Online Algorithms for Packing and Covering Problems with Convex Objectives[1]

Yossi Azar[2] Niv Buchbinder[3] T-H. Hubert Chan[4] Shahar Chen[5]
Ilan Reuven Cohen[2] Anupam Gupta[6] Zhiyi Huang[4] Ning Kang[4]
Viswanath Nagarajan[7] Joseph (Seffi) Naor[5] Debmalya Panigrahi[8]

Abstract

We study the online convex covering problem and online convex packing problem. The (offline) convex 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, each row of A comes online and the algorithm must maintain a feasible assignment x and may only increase x over time. The (offline) convex packing problem is modeled by the following convex program:

$$\max_{y\in\mathbb{R}^m_+} \textstyle\sum_{j=1}^m y_j - g(A^T y)$$

where $g : \mathbb{R}^n_+ \mapsto \mathbb{R}_+$ is a monotone and convex cost function. It is the Fenchel dual program of convex covering when g is the convex conjugate of f. In the online problem, each variable y_j arrives online and the algorithm must decide the value of y_j on its arrival.

[The result achieved]

---

[1]This paper contains results from three concurrent and independent manuscripts by Azar, Cohen, and Panigrahi [**?**], by Buchbinder, Chen, Gupta, Nagarajan, and Naor [**?**], and by Chan, Huang, and Kang [**?**].

[2]Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel. Email: {azar@tau.ac.il, ilanrcohen@gmail.com}. Research supported in part by ISF grant 1404/10 and by the I-CORE Center 4/11.

[3]Department of Statistics and Operations Research, Tel Aviv University, Tel Aviv, Israel. Email: niv.buchbinder@gmail.com. Research supported in part by ISF grant 954/11 and by BSF grant 2010426.

[4]Department of Computer Science, The University of Hong Kong, Hong Kong, China. Email: {hubert,zhiyi,nkang}@cs.hku.hk.

[5]Department of Computer Science, Technion - Israel Institute of Technology, Haifa, Israel. Email:{shahar.chen11@gmail.com, naor@cs.technion.ac.il}. Research supported in part by ISF grant 954/11 and BSF grant 2010426.

[6]Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Email: anupamg@cs.cmu.edu. Research supported in part by NSF awards CCF-1016799 and CCF-1319811.

[7]Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109, USA. Email: viswa@umich.edu.

[8]Department of Computer Science, Duke University, Durham, NC 27708, USA. Email: debmalya@cs.duke.edu. Research supported in part by a Duke University startup grant and a Google Faculty Research Award.

1 Introduction

We study here a very general online primal-dual framework that captures many existing and new applications. The primal problem is the following covering problem.

$$\min_{x \in \mathbb{R}^n_+} \; C(x) := f(x) \;\text{ subject to }\; Ax \geq \mathbf{1} \tag{1.1}$$

Above, $f : \mathbb{R}^n_+ \to \mathbb{R}$ is a monotone[1] convex function and $A_{m \times n}$ is non-negative. Let $f^\star$ be the Fenchel dual function of f[2]. The (Fenchel) dual problem of (1.1) is the following packing problem:

$$\max_{y \in \mathbb{R}^m_+} \; P(y) := \textstyle\sum_{j \in [m]} y_j - f^\star\left(A^T y\right) \tag{1.2}$$

In the online setting the rows of A, the covering constraints, arrive over time. This corresponds in the dual problem to an arrival of a new dual variable y_i along with its coefficients. The requirement in the primal setting is that the algorithm maintains a feasible fractional solution x, where x is required to be non-decreasing over time. In the dual setting the algorithm must decide the value of y_i on its arrival, and cannot change it later on. This framework generalizes the successful primal-dual framework [**?**] that currently only captures the case that $f(x)$ is a non-negative linear function.

This general framework captures a much broader class of problems. Consider, e.g., the mixed covering-packing problem from [**?**]. Here general covering constraints arrive online. There are also K "packing constraints" of the form $\sum_{j=1}^n b_{kj} \cdot x_j \leq \lambda_k$, for $k \in [K]$, that are given up-front. The entries b_{kj} are non-negative. The right hand sides λ_k of these packing constraints are themselves variables, and the objective is to minimize $\max_{k=1}^K \lambda_k$. By replacing the maximum by a "soft-max", the problem is to minimize $f(x) = \sum_{k=1}^K \left(\sum_{j=1}^n b_{kj} \cdot x_j\right)^p$. Clearly, the objective function is a monotone non-decreasing convex function, and we can use the covering part of our framework to give improved algorithms for it.

For the dual setting, consider the problem of social welfare maximization with production costs from [**?**, **?**]. In this setting a seller can produce and sell items that are of m types. The seller has a *production cost function* $g : \mathbb{R}^m_+ \to \mathbb{R}_+$, and producing z_j units of every item $j \in [m]$ costs him $g(z)$. There are n buyers who arrive online. Each buyer $i \in [n]$ is interested in subsets of items (bundles) that belong to a set family $\mathscr{S}_i \subseteq 2^{[m]}$. The value of buyer i for subset $S \in \mathscr{S}_i$ is given by $v_i(S)$, where $v_i : \mathscr{S}_i \to \mathbb{R}_+$ is her *valuation function*. If buyer i is allocated a bundle $T_i \in \mathscr{S}_i$, she pays the seller her valuation $v_i(T_i)$. The goal of the seller is to produce items and allocate subsets to buyers so as to maximize the social welfare $\sum_{i=1}^n v_i(T_i) - g(z)$, where $T_i \in \mathscr{S}_i$ denotes the subset allocated to buyer i and $z \in \mathbb{R}^m$ is the total quantity of all items produced. Previous work on this problem [**?**, **?**] studied only the *separable* case where each item j had a separate production cost function $g_j : \mathbb{R}_+ \to \mathbb{R}_+$, and $g(z) := \sum_{j=1}^m g_j(z_j)$. However, since this problem fits perfectly in our framework, our techniques allow us to handle a much broader class of *non-separable* production cost functions—e.g., we can handle $g(z) = (\sum_{j=1}^m z_j)^2$—which were beyond the scope of previous techniques.

[1] A function f is monotone if for every x', x such that $x'_i \geq x_i$ for all $i \in [n]$, we have $f(x') \geq f(x)$.

[2] The Fenchel dual function formally defined in (2.1); see, e.g., [**?**] for background and properties.

1.1 Our Results

We first present our results for the fractional framework. Then, we discuss integral applications for the framework and extensions.

Algorithms for the general fractional framework. Let d denote the *row sparsity* of the matrix A, i.e., the maximum number of non-zeroes in any row. Let ρ be an the maximum-to-minimum ratio of positive entries in any column of A.

Theorem 1 *[Online Covering/Packing Theorem] Let $f : \mathbb{R}_+^n \mapsto \mathbb{R}_+$ be a function that is monotone, differentiable, convex, such that ∇f is monotone and for all $x \geq 0$, $\langle \nabla f(x), x \rangle \leq p \cdot f(x)$. Then,*

1. *There exists an $O(p \log d)^p$-competitive algorithm that produces a monotone primal solution.*
2. *There exists an $O(p \log(\rho d))^p$-competitive algorithm that produces a monotone dual solution.*

Both of these competitive ratios are known to be best possible [**?**, **?**].

In some applications such as the problem of social welfare maximization with production costs, it is more natural to design and analyze online algorithms based on conditions on the dual cost function $f^\star$.

(Zhiyi: Is it possible to replace the third condition with something more intuitive?)

Theorem 2 *[Informal theorem for online convex packing] Suppose the packing cost function $f^\star$ satisfies the following conditions:*

1. *$f^\star$ is convex, differentiable, and monotone, and $f^\star(0) = 0$; $\nabla f^\star$ is monotone.*
2. ***[$f^\star$ has bounded growth.]** There exists some p such that for all $z \in \mathbb{R}_+^n$, $\langle \nabla f^\star(z), z \rangle \leq p \cdot f^\star(z)$.*
3. ***[$f^\star$ is super linear.]** There exists some $\lambda > 1$ such that for all $\rho \geq 1$, for all $z \in \mathbb{R}_+^n$, $\nabla f^\star(\rho z) \geq \rho^{\lambda - 1} \cdot \nabla f^\star(z)$.*

Then, there exists an $O\left(\frac{p\lambda}{\lambda - 1}\right)$-competitive algorithm for the online convex packing problem.

≪Niv's Comment 1.1: Should we write corollaries for p-norm?≫ DP 1.1

≪Niv's Comment 1.2: Do we want techniques subsection?≫ DP 1.2

Applications. The general framework captures many existing and new applications. We outline three such applications below and present a more extensive list of applications in the appendix.

(1) Mixed Covering and Packing LPs. As a direct application of our general framework, we obtain an $O(p \log d)$-competitive algorithm for this problem (described above), where $d \leq n$ is the row-sparsity of the covering constraint matrix A. Prior to our work, [**?**] gave an $O(\log K \cdot \log(d\kappa\gamma))$-competitive algorithm for the special case of $p = \log K$ (corresponding to $\|\lambda\|_\infty$, the makespan of the loads), where γ and κ are the maximum-to-minimum ratio of the entries in the covering and packing constraints.

(2) Social Welfare Maximization with Production Costs. Our main result is for the fractional version of the problem where the allocation to each buyer i is allowed to be any point in the convex hull of the set family $\mathcal{S}_i$. We show that for arbitrary valuation functions (assuming the access to demand oracles) and a large family of production cost functions, our framework provides a polynomial time online algorithm. As a concrete example, suppose the production cost function is monotone and convex degree-p polynomial with non-negative coefficients. In this case, we get an $O(p)$-competitive algorithm.

(3) Unrelated Machine Scheduling with Startup Cost. ≪Anupam's Comment 1.3: shorten to one para?≫ AG 1.3
Let M be a set of m machines, where machine i has *startup cost* $c_i \geq 0$, and J be a set of n jobs that arrive online. The *processing time* of job j on machine i is denoted $p_{ij} \geq 0$. A *schedule* is an assignment of jobs to machines, and the *load* L_i of a machine i in a schedule is the sum of processing times of all jobs assigned to it. The *open machines* M_o are the machines to which at least one job has been assigned and the cost of the schedule is the sum of startup costs of open machines. The goal is to obtain a schedule that simultaneously minimizes cost and some function f of machine loads. The typical functions for f are: (1) the *makespan* or maximum load among all machines, i.e., $f = \max_{i \in M} L_i$, (2) the total load over all machines, i.e., $f = \sum_{i \in M} L_i$, and (3) the more general ℓ_p-norm of the load, i.e., $f = (\sum_{i \in M} (L_i)^p)^{1/p}$ for any fixed $p \in [1, \log m]$ (since $\ell_p \equiv \ell_\infty$ for $p \geq \log m$). We assume that the input includes a pair of values $(\mathbf{C}, \mathbf{L})$ with the guarantee that there exists a schedule of startup cost at most $\mathbf{C}$ and ℓ_p-norm of the load at most $\mathbf{L}$ (p is fixed). We will compare the startup cost and the ℓ_p-norm of the load of the algorithm with $\mathbf{C}$ and $\mathbf{L}$ respectively and show a bicriteria competitive ratio. [3]

Using the syntactic definition of the ℓ_p-norm as the objective function leads to fractional convex program that falls into our general framework. However, this convex program has a polynomial integrality gap. Consider the following simple example. To overcome this integrality gap, we refine our definition of the ℓ_p-norm of the load on a partially open machine (for a fully open machine, we continue to use the syntactic definition of $\sum_{i \in M} \left(\sum_{j \in J} p_{ij} y_{ij}\right)^p$) to $\sum_{i \in M} \left(\frac{\sum_{j \in J} p_{ij} y_{ij}}{x_i}\right)^p x_i$, where y_{ij} is the assignment of job j to machine i and x_i is the fraction to which machine i is open. As a result, the objective is no longer convex, and we need constraints $y_{ij} \leq x_i$ which deviates from positive LPs as stated above. Nevertheless, we show that suitable problem-specific modifications to the generic algorithm can be used to solve this relaxation, even though it does not fit the framework exactly, which can then be rounded online to obtain an integer solution. Concretely, we show that there is a randomized online algorithm for this problem for arbitrary fixed p with a competitive ratio of $(O(\log m \log(mn)), O(p^2 \log^{1/p}(mn)))$.

A list of other applications appear in Appendix **??**.

1.2 Related Work

(Zhiyi: Copied from BCGNN)

This paper significantly advances the body of work in online primal-dual algorithms; see [**?**] for a survey. This approach has been applied to many problems: e.g., set cover [**?**], graph connectivity and cuts [**?**], caching [**?**], auctions [**?**], and scheduling [**?**]. Below we discuss only the work directly relevant here.

Online packing and covering *linear programs* were first considered in [**?**], where an $O(\log n)$-competitive algorithm for covering and an $O(\log(n \frac{a_{max}}{a_{min}}))$-competitive algorithm for packing was given. The competitive ratio for covering linear programs was improved to $O(\log d)$ in [**?**], where $d \leq n$ is the maximum number of non-zero entries in any row. The first algorithm for online *mixed packing and covering* LPs appeared in [**?**]. The algorithm had a competitive ratio of $O(\log K \cdot \log(d\kappa\gamma))$, for K being the number of packing constraints and γ (resp. κ) being the maximum-to-minimum ratio of entries in the covering (resp. packing) constraints. Our framework improves this bound to $O(\log K \cdot \log d)$, which is asymptotically best possible [**?**].

[3] Using standard doubling guesses, our formulation can be shown to be asymptotically equivalent to one where one objective needs to be optimized subject to a given bound on the other. Moreover, our formulation subsumes single objective formulations where the two objectives are combined using a linear function.

Online maximization problems with production costs were introduced by Blum et al. [**?**] and extended by Huang and Kim [**?**]. The key differences from our setting are: (i) these papers deal with an *auction* setting where the seller is not aware of the valuations of the buyers, whereas our setting is not strategic, and (ii) these papers are restricted to separable production costs, whereas we can handle much more general (non-separable) cost functions.

2 Preliminaries

≪Hubert's Comment 2.1: Copied from our writeup. Gives the notation, and can be shortened later.≫ HC 2.1

≪Anupam's Comment 2.2: I shortened it a little. Still has some overlap with the intro and later sections; needs a little more work. Should be 1 page at the most.≫ AG 2.2

For a positive integer n, we denote $[n] := \{1, 2, \ldots, n\}$. We use x to denote a column vector. For two vectors a and b of the same dimension, we write $a \geq b$ if each coordinate of a is at least the corresponding coordinate of b. We use $\langle a, b \rangle = a^\intercal b$ to denote the dot product of a and b. Let $\mathbf{0}$ and $\mathbf{1}$ denote the all zero's and all one's vectors, respectively. A function $g : \mathbb{R}_+^n \to \mathbb{R}_+^m$ is *monotone* if $a \leq b$ implies that $g(a) \leq g(b)$. For $i \in [n]$, we use e_i to denote the unit vector whose ith coordinate is 1.

Fenchel Conjugate: Given a function $f : \mathbb{R}_+^n \to \mathbb{R}_+$, its Fenchel conjugate $f^* : \mathbb{R}_+^n \to \mathbb{R}_+$ is defined as

$$f^*(z) = \max_{x \in \mathbb{R}_+^n} \left\{ \langle x, z \rangle - f(x) \right\} \tag{2.1}$$

For the rest of this paper, we will focus on convex functions f that are non-negative, monotone and differentiable, and $f(\mathbf{0}) = 0$. In this case, the conjugate satisfies the following properties:

1. The conjugate f^* is non-negative, monotone, convex, and $f^*(\mathbf{0}) = 0$.
2. If f is lower semi-continuous, $f^{**} = f$.

Convex Covering: In the *offline* convex covering problem, there are n resources. Producing x_i units of resource i for each $i \in [n]$ incurs a production cost of $f(x)$, where $f : \mathbb{R}_+^n \to \mathbb{R}_+$ is convex, non-negative, monotone, and differentiable, and $f(\mathbf{0}) = 0$. Given m covering constraints: $\sum_{i \in [n]} a_{ji} x_i \geq 1$ for $j \in [m]$, where a_{ji}'s are non-negative, the objective is to minimize the total production cost $f(x)$ while satisfying all covering constaints. Formally, let a_j denote the non-negative vector $(a_{j1}, a_{j2}, \ldots, a_{jn})$ and $A = (a_{ji})$ denote the $m \times n$ matrix whose rows are a_j's. The offline convex covering problem can be formulated as a convex program

$$\min_{x \in \mathbb{R}_+^n} \; C(x) := f(x) \;\text{ subject to }\; Ax \geq \mathbf{1} \tag{2.2}$$

In the *online* convex covering problem, the covering requests arrive one-by-one. In round $k \in [m]$, covering request k arrives with the vector $a_k \in \mathbb{R}_+^n$ (we use j as a generic index of covering constraints and k as the index of the current request), and the *covering player* must decide immediately how to increase some of the x_i's to satisfy the request $\sum_{i \in [n]} a_{ki} x_i \geq 1$ without knowing future requests. The solution x must be monotone: the algorithm cannot decrease the values of x_i's at any time.

Convex Packing: Again, in the *offline* problem, there are n resources. Producing z_i units of resource i for each $i \in [n]$ incurs a production cost of $g(z)$, where $g : \mathbb{R}_+^n \to \mathbb{R}_+$ is convex, non-negative, monotone, and differentiable, and $g(\mathbf{0}) = 0$. Let there be m packing requests, where each request $j \in [m]$ is specified by a non-negative vector $a_j = (a_{j1}, a_{j2}, \ldots, a_{jn})$ such that serving one unit of packing request j consumes a_{ji} units

of resource i. The objective is to maximize the total number of units of packing requests served *minus* the total production cost. If $A = (a_{ji})$ is the $m \times n$ matrix whose rows are a_j's (as above), and y_j is the number of units of packing request j that the algorithm decides to serve, then the amount of resource i consumed is $z_i = \sum_{j\in[m]} a_{ji} y_j$. The offline convex packing problem can be formulated as follows:

$$\max_{y\in\mathbb{R}^m_+} \; P(y) := \sum_{j\in[m]} y_j - g\left(A^T y\right) \tag{2.3}$$

In the online problem, the packing requests arrive one by one. In round $k \in [m]$, packing request k arrives with vector a_k, and the packing player must irrevocably pick a value $y_k \geq 0$. I.e., the vector y is initially $\mathbf{0}$; in round k, the online algorithm can only increase the k-th coordinate of y.

Convex Covering/Packing Duality: If the cost function of the convex packing problem is the conjugate of the cost function of the convex covering problem, namely $g(z) = f^*(z)$, the convex packing program (2.3) is the Fenchel dual program of the convex covering program (2.2). See, e.g., [**?**] for details on Fenchel duality.

Lemma 3 *[Weak Duality] For any $x \in \mathbb{R}^n_+$ such that $Ax \geq 1$ and any $y \in \mathbb{R}^m_+$, we have $C(x) \geq P(y)$, which holds even if f is not convex.*

In the rest of this paper, we will use f^* to denote the cost function in the convex packing problem, and the corresponding convex program becomes: $\max_{y\in\mathbb{R}^m_+} \; P(y) := \sum_{j\in[m]} y_j - f^*\left(A^T y\right)$. When we have explicit assumptions on f, we emphasize the performance of the covering player, and let the packing player play an auxiliary role. On the other hand, when we have explicit assumptions on the conjugate f^*, the roles are reversed, and we emphasize the performance of the packing player.

Online Primal Dual Framework

≪Anupam's Comment 2.3: Drop this whole subsection≫ AG 2.3

Online Primal Dual Algorithm: Our online primal dual algorithms maintain both a feasible covering assignment x and a feasible packing assignment $y \in \mathbb{R}^m_+$ online. More precisely, we can view an online primal dual algorithm as running an online covering algorithm and an online packing algorithm simultaneously as follows:

- **Covering Algorithm.** A vector $x \in \mathbb{R}^n_+$ is initially set to $\mathbf{0}$. In round $k \in [m]$, the covering player can increase some coordinates of x such that the covering constraint $\sum_{i\in[n]} a_{ki} x_i = \langle a_k, x\rangle \geq 1$ is satisfied. The goal is to minimize $C(x) := f(x)$ at the end of the process while satisfying all covering constraints.
- **Packing Algorithm.** In round $k \in [m]$, the packing player irrevocably picks a value $y_k \geq 0$. We can view that there is some vector y that is initially set to $\mathbf{0}$, and in round k, the packing player can only increase the k-th coordinate of y. The goal is to maximize $P(y)$.

Online Primal Dual Analysis: As the covering and the packing algorithms maintain their corresponding feasible solutions x and y such that after every round, we seek to preserve an invariant $C(x) \leq \alpha \cdot P(y)$, where α is a parameter that can depend on the function f (and its conjugate f^*) and n, but is independent on the number of requests m.

Because of weak duality $C(x) \geq C^{\text{opt}} \geq P^{\text{opt}} \geq P(y)$, it follows that $C(x) \leq \alpha \cdot P(y) \leq \alpha \cdot C^{\text{opt}}$, and $P^{\text{opt}} \leq C(x) \leq \alpha \cdot P(y)$. Hence, the online primal dual algorithm described above achieve a competitive ratio at most α for both the online covering and packing problems.

3 Online Covering Framework

≪Vish's Comment 3.1: The full proof of the covering algorithm is here- we can move some proofs to the appendix.≫ ≪Anupam's Comment 3.2: I'd like to change $\nabla_i f(x)$ to $(\nabla f(x))_i$.≫ VN 3.1 AG 3.2

In this section we present an online algorithm for the convex covering problem. The algorithm maintains simultaneously a feasible primal x and dual solution y. Observe that it is much cleaner than even the algorithm for the linear case [**?**].

Fractional Algorithm: When the k^{th} request $\sum_{i=1}^{n} a_{ki}x_i \geq 1$ arrives:

1. Let τ be a continuous variable denoting the current time.
2. While the new constraint is unsatisfied, i.e., $\sum_{i=1}^{n} a_{ki}x_i < 1$, increase τ at rate 1 and:
3. **Change primal variables:**
   - For each i with $a_{ki} > 0$, increase each x_i at rate

$$\frac{\partial x_i}{\partial \tau} = \frac{a_{ki}x_i + \frac{1}{d}}{\nabla_i f(x)}. \tag{3.1}$$

     Here d is an upper bound on the row sparsity of the matrix. $\nabla_i f(x)$ is the i^{th}-coordinate of the gradient $\nabla f(x)$.
4. **Change dual variables:**
   - Increase y_k at rate $r = \frac{\delta}{\log(1+d\cdot\rho)}$. Here ρ is an upper bound on the maximum-to-minimum ratio of positive entries in any column of A.
5. **Change dual variables (for analysis of the primal only):**
   - Increase y_k at rate $r = \frac{\delta}{\log(1+2d^2)}$.
   - If the for variable x_i, $\sum_{j=1}^{k} a_{ji}y_j = \delta \cdot \nabla_i f(\bar{x})$, then,
     - Let $m_i^\star = \arg\max_{j=1}^{k}\{a_{ji} | y_j > 0\}$.
     - Increase $y_{m_i^\star}$ at rate $-\frac{a_{ki}}{a_{m_i^\star i}} \cdot r$. (Note: this change occurs only if a_{ki} is strictly positive.)

The value $0 < \delta \leq 1$ is determined later. We emphasize that the primal algorithm does not depend on the value δ. To analyze the primal competitive ratio we use update step 5 for the dual. This update step may decrease the value of the dual variables. When analyzing the dual we use update step 4 that never decreases dual variables after their arrival. For the analysis, we denote x^τ and y^τ as the value of x and y at time τ, respectively. We first analyze the algorithm with the more complex update step 5 and then state the small (and simplifying) changes required to analyze the dual.

Observation 4 *[Feasible Solutions] The algorithm maintains a feasible monotonically non-decreasing primal solution. Also, for any $\delta > 0$, it maintains a feasible dual solution and for each i, $\sum_{j=1}^{k} a_{ji}y_j^\tau \leq \delta \cdot \nabla f(\bar{x})$.*

Proof: The first property follows by construction, since we only increase x till reaching a feasible solution. We prove that y is feasible by induction over the execution of the algorithm. While processing constraint k, if $\sum_{j=1}^{k} a_{ji}y_j^\tau < \delta \cdot \nabla f(\bar{x})$ for column i we are trivially satisfied. Suppose that during the processing of constraint k, we have $\sum_{j=1}^{k} a_{ji}y_j^\tau = \delta \cdot \nabla f(\bar{x})$ for some dual constraint i and time τ. Now the dual decrease

part of the algorithm kicks in, and the rate of change in the left-hand side of the dual constraint is:

$$\frac{d}{d\tau}\left(\sum_{j=1}^{k} a_{ji} y_j^\tau\right) = a_{ki} \cdot r - a_{m_i^\star i} \cdot \frac{a_{ki}}{a_{m_i^\star i}} \cdot r = 0$$

■

Before analyzing the competitive factor, let us first prove the following claim.

Claim 5 *[] For a variable x_i, let $T_i = \{j | a_{ji} > 0\}$ and let S_i be any subset of T_i. Then,*

$$x_i^\tau \geq \frac{1}{\max_{j \in S_i}\{a_{ji}\} \cdot d}\left(\exp\left(\frac{\ln\left(1+2d^2\right)}{\delta \cdot \nabla_i f(\bar{x})} \sum_{j \in S_i} a_{ji} y_j^\tau\right) - 1\right) \tag{3.2}$$

Proof: Let $\tau(j)$ denote the value of τ at the arrival of the jth primal constraint. We first note that the increase in the primal variables at any time $\tau(j) \leq \sigma \leq \tau(j+1)$ can be alternatively formulated by the following differential equation.

$$\frac{\partial x_i}{\partial y_j} = \frac{\log\left(1+2d^2\right)}{\delta} \cdot \frac{a_{ji} x_i + \frac{1}{d}}{\nabla_i f(x)} \geq \log\left(1+2d^2\right) \cdot \frac{a_{ji} x_i + \frac{1}{d}}{\delta \cdot \nabla_i f(\bar{x})}. \tag{3.3}$$

The inequality above uses the monotonicity of ∇f and $x \leq \bar{x}$. By solving the latter equation we get for any $\tau(j) \leq \sigma \leq \tau(j+1)$,

$$\frac{x_i^\sigma + \frac{1}{a_{ji} d}}{x_i^{\tau(j)} + \frac{1}{a_{ji} d}} \quad \geq \quad \exp\left(\frac{\ln\left(1+2d^2\right)}{\delta \cdot \nabla_i f(\bar{x})} \cdot a_{ji} y_j^\sigma\right), \tag{3.4}$$

Note that Inequality (3.4) is satisfied even when no decrease is performed on the dual variables, and such a decrease only makes the inequality stronger. To reduce notation below, we denote the current time $\tau = \tau(k+1)$ where k is the current constraint (note that the actual value of $\tau(k+1)$ has not been revealed by the algorithm yet). Multiplying over all indices in S_i we get,

$$\exp\left(\frac{\ln\left(1+2d^2\right)}{\delta \cdot \nabla_i f(\bar{x})} \sum_{j \in S_i} a_{ji} y_j^\tau\right) \leq \exp\left(\sum_{j \in S_i} \frac{\ln\left(1+2d^2\right)}{\delta \cdot \nabla_i f(\bar{x})} \cdot a_{ji} y_j^{\tau(j+1)}\right) \tag{3.5}$$

$$\leq \quad \prod_{j \in S_i} \frac{x_i^{\tau(j+1)} + \frac{1}{a_{ji} d}}{x_i^{\tau(j)} + \frac{1}{a_{ji} d}} \leq \prod_{j \in S_i} \frac{x_i^{\tau(j+1)} + \frac{1}{\max_{j \in S_i}\{a_{ji}\} \cdot d}}{x_i^{\tau(j)} + \frac{1}{\max_{j \in S_i}\{a_{ji}\} \cdot d}} \tag{3.6}$$

$$\leq \quad \prod_{j \in T_i} \frac{x_i^{\tau(j+1)} + \frac{1}{\max_{j \in S_i}\{a_{ji}\} \cdot d}}{x_i^{\tau(j)} + \frac{1}{\max_{j \in S_i}\{a_{ji}\} \cdot d}} = \frac{x_i^\tau + \frac{1}{\max_{j \in S_i}\{a_{ji}\} \cdot d}}{\frac{1}{\max_{j \in S_i}\{a_{ji}\} \cdot d}}. \tag{3.7}$$

Inequality (3.5) follows as each dual variable y_j achieves its highest value at time $\tau(j+1)$. Inequality (3.6) follows by substituting (3.4) into (3.5) with $\sigma = \tau(j+1)$. Inequality (3.7) follows as the value of x_i monotonically non-decreases in time. Finally, the last equality is obtained using a telescopic product and the fact that x_i increases only in rounds with $a_{ki} > 0$. ■

Theorem 6 *[] Let f be a non-negative function that is monotone, differentiable, convex, such that ∇f is monotone and for all $x \geq 0$, $\langle \nabla f(x), x \rangle \leq p \cdot f(x)$. Then,*

- *There exists an $O(p \log d)^p$-competitive algorithm that produces a monotone primal solution.*
- *There exists an $O(p \log(\rho d))^p$-competitive algorithm that produces a monotone dual solution.*

Here d is the maximal row sparsity of the matrix and ρ is $\max_j \max_{i,i' | a_{i,j} \neq 0} \{\frac{a_{i'j}}{a_{ij}}\}$.

Proof: Consider the update when primal constraint k arrives and τ is the current time. Let $U(\tau)$ denote the set of indices for which we have $a_{ki} > 0$ and $\sum_{j=1}^{k} a_{ji} y_i^\tau = \delta \cdot \nabla_i f(\bar{x})$. So $|U(\tau)| \leq d$ the row-sparsity of A. Moreover, let us define for every $i \in U(\tau)$, $S_i = \{j | a_{ji} > 0, y_j^\tau > 0\}$. Clearly, $\sum_{j \in S_i} a_{ji} y_j^\tau = \sum_{j=1}^{k} a_{ji} y_j^\tau = \delta \cdot \nabla_i f(\bar{x})$, hence by Claim 5 and the fact that $\sum_i a_{ki} x_i^\tau < 1$, we get for every $i \in U(\tau)$,

$$\frac{1}{a_{ki}} > x_i^\tau \geq \frac{1}{\max_{j \in S_i}\{a_{ji}\} \cdot d} \left(\exp\left(\ln(1+2d^2)\right) - 1\right),$$

and after simplifying we get $\frac{a_{ki}}{a_{m_i^\star i}} = \frac{a_{ki}}{\max_{j \in S_i}\{a_{ji}\}} \leq \frac{1}{2d}$. As a result, we can bound the rate of change in the dual expression $\sum_{j=1}^{k} y_j$ at any time τ:

$$\frac{d\left(\sum_{j=1}^{k} y_j\right)}{d\tau} = r - \sum_{i \in U(\tau)} \frac{a_{ki}}{a_{m_i^\star i}} \cdot r \geq r\left(1 - \sum_{i \in U(\tau)} \frac{1}{2d}\right) \geq \frac{1}{2} r, \tag{3.8}$$

where the last inequality follows as $|U(\tau)| \leq d$. On the other hand, when processing constraint k during the execution of the algorithm, the rate of increase of the primal objective f is:

$$\frac{df(x^\tau)}{d\tau} = \sum_i \nabla_i f(x^\tau) \frac{\partial x_i^\tau}{\partial \tau} = \sum_{i | a_{ki} > 0} \nabla_i f(x^\tau) \left(\frac{a_{ki} x_i^\tau + \frac{1}{d}}{\nabla_i f(x^\tau)}\right) = \sum_{i | a_{ki} > 0} \left(a_{ki} x_i^\tau + \frac{1}{d}\right) \leq 2. \tag{3.9}$$

The final inequality uses the fact that the covering constraint is unsatisfied, and that d is at least the number of non-zeroes in the vector a_k. From (3.8) and (3.9) we can now bound the following primal-dual ratio:

$$\frac{d\left(\sum_{j=1}^{k} y_j^\tau\right)}{df(x^\tau)} \geq \frac{r}{4} = \frac{\delta}{4\ln(1+2d^2)}. \tag{3.10}$$

Thus, if $\bar{y}$ is the final dual solution we get,

$$\sum_{i=1}^{m} \bar{y}_i \geq \frac{\delta}{4\ln(1+2d^2)} \cdot f(\bar{x}). \tag{3.11}$$

To complete the proof of Theorem 6, we use Lemma 38 along with the assumption that $\langle \nabla f(x), x \rangle \leq p \cdot f(x)$ to get that

$$f^*(A^t \bar{y}) \leq f^*(\delta \cdot \nabla f(\bar{x})) \leq \delta^{\frac{p}{p-1}} \cdot (p-1) \cdot f(\bar{x})$$

Combining with (3.13),

$$D = \sum_{i=1}^{m} \bar{y}_i - f^*(A^t \bar{y}) \geq \left(\frac{\delta}{4\ln(1+2d^2)} - \delta^{\frac{p}{p-1}} \cdot (p-1)\right) \cdot f(\bar{x})$$

The optimal choice of δ is $\frac{1}{(4p\ln(1+2d^2))^{p-1}}$. Plugging this value we get:

$$f(\bar{x}) \leq \left(4p\ln\left(1+2d^2\right)\right)^p D \leq \left(4p\ln\left(1+2d^2\right)\right)^p f(OPT)$$

Hence the proof.

Finally, we would like to get the result on the dual competitive ratio when using the non-decreasing dual update in step 4. We note first that the primal is still feasible. Also, Claim 5 is independent of the dual decrease. Therefore, we may apply it at the end of the execution (with $\ln(1+d\rho)$ instead of $\ln\left(1+2d^2\right)$) and with $S_i = \{j | a_{ji} > 0, \bar{y}_j > 0\}$. We get that,

$$\bar{x}_i \geq \frac{1}{\max_{j\in S_i}\{a_{ji}\}\cdot d}\left(\exp\left(\frac{\ln(1+d\rho)}{\delta\cdot\nabla_i f(\bar{x})}\sum_{j\in S_i} a_{ji}\bar{y}_j\right)-1\right) \tag{3.12}$$

Since $\bar{x}_i \leq \frac{1}{\min_{j\in S_i}\{a_{ji}\}}$ (since otherwise the constraint that determined the min value is satisfied and $\bar{y}_j$ would be zero). Hence, by simplifying and using that $\rho \geq \frac{\max_{j\in S_i}\{a_{ji}\}}{\min_{j\in S_i}\{a_{ji}\}}$ we get, $\sum_{j\in S_i} a_{ji}\bar{y}_j \leq \delta\cdot\nabla_i f(\bar{x})$, or equivalently $A^t\bar{y} \leq \delta\cdot\nabla f(\bar{x})$. Using the same arguments as above (the dual variables do not decrease) we get,

$$\sum_{i=1}^{m}\bar{y}_i \geq \frac{\delta}{2\ln(1+d\rho)}\cdot f(\bar{x}). \tag{3.13}$$

Plugging this we get,

$$D = \sum_{i=1}^{m}\bar{y}_i - f^*(A^t\bar{y}) \geq \left(\frac{\delta}{2\ln(1+d\cdot\rho)} - \delta^{\frac{p}{p-1}}\cdot(p-1)\right)\cdot f(\bar{x})$$

Optimizing, we get $\delta = \frac{1}{(2p\ln(1+d\cdot\rho))^{p-1}}$ yielding a competitive ratio $O(p\log(d\rho))^p$ for the dual. ■

≪Vish's Comment 3.3: Add the lower bound from Azar et al. in appendix.≫ VN 3.3

An example application: combining the above fractional solver with online randomized rounding (and some additional ideas to ensure small integrality gaps), we can obtain a competitive algorithm for the *discrete* set cover problem with an objective of minimizing the ℓ_p-norm of multiple costs. Details in §A.3.

Theorem 7 *[ℓ_p-norm Set Cover] There is an $O\left(\frac{p^3}{\log p}\log d\log e\right)$-competitive randomized online algorithm for set cover minimizing the ℓ_p-norm of multiple linear cost-functions. Here d is the maximum number of sets containing any element, and e is the number of elements.*

As another example (in §A.2), we get an optimal $O(\log d\log K)$-competitive ratio for the online "mixed-packing-covering" LP problem of [**?**], with row-sparsity d and K packing constraints.

4 Online Packing Framework

≪Hubert's Comment 4.1: We will shorten this part, and move some proofs to appendix.≫ HC 4.1

For the convex online packing problem, we will make explicit assumptions on the packing cost function f^* and exploit them to design and analyze our online algorithms. Hence, even though we use the same notation,

it is more natural to treat y as the primal vector and x as the dual vector. Again, we start with some regularity assumptions that we have discussed in the preliminary.

Assumption 8 *[Monotone differentiable convex function] The function f^* is convex, differentiable, and monotone, and $f^*(0) = 0$.*

Assumption 9 *[Monotone gradient] The gradient ∇f^* is monotone.*

Assumption 10 *[Bounded optimal] The offline convex packing problem has bounded optimal objective.* [4]

To further prove Theorem **??** for polynomials with linear terms, we will explain in Section 4.2 how to handle the linear terms and reduce the problem to polynomials of degree at least 2.

Theorem 11 *[Online packing framework] Suppose the packing cost function f^* satisfies the following conditions:*

1. *There exists some p such that for all $z \in \mathbb{R}^n_+$, $\langle \nabla f^*(z), z\rangle \le p \cdot f^*(z)$.*
2. *There exists some $\lambda > 1$ such that for all $\rho \ge 1$, for all $z \in \mathbb{R}^n_+$, $\nabla f^*(\rho z) \ge \rho^{\lambda-1} \cdot \nabla f^*(z)$.*

Then, there is an $O\big(\frac{p\lambda}{\lambda-1}\big)$-competitive online algorithm for the online convex packing problem.

4.1 Online Convex Packing Algorithm for Theorem 11

The details are described in Algorithm 1.

Initialize $x := z := 0$
while constraint vector $a_k = (a_{k1}, \ldots, a_{kn})$ arrives in round k **do**
  Set $y_k := 0$
  while $\sum_{i=1}^n a_{ki} x_i < 1$ **do**
    Continuously increase y_k
    Simultaneously for each $i \in [n]$, increase z_i at rate $\frac{dz}{dy_k} = a_{ki}$, and
    increase x according to $x = \nabla f^*(\rho z)$

Algorithm 1: Convex online packing

Here, the vector x plays an auxiliary role and is initialized to 0. Throughout the algorithm, we maintain the invariant $z = A^T y$ and $x = \nabla f^*(\rho z)$ for some parameter $\rho > 1$ to be determined later. In round $k \in [m]$, the vector $a_k = (a_{k1}, a_{k2}, \ldots, a_{kn})$ is given. The variable y_k is initialized to 0, and is continuously increased while $\sum_{i\in[n]} a_{ki} x_i < 1$. 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 $x = \nabla f^*(\rho z)$. Since ∇f^* is monotone, as y_k increases, both z and z increase monotonically. We show in Lemma 12 that unless the offline packing problem is unbounded, eventually $\sum_{i\in[n]} a_{ki} x_i$ reaches 1, at which moment y_k stops increasing and round k finishes.

[4] Note that there are instances for which the offline convex packing problem whose objective can be arbitrarily large. Consider, for example, a convex packing problem with a linear cost function $f^*(z) = \frac{1}{2n}\sum_{i\in[n]} z_i$ and the first request is $a_1 = 1$. Then, by letting y_1 to be arbitrarily large, we can get a feasible packing assignment with arbitrarily large objective. Obviously, such instances are not interesting for practical purposes. Hence, we will assume that the offline optimal is bounded.

Observe that the coordinates of x are increased monotonically throughout the algorithm. Below we show that at the end of the process, the constraints $\sum_{i\in[n]} a_{ji}x_i \geq 1$ are satisfied for all $j \in [m]$. Hence, the vector x is feasible for the covering problem.

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 $z^{(0)} := 0$.

Lemma 12 *[Dual Feasibility] Recall our assumption 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: During round $k \in [m]$, the algorithm increases y_k only when $\sum_{i=1}^{n} a_{ki}x_i < 1$. Therefore, recalling $z = A^T y$, when the algorithm increases y_k, it also increases each z_i at rate $\frac{dz_i}{dy_k} = a_{ki}$.

Hence, we have

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

where the first inequality follows from the assumption $\nabla f^*(\rho z) \geq \rho^{\lambda-1} \cdot \nabla f^*(z)$, and the last inequality follows because $\langle a_k, x\rangle < 1$ when y_k is increased.

Therefore, 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 $\lambda \geq 2$) as y_k increases, which means the offline packing problem is unbounded, contradicting our assumption. ■

4.2 Proof of Theorem ??

Polynomial with Linear Terms. Observe that if a polynomial f^* has linear terms, then for any $\lambda > 1$, it does not satisfy the condition that $\nabla f^*(\rho z) \geq \rho^{\lambda-1} \cdot \nabla f^*(z)$ for any $z \in \mathbb{R}^n_+$ and any $\rho > 1$. We write $f^*(z) = \langle c, z\rangle + \widehat{f}^*(z)$, where $c = \nabla f^*(0)$ and $\widehat{f}^*(z)$ contains terms with degree at least 2. Therefore, for all $\rho > 1$, $\nabla \widehat{f}^*(\rho z) \geq \rho \cdot \nabla \widehat{f}^*(z)$.

Then, the objective function becomes

$$P(y) = \langle 1, y\rangle - f^*(A^T y) = \langle 1 - Ac, y\rangle - \widehat{f}^*(A^T y) \ .$$

Moreover, the corresponding covering problem becomes $\min_{x\geq 0} \widehat{f}(x)$ subject to $Ax \geq 1 - Ac$. In other words, in round $k \in [m]$, as the vector a_k arrives, the covering constraint becomes $\langle a_k, x\rangle \geq b_k$, where $b_k := 1 - \langle a_k, c\rangle$. If $b_k \leq 0$, then the constraint is automatically satisfied, and we set $y_k = 0$ such that round k finishes immediately. Otherwise, we can run Algorithm 1 using the function $\widehat{f}^*$ and the constraint vector $\frac{a_k}{b_k}$ in round k.

Next, we present a proof of Theorem **??** based on the above discussion.

Proof: [Theorem **??**] As discussed above, we write $c := \nabla f^*(0)$ and $\widehat{f}^*(x) := f^*(x) - \langle c, x\rangle$ as the convex polynomial containing the terms of f^* with degree at least $\lambda = 2$. (Observe that if f^* contains only linear terms, then the problem is trivial because the objective is unbounded when there is some round $k \in [m]$ such that $b_k > 0$.)

For ease of exposition, we can assume that for each $k \in [m]$, $b_k := 1 - \langle a_k, c\rangle > 0$. Otherwise, we can

essentially ignore the variable y_k by setting it to 0. We denote B as the diagonal matrix whose (k,k)-th entry is b_k. By writing $w := By$, the objective function can be expressed in terms of w as $\widehat{P}(w) := \langle 1, w\rangle - \widehat{f^*}((B^{-1}A)^T w)$.

Hence, we can run Algorithm 1 using function $\widehat{f^*}$ such that in round $k \in [m]$, when the vector a_k arrives, we can transform it by dividing each coordinate by b_k before passing it to the algorithm.

By Theorem 11, using $\lambda = 2$, it follows that the algorithm has competitive ratio $O(p)$. ■

5 Applications of the Online Covering Framework

≪Debmalya's Comment 5.1: Copied from Azar *et al.* To be edited to make it consistent with previous sections and details moved to the appendix.≫ DP 5.1

In this section, we give an algorithm for the UMSC problem and prove Theorem **??**.

5.1 Fractional Algorithm for UMSC

Recall that the input contains the pair of values $(\mathbf{C}, \mathbf{L})$ with the guarantee that there exists a feasible assignment of cost at most $\mathbf{C}$ and ℓ_p-norm at most $\mathbf{L}$. We will fix such an assignment and call it the *optimal* solution (denoted OPT). We will also assume that the algorithm knows the number of jobs n, which is without loss of generality up to constant factors in the competitive ratio.

The algorithm has two phases — an offline pre-processing phase, and an online phase that (fractionally) schedules the arriving jobs.

Offline Pre-processing. First, we note that all machines whose startup cost exceeds $\mathbf{C}$ are unused in OPT; hence, the algorithm discards these machines at the outset. Let m be redefined to the number of machines with startup cost at most $\mathbf{C}$. Next, we multiply the costs of all machines by $\frac{m}{\mathbf{C}}$ so that the cost of OPT is m. For any machine i with $c_i \leq 1$, we set $c_i = 1$; this increases the optimal cost to at most $2m$. We initialize x_i as follows: if $c_i = 1$, we set $x_i = 1$; else ($1 < c_i \leq m$), we set $x_i = 1/m$. Finally, we multiply all processing times by $\frac{\beta^{1/p}}{\mathbf{L}}$, where $\beta = \frac{m \ln m}{(40p)^p}$; then an ℓ_p^p-norm of β with the scaled processing times implies an ℓ_p-norm of $\mathbf{L}$ with the original processing times.

Before describing the online phase, we need to introduce some notation. Let machine i be said to be *closed*, *partially open*, or *fully open* depending on whether $x_i = 0$, $0 < x_i < 1$ or $x_i = 1$ respectively. We distinguish between (fractions of) jobs that are assigned when a machine is partially open and those that are assigned when the machine is fully open; let us denote the respective sets of jobs $J_0^{(i)}$ and $J_1^{(i)}$. (There can be at most one job that is in both sets since machine i became fully open while the job was being assigned. For this job, we will consider the fraction of the job assigned while machine i was partially open as being in set $J_0^{(i)}$ and the remainder in set $J_1^{(i)}$). Recall that the load on machine i is $L_i = \sum_{j \in J} y_{ij} p_{ij}$. However, for partially open machines, calculating this load exactly turns out to be difficult. Instead, we maintain an upper bound of $c_i^{1/p} x_i$ on the load, which then allows us to define a proxy load $\tilde{L}_i = c_i^{1/p} x_i + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij}$.

Suppose the algorithm wants to assign an infinitesimal fraction of a job to the machines. Intuitively, it should prefer machines whose cost and fractional ℓ_p-norm increases the least on assigning the fractional job. To formalize this notion, we define a function ψ_{ij} that the algorithm uses to sort machines in increasing order of

preference when assigning a fraction of job j:

$$\psi_{ij} = \begin{cases} \max\{c_i^{(p-1)/p} p_{ij}, p_{ij}^p\} & \text{if } x_i < 1. \\ (\tilde{L}_i + p_{ij})^p - \tilde{L}_i^p & \text{if } x_i \geq 1. \end{cases}$$

Online Assignment. When a new job j arrives, we use Algorithm 2 to update x_i, y_{ij} in multiple steps until $\sum_{i \in M} y_{ij} = 1$. This is a polynomial-time implementation of a continuous multiplicative weight augmentation algorithm, N being the discretization parameter that we set to $nm \ln m$ to ensure that each discrete step is small enough. (For technical reasons, we maintain $y_{ij} \leq 2x_i$ instead of $y_{ij} \leq x_i$.)

while $\sum_{i \in M} y_{ij} < 1$, do the following:

- Sort the machines in non-increasing order by ψ_{ij} and let $P(j)$ be the minimal prefix[1] of this sorted order such that $\sum_{i \in P(j)} x_i \geq 1$.
- For each partially[2] open machine $i \in P(j)$, set $\Delta x_i = \frac{x_i}{c_i N}$.
- For each machine $i \in P(j)$, set $\Delta y_{ij} = \min\left(\frac{x_i}{\psi_{ij} N}, 2x_i - y_{ij}\right)$.
- Update $x_i \leftarrow x_i + \Delta x_i$, $y_{ij} \leftarrow y_{ij} + \Delta y_{ij}$, unless x_i or y_{ij} exceeds 1. In this case, we do a *small step*, i.e., we redefine Δx_i and Δy_{ij} with a value of $N' > N$ instead of N so that $\max_{i,j}\{x_i, y_{ij}\} = 1$.

Algorithm 2: Fractional assignment for a single job

5.2 Analysis of the fractional algorithm

We bound the cost and ℓ_p-norm of the fractional algorithm using a potential function defined as

$$\Phi_i = \begin{cases} c_i x_i & \text{if } x_i < 1. \\ \tilde{L}_i^p + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij}^p & \text{if } x_i = 1. \end{cases}$$

The overall potential function $\Phi = \sum_{i \in M} \Phi_i$. Note that the potential function is continuous and monotonically non-decreasing. First, observe that the potential of a partially open machine is exactly its fractional startup cost and becomes c_i when the machine is fully opened (i.e., when x_i becomes 1). Therefore, by monotonicity, $\Phi_i \geq c_i x_i$ during the entire run of the algorithm. Additionally, the algorithm ensures for each partially open machine, the following conditions are satisfied:

$$\sum_{j \in J_0^{(i)}} y_{ij} p_{ij} \leq c_i^{1/p} x_i \quad \text{and} \quad \sum_{j \in J_0^{(i)}} y_{ij} p_{ij}^p \leq c_i x_i. \tag{5.1}$$

Therefore, the potential also bounds the fractional objective function, i.e. the fractional ℓ_p^p-norm of the load.

[1] $P(j)$ is always defined since $\sum_{i \in M} x_i \geq 1$.

[2] Only the last machine in $P(j)$ may be fully open; all other machines are partially open.

Note that ψ_{ij} is a bound on the discrete differential $\frac{\Delta\Phi_i}{\Delta y_{ij}}$. For partially open machines, $\frac{\Delta\Phi_i}{\Delta y_{ij}} = \frac{c_i \Delta x_i}{\Delta y_{ij}}$, and from the two conditions in Eqn. 5.1 we get $\Delta y_{ij} p_{ij} \leq c_i^{1/p} \Delta x_i$, and $\Delta y_{ij} p_{ij}^p \leq c_i \Delta x_i$, which defines ψ_{ij}. For fully open machines, the discrete differential is immediate.

First, we bound the increase in potential in the pre-processing phase (Lemma 13), in each single step step (Lemma 14), and in all the *small steps* (Lemma 15).

Lemma 13 *[] At the end of the pre-processing phase, $\Phi \leq m$.*

Proof: After pre-processing, the potential $\Phi = \sum_{i \in M} c_i x_i$, where each $c_i x_i \leq 1$. ■

Lemma 14 *[] The increase in the potential in a single algorithmic step is at most $5/N$.*

Proof: The total increase in Φ for partially open machines in each step is

$$\sum_{i \in PA} c_i \Delta x_i \leq \sum_{i \in P(j)} c_i \frac{x_i}{c_i N} \leq \frac{2}{N}.$$

For a fully open machine i, $\Delta y_{ij} = \frac{1}{N((\tilde{L}_i + p_{ij})^p - \tilde{L}_i^p)} \leq \frac{1}{pNp_{ij}\tilde{L}_i^{p-1}}$, which increases the first term in Φ by

$$\begin{aligned}
(\tilde{L}_i + \Delta y_{ij} p_{ij})^p - \tilde{L}_i^p &\leq \left(\tilde{L}_i + \frac{1}{Np\tilde{L}_i^{p-1}}\right)^p - \tilde{L}_i^p \\
= \tilde{L}_i^p \left(\left(1 + \frac{1}{Np\tilde{L}_i^p}\right)^p - 1\right) &\leq \tilde{L}_i^p \left(\left(1 + \frac{2}{N\tilde{L}_i^p}\right) - 1\right) \leq \frac{2}{N}.
\end{aligned}$$

The penultimate inequality follows from $(1+\alpha)^p \leq 1 + 2\alpha p$ for $\alpha \leq 1/(2p)$, which in turn holds since $\tilde{L}_i^p \geq c_i \geq 1$ and $N \geq 2$. Additionally, note that

$$\Delta y_{ij} = \frac{1}{N((\tilde{L}_i + p_{ij})^p - \tilde{L}_i^p)} \leq \frac{1}{Np_{ij}^p}.$$

So the increase in the second term of Φ_i is at most $1/N$. Further, in each step, the load on at most one fully open machine increases. Hence, the total increase in potential is at most $5/N$. ■

Lemma 15 *[] The total increase in potential in all the small steps is at most 2.*

Proof: In each small step, either machine becomes fully open or a job is completely assigned. So, the total number of small steps is at most $n+m$. Therefore, by Lemma 14, the total increase in potential in the small steps is at most $\frac{m+n}{N} < \frac{n+m}{nm} \leq 2$. ■

This leaves us with the task of bounding the total number of regular (i.e., not small) steps. We classify these steps according to an optimal solution (denoted OPT). Let M_{OPT} denote the set of open machines in OPT and $\text{OPT}(j) \in M_{\text{OPT}}$ be the machine where job j is assigned to by OPT. The three categories are:

1. $\text{OPT}(j) \in P(j)$ and $\text{OPT}(j)$ is partially open
2. $\text{OPT}(j) \notin P(j)$ and $\text{OPT}(j)$ is partially open

3. $\text{OPT}(j)$ is fully open

We bound the total increase in potential in each of the three categories separately.

Lemma 16 *[] The total increase in potential in the first category steps is $O(m\log m)$.*

Proof: In any step of the first category, the value of $x_{\text{OPT}(j)}$ increases to $x_{\text{OPT}(j)}\left(1+\frac{1}{c_{\text{OPT}(j)}N}\right)$. Since x_i is initialized to at least $1/m$ for every machine i in the pre-processing phase and x_i cannot exceed 1, it follows that the total number of steps in the first category is at most

$$\sum_{i\in M_{\text{OPT}}} c_i N\log m = O(Nm\log m).$$

Using Lemma 14, we conclude that the increase in potential in these steps is $O(m\log m)$. ■

Lemma 17 *[] The total increase in potential in the second category steps is $O(m\log m)$.*

Proof: For any step, let $Q(j)$ denote the set of machines in $P(j)$ for which $\Delta y_{ij} = 2x_i - y_{ij}$, and let $R(j) = P(j)\setminus Q(j)$. Note that for any job j, an algorithmic step for which $\sum_{i\in Q(j)} x_i \geq 1/2$ must be its last step. This follows from the observation that in this step, $\sum_{i\in M}(y_{ij}+\Delta y_{ij}) \geq \sum_{i\in Q(j)}(y_{ij}+\Delta y_{ij}) = \sum_{i\in Q(j)} 2x_i \geq 1$. So, there are at most n steps of this kind.

Now, we bound the number of algorithmic steps where $\sum_{i\in R(j)} x_i \geq 1/2$. In any such step, using the fact that $\psi_{ij} \leq \psi_{\text{OPT}(j)j}$ (otherwise $\text{OPT}(j)\in P(j)$), we have

$$\sum_{i\in M}\Delta y_{ij} \geq \sum_{i\in R(j)}\Delta y_{ij} = \sum_{i\in R(j)}\frac{x_i}{N\psi_{ij}} \geq \frac{1}{2N\psi_{\text{OPT}(j)j}}.$$

Since $\text{OPT}(j)$ is partially open, $\psi_{\text{OPT}(j)j} = \max\{c_{\text{OPT}(j)}^{(p-1)/p} p_{\text{OPT}(j)j}, p_{\text{OPT}(j)j}^p\}$. Let L_i^{OPT} be the load on machine i in OPT. Summing over all jobs, we have

$$\begin{aligned}
&\sum_{j\in J}\psi_{\text{OPT}(j)j} \leq \sum_{j\in J}\left(c_{\text{OPT}(j)}^{(p-1)/p} p_{\text{OPT}(j)j} + p_{\text{OPT}(j)j}^p\right) \leq \sum_{j\in J} c_{\text{OPT}(j)}^{(p-1)/p} p_{\text{OPT}(j)j} + \beta \\
&= \sum_{i\in M}\sum_{j:\text{OPT}(j)=i} c_i^{(p-1)/p} p_{ij} + \beta = \sum_{i\in M} c_i^{(p-1)/p} L_i^{\text{OPT}} + \beta \leq m^{(p-1)/p}\beta^{\frac{1}{p}} + \beta \\
&\leq m^{(p-1)/p}\left(\frac{m\log m}{(40p)^p}\right)^{1/p} + \beta \leq \frac{m\log^{1/p} m}{40p} + \beta \leq 2m\log m,
\end{aligned}$$

where we use Hölder's inequality (se e.g., [**?**]) in the first inequality on the second line. Therefore, the total number of steps in this category is bounded by $O(Nm\log m)$. By Lemma 14, the total increase in potential in these steps is $O(m\log m)$. ■

Lemma 18 *[] The total increase in potential in the third category steps is $O(m\log m)$.*

Proof: Define L_i^* as $\tilde{L}_i$ at the end of the run of the algorithm. For each fully open machine i define $\psi_{ij}^* = (L_i^* + p_{ij})^p - L_i^{*p}$. By convexity of x^p, we have $\psi_{ij} \leq \psi_{ij}^*$. Recall the proof of Lemma 17 and the

definition $R(j)$. An identical argument shows that for each step in the third category we have,

$$\sum_{i\in M}\Delta y_{ij}\geq\sum_{i\in R(j)}\Delta y_{ij}=\sum_{i\in R(j)}\frac{x_i}{N\psi_{ij}}\geq\frac{1}{2N\psi_{\mathrm{OPT}(j)j}}\geq\frac{1}{2N\psi^*_{\mathrm{OPT}(j)j}}.$$

Therefore, by summing over all jobs, the total number of the third category steps is $\sum_{j\in J}2N\psi^*_{\mathrm{OPT}(j)j}$. By Lemma 14, the total increase in potential in third category steps is $10\left(\sum_{j\in J}\psi^*_{\mathrm{OPT}(j)j}\right)$. Define $\Delta_o\Phi$ as the increase in potential first and second category steps along with the small steps, and Φ_0 to be the potential after pre-processing. Also, let $\Delta_3\Phi$ be the increase in potential in third category steps and M_o be the set of machines that are fully opened by the algorithm. Then,

$$\begin{aligned}\Delta_3\Phi&\leq 10\left(\sum_{j\in J}\psi^*_{\mathrm{OPT}(j)j}\right)\leq\left(10\sum_{j\in J}\left((L^*_{\mathrm{OPT}(j)}+p_{\mathrm{OPT}(j)j})^p-{L^*_{\mathrm{OPT}(j)}}^p\right)\right)\\&\leq 10\left(\sum_{i\in M_{\mathrm{OPT}}\cap M_o}\sum_{j:\mathrm{OPT}(j)=i}((L^*_i+p_{ij})^p-L^{*p}_i)\right)\leq 10\left(\sum_{i\in M_{\mathrm{OPT}}\cap M_o}((L^*_i+L^{\mathrm{OPT}}_i)^p-L^{*p}_i)\right).\end{aligned}$$

Rearranging the terms,

$$\begin{aligned}&\left(\frac{\Delta_3\Phi}{10}+\sum_{i\in M_{\mathrm{OPT}}\cap M_o}(L^*_i)^p\right)^{1/p}\leq\left(\sum_{i\in M_{\mathrm{OPT}}\cap M_o}(L^*_i+L^{\mathrm{OPT}}_i)^p\right)^{1/p}\leq\left(\sum_{i\in M_{\mathrm{OPT}}\cap M_o}(L^*_i+L^{\mathrm{OPT}}_i)^p\right)^{1/p}\\&\leq\left(\sum_{i\in M_{\mathrm{OPT}}\cap M_o}L^{*p}_i\right)^{1/p}+\left(\sum_{i\in M_{\mathrm{OPT}}\cap M_o}(L^{\mathrm{OPT}}_i)^p\right)^{1/p}\leq\left(\sum_{i\in M_{\mathrm{OPT}}\cap M_o}L^{*p}_i\right)^{1/p}+\beta^{1/p}.\end{aligned}$$

Now, we have two cases. First, suppose $2(\Delta_3\Phi)>\sum_{i\in M_{\mathrm{OPT}}\cap M_o}L^{*p}_i$. Then, we have

$$\begin{aligned}&\left(\frac{\Delta_3\Phi}{10}+\sum_{i\in M_{\mathrm{OPT}}\cap M_o}(L^*_i)^p\right)^{1/p}-\left(\sum_{i\in M_{\mathrm{OPT}}\cap M_o}(L^*_i)^p\right)^{1/p}\\&\geq\left(\frac{\Delta_3\Phi}{10}+2(\Delta_3\Phi)\right)^{1/p}-(2(\Delta_3\Phi))^{1/p}\geq\frac{(2(\Delta_3\Phi))^{1/p}}{40p}.\end{aligned}$$

The two last equations imply $\frac{(2(\Delta_3\Phi))^{1/p}}{40p}\leq\beta^{1/p}$, which implies $2(\Delta_3\Phi)=O(m\log m)$. Next, consider $2(\Delta_3\Phi)\leq\sum_{i\in M_{\mathrm{OPT}}\cap M_o}L^{*p}_i$. Then, we have

$$2(\Delta_3\Phi)\leq\sum_{i\in M_{\mathrm{OPT}}\cap M_o}L^{*p}_i\leq\Phi_0+\Delta_o\Phi+\Delta_3\Phi,$$

which implies that $\Delta_3\Phi\leq\Phi_0+\Delta_o\Phi=O(m\log m)$ by Lemmas 13, 15, 16, and 17. ■

The overall bound on the potential now follows from Lemmas 13, 15, 16, 17, and 18.

Theorem 19 *[] At the end of the algorithm, the potential is $O(m\log m)=O((40p)^p\beta)$.*

Bounding the cost and objective function. Having provided a bound on the potential function, we now relate it to the fractional cost and ℓ_p-norm of machine loads using Lemma 20 and Lemma 21 respectively.

Lemma 20 *[] For each partially open machine i, $\Delta y_{ij} p_{ij} \leq \Delta x_i {c_i}^{1/p}$.*

Proof: In each update step of a partially open machine i,

$$\Delta y_{ij} p_{ij} = \frac{p_{ij} x_i}{\psi_{ij} N} = \frac{p_{ij} \Delta x_i c_i}{\psi_{ij}} \leq \frac{p_{ij} \Delta x_i c_i}{c_i^{(p-1)/p} p_{ij}} = \Delta x_i {c_i}^{1/p}. \quad \square$$

■

Lemma 21 *[] For each partially open machine i, $\Delta y_{ij} p_{ij}^p \leq \Delta x_i c_i$.*

Proof: In each update step of a partially open machine i,

$$\Delta y_{ij} p_{ij}^p = \frac{p_{ij}^p x_i}{\psi_{ij} N} = \frac{p_{ij}^p \Delta x_i c_i}{\psi_{ij}} \leq \frac{p_{ij}^p \Delta x_i c_i}{p_{ij}^p} = \Delta x_i c_i.$$

■

Finally, we give the overall bound for the fractional solution.

Theorem 22 *[] For the fractional solution, the objective (fractional ℓ_p^p-norm of loads) is bounded by $O((40p)^p \beta)$ and the total cost is bounded by $O(m \log m)$.*

Proof: The first term in the fractional objective is bounded using Lemma 20. If $x_i < 1$, then

$$\sum_j \left(\frac{y_{ij} p_{ij}}{x_i} \right)^p x_i \leq \left(\frac{c_i^{1/p} x_i}{x_i} \right)^p x_i = c_i x_i = \Phi_i.$$

On the other hand, if $x_i = 1$, then

$$\sum_j \left(\frac{y_{ij} p_{ij}}{x_i} \right)^p x_i \leq (c_i^{1/p} + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij})^p = \tilde{L}_i^p \leq \Phi_i.$$

The second term in the fractional objective is bounded using Lemma 21:

$$\sum_j y_{ij} p_{ij}^p = \sum_{j \in J_0^{(i)}} y_{ij} p_{ij}^p + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij}^p \leq c_i x_i + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij}^p \leq \Phi_i.$$

Summing over all machines, the fractional objective is at most $2\Phi = O((40p)^p \beta)$. Since for all machines, $c_i x_i \leq \Phi_i$, the total cost is also bounded by the potential function, which is $O(m \log m)$ by Theorem 19. ■

5.3 Online Rounding for UMSC with ℓ_p norm

There are two decisions that an integer algorithm must make on receiving a new job j. First, it needs to decide the set of machines that it needs to open. Note that since decisions are irrevocable in the online model, the open machines form a monotonically growing set over time. Next, the algorithm must decide which among the open machines should it assign job j to. As we describe below, both these decisions are made by the integer algorithm based on the fractional solution that it maintains using the algorithm given in the

Opening Machines: For every machine i whose blue copy is closed, open it with probability (w/p) $\min\left(\frac{\alpha(x_i(j)-x_i(j-1))}{1-\alpha\cdot x_i(j-1)}, 1\right)$. (Eqn. 5.2 is satisfied by this rule using conditional probabilities.)
Assigning Job j:
- if $\sum_{i\in M^{(1)}(j)} y_{ij} \geq \frac{1}{2}$, then assign to blue copy of $i \in M^{(1)}(j)$ w/p $\frac{y_{ij}}{\sum_{i\in M^{(1)}(j)} y_{ij}}$,
- else if $\sum_{i\in M_o^{(0)}(j)} z_{ij} \geq 1$, then assign to blue copy of $i \in M_o^{(0)}(j)$ w/p $\frac{z_{ij}}{\sum_{i\in M_o^{(0)}(j)} z_{ij}}$,
- else assign to red copy of $i^* = \arg\min_{i\in M}\left(\left(\hat{L}_i + p_{ij}\right)^p - \hat{L}_i^p\right)$ after opening it.

Algorithm 3: Assignment of a Single Job by the Integer Algorithm

previous section. Following nomenclature established by Alon *et al* [**?**], we call this process of producing an integer solution online based on a monotonically evolving fractional solution an *online randomized rounding* procedure.

To simplify the analysis later, we will consider two copies of each machine: a *blue* copy and a *red* copy. Note that this is without loss of generality, up to a constant factor loss in the competitive ratio for both the cost and ℓ_p-norm objectives. First, we define a randomized process that controls the opening of blue copies of machines in the integer algorithm. Let $M_o(j)$ denote the set of machines whose blue copies are open after job j has been assigned, and $X_i(j)$ be an indicator random variable whose value is 1 if machine $i \in M_o(j)$ and 0 otherwise. Let $x_i(j)$ be the value of variable x_i in the fractional solution after job j has been completely assigned (fractionally). The integer algorithm maintains the invariant

$$\mathbb{P}[X_i(j) = 1] = \min(\alpha \cdot x_i(j), 1) \text{ for some parameter } \alpha \text{ that we will set later.} \tag{5.2}$$

using the rule given in Algorithm 3. Next, we need to assign job j to one of the open machines. We partition the set of machines M into two sets based on the fractional solution: $M^{(0)}(j)$ represents machines i such that $x_i(j) < \frac{1}{\alpha}$ and $M^{(1)}(j)$ represents machines i such that $x_i(j) \geq \frac{1}{\alpha}$. Note that after job j, the blue copies of all machines in $M^{(1)}(j)$ are open (by Eqn. 5.2). On the other hand, the blue copies of some subset of machines in $M^{(0)}(j)$ are open; call this subset $M_o^{(0)}(j)$, i.e. $M^{(0)}(j) = M^{(0)}(j) \cap M_o(j)$. In addition let, $z_{ij} = \frac{4y_{ij}}{\alpha\cdot x_i}$ and $\hat{L}_i$ be the current sum of processing times of all jobs assigned to the red copy of machine i. The assignment rule for job j is given in Algorithm 3.

5.4 Analysis

First, we argue about the expected cost of the solution. To bound the cost of red copies, we show that Case 3 has low probability.

Lemma 23 *[] For any job j, the probability of case 3 is at most* $\exp(-\alpha/48)$.

Proof: Consider a machine $i \in M^{(0)}(j)$, i.e. $x_i(j) < \frac{1}{\alpha}$. Such a machine is open after job j with probability $\alpha x_i(j)$. Let us define a corresponding random variable

$$Z_{ij} = \begin{cases} z_{ij} & \text{if } i \in M_o^{(0)}(j) \\ 0 & \text{otherwise.} \end{cases}$$

We need to bound the probability that $\sum_{i\in M^{(0)}(j)} Z_{ij} < 1$.

First, we observe that $Z_{ij} \leq \frac{4y_{ij}}{\alpha \cdot x_i(j)} \leq \frac{8}{\alpha}$ since $y_{ij} \leq x_i(j)$. Now, consider random variable

$$\tilde{Z}_{ij} = \begin{cases} \frac{8}{\alpha} & \text{with probability } \frac{\alpha y_{ij}}{2} \\ 0 & \text{otherwise.} \end{cases}$$

Note that the expectations of Z_{ij} and $\tilde{Z}_{ij}$ are identical and both have only one non-zero value in their support, but Z_{ij} has a strictly smaller range. Therefore, any tail bounds that apply to $\tilde{Z}_{ij}$ also apply to Z_{ij}. Further, note that

$$\mathbb{E}\left[\sum_{i \in M^{(0)}(j)} Z_{ij}\right] = \mathbb{E}\left[\sum_{i \in M^{(0)}(j)} \tilde{Z}_{ij}\right] = 4 \sum_{i \in M^{(0)}(j)} y_{ij} \geq 2.$$

Therefore, by Chernoff-Hoeffding bounds (e.g., [**?**]),

$$\begin{aligned} &\mathbb{P}\left[\sum_{i \in M^{(0)}(j)} Z_{ij} < 1\right] \leq \mathbb{P}\left[\sum_{i \in M^{(0)}(j)} \tilde{Z}_{ij} < 1\right] = \mathbb{P}\left[\sum_{i \in M^{(0)}(j)} \frac{\alpha}{8} \tilde{Z}_{ij} < \frac{\alpha}{8}\right] \\ &\leq \exp\left(-\frac{\frac{1}{2^2} \cdot \frac{2 \cdot \alpha}{8}}{3}\right) = \exp\left(-\alpha/48\right). \end{aligned}$$

■ We choose $\alpha = 48 \ln(mn)$ to obtain the following corollary. (For now, $\alpha = 48 \ln n$ would have sufficed but we will need $\alpha \geq \ln m$ in a later step.)

Corollary 24 *[] For any job j, the probability of case 3 is at most $\frac{1}{mn}$.*

Recall that the cost of each individual machine is at most m. Using linearity of expectation and the above corollary, we can now claim that the expected cost of red copies of machines is at most m. Similarly, using linearity of expectation and Eqn. 5.2, we can claim that the expected cost of blue copies of machines is $\sum_{i \in M} c_i \alpha x_i \leq \alpha \Phi$. Overall, we get the following bound for the cost of machines opened by the integer algorithm.

Lemma 25 *[] The total expected cost of machines in the integer algorithm is at most $(\alpha+1)\Phi = O(\ln(mn))\Phi$.*

We are now left to bound the expected ℓ_p-norm of machine loads. First, we obtain a bound for red copies of machines. Note that the assignment of jobs in Case 3 follows a greedy algorithm assuming all machines are open. Therefore, the analysis of the ℓ_p-norm of the red machines follows directly from the corresponding analysis without startup costs [**?**].

Lemma 26 *[[**?**]] The competitive ratio of the ℓ_p-norm of the red copies of machines is $O(p)$.*

Finally, we will bound the ℓ_p-norm of blue copies of machines. Let us define an indicator random variable

$$Y_{ij} = \begin{cases} 1 & \text{if job } j \text{ is assigned to the blue copy of machine } i \text{ in the integer solution} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the ℓ_p^p-norm of the integer solution can be written as $\sum_{i \in M} \left(\sum_{j \in J} Y_{ij} p_{ij}\right)^p$. We will bound the expected ℓ_p^p-norm of each machine individually, and then use linearity of expectation over all the machines.

Our main technical tool in bounding the expected ℓ_p^p-norm of a single machine will be the following theorem (see e.g., [**?**] for a proof).

Theorem 27 *[] Let $W_1, W_2, \ldots, W_n$ be independent non-negative random variables. Let $p > 1$ and $K_p = \Theta\left(\frac{p}{\log p}\right)$. Then,*

$$\mathbb{E}\left[\left(\sum_{j=1}^{n} W_j\right)^p\right] \leq (K_p)^p \cdot \max\left(\left(\sum_{j=1}^{n} \mathbb{E}[W_j]\right)^p, \sum_{j=1}^{n} \mathbb{E}\left[(W_j)^p\right]\right).$$

Ideally, we would like to use this theorem directly with $W_j = Y_{ij} p_{ij}$. This is indeed possible if $x_i(j) \geq \frac{1}{\alpha}$ since the assignment of such jobs j to machine i are independent of each other. However, the assignment of jobs j for which $x_i(j) < \frac{1}{\alpha}$ are *not* independent; they depend on each other via the random variable X_i which denotes whether machine i is open or not. Let j_i be the job that opened machine i, i.e. $X_i(j_i - 1) = 0$ and $X_i(j_i) = 1$. Conditioned on j_i, the variables Y_{ij} for $j \geq j_i$ are indeed independent. First, we will reduce the conditioning to a single indicator random variable. Define an indicator random variable $X_i = 1$ if and only if machine i is open in the integer solution after all n jobs have been assigned. By Eqn. 5.2, $X_i = 1$ with probability $\min(\alpha x_i, 1)$, where x_i is the fractional variable after all n jobs have been (fractionally) assigned. Now, define a binary random variable $\tilde{Y}_{ij}$ with the following properties:

- if $X_i = 0$, then $\tilde{Y}_{ij} = 0$,
- else if job j is not assigned via case 2, then $\tilde{Y}_{ij} = Y_{ij}$,
- else, $\tilde{Y}_{ij} = 1$ with probability z_{ij}; furthermore, in this case, using shared randomness, we ensure that $\tilde{Y}_{ij} = 1$ whenever $Y_{ij} = 1$.

The last condition can be met since in case 2, $\mathbb{P}[Y_{ij} = 1] = \frac{z_{ij}}{\sum_{i \in M_o^{(0)}(j)} z_{ij}} \leq z_{ij}$. Note that conditioned on $X_i = 1$, $\tilde{Y}_{ij}$ for different jobs j are independent random variables. Furthermore, $\tilde{Y}_{ij}$ *stochastically dominates* Y_{ij}, i.e. $Y_{ij} = 1$ implies $\tilde{Y}_{ij} = 1$. Therefore, it suffices to bound $\left(\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij} p_{ij}]\right)^p$ and $\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij}] (p_{ij})^p$, conditioned on $X_i = 1$. In the next lemma, we bound the first term.

Lemma 28 *[] For any machine i, conditioned on the event $X_i = 1$, we have*

$$\left(\sum_{j \in J} \mathbb{E}[\tilde{Y}_{ij} p_{ij}]\right)^p \leq \left(5c_i^{1/p} + \sum_{j \in J_1^{(i)}} y_{ij} p_{ij}\right)^p.$$

Proof: We consider two phases for machine i: $x_i < 1$ and $x_i = 1$. Recall that the jobs assigned in the first phase are denoted $J_0^{(i)}$ and those in the second phase are denoted $J_1^{(i)}$. First, we note that for jobs $j \in J_1^{(i)}$, $\mathbb{E}[\tilde{Y}_{ij}] \leq y_{ij}$. Therefore, $\sum_{j \in J_0^{(i)}} \mathbb{E}[\tilde{Y}_{ij} p_{ij}] \leq \sum_{j \in J_0^{(i)}} y_{ij} p_{ij}$. On the other hand, for jobs $j \in J_0^{(i)}$, we need to distinguish between jobs assigned via case 2 while $x_i(j) < \frac{1}{\alpha}$ (call this set $J_0^{(i)}(2)$) and those that are assigned via case 3 after $x_i(j) \geq \frac{1}{\alpha}$ (call this set $J_0^{(i)}(3)$). Then,

$$\sum_{j \in J_0^{(i)}} \mathbb{E}[\tilde{Y}_{ij} p_{ij}] \leq \sum_{j \in J_0^{(i)}(2)} z_{ij} p_{ij} + \sum_{j \in J_0^{(i)}(3)} y_{ij} p_{ij} \leq \frac{4}{\alpha} \sum_{j \in J_0^{(i)}(2)} \frac{y_{ij} p_{ij}}{x_i(j)} + \sum_{j \in J_0^{(i)}(3)} y_{ij} p_{ij}$$

$$\leq \frac{4}{\alpha}\int_{1/m}^{1/\alpha} c_i^{1/p}\frac{dx}{x} + c_i^{1/p}\left(1-\frac{1}{\alpha}\right) \leq \frac{4}{\alpha}\int_{1/m}^{1} c_i^{1/p}\frac{dx}{x} + c_i^{1/p} \leq 5c_i^{1/p},$$

since $\alpha = 48\ln(mn) \geq \ln m$. Combining all jobs,

$$\left(\sum_{j\in J}\mathbb{E}[\tilde{Y}_{ij}p_{ij}]\right)^p \leq \left(5c_i^{1/p} + \sum_{j\in J_1^{(i)}} y_{ij}p_{ij}\right)^p.$$

■ Next, we bound $\sum_{j\in J}\mathbb{E}[\tilde{Y}_{ij}]\,(p_{ij})^p$, conditioned on $X_i = 1$.

Lemma 29 *[] For any machine i, conditioned on the event $X_i = 1$, we have*

$$\sum_{j\in J}\mathbb{E}[\tilde{Y}_{ij}]\,(p_{ij})^p \leq 5c_i + \sum_{j\in J_1^{(i)}} y_{ij}\,(p_{ij})^p.$$

Proof: As in the previous proof, we consider two phases for machine i: $x_i < 1$ and $x_i = 1$. As earlier, the set of jobs assigned in the first phase is denoted $J_0^{(i)}$ and that assigned in the second phase is denoted $J_1^{(i)}$. First, we note that for jobs $j \in J_1^{(i)}$, $\mathbb{E}[\tilde{Y}_{ij}] \leq y_{ij}$. Therefore, $\sum_{j\in J_0^{(i)}}\mathbb{E}[\tilde{Y}_{ij}\,(p_{ij})^p] \leq \sum_{j\in J_0^{(i)}} y_{ij}\,(p_{ij})^p$. On the other hand, for jobs $j \in J_0^{(i)}$, we need to distinguish between jobs assigned via case 2 while $x_i(j) < \frac{1}{\alpha}$ (called $J_0^{(i)}(2)$) and those that are assigned via case 3 after $x_i(j) \geq \frac{1}{\alpha}$ (called $J_0^{(i)}(3)$). Then,

$$\begin{aligned}
&\sum_{j\in J_0^{(i)}}\mathbb{E}[\tilde{Y}_{ij}]\,(p_{ij})^p \leq \sum_{j\in J_0^{(i)}(2)} z_{ij}\,(p_{ij})^p + \sum_{j\in J_0^{(i)}(3)} y_{ij}\,(p_{ij})^p \\
&\leq \frac{4}{\alpha}\sum_{j\in J_0^{(i)}(2)}\frac{y_{ij}\,(p_{ij})^p}{x_i(j)} + \sum_{j\in J_0^{(i)}(3)} y_{ij}\,(p_{ij})^p \leq \frac{4}{\alpha}\int_{1/m}^{1/\alpha} c_i\frac{dx}{x} + c_i(1-1/\alpha) \\
&\leq \quad \frac{4}{\alpha}\int_{1/m}^{1} c_i\frac{dx}{x} + c_i \leq 5c_i,
\end{aligned}$$

since $\alpha = 48\ln(mn) \geq \ln m$. Combining all jobs,

$$\sum_{j\in J}\mathbb{E}[\tilde{Y}_{ij}]\,(p_{ij})^p \leq 5c_i + \sum_{j\in J_1^{(i)}} y_{ij}\,(p_{ij})^p.$$

■ Finally, we apply Theorem 27 to Lemmas 28 and 29, and remove the conditioning on X_i.

Theorem 30 *[]For any machine i,*

$$\mathbb{E}\left[\left(\sum_{j\in J} Y_{ij}p_{ij}\right)^p\right] \leq ((5\alpha)^{1/p}K_p)^p\Phi_i,$$

where $K_p = \theta\left(\frac{p}{\log p}\right)$.

Proof: For any machine i, conditioned on the event $X_i = 1$, we have

$$\begin{aligned}
&\mathbb{E}\left[\left(\sum_{j\in J} Y_{ij}p_{ij}\right)^p\right] \le \mathbb{E}\left[\left(\sum_{j\in J} \tilde{Y}_{ij}p_{ij}\right)^p\right] \quad \text{(since } \tilde{Y}_{ij} \text{ stochastically dominates } Y_{ij}\text{)} \\
&\le (K_p)^p \max\left(\left(\sum_{j\in J} \mathbb{E}[\tilde{Y}_{ij}p_{ij}]\right)^p, \sum_{j\in J} \mathbb{E}[\tilde{Y}_{ij}]\,(p_{ij})^p\right) \quad \text{(using Theorem 27)} \\
&\le (K_p)^p \max\left(\left(5c_i^{1/p} + \sum_{j\in J_1^{(i)}} y_{ij}p_{ij}\right)^p, 5c_i + \sum_{j\in J_1^{(i)}} y_{ij}\,(p_{ij})^p\right).
\end{aligned}$$

We now have three cases. First, suppose machine i satisfies $x_i = 1$ after all the jobs have been fractionally assigned. Then $X_i = 1$ deterministically, and the above inequality holds unconditionally. Therefore,

$$\begin{aligned}
&\mathbb{E}\left[\left(\sum_{j\in J} Y_{ij}p_{ij}\right)^p\right] \le (K_p)^p \max\left(\left(5c_i^{1/p} + \sum_{j\in J_1^{(i)}} y_{ij}p_{ij}\right)^p, 5c_i + \sum_{j\in J_1^{(i)}} y_{ij}\,(p_{ij})^p\right) \\
&\le (5K_p)^p\,\Phi_i.
\end{aligned}$$

Next, consider machines i such that $\frac{1}{\alpha} \le x_i < 1$ after all the jobs have been fractionally assigned. As in the previous case, $X_i = 1$ deterministically, and therefore the above inequality holds unconditionally. However, for such machines, $J_1^{(i)} = \emptyset$. Therefore,

$$\begin{aligned}
&\mathbb{E}\left[\left(\sum_{j\in J} Y_{ij}p_{ij}\right)^p\right] \le (K_p)^p \max\left(\left(5c_i^{1/p}\right)^p, 5c_i\right) \\
&= 5\,(K_p)^p\,c_i \le \left((5\alpha)^{1/p}K_p\right)^p \Phi_i.
\end{aligned}$$

Finally, consider machines i such that $x_i < \frac{1}{\alpha}$ after all the jobs have been fractionally assigned. As in the previous case, for such machines, $J_1^{(i)} = \emptyset$. However, $X_i = 1$ with probability αx_i. Therefore,

$$\begin{aligned}
&\mathbb{E}\left[\left(\sum_{j\in J} Y_{ij}p_{ij}\right)^p\right] = \mathbb{E}\left[\left(\sum_{j\in J} Y_{ij}p_{ij}\right)^p \middle| X_i = 1\right] \cdot \mathbb{P}[X_i = 1] \\
&\le (K_p)^p\,(5c_i)\alpha x_i \le \left((5\alpha)^{1/p}K_p\right)^p \Phi_i.
\end{aligned}$$

■ This completes the proof of Theorem **??**.

5.5 Online Rounding for UMSC with ℓ_1 norm

We now present an online rounding algorithm specifically tailored to the important special case of $p = 1$, i.e., the ℓ_1-norm. The rule for opening machines is identical (with a smaller value of α that we will shortly calculate) to the rounding algorithm for general p. However, the assignment rule for a job is now simpler and is given in Algorithm 4. Here, $M(j)$ denotes the machines sorted in non-decreasing order of p_{ij} and $M_{1/2}(j)$ is the minimal prefix of $M(j)$ that satisfies $\sum_{i\in M_{1/2}(j)} y_{ij} \ge 1/2$. As earlier, for clarity, we use two copies of

Opening Machines: For every machine i whose blue copy is closed, open it with probability $\min\left(\frac{\alpha(x_i(j)-x_i(j-1))}{1-\alpha\cdot x_i(j-1)}, 1\right)$. (Eqn. 5.2 is satisfied by this rule using conditional probabilities.
Assigning Job j:
- if $M_o(j)\cap M_{1/2}(j)\neq\emptyset$, then assign to blue copy of any machine in $M_o(j)\cap M_{1/2}(j)$,
- else assign to red copy of machine $i^*=\arg\min_{i\in M} p_{ij}$, after opening it if necessary.

Algorithm 4: Assignment of a Single Job by the Integer Algorithm for the ℓ_1-norm

each machine, a blue copy and a red copy, and let $M_o(j)$ be the machines whose blue copies are open after job j.

5.6 Analysis

First, we argue about the expected cost of the solution. To bound the cost of red copies, we show that Case 2 has low probability.

Lemma 31 *[] For any job j, the probability of case 2 is at most* $\exp(-\alpha/4)$.

Proof: Note that

$$\sum_{i\in M_{1/2}(j)} x_i(j) \geq \sum_{i\in M_{1/2}(j)} \frac{y_{ij}}{2} \geq \frac{1}{4}.$$

Therefore, the probability of case 2 is

$$\prod_{i\in M_{1/2}(j)} (1-\alpha x_i(j)) \leq \left(1-\frac{\alpha\sum_{i\in M_{1/2}(j)} x_i(j)}{k}\right)^k$$
$$\leq \exp\left(-\alpha\sum_{i\in M_{1/2}(j)} x_i(j)\right) \leq \exp\left(-\alpha/4\right).$$

■ We choose $\alpha = 4\ln n$ to obtain the following corollary.

Corollary 32 *[] For any job j, the probability of case 2 is at most* $\frac{1}{n}$.

Recall that the cost of each individual machine is at most m. Using linearity of expectation and the above corollary, we can now claim that the expected cost of red copies of machines is at most m. Similarly, using linearity of expectation and Eqn. 5.2, we can claim that the expected cost of blue copies of machines is $\sum_{i\in M} c_i\alpha x_i \leq \alpha\Phi$. Overall, we get the following bound for the cost of machines opened by the integer algorithm.

Lemma 33 *[] The total expected cost of machines in the integer algorithm is at most* $(\alpha+1)\Phi = O(\ln n)\Phi$.

We are now left to bound the ℓ_1-norm of the assignment. First, consider the red copies of machines. Note that the assignment of jobs in Case 2 follows a greedy algorithm assuming all machines are open. Therefore, the ℓ_1-norm of red copies of machines is optimal. The next lemma complements this observation by bounding the ℓ_1-norm of blue copies of machines.

Lemma 34 *[] The expected ℓ_1-norm of blue copies of machines is at most* 2Φ.

Proof: Suppose we assigned job j to the blue copy of machine $\hat{i}$. Also, let $k(j)$ be the last machine in the prefix $M_{1/2}(j)$ and let $\overline{M}_{1/2}(j) = (M \setminus M_{1/2}(j)) \cup \{k(j)\}$. Then, we have $\sum_{i \in \overline{M}_{1/2}(j)} y_{ij} \geq 1/2$ by minimality of the prefix $M_{1/2}(j)$ and $p_{\hat{i}j} \leq p_{ij}$ for all machines $i \in \overline{M}_{1/2}(j)$. Then, the increase in ℓ_1-norm of the integer solution is is $p_{\hat{i}j}$ whereas the corresponding increase in Φ for the fractional solution is

$$\sum_{i \in M} y_{ij} p_{ij} \geq \sum_{i \in \overline{M}_{1/2}(j)} y_{ij} p_{ij} \geq p_{\hat{i}j} \sum_{i \in \overline{M}_{1/2}(j)} y_{ij} \geq \frac{p_{\hat{i}j}}{2}.$$

The lemma now follows by summing over all jobs. ■ This completes the proof of Theorem **??**.

Appendix A: Missing Proofs from Section 3

In this section we give some missing proofs about the (primal) covering framework of §3

A.1 Proof of Theorem ??

≪Anupam's Comment A.1: Don't think we need this any more? If so, fix references.≫ AG A.1

Let f be a homogeneous degree-p polynomial (i.e. sum of monomials each having total degree exactly p) which satisfies Assumptions **??** and **??**. Fix any constant $c > 0$. The following facts are immediate:

$$\frac{\nabla_\ell f(z)}{\nabla_\ell f(cz)} = \frac{1}{c^{p-1}}$$

$$z^\intercal \nabla f(z) = p \cdot f(z)$$

$$\frac{z^\intercal \nabla f(z) - f(z)}{f(cz)} = \frac{(p-1)f(z)}{f(cz)} = \frac{p-1}{c^p}$$

Plugging this into (**??**) we get:

$$\frac{\text{Dual}}{\text{Primal}} \geq \left(\frac{1}{Lc^{p-1}} - \frac{(p-1)}{c^p}\right), \tag{A.1}$$

where $L := 4\ln(1+2d^2)$. Maximizing the above expression over c, we obtain a primal-dual ratio of $1/(pL)^p$. This completes the proof of Theorem **??**.

Remark 35 *[] The above result also holds if function f is the sum of linear functions and p^{th} powers of linear functions, i.e. $f(x) = \sum_{k=1}^K (B_k x)^p + \sum_{l=1}^L (D_l x)$ where each $B_k \in \mathbb{R}_+^n$ and $D_l \in \mathbb{R}_+^n$. For this case also we obtain an $O(p \log d)^p$-competitive algorithm.*

Note that here $\nabla_i f(x) = p\sum_{k=1}^K b_{ki} \cdot (B_k x)^{p-1} + \sum_{l=1}^L d_{li}$. So for all $c \geq 1$, $x \in \mathbb{R}_+^n$ and $1 \leq i \leq n$:

$$\frac{\nabla_i f(z)}{\nabla_i f(cz)} \geq \min\left\{\frac{1}{c^{p-1}}, 1\right\} = (1/c)^{p-1}$$

$$\frac{\sum_{i=1}^n z_i \cdot \nabla f(z)_i - f(z)}{f(cz)} = \frac{p\sum_{k=1}^K (B_k z)^p + \sum_{l=1}^L (D_l z) - f(z)}{f(cz)} = \frac{(p-1)\sum_{k=1}^K (B_k z)^p}{f(cz)} \leq (p-1)\frac{1}{c^p}.$$

Exactly as above, we get an $O(p \log d)^p$-competitive algorithm using (**??**).

A.2 Solving Mixed Packing-Covering LPs Online

We consider the problem of solving a mixed packing-covering linear program online, as defined by [**?**]. The covering constraints $Ax \geq 1$ arrive online, as in the above setting. There are also K "packing constraints" $\sum_{i=1}^n b_{ki} \cdot x_i \leq \lambda_k$ for $k \in [K]$ that are given up-front. The right sides λ_k of these packing constraints are themselves variables, and the objective is to minimize $\sum_{k=1}^K \lambda_k^p$ or alternatively, $\|\lambda\|_p = \sqrt[p]{\sum_{k=1}^K \lambda_k^p}$. All the entries in the constraint matrices $A = (a_{ji})$ and $B = (b_{ki})$ are non-negative. Using the convex function $f(x) = \|Bx\|_p^p = \sum_{k=1}^K (\sum_{i=1}^n b_{ki} \cdot x_i)^p$, Theorem 6 immediately implies the following result.

Theorem 36 *[ℓ_p-norm of Packing Constraints] There is an $O(p \log d)$-competitive online algorithm for fractional covering with the objective of minimizing ℓ_p-norm of multiple packing constraints.*

When $p = \Theta(\log K)$, the norms $\|x\|_p = \Theta(\|x\|_\infty)$ for all $x \in \mathbb{R}^K$. Hence for the online mixed packing-covering LP (OMPC) problem [**?**], Theorem 36 gives an improved $O(\log d \log K)$-competitive ratio, where d is the row-sparsity of the matrix A and K the number of packing constraints. This competitive ratio is asymptotically best possible [**?**, Theorem 1.2].

≪Vish's Comment A.2: Add the lower bound for general p here.≫ VN A.2

A.3 Proof of Theorem 7: ℓ_p Set Cover

Consider the online set-cover problem [**?**] with n sets $\{S_j\}_{j=1}^n$ over some ground set U. Apart from the set system, we are also given K cost functions $B_k : [n] \to \mathbb{R}_+$ for $k \in [K]$. Elements from U arrive online and must be covered by some set upon arrival; the decision to select a set into the solution is irrevocable. The goal is to maintain a set-cover that minimizes the ℓ_p norm of the K cost functions. We use the above fractional online algorithm along with a rounding scheme (similar to [**?**]) to obtain Theorem 7 (paraphrased below).

Theorem 37 *[ℓ_p-norm Set Cover] There is an $O\left(\frac{p^3}{\log p} \log d \log r\right)$-competitive randomized online algorithm for set cover minimizing the ℓ_p-norm of multiple cost-functions. Here d is the maximum number of sets containing any element, and $r = |U|$ is the number of elements.*

Proof: We use the following convex relaxation. There is a variable x_j for each set $j \in [n]$ which denotes whether this set is chosen.

$$\begin{aligned} \min \quad & g(x) = \sum_{k=1}^{K} \left(\sum_{j=1}^{n} b_{kj} \cdot x_j \right)^p + \sum_{j=1}^{n} \left(\sum_{k=1}^{K} b_{kj}^p \right) \cdot x_j \\ s.t. \quad & \sum_{j : e \in S_j} x_j \geq 1, \qquad \forall e \in U \\ & x \geq 0. \end{aligned}$$

We can use our framework to solve this fractional convex covering problem online. Although the objective has a linear term in addition to the p-powers, we obtain an $O(p \log d)^p$-competitive algorithm as noted in Remark 35. ≪Anupam's Comment A.3: Should change this, given that the main covering theorem changed.≫ AG A.3

Let $C^\star$ denote the p^{th} power of the optimal objective of the given set cover instance. Then it is clear that the optimal objective of the above fractional relaxation is at most $2C^\star$. Thus the objective of our fractional online solution $g(x) = O(p \log d)^p \cdot C^\star$.

To get an integer solution, we use a simple online randomized rounding algorithm. For each set $j \in [n]$, define X_j to be a $\{0,1\}$-random variable with $\Pr[X_j = 1] = \min\{4p \log r \cdot x_j, 1\}$. This can easily be implemented online. It is easy to see by a Chernoff bound that for each element e, it is not covered with probability at most $\frac{1}{r^{2p}}$. If an element e is not covered by this rounding, we choose the set minimizing $\min_{j=1}^n \{\sum_{k=1}^K b_{kj}^p : e \in S_j\}$; let $\bar{e} \in [n]$ index this set and $C_{\bar{e}} = \sum_{k=1}^K b_{k\bar{e}}^p$. Observe that $C_{\bar{e}} \leq C^\star$ for all $e \in U$.

To bound the ℓ_p-norm of the cost, let $C_k = \sum_{j=1}^n b_{kj} \cdot X_j$ be the cost of the randomly rounded solution under the k^{th} cost function, and let $C := \sum_{k=1}^K C_k^p$. Also for each element $e \in U$, define:

- $D_{ek} = b_{k\bar{e}}$ for all $k \in [K]$ and $D_e = C_{\bar{e}}$, if e is not covered by the rounding.
- $D_{ek} = 0$ for all $k \in [K]$ and $D_e = 0$, otherwise.

Note that $D_e = \sum_{k=1}^{K} D_{ek}^p$. The p^{th} power of the objective function is:

$$\begin{aligned} \overline{C} &= \sum_{k=1}^{K} \left(C_k + \sum_{e \in U} D_{ek} \right)^p \leq 2^p \sum_{k=1}^{K} C_k^p + 2^p \sum_{k=1}^{K} \left(\sum_{e \in U} D_{ek} \right)^p \leq 2^p \cdot C + 2^p \sum_{k=1}^{K} r^p \sum_{e \in U} D_{ek}^p \\ &= 2^p \cdot C + (2r)^p \sum_{e \in U} D_e \end{aligned} \tag{A.2}$$

We now bound $\mathbb{E}[\overline{C}]$ using (A.2). Observe that $\mathbb{E}[C_k] \leq 4p \log r \cdot \sum_{j=1}^{n} b_{kj} \cdot x_j$. Since each C_k is the sum of independent non-negative random variables, we can bound $\mathbb{E}[C_k^p]$ using a concentration inequality involving p^{th} moments [**?**]:

$$\mathbb{E}[C_k^p] \leq K_p \cdot \left(\mathbb{E}[C_k]^p + \sum_{j=1}^{n} \mathbb{E}[b_{kj}^p \cdot X_j^p] \right) \leq K_p \cdot \left((4p \log r)^p \left(\sum_{j=1}^{n} b_{kj} \cdot x_j \right)^p + 4p \log r \sum_{j=1}^{n} b_{kj}^p \cdot x_j \right).$$

Above $K_p = O(p/\log p)^p$. By linearity of expectation,

$$\mathbb{E}[C] = \sum_{k=1}^{K} \mathbb{E}[C_k^p] \leq K_p (4p \log r)^p \sum_{k=1}^{K} \left(\left(\sum_{j=1}^{n} b_{kj} \cdot x_j \right)^p + \sum_{j=1}^{n} b_{kj}^p \cdot x_j \right) = K_p (4p \log r)^p \cdot g(x).$$

Thus we have $\mathbb{E}[C] = O\left(\frac{p^3}{\log p} \cdot \log d \cdot \log r \right)^p \cdot C^\star$.

Observe that $\mathbb{E}\left[\sum_{e \in U} D_e\right] = \sum_{e \in U} \Pr[e \text{ uncovered}] \cdot C_{\bar{e}} \leq r^{-2p} \cdot \sum_{e \in U} C^\star = r^{1-2p} \cdot C^\star$. Using these bounds in (A.2), we have $\mathbb{E}[\overline{C}] \leq 2^p \cdot \mathbb{E}[C] + (2r)^p \sum_{e \in U} \mathbb{E}[D_e] = O\left(\frac{p^3}{\log p} \cdot \log d \cdot \log r \right)^p \cdot C^\star$. ■

Appendix B: Competitive Analysis for Theorem 11

Lemma 38 *[Convexity of f] Suppose that for some $p > 1$, for all $\mathbf{x} \in \mathbb{R}_+^n$, $\langle \nabla f(\mathbf{x}), \mathbf{x} \rangle \leq p \cdot f(\mathbf{x})$. Then, the following statements hold.*

(a) For $\delta \geq 1$, for $\mathbf{x} \in \mathbb{R}_+^n$, $f(\delta \mathbf{x}) \leq \delta^p f(\mathbf{x})$.
(b) For $0 < \gamma \leq 1$, for $\mathbf{z} \in \mathbb{R}_+^n$, $f^*(\gamma \mathbf{z}) \leq \gamma^{\frac{p}{p-1}} \cdot f^*(\mathbf{z})$.
(c) Suppose further that f is convex. Then, for all $\mathbf{x} \in \mathbb{R}_+^n$, $f^*(\nabla f(\mathbf{x})) = \langle \mathbf{x}, \nabla f(\mathbf{x}) \rangle - f(\mathbf{x}) \leq (p-1) \cdot f(\mathbf{x})$.

In particular, statements (b) and (c) implies that for $0 \leq \gamma \leq 1$, $f^*(\gamma \cdot \nabla f(\mathbf{x})) \leq \gamma^{\frac{p}{p-1}} \cdot (p-1) \cdot f(\mathbf{x})$.

Proof: For statement (a), define the function $g : [1, +\infty) \to \mathbb{R}$ by $g(\theta) := \ln f(\theta \mathbf{x})$. Then, $g'(\theta) = \frac{\langle f(\theta \mathbf{x}), \mathbf{x} \rangle}{f(\theta \mathbf{x})} \leq \frac{p}{\theta}$, where the last inequality follows because $\langle f(\theta \mathbf{x}), \theta \mathbf{x} \rangle \leq p f(\theta \mathbf{x})$. Integrating over $\theta \in [1, \delta]$ gives $g(\delta) - g(1) \leq p \ln \delta$, which is equivalent to $f(\delta \mathbf{x}) \leq \delta^p f(\mathbf{x})$.

For statement (b), for $0 < \gamma \leq 1$, let $\delta := (\frac{1}{\gamma})^{\frac{1}{p-1}} \geq 1$, we have

$$\begin{aligned} f^*(\gamma \mathbf{z}) &= \max_{\mathbf{x} \in R^n_+} \{\langle \mathbf{x}, \gamma \mathbf{z} \rangle - f(\mathbf{x})\} = \gamma^{\frac{p}{p-1}} \max_{\mathbf{x} \in R^n_+} \{\langle \delta \mathbf{x}, \mathbf{z} \rangle - \delta^p f(\mathbf{x})\} \\ &\leq \gamma^{\frac{p}{p-1}} \max_{\mathbf{x} \in R^n_+} \{\langle \delta \mathbf{x}, \mathbf{z} \rangle - f(\delta \mathbf{x})\} = \gamma^{\frac{p}{p-1}} f^*(\mathbf{z}) \ , \end{aligned}$$

where the inequality follows from statement (a).

For statement (c), observe that $f^*(\nabla f(\mathbf{x})) = \max_{\mathbf{w} \in \mathbb{R}^n_+} h(\mathbf{w})$, where $h(\mathbf{w}) := \langle \mathbf{w}, \nabla f(\mathbf{x}) \rangle - f(\mathbf{w})$. Hence, $\nabla h(\mathbf{w}) = \nabla f(\mathbf{x}) - \nabla f(\mathbf{w})$. Since f is a convex function, h is a concave function and $\nabla h(\mathbf{x}) = \mathbf{0}$; it follows that h is maximized when $\mathbf{w} = \mathbf{x}$. Therefore, we have $f^*(\nabla f(\mathbf{x})) = \langle \mathbf{x}, \nabla f(\mathbf{x}) \rangle - f(\mathbf{x}) \leq (p-1) \cdot f(\mathbf{x})$, where the last inequality follows from the assumption on f. ■

Lemma 39 *[Bounding Increase in y] For $k \in [m]$, let $z^{(k)}$ denote the vector z at the end of round k, where $z^{(0)} := 0$. Then, at the end of round k when y_k stops increasing (by Lemma 12)*

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

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

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

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

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

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

Proof: [Theorem 11] Suppose $z^{(m)} = A^T y$ is the vector at the end of the algorithm, and $x^{(m)} = \nabla f^*(\rho z^{(m)})$. Then, we have

$$C(x^{(m)}) = f(\nabla f^*(\rho z^{(m)})) \leq (p-1) \cdot f^*(\rho z^{(m)}) \ ,$$

where the inequality follows from applying Lemma 38(c) with the roles of f and f^* reversed.

On the other hand,

$$P(y) = \textstyle\sum_{k \in [m]} y_k - f^*(z^{(m)}) \geq \tfrac{1}{\rho} \cdot f^*(\rho z^{(m)}) - f^*(z^{(m)}) \ . \tag{B.1}$$

Hence, it follows that

$$\frac{C(x)}{P(y)} \leq \frac{(p-1) \cdot f^*(\rho z^{(m)})}{\frac{1}{\rho} \cdot f^*(\rho z^{(m)}) - f^*(z^{(m)})} \leq \frac{(p-1) \cdot \rho^\lambda \cdot f^*(z^{(m)})}{\frac{1}{\rho} \cdot \rho^\lambda \cdot f^*(z^{(m)}) - f^*(z^{(m)})} = (p-1) \cdot \frac{\rho^\lambda}{\rho^{\lambda-1} - 1} \ , \tag{B.2}$$

where the penultimate inequality follows because the assumption $\nabla f^*(\rho z) \geq \rho^{\lambda-1} \cdot \nabla f^*(z)$ implies that $f^*(\rho z^{(m)}) = \rho \int_{z=0}^{z^{(m)}} \langle \nabla f^*(\rho z), dz \rangle \geq \rho^{\lambda} \int_{z=0}^{z^{(m)}} \langle \nabla f^*(z), dz \rangle = \rho^{\lambda} \cdot f^*(z^{(m)}))$, and the function $t \mapsto \frac{t}{\frac{t}{\rho} - f^*(z^{(m)})}$ is decreasing.

Choosing $\rho := \lambda^{\frac{1}{\lambda-1}}$ and observing that $x^{(m)}$ is feasible for the covering problem, we have

$$\frac{P^{\text{opt}}}{P(y)} \leq \frac{C^{\text{opt}}}{P(y)} \leq \frac{C(x^{(m)})}{P(y)} \leq (p-1) \cdot \frac{\lambda^{\frac{\lambda}{\lambda-1}}}{\lambda - 1} \ .$$

The result then follows because $\lambda^{\frac{1}{\lambda-1}} = \big(1 + (\lambda - 1)\big)^{\frac{1}{\lambda-1}} \leq e$. ■

Appendix C: Applications of the Packing Framework

In this section, we explain how to extend our algorithm for the online convex packing problem to a more general problem known as online combinatorial auction with production cost.

In an online combinatorial auction, there is a seller with n types of items (i.e., resources) that are known upfront and a convex production cost function $g = f^* : \mathbb{R}_+^n \to \mathbb{R}_+$. Producing z_i units of resource i for all $i \in [n]$ incurs a production cost of $f^*(z)$. There are m buyers (i.e., requests) that arrive online. Each buyer j is associated with a value function[5] $v_j : 2^{[n]} \to \mathbb{R}_+$ such that $v_j(S)$ is buyer j's value for getting a subset of items $S \subseteq [n]$. On the arrival of a buyer, the seller must choose a subset of items S_j to allocate to the buyer immediately without any information of future buyers. The objective is to maximize the social welfare, which is defined as total value of buyers, $\sum_{j \in [m]} v_j(S_j)$, minus the production cost $f^*(z)$, where z_i is the number of j's such that $i \in S_j$.

We assume that f^* have the same properties as in Section 4, i.e., f^* is convex and differentiable, and both f^* and ∇f^* are monotone, and $f^*(0) = 0$. In addition, we shall consider a couple more technical assumptions on f^*, which are true for most interesting functions such as polynomials.

C.1 Proof of Theorem ??

Consider the following standard convex program relaxation of combinatorial auction with production costs and its Fenchel dual program:

$$\begin{aligned} \max_{y \geq 0} \quad & P(y) := \textstyle\sum_{j \in [m]} \sum_{S \subseteq [n]} v_j(S) \cdot y_{jS} - f^*(z) \quad \text{s.t.} \\ \forall i \in [n]: \quad & \textstyle\sum_{j \in [m]} \sum_{S \ni i} y_{jS} = z_i \\ \forall j \in [m]: \quad & \textstyle\sum_{S \subseteq [n]} y_{jS} \leq 1 \end{aligned}$$

$$\begin{aligned} \min_{x,u \geq 0} \quad & C(x,u) := f(x) + \textstyle\sum_{j \in [m]} u_j \quad \text{s.t.} \\ \forall j \in [m], \forall S \subseteq [n]: \quad & \textstyle\sum_{i \in S} x_i + u_j \geq v_j(S) \end{aligned}$$

[5]For efficiency issues, we might need to assume that v_j is supermodular (i.e., for subsets A and B, $v_j(A) + v_j(B) \leq v_j(A \cap B) + v_j(A \cup B)$), but otherwise we do not need further assumptions.

Observe that in the packing problem, the objective function can be succinctly expressed as $P(y) := \langle v, y \rangle - f^*(A^T y)$, where the coordinates of v are indexed by $[m] \times 2^{[n]}$, and A is the $\{0,1\}$-matrix whose rows are indexed by $[m] \times 2^{[n]}$ and columns are indexed by $[n]$ such that for $j \in [m]$, $S \subseteq [n]$ and $i \in [n]$, the (jS, i)-th entry is 1 *iff* $i \in S$. We also denote $x_S := \sum_{i \in S} x_i$.

In this paper, we present an online algorithm that solves the above convex program fractionally. In each round $k \in [m]$, the value function v_k arrives, and the algorithm irrevocably chooses non-negative values for y_{kS} for all $S \subseteq [n]$ such that $\sum_{S \subseteq [n]} y_{kS} \leq 1$. Observe that the algorithm knows the rows of A in advance, although it may not know the number m of rounds. Translating fractional algorithms into integral one is relatively straightforward and readers are referred to Huang and Kim [**?**] for details. The algorithm is given in Algorithm 5.

Initialize $x := z := 0$
while buyer k with value function v_k arrives in round k **do**
  Set $y_k := 0$ and $u_k := 0$
  while t increases continuously from 0 to 1 **do**
    Define $r(t) := \max_{A \subseteq [n]} \gamma_A(t)$, where $\gamma_A(t) := v_k(A) - \sum_{i \in A} x_i(t)$.
    Pick any $S \subseteq [n]$ that attains $r(t)$.
    while $\gamma_S(t) \geq r(t) - \varepsilon$ **do**
      Increase t continuously:
        1. Increase y_{kS} at rate $\frac{dy_{kS}}{dt} = 1$
        2. To maintain the invariant $z = A^T y$, for each $i \in S$, z_i is increased at rate $\frac{dz_i}{dt} = 1$.
        3. As z is increased, we maintain $x := \nabla f^*(\rho z)$ for some parameter $\rho > 1$.
        4. Increase u_k at rate $\frac{du_k}{dt} = r(t)$. (Note that u_k is for analysis only.)

Algorithm 5: Online combinatorial auction with production cost

Explanation of Algorithm. In each round $k \in [m]$, the variables are changed continuously as functions of some time parameter $t \in [0,1]$ such that $t = 0$ corresponds to the beginning of round k, and round k finishes when t reaches 1. Initially, $y_k(0) := 0$ and $u_k(0) := 0$. The following invariants are maintained.

Invariant 1: The sum $\sum_{S \subseteq [n]} y_{kS}(t)$ increases at rate 1 with respect to t.

In fact, the algorithm ensures that at any time t, there is exactly one $S \subseteq [n]$ such that y_{kS} is increased at rate 1. This ensures that when t reaches 1, the sum $\sum_{S \subseteq [n]} y_{kS}(t)$ is 1 to maintain the feasibility of y. To decide which y_{kS}'s to increase at time t, a parameter is defined $r(t) := \max_{A \subseteq [n]} \gamma_A(t)$, where $\gamma_A(t) := v_k(A) - x_A(t)$. Intuitively, the sets S that attain $r(t)$ are the most worthwhile to be selected. One technical issue is that whether $r(t)$ can be computed efficiently. If the set function v_k is supermodular, then $r(t)$ can be computed efficiently.

Invariant 2: A parameter $\varepsilon > 0$ is chosen such that a variable y_{kS} is increased at time t only if $\gamma_S(t) \geq r(t) - \varepsilon$.

We shall see that this invariant is used to bound the competitive ratio. One might attempt to define Invariant 2 with $\varepsilon = 0$. The problem is that when y_{kS} is increased, the vectors $z = A^T y$ and $x := \nabla f^*(\rho z)$ will also be increased such that the set S might no longer satisfy $\gamma_S(t) \geq r(t)$, even when t is increased infinitesimally. To choose the value of ε and facilitate the implementation of the algorithm, we place some technical assumptions on f^*, which are true for interesting functions.

- The gradient ∇f^* is locally Lipschitz with respect to the ℓ_1-norm, i.e., for all z, for all $R > 0$, there

exists some L such that $\|z-a\|_1$, $\|z-b\|_1 \le R$ implies that $\|\nabla f^*(a) - \nabla f^*(b)\|_1 \le L \cdot \|a-b\|_1$.

- For all $R > 0$, the infimum $\inf_{\|z\|_1 \ge R} \frac{f^*(z)}{\|z\|_1}$ is positive. This assumption means that the production cost cannot be zero as long as some resource is being used, and must grow at least proportionately as more resources are used.

We shall see that it is sufficient to choose $\varepsilon := \frac{1}{10} \inf_{\|z\|_1 \ge R} \frac{f^*(z)}{\|z\|_1} > 0$, where R depends on the first value function v_1 and the local Lipschitz constant of ∇f^* around 0.

Continuous vs Discrete Increments. Observe that during round k, the algorithm needs to change the y_{kS} to increase when $\gamma_S(t) < r(t) - \varepsilon$; this means that $\gamma_S(t)$ must have decreased by at least ε from the time S is selected. Moreover, observe that $r(t) \ge \gamma_\emptyset(t) \ge 0$. Hence, it follows that there can be at most $2^n \cdot \frac{r(0)}{\varepsilon}$ changes of S before t reaches 1.

We show that the local Lipschitz property of ∇f^* implies that instead of monitoring $r(t)$ continuously, the algorithm can be implemented by discrete increments, even though it is more convenient to analyze it continuously.

Notice that in round k, the ℓ_1-norm of the vector $z = A^T y$ can increase by at most n, which happens if the complete set $[n]$ is chosen throughout. Hence, it follows that $\|\rho z\|_1$ can change by at most ρn according to the ℓ_1-distance. Let L be the local Lipschitz constant for ∇f^* in this vicinity of ρz during round k. Observe that the mapping $t \mapsto x(t) := \nabla f^*(\rho z(t))$ is $L\rho n$-Lipschitz with respect to the ℓ_1-norm.

Therefore, if t is increased by $\delta := \frac{\varepsilon}{L\rho n}$, $\|x(t)\|_1$ can increase by at most ε. Hence, it follows that if $\gamma_S(t_0) = r(t_0)$, then for all $t \in [t_0, t_0+\delta]$, we have $\gamma_S(t) \ge \gamma_S(t_0) - \varepsilon \ge r(t) - \varepsilon$, since $r(t)$ is non-increasing. As a result, we can increase t at increments of δ, and compute $r(t)$ and change subsets S for only $\frac{1}{\delta}$ times before t reaches 1, and round k finishes.

Feasibility of Covering Problem. Observe that to maintain the feasibility of (x,u), at the end of round k, one can simply set $u_k := \max_{S \subseteq [n]} v_k(S) - x_S$. However, to facilitate the competitive analysis, we increase u_k at rate $\frac{du_k}{dt} = r(t)$ throughout round k. The next lemma shows that this also maintains the feasibility of (x,u).

Lemma 40 *[Feasibility of (x,u)] After each round $k \in [m]$, for all $S \subseteq [n]$, $u_k \ge v_k(S) - x_S$.*

Proof: Recall that the time parameter $t \in [0,1]$ denotes the beginning of round k with $t = 0$ and the end of round k with $t = 1$. Observing that x is increased monotonically, for all $t \in [0,1]$, $x_S(t) \le x_S(1)$.

Hence, we have $\frac{du_k}{dt} = r(t) \ge v_k(S) - x_S(t) \ge v_k(S) - x_S(1)$. Integrating with respect to t from 0 to 1 gives the result. ∎

Competitive Analysis. The analysis is along the same lines as that of the convex packing problem. The main difference lies in how we bound the increase in y. Concretely, we will use the following lemma, which is an analogue of Lemma 39 in the convex packing problem.

Lemma 41 *[Bounding Increase in y] For $k \in [m]$, let $z^{(k)}$ denote the vector z at the end of round k, where $z^{(0)} := 0$. Then, at the end of round k,*

$$\sum_{S \subseteq [n]} v_k(S) \cdot y_{kS} \ge u_k + \frac{1}{\rho} \cdot \left(f^*(\rho z^{(k)}) - f^*(\rho z^{(k-1)}) \right) - \varepsilon R_k \ ,$$

where R_k is the amount of time in $[0,1]$ during round k in which some non-empty set S is chosen to increase the variable y_{kS}.

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

$$\langle v, y \rangle = \sum_{j \in [m]} \sum_{S \subseteq [n]} v_j(S) \cdot y_{jS} \geq \sum_{j \in [m]} u_j + \frac{1}{\rho} \cdot f^*\left(\rho z^{(m)}\right) - \varepsilon \sum_{j \in [m]} R_j \ .$$

Proof: Recall that by Invariant 2, at time $t \in [0,1]$, y_{kS} increases only if $\gamma_S(t) \geq r(t) - \varepsilon$. On the other hand, observe that if the empty set $S = \emptyset$ is chosen such that $y_{k\emptyset}$ is increased, both z and x remains the same. Hence, the invariant $\gamma_\emptyset(t) = r(t)$ is actually maintained with no error term.

We define the indicator function $\chi : [0,1] \to \{0,1\}$ such that $\chi(t) = 1$ *iff* a non-empty set S is chosen to increase y_{kS} at time t. Then, we have $\gamma_S(t) \geq r(t) - \varepsilon \cdot \chi(t)$

Recall that $\frac{du_k}{dt} = r(t)$. Hence, if y_{kS} is increased at time t, we have

$$v_k(S) \geq \frac{du_k}{dt} + x_S(t) - \varepsilon \cdot \chi(t).$$

By Invariant 1, $\sum_{S' \subseteq [n]} \frac{dy_{kS'}}{dt} = 1$. Observe that for other $S' \neq S$, $\frac{dy_{kS'}}{dt} = 0$. Hence, we can multiply the above equation by $\frac{dy_{kS}}{dt} = 1$, and include the zero terms in the sum for $S' \neq S$ to obtain the following.

$$\sum_{S \subseteq [n]} v_k(S) \cdot \frac{dy_{kS}}{dt} \geq \frac{du_k}{dt} + \sum_{S \subseteq [n]} \sum_{i \in S} x_i \cdot \frac{dy_{kS}}{dt} - \varepsilon \cdot \chi(t) = \frac{du_k}{dt} + \sum_{i \in [n]} x_i \cdot \frac{dz_i}{dt} - \varepsilon \cdot \chi(t),$$

where the last equality follows from interchanging the order of summation, and $\frac{dz_i}{dt} = \sum_{S \subseteq [n] : i \in S} \frac{dy_{kS}}{dt}$.

Observe that $x = \nabla f^*(\rho z)$. Hence, integrating with respect to t from 0 to 1, the vector z increases from $z^{(k-1)}$ to $z^{(k)}$, and we have:

$$\sum_{S \in [n]} v_k(S) \cdot y_{kS} \geq u_k + \int_{z = z^{(k-1)}}^{z^{(k)}} \langle \nabla f^*(\rho z), dz \rangle - \varepsilon \cdot R_k = u_k + \frac{1}{\rho} \cdot (f^*(\rho z^{(k)}) - f^*(\rho z^{(k-1)})) - \varepsilon \cdot R_k \ ,$$

where $R_k := \int_0^1 \chi(t) dt$ is the amount of time in which a non-empty set S is chosen to increase y_{kS}, and the last equality comes from the fundamental theorem of calculus for path integrals of vector fields. ∎

Proof: [Theorem **??**] We first use the trick in Section 4.2 to absorb the linear terms of f^* into $\langle v, y \rangle$ in the objective function $P(y) = \langle v, y \rangle - f^*(A^T y)$. Hence, we can assume that each term in f^* has degree at least 2.

By Lemma 41, after round m, we have

$$P(y) = \langle v, y \rangle - f^*(z^{(m)}) \geq \langle \mathbf{1}, u \rangle + \tfrac{1}{\rho} \cdot f^*\left(\rho z^{(m)}\right) - f^*(z^{(m)}) - \varepsilon R,$$

where $R := \sum_{j \in [m]} R_j$.

Recalling that $x = \nabla f^*(\rho z^{(m)})$, $C(x,u) := \langle \mathbf{1}, u \rangle + f(\nabla f^*(\rho z^{(m)})) \leq \langle \mathbf{1}, u \rangle + (p-1) \cdot f^*(\rho z^{(m)})$, where the last inequality follows from Lemma 38(c), since the polynomial f^* is has maximum degree p.

Hence, similar to inequality (B.2) in the proof of Theorem 11, we have:

$$\frac{C(x,u)}{P(y)} \leq \frac{\langle \mathbf{1},u\rangle + (p-1)\cdot f^*(\rho z^{(m)})}{\langle \mathbf{1},u\rangle + \frac{1}{\rho}\cdot f^*(\rho z^{(m)}) - f^*(z^{(m)}) - \varepsilon R} \leq \max\{1, \frac{(p-1)\cdot f^*(\rho z^{(m)})}{\frac{1}{\rho}\cdot f^*(\rho z^{(m)}) - f^*(z^{(m)}) - \varepsilon R}\}.$$

Observe that the second argument of the maximum operator is exactly the same expression appearing in inequality (B.2) apart from the negative εR error term in the denominator. We use the assumption that every term in f^* has degree at least $\lambda = 2$ and set $\rho := 2$.

We next show how to choose $\varepsilon > 0$ such that $\varepsilon R \leq \frac{1}{10} \cdot f^*(z^{(m)})$. Given the first value function v_1, we can give a lower bound on $\|z^{(1)}\|_1$ using the local Lipschitz property of f^*.

Let $\beta := \max_{S\subseteq[n]} v_1(S) - v_1(\emptyset)$. We can assume $\beta > 0$; otherwise, the empty set will be chosen in this round, and z and x remains zero.

Observe that as t increases from 0 to 1, $\|\rho z\|_1 \leq \rho n$. Suppose L is the local Lipschitz constant of ∇f^* in the ℓ_1-ball of radius ρn around 0. Then, the function $z \mapsto x := \nabla f^*(\rho z)$ has local Lipschitz constant ρL.

Observe that regardless of the value of ε, if the empty set is ever chosen, the norm $\|x\|_1$ must have increased by at least β, which means $\|z\|_1$ has increased by at least $\frac{\beta}{\rho L}$. On the other hand, if the empty set is never chosen, then as t increases, $\|z\|_1$ increases at rate at least 1. Hence, in any case, no matter what the value of ε is, at the end of the first round, $\|z^{(1)}\|_1 \geq R_0 := \min\{1, \frac{\beta}{\rho L}\}$.

Hence, after seeing the first value function v_1, the algorithm can choose $\varepsilon := \frac{1}{10}\inf_{\|z\|_1 \geq R_0} \frac{f^*(z)}{\|z\|_1}$. Observing that z increases monotonically throughout the algorithm, at the end of round m, $\|z^{(m)}\|_1 \geq R_0$. Hence, we have $\varepsilon R \leq \frac{1}{10} \cdot \frac{f^*(z^{(m)})}{\|z^{(m)}\|_1} \cdot R \leq \frac{1}{10} \cdot f^*(z^{(m)})$, where the last inequality follows because during the time period of measure R in which a non-empty set is chosen, $\|z\|_1$ increases at rate at least 1.

Therefore, by a similar analysis as in the proof of Theorem 11, the competitive ratio is $O(p)$, as required. ■