Competitive Algorithms for Online Matching and Vertex Cover Problems

by

Chiu Wai Wong

S.B., Massachusetts Institute of Technology (2012)

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

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2013

Author ....
Department of Electrical Engineering and Computer Science
August 23, 2013

Certified by ....
Michel X. Goemans
Professor
Thesis Supervisor

Accepted by ....
Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

Competitive Algorithms for Online Matching and Vertex Cover Problems

by

Chiu Wai Wong

Submitted to the Department of Electrical Engineering and Computer Science
on August 23, 2013, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The past decade has witnessed an explosion of research on the online bipartite matching problem. Surprisingly, its dual problem, online bipartite vertex cover, has never been *explicitly* studied before. One of the motivation for studying this problem is that it significantly generalizes the classical ski rental problem. An instance of such problems specifies a bipartite graph $G = (L, R, E)$ whose left vertices L are offline and right vertices arrive online one at a time. An algorithm must maintain a valid vertex cover from which no vertex can ever be removed. The objective is to minimize the *size* of the cover.

In this thesis, we introduce a charging-based algorithmic framework for this problem as well as its generalizations. One immediate outcome is a simple analysis of an optimal $\frac{1}{1-1/e}$-competitive algorithm for online bipartite vertex cover. By extending the charging-based analysis in various nontrivial ways, we also obtain optimal $\frac{1}{1-1/e}$-competitive algorithms for the edge-weighted and submodular versions of online bipartite vertex cover, which all match the best performance of ski rental.

As an application, we show that by analyzing our algorithm in the primal-dual framework, our result on submodular vertex cover implies an optimal $(1-1/e)$-competitive algorithm for its dual, online bipartite submodular matching. This problem is a generalization of online bipartite matching and may have applications in display ad allocation.

We consider also the more general scenario where all the vertices are online and the graph is not necessarily bipartite, which is known as the online fractional vertex cover and matching problems. Our contribution in this direction is a primal-dual 1.901-competitive (or $1/1.901 \approx 0.526$) algorithm for these problems. Previously, it was only known that they admit a simple well-known 2-competitive (or 1/2) greedy algorithm. Our result is the first successful attempt to beat the greedy algorithm for these two problems.

Moreover, our algorithm for the online matching problem significantly generalizes the traditional online bipartite graph matching problem, where vertices from only one side of the bipartite graph arrive online. In particular, our algorithm improves upon the result of the fractional version of the online edge-selection problem in Blum et. al. (JACM '06).

Finally, on the hardness side, we show that no randomized online algorithm can achieve a competitive ratio better than 1.753 and 0.625 for the online fractional vertex cover problem and the online fractional matching problem respectively, even for bipartite graphs.

Thesis Supervisor: Michel X. Goemans
Title: Professor

Acknowledgments

First and foremost, I am hugely indebted to my mentor Yajun Wang with whom I spent two wonderful summers at Microsoft Research. He was instrumental in shaping my approach to research, from formulating problems and laying out a research agenda to attacking an otherwise impossible problem and writing up the papers. His relentless passion for research has always been contagious and an important drive for me.

I also thank Yajun for exposing me to the fascinating area of online bipartite matching back in 2011. From there we started looking at the problem in different dimensions and much of the materials in this thesis originated from the many energizing, albeit sometimes long and late night, meetings we had. I remember vividly that I often returned to brainstorming with numerous fresh ideas after meeting with him.

The most important lesson I learned from Yajun is probably the power of simplicity. Most of our results had a much more convoluted and tedious proof at the very beginning, which would likely stay the same way without his constant push for simplicity. In retrospect, coming up with a simpler and more elegant proof enabled us to discover more general structural properties and generalize our results.

My gratitude goes to my supervisor Michel Goemans as well. He has been both an inspiring teacher and research advisor. I was first attracted to the fascinating area of combinatorial optimization thanks to his undergraduate class on the field. He had laid out the basics of combinatorial optimization in a mathematically elegant manner. The interplay between polygons and combinatorial objects was nothing but magical. Part of this thesis, especially chapter 3, would not have been here had I not taken the class with Michel. He has been influential in shaping my own research interest.

In terms of research, Michel is known for his elegant approach to combinatorial optimization, which I greatly benefited from. The result in chapter 4 was initially established only for a much more restricted setting. It was partly due to his suggestion that we discovered a more elegant and unified approach to the problem. I was particularly impressed by his unique perspective at problems which would often open up an entirely new approach.

Before MIT, I have had a fabulous secondary school life at SJC, which provided me the unparalleled flexibility to develop my interest in mathematics. I was especially fortunate to have Mr. Patrick Ching (or Ching sir) as my informal mentor over the years. From

my very first month at SJC onwards, he had been very encouraging and offered me much mentoring in my math learning. My growth had been tremendously accelerated thanks to the opportunities he had given to this young boy. Thank you for always believing in me. Mr. W.Y. Yip and Y.K. Ng, who were also math teachers at SJC, had also provided me with much mentoring and guidance.

Outside of school, I received a great deal of advice from Dr. Kin Li of HKUST. He not only introduced me to higher math, but also convinced me of the possibility of a math-oriented career. Back then I would sometimes just drop by his office and enjoy hours-long discussion with him, academic or not. My deepest appreciation for his constant support and generosity.

Finally, I am extremely fortunate and grateful that my parents have been supportive of my academic pursuit. Unlike most Hong Kong parents, they have always granted me the absolute freedom to fully engage in my true passion. Their nurturing and unwavering support have been the single most important factor in my achievement and success. This thesis is dedicated to them.

Contents

List of Figures

Chapter 1

Introduction

In classical optimisation, the entire input is often given to the algorithm before it starts executing. This setting is realistic in most settings where full information about the scenario is known well in advance (e.g. shortest path, Steiner tree). Nevertheless, in more specific application domains, the input may be generated incrementally and some irrevocable decision may have to be made along the way. For example, a trading algorithm must decide whether to buy or sell a stock before the next-step price fluctuation is revealed. Problems of this nature are called *online* optimisation problems, where the input is given in the online fashion and, typically certain properties or invariants must be maintained and hence force the algorithm to act before the full input is known.

The subject of this thesis is the problems of online matching and vertex cover in various settings. The online bipartite matching problem, which was first studied in 1990, has enjoyed a tremendous amount of attention from the theoretical computer science community in the past decade. The revival of the interest in the problem is indebted to the discovery of its connection with online advertising, which was established in [28]. Since then, the online bipartite matching problem has been generalised in different directions and investigated in less stringent models.

An instance of this problem specifies a bipartite graph $G = (L, R, E)$ with edges $E \subseteq L \times R$ between left (also called offline) vertices L and right (also called online) vertices R. An algorithm for online bipartite matching maintains a matching and strives to make it as large as possible. Initially all the left vertices are offline and hence known to the algorithm. In the online phase, at each step a vertex $v \in R$ arrives along with its incident edges and the algorithm must then irrevocably decide to which unmatched neighbor $u \in N(v)$ (if any) v is matched. It is known that the optimal competitive ratio is 2 for deterministic algorithms and $1 - 1/e$ [6, 22] for randomised algorithms.

We initiate the study of the dual of this problem, called online bipartite vertex cover, in various settings. An algorithm for this problem maintains a vertex cover instead of a matching with the requirement that no vertex can leave the cover once entered. The goal is to minimise the size of the final vertex cover.

The motivation for studying this problem is two-fold. Firstly, it is of interest in its own right since it generalises the well-known classical ski rental problem considerably (see chapter 2). Secondly, as we shall see in chapters 3 and 4, algorithms for online bipartite vertex cover often inspire corresponding competitive algorithms for online bipartite matching via the primal-dual framework. In fact, one of our main results is an optimal algorithm for a new generalisation of online bipartite matching which is obtained using this recipe.

1.1 Prior work in online matching

The online bipartite matching problem was first studied in the seminal paper by Karp et al. [22]. They gave an optimal $1-1/e$-competitive algorithm. Subsequent works studied its variants such as b-matching [17], vertex weighted version [1, 11], adwords [6, 12, 28, 11, 10, 16, 1] and online market clearing [4].

Another line of research studies the problem under more relaxed adversarial models by assuming certain inherent randomness in the inputs [13, 27, 26, 18]. Online matching for general graphs have been studied under similar stochastic models [3].

1.2 Our contribution

Our major contribution is a new charging-based algorithmic framework for online vertex cover problems. By leveraging this approach, we have successfully obtained optimal algorithms for online bipartite vertex cover in the basic, edge-weighted and submodular settings, as well as the first nontrivial competitive algorithms for online vertex cover on both bipartite and general graphs in which all vertices are online.

Another theme of this thesis is the emphasis on reverse-engineering the charging-based analysis to achieve a primal-dual analysis. This is more than just yet another analysis as it implies the dual result on online matching. In retrospect, it seems much easier to design algorithms via the charging framework as the starting point, after which one can try to apply the primal-dual method to obtain the corresponding result on matching. A more elaborate explanation of this approach can be found in chapters 3 and 4.

We summarise the main results presented in this thesis below.

- An optimal $\frac{1}{1-1/e}$-competitive algorithm for online (vertex-weighted) bipartite vertex cover (chapter 2).
- An optimal $\frac{1}{1-1/e}$-competitive algorithm for online edge-weighted bipartite vertex cover (chapter 2).
- An optimal primal-dual optimal $\frac{1}{1-1/e}$-competitive algorithm for online submodular bipartite vertex cover and matching (chapter 3).
- A primal-dual 1.901-competitive algorithm for online (fractional) vertex cover and matching in general graphs (chapter 4).

We remark that the first three results on vertex cover above generalise the well-known ski rental problem, whereas the third result on matching generalises the classical online bipartite matching problem.

Moreover, we offer information-theoretical hardness results for online vertex cover and matching in general graphs.

1.3 Preliminaries

In this section we introduce the concepts and tools used frequently throughout the thesis. The more specific tools needed are explained in the respective chapters'. We assume basic familiarity with algorithms.

1.3.1 Notation

An (undirected) graph G consists of a (finite) set of vertices V and a set of edges $M \subseteq \{(u,v) : u,v \in V\}$, where (u,v) is an unordered tuple. Given $v \in V$, the neighbours of v, denoted by $N(v)$, are the vertices adjacent to v, i.e. $u \in N(v)$ if $(u,v) \in E$.

1.3.2 Vertex cover and matching

Given $G = (V, E)$, a vertex cover of G is a subset of vertices $C \subseteq V$ such that for each edge $(u, v) \in E$, $C \cap \{u, v\} \neq \emptyset$. A matching of G is a subset of edges $M \subseteq E$ such that each vertex $v \in V$ is incident to at most one edge in M.

$\mathbf{y} \in [0,1]^V$ is a fractional vertex cover if for any edge $(u, v) \in E$, $y_u + y_v \geq 1$. We call y_v the *potential* of v. $\mathbf{x} \in [0,1]^E$ is a fractional matching if for each vertex $u \in V$, $\sum_{v \in N(u)} x_{uv} \leq 1$. It is well-known that vertex cover and matching are dual of each other.

A standard fact in combinatorial optimisation states that for bipartite graphs, the minimum vertex cover and the maximum matching problems are solvable in polynomial time (offline). In contrast, one can find a maximum matching efficiently in general graphs but the minimum veretx cover problem is NP-hard.

1.3.3 Competitive analysis

Competitive analysis, first introduced in [30], is the predominant framework for evaluating the performance of algorithms for online optimisation problem. The idea is to compare the size of the solution maintained by the online algorithm against the offline optimal solution. More formally, an algorithm, possibly randomised, is said to be *c-competitive* if there exists a constant b such that for any instance, the size of the solution ALG found by the algorithm and the size of the optimal solution OPT satisfy

$$\mathbb{E}[ALG] - c \cdot OPT \leq b \quad \text{or} \quad \mathbb{E}[ALG] - c \cdot OPT \geq b$$

depending on whether the optimisation is a minimisation or maximisation problem. The constant c is called the *competitive ratio.*

1.3.4 Adversary Models

A few different adversarial models have been considered in the literature. In this thesis, we focus on the *oblivious adversarial* model, in which the adversary must specify the input once-and-for-all at the beginning and is not given access to the randomness used by the algorithm. We remark that for deterministic algorithms, all of these adversarial models are equivalent. Readers are referred to any standard online algorithm textbook (e.g. [5]) for more details.

In the case of online bipartite matching and vertex cover, the oblivious adversary specifies the input graph and the vertex arrival order before the algorithm receives the input and is denied access to the randomness used by the algorithm.

1.3.5 Primal-dual method

The primal-dual method has its origin in approximation algorithms and has recently been applied to online algorithms [8]. Informally, given a pair of primal and dual linear programs (LP), the primal-dual method ensures that the primal and dual objective values are within some constant factor of each other. The benefit of maintaining this invariant is that we can argue that such an algorithm is competitive for both the primal and dual problems via weak duality.

We summarise the discussion in the lemma below.

Lemma 1.3.1. *Given a primal LP* $\max cx$ *s.t.* $Ax \leq b, x \geq 0$ *and dual LP* $\min by$ *s.t.* $A^T y \geq c, y \geq 0$*, let* x *and* y *be feasible solutions. If an online algorithm maintains* x *and* y *in such a way that they always remain feasible and* $cx \geq \gamma \cdot by$*, then the algorithm is* γ*-competitive for the primal problem and* $1/\gamma$*-competitive for the dual problem.*

Proof. Let x^* and y^* be the optimal solutions. By LP weak duality, we have $cx \geq \gamma \cdot by \geq \gamma \cdot by^* \geq \gamma \cdot cx^*$ and $\gamma \cdot by \leq cx \leq cx^* \leq by^*$, as desired. □

The primal-dual method is indispensable in extending the results on vertex cover (dual) to matching (primal) in chapters 3 and 4.

1.4 Organisation

This thesis is organised as follows. In chapter 2, we study the basic and edge-weighted versions of online bipartite vertex cover and, in particular, present the optimal algorithms for them. Chapter 3 extends the results to the submodular setting with an application to online matching. In chapter 4, we study the general version of online matching and vertex cover in which all vertices are online and give the first nontrivial algorithms which break the barrier of 2 implied by the greedy algorithm. We conclude in chapter 5 with some future research directions.

Chapter 2

Online Bipartite Vertex Cover

In this chapter, we study the online bipartite vertex cover problem (OBVC) in the basic, vertex-weighted and edge-weighted settings, the latter two of which generalise the first.

The basic OBVC is defined as follows. For a bipartite graph $G = (L, R, E)$ with edges $E \subseteq L \times R$ between left (also called offline) vertices L and right (also called online) vertices R, all the left vertices are offline, i.e. they are known to us in advance. In the online phase, at each step a vertex $v \in R$ arrives along with its incident edges. We are required to maintain a vertex cover C at all time by only inserting additional vertices into C, i.e. no vertex removal is allowed. Our objective is to minimise the size of the vertex cover at the end. The vertex-weighted setting is identical except that each vertex v has a weight $w_v \geq 0$ and the objective is to minimise the total weight of the vertices in the cover.

The edge-weighted setting, on the other hand, differs more substantially from the basic one. In this setting, each edge e carries a weight $w_e \geq 0$ and the algorithm is required to maintain vertex potentials $y \in [0, \infty)^L, z \in [0, \infty)^R$ such that $y_u + z_v \geq w_{uv}$ for any edge $(u, v) \in E$.

In addition to its relationship with online bipartite matching, OBVC is of interest because it generalises the ski rental problem, which is a classical topic in online algorithms. Thus there is a connection between ski rental and online bipartite matching via OBVC.

Online bipartite vertex cover as combinatorial ski rental. The ski rental problem is perhaps one of the most studied online problems. Recall that in this problem, a skier goes on a ski trip for N days but has no information about N. On each day, he has the choice of renting the ski equipment for 1 dollar or buying it for $B > 1$ dollars. His goal is to minimise the amount of money spent.

Ski rental can be reduced to online bipartite vertex cover via a complete bipartite graph with B left vertices and N right vertices. One may view this problem as ski rental with a combinatorial structure imposed. We show that the optimal competitive ratio of online bipartite vertex cover is $\frac{1}{1-1/e}$. In other words, we still have the same performance guarantee even though the online bipartite vertex cover problem is considerably more general than the ski rental problem.

Extending ski rental to the online vertex cover problem also has practical implications. Consider a factory owner receiving incoming orders which he has to process one by one[1]. Each order specifies a product v, the set of machineries $N(v)$ required to produce v and the cost $w(v)$ that the customer is willing to pay for producing v. Each machinery $u \in N(v)$ requires a capital investment of $w(u)$. The owner can accept or reject the order. In the former case, he needs to purchase $N(v)$ which may incur a huge capital investment. If he chooses to reject, he instead suffers a loss of $w(v)$, the revenue he would otherwise receive. The owner's objective is to maximize the profit, or equivalently, to minimize the sum of the total investment cost and total loss. The dilemma here is that while the cost of buying $N(v)$ may be enormous compared with $w(v)$, it could be amortized over many future incoming orders, some of which may require similar machineries to produce. This problem can be modeled as an instance of online bipartite vertex cover with weights $w(\cdot)$ on the vertices, i.e. the objective is to minimise the total weight of a vertex cover.

The connection. Recall that bipartite matching and vertex cover are dual of each other in the offline setting. It turns out that the analysis of an algorithm for online bipartite fractional matching in [6] implies an optimal algorithm for online bipartite vertex cover. On the other hand, online bipartite vertex cover generalises ski rental. This connection is especially interesting because online bipartite matching does not generalise ski rental but is the dual of its generalisation[2].

[1]This is modeled by the weighted version of our problem.

[2]Coincidentally, the first papers on online bipartite matching and ski rental were both published in 1990 but to our knowledge, their connection was not realized, or at least explicitly stated.

2.1 Related work

The ski rental problem was first studied in [21]. Karlin et al. gave an optimal $\frac{1}{1-1/e}$-competitive algorithm in the oblivious adversarial model [20]. There are many generalizations of ski rental. Of particular relevance are multislope ski rental [24] and TCP acknowledgment [19], where the competitive ratio $\frac{1}{1-1/e}$ is still achievable. The online vertex-weighted bipartite vertex cover problem presented in this chapter is also of this nature and, in fact, further generalizes multislope ski rental, as shown in the appendix.

Another line of related research deals with online integral and fractional covering programs of the form $\min\{cx \mid Ax \geq 1, 0 \leq x \leq u\}$, where $A \geq 0, u \geq 0$, and the constraints $Ax \geq 1$ arrive one after another [7]. Our online vertex cover problem also falls under this category. The key difference is that the online covering problems are so general that the optimal competitive ratios are usually not constant but logarithmic in some parameters of the input.

Finally, online vertex cover for general graphs was studied by Demange et al. [9] in a model substantially different from ours. Their competitive ratios are characterized by the maximum degree of the graph.

2.2 Problem statement

We formally state the problems tackled in this chapter below.

OBVC The left vertices L are offline and the right vertices in R arrive online one at a time. When a vertex $v \in R$ arrives, all of its incident edges are revealed. The algorithm must maintain at all time a valid monotone vertex cover C, i.e. no vertex can ever be removed from C once it is put into C. Thus the algorithm essentially decides whether to assign v or $N(v)$ to C upon the arrival of an online vertex v. The objective is to minimise the size of C at the end.

Vertex-weighted OBVC This differs from OBVC only in that each vertex v has a nonnegative weight w_v and the objective is to minimise $w(C) := \sum_{v \in C} w_v$.

Edge-weighted OBVC Each edge e carries a weight $w_e \geq 0$. The algorithm must maintain at all time a valid monotone fractional vertex cover $(\mathbf{y}, \mathbf{z})$, i.e. y_u, z_v never decrease for any $u \in L, v \in R$. Thus when an online vertex v arrives, the algorithm essentially initializes z_v and possibly increases some y_u's so that $y_u + z_v \geq w_{uv}$ for $u \in N(v)$.

The objective is to minimise the size of $(\mathbf{y}, \mathbf{z})$, i.e. $\sum_{u \in L} y_u + \sum_{z \in R} z_v$.

2.3 Our result and technique

We present *optimal* $\frac{1}{1-1/e}$-competitive algorithms for all of the problems considered in this chapter. The hardness side of our results simply follows from the fact that the optimal competitive ratio of ski rental, which is a special case of OBVC, is precisely $1 + \alpha := \frac{1}{1-1/e}$. We remark that a similar algorithm for the basic version of OBVC is implied by the primal-dual analysis of a waterfilling algorithm for online bipartite fractional matching in [6].

Lemma 2.3.1. *Online bipartite vertex cover generalizes ski rental. In particular, no algorithm for online bipartite vertex cover achieves a competitive ratio better than* $1+\alpha := \frac{1}{1-1/e}$, *which is the optimal ratio for ski rental [20].*

Proof. See appendix A for a reduction from an even more general version of ski rental to online bipartite vertex cover. □

We first present an optimal algorithm for online bipartite vertex cover (in its most basic form). The analysis of this algorithm is based on an elegant charging scheme which turns out to be quite powerful for tackling the other variants. By extending it in various nontrivial ways, we obtain charging schemes capable of tackling the more general vertex-weighted and edge-weighted versions of online bipartite vertex cover.

The first step of our approach is to show that a well-known generic rounding scheme for bipartite vertex cover also works in the online setting. Thus by leveraging this scheme, one can without loss of generality focus exclusively on the fractional version of the problem.

As will be clear in the next section, our algorithms for OBVC are waterfilling in nature. Waterfilling algorithms have been previously used for a few variants of the online bipartite matching problem (e.g. [17, 6]).

2.3.1 Rounding scheme

We present a generic rounding scheme that converts any given algorithm for online *fractional* vertex cover to an algorithm for online *integral* vertex cover in bipartite graphs. This allows us to obtain the integral version of our results on fractional vertex cover for bipartite graphs.

Let $(\mathbf{y}, \mathbf{z})$ be the fractional vertex cover maintained by the algorithm, where $\mathbf{y}$ and $\mathbf{z}$ are indexed by L and R respectively. Sample $t \in [0, 1]$ uniformly at random before the first online vertex arrives. Throughout the execution of the algorithm, assign $u \in L$ to the cover if $y_u \geq t$ and $v \in R$ to the cover if $z_v \geq 1 - t$, where L and R are the left and right vertices of the graph G respectively. As y_u and z_v never decrease in the online algorithm, our rounding procedure guarantees that once a vertex enters the cover, it will always stay there.

We next claim that this scheme gives a valid cover. Since $(\mathbf{y}, \mathbf{z})$ is always feasible, we have $y_u + z_v \geq 1 \forall (u, v) \in E$ and hence at least one of $y_u \geq t$ and $z_v \geq 1 - t$ must hold. In other words, one of u and v must be in the cover. Therefore the cover obtained by applying this scheme is indeed valid and monotone, as required.

Finally, for each vertex v with final potential y_v (or z_v), the probability that v is in the cover given by the rounding is exactly y_v (or z_v). Therefore, by linearity of expectation, the expected size of the integral vertex cover after the rounding is exactly $\sum_{u \in L} y_u + \sum_{v \in R} z_v$. Hence, this rounding scheme does not incur a loss.

We remark that a more general rounding scheme is given in the next chapter for the submodular version of OBVC.

2.4 The waterfilling algorithm and the charging-based method

In this section, we present the algorithm *GreedyAlloction* for OBVC as well its charging-based analysis.

Algorithm 1: *GreedyAllocation*

```
Input: $L$
Initialize for each $u \in L$, $y_u = 0$;
for each online vertex $v$ do
    $\max a \leq 1$ s.t. $(1 - a) + \sum_{u \in N(v)} \max\{a - y_u, 0\} \leq 1 + \alpha$;
    Let $X = \{u \in N(v) | y_u < a\}$;
    For each $u \in X$, $y_u \leftarrow a$;
    $z_v \leftarrow 1 - a$;
end
```

When an online vertex v arrives, we can choose to place v in the cover which has a cost of

1. Alternately, we can put all the vertices from $N(v)$ into the cover. In *GreedyAllocation*, we attempt to put as much $N(v)$ into the cover with a resource constraint of $1+\alpha$. *GreedyAllocation* is greedy in the sense that we try to make a, i.e. the potential on $N(v)$, as large as possible.

Now we present the charging-based analysis of *GreedyAllocation*. Let C^* be a minimum vertex cover of G. We will charge the potential increment to vertices of C^* so that each vertex of C^* is charged at most $1+\alpha$.

Given an online vertex v, we consider the following two cases.

(1) $v \in C^*$. In this case, we charge the potential increments in both $N(v)$ and v in the algorithm to v. In particular, v will be charged at most $1+\alpha$.

(2) $v \notin C^*$ which implies $N(v) \subseteq C^*$. In this case, $N(v)$ should take care the potential increment on themselves and well as $z_v = 1-a$ used by v. We describe how to charge $1-a$ to $N(v)$.

Intuitively, if $\sum_{u\in N(v)}(a-y_u) = a+\alpha$, we should charge $\frac{1-a}{a+\alpha}(a-y_u)$ to $u \in X$ since the fair "unit charge" is $\frac{1-a}{a+\alpha}$. Because $\frac{1-a}{a+\alpha}$ is decreasing in a, $\frac{1-a}{a+\alpha}(a-y_u)$ can be upper bounded by

$$\int_{y_u}^{y} \frac{1-t}{f(t)}dt.$$

The next lemma indicates that the total charge is sufficient.

Lemma 2.4.1. *Let $F(x) = \int_0^x \frac{1-t}{t+\alpha}dt$. If $\sum_{u\in X}(a-y_u) = a+\alpha$ for some set X and $a \geq y_u$ for $u \in X$, then*

$$1-a \leq \sum_{u\in X}\left(F(a)-F(y_u)\right).$$

Proof. We have the following

$$\begin{aligned}\sum_{u\in X}(F(a)-F(y_u)) &= \sum_{u\in X}\int_{y_u}^{a}\frac{1-t}{a+\alpha}dt \\ &\geq \sum_{u\in X}(a-y_u)\frac{1-a}{a+\alpha} = 1-a,\end{aligned}$$

where the inequality above holds as $\frac{1-t}{t+\alpha}$ is decreasing. □

We are ready to evaluate the performance of *GreedyAllocation*.

Theorem 2.4.2. *GreedyAllocation is $1 + \alpha$-competitive and hence optimal for the online bipartite (fractional) vertex cover problem.*

Proof. As discussed earlier, we charge the potentials used to the vertices of the minimum cover C^*. We focus on the case $v \notin C^*$, which implies $N(v) \subseteq C^*$.

The potential spent on $u \in N(v) \subseteq C^*$ is charged to u itself. The potential spent on v is $z_v = 1 - a$, where a is the final water level after processing v. Let $X \subseteq N(v)$ be the set of vertices whose potentials increased when processing v. The case $a = 1$ $(1 - a = 0)$ is trivial. When $a < 1$, we must have exhausted our resources $1+\alpha$ (otherwise one can further increase a) and thus $\sum_{u \in X}(a - y_u) = a + \alpha$, where y_u is the potential of u before processing v. We charge each vertex $u \in X$ by $F(a) - F(y_u)$. By Lemma 2.4.1, $1 - a \leq \sum_{u \in X}(F(a) - F(y_u))$, i.e. the total charges are sufficient.

Now each online vertex of C^* is responsible for $1 + \alpha$ potential used in processing itself. For each left vertex $u \in C^*$, it takes care of its own potential (which contributes at most 1 to C) as well as the incoming charges from $N(u)$. The sum of these charges cannot exceed $F(1) - F(0)$. Therefore a left vertex takes care an amount of resources at most $1 + F(1) - F(0) = 1 + \int_0^1 \frac{1-t}{t+\alpha} dt = 1 + \alpha$, where the equality holds because $\alpha = \frac{1}{1-1/e}$. This gives our desired result. □

As mentioned in section 2.3.1, it is possible to round any fractional vertex cover algorithms so the same performance is achievable in the integral vertex cover setting in expectation.

2.4.1 Extension to vertex-weighted setting

We first note that it is straightforward to show that the rounding scheme in section 2.3.1 is still applicable to the vertex-weighted setting.

To extend our approach to vertex weighted OBVC, we modify *GreedyAllocation* as follows by including the vertex weights in the constraint $(1-a) + \sum_{u \in N(v)} \max\{a - y_u, 0\} \leq 1 + \alpha$. The underlying philosophy of this is the same as before, namely that we try to cover $N(v)$ as much as possible while not using too much resources. Thus $w_v(1 - a) + \sum_{u \in N(v)} w_u \max\{a - y_u, 0\}$, which is exactly the increment in the value of our objective function, should be used in the new constraint.

The analysis of the algorithm requires a modified charging lemma.

Algorithm 2: *GreedyAllocation*

```
Input: $L$
Initialize for each $u \in L$, $y_u = 0$;
for each online vertex $v$ do
    $\max a \le 1$ s.t. $w_v(1-a) + \sum_{u \in N(v)} w_u \max\{a - y_u, 0\} \le w_v(1+\alpha)$;
    Let $X = \{u \in N(v) | y_u < a\}$;
    For each $u \in X$, $y_u \leftarrow a$;
    $z_v \leftarrow 1 - a$;
end
```

Lemma 2.4.3. *Let $F(x) = \int_0^x \frac{1-t}{t+\alpha} dt$. If $\sum_{u \in X} w_u(a - y_u) = w_v(a + \alpha)$ for some set X and $a \ge y_u$ for $u \in X$, then*

$$w_v(1-a) \le \sum_{u \in X} w_u \left(F(a) - F(y_u)\right).$$

Proof. Almost the same as Lemma 2.4.1. □

By using this modified charging lemma, our result follows from a very similar charging scheme.

Theorem 2.4.4. *GreedyAllocation is $1 + \alpha$-competitive and hence optimal for vertex-weighted OBVC.*

Proof. We charge the potentials used to the vertices of the minimum cover C^*. As before, the case $v \in C^*$ is trivial. We focus on the case $v \notin C^*$, which implies $N(v) \subseteq C^*$.

The potential spent on $u \in N(v) \subseteq C^*$ is charged to u itself. The resources spent on v is $w_v z_v = w_v(1-a)$, where a is the final water level after processing v. Let $X \subseteq N(v)$ be the set of vertices whose potentials increased when processing v. The case $a = 1$ $(1 - a = 0)$ is trivial. When $a < 1$, we must have exhausted our resources $w_v(1+\alpha)$ (otherwise one can further increase a) and thus $\sum_{u \in X} w_u(a - y_u) = w_v(a + \alpha)$, where y_u is the potential of u before processing v. We charge each vertex $u \in X$ by $w_u(F(a) - F(y_u))$. By Lemma 2.4.3, $w_v(1-a) \le \sum_{u \in X} w_u(F(a) - F(y_u))$, i.e. the total charges are sufficient.

The rest of the proof is the same as Theorem 2.4.2.

□

2.5 The edge-weighted setting

Edge-weighted OBVC requires a more intricate idea than the simple vertex-weighted generalisation above. This problem is more than just yet another generalization of OBVC and in turn ski rental. One of the incentives for studying this problem is that its dual is online bipartite weighted matching, which is an important open problem that generalizes online bipartite matching [14]. Our result shows that at least for the dual of this problem, one can still achieve the competitive ratio $1 + \alpha$.

2.5.1 Integral weighted case

We first handle the special case $w_e \in \mathbb{N}$ since it already illustrates all the necessary ingredients for the general case. Moreover, this special case deserves separate attention as it generalizes integral OBVC rather than just fractional OBVC.

Reduction to the unweighted case. The new idea needed for the integer edge-weighted version of OBVC is a reduction to the unweighted case. We split each vertex u into distinct vertices $\{u(t)\}$ for $t \in \mathbb{N}$. For each online vertex $v \in R$ and its neighbor vertex $u \in L$, $v(t)$ is connected to $u(w_{uv} - t + 1)$ for $1 \leq t \leq w_{uv}$. We can conceptually treat the vertices $\{v(t)\}$ as if they are arriving one by one.

Clearly, for a vertex cover C in the new unweighted graph, we can construct a vertex cover in the original graph by setting $y_u = |\{t \mid u(t) \in C\}|$ and $z_v = |\{t \mid v(t) \in C\}|$ for $u \in L$ and $v \in R$. This is because there are w_{uv} edges between the sets of vertices $\{u(t)\}$ and $\{v(t)\}$ in the new graph.

On the other hand, given a cover $C' = \{y'_u, z'_v\}$ for the original weighted graph, we can construct a vertex cover $C = \{y_u(t)\}_{u,t}$ in the unweighted graph as follows. Given vertex u, if $t \leq y'_u$, $y_u(t) = 1$ and $y_u(\lfloor y'_u \rfloor + 1) = y'_u - \lfloor y'_u \rfloor$. All other $y_u(t)$'s are 0. $z_v(t)$'s are set analogously.

We claim that this is a valid cover. Consider an edge e between $v(t)$ and $u(w_{uv}-t+1)$. We have several cases. (1) $t \leq z'_v$ or $w_{uv}-t+1 \leq y'_u$, e is covered. (2) $z'_v < t$ and $y'_u < w_{uv}-t+1$, then $y'_u \geq w_{uv} - z'_v > w_{uv} - t$ and hence $w_{uv} - t < y'_u < w_{uv} - t + 1$. A similar argument gives $t - 1 < z'_v < t$. Therefore, by our construction, we have $y_u(w_{uv} - t + 1) = y'_u - w_{uv} + t$ and $z_v(t) = z'_v - t + 1$, which implies that e is indeed covered.

Therefore, we can simply reuse our algorithm for the online bipartite vertex cover to solve the integer weighted case. Furthermore, the rounding scheme in Section 2.3.1 is still applicable to obtain an integral vertex cover.

Theorem 2.5.1. *Our algorithm is $1+\alpha$-competitive for online bipartite edge-weighted (integral) vertex cover and hence optimal.*

Proof. Simply apply the rounding scheme in Section 2.3.1 to $y_u(t)$ and $z_v(t)$. More concretely, we sample $\gamma \in [0,1]$ uniformly at random. Now $y_u(t)$ contributes 1 to y_u iff $y_u(t) \geq \gamma$. Similarly, $z_v(t)$ contributes 1 to z_v iff $z_v(t) \geq 1-\gamma$. This is a valid VC as we always have $y_u(w_{uv}-t+1)+z_v(t) \geq 1$ for any edge uv and hence at least one of $y_u(w_{uv}-t+1)$ and $z_v(t)$ contribute 1.

Furthermore, this scheme is still lossless as the expected contribution of $u(t)$ (or $v(t)$) is exactly $y_u(t)$ (or $z_v(t)$). □

2.5.2 The general case $w_e \in \mathbb{R}_{\geq 0}$

This case is in fact a mathematical treatment of the previous reduction in the limit. Instead of $y_u(t)$ and $z_v(t)$ for $t = 1, 2, ..., y_u, z_v : \mathbb{R}_{\geq 0} \longrightarrow [0,1]$ are piecewise constant functions on the nonnegative reals. We set[3]

$$y_u = \int_0^\infty y_u(t)dt, \quad z_v = \int_0^\infty z_v(t)dt.$$

We modify our algorithm as follows:

Lemma 2.5.2. (y,z) *as maintained by the algorithm is a valid fractional vertex cover.*

Proof. For any edge uv, from the description of the algorithm we know that for any $t \in [0, w_{uv}]$,

$$y_u(w_{uv}-t)+z_v(t) \geq a(t)+(1-a(t)) = 1.$$

Thus,

$$y_u+z_v = \int_0^\infty y_u(t)dt + \int_0^\infty z_v(t)dt \geq \int_0^{w_{uv}} (y_u(w_{uv}-t)+z_v(t))dt \geq w_{uv},$$

as desired. □

[3]Here we abuse notations by using y_u, z_v for both the cover variables maintained by the algorithm as well as functions on $[0,\infty)$.

Algorithm 3: *GreedyAllocation*

```
Input: $L$
Initialize for each $u \in L$, $y_u(t) = 0$ for $t \geq 0$;
for each online vertex $v$ do
    for $t \in [0, \max_{u \in N(v)} w_{uv}]$ do
        $\max a(t) \leq 1$ s.t.
            $(1 - a(t)) + \sum_{u \in N(v): w_{uv} \geq t} \max\{a(t) - y_u(w_{uv} - t), 0\} \leq 1 + \alpha$
        Let $X_v(t) = \{u \in N(v) \mid y_u(w_{uv} - t) < a\}$;
        For each $u \in X_v(t)$, $y_u(w_{uv} - t) \leftarrow a(t)$;
        $z_v(t) \leftarrow 1 - a(t)$;
    end
end
```

Theorem 2.5.3. *Our algorithm is* $1+\alpha$*-competitive for online bipartite edge-weighted vertex cover and hence optimal.*

Proof. Let (y^*, z^*) be an optimal solution. Consider an iteration of the algorithm.

For $t \in [0, z_v^*]$, we charge the resource density $(1 + \alpha)$ used to $z_v(t)$.

For $t > z_v^*$, we can charge the resource density $1 - a(t)$ used by v to $y_u(w_{uv} - t)$ for $u \in X_v(t)$ since their potential increased. Notice that $y_u^* \geq w_{uv} - z_v^* \geq w_{uv} - t$. The exact density to be charged to each $y_u(w_{uv} - t)$ is analogous to before. We charge $a - y'_u(w_{uv} - t)$ to $y_u(w_{uv} - t)$ itself. For $z_v(t) = 1 - a(t)$, we distribute the amount by charging

$$\int_{y'_u(w_{uv}-t)}^{a} \frac{1-x}{x+\alpha} \mathrm{d}x$$

to each $u \in X_v(t)$. Proceeding in the same way as the proof of Theorem 2.4.2, their total charge density is indeed at least $1 - a(t)$.

Now each $v \in R$, $z_v(t)$ is charged at most $(1 + \alpha) \cdot z_v^*$.

For $u \in L$ and $t \leq y_u^*$, $y_u(t)$ receives a self-charge of at most 1 and incoming charge from $N(u)$ which amounts to at most

$$\int_0^1 \frac{1-x}{x+\alpha} \mathrm{d}x = \alpha.$$

For $t \geq y_u^*$, the resource is taken care of by the neighbors of u. Therefore $y_u(t)$ is charged at most $1 + \alpha$ for $t < y_u^*$ and as a result, u is charged at most $y_u^* \cdot (1 + \alpha)$.

Therefore our algorithm is $1 + \alpha$-competitive. $\square$

Finally, we note that it is possible to put together both the vertex-weighted and edge-weighted settings. However, we have chosen not to do so as it involves no new ideas.

Chapter 3

Online Bipartite Submodular Vertex Cover and Matching

In this chapter, we study the submodular version of online bipartite vertex cover. We are able to obtain a *tight* $\frac{1}{1-1/e}$-competitive algorithm for it, thus matching the optimal result in the last chapter. Our algorithm is still greedy in nature and the analysis depend on a significant extension of the previous elegant charging scheme.

In addition to the charging-based analysis mentioned above, we also successfully analyzed our algorithm in the primal-dual framework. This implies an optimal $1 - 1/e$-competitive algorithm for online bipartite submodular matching, which generalizes online bipartite matching and has the potential to be applicable in practice, especially in online advertising.

The Adwords problem [28] generalizes online bipartite matching and has had enormous applications in online advertising [29]. In Adwords, each left vertex typically represents an advertisement (ad) to be displayed (matched) to incoming impressions, which are modeled by the online vertices on the right. Each ad $u \in L$ is associated with some budget B_u which is the maximum amount that the advertiser is willing to spend on the ad u. When an impression arrives, we have to decide immediately which ad to be displayed. The optimal ratio for this problem is also $1 - 1/e$ [28, 6].

Nevertheless, this abstraction has the shortcoming that the budget for each ad is specified independently. Indeed, it is possible that an advertiser is hosting a few different ads. Adwords would require him to specify his budget for each ad. This could potentially be wasteful since in reality, he may be willing to spend more on an ad if his budget for another ad is not exhausted. For instance, suppose that a soft drink distributor is advertising for both coke and sprite. He is willing to spend \$5 on coke, \$4 on sprite but only \$8 on both.

From the perspective of the search engine, imposing such constraints also makes sense since it may not want to serve too many ads on, say, soft drinks alone. Serving a particular category of ads too often would deprive the opportunity that the ads in other categories are displayed. It is probably in the long-term interest of the search engine company to satisfy the demand from most of its customers rather than a small subset of them.

We address this limitation by allowing an advertiser to specify the amount he is willing to spend on *each* subset of his ads. More generally, in online bipartite submodular matching, any subset $S \subseteq L$ can be matched at most $f(S)$ times, where f is a monotone submodular function on L. For this problem we are able to obtain an optimal $1 - 1/e$-competitive algorithm. Given the practicality of the previous algorithms for online matching and Adwords [29], we are hopeful that some of the ideas introduced by our algorithm will be applicable.

Finally, we argue that the assumption on f is only mild. Monotonicity is clearly reasonable. Submodularity also makes sense as one should expect to observe diminishing marginal returns for such a function f specified by an advertiser. Returning to our example on coke and sprite, if \$5 is already spent on coke, our advertiser may think that the soft drink market is more saturated than before and hence spend less on sprite (\$3) than he otherwise would (\$4).

3.1 Submodular matching and vertex cover

A set function $f : 2^L \longrightarrow \mathbb{R}$ is said to be **submodular** if for all $A, B \subseteq L$,

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B).$$

One often finds the following equivalent definition useful: for every $A, B \subseteq L$ with $A \subseteq B$ and every $e \in L$,

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B).$$

Loosely speaking, this says that the marginal return of adding an element e to a larger set is smaller. This property makes submodular functions appealing beyond its mathematical beauty as this phenomenon is observed in many real-life scenarios, especially those which arise from economic settings.

In addition, a submodular function f is **monotone** if $f(B) \geq f(A)$ for every $A, B \subseteq L$ with $A \subseteq B$.

Given a nonnegative[1] monotone submodular function $f(\cdot)$, $\mathbf{x} \in [0,1]^E$ is a **submodular matching** defined by f if for all $v \in R$,

$$x_v := \sum_{u \in N(v)} x_{uv} \leq 1,$$

and for all $S \subseteq L$,

$$x_S := \sum_{u \in S} x_u = \sum_{u \in S} \sum_{v \in N(u)} x_{uv} \leq f(S).$$

It is easy to see that this is indeed a generalization of the usual (fractional) matching which corresponds to $f(S) = |S|$.

3.2 Problem statement

We formally define the online bipartite submodular vertex cover and matching problems here.

Online bipartite submodular vertex cover (OBSVC) The setting is exactly identical to OBVC except that the objective function is $f(C \cap L) + |C \cap R|$ instead of $|C|$. Here $f(\cdot)$ is a *nonnegative monotone submodular* function. OBVC is a special case of OBSVC in which f is simply the cardinality function $f(S) = |S|$.

Online bipartite submodular matching (OBSM) The setting of OBSM is similar to OBVC. Here we have a nonnegative monotone submodular function $f(\cdot)$ on the left vertices and the algorithm maintains a submodular matching $\mathbf{x}$ instead of a vertex cover. When an online vertex v arrives, the algorithm must initialize all x_{uv} for $u \in N(v)$ so that $\mathbf{x}$ is still a valid submodular matching. The objective is to maximize the size of $\mathbf{x}$, i.e. $\sum_{e \in E} x_e = \sum_{u \in L} x_u = \sum_{v \in R} x_v$.

[1]Our results actually still hold even if $f(S) < 0$ for some $S \subseteq L$, in which case we can just remove S as its vertices can never be matched.

Although not directly related to our results, the offline version of both problems can be solved by polymatroid intersection in polynomial time.

3.3 Our result and technique

By extending our previous charging scheme in a nontrivial manner, we are able to tackle the more general submodular version of online bipartite vertex cover and hence obtain an optimal $\frac{1}{1-1/e}$-competitive algorithm for online bipartite submodular vertex cover (OBSVC). Our algorithm is still greedy in nature.

Furthermore, unlike the last chapter, we also give an alternate primal-dual analysis of our algorithm for OBSVC. As a by-product, we have the following result on online bipartite submodular matching.

- An optimal $(1 - 1/e)$-competitive algorithm for online bipartite submodular matching (OBSM).

Our charging scheme for OBSVC is a significant extension of the one developed in the last chapter. To tackle submodularity, we invoke the notion of Lovasz extension and introduce a bar chart representation for it. Not only is this helpful for intuition, the representation also enables us to explain our analysis more succiently. Our new scheme is based on a *two-dimensional* charging function of the bar chart diagram. The analysis is therefore much more involved than the original scheme.

Our primal-dual analysis of OBSM and OBSVC generalises the previous scheme for online bipartite matching given in [6]. We have overcome several technical hurdles along the way. First of all, convex programming duality is needed to address submodularity. The resultant program, however, involves exponentially many constraints and we must carefully update our primal variables in order to satisfy all of them.

Lastly, the previous primal-dual method is stated from the perspective of matching rather than vertex cover. For our problems, it turns out that the right approach is to start from vertex cover, which in turn brings us through a journey involving the Lovasz extension and submodular polytope.

We remark that our results on vertex cover hold even in the vertex-weighted setting. In other words, given nonnegative weight w on the vertices, our algorithms can be modified to handle objective functions of the forms $\sum_{u\in C\cap L} w_u + \sum_{v\in C\cap R} w_v$, $f(C\cap L) + \sum_{v\in C\cap R} w_v$ and $\sum_{u\in L} w_u y_u + \sum_{v\in R} w_v z_v$.

In the case of submodular matching, the constraint $x_v \leq 1$ (where $v \in R$) can be replaced by the more general $x_v \leq w_v$. Nevertheless, these extensions are not discussed here as they tend to add unnecessary complexity to the description and obscure the main theme.

3.4 Preliminaries

We need a few new notions and tools from combinatorial optimisation in order to address submodularity.

3.4.1 Lovasz extension of submodular functions

Given a submodular function $f : 2^L \longrightarrow \mathbb{R}$, the Lovasz extension $\hat{f} : [0,1]^L \longrightarrow \mathbb{R}$ is a continuous convex relaxation of f and is defined by

$$\hat{f}(\mathbf{y}) = \mathbb{E}_t[f(L(t))],$$

where $L(t) = \{u \in L : y_u \geq t\}$ and the expectation is taken over t chosen uniformly at random from $[0,1]$. It is easy to check one does have $f(S) = \hat{f}(I_S)$. Here I_S is the indicator variable for $S \subseteq L$.

While this is the standard definition of Lovasz extension, we make heavy use of an equivalent definition in this chapter. Given $\mathbf{y} \in [0,1]^L$, order the vertices of $L = \{1,2,...,n\}$ in such a way that $0 = y_0 \leq y_1 \leq y_2 \leq ... \leq y_n$. Let $Y_i = \{i, i+1, \ldots, n\}$ and $Y_{n+1} = \emptyset$. Then

$$\hat{f}(\mathbf{y}) = \sum_{i=1}^{n} (y_i - y_{i-1}) f(Y_i).$$

An immediate implication of this formulation is that by restricting $\mathbf{y} \in [0,1]^L$ to some fixed ordering $\sigma : \{1,2,...,|L|\} \longrightarrow L$, $\hat{f}(\mathbf{y})$ is a linear function. This property will be used in various places.

Finally, note that for monotone submodular function f, its Lovasz extension $\hat{f}(\mathbf{y})$ is monotonically increasing (in each coordinate).

Bar-chart representation

We introduce a bar chat interpretation of the Lovasz function. This representation plays an extremely important role in analyzing OBSVC and OBSM.

Given $\mathbf{y} \in [0,1]^L$, the bar chart representation of $\hat{f}(\mathbf{y})$ is the set

$$\bigcup_{t \in [0,1]} \{t\} \times [0, f(L(t))].$$

Notice that the bars are decreasing in height as t increases because f is monotone. If we order $L = \{1, 2, ..., n\}$ in such a way that $y_1 \leq y_2 \leq ... \leq y_n$. Using the notation in the last section, the bar chart representation consists of the bars

$$[0, y_1] \times [0, f(Y_1)], [y_1, y_2] \times [0, f(Y_2)], ..., [y_{n-1}, y_n] \times [0, f(Y_n)], [y_n, 1] \times [0, f(Y_{n+1})].$$

This is often a useful way to visualize the Lovasz extension $\hat{f}(\mathbf{y}) = \sum_{i=1}^{n}(y_i - y_{i-1})f(Y_i)$ as each term in the summand corresponds to precisely a bar in the bar-chart representation. In particular, $\hat{f}(\mathbf{y})$ is the area of the bar chart.

Strictly speaking, a bar can be empty (e.g. when $y_i = y_{i+1}$) but we shall implicitly disregard them hereafter as it does not affect our proofs in any way and would only make the notations more cumbersome.

Readers may find that it is sometimes more intuitive to view the bar chart as the function $t \mapsto f(L(t))$ for $t \in [0,1]$.

Figure 3-1 below gives an example of a bar chart representation with 5 bars, of which the last one corresponds to the empty set and has height $f(\emptyset) = 0$.

3.4.2 Convex program for bipartite submodular matching and vertex cover

Recall the notations $x_v := \sum_{u \in N(v)} x_{uv}$ and $x_S := \sum_{u \in S} x_u := \sum_{u \in S} \sum_{v \in N(u)} x_{uv}$. The primal and dual convex programs below are used in the primal-dual analysis of our algorithms for OBSM and OBSVC.

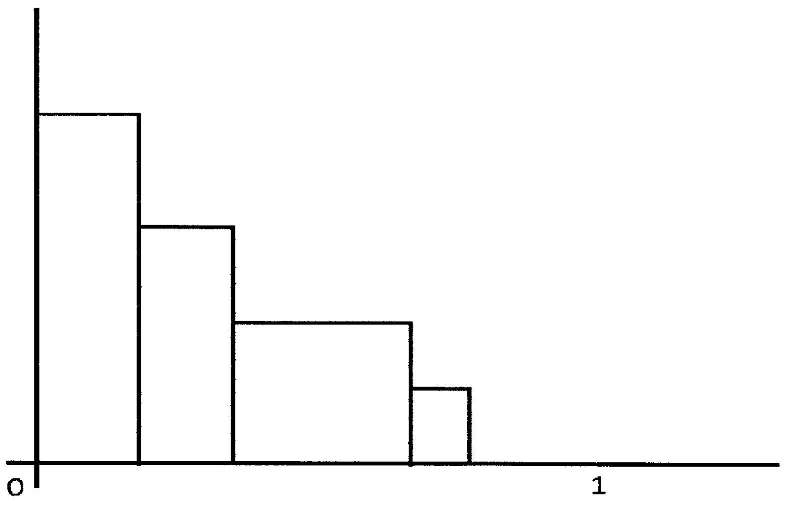

Figure 3-1: Bar chart representation of Lovasz extension

Primal:		Dual:	
	$\max \sum_{e \in E} x_e$		$\min \hat{f}(\mathbf{y}) + \sum_{v \in R} z_v$
s.t.	$x_v \leq 1, \forall v \in R$	s.t.	$y_u + z_v \geq 1, \forall (u,v) \in E$
	$x_S \leq f(S), \forall S \subseteq L$		$\mathbf{y}, \mathbf{z} \geq 0$
	$\mathbf{x} \geq 0$		

Readers who are familiar with polymatroid intersection should recognize that the primal is actually the polytope associated with the intersection of a partition matroid on R and a polymatroid on L defined by the submodular function f.

As in the usual primal-dual method, weak duality[2] is required in order to bound the size of the primal and dual solutions.

Lemma 3.4.1. *(weak duality) For any feasible solutions* $\mathbf{x}$ *and* $(\mathbf{y}, \mathbf{z})$ *to the primal and dual programs above, we have*

$$\sum_{e \in E} x_e \leq \hat{f}(\mathbf{y}) + \sum_{v \in R} z_v.$$

Proof. Let $L(t) = \{u \in L : y_u \geq t\}$ and define $R(1-t)$ analogously. For every $t \in [0,1]$, we claim that

$$C(t) := L(t) \cup R(1-t)$$

[2]In fact, even strong duality holds but this is not needed for our analysis.

is a vertex cover of G. Consider any edge $(u,v) \in E$. If $y_u \geq t$ then (u,v) is certainly covered. Otherwise, we have $z_v \geq 1 - y_u \geq 1 - t$ in which case $v \in R(1-t)$.

We are now ready to prove the lemma. For every vertex cover $C(t)$, since there are no edges between $L \backslash L(t)$ and $R \backslash R(1-t)$, we have

$$\sum_{e \in E} x_e \leq x_{L(t)} + \sum_{v \in R(1-t)} x_v \leq f(L(t)) + |R(1-t)|.$$

Our result then follows by noting that $\hat{f}(\mathbf{y}) = \mathbb{E}_t[f(L(t))]$ and $\sum_{v \in R} z_v = \mathbb{E}_t[|R(1-t)|]$, where the latter equality holds because each $v \in R$ is chosen to be in $R(1-t)$ with probability z_v. □

3.4.3 Rounding scheme for online bipartite vertex cover

For a fractional vertex cover $(\mathbf{y}, \mathbf{z})$ maintained by an online algorithm for OBSVCr, we can always round it to an integral solution by extending the rounding scheme in the last chapter. We first sample γ uniformly at random from $[0,1]$. Afterwards, for any vertex $u \in L$, we place u in the cover as long as $y_u \geq \gamma$. On the other hand, for any vertex $v \in R$, we place v in the cover when $z_v \geq 1 - \gamma$. It is not hard to verify that this rounding scheme indeed maintains a monotone vertex cover.

Now consider an algorithm for the online submodular bipartite vertex cover problem, with fractional solution $(\mathbf{y}, \mathbf{z})$. Let $C(\gamma)$ be the vertices in the integral cover given by the rounding with γ. The performance of our algorithm with this rounding scheme is

$$\mathbb{E}_\gamma[f(C(\gamma) \cap L)] + \mathbb{E}_\gamma[|C(\gamma) \cap R|] = \hat{f}(\mathbf{y}) + \sum_{v \in R} z_v.$$

Therefore, this rounding scheme does not incur a loss for OSBVC.

3.4.4 α and two integrals

Recall that we denote the optimal competitive ratio as

$$1 + \alpha := \frac{1}{1 - 1/e}$$

throughout this thesis. In our analyses, the following two definite integrals will often be useful.

$$\int_0^1 \frac{1-t}{t+\alpha}dt = \alpha, \qquad \int_0^1 \frac{1}{t+\alpha}dt = 1$$

3.5 The generalised waterfilling algorithm

As in OBVC, we first consider the fractional version of OBSVC. Our objective is then to minimize $\hat{f}(\mathbf{y}) + \sum_{v \in R} z_v$, which is a convex relaxation of $f(C \cap L) + |C \cap R|$. Our algorithm for fractional OBSVC can be converted to one for integral OBSVC by the rounding scheme in section 3.4.3.

Our algorithm for OBSVC is still greedy. The analysis, however, relies on a "two-dimensional" charging scheme in which the new additional regions of the bar chart representation (introduced in Section 3.4.1) are charged. We will see that the previous charging scheme for OBVC is a simplistic version of this more sophisticated scheme.

We also give an alternate primal-dual analysis of our algorithm which will imply a corresponding result for online bipartite submodular matching as a by-product. Our method builds on the previous scheme [6] for online bipartite matching (and effectively OBVC).

Nevertheless, unlike OBVC, the bar chart representation is crucial in getting a primal-dual analysis of the algorithm. In OBVC, one can carry out the primal-dual analysis from the perspective of either matching or vertex cover. Our primal-dual analysis of OBSVC and its dual OBSM instead relies inherently on vertex cover. There seems no natural variant of the algorithm which can be stated solely in terms of OBSM and makes no use of the structure of dual solution for OBSVC as guidance for updating the primal variables.

To the best of our knowledge, this is the first time that the primal-dual analysis is applied to an online problem which involves submodularity. The only closest example that we are aware of is [10], which involves *continuous* concave functions rather than *discrete* submodular functions. We hope that the primal-dual analysis method will emerge as a powerful tool for tackling submodular-flavored online problems.

Since the primal-dual analysis implies both the results on OBSM and OBSVC, it is tempting to question the value of the charging analysis. We stress that *both the charging-based and primal-dual analyses are of interest.* Our charging-based analysis is very clean.

It was precisely for this reason that we were able to establish the result on OBSVC first and "reverse-engineer" a primal-dual analysis which is, in contrast, somewhat complicated. In retrospect, without the charging analysis, we probably would not be able to come up with the primal-dual analysis or even to realize that these problems admit $1+\alpha$-approximation. Nonetheless, the primal-dual analysis is still important since it implies an interesting result on OBSM.

3.5.1 The algorithm

The design of our algorithm for OBSVC is in the same spirit as OBVC. In fact, the major modification needed is to replace $\sum_{u\in N(v)} \max\{a-y_u, 0\}$ by $\hat{f}(\mathbf{y}')-\hat{f}(\mathbf{y})$ as now the objective function on L is $f(\mathbf{y})$ rather than $\sum_{u\in L} y_u$.

```
Algorithm 4: GreedyAllocationSubmodular
  Input: $L$
  Initialize for each $u \in L$, $y_u = 0$;
  for each online vertex $v$ do
      $\max a \le 1$ s.t. $(1-a) + \hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y}) \le 1+\alpha$, where $y'_u = \max\{y_u, a\}$ for
      $u \in N(v)$ and $y'_u = y_u$ for other $u$;
      Let $X = \{u \in N(v) \mid y_u < a\}$;
      For each $u \in X$, $y_u \leftarrow a$;
      $z_v \leftarrow 1-a$;
  end
```

The analysis of *GreedyAllocationSubmodular* makes extensive use of the bar chart representation introduced in Section 3.4.1. It is thus helpful to interpret our algorithm in terms of the bar chart. This will hopefully also make the change in the Lovasz extension $\hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y})$ more intuitive and easier to visualize.

Bar chart interpretation of the algorithm

We take a closer look at how the bar chart changes after processing an online vertex. Recall that

$$L(t) = \{u \in L | y_u \ge t\}$$

and $f(L(t))$ is the height of the bar chart at t. First of all, observe that for the bars at $t \le a$, the height changes from $f(L(t))$ to $f(L(t) \cup X)$ since the potential of the vertices from X increased to a and no other vertex increased in potential. As a result, the bar at $t > a$ remains at the same height.

With this observation in mind, we see that a new rectangular region (possibly empty) of height $f(L(t) \cup X) - f(L(t))$ is added to the top of the bar at $t < a$. Moreover, the bar at $t = a$ is effectively split into two[3]: the right one has the same height $f(L(a))$ whereas the left one has a larger height $f(L(a) \cup X) \geq f(L(a))$.

Our charging scheme in the next section makes critical use of the following two properties:

- All the new rectangular regions are added to the bars at $t \leq a$.

- The total area of the new rectangular regions is $\hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y})$.

The mechanism by which $1 - a$ is charged to $u \in X$ lies in the heart of the previous charging scheme for OBVC. This idea does not quite work anymore as our objective function is submodular rather than modular. The key insight in our new analysis is to charge $1 - a$ to the new rectangular regions of the bar chart. This is in contrast to the previous scheme which charges to individual $u \in X$.

Our analysis in a nutshell is a careful study of figure 3-2. The red regions are the new rectangles added to the bar chart. Note that the first three bars increased in height with the third one being split into two at a. All of the new regions are found at $t \leq a$. It is no coincidence that the height of the red rectangles decreases along the horizontal axis. Although not needed for the proof, it is instructive to check that this phenomenon is an artifact of submodularity and monotonicity.

In the next section, we propose a charging scheme in which the red new regions are charged to compensate for $z_v = 1 - a$.

Finally, we remark that the bar chart is just a pictorial representation of the Lovasz extension. We could have carried out the analysis without it at the expense of added notational complexity. It is for the same reason that various degenerate cases are deemphasized (e.g. we speak of the bar at t but t can happen to be at the boundary between two consecutive bars).

[3]It is possible to have the degenerate case where a coincides with the boundary of a bar.

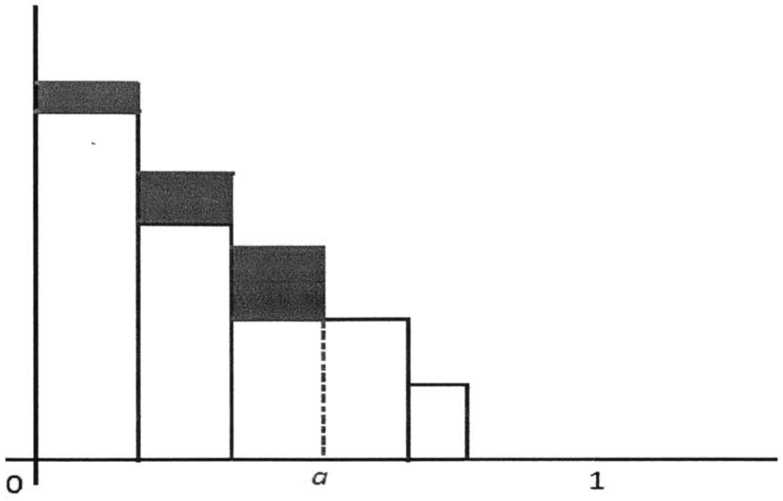

Figure 3-2: Bar chart being split at a

3.5.2 Charging-based analysis

When an online vertex not in the optimal cover is processed, we will charge all the potential used on this vertex to its neighbors, which must in the optimal cover. More concretely, we charge the cost to the bar chart representing $\hat{f}(\cdot)$. For each point (x, y) of the bar chart, the charging density is $\frac{1-x}{x+\alpha}$. We first show that such a charging density is sufficient to account for the potential of the online vertex.

Lemma 3.5.1. *Let B and B' be the bar charts before and after processing online vertex v. Let a be the final water-level on the neighboring vertices of v after processing v. We have*

$$\int_{B' \setminus B} \frac{1-x}{x+\alpha} \mathrm{d}A \geq 1 - a.$$

Proof. The main idea is to charge $1 - a$ to the new region of the bar chart. From the discussion in the last section, all the new regions have x-coordinates at most a. Therefore, we have

$$\int_{B' \setminus B} \frac{1-x}{x+\alpha} \mathrm{d}A \geq \frac{1-a}{a+\alpha} \int_{B' \setminus B} \mathrm{d}A = \frac{1-a}{a+\alpha} \cdot (\hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y})) = 1 - a, \tag{3.1}$$

where the last equality obviously holds if $a = 1$. If $a < 1$, then we must have not exhausted all of our resources $1 + \alpha$ (otherwise a would be larger) and hence we have $\hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y}) = a + \alpha$. □

Now we show that the total charges to the left vertices by online vertices not in the optimal cover C^* are at most $\alpha \cdot f(L \cap C^*)$.

Lemma 3.5.2. *The total charges received from online vertices $R \setminus C^*$ are at most $\alpha \cdot f(L \cap C^*)$.*

Proof. Let B^* be the union of the new regions in the bar chart generated by processing online vertices $R \setminus C^*$. Therefore, the total charges are

$$\int_{B^*} \frac{1-x}{x+\alpha} \mathrm{d}A.$$

For $t \in [0,1]$, let $B^*(t)$ be the intersection of B^* with the line $x = t$. We have

$$\int_{B^*} \frac{1-x}{x+\alpha} \mathrm{d}A = \int_0^1 \int_{B^*(x)} \frac{1-x}{x+\alpha} \mathrm{d}y \mathrm{d}x \leq \int_0^1 \frac{1-x}{x+\alpha} \mathrm{d}x \sup_{t \in [0,1]} \int_{B^*(t)} \mathrm{d}y = \alpha \cdot \sup_{t \in [0,1]} \int_{B^*(t)} \mathrm{d}y.$$

It is then sufficient to show that for $t \in [0,1]$,

$$\int_{B^*(t)} \mathrm{d}y \leq f(L \cap C^*).$$

Notice that $\int_{B^*(t)} \mathrm{d}y$ is the total heights of regions added to the bar chart at $x = t$ when processing online vertices in $R \setminus C^*$.

Although the argument below looks somewhat technical, the key idea is simple. Suppose that all of the vertices in $L \backslash C^*$ are removed, i.e. $L \subseteq C^*$. Now the height of the bar chart is at most $f(L) = f(C^* \cap L)$ so our claim is clear. If we add back $L \backslash C^*$, recall that we care only about the rectangles added for $v \notin C^*$. The height of the additional rectangle is just the marginal difference, which cannot be worse than before because of diminishing marginal return. We formalize this below.

Let $L_i(t)$ be the set $L(t) = \{u \in L \mid y_u \geq t\}$ *after* processing the i-th online vertex v_i. Since y_u can never decrease for all $u \in L$, we have

$$L_0(t) \subseteq L_1(t) \subseteq \cdots \subseteq L_{|R|}(t).$$

Furthermore, $L_i(t) \backslash L_{i-1}(t) \subseteq N(v_i)$ since only y_u for $u \in N(v_i)$ can increase when processing v_i. In particular, for $v_i \notin C^*$ we have that $L_i(t) \setminus L_{i-1}(t) \subseteq C^*$ as $v_i \notin C^*$

implies $N(v_i) \subseteq C^*$. Submodularity and $L_i(t) \setminus L_{i-1}(t) \subseteq C^*$ for $v_i \notin C^*$ give

$$f(L_i(t)) - f(L_{i-1}(t)) \leq f(L_i(t) \cap C^*) - f(L_{i-1}(t) \cap C^*). \tag{3.2}$$

Finally, when processing v_i, the height of the new rectangular region[4] at t is precisely $f(L_i(t)) - f(L_{i-1}(t))$. Now the sum of the height of the rectangular regions at t added when processing $v_i \notin C^*$ is

$$\begin{aligned}
\int_{B^*(t)} \mathrm{d}y &= \sum_{v_i \in R \setminus C^*} f(L_i(t)) - f(L_{i-1}(t)) \\
&\leq \sum_{v_i \in R \setminus C^*} f(L_i(t) \cap C^*) - f(L_{i-1}(t) \cap C^*) && \text{(submodularity)} \\
&\leq \sum_{i=1}^{|R|} f(L_i(t) \cap C^*) - f(L_{i-1}(t) \cap C^*) && \text{(monotonicity)} \\
&= f(L_{|R|}(t) \cap C^*) - f(L_0(t) \cap C^*) \leq f(L \cap C^*).
\end{aligned}$$

Here the last inequality follows from monotonicity and non-negativeness of f.

□

Lemma 3.5.3. *The total resources used in processing online vertices $R \setminus C^*$ are at most $(1+\alpha) \cdot f(L \cap C^*)$.*

Proof. For the i-th online vertex $v_i \in R$, we define $\mathbf{y}_i$ to be the vector of potentials on L *after* processing v_i. Then, by our algorithm and the last lemma, the total resources used in processing $R \setminus C^*$ are at most

$$\alpha \cdot f(L \cap C^*) + \sum_{v_i \in R \setminus C^*} \hat{f}(\mathbf{y}_i) - \hat{f}(\mathbf{y}_{i-1}).$$

Since for $v_i \in R \setminus C^*$, $L_i(t) \setminus L_{i-1}(t) \subseteq C^*$ for any $t \in [0,1]$, where $L_i(t)$ is defined as

[4]Of course, it is possible that no region is added in which case this is still okay as $f(L_i(t)) = f(L_{i-1}(t))$.

before. By Eqn.(3.2) and the definition of $\hat{f}(\cdot)$, we have

$$\begin{aligned}\sum_{v_i\in R\setminus C^*} \hat{f}(\mathbf{y}_i) - \hat{f}(\mathbf{y}_{i-1}) &\le \sum_{v_i\in R\setminus C^*} \hat{f}(\mathbf{y}_i|_{L\cap C^*}) - \hat{f}(\mathbf{y}_{i-1}|_{L\cap C^*}) \\ &\le \sum_{v_i\in R} \hat{f}(\mathbf{y}_i|_{L\cap C^*}) - \hat{f}(\mathbf{y}_{i-1}|_{L\cap C^*}) \\ &= \hat{f}(\mathbf{y}_{|R|}|_{L\cap C^*}) - \hat{f}(0|_{L\cap C^*}) \le f(L\cap C^*),\end{aligned}$$

where $\mathbf{y}_i|_{L\cap C^*}$ restricts the vector $\mathbf{y}_i$ to the vertices $L \cap C^*$ by setting the other entries to 0. This concludes the proof. $\square$

Therefore, our algorithm uses resources at most $(1+\alpha)\cdot f(L\cap C^*)$ when processing vertices in $R\setminus C^*$. On the other hand, it uses resources at most $(1+\alpha)\cdot|R\cap C^*|$ for other online vertices as processing each of them increased the total potentials by at most $1+\alpha$. Our algorithm is thus $1+\alpha$-competitive for the fractional online bipartite submodular vertex cover problem. Since we can always round a fractional solution to a randomized integral solution (section 3.4.3), we have the following theorem.

Theorem 3.5.4. *There exists an optimal $1+\alpha$-competitive algorithm for the online submodular bipartite integral vertex cover problem.*

3.5.3 Primal-dual analysis

We first review the key ingredients used in the original primal-dual analysis of online bipartite matching in [6], which largely consists of two steps:

- **Employs such constraints as $x_u = g(y_u)$ (or $x_u \le g(y_u)$) for some suitable increasing function g.** The motivation for doing this is to enforce some correlation between the primal and dual variables so that, for instance, when x_u is small, y_u is not too big which allows room to pay for the future increase in x_u.

- **Relates the size of the primal and dual solutions by $\sum(g(a)-g(y_u)) \approx c(1-a+\sum(a-y_u))$ for some constant c.** As in the usual primal-dual method, this is essential for bounding the size of the solution via weak duality.

This scheme depends crucially on the fact that the cost function is modular. For submodular cost functions, one may try to imitate that by using constraints like $x_S \le f(S)g(h(\mathbf{y}|S))$

($\mathbf{y}|S$ is the vector restricted to S), where g is the same as before and $h : [0,1]^S \longrightarrow [0,1]$ is some suitable function.

Considering the Lovasz extension, the most natural choice is probably $h(\mathbf{y}|S) = \min_{u \in S} y_u$. But this is fundamentally flawed as one may have a very small y_u with other $y_{u'} = 1$. It turns out that, perhaps somewhat counter-intuitively, the correct function is $h(\mathbf{y}|S) = \max_{u \in S} y_u$.

Even more surprisingly, the constraint $x_S \leq f(S)g(h(\mathbf{y}|S))$ alone is not enough to relate the cost of the primal and dual solutions. Recall that $\hat{f}(\mathbf{y}) = \sum f(Y_i)(y_i - y_{i-1})$ for a fixed ordering of y. Thus one might hope to consider $S = Y_1, Y_2, ...$ in order to relate the increment in the size of the primal and dual solutions. Unfortunately, this does not work as the ordering of $\mathbf{y}$ typically changes over the execution of the algorithm.

To rescue this, we turn to the bar chart representation again. Instead of one *global* ordering, a *local* ordering is imposed on each bar of the bar chart. More precisely, for a bar at t, we maintain an ordering σ_t of its existing vertices $L(t)$. When $L(t)$ increases, we extend the current ordering by arbitrarily appending the new vertices to its end. We formalize our ideas in the rest of this section.

To simplify our notation, we view x_u as a function on $[0,1]$ and the value of x_u[5] is

$$x_u = \int_0^1 x_u(t)dt.$$

This perspective will be useful when we analyze our algorithm using the bar chart representation (which can be seen as a function on $[0,1]$). The x_u produced by the algorithm will be a piecewise constant function. Conceptually, $\int_0^1 x_u(t)dt$ aggregates over the contribution of each bar to x_u.

At the first glance, our primal update seems somewhat convoluted. The underlying philosophy is nevertheless much simpler. Before proceeding to the analysis, we first unpack the details of the algorithm along with some simple observations.

First of all, in our algorithm we focus on $x_u(t)$ rather than x_{uv}. This is more convenient in the analysis since what matters is the extent to which u is matched (recall: $x_S \leq f(S)$) but not which edge is assigned to u. Thus in the algorithm, we determine only how much x_u increases and retroactively what x_{uv} is.

[5]Here we abuse notations by using x_u for both the primal variable maintained by the algorithm as well as a function on $[0,1]$.

Algorithm 5: *GreedyAllocationSubmodularPD*

```
Input: $L$
Initialize for each $u \in L$, $y_u = 0, x_u(t) = 0 \forall t \in [0,1]$;
for each online vertex $v$ do
    Dual:;
    $\max a \leq 1$ s.t. $(1-a) + \hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y}) \leq 1+\alpha$, where $y'_u = \max\{y_u, a\}$ for
    $u \in N(v)$ and $y'_u = y_u$ for other $u$;
    Let $X = \{u \in N(v) \mid y_u < a\}$;
    For each $u \in X$, $y_u \leftarrow a$;
    $z_v \leftarrow 1 - a$;
    Primal:;
    for each bar of the bar chart at $[p,q] \ni t$ with a new rectangular region
    $[p,q] \times [f(L(t)), f(L(t) \cup X)]$ do
        Extend the current ordering $\sigma_t$ of $L(t)$ to $L(t) \cup X$ by appending $X \backslash L(t)$
        arbitrarily to the end $\sigma_t(|L(t)|+1), ..., \sigma_t(|L(t) \cup X|)$;
        For $t \in (p,q)$ and $|L(t)|+1 \leq k \leq |L(t) \cup X|$, set
            $$x_{\sigma_t(k)}(t) = \left( f\left(\bigcup_{i=1}^{k} \sigma_t(i)\right) - f\left(\bigcup_{i=1}^{k-1} \sigma_t(i)\right)\right) \frac{1}{a+\alpha}$$
    end
    For each $u \in N(v)$, set $x_{uv}$ to be the increment of $x_u = \int_0^1 x_u(t)dt$ in this
    iteration;
end
```

Moreover, note that since each vertex can be added at most once to $L(t)$, $x_u(t)$ can increase at most once and this increment will be from $x_u(t) = 0$ to $x_u(t) = \left(f\left(\bigcup_{i=1}^{k} \sigma_t(i)\right) - f\left(\bigcup_{i=1}^{k-1} \sigma_t(i)\right)\right) \frac{1}{a+\alpha}$ where $u = \sigma_t(k)$.

Lastly, we emphasize the role of the ordering σ_t. This is the key ingredient that makes the analysis possible. See Proposition 3.5.5 and Lemma 3.5.6 for more details.

We are now ready to analyze the algorithm. There are three major components:

- (feasibility) $x_S \leq f(S)$ for all $S \subseteq L$.
- (feasibility) $x_v \leq 1$, i.e. the total increment of all x_u in each iteration is at most 1.
- (competitiveness) $\triangle D = (1+\alpha)\triangle P$, where $\triangle D$ and $\triangle P$ are the increments in the size of the dual and primal solutions respectively.

Once the above have been established, we can conclude that our algorithm is correct and achieves a competitive ratio of $1+\alpha$ via weak duality.

The following well-known property of the sumbodular polytope will be used in the proof:

Proposition 3.5.5. *Let $f : \Omega \longrightarrow \mathbb{R}_{\geq 0}$ be a nonnegative monotone submodular function and fix an ordering $\sigma : \{1, 2, ..., |\Omega|\} \longrightarrow \Omega$. Then the solution*

$$x_{\sigma(k)} = f\left(\bigcup_{i=1}^{k} \sigma(i)\right) - f\left(\bigcup_{i=1}^{k-1} \sigma(i)\right)$$

satisfies the inequalities $x_S \leq f(S) \forall S \subseteq \Omega$.

Proof. Let $T_j = \cup_{i=1}^{j} \sigma(i)$. Then $x_{\sigma(k)} = f(T_k) - f(T_{k-1})$. Let $S = \{s_1, s_2, \ldots, s_\ell\}$. We have

$$\begin{aligned} x_S = \sum_{i=1}^{\ell} x_{s_i} &= \sum_{i=1}^{\ell} f(T_{\sigma^{-1}(s_i)}) - f(T_{\sigma^{-1}(s_i)-1}) \\ &\leq \sum_{i=1}^{\ell} f(T_{\sigma^{-1}(s_i)} \cap S) - f(T_{\sigma^{-1}(s_i)-1} \cap S) \\ &\leq \sum_{i=1}^{|\Omega|} f(T_i \cap S) - f(T_{i-1} \cap S) = f(S) - f(\emptyset) \leq f(S), \end{aligned}$$

where the first and second inequalities follow from submodularity and monotonicity. □

Lemma 3.5.6. *In GreedyAllocationSubmodularPD, we have $x_S \leq f(S)$ for all $S \subseteq L$.*

Proof. We first show that

$$x_S(t) \leq \frac{f(S)}{t + \alpha}$$

for each t.

Consider any $\sigma_t(k) \in S$ for which $x_{\sigma_t(k)}(t) > 0$. Then we must have set

$$x_{\sigma_t(k)}(t) = \left(f\left(\bigcup_{i=1}^{k} \sigma_t(i)\right) - f\left(\bigcup_{i=1}^{k-1} \sigma_t(i)\right)\right) \frac{1}{a + \alpha} \leq \left(f\left(\bigcup_{i=1}^{k} \sigma_t(i)\right) - f\left(\bigcup_{i=1}^{k-1} \sigma_t(i)\right)\right) \frac{1}{t + \alpha},$$

where the inequality follows from the fact that only the bars on the left of a increase in height and hence $t \leq a$.

Now by Proposition 3.5.5, we have $x_S(t) \leq \frac{f(S)}{t+\alpha}$. Our desired result thus follows:

$$x_S = \int_0^1 x_S(t) dt \leq \int_0^1 \frac{f(S)}{t + \alpha} dt = f(S).$$

□

Lemma 3.5.7. *For each iteration of the algorithm, the increases in the size of the primal and dual solutions satisfy*

$$\triangle D = (1+\alpha)\triangle P.$$

Proof. Recall that $\triangle D = \hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y}) + 1 - a$ and $\hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y})$ is the total area of the new rectangular regions needed to the bar chart.

On the other hand, $\triangle P$ is the sum of the increments of all x_u. We restrict our attention to each bar via the following:

$$\begin{aligned}
\sum_{u\in X} \Delta x_u &= \int_0^1 \sum_{u\in X\setminus L(t)} x_u(t)dt = \int_0^1 \sum_{k=|L(t)|+1}^{|L(t)\cup X|} x_{\sigma_t(k)}(t)\mathrm{d}t \\
&= \int_0^1 \sum_{k=|L(t)|+1}^{|L(t)\cup X|} \left(f\left(\bigcup_{i=1}^{k} \sigma_t(i)\right) - f\left(\bigcup_{i=1}^{k-1} \sigma_t(i)\right)\right) \frac{1}{a+\alpha}\mathrm{d}t \\
&= \int_0^1 \left(f(L(t)\cup X) - f(L(t))\right) \frac{1}{a+\alpha}\mathrm{d}t \\
&= \frac{\hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y})}{a+\alpha},
\end{aligned}$$

where the last equality holds as $f(L(t)\cup X) - f(L(t))$ is the height of the new rectangular region at t.

In other words,

$$\triangle P = \frac{\hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y})}{a+\alpha}.$$

The rest of the proof is now easy. The case $a = 1$ is trivial as $\triangle D = \hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y})$.

If $a < 1$, then we must have exhausted all of our resources $1+\alpha$. Hence we have $\triangle D = 1 + \alpha$ and $\hat{f}(\mathbf{y}') - \hat{f}(\mathbf{y}) = a + \alpha$. This gives $\triangle P = 1$. □

Corollary 3.5.8. $x_v \leq 1$, *i.e. the total increment of all* x_u *in each iteration is at most 1.*

Proof. The dual solution can increase by at most $1+\alpha$ and hence $\triangle P$, which is just x_v, is at most 1 by Lemma 3.5.7. □

Combining all the pieces, we obtain our main theorem.

Theorem 3.5.9. *Our algorithm is* $1 - 1/e$*-competitive for online bipartite submodular matching and* $1+\alpha$*-competitive for online bipartite submodular vertex cover.*

Proof. By Lemma 3.5.7, we always have $D = (1+\alpha)\cdot P$. By weak duality (see Lemma 3.4.1), we can bound P and D against the optimal solutions D^* and P^* as follows,

$$P^* \le D = (1+\alpha)\cdot P \le (1+\alpha)\cdot D^*.$$

This shows that $P \ge (1-1/e)P^*$ and $D \le (1+\alpha)D^*$, as desired. $\square$

Finally, we remark that we do have $x_S \le f(S)\frac{\max_{u\in S} y_u + \int_0^{y_u} \frac{1-t}{t+\alpha} dt}{1+\alpha}$ (i.e. $x_S \le f(S) g(\max_{u\in S} y_u)$) as mentioned earlier. Although not needed for the proof, it has served as a useful inspiration when we were developing this primal-dual analysis.

Chapter 4

Online Vertex Cover and Matching: when all Vertices are Online

In this chapter, we study the online vertex cover problem in bipartite and general graphs where all vertices are online. In this new setting, upon the arrival of an online vertex v, only the edges incident to the previously arrived vertices are revealed. The algorithm is still required to maintain a (fractional or integral) *monotone* vertex cover for the revealed subgraph at all time. In particular, no vertices can be removed from the cover once added. The objective is to minimize the size of the final vertex cover.

Our main result is a 1.901-competitive algorithm for this problem. This employs yet another generalisation of the charging scheme presented in chapter 2. In the case of bipartite graphs, the rounding scheme in section 2.3.1 is still applicable.

Moreover, as in the last chapter, we give an alternate primal-dual analysis for our algorithm, which in turns implies a $1/1.901 \approx 0.526$-competitive algorithm for online fractional matching. In particular, our algorithm improves upon the result of the fractional version of the online edge-selection problem in Blum et. al. [4]. This addresses a limitation inherent to online bipartite matching and almost all of its variants studied in the literature, which share the common feature that vertices of only one side of the bipartite graph arrive online. While this property indeed holds in many applications, it does not necessarily reflect the reality in general. We exemplify this by the following application:

Online market clearing In a commodity market, buyers and sellers are represented by the left and right vertices. An edge between a buyer and seller indicates that the price that the buyer is willing offer is higher than the price at which the seller is willing to take. The objective is to maximize the number of trades, or the size of the matching. In this problem, both the buyers and sellers arrive and leave online continuously.

The greedy algorithm There is a simple well-known greedy algorithm for online matching and vertex cover in general graphs. As each vertex arrives, we match it to an arbitrary unmatched neighbor (if any) and put both of them into the vertex cover. It is easy to show that this algorithm is 1/2-competitive for online matching and 2-competitive for online vertex cover.

The greedy algorithm for the vertex cover problem is optimal assuming the Unique Game Conjecture even in the offline setting [23]. Thus without disproving the conjecture, there is no hope of doing better than 2 if we restrict ourselves to integral vertex covers in general graphs. For the other problems studied in this chapter, e.g. matching and vertex cover in bipartite graphs and matching in general graphs, no known algorithm beats the greedy algorithm in the online setting.

We present the first successful attempt in breaking the barrier of 2 (or 1/2) achieved by the greedy algorithm. In the fractional setting, our algorithm is 1.901-competitive (against the minimum fractional cover) for online vertex cover and $\frac{1}{1.901} \approx 0.526$-competitive (against the maximum fractional matching) for online matching in general graphs. It is possible to convert the fractional algorithm to a randomized integral algorithm for online vertex cover in bipartite graphs via the rounding scheme in section 2.3.1. On the other hand, it is not clear whether it is possible to round our algorithm or its variants for online matching in either bipartite graphs or general graphs.

We stress that the fractional setting is still of interest for two reasons:

- As well-articulated in [6], some commodities are divisible and hence should be modeled as fractional matchings. In fact, for divisible commodities one would even prefer a fractional matching assignment since the maximum fractional matching may be larger than the maximum integral matching in general graphs. Thus a c-competitive algorithm against fractional matching would be preferable to a c-competitive algorithm against integral matching.

- Our 0.526-competitive algorithm for fractional matching suggests that it may be possible to beat the greedy algorithm for online integral matching in the oblivious adversarial model.

4.1 Related work

Besides the work on ski rental and online matching mentioned in previous chapters, analyzing greedy algorithms for maximum matching in the offline setting is another related research area. Aronson et al. [2] showed that a randomized greedy algorithm is a $\frac{1}{2}+\frac{1}{400,000}$-approximation. The factor was recently improved to $\frac{1}{2}+\frac{1}{256}$ [25]. A new greedy algorithm with better ratio was presented in [15]. Our 0.526-competitive algorithm for online fractional matching complements these results.

4.2 Problem statement

In the online setting, the vertices of G arrive one at a time in an order determined by the adversary. When an online vertex v arrives, all of its edges incident to the *previously arrived* vertices are revealed. We denote the set of arrived vertices by $T \subset V$ and $G(T)$ is the subgraph of G induced by T.

An algorithm for online integral matching maintains a monotone matching M. As each vertex v arrives, it must decide if (u, v) should be added to M for some previously unmatched $u \in N(v) \cap T$, where $N(v)$ is neighbors of v in G. No edge can be removed from M. The objective is to maximize the size of the final matching M. For online fractional matching, a fractional matching $\mathbf{x}$ for $G(T)$ is maintained and at each step, x_{uv} must be initialized for $u \in N(v) \cap T$ so that $\mathbf{x}$ remains a fractional matching. The objective is to maximize the final $\sum_{e \in E} x_e$.

An algorithm for online integral vertex cover maintains a monotone vertex cover C. As each vertex v arrives, it must insert a subset of $\{v\} \cup N(v) \backslash C$ into C so that it remains a vertex cover. No vertex can be removed from C. The objective is to minimize the size of the final cover C. For online fractional vertex cover, a fractional vertex cover $\mathbf{y}$ for $G(T)$ is maintained and at each step, we must initialize y_v and possibly increase some y_u for $u \in T$ so that $\mathbf{y}$ remains a fractional vertex cover. The objective is to minimize the final $\sum_{v \in V} y_v$.

Weighted vertex cover and b-matching. Our results can be generalized to cases of weighted vertex cover and b-matching. For vertex cover, the objective function becomes $\sum_{v \in C} w_v$ (integral) or $\sum_{v \in V} w_v y_v$ (fractional), where $w_v \geq 0$ are weights on the vertices that are revealed to the algorithm when v arrives.

For b-matching, the only difference is that each vertex can be matched up to $w_v \in \mathbb{N}$ times instead of just 1 (integral) or the constraint $x_v := \sum_{u \in N(v)} x_{uv} \leq w_v$, where $w_v \geq 0$, replaces $x_v \leq 1$ (fractional). See below the LP formulation of the two problems for the fractional solution.

4.3 Our result and technique

Our main result is the first nontrivial algorithm for the online vertex cover problem in general graphs.

- A 1.901-competitive algorithm for online fractional vertex cover in general graphs.

We stress that the fact that our result holds only for the *fractional* version of online vertex cover in general graphs is reasonable. In fact, even in the offline setting, the best known approximation algorithm for minimum vertex cover is just the simple 2-approximate greedy algorithm. Getting anything better than 2 would disprove the Unique Game Conjecture even in the offline setting [23] and have profound implications to the theory of approximability.

Our algorithms can also be analyzed in the prime-dual framework [6]. As by-products, we obtain dual results on the maximum matching as follows:

- A 0.526-competitive algorithm for online fractional matching in general graphs. This improves the result on the online edge-selection problem studied in [4].

All of these results also hold in the vertex-weighted setting (for vertex cover) and the b-matching setting. Moreover, it is easy to verify that the same rounding scheme in section 2.3.1 can still be used to convert essentially any algorithm for online *fractional* vertex cover to an algorithm for online *integral* vertex cover in the case of bipartite graphs with the same (expected) performance.

We note that our results can be combined with the techniques for handling edge-weighted OBVC in chapter 2 to yield algorithms for the edge-weighted setting in general graphs.

On the hardness side, we establish the following lower bound (for vertex cover) and upper bound (for matching) on the competitive ratios. Notice that these bounds also apply to the more general integral version of the problems.

- A lower bound of $1 + \sqrt{\frac{1}{2}\left(1 + \frac{1}{e^2}\right)} \approx 1.753$ for the online *fractional* vertex cover problem in bipartite graphs.
- An upper bound of 0.625 for the online *fractional* matching problem in bipartite graphs.

Main ingredients. Recall that in our previous charging scheme, for an online vertex in the optimal cover, we charge all the water used in processing this vertex to itself. For an online vertex not in the optimal cover, we charge the water spent on the online vertex to its neighbors, which must be in the optimal cover. In particular, in the bipartite graph case with one-sided online vertices, an online vertex in the optimal cover will take care of the cost processing itself wherea an offline vertex in the optimal cover is responsible for the charge from its online neighbors.

In generalizing the charging scheme to the two-sided online bipartite and the general graph cases, a vertex must take care of both the cost in processing itself and the charges received from future neighbors. In such generalizations, we cannot use a fixed amount of water in processing each vertex. A key insight behind our algorithm is that the amount of water used should be related to the actual final water level. In other words, for a final water level y, the amount of water used should be $f(y)$ for some allocation function $f(\cdot)$. By extending our previous charging scheme, the competitive ratio of our new water-filling algorithm for the online fractional vertex cover problem in general graphs can be written as a function of $f(\cdot)$. We also derive the constraints which $f(\cdot)$ must satisfy in order to make the analysis work.

As a result, we are left with a non-conventional minimax optimization problem. (See Eqn.(4.1).) The most exciting part, however, is that we can actually solve this optimization problem *optimally*.[1] The optimal allocation function in Theorem 4.4.6 implies a competitive ratio of 1.901 for the online fractional vertex cover problem in general graphs. Our primal-dual analysis for the online fractional matching problem in general graphs is obtained by

[1]Our solution is optimal in *our framework*. It may not be optimal for the online fractional vertex cover problem.

reverse-engineering the charging-based analysis.

LPs for fractional vertex cover and matching

Primal:		Dual:	
	$\max \sum_{e\in E} x_e$		$\min \sum_{v\in V} y_v$
s.t.	$x_v := \sum_{u\in N(v)} x_{uv} \leq 1,\ \forall v \in V$	s.t.	$y_u + y_v \geq 1,\ \forall (u,v) \in E$
	$x \geq 0$		$y \geq 0$

The matching and vertex cover LPs are called the primal and dual LPs, respectively. By weak duality, we have

$$\sum_{e\in E} x_e \leq \sum_{v\in V} y_v$$

for any feasible fractional matching $\mathbf{x}$ and vertex cover $\mathbf{y}$.

4.4 Online fractional vertex cover in general graphs

For each vertex v, we maintain a non-decreasing cover potential y_v which is initialized to 0. When an online vertex v arrives, the edges between v and $N(v) \cap T$ are revealed. In order to cover these new edges, we must increase the potential of v and its neighbors. Suppose that we set $y_v = 1 - y$ after processing v. To maintain a feasible vertex cover, we must increase any $y_u < y$ for $u \in N(v)$ to y. We call y the *water level.*

The trick here lies in how y is determined. We consider a simple scheme in which y is related to the total potential increment of $N(v)$. More precisely, we require that the total potential increment $\sum_{u\in N(v): y_u < y}(y - y_u)$ be at most $f(y)$, where f is a positive continuous function on $[0, 1]$.

We begin with one important observation that this new algorithm reduces to the one for OBVC in chapter 2 for the allocation function $f(y) = y + \alpha$. In this section, we present a generalised charging scheme for general graphs. Before getting into the details, we revisit the analysis in chapter 2 to gain some insights which will be helpful to tackle the general graph version of the problem. In our charging argument, each vertex in $L \cap C^*$ is responsible for the charges from its neighbors. On the other hand, a vertex in $R \cap C^*$ is only responsible for the potential increment when processing itself. However, if vertices in both L and R are online, an online vertex $v \in C^*$ should be responsible for the potential used to process it when it arrives as well as the charges from its future neighbors.

Algorithm 6: *GreedyAllocation* with allocation function $f(\cdot)$

```
Input: Online graph $G = (V, E)$ with offline vertices $U \subset V$
Output: A fractional vertex cover of $G$
Initialize for each $u \in U$, $y_u = 0$;
Let $T$ be the set of known vertices. Initialize $T = U$;
for each online vertex $v$ do
    Maximize $y \leq 1$, s.t., $\sum_{u \in N(v) \cap T} \max\{y - y_u, 0\} \leq f(y)$;
    For each $u \in N(v) \cap T$, $y_u \leftarrow \max\{y_u, y\}$;
    $y_v \leftarrow 1 - y$;
    $T \leftarrow T \cup \{v\}$;
end
Output $\{y_v\}$ for all $v \in V$;
```

The following lemma adapts the previous charging method in chapter 2 to an arbitrary allocation function.

Lemma 4.4.1. *Let $f : [0,1] \longrightarrow \mathbb{R}_+$ be continuous such that $\frac{1-t}{f(t)}$ is decreasing, and $F(x) = \int_0^x \frac{1-t}{f(t)} \mathrm{d}t$. If $\sum_{u \in X}(y - y_u) = f(y)$ for some set X and $y \geq y_u$ for $u \in X$, then*

$$1 - y \leq \sum_{u \in X} \left(F(y) - F(y_u)\right).$$

Proof. We have the following

$$\begin{aligned} \sum_{u \in X} (F(y) - F(y_u)) &= \sum_{u \in X} \int_{y_u}^{y} \frac{1-t}{f(t)} \mathrm{d}t \\ &\geq \sum_{u \in X} (y - y_u) \frac{1-y}{f(y)} = 1 - y, \end{aligned}$$

where the inequality above holds as $\frac{1-t}{f(t)}$ is decreasing. □

Let $f(x)$ be a general allocation function such that $\frac{1-t}{f(t)}$ is decreasing. Informally, if the water level when processing v is $y < 1$, i.e. the initial potential of v is $1-y$, we use potential $f(y)$ on v's neighbors and $1 - y$ on v itself. Afterwards, v will take charges from its future neighbors. Notice that v's potential will grow from $1 - y$ to at most 1. By Lemma 4.4.1, v will take charges at most $\int_{1-y}^{1} \frac{1-t}{f(t)} \mathrm{d}t$. Putting the two pieces together, the total charges to each $v \in C^*$ and hence the competitive ratio are at most

$$\beta(f) = \max_{z \in [0,1]} 1 + f(1 - z) + \int_z^1 \frac{1-t}{f(t)} \mathrm{d}t.$$

We will show how to compute the optimal allocation function $f(\cdot)$ in section 4.4.1. For the rest of this section, we formally show that the performance of *GreedyAllocation* in general graphs with allocation function $f(\cdot)$ is at most $\beta(f)$.

Lemma 4.4.2. *Let $f(\cdot)$ be the allocation function. In processing vertex v in GreedyAllocation, we must have either $y = 1$ or $\sum_{u \in N(v)} \max\{y - y_u, 0\} = f(y)$.*

Proof. Let $H(t) = \sum_{u \in N(v)} \max\{t - y_u, 0\} - f(t)$. Note that H is continuous and $H(0) = -f(0) < 0$.

Assume $y < 1$. Notice that $H(1) > 0$. Otherwise, we can set $y = 1$. If $H(y) < 0$, then by intermediate value theorem there is some $t \in (y, 1)$ for which $H(t) = 0$. This contradicts the maximality of y. Hence $H(y) = 0$, as desired. □

Our previous discussion implies that *GreedyAllocation* is competitive against the minimum *integral* vertex cover. In fact, our algorithm is also competitive against the minimum *fractional* vertex cover in general graphs.

Theorem 4.4.3. *Let $f : [0,1] \longrightarrow \mathbb{R}_+$ be a continuous allocation function such that $\frac{1-t}{f(t)}$ is decreasing. Let $\beta = \max_{z \in [0,1]} 1 + f(1-z) + \int_z^1 \frac{1-t}{f(t)} \mathrm{d}t$ and $F(x) = \int_0^x \frac{1-t}{f(t)} \mathrm{d}t$. Then GreedyAllocation$(f)$ is β-competitive against the optimal fractional vertex cover in general graphs.*

Proof. Let y^* be the minimum fractional vertex cover. Denote by v the current online vertex. Consider the following charging scheme.

- Charge $(f(y) + 1 - y)\, y_v^*$ to v.
- Charge $(y - y_u + F(y) - F(y_u))\, y_u^*$ to $u \in X$, where $X = \{u \in N(v) \mid y_u < y\}$.

We claim that the total charges are sufficient to cover the potential increment $1 - y + \sum_{u \in X}(y - y_u)$.

Observe that since $y_v^* + y_u^* \geq 1$ for all $u \in N(v)$ and $f(y) \geq \sum_{u \in X}(y - y_u)$, we have

$$\begin{aligned} f(y)y_v^* + \sum_{u \in X}(y - y_u)y_u^* &\geq \sum_{u \in X}(y - y_u)(y_v^* + y_u^*) \\ &\geq \sum_{u \in X}(y - y_u). \end{aligned}$$

Furthermore,

$$\begin{aligned}
& (1-y)y_v^* + \sum_{u \in X} (F(y) - F(y_u))\, y_u^* \\
\geq\ & (1-y)y_v^* + \sum_{u \in X} (F(y) - F(y_u))\, (1 - y_v^*) \\
\geq\ & 1 - y,
\end{aligned}$$

where the last inequality follows from Lemmas 4.4.2 and 4.4.1.

The above shows that the proposed charging scheme indeed accounts for the total potential increment. Now we bound the total charges to a vertex v over the execution of the algorithm.

When v arrives, y_v is initialized as $1-y$ and v is charged $(f(y) + 1 - y)\, y_v^*$. After that, when y_v increases from a to b, v is charged $(a - b + F(a) - F(b))\, y_v^*$. Note that the sum of these terms telescopes and is at most

$$(1 - (1-y) + F(1) - F(1-y))\, y_v^* = (y + F(1) - F(1-y))\, y_v^*.$$

Therefore the total charges to v are at most

$$\begin{aligned}
& (f(y) + 1 - y)\, y_v^* + (y + F(1) - F(1-y))\, y_v^* \\
=\ & \left(1 + f(1-y) + \int_y^1 \frac{1-t}{f(t)} \mathrm{d}t\right) y_v^* \\
\leq\ & \beta y_v^*.
\end{aligned}$$

This implies that the total potential is bounded by $\beta \sum_{v \in V} y_v^*$, which shows that our algorithm is β-competitive. $\square$

4.4.1 Computing the optimal allocation function

The next step is then to find a good $f(y)$ to get a small β. In essence, the goal is to solve the following optimization problem

$$\inf_{f \in \mathcal{F}} \max_{z \in [0,1]} 1 + f(1-z) + \int_z^1 \frac{1-t}{f(t)} \mathrm{d}t, \tag{4.1}$$

where $\mathcal{F}$ is the class of positive continuous functions on $[0,1]$ such that $\frac{1-t}{f(t)}$ is decreasing for each $f \in \mathcal{F}$.

To the best of our knowledge, there is no systematic approach to tackle a minimax optimization problem of this form. A natural way is to first express the optimal z in terms of f, and then use techniques from calculus of variation to compute the best f. However, a major difficulty is that there is no closed form expression for the optimal z.

To overcome this hurdle, we first disregard the requirement that $\frac{1-t}{f(t)}$ be decreasing. (Though, our final optimal solution turns out to satisfy this condition.) We show that such a relaxation of the optimization problem admits a very nice optimality condition, namely that there exists some optimal f such that $1 + f(1-z) + \int_z^1 \frac{1-t}{f(t)}\mathrm{d}t$ is constant for all z. We characterize this property in the following lemma.

Lemma 4.4.4. *Let $r : [0,1] \longrightarrow \mathbb{R}_+$ be a continuous function such that for $\forall p \in [0,1]$, $r(p) + \int_{1-p}^1 \frac{1-x}{r(x)}\mathrm{d}x \leq \gamma$ for some $\gamma > 0$. Then there exists a continuous function $f : [0,1] \longrightarrow \mathbb{R}_+$ such that $\forall p \in [0,1]$, $f(p) + \int_{1-p}^1 \frac{1-x}{f(x)}\mathrm{d}x \equiv \gamma$.*

Proof. Let $r_1 = r$ and $R_1(p) = r_1(p) + \int_{1-p}^1 \frac{1-x}{r_1(x)}\mathrm{d}x$. Define two sequences of functions $\{r_i\}, \{R_i\}$ recursively as follows:

$$r_{i+1} = r_i + \gamma - R_i, R_{i+1}(p) = r_{i+1}(p) + \int_{1-p}^1 \frac{1-x}{r_{i+1}(x)}\mathrm{d}x.$$

Note that r_i, R_i are positive and continuous for every i. We first show $R_i \leq \gamma$ by induction. The base case for $i = 1$ is trivial. Now we assume $R_i \leq \gamma$ for some i. This implies that $r_i \leq r_{i+1}$. We then have

$$\begin{aligned} R_{i+1}(p) = r_{i+1}(p) + \int_{1-p}^1 \frac{1-x}{r_{i+1}(x)}\mathrm{d}x &\leq r_{i+1}(p) + \int_{1-p}^1 \frac{1-x}{r_i(x)}\mathrm{d}x \\ &= r_{i+1}(p) + R_i(p) - r_i(p) = \gamma. \end{aligned}$$

Therefore $R_i \leq \gamma$ for all i and consequently $r_i \leq r_{i+1}$.

Observe that r_i converges pointwise as r_i is bounded by γ and monotonically increases. Let $r_\infty = \lim_{i\to\infty} r_i$.

Moreover, since $r_{i+1} = r_i + \gamma - R_i$, $R_\infty = \lim_{i\to\infty} R_i \equiv \gamma$. On the other hand, we have

$$\begin{aligned}\gamma = R_\infty(p) &= \lim_{i\to\infty}\left(r_i(p) + \int_{1-p}^{1}\frac{1-x}{r_i(x)}\mathrm{d}x\right)\\ &= r_\infty(p) + \lim_{i\to\infty}\int_{1-p}^{1}\frac{1-x}{r_i(x)}\mathrm{d}x.\end{aligned}$$

By the dominated convergence theorem, $\lim_{i\to\infty}\int_{1-p}^{1}\frac{1-x}{r_i(x)}\mathrm{d}x = \int_{1-p}^{1}\frac{1-x}{r_\infty(x)}\mathrm{d}x$ since $\frac{1-x}{r_i(x)}$ is bounded by $\frac{1-x}{r_1(x)}$.

By taking limit in the second recurrence, we get

$$r_\infty(p) = \gamma - \int_{1-p}^{1}\frac{1-x}{r_\infty(x)}dx$$

which implies r_∞ is continuous and hence satisfies our requirement. □

Therefore, it is sufficient to consider functions f that satisfy this optimality condition. A consequence is that such a function $f(1-z) = \beta - 1 - \int_z^1 \frac{1-t}{f(t)}\mathrm{d}t$ is actually differentiable. Differentiating $1 + f(1-z) + \int_z^1 \frac{1-t}{f(t)}\mathrm{d}t$ yields $-f'(1-z) - \frac{1-z}{f(z)} = 0$, or equivalently,

$$f(z)f'(1-z) = z - 1.$$

Although this differential equation is atypical as $f(z)$ and $f'(1-z)$ are not taken at the same point, surprisingly it has closed form solutions, as given below.

Lemma 4.4.5. *Let r be a non-negative differentiable function on $[0,1]$ such that $r(z)r'(1-z) = z-1$. Then*

$$r(z) = \left(\frac{1+k}{2} - z\right)^{\frac{1+k}{2k}}\left(z + \frac{k-1}{2}\right)^{\frac{k-1}{2k}},$$

where $k \geq 1$. Moreover, $\frac{1-t}{r(t)}$ is decreasing for $t \in [0,1]$.

Proof. We have

$$r(p)r'(1-p) = p - 1.$$

Replacing p by $1-p$, we get

$$r(1-p)r'(p) = -p. \tag{4.2}$$

Hence,

$$(r(p)r(1-p))' = 1 - 2p \implies r(p)r(1-p) = p - p^2 + c \tag{4.3}$$

for some c. Note that $r(0)r(1) = c \geq 0$. From Eqn (4.2) and (4.3), we get $r'(p)/r(p) = p/(p^2 - p - c)$. Let $k = \sqrt{1+4c} \geq 1$. By taking partial fraction and using $(\ln r(p))' = r'(p)/r(p)$,

$$\frac{r'(p)}{r(p)} = \frac{1}{2k}\left(\frac{1+k}{p - \frac{1+k}{2}} - \frac{1-k}{p - \frac{1-k}{2}}\right) \implies r(p) = D\frac{\left|p - \frac{1+k}{2}\right|^{\frac{1+k}{2k}}}{\left|p - \frac{1-k}{2}\right|^{\frac{1-k}{2k}}}$$

for some constant D. It is easy to check that $r(p)r(1-p) = D^2(p - p^2 - c) \implies D = 1$. Since $k \geq 1$, we get the required $r(p)$.

Now we show that $\frac{1-t}{r(t)}$ is decreasing for $t \in [0,1]$. Taking the derivative of $\frac{1-t}{r(t)}$, we have $-1 - (1-t)r'(t)/r(t) = -1 - (1-t)t/(t^2 - t - c) = c/(t^2 - t - c) \leq 0$, as desired. $\square$

The final step is just to select the best f from the family of solutions. Since $1 + f(1 - z) + \int_z^1 \frac{1-t}{f(t)}dt$ is constant, it suffices to find the smallest $1 + f(0)$, which corresponds to the case $k \approx 1.1997$.

Theorem 4.4.6. *Let $f(z) = \left(\frac{1+k}{2} - z\right)^{\frac{1+k}{2k}}\left(z + \frac{k-1}{2}\right)^{\frac{k-1}{2k}}$, where $k \approx 1.1997$. GreedyAllocation(f) is 1.901-competitive for the online fractional vertex cover problem in general graphs.*

Finally, we remark that our algorithm can be viewed as a generalization of the well-known greedy algorithm because the solution $f(z) = 1 - z$ (with $k = 1$) is equivalent to a variant of the greedy algorithm.

4.5 Online fractional matching in general graphs

We give a primal-dual analysis of the algorithm given in the last section. A by-product of this primal-dual analysis is a $\frac{1}{1.901} \approx 0.526$-competitive algorithm for online fractional matching in general graphs.

Let $\beta \approx 1.901$ be the competitive ratio established in the last section and $f(z)$ be the same as that of Theorem 4.4.6. Our primal-dual analysis shares some similarities with the one for online bipartite fractional matching by Buchbinder et al. [6].

Our algorithm *PrimalDual* applies to both online fractional vertex cover and matching. When restricted to the dual, it is identical to *GreedyAllocation*.

To analyze the performance, we claim that the following two invariants hold throughout the execution of the algorithm.

Algorithm 7: *PrimalDual*

```
Input: Online graph $G = (V, E)$
Output: A fractional vertex cover $\{y_v\}$ of $G$ and a fractional matching $\{x_{uv}\}$.
Let $T$ be the set of known vertices. Initialize $T = \emptyset$;
for each online vertex $v$ do
    Maximize $y \leq 1$, s.t., $\sum_{u \in N(v) \cap T} \max\{y - y_u, 0\} \leq f(y)$;
    Let $X = \{u \in N(v) \cap T \mid y_u < y\}$;
    for each $u \in X$ do
        $y_u \leftarrow y$;
        $x_{uv} \longleftarrow \frac{y-y_u}{\beta}\left(1 + \frac{1-y}{f(y)}\right)$;
    end
    For each $u \in (N(v) \cap T) \setminus X$, $x_{uv} \longleftarrow 0$;
    $y_v \leftarrow 1 - y$;
    $T \leftarrow T \cup \{v\}$;
end
Output $\{y_v\}$ for all $v \in V$;
```

Invariant 1:

$$\frac{y_u + f(1 - z_u) + \int_{z_u}^{y_u} \frac{1-t}{f(t)} \mathrm{d}t}{\beta} \geq x_u,$$

where z_u is the potential of u set upon its arrival, y_u is the current potential of u and $x_u = \sum_{v \in N(u)} x_{uv}$ is the sum of the potentials on the edges incident to u. Note that the LHS is at most 1 (see last section for details), which guarantees that the primal is feasible as long as the invariant holds.

Invariant 2:

$$\sum_{u \in T} y_u = \beta \sum_{(u,v) \in E \cap T^2} x_{uv}$$

Invariant 2 guarantees that the primal and dual objective values are within a factor of β from each other. By weak duality, this implies that the algorithm is β-competitive for online fractional vertex cover and $\frac{1}{\beta}$-competitive for online fractional matching in general graphs. Note that both invariants trivially hold at the beginning.

The idea behind Invariant 1 is to enforce some kind of correlation between y_u and x_u. For instance, when y_u is small, x_u should not be excessively large because x_u must be increased to (partially) offset any future increase in y_u in order to maintain Invariant 2.

We claim that both invariants are preserved.

Lemma 4.5.1 (Invariant 2). *In each iteration of the algorithm, the increase in the dual objective value is exactly β times that of the primal.*

Proof. The dual increment is

$$1 - y + \sum_{u \in X} (y - y_u)$$

and the primal increment is

$$\sum_{u \in X} \frac{y - y_u}{\beta} \left(1 + \frac{1 - y}{f(y)}\right).$$

Thus it suffices to show that $1 - y = \sum_{u \in X} (y - y_u) \frac{1-y}{f(y)}$. This just follows from Lemma 4.4.1, which states that we have either $y = 1$ or $\sum_{u \in X} (y - y_u) = f(y)$. $\square$

Lemma 4.5.2 (Invariant 1). *After processing online vertex v, we have $x_v \leq \frac{y_v + f(1-y_v)}{\beta}$ and $x_u \leq \frac{y + f(1-z_u) + \int_{z_u}^{y} \frac{1-t}{f(t)} \mathrm{d}t}{\beta}$ for $u \in X$.*

Proof. Note that $x_v = \sum_{u \in X} x_{uv}$ is just the increase in the primal objective value. By Invariant 2, $x_v = \frac{1 - y + \sum_{u \in X}(y - y_u)}{\beta}$. Our claim for x_v follows since $y_v = 1 - y$ and $\sum_{u \in X}(y - y_u) \leq f(y)$.

By Invariant 1, the previous x_u satisfies

$$x_u - x_{uv} \leq \frac{y_u + f(1 - z_u) + \int_{z_u}^{y_u} \frac{1-t}{f(t)} \mathrm{d}t}{\beta}.$$

This proof is finished by noticing that

$$x_{uv} = \frac{y - y_u}{\beta} \left(1 + \frac{1 - y}{f(y)}\right) \leq \frac{1}{\beta} \left(y - y_u + \int_{y_u}^{y} \frac{1 - t}{f(t)} \mathrm{d}t\right),$$

as $\frac{1-t}{f(t)}$ is a decreasing function. $\square$

Finally, it is clear that the dual is always feasible. The primal is feasible because $x \geq 0$ and Invariant 1 guarantees that $x_v \leq 1$, as discussed earlier. Combining this and the two lemmas, we have our main result.

Theorem 4.5.3. *Our algorithm is $\beta \approx 1.901$-competitive for online fractional vertex cover and $\frac{1}{\beta} \approx 0.526$-competitive for online fractional matching for general graphs.*

It is possible to extend our algorithm to the vertex-weighted fractional vertex cover problem and the fractional b-matching problem. As in the previous chapters, we have chosen not to do so as this is relatively straightforward and involves little new ideas.

4.6 Hardness results

In this section, we present new hardness results for the problems considered in this chapter. All of our hardness results are obtained by considering appropriate bipartite graphs. Let $G = (L, R, E)$ be a bipartite graph with left vertices L and right vertices R. We study different variants of the online vertex cover and matching problems by imposing certain constraints on the vertex arrival order.

- **1-alternation.** The left vertices L are offline and the right vertices in R arrive online. When a vertex $v \in R$ arrives, all its incident edges are revealed. This is simply the most basic online bipartite matching and vertex cover.
- **k-alternation:** There are k phases and $L_0 \subseteq L$ is the set of offline vertices. In each phase $1 \leq i \leq k$, if i is odd (resp. even), vertices from a subset of R (resp. L) arrive one by one. Note that the case $k = \infty$ effectively removes any constraint on the vertex arrival order.

Our hardness results hold for the fractional version of the problems and hence the more general integral version as well. This is because any randomized algorithm for the latter can be converted into a deterministic algorithm for the fractional version simply by setting all the variables equal to their expected values.

4.6.1 Lower bounds for the online vertex cover problem

We give lower bounds on the competitive ratios for online bipartite vertex cover with 2- and 3-alternation, and an upper bound for online bipartite matching with 2-alternation. These hardness results also apply to the more general problems of online vertex cover and matching in general graphs.

Proposition 4.6.1. *There is a lower bound of* $1 + \frac{1}{\sqrt{2}} \approx 1.707$ *for online bipartite vertex cover with 2-alternation.*

Proof. Suppose that an algorithm A is $(1 + \beta)$-competitive. Without loss of generality, we may assume that A is deterministic. Our approach is to bound β by considering a family of complete bipartite graphs. Thus a new online vertex is always adjacent to all the vertices on the other side.

Let $|L_0| = d$ and y be the fractional vertex cover maintained by A. We claim that after processing the i-th vertex in R_1, we have

$$\sum_{u \in L_0} y_u \leq i\beta.$$

The reason is that the adversary can generate infinitely many left online vertices in phase 2 and hence y_v, for any $v \in R_1$, converges to 1 (otherwise, if y_v, which monotonically increases, converges to some $l < 1$, then $y_u \geq 1 - l$ for $u \in L_2$ and the cost of the vertex cover found is unbounded while the optimal solution is at most i).

Let $v_i^{(1)}$ be the ith vertex in R_1. Next we claim that

$$y_{v_i^{(1)}} \geq 1 - \frac{i\beta}{d}.$$

Since $\sum_{u \in L_0} y_u \leq i\beta$ after processing $v_i^{(1)}$, by Pigeonhole Principle there must be some $y_u \leq \frac{i\beta}{d}$. To maintain a valid vertex cover, we need $y_{v_i^{(1)}} \geq 1 - \frac{i\beta}{d}$.

Finally, we have

$$\sum_{v \in R_1} y_v \leq d\beta. \tag{4.4}$$

Otherwise, the adversary can generate infinitely many online vertices to append R_1 in which case y_u will be increased to 1 eventually for all $u \in L_0$, i.e., $\sum_{u \in L_0} y_u = d$. This contradicts the fact that A is $(1 + \beta)$-competitive.

Now by taking $|R_1| = \sqrt{2}d$, we get

$$\sum_{i=1}^{\sqrt{2}d} \left(1 - \frac{i\beta}{d}\right) \leq \sum_{v \in R_1} y_v \leq d\beta,$$

from which our desired result follows by taking $d \longrightarrow \infty$. □

Proposition 4.6.2. *There is a lower bound of* $1 + \sqrt{\frac{1}{2}\left(1 + \frac{1}{e^2}\right)} \approx 1.753$ *for online bipartite vertex cover with 3-alternation.*

Proof. Again, let $d = |L_0|$. We extend the idea used in the proof of the bound $1 + \frac{1}{\sqrt{2}}$ for 2-alternation. Let x_i be the amount of resources spent on L_0 by the i-th vertex of R_1, i.e. the increment in the potential of L_0. Let y_i be its own potential. Then $y_i \geq 1 - (x_1 + \cdots + x_i)/d$, $x_1 + \cdots + x_i \leq i\beta$ and $y_1 + \cdots + y_i \leq d\beta$ by the argument used in the proof of Proposition

1.

The new idea is that in phase 2, assuming $|R_1| = i$, at most $i\beta$ resources can be spent on L_0 and L_2. (This is because the adversary can append infinitely many vertices to the current L_2.) Now consider the j-th vertex u_j in L_2. Similar to Eqn.(4.4), we have $\sum_{v \in R_1} y_v \leq (d+j) \cdot \beta$ after processing u_j, since the adversary can append infinitely many online vertices to R_3. Consequently, $y_{u_j} \geq 1 - \min\{y_v \mid v \in R_1\} \geq 1 - (d+j) \cdot \beta/i$ by the pigeonhole principle. Therefore,

$$x_1 + \cdots + x_i \leq i\beta - \sum_{j=1}^{\ell} \left(1 - \frac{(d+j)\beta}{i}\right),$$

where $\ell = |L_2|$.

Let $X(i) = x_1 + x_2 + \cdots + x_i$. If $i \leq d\beta$, we have $X(i) \leq i\beta$. When $i > d\beta$, by setting $\ell = \frac{i}{\beta} - d$, we have

$$\begin{aligned} X(i) &= i\beta - \left(1 - \frac{d\beta}{i}\right)\ell + \frac{\beta}{2i}\ell^2 + O(1) \\ &= d + i\beta - \frac{i}{2\beta} - \frac{\beta d^2}{2i} + O(1). \end{aligned}$$

Notice that our bound on $X(i)$ holds for arbitrary i, since the adversary can arbitrarily manipulate the future input graph to fool the deterministic algorithm.

Since $y_i \geq 1 - X(i)/d$ and $\sum_{i=1}^{k} y_i \leq d\beta$ for any k, we have

$$\sum_{i=1}^{k} \left(1 - \frac{X(i)}{d}\right) \leq d\beta.$$

Let $\alpha = \frac{1}{\sqrt{2\beta^2 - 1}}$. By setting $k = \beta\alpha d$ and considering $i \leq d\beta$, $i > d\beta$ separately, we get

$$d\beta \geq \sum_{i=1}^{d\beta} \left(1 - \frac{i\beta}{d}\right) + \sum_{i=d\beta+1}^{k} \left(\frac{i}{2\beta d} + \frac{\beta d}{2i} - \frac{i\beta}{d} + O(1/d)\right)$$

By taking $d \longrightarrow \infty$ and using $\sum_{i=1}^{n} 1/i \approx \ln n$, we have the desired result.

□

4.6.2 Upper bounds for the online matching problem

Before establishing our last result on the upper bound for online bipartite matching with 2-alternation, we review how the bound $1 - 1/e$ is proved for the original problem (i.e. 1-alternation) as the same technique is used in a more complicated way. The next proof is a variant of that in [22].

Proposition 4.6.3. *There is an upper bound of* $1-1/e \approx 0.632$ *for online bipartite matching (with 1-alternation).*

Proof. Again, we can consider only the fractional version of the problem and deterministic algorithms. Suppose that an algorithm maintains a fractional matching x. Let $L = \{u_1, ..., u_n\}$ and $R = \{v_1, ..., v_n\}$, with v_i adjacent to $u_1, ..., u_{n+1-i}$. The size of the maximum matching is clearly n. Let $v_1, ..., v_n$ be the order in which the online vertices arrive.

Observe that when v_i arrives, $u_1, ..., u_{n+1-i}$ are indistinguishable from each other. Thus $x_{v_i} := \sum_{u \in N(v_i)} x_{uv_i}$ should be evenly distributed to $u_1, ..., u_{n+1-i}$, i.e. $x_{uv_i} = \frac{x_{v_i}}{n+1-i}$. This argument can be made formal by considering graphs isomorphic to G with the labels of vertices in L being randomly permuted.

Thus, after processing v_k we have

$$x_{u_i} = \frac{x_{v_1}}{n} + ... + \frac{x_{v_k}}{n+1-k}.$$

Moreover, the size of the matching found is $x_{v_1} + \cdots + x_{v_n}$ and $x_{v_1}, \cdots, x_{v_n}$ satisfy $\frac{x_{v_1}}{n} + \cdots + \frac{x_{v_n}}{1} \leq 1$.

Viewing the above as a LP, it is easy to see that $x_{v_1} + ... + x_{v_n}$ is maximized when $x_{v_n}, ..., x_{v_k} = 1, x_{v_{k+1}}, ..., x_{v_n} = 0$ and $\frac{1}{n} + ... + \frac{1}{n-k+1} \approx 1$ for some k. Now when n is large, $\frac{1}{n} + ... + \frac{1}{n-k+1} \approx \ln \frac{n}{n-k}$.

Finally,

$$x_{v_1} + \cdots + x_{v_n} = k = n(1 - 1/e) = (1 - 1/e) \cdot OPT.$$

□

Proposition 4.6.4. *There is an upper bound of 0.6252 for the online matching problem in bipartite graphs with 2-alternation.*

Proof. Again, we can consider only the fractional version of the problem and deterministic algorithms. Suppose that an algorithm is γ-competitive and maintains a fractional matching x.

Let $|L_0| = |L_2| = n, |R_1| = 2n$. The first n vertices of R_1 are adjacent to all vertices in L_0. The two subgraphs induced by L_0 & the last n vertices of R_1 and L_1 & the first n vertices of R_1 are isomorphic to the graph used in the proof of the last theorem. Note that the size of maximum matching is $2n$.

The most important observation here is that after processing the first n vertices of R_1, the fractional matching found must have size at least $n\gamma$ as the current optimal solution has size n. In other words, we have

$$x_{u_1^{(1)}} + \ldots + x_{u_n^{(1)}} = x_{v_1^{(1)}} + \ldots + x_{v_n^{(1)}} \geq n\gamma$$

after the first n vertices of R_1 arrive.

Now the next n vertices of R_1, by the same reasoning in the last theorem, are matched to the extent of k such that $\gamma + \frac{1}{n} + \ldots + \frac{1}{n+1-k} \approx 1$, from which we obtain $k = n(1 - 1/e^{1-\gamma})$. Similarly, L_2 is also matched to an extent of $n(1 - 1/e^{1-\gamma})$.

Putting all the pieces together, we have the inequality

$$\frac{n\gamma + 2n(1 - 1/e^{1-\gamma})}{2n} \geq \gamma \Rightarrow 1 - \frac{1}{e^{1-\gamma}} - \frac{\gamma}{2} \geq 0.$$

The function $1 - \frac{1}{e^{1-\gamma}} - \frac{\gamma}{2}$ is decreasing and has root approximately at 0.6252. $\square$

Chapter 5

Conclusion

A recurring theme in this thesis is the application of the various extensions of an elegant charging scheme to different variants of the online vertex cover problems. We have shown how this yields competitive algorithms for the basic, edge-weighted and submodular versions of online bipartite vertex cover as well as the online vertex cover problem in general graphs.

Another main feature permeative in our work is that by reverse-engineering the charging analysis, one can often in hindsight design a primal-dual analysis for the problem considered. As in chapters 3 and 4, a by-product of this recipe is a primal-dual algorithm for the corresponding dual matching problem. This also suggests a potentially viable approach to a prominent open problem in the area of online matching.

In [14], the online bipartite weighted matching problem was first studied and an optimal $1 - 1/e$-competitive algorithm was given in a special case. The edge-weighted version of OBVC is precisely the dual of this problem and our work suggests that by analyzing our algorithm or some variant of it in the primal-dual framework, one may hope to obtain a solution to the general case.

The fact that the charging and primal-dual analyses come hand-in-hand together is perhaps reminiscent of an old tenet in mathematics. Even when a problem is solved, it is often still useful to propose alternate solutions to it as they may shed light on the greater picture of the story. Some arguments are simply more amenable to generalizations than others. In our case, the charging scheme proves to be easier to intuit and extend than the primal-dual method. We are eager to see yet more generalizations of online matching inspired by extending our charging scheme.

Finally, we conclude the thesis with some possible future research directions.

5.1 Future work

Online bipartite weighted matching As mentioned earlier, our result on online bipartite edge-weighted vertex cover may suggest a new direction for tackling this major open problem related to online matching.

Charging scheme for dual Adwords The Adwords problem [28], which generalizes online bipartite matching, is arguably the most important open problem in the area. While our work shows that the dual of online bipartite matching seems easier to tackle, it is not clear if this is also the case for Adwords. It would be interesting to obtain similar charging schemes for the dual of the special cases of Adwords solved in the literature, and perhaps, even the general case.

Submodular in other online problems In online algorithms, submodularity seems to be considered less often than other themes in combinatorial optimization. We hope that our work will stimulate the interest in combining submodularity with existing online problems in the literature. To certain extent, our results on OBSM and OBSVC show that various ingredients used in offline submodular optimization are still applicable online. We are hopeful that some of the powerful machineries developed for handling submodularity over the past few decades will find applications in various online settings.

Online integral matching One open problem left unanswered in chapter 4 is whether the barrier of $1/2$ can be overcome for online integral matching in general and bipartite graphs. Our result on fractional matching suggests that it may be possible.

Online vertex cover in weaker adversary models The online bipartite matching problem has been studied in many different models, some of which are weaker than the oblivious adversary model. In particular, competitive ratios better than $1 - 1/e$ were obtained in the stochastic [13, 27] and random arrival models [26, 18]. Can we beat $\frac{1}{1-1/e}$ in these models for online bipartite vertex cover?

Appendix A

Multislope Ski Rental

Reduction from Multislope Ski Rental to Online vertex-weighted Bipartite Vertex Cover There is a total of n states $[n]$ in the multislope ski rental problem. Each state i associated with buying cost b_i and rental cost r_i. As argued in [.], we may assume that we start in state 1 and have $0 = b_1 \leq b_2 \leq \ldots \leq b_n, r_1 \geq r_2 \geq \ldots \geq r_n \geq 0$. The game starts at time 0 and ends at some unknown time t_{end} determined by the adversary.

At each time $t \in [0, t_{end}]$, we can transition from the current state i to some state $j > i$. Let state f be the final state at time t_{end}. The total cost incurred is given by

$$b_f + \sum_{i=1}^{f} x_i r_i,$$

where x_i is the amount of time spent in state i. The classical ski rental problem corresponds to $n = 2$ and $b_1 = 0, b_2 = B, r_1 = 1, r_2 = 0$.

Consider now the discrete version of this problem. We discretize time into consecutive intervals of length ϵ for some small $\epsilon > 0$. At the beginning of each interval, we can stay in the current state i or transition from i to some state $j > i$. Each of the two choices correspond to a cost of $r_i\epsilon$ or $b_j - b_i + r_j\epsilon$.

We are ready to describe the reduction to online vertex-weighted bipartite vertex cover. Let $L = \{1, 2, \ldots, n\}$ with weights $w_i = b_{i+1} - b_i$ for $i < n$ and $w_n = \infty$. The $(qn + r)$-th online vertex $v_{qn+k} \in R$, where q is a nonnegative integer and $1 \leq k \leq n$, has weight $(r_k - r_{k+1})\epsilon$ (with $r_{n+1} = 0$) and is adjacent to the left vertices $1, \ldots, k$.

Intuitively, the $(q + 1)$-th time interval is represented by the online vertices $qn +$

$1, \ldots, qn+n$. If we are in state i, (1) the left vertices $1, \ldots, i-1$ should be covered and have total weight $b_i - b_0 = b_i$ and, (2) the online vertices $qn+i \ldots qn+n$ should be covered and have total weight $r_i\epsilon$. Thus when we transition from state i to state $j > i$, the vertices $i, \ldots, j-1$ should be added to the cover. Moreover, the left vertex n, which has infinite weight, is used to ensure that the algorithm is forced to put the online vertex $qn+n$, which has weight r_n, into the cover.

Finally, we show that a c-competitive algorithm for online vertex-weighted bipartite vertex cover gives a c-competitive algorithm for multislope ski rental under the above reduction. Consider the vertex cover maintained by the algorithm after processing online vertices $qn+1, \ldots, qn+n$. Suppose that $1, \ldots, i-1$ are in the cover but i is not. Then the online vertices $qn+i \ldots qn+n$ must also be in the cover. Thus we can simply stay in (or transition to if the previous state is smaller) state i. It is clear that this strategy is valid by the preceding discussion. Furthermore, the cost incurred by the algorithm for multislope ski rental is no greater than the counterpart for vertex cover.

Bibliography

[1] G. Aggarwal, G. Goel, C. Karande, and A. Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1253–1264. SIAM, 2011.

[2] J. Aronson, M. Dyer, A. Frieze, and S. Suen. Randomized greedy matching. ii. *Random Structures & Algorithms*, 6(1):55–73, 1995.

[3] N. Bansal, A. Gupta, J. Li, J. Mestre, V. Nagarajan, and A. Rudra. When lp is the cure for your matching woes: improved bounds for stochastic matchings. *Algorithms–ESA 2010*, pages 218–229, 2010.

[4] A. Blum, T. Sandholm, and M. Zinkevich. Online algorithms for market clearing. *Journal of the ACM (JACM)*, 53(5):845–879, 2006.

[5] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*, volume 53. Cambridge University Press Cambridge, 1998.

[6] N. Buchbinder, K. Jain, and J. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. *Algorithms-ESA 2007*, pages 253–264, 2007.

[7] N. Buchbinder and J.S. Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009.

[8] Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal: Dual approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2):93–263, 2009.

[9] M. Demange and V.T. Paschos. On-line vertex-covering. *Theoretical Computer Science*, 332(1):83–108, 2005.

[10] N.R. Devanur and K. Jain. Online matching with concave returns. In *Proceedings of the 44th symposium on Theory of Computing*, pages 137–144. ACM, 2012.

[11] N.R. Devanur, K. Jain, and R.D. Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *SODA '13: Proceedings of the thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2013. to appear.

[12] Nikhil R. Devenur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *EC '09: Proceedings of the tenth ACM conference on Electronic commerce*, pages 71–78, New York, NY, USA, 2009. ACM.

[13] J. Feldman, A. Mehta, V. 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.

[14] Jon Feldman, Nitish Korula, Vahab Mirrokni, S Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In *Internet and Network Economics*, pages 374–385. Springer, 2009.

[15] Goel G. and Tripathi P. Matching with our eyes closed. In *Foundations of Computer Science, 2012. FOCS'12. 53rd Annual IEEE Symposium on.* IEEE, 2012.

[16] G. Goel and A. Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, volume 8, pages 982–991, 2008.

[17] B. Kalyanasundaram and K.R. Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1):319–325, 2000.

[18] C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 587–596. ACM, 2011.

[19] A.R. Karlin, C. Kenyon, and D. Randall. Dynamic tcp acknowledgement and other stories about e/(e-1). In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 502–509. ACM, 2001.

[20] A.R. Karlin, M.S. Manasse, L.A. McGeoch, and S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

[21] A.R. Karlin, M.S. Manasse, L. Rudolph, and D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.

[22] R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358. ACM, 1990.

[23] S. Khot and O. Regev. Vertex cover might be hard to approximate to within 2-[epsilon]. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[24] Z. Lotker, B. Patt-Shamir, D. Rawitz, and S. Albers. Rent, lease or buy: Randomized algorithms for multislope ski rental. In *25th International Symposium on Theoretical Aspects of Computer Science (STACS 2008)*, volume 1, pages 503–514, 2008.

[25] Poloczek M. and Szegedy M. Randomized greedy algorithms for the maximum matching problem with new analysis. In *Foundations of Computer Science, 2012. FOCS'12. 53rd Annual IEEE Symposium on.* IEEE, 2012.

[26] M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 597–606. ACM, 2011.

[27] V.H. Manshadi, S.O. Gharan, and A. Saberi. Online stochastic matching: Online actions based on offline statistics. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1285–1294. SIAM, 2011.

[28] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22, 2007.

[29] Aranyak Mehta and Vahab Mirrokni. Online ad serving: Theory and practice, 2011.

[30] Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.