

Online Combinatorial Optimization Problems with Non-linear Objectives



Zhiyi Huang

Abstract We survey some recent progress on the design and the analysis of online algorithms for optimization problems with non-linear, usually convex, objectives. We focus on an extension of the online primal dual technique, and highlight its application in a number of applications, including an online matching problem with concave returns, an online scheduling problem with speed-scalable machines subjective to convex power functions, and a family of online covering and packing problems with convex objectives.

1 Introduction

Online combinatorial optimization problems are ubiquitous. In these problems, partial decisions must be made irrevocably based on the information revealed so far. For example, in the online bipartite matching problem, only one side of the vertices is given at the beginning. Then, the vertices on other side arrive one by one. On the arrival of an online vertex, its incident edges are revealed and the algorithm must irrevocably decide how to match it without any knowledge of the vertices that will arrive later. Due to the uncertainty of future, it is impossible in general to guarantee a maximum cardinality matching in the online setting. The performance of the algorithm is measured by the ratio of the size of the obtained matching to that of the maximum matching in hindsight. The competitive ratio of the algorithm is defined to be the above ratio in the worst case.

The online bipartite matching problem and its many variants are extensively studied in the literature. So are other classic online combinatorial optimization problems, including online covering and packing problems, online caching and

Z. Huang (✉)

The University of Hong Kong, Pokfulam, Hong Kong

e-mail: zhiyi@cs.hku.hk

© Springer Nature Switzerland AG 2019

D.-Z. Du et al. (eds.), *Nonlinear Combinatorial Optimization*,

Springer Optimization and Its Applications 147,

https://doi.org/10.1007/978-3-030-16194-1_8

paging, and online scheduling. Most of the previous work have focused on problems with linear objectives in the sense that the objective function can be written as a linear function of the decision variables. For example, the standard formulation of online bipartite matching uses an indicator variable $x_e \in \{0, 1\}$ to denote whether an edge e is chosen in the matching. The objective, i.e., the cardinality of the matching, is simply the sum of all such indicator variables. It remains linear even if we consider the generalization which allows edge weights and seeks to maximize the total weight instead of the cardinality of the matching.

However, there are also a wide range of problems whose natural formulations involve non-linear (often convex or concave) objectives. For instance, a variant of the online bipartite matching problem originated from the Adwords problem allows an offline vertex to match multiple online vertices, but imposes a cap (e.g., an advertiser's budget in the Adwords problem) on the total gain of an offline vertex. In other words, the contribution of an offline vertex to the objective function is the smaller of its cap and the sum of weights of matched edges incident to it. The objective is then the sum of such cap-additive functions of the decision variables, which are concave instead of linear. Other examples of online combinatorial optimization problems with non-linear objectives include online scheduling problems with speed-scalable machines in which the energy consumption of a machine is a convex function of its speed, and a generalized online resource allocation problems in which each resource has a "soft capacity constraint" specified by a convex production cost function.

In this chapter, we will survey some recent progress in the design and analysis of online algorithms for online combinatorial optimization problems with non-linear objectives. We will talk about a line of research on generalizing the online primal dual technique by Buchbinder and Naor [6], which was originally designed for linear objectives, to handle convex and concave objectives. The generalization allows us to solve the above-mentioned problems that do not admit natural linear program relaxations.

In particular, we will focus on convex programs and Fenchel's duality in the non-stochastic setting. Readers are also referred to other interesting work along this line, including using the online primal dual technique (or dual fitting) with Lagrangian duality by Anand et al. [2], Gupta et al. [12], Nguyen [17], and online stochastic convex optimization by Agrawal and Devanur [1].

1.1 Organization

We will first recap the online primal dual technique (Section 2), and explain how to extend it to handle convex programs via Fenchel's duality (Section 3). Then, we will talk about three applications of the extension: online matching with concave returns [11] (Section 4), online scheduling with speed scaling [10] (Section 5), and online covering and packing problems with convex objectives [3] (Section 6).

2 Online Primal Dual for Linear Objectives

We first give a brief introduction to the original online primal dual technique for problems with linear objectives by demonstrating its application in the ski-rental problem (e.g., Buchbinder and Naor [6]). Readers who are familiar with the original online primal dual technique may skip this section.

In the ski-rental problem, a skier arrives at a ski resort but does not know when the ski season will end. Every day, if the skier has not bought skis yet, she needs to decide whether to rent skis at a cost of \$1 or to buy skis at a cost of \$ B . We will assume for simplicity of our discussion that B is a positive integer. The goal is to minimize the total cost.

We follow the standard framework of competitive analysis of online algorithms. That is, we use the optimal cost in hindsight, which equals $\$T$ if $T \leq B$ and $\$B$ otherwise, as the benchmark. The performance of an algorithm for the ski-rental problem and, in general, for any cost minimization problem, is measured by the ratio of the expected cost of the algorithm to the optimal cost. We say that an algorithm is F -competitive or it has competitive ratio F if the aforementioned ratio is at most F for any instance of the problem. Obviously, the competitive ratio of an algorithm is always greater than or equal to 1, and the smaller the better.

Consider a natural linear program formulation of the ski-rental problem and the corresponding dual program below.

$$\begin{array}{ll}
 \text{minimize} & B \cdot x + \sum_{t=1}^T y_t \\
 \text{subject to} & x + y_t \geq 1 \quad t = 1, \dots, T \\
 & x, y_t \geq 0 \quad t = 1, \dots, T \\
 \\
 \text{maximize} & \sum_{t=1}^T \alpha_t \\
 \text{subject to} & \sum_{t=1}^T \alpha_t \leq B \\
 & 0 \leq \alpha_t \leq 1 \quad t = 1, \dots, T
 \end{array}$$

Here, x is the indicator of whether the algorithm buys skis, and y_t is the indicator of whether the algorithm rents skis on day t . For simplicity, we will only discuss solving the linear programs online to minimize the expected primal objective value. Readers are referred to the survey by Buchbinder and Naor [6, Section 3] for an online rounding algorithm which convert the online fractional solution into a randomized integral algorithm for the ski-rental problem with the same competitive ratio. Roughly speaking, the fractional value of x denotes the probability that the algorithm buys skis and the fractional value of y_t denotes the probability that the algorithm rents skis on day t .

2.1 High-Level Plan

As the input information being revealed over time piece by piece, more variables and constraints of the primal and dual programs are presented to the algorithm. On each day t , a new primal variable $y_t \geq 0$, a new primal constraint $x + y_t \geq 1$, and a new dual variable $1 \geq \alpha_t \geq 0$ arrive. Online primal dual algorithms maintain at all time a feasible primal assignment and a feasible dual assignment simultaneously. Let $\Delta_t P$ and $\Delta_t D$ be the changes of the primal and dual objectives, respectively, on day t . The goal is to update the primal and dual variables to satisfy:

$$\Delta_t P \leq F \cdot \Delta_t D \quad (1)$$

for some fixed parameter $F \geq 1$. If Equation (1) holds and the initial values of the primal and dual objectives are zero, the algorithm is F -competitive because the final primal objective P is at most F times the final dual objective D , which by weak duality is less than or equal to the optimal primal objective.¹

2.2 Relaxed Complementary Slackness

In order to understand how an online primal dual algorithms can be derived from the structure of the linear programs, we need to look into their optimality conditions. It is known that the offline optimal primal and dual solutions satisfy the complementary slackness conditions, which state that a primal (resp., dual) variable must be zero unless the corresponding dual (resp., primal) constraint is tight (e.g., [9]). Specifically, for the ski-rental problem, we have the followings:

- (a) x must be zero unless $\sum_{t=1}^T \alpha_t = B$;
- (b) y_t must be zero unless $\alpha_t = 1$;
- (c) α_t must be zero unless $x + y_t = 1$.

However, it is generally impossible to satisfy all complementary slackness conditions exactly in an online problem. In particular, it is not possible to guarantee satisfying conditions (a) and (c) exactly in the ski-rental problem. The best we could hope for is to satisfy the complementary slackness conditions approximately. Online primal dual algorithms are therefore driven by satisfying these conditions approximately, where the value of a primal (resp., dual) variable depends on the tightness of the corresponding dual (resp., primal) constraint. Concretely, consider

¹Some applications of the online primal dual technique maintain an alternative set of invariants, e.g., one may consider keeping primal and dual objectives equal and guaranteeing primal feasibility, while showing approximate dual feasibility. However, such variants can be easily rewritten to fit into the framework in this chapter.

the following relaxed conditions for the ski-rental problem. Note that condition (b) will remain the same as it can be satisfied exactly even in the online setting.

- (a') x depends on the tightness of the corresponding dual constraint $\sum_{t=1}^T \alpha_t \leq B$, i.e., it is an increasing function of $\sum_{t=1}^T \alpha_t$;
- (c') α_t must be zero unless $x + y_t = 1$, at the end of day t (the constraint may have slack in the future because the algorithm may increase x).

2.3 Online Primal Dual Algorithms

Fix any day $1 \leq i \leq T$, first consider the new dual variable α_i . To maximize the dual objective, letting $\alpha_i = \min\{1, B - \sum_{t=1}^{i-1} \alpha_t\}$ is the most natural choice in light of the dual constraints. Recall that B is an integer, this is equivalent to letting $\alpha_i = 1$ if $i \leq B$ and $\alpha_i = 0$ otherwise. As a result, $\sum_t \alpha_t$ increases by 1 on each day $i \leq B$ and the algorithm increases x according to condition (a'). Let x_i denote the value of x after day i for $i = 1, \dots, B$, and $x_0 = 0$. After day B , $\sum_t \alpha_t$ and, thus, the value of x must remain constant because of condition (a'). Further, let $y_i = 1 - x_i$ to satisfy condition (c'). Note that this is also the most natural choice to minimize the primal objective. Finally, since we let $\alpha_i = 0$ for $i > B$, we must have $y_i = 0$ and $x = 1$ on any day $i > B$ according to condition (b) and, thus, $x_B = 1$ must hold.

In sum, for every monotone sequence $x_t, t = 0, 1, \dots, B$, such that $x_0 = 0$ and $x_B = 1$, there is an online primal dual algorithm as follows:

1. On day $i = 1, \dots, B$, let $\alpha_i = 1, x = x_i$, and $y_i = 1 - x_i$.
2. On day $i > B$, let $\alpha_i = 0, x$ remains the same (i.e., equals 1), and $y_i = 0$.

2.4 Online Primal Dual Analysis

It remains to find the best monotone sequence $\{x_t\}_{t=1, \dots, B}$ such that Equation (1) holds with the smallest possible $F \geq 1$. Note that both the primal and dual objectives remain the same after day B , so it suffices to analyze Equation (1) on the first B days. On each day $t = 1, \dots, B, x$ changes from x_{t-1} to x_t , and $y_t = 1 - x_t$. So the change of primal objective is equal to:

$$\Delta_t P = B \cdot (x_t - x_{t-1}) + (1 - x_t) = (B - 1) \cdot x_t - B \cdot x_{t-1} + 1$$

On the other hand, the algorithm sets $\alpha_t = 1$. So the change of dual objective is

$$\Delta_t D = 1$$

So Equation (1) becomes $(B - 1) \cdot x_t - B \cdot x_{t-1} \leq F - 1$. Reorganizing terms, it is equivalent to the followings:

$$x_t + F - 1 \leq \frac{B}{B-1}(x_{t-1} + F - 1) ,$$

Using the above inequality for $1, 2, \dots, t$, we get that:

$$x_t \leq \left(\frac{B}{B-1}\right)^t (x_0 + F - 1) - (F - 1) = \left(\left(\frac{B}{B-1}\right)^t - 1\right)(F - 1) \quad (2)$$

Let $e(B) = \left(\frac{B}{B-1}\right)^B$. We have $e(B) \geq e$ and $\lim_{B \rightarrow +\infty} e(B) = e \approx 2.718$. By $x_B = 1$ and the above inequality, we have $F \geq \frac{e(B)}{e(B)-1}$. Let $F = \frac{e(B)}{e(B)-1}$ and let x_t be such that Equation (2) holds with equality. Then, we have an online primal dual algorithm with competitive ratio $\frac{e(B)}{e(B)-1} \leq \frac{e}{e-1} \approx 1.582$. This competitive ratio is in fact the best possible (e.g., Buchbinder and Naor [6]).

As a concluding remark of the section, we highlight that the derivation of the online primal dual algorithm for the ski-rental problem and its analysis follow mechanically from the primal and dual linear programs and the corresponding relaxed complementary slackness conditions. No cleverness is needed to derive the optimal competitive ratio. This is, in my opinion, the main strength of the online primal dual framework. In more complicated problems, it is non-trivial to obtain a good enough understanding of the mathematical programs and to find the right relaxation of their optimality conditions. Once we figure them out, however, the design of the algorithm and the analysis will again become mechanical.

3 Online Primal Dual for Convex and Concave Objectives

Let us first introduce some necessary background on the conjugates of convex and concave functions and a duality theory for convex programs.

3.1 Conjugates

Let $f : \mathbb{R}_+^n \mapsto \mathbb{R}_+$ be a convex function. Its *convex conjugate* is defined as:

$$f^*(x^*) = \max_{x \geq 0} \{ \langle x, x^* \rangle - f(x) \} .$$

Here, x^* is also an n -dimensional vector. $\langle x, x^* \rangle$ denotes the inner product of vectors x and x^* . For example, suppose $f(x) = \frac{1}{\alpha} x^\alpha$ is a polynomial. Then, $f^*(x^*) = (1 - \frac{1}{\alpha}) x^{*\frac{\alpha}{\alpha-1}}$ is also a polynomial. Here, we add the coefficient $\frac{1}{\alpha}$ to f so that the coefficient of the conjugate f^* is simple, without changing the nature of the functions. We will have similar treatments throughout the chapter.

For simplicity, we will further assume that f is non-negative, non-decreasing, strictly convex, differentiable, and normalized such that $f(0) = 0$ in this chapter. In this case, the conjugate satisfies the following properties:

- f^* is non-negative, non-decreasing, strictly convex, differentiable, and normalized such that $f^*(0) = 0$;
- $f^{**} = f$;
- ∇f and ∇f^* are inverse of each other, and we say that x and x^* form a complementary pair if $x = \nabla f^*(x^*)$ and $x^* = \nabla f(x)$.

Next, consider a concave function $g : \mathbb{R}_+^n \mapsto \mathbb{R}$. Its *concave conjugate* is defined similarly as follows:

$$g_*(x^*) = \min_{x \geq 0} \{ \langle x, x^* \rangle - g(x) \} .$$

Similar to the convex case, we will further assume g to be non-negative, non-decreasing, strictly concave, differentiable, and normalized such that $g(0) = 0$ in this chapter. In this case, the concave conjugate satisfies the following properties:

- g_* is non-positive, non-decreasing, strictly concave, and differentiable;
- $g_{**} = g$;
- ∇g and ∇g_* are inverse of each other, and we say that x and x^* form a complementary pair if $x = \nabla g_*(x^*)$ and $x^* = \nabla g(x)$.

Assuming some mild conditions which hold for all problems in this chapter and, hence, are omitted, the following strong duality holds. It is known as Fenchel’s duality theorem.

$$\underset{x \geq 0}{\text{minimize}} \{ f(x) - g(x) \} = \underset{x^* \geq 0}{\text{maximize}} \{ g_*(x^*) - f^*(x^*) \} . \tag{3}$$

3.2 An Example: Online Auction of an Item with Production Cost

In the rest of the chapter, we will restrict our attentions to convex programs with linear constraints. In this section, we will consider an online auction of one item with production cost as a simple running example to demonstrate the application of Fenchel’s duality as well as how the online primal dual technique works.

Let there be a seller with one item for sale. Let there be n buyers who arrive online. Each buyer i has a value $v_i \in \mathbb{R}_+$ that specifies the maximum price i is willing to pay for a copy of the item. The technique can actually handle much more general settings with multiple heterogeneous items and combinatorial valuations of agents. We consider this simple case an illustrative example in this section, and refer readers to Huang and Kim [13] for further discussions on the general case. For simplicity, let us omit the strategic behaviors of buyers and assume that the

valuation of each buyer is revealed to the seller at the buyer's arrival. On the arrival of a buyer i , the seller decides whether to allocate a copy of the item to buyer i or not. The seller may produce an arbitrary number of copies of the item subject to a production cost function f , i.e., producing y copies of the item leads to a production cost of $f(y)$. The goal is to maximize the social welfare, i.e., the sum of values of the buyers for the allocated bundle of items less the production cost.

Below is a natural convex program relaxation of the online combinatorial auction with production costs. Here, we assume for simplicity that f is defined for all non-negative real numbers. Readers may think of it as, e.g., $f(y) = \frac{1}{2}y^2$, for concreteness.

$$\begin{aligned} & \text{maximize} && \sum_{i=1}^n v_i x_i - f(y) \\ & \text{subject to} && \sum_{i=1}^n x_i = y \\ & && 0 \leq x_i \leq 1 && i \in [n] \\ & && y \geq 0 \end{aligned}$$

Here, x_i is the indicator of whether buyer i gets a copy of the item, and y is the total number of allocated copies.

The Fenchel's dual convex program can be derived from the Lagrangian dual and the definition of convex conjugates. Taking the Lagrangian dual, we have

$$\text{minimize}_{u \geq 0, p} \text{maximize}_{x, y \geq 0} \sum_{i=1}^n v_i x_i - f(y) + \sum_{i=1}^n u_i (1 - x_i) + p(y - \sum_{i=1}^n x_i)$$

First, consider the maximization problem w.r.t. x_i , namely

$$\text{maximize}_{x_i \geq 0} (v_i - u_i - p) \cdot x_i = \begin{cases} 0 & \text{if } u_i + p \geq v_i, \\ +\infty & \text{otherwise.} \end{cases}$$

Thus, it imposes a linear constraint $u_i + p \geq v_i$ in the dual problem.

Next, consider the maximization problem w.r.t. y , namely

$$\text{maximize}_{y \geq 0} py - f(y)$$

By the definition of convex conjugates, the optimal value of the above maximization problem is $f^*(p)$.

In sum, the Lagrangian dual can be simplified as follows:

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n u_i + f^*(p) && (4) \\ & \text{subject to} && u_i + p \geq v_i && i \in [n] \\ & && u_i \geq 0 && i \in [n] \end{aligned}$$

We leave it to interested readers to verify the above program is equivalent to the Fenchel's dual as defined in Equation (3) of the online auction of an item with production cost. In this dual program, we can interpret p as the price for a copy of the item and u_i as the utility of buyer i , i.e., his value for the allocated bundle less the total price.

3.3 *Optimality Conditions*

Similar to their counterparts for linear programs, the online primal dual algorithms for convex programs are also driven by the optimality conditions of the programs and their dual programs. There are different ways to formulate such conditions. We will use the most familiar one known as the Karush–Kuhn–Tucker (KKT) conditions [14, 15]. We refer readers to Boyd and Vandenberghe [5] for an extensive discussion on the optimality conditions of convex programs. In this chapter, we will explain the conditions only on a problem-by-problem basis. For the running example of online auction of an item with production cost, the conditions are

- (a) x_i must be zero unless $u_i + p = v_i$;
- (b) u_i must be zero unless $x_i = 1$;
- (c) y and p form a complementary pair.

Here, the first two conditions concern primal/dual linear constraints and the corresponding dual/primal variables. They are complementary slackness conditions just like in the case of linear programs. The third condition is about variables involved in the non-linear parts of the primal and dual objectives. It states that they must form complementary pairs in the sense that we defined at the beginning of the section. Next, we will show how one can derive an online algorithm from the principle of satisfying these conditions approximately.

3.4 *High-Level Plan*

A meta online primal dual algorithm, much like their counterparts for linear programs, proceeds as follows. It maintains a feasible dual at all time. At the beginning, it is just the value of p since none of the u_i 's has arrived yet. On the arrival of a buyer i , it decides whether to allocate a copy of the item to i , i.e., the value of x_i , based on the current dual, sets values to the new dual variable u_i , and updates dual variable p . The high-level principle guiding these decisions is to satisfy the aforementioned optimality conditions as much as possible. We shall elaborate how shortly. Finally, the competitive ratio follows by comparing the increments in the primal and dual objectives in each step.

3.5 (Approximate) Complementary Slackness

Recall that the KKT conditions for linear constraints are the same as complementary slackness. An online primal dual algorithm handles these conditions the same way as in the original approach for linear programs.

In particular, let us consider conditions (a) and (b) in our running example. On buyer i 's arrival, a new dual variable u_i shows up and it is subject to two constraints $u_i + p \geq v_i$ and $u_i \geq 0$. Therefore, if $p > v_i$, the first constraint cannot hold with equality and x_i must be 0, i.e., we must not allocate a copy of the item to i , according to condition (a). On the other hand, if $p < v_i$, the value of u_i must be positive, and x_i must be 1, i.e., we must allocate a copy of the item to i , according to condition (b). The decision in the tie-breaking case when $v_i = p$ does not matter; we will assume that the algorithm does allocate a copy in the tie-breaking case. To this end, the algorithm shall interpret the current value of the dual variable p as a take-it-or-leave-it price for a copy of the item. It allocates a copy to buyer i if and only if its value v_i is at least the price p . Further, it shall let $u_i = \max\{v_i - p, 0\}$. Doing so will satisfy the complementary slackness conditions (a) and (b) at the end of i 's arrival. However, the algorithm may increase p in the future at which point condition (a) will be violated. Condition (b), on the other hand, will be satisfied exactly since x_i and u_i will not change in the future.

3.6 (Approximate) Complementary Pairs

So far, we have pinned down how to decide the allocation, i.e., the value of x_i , at each buyer i 's arrival based on the current dual, in particular the current price p . We have also explained how to set u_i accordingly. There is still a missing piece, namely how to update the price p , after i 's arrival.

This last piece of the algorithm is driving by the last condition. Recall it says that in the offline primal and dual solutions, y and p form a complementary pair, i.e., $p = f'(y)$. This coincides with the economic intuition, namely the unit price of a copy of the item shall equal to its marginal production cost. In the online problem, however, the algorithm knows only the current demand y of the item, but not its final value at the end of the day. As a result, the algorithm needs to in some sense predict the final demand according to the current demand, and set the price according to the predicted final demand. For example, we may simply predict the final demand to be twice the current demand. It turns out this simple heuristic already gives reasonably good competitive ratio for nice production cost functions such as polynomials. For simplicity of exposition, we will use a slight variant of this simple heuristic and assume a specific production function $f(y) = \frac{1}{2}y^2$ to explain the primal dual analysis in the rest of the section.

3.7 Online Primal Dual Algorithms

Putting together, consider the following online primal dual algorithm:

1. Initially, $y = 0$, $p = f'(2(y + 1)) = 2$.
2. On the arrival of each buyer i :
 - 2a. Let $x_i = 1$ if $v_i \geq p$, and let $x_i = 0$ otherwise.
 - 2b. Let $u_i = \max\{v_i - p, 0\}$.
 - 2c. Update $y = y + x_i$ and, subsequently, $p = f'(2(y + 1)) = 2(y + 1)$.

We remark that a more principled approach is to leave the prediction mapping as an unknown function g , i.e., let $g(y)$ be the predicted final demand if the current demand is y and set $p = f'(g(y))$, and to derive the optimal prediction mapping from the analysis. Interested readers are referred to Huang and Kim [13] for the details.

3.8 Primal Dual Analysis

Recall that $f(y) = \frac{1}{2}y^2$. Hence, we have $f^*(p) = \frac{1}{2}p^2$. Clearly, the algorithm maintains feasible primal and dual at all time by design. It remains to compare the increments of the objectives due to the arrival of each buyer i .

If i does not get a copy of the item, both the primal and the dual objectives remain the same. So the increments are both zero and, thus, equal.

If i does get a copy of the item, the primal objective changes by v_i , the gain from allocating a copy to i , less $f(y + 1) - f(y) = y + \frac{1}{2}$, the marginal cost of producing an extra copy of the item, where y denotes the demand for the item before i 's arrival. The changes in the dual, on the other hand, equals the utility of i , $u_i = v_i - p = v_i - 2(y + 1)$, plus the change due to the update of price p , $f^*(2(y + 2)) - f^*(2(y + 1)) = 4y + 6$. We claim that the increment in the dual objective is at most 4 times that in the primal objective. That is,

$$4(v_i - (y + \frac{1}{2})) \geq v_i - 2(y + 1) + (4y + 6)$$

By the coefficient of v_i and that $v_i \geq p = 2(y + 1)$ (otherwise, i would not have gotten a copy of the item), it suffices to show the inequality for $v_i = 2(y + 1)$. In that case, the inequality becomes

$$4y + 6 \geq 4y + 6$$

and holds with equality.

Note that the initial value of the primal objective is 0 while that of the dual objective is 2 because the initial value of $p = 2$. We have that:

$$4 \cdot \text{Primal} \geq \text{Dual} - 2$$

So the online primal dual algorithm is 4-competitive.

4 Application: Online Matching with Concave Returns

In this section, we will talk about the problem of online matching with concave returns. The problem and its analysis were originally from Devanur and Jain [11]. To be consistent with the other parts of the chapter, our exposition of the results and techniques will be somewhat different from the original version. Nonetheless, the core maths and the underlying ideas are the same.

4.1 Problem Definition

Let there be m agents (left-hand side offline vertices) that are known upfront. Let there be n items (right-hand side online vertices) that arrive online. On the arrival of an item, the online algorithm must immediately decide which agent shall get the item. Let v_{ij} denote agent j 's value for item i . Let there be a concave, non-decreasing function $g : \mathbb{R}_+ \mapsto \mathbb{R}_+$ such that an agent j 's value for getting a bundle S of items is $g(\sum_{i \in S} v_{ij})$. We shall interpret g as a discount function of an agent's value for bundles of items. Further, we do not discount the value if no items have been allocated to the agent so far, i.e., $g'(0) = 1$. This is without loss of generality up to scaling of the g function. The goal is to allocate the items to the agents to maximize the sum of values of all agents.

We remark that the online primal dual technique can in fact handle a more general version where different agents have different discount functions. We will omit the general version in this chapter for simplicity of discussion. Interested readers are referred to Devanur and Jain [11].

Next, recall that the concave conjugate of g is denoted as g_* . We give a convex program relaxation of the problem and its dual program below:

$$\begin{array}{ll}
 \text{maximize} & \sum_{j=1}^m g(y_j) \\
 \text{subject to} & \sum_{i=1}^n x_{ij} = 1 & i \in [n] \\
 & y_j = \sum_{i=1}^n v_{ij} x_{ij} & j \in [m] \\
 & x_{ij}, y_j \geq 0 & i \in [n], j \in [m] \\
 \\
 \text{minimize} & \sum_{i=1}^n \alpha_i - \sum_{j=1}^m g_*(\beta_j) \\
 \text{subject to} & \alpha_i \geq v_{ij} \beta_j & i \in [n], j \in [m]
 \end{array}$$

Here, x_{ij} is the indicator of whether item i is allocated to agent j . y_j denote the sum of values of the items allocated to agent j . We will focus on solving the fractional problem. That is, x_{ij} 's may take fractional values between 0 and 1.

4.2 Relaxed KKT Conditions

The online primal dual algorithms are driven by the optimality conditions of the primal and dual programs. We list below the relaxed conditions.

- (a) x_{ij} must be zero unless $\alpha_i = v_{ij}\beta_j$ when item i comes;
- (b) $\beta_j = \beta(y_j)$ is a function of the current total value y_j .

Before moving on to the design of online algorithms based on these conditions, let us briefly discuss how they differ from the exact versions and why relaxations are necessary in the online setting. The exact version of (a) requires the condition to hold at the end of the algorithm. In the online problem, however, the constraint may gain slack in the future because the algorithm may decrease β_j 's in the future. The exact version of (b) requires that y_j and β_j form a complementary pair. Similar to the previous example of online auction with production cost, instead of setting β_j based on the current value of y_j , a smarter algorithm shall anticipate y_j to further increase in the future and predict its final value. Hence, we set $\beta_j = \beta(y_j)$ to be a function of y_j where the function will be chosen based on the analysis.

4.3 Online Primal Dual Algorithms

The algorithm maintains a pair of feasible primal and dual at all time. Initially, the primal has only the y_j 's which will be set to 0. The dual has only the β_j 's which will be set to $\beta(0)$ according to condition (b).

Then, on the arrival of an item i , the relaxed optimality conditions (a) and (b) determine the allocation of the item. Concretely, conditions (a) and (b), together with the need of maintaining a feasible dual, indicate that i shall be allocated to the agent j such that $v_{ij}\beta_j$ is maximized. Here, β_j serves as a discount factor of the value of allocating an item to agent j . Therefore, it makes sense to have the discount function start from $\beta(0) = g'(0) = 1$ and be non-increasing in the agent's total value. Once the item is allocated to an agent, the algorithm updates y_j accordingly. Finally, we need to update the dual variables β_j as a result of the change in y_j .

The actual algorithm is slightly different from the above in that it allocates each item fractionally to multiple agents since we consider the fractional problem. We present the algorithm below.

1. Initially, let $y_j = 0$ and $\beta_j = \beta(y_j) = 1$ for $j = 1, \dots, m$.
2. Maintain $\beta_j = \beta(y_j)$ and $y_j = \sum_{i=1}^n v_{ij}x_{ij}$ throughout.
3. On the arrival of each item i , initialize all x_{ij} 's to be zero, and continuously increase x_{ij} 's as follows until $\sum_{j=1}^m x_{ij} = 1$:
 - 3a. Let $j^* = \arg \max_{j \in [m]} v_{ij}\beta_j$, breaking ties arbitrarily.
 - 3b. Increase x_{ij^*} by dx and α_i by $v_{ij^*}\beta_{j^*} \cdot dx$.
 - 3c. Update y_j 's and, thus, β_j 's accordingly.

4.4 Online Primal Dual Analysis

Again, since the algorithm maintains a pair of feasible primal and dual assignments at all time, it remains to compare the changes in the primal and dual objectives due to the arrival of each item. In particular, we would like to show that as the algorithm continuously increases x_{ij} 's in step 3 of the algorithm, it always holds that $F \cdot dP \geq dD$ for some fixed parameter $F \geq 1$, where dP and dD are the changes of the primal and dual objectives, respectively.

When the algorithm increases x_{ij^*} by some infinitesimal amount dx in step 3b (recall that j^* is the agent with the largest discounted value $v_{ij}\beta_j$), by simple calculus, the primal objective increases by:

$$dP = g'(y_{j^*})dy_{j^*}$$

The dual objective, on the other hand, increases by:

$$dD = \underbrace{v_{ij^*}\beta_{j^*}dx}_{\text{due to } \alpha_i} - \underbrace{g'_*(\beta_{j^*})d\beta_{j^*}}_{\text{due to } \beta_{j^*}} = \beta_{j^*}dy_{j^*} - g'_*(\beta_{j^*})d\beta_{j^*}$$

Therefore, to ensure $F \cdot dP \geq dD$, it boils down to choosing a non-increasing function β such that the following differential equation is feasible for the smallest possible $F \geq 1$:

$$\forall y \geq 0 : \quad F \cdot g'(y) \cdot dy \geq \beta(y) \cdot dy - g'_*(\beta(y)) \cdot d\beta(y) \quad (5)$$

subject to the boundary conditions that $\beta(0) = g'(0) = 1$ and $\beta(y) \geq 0$ for all $y \geq 0$.

Then, we have recovered the main algorithmic result in Devanur and Jain [11].

Theorem 1 *Suppose there is a non-increasing function β that satisfies the differential equation (5) for some $F \geq 1$. Then, there is an F -competitive online primal dual algorithm for the online matching problem with concave return function g .*

4.5 An Example: Additive Agents with Budgets

Differential equation (5) may look mysterious to the readers. For concreteness, we next present the optimal solution to the differential equation for a concrete example, namely $g(y) = \min\{y, B\}$ or some $B > 0$. This corresponds to the Adwords problem, where the agents are the advertisers and the items are ad slots. An advertiser has a budget B that caps its contribution to the seller's revenue. Again, one may consider a more general case when different advertisers have different budgets, which can be solved under the same framework. We will assume equal budget for the simplicity of our discussions.

In this case, the conjugate function g_* is

$$g_*(y^*) = \begin{cases} B(y^* - 1) & \text{if } y^* \leq 1 \\ 0 & \text{if } y^* > 1 \end{cases}$$

Therefore, in the special case, the differential equation becomes the following (recall that $\beta(0) = 1$ and therefore $\beta(y) \leq 1$ for all y):

$$\forall 0 \leq y \leq B : F \cdot dy \geq \beta(y) \cdot dy - B \cdot d\beta(y)$$

and

$$\forall y > B : 0 \geq \beta(y) \cdot dy - B \cdot d\beta(y)$$

with boundary conditions that $\beta(0) = 1$ and $\beta(y) \geq 0$ for all $y \geq 0$.

Note that β is non-increasing. It implies that the second part concerning $y > B$ is feasible only when $\beta(y) = 0$ for all $y = B$.

Next, we solve for the best β to satisfy the first part for $0 \leq y \leq B$ for the smallest possible F , subject to the boundary condition that $\beta(0) = 1$ and $\beta(B) = 0$. It is easy to check that it suffices to consider the differential equation with equality. Rearrange terms, it becomes

$$\forall 0 \leq y \leq B : (F - \beta(y)) \cdot dy = B \cdot d(F - \beta(y))$$

Together with $\beta(0) = 1$, we get that:

$$F - \beta(y) = (F - 1) \cdot e^{y/B}$$

Plugging in $\beta(B) = 0$, we get that

$$F = F - \beta(B) = (F - 1) \cdot e$$

which implies $F = \frac{e}{e-1}$. This is the best possible competitive ratio as it matches known hardness results from previous work (e.g., [8, 16]).

The corresponding discount function β that achieves this optimal ratio is

$$\beta(y) = \frac{1}{e-1}(e - e^{y/B})$$

As a concluding remark, the primal dual technique gives optimal competitive ratio not only for the above example of additive agents with budgets, but in fact for arbitrary concave functions g . The cool thing about it is that Devanur and Jain [11] established the optimality of the algorithm without explicitly giving the optimal competitive ratio, other than the characterization using the differential equation (5). Instead, they constructed a family of hard instances that are parameterized by a function and showed that solving for the worst function is equivalent to solving the

differential equation (5). This chapter focuses on positive results and will not get into further details of the hardness. Readers are referred to Devanur and Jain [11].

5 Application: Online Scheduling with Speed Scaling

This section considers another application of the online primal dual framework in online optimization problems with non-linear objectives. We will consider an online scheduling problem with speed-scalable machines, where the energy consumption is naturally a convex function of the speed. The results were originally from Devanur and Huang [10], which built on an earlier dual fitting approach by Anand et al. [2].

5.1 Problem Definition

Let there be m machines that are known upfront. Let there be n jobs that arrive online. Each machine j can run at different speeds subject to different energy consumptions. Let f be the power function such that running a machine at speed y consumes $f(y)$ energy per unit of time. For concreteness, readers may think of $f(y) = \frac{1}{3}y^3$ as a typical cubic energy function. Each job i is defined by its arrival time r_i , and its volumes v_{ij} when allocated to each machine j . That is, we consider the unrelated machine setting in which different machines may take different amount of computational resources to complete the same job. A feasible schedule allocates each job to a machine and more specifically a subset of time slots on the machine, and specifies how fast the machine runs in each time slot, such that the total computational resources assigned to the job, i.e., the sum of speeds of the allocated time slots, are equal to its volume on the machine.

Again, the primal dual framework can in fact handle a more general setting where (1) jobs have weights and (2) machines may have different energy functions. We will omit such generalizations and refer interested readers to Devanur and Huang [10].

There are two natural objectives for this problem. The first one is to minimize total energy consumption, and the second is to minimize the total delay experienced by the jobs. Here, there are different ways to define delay. We will consider a popular one known as flow time. The flow time of a job is the difference between its completion time and arrival time. The flow time of a feasible schedule is the sum of the flow time of all jobs. Minimizing either objective without considering the other is trivial. One could get arbitrarily small flow time by running the machines at extremely high speed, but paying an arbitrarily high energy consumption. One could also have arbitrarily small energy consumption by running the machines at extremely low speed, but suffering an arbitrarily large flow time. We will use the standard combination of minimizing the sum of flow time and energy.

This problem is more complicated than the previous examples in the sense that the online algorithm needs to make multiple types of online decisions. At each time

slot, the online algorithm must decide which job shall be run on each machine (*job selection*) and at what speed (*speed scaling*). Further, on the arrival of a job, the online algorithm must immediately dispatch the job to one of the machines (*job assignment*). We consider the preemptive model in which the algorithm may preempt the current job processing on a machine with the new job and resume after the new job is finished.

The convex program relaxation and its dual are given below.

$$\begin{aligned}
 &\text{minimize} && \sum_{i=1}^n \sum_{j=1}^m \int_{r_i}^{+\infty} (t - r_i)(x_{ijt}/v_{ij})dt \\
 & && + \sum_{j=1}^m \int_0^{+\infty} f(y_{jt})dt \\
 & && + \sum_{i=1}^n \sum_{j=1}^m \int_{r_i}^{+\infty} (f^*)^{-1}(1)x_{ijt}dt \\
 &\text{subject to} && \sum_{j=1}^m \int_{r_i}^{+\infty} (x_{ijt}/v_{ij})dt = 1 && i \in [n] \\
 & && y_{jt} = \sum_{i=1}^n x_{ijt} && j \in [m], t \geq 0 \\
 & && x_{ijt}, y_{jt} \geq 0 && i \in [n], j \in [m], t \geq 0 \\
 \\
 &\text{maximize} && \sum_{i=1}^n \alpha_i - \sum_{j=1}^m \int_0^{+\infty} f^*(\beta_{jt})dt \\
 &\text{subject to} && \alpha_i \leq v_{ij}\beta_{jt} + t - r_i + v_{ij}(f^*)^{-1}(1) && i \in [n], j \in [m], t \geq r_i
 \end{aligned}$$

Here, x_{ijt} is the speed of machine j at time t that is devoted to processing job i . y_{jt} is the total speed of machine j at time t . The first primal constraint is a relaxation of the feasibility constraint. Recall the original constraint says that each job must be completed on a machine j such that the sum of speed devoted to the job is equal to v_{ij} , i.e., $\int_{r_i}^{+\infty} x_{ijt}dt = v_{ij}$. The relaxed constraint, on the other hand, allows the job to be fractionally processed on different machines so long as the overall effort processes the entire job. The second primal constraint says that the speed of a machine at any time must be high enough to execute the schedule given by x_{ijt} 's.

The first term in the primal objective is a relaxation of flow time in the sense that the fraction of a job i that is completed at time t experiences a flow time defined by the current time, i.e., $t - r_i$, instead of the flow time given by the completion of the job. This is usually known as the fractional flow time of the schedule, which is widely used as an intermediate objective in the study of flow time minimization.

The second term is exactly the energy consumption.

It turns out that these two terms on their own suffer from a large integrability gap. Consider the case of having only one job arriving at time 0 with volume v on every machine. An integral schedule must process the job on the same machine and hence pays a non-zero flow time plus energy. Interested readers may verify that the minimum cost equals $v \cdot (f^*)^{-1}(1)$. A feasible schedule of the relaxed linear program, however, can split the workload and process a $1/m$ fraction of the job on each machine. Such a fractional schedule effectively pays 0 in the objective when m goes to infinity. Introducing the third term fixes this problem. Since the minimum

cost of assigning a job of volume v is $v \cdot (f^*)^{-1}(1)$, the extra unit cost of $(f^*)^{-1}(1)$ for the x_{ijt} volume of a job i that is processed on a machine j at time t sums to at most the actual flow time plus energy for any feasible integral schedule. Hence, adding the third term increases the optimal by at most a factor of 2.

5.2 A Simple Online Primal Dual Algorithm

As usual, we start with a set of relaxed optimality conditions that drive the design of the online primal dual algorithms.

- (a) If a job i is dispatched to a machine j and is tentatively scheduled at time t when it comes, it must be that $\alpha_i = v_{ij}\beta_{jt} + t - r_i + v_{ij}(f^*)^{-1}(1)$ (there may be some slack in the future because the algorithm may increase β_{jt} 's);
- (b) $\beta_{jt} = \beta(y_{jt})$ is a function of the current total speed y_t .

Condition (a) is a relaxation in the sense that the actual complementary slackness condition holds for the optimal primal and dual assignments at the end of the instance, while here it holds only at the moment when i comes. Again, this is unavoidable and shall look standard to the readers after seeing similar conditions in the previous two examples. Condition (b) is a relaxation of the complementary pair condition saying that y_{jt} and β_{jt} will form a complementary pair. Intuitively, the online algorithm will predict the final value of y_{jt} , i.e., the speed of machine j at time t , based on its current value, and set the dual variable β_{jt} to form a complementary pair with the predicted final value.

We present below the meta online primal dual algorithm driven by the above relaxed optimality conditions, given any function $\beta : \mathbb{R}_+ \mapsto \mathbb{R}_+$.

1. Initially, let $y_{jt} = 0$ and $\beta_{jt} = \beta(0)$ for $j = 1, \dots, m, t \geq 0$.
2. Maintain $\beta_{jt} = \beta(y_{jt})$ and $y_{jt} = \sum_{i=1}^n x_{ijt}$ throughout.
3. On the arrival of each job i :
 - 3a. Let j^* and t^* be such that $v_{ij^*}\beta_{j^*t^*} + t^* - r_i$ is minimized. Dispatch the job i to machine j^* and tentatively schedule it at time t^* .
 - 3b. Update $y_{j^*t^*}$'s and, thus, $\beta_{j^*t^*}$'s accordingly.

5.3 A Simple Greedy Algorithm

In this chapter, we will consider a particularly simple greedy algorithm and analyze it using the online primal dual framework.

The greedy algorithm processes the current jobs using the optimal schedule assuming that no other jobs will arrive in the future. In particular, given how the current jobs are assigned to the machines, the greedy algorithm processes the jobs on each machine from the shortest to the longest. The shortest job first principle is the best strategy assuming no future jobs will arrive. To see the intuition behind its

optimality, note that the processing time of the first job will contribute to the flow time of all remaining jobs, that of the second job will contribute to the flow time of all but one remaining jobs, and so on. Hence, it makes sense to prioritize on shorter jobs.

In terms of the choice of speed, it will run the machine at a speed that depends on the number of remaining jobs on the machines. Concretely, suppose there are k remaining jobs. Then, the speed y is set such that $f^*(f'(y)) = k$. To see why this is the right choice, suppose we process a job, say, with volume v , at speed y . Then, the amount of time needed to process the job is v/y . All k jobs will suffer from this delay in their flow time and, thus, the total contribution to the objective in terms of flow time is $(v/y) \cdot k$. On the other hand, the energy consumption during this period of time is $(v/y) \cdot f(y)$. Therefore, the optimal speed is the one that minimizes $(v/y) \cdot (k + f(y))$. Then, it is easy to verify that our choice of speed minimizes this quantity, which equals $v \cdot (f^*)^{-1}(k)$.

Finally, we will use a simple β mapping such that $\beta_{jt} = f'(y_{jt})$ forms a complementary pair with y_{jt} . When a new job arrives, the algorithm dispatches the job according to step 3a in the meta algorithm. By our choice of speed, for any machine j and any time t , we have $\beta_{jt} = (f^*)^{-1}(k)$ where k is the number of remaining jobs on machine j at time t . Hence, the right-hand side of the dual constraint becomes

$$\begin{aligned} v_{ij}\beta_{jt} + t - r_i + v_{ij}(f^*)^{-1}(1) &= t - r_i + v_{ij}((f^*)^{-1}(k) + (f^*)^{-1}(1)) \\ &\geq t - r_i + v_{ij}(f^*)^{-1}(k + 1) \end{aligned}$$

Here, the inequality follows by the concavity of $(f^*)^{-1}$, which is due to the convexity of f^* . Having a closer look at the right-hand side of the above inequality, the $t - r_i$ term corresponds to the flow time for having job i wait until time t before we process it. The last term, on the other hand, is the energy consumption of processing job i on machine j plus the delay it causes to the flow time of the existing k jobs and itself, provided that we run the machine with the aforementioned greedy speed. Hence, we shall interpret the right-hand side as the total increase in the objective if we decide to dispatch the job to machine j and tentatively schedule it at time t . Therefore, our job dispatch rule (step 3a) simply greedily minimizes this increment.

5.4 Online Primal Dual Analysis

Following the above discussion, we shall let α_i be the increment in the objective due to the arrival of job i . Recall that the dual is equal to

$$\sum_{i=1}^n \alpha_i - \sum_{j=1}^m \int_0^{+\infty} f^*(\beta_{jt}) dt$$

The first term, by definition, sums to the objective of flow time plus energy consumption since each α_i accounts for the increment in the objective due to the

corresponding job i . The second term, on the other hand, is equal to the flow time because $f^*(\beta_{jt}) = f^*(f'(y_{jt}))$ equals the number of remaining jobs on machine j at time t by our choice of y_{jt} and β_{jt} . Hence, the dual objective is in fact extremely simple for the greedy algorithm: It equals the energy consumption.

So far, we did not use any special property of the power function f . For arbitrary power functions, no competitive algorithms exist for this problem. Next, we focus on polynomial power functions $f(y) = \frac{1}{\alpha}y^\alpha$, where α is between 2 and 3 for typical power functions in practice. In this case, we have $f^*(\beta) = (1 - \frac{1}{\alpha})\beta^{\alpha/(\alpha-1)}$ and, thus, our choice of speed implies that $k = f^*(f'(y)) = (1 - \frac{1}{\alpha})y^\alpha$. Therefore, at any time, assuming there are k jobs remaining, the contribution (per unit of time) to flow time, i.e., $k = (1 - \frac{1}{\alpha})y^\alpha$ is exactly $\alpha - 1$ times the contribution to energy consumption, i.e., $f(y) = \frac{1}{\alpha}y^\alpha$. Therefore, the fact that the dual objective equals the energy consumption means that it is exactly a $1/\alpha$ fraction of the flow time plus energy of the algorithm. This implies the following theorem.

Theorem 2 *The greedy algorithm is $O(\alpha)$ -competitive for minimizing flow time plus energy when the power function is $f(y) = \frac{1}{\alpha}y^\alpha$.*

5.5 Better Online Primal Dual Algorithms

The main drawback of the simple algorithm lies in that it fails to anticipate future jobs and is conservative in its choice of speed. We can improve the greedy algorithm with a speed-up. In particular, the greedy algorithm achieves the optimal flow time plus energy if there are indeed no future jobs, but is far from optimal when there are a lot of future jobs. A smarter algorithm shall run faster than the optimal schedule of the remaining instance to hedge between the two cases: If there are no future jobs, this approach pays more in energy comparing to the simple greedy algorithm; if there are a lot of future jobs, however, it is better than the simple greedy algorithm because it has predicted the arrival of future jobs and has run faster in the past.

Specifically, Devanur and Huang [10] considered a family of such algorithms parameterized by a parameter $C \geq 1$ such that the algorithm runs C times faster than the optimal schedule of the remaining instance. Such an algorithm is called the C -aggressive greedy algorithm, as given below.

Speed Scaling:

- Choose speed s.t. $f^*(f'(\frac{y_{jt}}{C}))$ equals the number of remaining jobs.
- Let $\beta_{jt} = \frac{1}{C}f'(\frac{y_{jt}}{C})$ s.t. $f^*(C\beta_{jt})$ equals the number of remaining jobs.

Job Selection:

- Schedule jobs from the shortest to the longest.

Job Assignment:

- Dispatch the job to a machine that minimizes the increment in the objective.

Moreover, Devanur and Huang [10] showed an online primal dual analysis of the C -aggressive greedy algorithms for all $C \geq 1$ and chose C to balance the case with no future jobs and the case with a lot of future jobs. By doing so, they showed the following result. We refer readers to Devanur and Huang [10] for the proofs.

Theorem 3 *For polynomial power function $f(y) = \frac{1}{\alpha}y^\alpha$, the C -aggressive greedy online primal dual algorithm is $O\left(\frac{\alpha}{\log \alpha}\right)$ -competitive for minimizing flow time plus energy on unrelated machines with $C \approx 1 + \frac{\log \alpha}{\alpha}$.*

If there are at least two machines, then Devanur and Huang [10] further showed that the above competitive ratio is asymptotically tight.

Theorem 4 *For polynomial power function $f(y) = \frac{1}{\alpha}y^\alpha$, there are no $o\left(\frac{\alpha}{\log \alpha}\right)$ -competitive online algorithm for minimizing fractional flow time plus energy with at least two machines.*

Recall that in the previous example of online matching, the upper and lower bounds match exactly as they both reduce to the same differential equation. In this example of online scheduling for minimizing flow time plus energy, however, there is a constant gap between the upper and lower bounds. One of the reasons is that the algorithms in this example assign jobs to machines integrally, i.e., a job cannot be processed on multiple machines in parallel as in a fractional solution to the convex program relaxation. We do not know whether we can derive the same form of tight upper and lower bounds as in the previous two examples if we allow parallel processing and truly focus on solving the primal convex program online.

Further, if there is only one machine, Bansal et al. [4] gave a 2-competitive online algorithm for minimizing fractional flow time plus energy with arbitrary power functions using a potential function argument. It is an interesting open question whether there is a 2-competitive online primal dual algorithm for single machine.

Finally, essentially the same algorithm and analysis can be further used to derive better competitive ratios in the resource augmentation setting, where the online algorithm can run the machines $1 + \epsilon$ times faster than the offline benchmark using the same amount of energy.

Theorem 5 *There is a $(1 + \epsilon)$ -speed and $O\left(\frac{1}{\epsilon}\right)$ -competitive online primal dual algorithm for minimizing flow time plus energy with arbitrary power functions.*

6 Application: Online Covering and Packing Problems with Convex Objectives

As the last example of the chapter, we present a result by Azar et al. [3] that gave online algorithms for a large family of covering and packing problems with convex objectives. It generalizes the original online covering and packing problems that considers linear objectives (e.g., Buchbinder and Naor [7]).

6.1 Problem Definition

We study the following online covering and packing problems with convex objectives. The (offline) covering problem is modeled by the following convex program:

$$\min_{x \in \mathbb{R}_+^n} f(x) \text{ s.t. } Ax \geq 1$$

where $f : \mathbb{R}_+^n \mapsto \mathbb{R}_+$ is a monotone and convex cost function, and A is an $m \times n$ matrix with non-negative entries. Each row of the constraint matrix A corresponds to a covering constraint. In the online problem, the rows of A come online and the algorithm must maintain a feasible assignment x that is non-decreasing over time.

The (offline) packing problem, on the other hand, is modeled by the following convex program:

$$\max_{y \in \mathbb{R}_+^m} \sum_{j=1}^m y_j - f^*(A^T y)$$

where $f^* : \mathbb{R}_+^m \mapsto \mathbb{R}_+$ is the convex conjugate of f . It is the Fenchel's dual program of covering problem. In the online problem, each variable y_j arrives online and the algorithm must decide the value of y_j on its arrival.

When f is a linear function in the covering problem, e.g., $f(x) = \langle c, x \rangle$ for some positive vector c , f^* becomes a $0-\infty$ step function that imposes supply constraints $A^T y \leq c$ in the packing problem. Then, the problems become the original online covering and packing problems with linear objectives.

In the rest of the section, we will focus on the packing problem, and refer readers to Azar et al. [3] for the algorithms and the corresponding analysis for the covering problem. In other words, we will take the packing program as the primal program and the covering one as the dual.

6.2 Relaxed Optimality Conditions

As usual, we will start with a set of relaxed optimality conditions for the (primal) packing program and the (dual) covering program. Let $A_j = (a_{j1}, a_{j2}, \dots, a_{jn})$ denote the j -th row of matrix A .

- (a) y_j must be zero unless $A_j x = 1$ at the end of the round when covering constraint j and, thus, packing variable j , arrive.
- (b) $x = \nabla g(A^T y)$ for some convex function g .

Condition (a) is a relaxation of the actual complementary condition, which holds for the final value of x and y . Condition (b) is a relaxation of the condition that x and $A^T y$ form a complementary pair with respect to f and f^* , i.e., $x = \nabla f^*(A^T y)$. Just

like in the previous examples, a smart algorithm shall anticipate that the variables y and, thus, $A^T y$ will further increase in the future and set x accordingly.

6.3 Online Primal Dual Packing Algorithm

Driven by the above relaxed optimality conditions, we can define the following meta algorithm that depends on the choice of a convex function g .

1. Initialize $x = z = 0$.
2. Maintain $x = \nabla g(z)$ and $z = A^T y$ throughout.
3. When y_k and $A_k = (a_{k1}, \dots, a_{kn})$ arrive in round k , do the followings:
 - 2a. Initialize $y_k := 0$;
 - 2b. While $\sum_{i=1}^n a_{ki} x_i < 1$, continuously increase y_k and do the followings:
 - (2b.1) Simultaneously for each $i \in [n]$, increase z_i at rate $\frac{dz}{dy_k} = a_{ki}$.
 - (2b.2) Increase x according to $x = \nabla g(z)$.

Here, the vector x plays an auxiliary role and is initialized to 0. We can interpret x as a price vector such that x_i is the unit price of the i -th resource in the packing problem. Then, we keep including more of the k -th item into the packing solution as long as its unit gain of 1 can pay for the total price of the resources that it demands.

We will consider a particularly simple mapping function in this chapter:

$$g(z) = \frac{1}{\rho} f^*(\rho z) = \frac{1}{\rho} f^*(\rho A^T y)$$

and, thus, $x = \nabla g(z) = \nabla f^*(\rho z)$, for some parameter $\rho > 1$ to be determined in the analysis. Intuitively, this means that the algorithm anticipates $A^T y$ will further increase by a factor of ρ in the future, and pick x accordingly.

In round $k \in [m]$, the vector $a_k = (a_{k1}, a_{k2}, \dots, a_{kn})$ is revealed. The variable y_k is initialized to 0, and is continuously increased while $\sum_{i \in [n]} a_{ki} x_i < 1$, i.e., the total price for the amount of resources needed to produce a unit of resource k is less than 1. To maintain $z = A^T y$, for each $i \in [n]$, z_i is increased at rate $\frac{dz}{dy_k} = a_{ki}$. As the coordinates of z are increased, the vector x is increased according to the invariant $x = \nabla f^*(\rho z)$. We shall assume that ∇f^* is monotone. Hence, both x and z increase monotonically as y_k increases.

6.4 Online Primal Dual Analysis

For simplicity of our discussion, we will make an assumption that f^* is a homogeneous polynomial with non-negative coefficients (so that the gradient is monotone) and degree $\lambda > 1$. The competitive ratio will depend on λ .

We first show that the algorithm will not keep increasing y_k forever in any round k and it maintains a feasible primal solution throughout. Concretely, the following lemma indicates that unless the offline packing problem is unbounded, eventually $\sum_{i \in [n]} a_{ki} x_i$ will reach 1. At this moment, y_k will stop increasing and we complete round k . Recall that the coordinates of x increase monotonically as the algorithm proceeds. It implies that the covering constraints $\sum_{i \in [n]} a_{ji} x_i \geq 1$ are satisfied for all $j \in [m]$ at the end of the algorithm since each constraint is satisfied at the end of the corresponding round. Hence, the vector x is feasible for the dual covering problem.

Lemma 1 *Suppose that the offline optimal packing objective is bounded. Then, in each round $k \in [m]$, eventually we have $\sum_{i \in [n]} a_{ki} x_i \geq 1$, and y_k will stop increasing.*

Proof We have

$$\begin{aligned} \frac{\partial P(y)}{\partial y_k} &= 1 - \langle a_k, \nabla f^*(z) \rangle \\ &= 1 - \frac{1}{\rho^{\lambda-1}} \cdot \langle a_k, \nabla f^*(\rho z) \rangle \\ &= 1 - \frac{1}{\rho^{\lambda-1}} \cdot \langle a_k, x \rangle \\ &\geq 1 - \frac{1}{\rho^{\lambda-1}} \end{aligned}$$

Here, the first equality is due to $z = A^T y$, which indicates that when the algorithm increases y_k , it also increases each z_i at rate $\frac{dz_i}{dy_k} = a_{ki}$. The second equality follows by that f^* is a homogeneous polynomial of degree λ . The third equality is because of our choice of x . Finally, the last inequality follows because $\langle a_k, x \rangle < 1$ when y_k is increased by the definition of the algorithm.

Suppose for contrary that $\langle a_k, x \rangle$ never reaches 1. Then, the objective function $P(y)$ increases at least at some positive rate $1 - \frac{1}{\rho^{\lambda-1}}$ (recalling $\rho > 1$ and $\lambda > 1$) as y_k increases, which means the offline packing problem is unbounded, contradicting our assumption.

To complete the competitive analysis, it remains to compare the primal and dual objectives. In the rest of the section, for $k \in [m]$, we let $z^{(k)}$ denote the vector z at the end of round k , where $z^{(0)} := 0$. Let $P(y)$ denote the packing objective for any vector y and the induced vector $z = A^T y$. Similarly, let $C(x)$ denote the covering objective for any feasible covering solution x .

First, let us look into the contribution to the packing objective from y .

Lemma 2 *At the end of round k when y_k stops increasing (by Lemma 1)*

$$y_k \geq \frac{1}{\rho} \cdot (f^*(\rho z^{(k)}) - f^*(\rho z^{(k-1)})) .$$

In particular, since $f^*(0) = 0$, this implies that at the end of the algorithm,

$$\sum_{k \in [m]} y_k \geq \frac{1}{\rho} \cdot f^*(\rho z^{(m)}) .$$

Proof Recall again that y_k increases only when $\langle a_k, x \rangle < 1$, we have

$$1 \geq \sum_{i \in [n]} a_{ki} x_i = \sum_{i \in [n]} x_i \cdot \frac{dz_i}{dy_k} .$$

Hence, integrating this with respect to y_k throughout round k , and observing that $x = \nabla f^*(\rho z)$, we have

$$y_k \geq \int_{z=z^{(k-1)}}^{z^{(k)}} \langle \nabla f^*(\rho z), dz \rangle = \frac{1}{\rho} \cdot (f^*(\rho z^{(k)}) - f^*(\rho z^{(k-1)})) ,$$

where the last equality comes from the fundamental theorem of calculus for path integrals of vector fields.

Therefore, the dual objective is lower bounded by:

$$\frac{1}{\rho} f^*(\rho z^{(m)}) - f^*(z^{(m)}) = (\rho^{\lambda-1} - 1) f^*(z^{(m)})$$

Next, we consider the dual (covering) objective. Suppose $z^{(m)} = A^T y$ is the vector at the end of the algorithm, and $x^{(m)} = \nabla f^*(\rho z^{(m)})$. We have the following.

Lemma 3 *The covering objective satisfies that:*

$$C(x^{(m)}) = (\lambda - 1) \rho^\lambda f^*(z^{(m)})$$

Proof By how our algorithm maintains the covering solution, we have

$$\begin{aligned} C(x^{(m)}) &= f(\nabla f^*(\rho z^{(m)})) \\ &= \max_{z \geq 0} \{ \langle z, \nabla f^*(\rho z^{(m)}) \rangle - f^*(z) \} \\ &= \rho z^{(m)} \nabla f^*(\rho z^{(m)}) - f^*(\rho z^{(m)}) \\ &= (\lambda - 1) f^*(\rho z^{(m)}) \\ &= (\lambda - 1) \rho^\lambda f^*(z^{(m)}) , \end{aligned}$$

where the second equality follows by the definition of convex conjugate functions, the third equality is because the maximum is achieved at $z = \rho z^{(m)}$ by first order condition, the last two equalities are due to our assumption that f^* is a homogeneous polynomial of degree λ .

Putting together our bounds on the packing and covering objectives, we conclude that they are with the following bounded factor from each other:

$$\frac{C(x)}{P(y)} \leq \frac{(\lambda - 1)\rho^\lambda}{\rho^{\lambda-1} - 1}, \quad (6)$$

Choosing $\rho := \lambda^{\frac{1}{\lambda-1}}$, the above ratio becomes

$$\lambda^{\frac{\lambda}{\lambda-1}} = O(\lambda)$$

Hence, we conclude that:

Theorem 6 *Suppose f^* is a convex, homogeneous polynomial with non-negative coefficients and degree λ . Then, there is an $O(\lambda)$ -competitive online algorithm for the online packing problem with a convex objective defined with f^* .*

References

1. Agrawal, S., Devanur, N.R.: Fast algorithms for online stochastic convex programming. In: Proceedings of the 26th Annual Symposium on Discrete Algorithms. SIAM, Philadelphia (2015)
2. Anand, S., Garg, N., Kumar, A.: Resource augmentation for weighted flow-time explained by dual fitting. In: Proceedings of the 23rd Annual Symposium on Discrete Algorithms, pp. 1228–1241. SIAM, Philadelphia (2012)
3. Azar, Y., Buchbinder, N., Chan, T.H., Chen, S., Cohen, I.R., Gupta, A., Huang, Z., Kang, N., Nagarajan, V., Naor, J., Panigrahi, D.: Online algorithms for covering and packing problems with convex objectives. In: 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), pp. 148–157. IEEE, Piscataway (2016)
4. Bansal, N., Chan, H.L., Pruhs, K.: Speed scaling with an arbitrary power function. In: Proceedings of the 20th Annual symposium on discrete algorithms, pp. 693–701. SIAM, Philadelphia (2009)
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
6. Buchbinder, N., Naor, J.: The design of competitive online algorithms via a primal: dual approach. Found. Trends Theor. Comput. Sci. **3**(2–3), 93–263 (2009)
7. Buchbinder, N., Naor, J.: Online primal-dual algorithms for covering and packing. Math. Oper. Res. **34**(2), 270–286 (2009)
8. Buchbinder, N., Jain, K., Naor, J.S.: Online primal-dual algorithms for maximizing ad-auctions revenue. In: Algorithms–ESA, pp. 253–264. Springer, Berlin (2007)
9. Dantzig, G.B.: Linear Programming and Extensions. Princeton University Press, Princeton (1998)
10. Devanur, N.R., Huang, Z.: Primal dual gives almost optimal energy efficient online algorithms. In: Proceedings of the 25th Annual Symposium on Discrete Algorithms, pp. 1123–1140. SIAM, Philadelphia (2014)
11. Devanur, N.R., Jain, K.: Online matching with concave returns. In: Proceedings of the 44th Annual Symposium on Theory of Computing, pp. 137–144. ACM, New York (2012)

12. Gupta, A., Krishnaswamy, R., Pruhs, K.: Online primal-dual for non-linear optimization with applications to speed scaling. In: *Approximation and Online Algorithms*, pp. 173–186. Springer, Berlin (2013)
13. Huang, Z., Kim, A.: Welfare maximization with production costs: a primal dual approach. In: *Proceedings of the 26th Annual Symposium on Discrete Algorithms*. SIAM, Philadelphia (2015)
14. Karush, W.: Minima of functions of several variables with inequalities as side constraints. PhD thesis, Master's Thesis, Department of Mathematics, University of Chicago (1939)
15. Kuhn, H.W., Tucker, A.W.: Nonlinear programming. In: *Proceedings of 2nd Berkeley Symposium* (1951)
16. Mehta, A., Saberi, A., Vazirani, U., Vazirani, V.: Adwords and generalized online matching. *J. ACM* **54**(5), 22 (2007)
17. Nguyen, K.T.: Lagrangian duality in online scheduling with resource augmentation and speed scaling. In: *Algorithms–ESA*, pp. 755–766. Springer, Berlin (2013)