow.

Let ξ be some small number that we set later. It is easy to see that one of the following two actions decreases the objective function, since the effect of one on the objective function is the negative of the effect of the other one. The action that decreases the objective function is the one that we do.

- For all $\beta_1 < j \leq \beta_2$ decrease x_j by $\frac{\xi}{\beta_2 - \beta_1}$ and for all $\beta_2 < \ell \leq \beta_3$ increase x_ℓ by $\frac{\xi}{\beta_3 - \beta_2}$

- For all $\beta_1 < j \leq \beta_2$ increase x_j by $\frac{\xi}{\beta_2 - \beta_1}$ and for all $\beta_2 < \ell \leq \beta_3$ decrease x_ℓ by $\frac{\xi}{\beta_3 - \beta_2}$

We can set ξ, such that the mathematical program remains feasible and one of the following situations happen.

- x_js in the range $(\beta_1, \beta_2]$ are all 0.
- x_js in the range $(\beta_2, \beta_3]$ are all x_{β_3+1}.
- The second constraint of the MP is tight for $j = \beta_2$.

In the first two cases, the number of distinct x_is is reduced by one and we are done in these cases.

Therefore, we assume that we are in the third case. Now suppose that $k > 3$ (that is, there are more than 3 distinct values of x_j); later, we consider the case that $k = 3$. Let ξ' be some small number that we set later. For all $\beta_2 < j \leq \beta_{k-1}$ we increase x_j by $\frac{\xi'}{\beta_{k-1} - \beta_2}$ and for all $\beta_{k-1} < \ell \leq \beta_k$ we decrease x_ℓ by $\frac{\xi'}{\beta_k - \beta_{k-1}}$. Below, we show that by doing so, the objective function does not increase. We can set ξ' to make the largest and second largest value of x_js become equal. This decreases the number of distinct values of x_is.

The second constraint in MP 8.2 for $j = \beta_{k-1}$ says that the fraction of harmful agents in the range $(\beta_{k-1}, \beta_k]$ is at most $1 - x_n$. This constraint is tight for $j = \beta_2$. Thus, the fraction of harmful agents in the range $(\beta_2, \beta_k]$ is at least $1 - x_n$, which means the fraction of harmful agents in the range $(\beta_2, \beta_{k-1}]$ is at least $1 - x_n$. Thus, taking some fraction of x_ℓs in the range $(\beta_{k-1}, \beta_k]$ and distributing it equally over the range $(\beta_2, \beta_{k-1}]$ increases the sum of z_js and thus decreases the objective function.

In the case that $k = 3$, we move the harmful agents in the range $(\beta_1, \beta_2]$ to some higher indices, to make the second constraint tight. Then, we can use the above technique and decrease the value of the only two non zero distinct x_js to become equal.

Now, given the property 8.8 we can easily bound the objective function of the mathematical program as follow. If $x^* \leq \lambda$, all z_js are at most λ, while y_js are all 1. This bounds the objective function by $1 - \lambda$. On the other hand, if $x^* > \lambda$, the second constraint in MP 8.2 says that the number of harmful agents is at most $1 - x^*$ fraction of the value from the stochastic items. This means that sum of z_js is at most $1 - x^*$ fraction of value of the stochastic items. Thus, it is at most $(1 - x^*)\frac{\lambda}{1-\lambda} \leq \lambda$ and this bounds the objective function by $1 - \lambda$. □

The following lemma is an immediate result of combining Lemma 8.6 and Lemma 8.7.

Lemma 8.8. *The optimum allocation of adversarial items on the remaining capacities is at least* $(1 - \lambda - O(\delta))(Opt(S \cup A) - Opt(S))$.

8.3.2 Weighted Robust Budgeted Allocation

We can now describe our algorithm for general weighted case of Budgeted Allocation. As in the unweighted case, we combine the allocations of two algorithms: One that runs on the (stochastic) items that we guess are drawn from the forecast, and one on the items that we guess are constructed by the adversary.

For the stochastic items, we start with a base algorithm for Stochastic Budgeted Allocation that is *not necessarily optimal.* Instead of maximizing the weight of the allocation from the stochastic items, we start with an algorithm Alg_{pot} that *maximizes the potential* of the allocation, as defined below.

Definition 8.6. *Let $X = (x_1, x_2, \ldots, x_n)$ be a vector of numbers, such that for all $1 \leq j \leq n$, we have $0 \leq x_j \leq 1$. We define the potential of x_j, $Pot(x_j)$ to be $x_j - e^{(x_j - 1)}$. We define the potential of the vector X, $Pot(X)$ to be $\sum_{j=1}^{n} Pot(x_j)$.*

Let x_j denote the fraction of capacity c_j used by the potential-maximizing allocation of Alg_{pot}. Similarly to the unweighted case, when items arrive, we send them to *DeRand*(Alg_{pot}); for those items that are unmatched, by *DeRand*(Alg_{pot}), we send them to the Balance algorithm using the remaining capacities $1 - x_j$. Exactly the same argument that we provide for the unweighted case works here to show that by losing $O(\epsilon)$ on the competitive ratio, we can assume that we match all stochastic items using *DeRand*(Alg_{pot}) and all adversarial items using the Balance algorithm. We use *Alg* to denote this combined algorithm. In order to analyze our algorithm *Alg*, we need to define another potential function based on $Pot(X)$.

Definition 8.7. *Let $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ be two vectors of numbers between 0 and 1. For each $1 \leq j \leq n$, we define $Pot_X(y_j)$ as follow. If $x_j \leq y_j$, we have $Pot_X(y_j) = Pot(y_j)$. Otherwise, we have $Pot_X(y_j) = Pot(x_j) + (y_j - x_j)Pot'(x_j)$, where $Pot'(.)$ is the first derivative of $Pot(.)$. Thus, for $x_j < y_j$ we have:*

$$Pot_X(y_j) = x_j - e^{x_j - 1} + (y_j - x_j)(1 - e^{x_j - 1})$$

We define $Pot_X(Y)$ to be $\sum_{j=1}^{n} Pot_X(y_j)$.

Note that the second derivative of $Pot(x_j)$ is $-e^{x_j-1}$ which is always negative. Thus, $Pot(x_j)$ is a concave function. Therefore, for a fixed x_j, $Pot_X(y_j)$ is concave in the range $[0, x_j]$. Moreover, $Pot_X(y_j)$ is linear in the range $(x_j, 1]$ and thus, $Pot_X(y_j)$ is a concave function.

Consider that, in the range $(x_j, 1]$, the function $Pot_X(y_j)$ is the degree 2 Taylor series of $Pot(y_j)$ at point x_j. In addition, the second derivative of $Pot(y_j)$ in the range $[0, 1]$ is lower-bounded by -1, yielding the following lemma.

Lemma 8.9. *For any constant ϵ and any vector of positive numbers X and any y_j such that $0 \leq y_j \leq x_j + \epsilon \leq 1$ we have $|Pot_X(y_j) - Pot(y_j)| \leq \epsilon^2$.*

We now have the tools to analyze *Alg* via means of a mathematical program: First, in Lemma 8.10, we show that the competitive ratio of our algorithm is lower bounded by the solution of the Mathematical program 8.4. Next, in Lemma 8.11, we lower bound the solution of the mathematical program; this lower bound is shown in Figure 8.3. Together, these lemmas allow us to prove the following theorem:

Theorem 8.9. *There exists an algorithm for λ-stochastic weighted budgeted allocation with competitive ratio presented in Figure 8.3.*

Lemma 8.10. *The competitive ratio of Alg is bounded by the solution of the Mathematical program 8.4.*

Proof. For each agent j, let x_j be the expected fraction of c_j used by Alg_{pot} on the stochastic items. Consider the optimum assignment Opt_λ that maximizes $\frac{E[Stochastic(Opt_\lambda)]}{E[Opt_\lambda]}$. For each agent j, let t_j and y_j respectively denote the expected

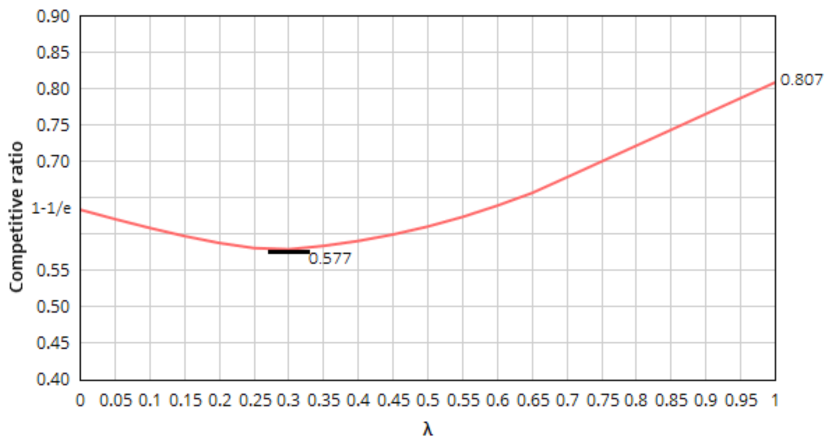


Figure 8.3: Competitive ratio of Alg parametrized by λ.

$$\text{Minimize:} \quad \frac{\sum_{j=1}^{n} x_j + (1-\frac{1}{e})\sum_{j=1}^{n}(y_j - z_j)}{\sum_{j=1}^{n} y_j + \sum_{j=1}^{n} t_j} \tag{8.1}$$

$$\text{Subject to:} \quad z_j \leq \max(0, x_j + y_j - 1) \qquad \forall 1 \leq j \leq n \tag{8.2}$$

$$Pot_X(T) \leq Pot(X) \tag{8.3}$$

$$t_j + y_j \leq 1 \qquad \forall 1 \leq j \leq n \tag{8.4}$$

$$\lambda(\sum_{j=1}^{n} t_j + \sum_{j=1}^{n} y_j) = \sum_{j=1}^{n} t_j \tag{8.5}$$

$$0 \leq t_j, x_j, y_j, z_j \leq 1 \qquad \forall 1 \leq j \leq n \tag{8.6}$$

Figure 8.4: Mathematical Program 4 to bound the competitive ratio of Alg.

fraction of c_j used by stochastic items and adversarial items in Opt_λ. Therefore, the expected fraction of c_j used in Opt_λ is $t_j + y_j$, which is upper-bounded by 1. This gives us the Inequality 8.4.

By definition of λ, the contribution of the stochastic items to Opt_λ is λ fraction of the total value; equality 8.5 captures this fact.

For any agent j, if we have $y_j \leq 1 - x_j$, an offline algorithm can allocate all the same adversarial items to agent j as Opt_λ did by using the remaining $1 - x_j$ fraction of c_j. On the other hand, if we have $y_j > 1 - x_j$, the offline algorithm can use the entire remaining capacitity $1 - x_j$ of agent j. Thus, an offline algorithm that assigns all the adversarial items using the $1 - x_j$ fraction of capacities that remain (after allocating the stochastic items according to Alg_{pot}) only loses at most $\max(0, x_j + y_j - 1)$ from agent j when compared to the allocation of adverarial items by Opt_λ. We denote $\max(0, x_j + y_j - 1)$ by z_j, as shown in Inequality 8.2.

One can see that the numerator of the objective function lower-bounds the expected outcome of Alg and the denominator is the expected outcome of Opt_λ. Thus, the objective function is a lower bound on the competitive ratio of Alg.,

It remains only to verify inequality 8.3. By definition of Alg_{pot} as an algorithm maximizing potential, we know that $Pot(T) \leq Pot(X)$. However, this does not imply the inequality directly. We will show that if the inequality does not hold, we will be able to construct a new allocation with a larger potential, contradicting the definition of X. Suppose by way of contradiction that there exists some T, and some positive number δ such that $Pot_X(T) - Pot(X) \geq \delta$. For some arbitrary small constant η, consider the vector $\eta T + (1-\eta)X$, which allocates η fraction of each stochastic item according to T and $1-\eta$ fraction of each stochastic item according to X. By concavity of $Pot_X(.)$ we have

$$Pot_X(\eta T+(1-\eta)X) \geq \eta Pot_X(T)+(1-\eta)Pot_X(X) = \eta Pot_X(T)+(1-\eta)Pot(X).$$

Together with the assumption that $Pot_X(T) - Pot(X) \geq \delta$, this gives $Pot_X(\eta T + (1-\eta)X) \geq Pot(x) + \eta\delta$. On the other hand, Lemma 8.9 implies that $Pot_X(\eta T+(1-\eta)X) \leq Pot(x)+n\eta^2$. By setting η to be smaller than $\frac{\delta}{n}$, we obtain a contradiction. Thus, for any solution T we have $Pot_X(T) \leq Pot(X)$ which is the inequality 8.3. □

Lemma 8.11. *The solution of Mathematical Program 8.4 is lower-bounded by the curve in Figure 8.3.*

Proof. Again, as in the unweighted case, given any solution to Mathematical program 8.4, we find a new solution with a restricted structure that only increases the objective function by $O(\delta)$. Eventually, we lower bound the objective value of this new solution; this provides a lower bound on the original competitive ratio.

Our restricted solution will satisfy properties 8.10, 8.11, 8.12 and 8.13 below. Similar to the unweighted case, we can replace each variable with with $O(1/\delta)$ copies with the same value in order to satisfy Property 8.10, which implies that if we change a constant number of the y_j and / or t_j variables, it changes the objective function by at most $O(\delta)$.

Property 8.10. *For any positive number δ, we can create an equivalent solution such that $\frac{1}{\sum_{j=1}^{n} y_j} \leq \delta$ and $\frac{1}{\sum_{j=1}^{n} t_j} \leq \delta$.*

Again as in the unweighted section, we call an agent j with positive z_j a *harmful* agent. We say that an agent j is a *source* if both z_j and t_j are zero. If an agent is neither harmful nor a source, we call it a *sink*.

Property 8.11. *All y_js and t_js are either 0 or 1.*

To satisfy this property, we add some new extra agents with $x_{\cdot} = 0$, $y_{\cdot} = 0$, $z_{\cdot} = 0$, $t_{\cdot} = 0$ initially. (Note that this requires slightly modifying the mathematical program to add constraints for the new agents, and including them in the summations. However, it is easy to see that this does not change the objective or feasibility of any constraint.)

Now, for any original agent j with positive y_j and such that j is either a sink or a source, we decrease y_j to 0 and increase y_k by the same amount for some newly added agent k. It is easy to see that this does not affect the sum of the y variables (and hence the objective function or the feasibility of any constraint), and that this can be done so that only a single agent has a fractional value of y.

Let Γ be an integer such that the sum of t_js for all sinks is in the range $[\Gamma, \Gamma + 1)$. (Note that sources have $t_j = 0$.) We redistribute the t_j values to be 1 on the Γ sources and sinks with the maximum x_js, leaving at most one sink with fractional t_j. Using the concavity of $Pot_X(T)$, this change only decreases $Pot_X(T)$. Since all sources and sinks now have $y_j = 0$, we still have $t_j + y_j \leq 1$ for each agent j. (Note that some agents may change status from source to sink or vice versa during this process.)

Now, y_j and t_j are either 0 or 1 for all except the harmful agents. (Further,

y_j is 1 only on newly added agents, and t_j is 1 only on sink agents.)

For the harmful agents, let Y_H and T_H denote the sum of their y_j and t_j values respectively. From constraint (8.4), we know that $Y_H + T_H$ is less than the total number of harmful agents. We redistribute the y_j and t_j values as follows: Set $y_j = 1$ on the $\lfloor Y_H \rfloor$ harmful agents with the smallest values of x_j, and 0 otherwise; similarly, set $t_j = 1$ on the $\lfloor T_H \rfloor$ harmful agents with the largest values of x_j, and 0 otherwise. In this process, we lose the fractional y_j and t_j values of at most one agent, possibly increasing the competitive ratio by $O(\delta)$. Further, note that this process leaves some of these agents harmful (those with $y_j = 1$), while the others (all but $\lfloor Y_H \rfloor$ of them) are no longer harmful; $\lfloor T_H \rfloor$ of them now become sinks. There are two further effects of this move: First, as argued above for the sinks, by setting $t_j = 1$ on the agents with highest x_j, $Pot_X(T)$ only decreases. Second, the redistribution of y_j values affects the z_j values. To satisfy constraint (8.2), we must decrease z_j to 0 on each agent which is no longer harmful. However, we can increase the z_j values on the the $\lfloor Y_H \rfloor$ agents which are still harmful, and it is easy to see that this only decreases the objective function (because the increase in these z_j values exceeds the decrease on the agents which are no longer harmful).

Therefore, we satisfy Property 8.11 while increasing the objective by at most $O(\delta)$.

For each harmful vertex i, Inequality (8.2) says that $z_j \leq x_j$ (since $y_j = 1$). Increasing z_j to x_j only decreases the objective function. Thus, in the rest of the proof, for any harmful vertex j we assume $z_j = x_j$. Changing x_j of a harmful vertex

j implicitly affects z_j as well.

Property 8.12. *All harmful agents have the same x_j, denoted by α_1. All sink agents have the same x_j, denoted by α_2.*

We replace the x_j of all harmful agents with their average. By concavity of the potential function, this does not decrease $Pot(X)$. (Note that all harmful agents j have $t_j = 0$.) It also does not change $\sum_{j=1}^n x_j$ and $\sum_{j=1}^n z_j$, and thus, keeps all the constraints feasible.

We also replace the x_j of sinks with their average. One can rewrite Inequality 8.3 as $\sum_{j=1}^n (Pot_X(t_j) - Pot(x_j)) \leq 0$. The left term for sink j is $Pot(x_j) + (1 - x_j)(1 - e^{x_j - 1}) - Pot(x_j) = (1 - x_j)(1 - e^{x_j - 1})$. Thus, for each sink, the left term is a convex function. This means that this change does not increase the left hand side of $\sum_{j=1}^n (Pot_X(t_j) - Pot(x_j)) \leq 0$. This change also does not affect $\sum_{j=1}^n x_j$ and $\sum_{j=1}^n z_j$, and hence keeps all the constraints feasible.

We use β_1 to denote the number of agents j with $y_j = 1$; this includes harmful agents and newly added agents. We use β_2 and β_3 to denote the number of sinks and sources respectively. We use γ to denote $\sum_j x_j$ for sources.

Property 8.13. *$\sum_{i \text{ is source}} Pot(x_i)$ is bounded by $(1 - \frac{1}{e})\gamma - \frac{1}{e}\beta_3$*

This follows from the fact that $Pot(x)$ is always less than $(1 - \frac{1}{e})x - \frac{1}{e}$.

Now, using Properties 8.11, 8.12 and 8.13, one can show that the objective function is lower bounded by

$$\frac{(\alpha_1\beta_1 + \alpha_2\beta_2 + \gamma) + (1 - \frac{1}{e})(1 - \alpha_1)\beta_1}{\beta_1 + \beta_2}. \tag{8.7}$$

(This is a lower bound because in the numerator, we have $\sum_{(} y_j - z_j)$, which is 1 for the newly added agents included in β_1, and $(1-\alpha_1)$ for the harmful agents.)

We can write Equality (8.5) as

$$\lambda(\beta_1 + \beta_2) = \beta_2, \tag{8.8}$$

and write Inequality (8.3) as

$$\beta_1(0 - e^{-1}) + \beta_2(\alpha_2 - e^{\alpha_2 - 1} + (1-\alpha_2)(1 - e^{\alpha_2 - 1})) + \beta_3(0 - e^{-1}) \leq$$
$$\beta_1(\alpha_1 - e^{\alpha_1 - 1}) + \beta_2(\alpha_2 - e^{\alpha_2 - 1}) + (1 - \frac{1}{e})\gamma + \beta_3(0 - e^{-1}).$$

One can drop the terms $\beta_2(\alpha_2 - e^{\alpha_2 - 1})$ and $\beta_3(0 - e^{-1})$ from both sides of the inequality, and rearrange as follows:

$$\frac{e}{e-1}(\beta_1(-\frac{1}{e} - \alpha_1 + e^{\alpha_1 - 1}) + \beta_2(1-\alpha_2)(1 - e^{\alpha_2 - 1})) \leq \gamma.$$

We replace γ in the objective function (Equation 8.7 with the max of zero and left hand side of the inequality. Then, we replace β_1 with $\frac{1-\lambda}{\lambda}\beta_2$ using Equality (8.8). Now we can cancel out β_2 and simplify the objective function as follows.

$$\frac{\alpha_1}{e}(1-\lambda) + (1-\lambda)(1-\frac{1}{e}) + \lambda\alpha_2 +$$
$$\max(0, \frac{e}{e-1}((1-\lambda)(-\frac{1}{e} - \alpha_1 + e^{\alpha_1 - 1}) + \lambda(1-\alpha_2)(1 - e^{\alpha_2 - 1})))$$

Unfortunately, it is hard to solve this analytically to obtain a closed-form experssion for the competitive ratio as a function of λ. When $\lambda \geq 0.6882$, we can lower bound the competitive ratio by the line $0.3714 + 0.4362\lambda$. However, this is not necessarily tight. Numerically solving, the curve lower bounding the competitive ratio as a function of λ can be seen in Figure 8.3 □

8.3.3 Hardness of Robust Budgeted Allocation

Theorem 8.14. *No online algorithm for Robust Budgeted Allocation (even in the unweighted case) has competitive ratio better than* $1 - \frac{1-\lambda}{e^{1-\lambda}}$ *for* λ*-stochastic inputs.*

Proof. Theorem 8.14 Consider the following instance: We have n agents and n items. Initially, all of the agents are *unmarked*. Each item is connected to all agents which are unmarked upon its arrival. The first λn items are connected to all agents. After the arrival of the first λn items, we *mark* the λn agents with minimum [expected] load. Subsequently, after assigning each item we *mark* the agent with the minimum load among all unmarked agents.

Note that all of the first λn items are adjacent to all agents, and there is no uncertainty about them. Thus, one can consider the first λn items as being part of the forecast $F = (D, \lambda n)$, where the distribution D has a single type (items adjacent to all agents). The last $(1-\lambda)n$ vertices as adversarially chosen items. Let v_j denote the jth agent that we mark. An optimum algorithm that knows the whole graph in advance can match the jth item to the jth agent and obtain a matching of size n.

Now consider an arbitrary online algorithm. Let ℓ_j^i be the [expected] load of agent v_j after assigning the ith item and let $\ell_j = \ell_j^n$ be its [expected] load at the end of the algorithm. One can see that $\ell_j = \ell_j^{\max(\lambda n, j)}$. The first λn items put a load of λn on the agents and agents v_1 to $v_{\lambda n}$ are the λn agents with lowest loads. Thus, we have

$$\sum_{j=1}^{\lambda n} \ell_j \leq \lambda^2 n.$$

In addition, for each $\lambda n < j \leq n$, the first j items put a load of at most j

$$
\begin{aligned}
\text{Maximize:} \quad & \frac{\sum_{j=1}^{n} \ell_j}{n} \\
\text{Subject to:} \quad & \sum_{j=1}^{\lambda n} \ell_j \le \lambda^2 n \\
& \ell_j \le \frac{j - \sum_{k=1}^{j-1} \ell_k}{n-j+1} \qquad \forall\, \lambda n + 1 \le j \le n \\
& 0 \le \ell_j \le 1 \qquad\qquad\qquad \forall\, 1 \le j \le n
\end{aligned}
$$

Figure 8.5: Linear program to bound the competitive ratio of the best online algorithm.

on the agents, and the load on the agents v_1 to v_{j-1} is $\sum_{k=1}^{j-1} \ell_k$. Therefore, we can bound the load of the jth agent as follows:

$$\ell_j \le \frac{j - \sum_{k=1}^{j-1} \ell_k}{n-j+1}.$$

Further, the expected load of an agent cannot exceed 1, and the optimum solution is a matching in which each agent has a load of 1. Therefore, the competitive ratio of any algorithm for Robust Budgeted Allocation on λ-stochastic inputs is bounded by Linear program 8.5.

First note that w.l.o.g., we can assume that in any optimal solution to LP 8.5, each ℓ_j for $1 \le j \le \lambda n$ has the same value. Consider an optimal solution in which as many constraints as possible are tight. It is easy to see that if the constraint for ℓ_j is not tight, we can raise ℓ_j by ϵ and decrease ℓ_k by at most ϵ (to maintain feasibility) for the first $k > j$ such that $\ell_k > 0$. By repeating this procedure iteratively, we make all of the constraints tight. One can see that, this solution is correspond to the algorithm that matches items to their eligible agents uniformly, and that the objective function sums up to $1 - \frac{1-\lambda}{e^{1-\lambda}}$. $\square$

8.4 Approximating Adversarial and Stochastic Budgeted Allocation

In this section, we study a class of algorithms for Budgeted Allocation problem that provide good approximation ratios in both stochastic and adversarial settings. We say an algorithm is (α, β)-competitive if it is α-competitive in the adversarial setting and β-competitive in the stochastic setting.

The best algorithms for the stochastic setting have a competitive ratio of $1-\epsilon$ when the input is stochastic. However, these algorithms may have a competitive ratio close to zero when the input is adversarial. On the other hand, the Balance algorithm has the best possible competitive ratio of $1-\frac{1}{e}$ when the input is adversarial, but, under some mild assumption, is 0.76-competitive for stochastic inputs [187]. Indeed, this result holds when, 1) the capacities are large i.e. $\max_{i,j} \frac{w_{i,j}}{c_j} \leq \epsilon^3$, and 2) the optimum solution is large i.e. $\epsilon^{-6} \sum_j \max_{i:opt(i)=j} w_{i,j} \leq Opt$, where $Opt(i)$ is the agent the optimum matches to node i, and ϵ is a vanishingly small number. In this section, we refer to $\max_{i,j} \frac{w_{i,j}}{c_j} \leq \epsilon^3$ as the large capacities condition, and refer to $\max\left(\epsilon^{-6} \sum_j \max_{i:opt(i)=j} w_{i,j}, \epsilon^{1/3} m \cdot \max_{i,j} w_{i,j}\right) \leq Opt$ as the large optimum condition.

Of course, one can design algorithms with 'intermediate' performance: Simply use the stochastic algorithm with probability p and Balance with probability $1-p$; this yields an algorithm with competitive ratios of $((1-p)(1-\frac{1}{e})$ and $(p + (1 -$

$p) \times 0.76) - O(\epsilon))$ in the adversarial and stochastic settings respectively. These competitive ratios for different values of p lie on the straight line with endpoints $(1 - \frac{1}{e}, 0.76)$ and $(0, 1 - \epsilon)$. In this section, we use the tools that we developed in Section 8.2 to obtain competitive ratios those are 'above' this straight line.

We devise a mixed algorithm for this setting as follows: Divide the capacity of each agent / offline vertex into two parts. The first part has $1 - p$ fraction of the capacity, and the second part has the remaining p fraction of the capacity. Given a forecast $F = (D, f)$, at the beginning, run the Balance algorithm on $1 - p$ fraction of the forecast input (that is, on the first $(1 - p) \cdot f$ items, using the first $(1 - p)$ fraction of the capacities. Then, for any $1 - \epsilon$-competitive stochastic algorithm *Alg*, run *Alg* on the rest of the forecast input (that is, the next pf items), using the remaining p fraction of the capacities. If at any time during the algorithm, we observe that the input sequence is ϵ-far from D, or the length of the input is more than f, we can detect that (w.h.p.) we are in the adversarial setting. We therefore flush out the items we assigned to the second part (the p fraction) of the capacities, if any, and return to the Balance algorithm. However, we now run two independent copies of Balance: The first using $1 - p$ fraction of the capacities (with some items previously allocated by Balance), and the second using the remaining p fraction of the capacities (beginning with an empty allocation of items to this part of the capacities). For each item arriving online, with probability $1 - p$ we send it to the Balance algorithm on the first part of the capacities, and with probability p we send it to the Balance algorithm on the second part of the capacities.

In the next two lemmas, we bound the competitive ratios of our mixed algorithm when the input is stochastic and adversarial respectively.

Lemma 8.12. *Assuming the long input, the large capacities and the large optimum conditions, if the input is stochastic, for any constant* $p \in [0,1]$ *the mixed algorithm has a competitive ratio of* $(1-p) * \beta + p - O(\epsilon)$, *where* β *is* 0.76.

Proof. From Theorem 8.1, we know that by ignoring the inputs which are ϵ-far from the distribution, we lose only $O(\epsilon)$ fraction in the competitive ratio. We use $1-p$ fraction of the input sequence on the first $1-p$ fraction of the capacities. Therefore, the Balance algorithm gets $(1-p) * \beta$ fraction of the optimal solution from this part of the input, where $\beta = 0.76$ is the competitive ratio for Balance on stochastic inputs [187]. On the other hand, we use p fraction of the input on the remaining p fraction of the capacities. Since we use a $1-\epsilon$ competitive stochastic algorithm, we get $p - O(\epsilon)$ fraction of the optimal solution from this part of the input sequence. □

Lemma 8.13. *Assuming the long input, the large capacities and the large optimum conditions, if the input is adversarial, for any constant* $p \in [0,1]$ *the mixed algorithm has a competitive ratio of* $\frac{0.76(1-p)}{1+0.202(1-p)} - O(\epsilon^{1/3})$.

Proof. Let S be the maximal prefix of the input before observing that the input is not ϵ-close to the distribution and let A be the rest of the input. (Note that either S or A may be empty.) If S contains at least $\epsilon^{2/3}m$ nodes, since S is $\frac{\epsilon}{\epsilon^{2/3}} = \epsilon^{1/3}$-close to the distribution and satisfies the long input condition, we obtain at least $\beta(1-p-O(\epsilon^{1/3}))Opt(S)$ from this fraction of the input (Note that if S is smaller than $(1-p) \cdot f$, for example, we would obtain a larger fraction of $Opt(S)$.). On the other hand, if S contains less than $\epsilon^{2/3}m$ nodes, using large optimum condition, we have $Opt(S) \leq \epsilon^{2/3}m \cdot max_{i,j}w_{i,j} \leq \epsilon^{1/3}Opt(A \cup S)$. Thus, the competitive ratio is lower bounded by

$$\frac{\beta(1-p)Opt(S)}{Opt(S \cup A)} - O(\epsilon^{1/3}) \geq \frac{\beta(1-p)Opt(S)}{Opt(S) + Opt(A)} - O(\epsilon^{1/3}).$$

On the other hand, if we ignore ϵm nodes, the Balance algorithm on the first $1-p$ fraction of the capacities always runs on at least $1-p$ fraction of the items, we always get at least $(1-\frac{1}{e})(1-p-O(\epsilon^{2/3}))Opt(S \cup A)$ from this fraction of the capacities. Further, once the input stops being ϵ-close to the distribution (that is, for the entire sequence A), we use the remaining p fraction of the capacities on p fraction of A. Therefore, we always get at least $(1-\frac{1}{e})pOpt(A)$ from this p fraction of the capacities. Thus, the competitive ratio is lower bounded by

$$\frac{(1-\frac{1}{e})(1-p-O(\epsilon^{2/3}))Opt(S \cup A) + (1-\frac{1}{e})pOpt(A)}{Opt(S \cup A)} =$$
$$\left(1-\frac{1}{e}\right)(1-p) + \frac{(1-\frac{1}{e})pOpt(A)}{Opt(S \cup A)} - O(\epsilon^{2/3}) \geq$$
$$\left(1-\frac{1}{e}\right)(1-p) + \frac{(1-\frac{1}{e})pOpt(A)}{Opt(S) + Opt(A)} - O(\epsilon^{2/3})$$

where the inequality follows from the subadditivity of Budgeted Allocation.

The first of these lower bounds is decreasing in $Opt(A)$, while the second lower bound is increasing in $Opt(A)$. Thus, the worst case happen when these two bounds are equal. By setting the two equal and $\beta = 0.76$, one can achieve the desired bound. □

The following theorem is the immediate result of combining Lemmas 8.12 and 8.13.

Theorem 8.15. *Assuming the long input, the large capacities and the large optimum conditions, for any constant* $p \in [0,1]$ *the mixed algorithm is* $(1-p) * 0.76 + p - O(\epsilon)$ *competitive for stochastic input and* $\frac{0.76(1-p)}{1+0.202(1-p)} - O(\epsilon^{1/3})$ *competitive for adversarial input.*

Chapter 8: Conclusions and Open Problems

In Chapter 2, we provide a streaming algorithm to estimate the size of the maximum matching in bounded arboricity graphs using a sublinear space. Later in Chapter 3 we extend our results to dynamic streams. In fact, an interesting problem left open here is to extend this result to general graphs.

Challenge 8.1. *What is the best approximation algorithm to estimate the size of the maximum matching in the streaming setting, using a sublinear space?*

Another interesting open problem is to provide the best approximation algorithm that finds an approximate maximum matching in the streaming setting using $\tilde{O}(n)$.

Challenge 8.2. *What is the best approximation algorithm to find an approximate matching in the streaming setting using $\tilde{O}(n)$ space?*

In Chapter 5, we study the prophet secretary problem. In particular, we present a $1 - \frac{1}{e}$ approximation algorithm for this problem, while showing that there is no algorithm with approximation factor better than 0.75. In Chapter 6 we improve this approximation factor to 0.738 for iid distributions. An interesting problem left open here is to improve the result for prophet secretary.

Challenge 8.3. *What is the best approximation algorithm for the prophet secretary problem?*

Another interesting challenge is to generalizing this result to matroids.

Challenge 8.4. *What is the best approximation algorithm for matroid prophet secretary, where we are allowed to pick more than one item, but they should form an independent set of a fixed matroid?*

Bibliography

[1] L. Lovasz and M.D. Plummer. Matching theory. In *North-Holland, Amsterdam-New York*, 1986.

[2] S. Micali and V. V. Vazirani. An $o(\sqrt{|V|}|e|)$ algorithm for finding maximum matching in general graphs. *Proceedings of the 21st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 17–27, 1980.

[3] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2):207–216, 2005.

[4] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci.*, 381(1-3):183–196, 2007.

[5] H. N. Nguyen and K. Onak. Constant-time approximation algorithms via local improvements. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 327–336, 2008.

[6] Y. Yoshida, M. Yamamoto, and H. Ito. An improved constant-time approximation algorithm for maximum matchings. *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 225–234, 2009.

[7] A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 22–31, 2009.

[8] Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *SODA*, pages 1123–1131, 2012.

[9] M. Kapralov, S. Khanna, and M. Sudan. Approximating matching size from random streams. *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 734–751, 2014.

[10] Alexandr V. Kostochka. Lower bound of the hadwiger number of graphs by their average degree. *Combinatorica*, 4(4):307–316, 1984.

[11] Zoran Ivkovic and Errol L. Lloyd. Fully dynamic maintenance of vertex cover. In *WG*, pages 99–111, 1993.

[12] K. Onak and R. Rubinfeld. Maintaining a large matching and a small vertex cover. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 457–464, 2010.

[13] S. Baswana, M. Gupta, and S. Sen. Fully dynamic maximal matching in $O(\log n)$ update time. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 383–392, 2011.

[14] O. Neiman and S. Solomon. Simple deterministic algorithms for fully dynamic maximal matching. *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 745–754, 2013.

[15] Manoj Gupta and Richard Peng. Fully dynamic $(1 + \epsilon)$-approximate matchings. In *FOCS*, pages 548–557, 2013.

[16] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM J. Comput.*, 38(5):1709–1727, 2008.

[17] V. Guruswami and K. Onak. Superlinear lower bounds for multipass graph processing. *Proceedings of the 28th Annual IEEE Conference on Computational Complexity (CCC)*, pages 287–298, 2013.

[18] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *SODA*, pages 468–485, 2012.

[19] M. Kapralov. Better bounds for matchings in the streaming model. *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1679–1697, 2013.

[20] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 352–358, 1990.

[21] List of open problems in sublinear algorithms: Problem 60. http://sublinear.info/60.

[22] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *APPROX-RANDOM*, pages 231–242, 2012.

[23] A. McGregor. Finding graph matchings in data streams. In *Proceedings of the 8th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 170–181, 2005.

[24] A. McGregor. Graph mining on streams. In *Encyclopedia of Database Systems*, pages 1271–1275, 2009.

[25] S. Eggert, L. Kliemann, P. Munstermann, and A. Srivastav. Bipartite graph matchings in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, 2012.

[26] Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, 2012.

[27] Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *ICALP (2)*, pages 526–538, 2011.

[28] Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012.

[29] Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011.

[30] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

[31] Yijie Han. Matching for graphs of bounded degree. In *Frontiers in Algorithmics*, pages 171–173. Springer, 2008.

[32] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*, volume 6. Macmillan London, 1976.

[33] Ziv Bar-Yossef, T. S. Jayram, and Iordanis Kerenidis. Exponential separation of quantum and classical one-way communication complexity. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 128–137, 2004.

[34] Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. Exponential separation for one-way quantum communication complexity, with applications to cryptography. *SIAM J. Comput.*, 38(5):1695–1708, 2008.

[35] Elad Verbin and Wei Yu. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 11–25, 2011.

[36] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pages 459–467, 2012.

[37] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 5–14, 2012.

[38] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Spectral sparsification in dynamic graph streams. In *APPROX*, pages 1–10, 2013.

[39] Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570, 2014.

[40] Michael Kapralov and David P. Woodruff. Spanners and sparsifiers in dynamic streams. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 272–281, 2014.

[41] Ashish Goel, Michael Kapralov, and Ian Post. Single pass sparsification in the streaming model with edge deletions. *CoRR*, abs/1203.4900, 2012.

[42] Konstantin Kutzkov and Rasmus Pagh. Triangle counting in dynamic graph streams. In *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, pages 306–318, 2014.

[43] Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos E. Tsourakakis. Space and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *STOC*, 2015.

[44] Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2237–2246, 2015.

[45] Sudipto Guha, Andrew McGregor, and David Tench. Vertex and hypergraph connectivity in dynamic graph streams. In *PODS*, 2015.

[46] Hossein Esfandiari, MohammadTaghi Hajiaghayi, and David P. Woodruff. Applications of uniform sampling: Densest subgraph and beyond. *CoRR*, abs/1506.04505, 2015.

[47] Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa Vu. Densest subgraph in dynamic graph streams. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milano, Italy, August 24-28, 2014. Proceedings, Part I*, 2015.

[48] Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014.

[49] Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1234–1251, 2015.

[50] Stefan Fafianie and Stefan Kratsch. Streaming kernelization. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 275–286, 2014.

[51] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, New York, 1999.

[52] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.

[53] Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *ICALP (2)*, pages 526–538, 2011.

[54] Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011.

[55] Kook Jin Ahn and Sudipto Guha. Laminar families and metric embeddings: Non-bipartite maximum matching problem in the semi-streaming model. *Manuscript, available at http://arxiv.org/abs/1104.4058*, 2011.

[56] Andrew McGregor. Finding graph matchings in data streams. *APPROX-RANDOM*, pages 170–181, 2005.

[57] Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012.

[58] Michael Crouch and Daniel S. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 96–104, 2014.

[59] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2):207–216, 2005.

[60] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751, 2014.

[61] Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697, 2013.

[62] Christian Konrad and Adi Rosén. Approximating semi-matchings in streaming and in two-party communication. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 637–649, 2013.

[63] List of open problems in sublinear algorithms: Problem 64. http://sublinear.info/64.

[64] S. Muthukrishnan. *Data Streams: Algorithms and Applications.* Now Publishers, 2006.

[65] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Tight bounds for linear sketches of approximate matchings. *CoRR*, abs/1505.01467, 2015.

[66] Christian Konrad. Maximum matching in turnstile streams. *CoRR (To appear at ESA 2015)*, abs/1505.01460, 2015.

[67] Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. *CoRR (To appear at ESA 2015)*, abs/1505.02019, 2015.

[68] Barna Saha and Lise Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 697–708, 2009.

[69] Yuval Emek and Adi Rosén. Semi-Streaming Set Cover - (Extended Abstract). In *ICALP*, pages 453–464, 2014.

[70] Jaikumar Radhakrishnan and Saswata Shannigrahi. Streaming algorithms for 2-coloring uniform hypergraphs. In *Algorithms and Data Structures - 12th International Symposium, WADS 2011, New York, NY, USA, August 15-17, 2011. Proceedings*, pages 667–678, 2011.

[71] He Sun. Counting hypergraphs in data streams. *CoRR*, abs/1304.7456, 2013.

[72] Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 367–376, 2015.

[73] Bjarni V. Halldórsson, Magnús M. Halldórsson, Elena Losievskaja, and Mario Szegedy. Streaming algorithms for independent sets. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, pages 641–652, 2010.

[74] H. Jowhari, M. Saglam, and G Tardos. Tight bounds for L_p samplers, finding duplicates in streams, and related problems. In *Proceedings of the 17th ACM SIGMOD Symposium on Principles of Database Systems (PODS)*, pages 49–58, 2011.

[75] Graham Cormode and Donatella Firmani. A unifying framework for ℓ_0-sampling algorithms. *Distributed and Parallel Databases*, 32(3):315–335, 2014.

[76] P. Erdos and R. Rado. Intersection theorems for systems of sets. *J. London Math. Soc.*, 35:85–90, 1960.

[77] Rajesh Hemant Chitnis, Graham Cormode, Hossein Esfandiari, Mohammad-Taghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to dynamic graph streams. *CoRR*, abs/1505.01731, 2015.

[78] Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1217–1233, 2015.

[79] Anna C. Gilbert and Piotr Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.

[80] Farid M. Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theor. Comput. Sci.*, 157(2):139–159, 1996.

[81] Eyal Kushilevitz and Noam Nisam. *Commmunication Complexity*. Cambridge University Press, 1997.

[82] Maria-Florina Balcan, Avrim Blum, Jason D. Hartline, and Yishay Mansour. Reducing mechanism design to algorithm design via machine learning. *Journal of Computer and System Sciences*, 74(8):1245 – 1270, 2008. Learning Theory 2005.

[83] Richard Cole and Tim Roughgarden. The sample complexity of revenue maximization. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 243–252, New York, NY, USA, 2014. ACM.

[84] Peerapong Dhangwatnotai, Tim Roughgarden, and Qiqi Yan. Revenue maximization with a single sample. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, EC '10, pages 129–138, New York, NY, USA, 2010. ACM.

[85] Justin Hsu, Jamie Morgenstern, Ryan M. Rogers, Aaron Roth, and Rakesh Vohra. Do prices coordinate markets? *CoRR*, abs/1511.00925, 2015.

[86] Jamie Morgenstern and Tim Roughgarden. The pseudo-dimension of nearly-optimal auctions. In *NIPS*, page Forthcoming, 12 2015.

[87] Venkatesan Guruswami, Jason D Hartline, Anna R Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1164–1173. SIAM, 2005.

[88] Rajesh Chitnis, Graham Cormode, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: maximal matching and vertex cover. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1234–1251. SIAM, 2015.

[89] M. Cheung and C. Swamy. Approximation algorithms for single-minded envy-free profit-maximization problems with limited supply. In *Foundations of Computer Science, 2008. FOCS '08. IEEE 49th Annual IEEE Symposium on*, pages 35–44, Oct 2008.

[90] Ning Chen, Arpita Ghosh, and Sergei Vassilvitskii. Optimal envy-free pricing with metric substitutability. *SIAM Journal on Computing*, 40(3):623–645, 2011.

[91] Ning Chen and Xiaotie Deng. Envy-free pricing in multi-item markets. *ACM Trans. Algorithms*, 10(2):7:1–7:15, February 2014.

[92] Patrick Briest and Piotr Krysta. Buying cheap is expensive: Approximability of combinatorial pricing problems. *SIAM J. Comput.*, 40(6):1554–1586, December 2011.

[93] Parinya Chalermsook, Julia Chuzhoy, Sampath Kannan, and Sanjeev Khanna. Improved hardness results for profit maximizations pricing problems with unlimited supply. In *Proceedings of APPROX*, 2012.

[94] L. S. Shapley and M. Shubik. The assignment game i: The core. *International Journal of Game Theory*, 1:111–130, 1971. 10.1007/BF01753437.

[95] Sushil Bikhchandani and John W. Mamer. Competitive Equilibrium in an Exchange Economy with Indivisibilities. *Journal of Economic Theory*, 74(2):385–413, June 1997.

[96] Hu Fu, Nicole Immorlica, Brendan Lucier, and Philipp Strack. Randomization beats second price as a prior-independent auction. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 323–323, New York, NY, USA, 2015. ACM.

[97] Zhiyi Huang, Yishay Mansour, and Tim Roughgarden. Making the most of your samples. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 45–60, New York, NY, USA, 2015. ACM.

[98] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to dynamic graph streams. pages 1326–1344, 2016.

[99] Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 170–181. Springer, 2005.

[100] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 734–751. SIAM, 2014.

[101] Hossein Esfandiari, Mohammad T Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1217–1233. SIAM, 2015.

[102] Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. *arXiv preprint arXiv:1505.02019*, 2015.

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

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

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

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

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

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

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

[110] D. P. Kennedy. Prophet-type inequalities for multi-choice optimal stopping. In *In Stoch. Proc. Applic.* 1978.

[111] U. Krengel and L Sucheston. Semiamarts and finite values. In *Bull. Am. Math. Soc.* 1977.

[112] U. Krengel and L Sucheston. On semiamarts, amarts, and processes with finite value. In *In Kuelbs, J., ed., Probability on Banach Spaces.* 1978.

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

[114] 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.

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

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

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

[118] David Assaf, Larry Goldstein, and Ester Samuel-cahn. Ratio prophet inequalities when the mortal has several choices. *Ann. Appl. Prob*, 12:972–984, 2001.

[119] S. Alaei. Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers. In *FOCS*. 2011.

[120] Robert Kleinberg and S. Matthew Weinberg. Matroid prophet inequalities. In *STOC*, pages 123–136, 2012.

[121] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *EC*, pages 18–35, 2012.

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

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

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

[125] O. Lachish. $o(\log\log \mathrm{rank})$-competitive-ratio for the matroid secretary problem. In *FOCS*. 2014.

[126] DP Kennedy. Prophet-type inequalities for multi-choice optimal stopping. *Stochastic Processes and their Applications*, 24(1):77–88, 1987.

[127] Shuchi Chawla, Jason Hartline, David Malec, and Balasubramanian Sivan. Multi-parameter mechanism design and sequential posted pricing. 2010.

[128] Qiqi Yan. Mechanism design via correlation gap. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 710–719, 2011.

[129] Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Morteza Monemizadeh. Prophet secretary. In *Algorithms-ESA 2015*, pages 496–508. Springer, 2015.

[130] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, pages 253–264. Springer, 2007.

[131] Nikhil Devanur and Thomas Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *EC*, pages 71–78, 2009.

[132] Hossein Esfandiari, Nitish Korula, and Vahab Mirrokni. Online allocation with traffic spikes: Mixing adversarial and stochastic models. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 169–186. ACM, 2015.

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

[134] Vahab S Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1690–1701. SIAM, 2012.

[135] Theodore P Hill and Robert P Kertz. Comparisons of stop rule and supremum expectations of iid random variables. *The Annals of Probability*, pages 336–345, 1982.

[136] Uriel Feige and Jan Vondrak. Approximation algorithms for allocation problems: Improving the factor of 1-1/e. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 667–676. IEEE, 2006.

[137] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 117–126. IEEE, 2009.

[138] Martin L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979.

[139] Robert Kleinberg, Bo Waggoner, and E. Glen Weyl. Descending price optimally coordinates search. In *Proceedings of the 17th ACM Conferece on Economics and Computation*, pages 23–24, 2016.

[140] Shuchi Chawla, Jason D. Hartline, and Robert D. Kleinberg. Algorithmic pricing via virtual valuations. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 243–251, 2007.

[141] Robert Kleinberg and Seth Matthew Weinberg. Matroid prophet inequalities. In *STOC*. ACM, 2012.

[142] Michal Feldman, Nick Gravin, and Brendan Lucier. Combinatorial auctions via posted prices. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms*, pages 123–135, 2015.

[143] DP Kennedy. Optimal stopping of independent random variables and maximizing prophets. *The Annals of Probability*, 13(2):566–571, 1985.

[144] Robert P Kertz. Comparison of optimal value and constrained maxima expectations for independent random variables. *Advances in applied probability*, pages 311–340, 1986.

[145] Paul Dütting and Robert Kleinberg. Polymatroid prophet inequalities. In *Algorithms-ESA 2015*, pages 437–449. Springer, 2015.

[146] Saeed Alaei, Mohammad T Hajiaghayi, Vahid Liaghat, Dan Pei, and Barna Saha. Adcell: Ad allocation in cellular networks. In *ESA*. 2011.

[147] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The online stochastic generalized assignment problem. In *Approx*. Springer, 2013.

[148] Aviad Rubinstein. Beyond matroids: Secretary problem and prophet inequality with general constraints. *arXiv preprint arXiv:1604.00357*, 2016.

[149] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443. Society for Industrial and Applied Mathematics, 2007.

[150] Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. Online independent set beyond the worst-case: Secretaries, prophets, and periods. In *International Colloquium on Automata, Languages, and Programming*, pages 508–519. Springer, 2014.

[151] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

[152] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Cliff Stein. Online stochastic packing applied to display ad allocation. In *ESA*, pages 182–194. Springer, 2010.

[153] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. *Computing Research Repository*, 2009.

[154] Erik Vee, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. Optimal online assignment with forecasts. In *EC*, pages 109–118, 2010.

[155] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *STOC*, pages 587–596, 2011.

[156] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: A strongly factor revealing lp approach. In *STOC*, pages 597–606, 2011.

[157] Vahab S. Mirrokni, Shayan Oveis Gharan, and Morteza ZadiMoghaddam. Simultaneous approximations of stochastic and adversarial budgeted allocation problems. In *SODA*, pages 1690–1701, 2012.

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

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

[160] Anand Bhalgat, Jon Feldman, and Vahab Mirrokni. Online allocation of display ads with smooth delivery. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1213–1221. ACM, 2012.

[161] 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.

[162] 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.

[163] Nitish Korula, Vahab Mirrokni, and Morteza Zadimoghaddam. Online submodular welfare maximization: Greedy beats 1/2 in random order. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 889–898. ACM, 2015.

[164] Nitish Korula, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Bicriteria online matching: Maximizing weight and cardinality. In *Web and Internet Economics - 9th International Conference, WINE 2013, Cambridge, MA, USA, December 11-14, 2013, Proceedings*, pages 305–318, 2013.

[165] Gagan Aggarwal, Yang Cai, Aranyak Mehta, and George Pierrakos. Biobjective online bipartite matching. In *Web and Internet Economics - 10th International Conference, WINE 2014, Beijing, China, December 14-17, 2014. Proceedings*, pages 218–231, 2014.

[166] Christos H Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 86–92. IEEE, 2000.

[167] Mihalis Yannakakis. Approximation of multiobjective optimization problems. In *Algorithms and Data Structures, 7th International Workshop, WADS 2001, Providence, RI, USA, August 8-10, 2001, Proceedings*, page 1, 2001.

[168] Hossein Esfandiari, Nitish Korula, and Vahab Mirrokni. Online allocation with traffic spikes: Mixing stochastic and adversarial inputs. In *EC*. ACM, 2015.

[169] Dragos Florin Ciocan and Vivek F. Farias. Model predictive control for dynamic resource allocation. *Math. Oper. Res.*, 37(3):501–525, 2012.

[170] Bo Tan and R Srikant. Online advertisement, optimization and stochastic networks. In *CDC-ECC*, pages 4504–4509. IEEE, 2011.

[171] 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.

[172] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *stoc*, pages 67–74, 2008.

[173] Vahab S. Mirrokni, Michael Schapira, and Jan Vondrák. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In *ACM Conference on Electronic Commerce*, pages 70–77, 2008.

[174] Michael Kapralov, Ian Post, and Jan Vondrák. Online submodular welfare maximization: Greedy is optimal. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1216–1225, 2013.

[175] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1 - 1/e. In *FOCS*, pages 117–126, 2009.

[176] Vahideh H. Menshadi, Shayan Oveis Gharan, and A. Saberi. Online stochastic matching: Online actions based on offline statistics. In *SODA*, pages 1285–1294, 2011.

[177] B. Haeupler, V. Mirrokni, and M. ZadiMoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *WINE*, pages 170–181, 2011.

[178] Nikhil R Devanur, Balasubramanian Sivan, and Yossi Azar. Asymptotically optimal algorithm for stochastic adwords. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 388–404. ACM, 2012.

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

[180] Ashish Goel, Adam Meyerson, and Serge Plotkin. Approximate majorization and fair online load balancing. *ACM Transactions on Algorithms (TALG)*, 1(2):338–349, 2005.

[181] Niv Buchbinder and Joseph Seffi Naor. Fair online load balancing. *Journal of Scheduling*, 16(1):117–127, 2013.

[182] Lehman, Lehman, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behaviour*, pages 270–296, 2006.

[183] R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An optimal algorithm for online bipartite matching. In *STOC*, pages 352–358, 1990.

[184] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. COPE: traffic engineering in dynamic networks. *ACM SIGCOMM*, 36(4):99–110, 2006.

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

[186] Dimitris Bertsimas, Dessislava Pachamanova, and Melvyn Sim. Robust linear optimization under general norms. *Operations Research Letters*, 32(6):510–516, 2004.

[187] Vahab S. Mirrokni, Shayan Oveis Gharan, and Morteza ZadiMoghaddam. Simultaneous approximations of stochastic and adversarial budgeted allocation problems. In *SODA*, pages 1690–1701, 2011.

[188] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *European Symposium on Algorithms*, pages 170–181. Springer, 2010.

[189] Patrick Jaillet and Xin Lu. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39(3):624–646, 2013.

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

[191] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 557–563, 2006.

[192] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Racke. Optimal oblivious routing in polynomial time. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 383–388. ACM, 2003.

[193] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 288–294. ACM, 2007.

[194] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 18–35. ACM, 2012.

[195] Bala Kalyanasundaram and Kirk R. Pruhs. An optimal deterministic algorithm for online b -matching. *Theoretical Computer Science*, 233(1–2):319–325, 2000.