Online Mixed Packing and Covering*

Yossi Azar† Umang Bhaskar‡ Lisa Fleischer§ Debmalya Panigrahi¶

Abstract

Recent work has shown that the classical framework of solving optimization problems by obtaining a fractional solution to a linear program (LP) and rounding it to an integer solution can be extended to the online setting using primal-dual techniques. The success of this new framework for online optimization can be gauged from the fact that it has led to progress in several long-standing open questions. However, to the best of our knowledge, this framework has previously been applied to LPs containing only packing or only covering constraints, or minor variants of these. We extend this framework in a fundamental way by demonstrating that it can be used to solve mixed packing and covering LPs online, where packing constraints are given offline and covering constraints are received online. The objective is to minimize the maximum multiplicative factor by which any packing constraint is violated, while satisfying the covering constraints. Our results represent the first algorithm that obtains a polylogarithmic competitive ratio for solving mixed LPs online.

We then consider two canonical examples of mixed LPs: unrelated machine scheduling with startup costs, and capacity constrained facility location. We use ideas generated from our result for mixed packing and covering to obtain polylogarithmic-competitive algorithms for these problems. We also give lower bounds to show that the competitive ratios of our algorithms are nearly tight.

*Yossi Azar was supported in part by the Israel Science Foundation grant 1404/10, Umang Bhaskar and Lisa Fleischer by NSF grants CCF-0728869 and CCF-1016778, and Debmalya Panigrahi by NSF grant CCF-1117381.

†Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: azar@tau.ac.il. Part of this work was done while visiting Microsoft Research, Redmond.

‡Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125, USA. Email: umang@caltech.edu. Work done while a student at Dartmouth College.

§Department of Computer Science, Dartmouth College, 6211 Sudikoff Lab, Hanover NH 03755. Email: lkf@cs.dartmouth.edu.

¶Microsoft Research, Redmond, WA 98052. Email: depan@microsoft.com. Work done as a graduate student at MIT and as an intern at Microsoft Research, Redmond.

1 Introduction

Mixed packing and covering linear programs (LPs) model a wide range of problems in combinatorial optimization and operations research, including well-studied problems such as facility location, machine scheduling, and circuit routing. The input to such problems consists of a set of resources and a set of requests for these resources, and the goal is to find an allocation of resources that satisfies the requests. In many situations, while the overall set of resources (packing constraints) is known offline, the requests (covering constraints) arrive online and must be satisfied immediately on arrival. Further, resource allocations are typically impossible (or prohibitively expensive) to revoke. This gives rise to the OMPC problem.

Online Mixed Packing and Covering (OMPC). We are given an LP on a set of variables $\mathbf{x}, \lambda$ with packing constraints $\mathbf{Px} \leq \lambda\mathbf{p}$ given offline, and covering constraints $\mathbf{Cx} \geq \mathbf{c}$ that arrive online one at a time. All coefficients and variables are non-negative. On the arrival of a new covering constraint, we must increment the values of $\mathbf{x}, \lambda$ so that all constraints are satisfied. The goal is to minimize the final value of λ.

An increasingly popular approach for solving packing **or** covering problems online is the following: in each online step, we first update the variable values to obtain a feasible fractional solution, and then round the current fractional solution to obtain an integer solution (see, e.g., [1, 2, 8, 9]). Our OMPC algorithm solves the first step in this framework, i.e. obtains a fractional solution, for problems that can be expressed as a mixed packing and covering LP. Unlike in the offline case where LPs can be optimally solved in polynomial time, obtaining a fractional solution to an LP online often turns out to be a significant challenge. This has led to extensive research over the last few years in obtaining fractional solutions to online LPs. However, to the best of our knowledge, the proposed techniques apply only to LPs that either contain only packing constraints or only covering constraints, or minor variants of these. Therefore, an important open question was whether one could extend this framework of solving LPs online to mixed packing and covering LPs. We answer this question in the affirmative in this paper by giving the first algo-

rithm to obtain a polylogarithmic competitive ratio for the OMPC class of problems.

We also consider two natural problems that fall in this category.

Unrelated Machine Scheduling with Startup Costs (UMSC). We are given a set of m machines M with startup cost c_i for machine i. A set of jobs J arrives online, one at a time. Each job j requires p_{ij} time to be processed on machine i. When job j arrives, we must determine whether to open new machines by paying their startup cost, and then assign the job to one of the open machines. A *schedule* is an assignment of jobs to machines. Our objective is to obtain a schedule of minimum startup cost, subject to the constraint that the *makespan* — the maximum over machines of the sum of processing times of the jobs assigned to a machine — of the schedule is at most a target makespan **L**.

Note that the UMSC problem generalizes online set cover [2] (when all processing times are either 0 or ∞) and online unrelated machine scheduling [5] (when all costs are 0). In addition to its theoretical significance, one of the motivations for this problem comes from the need to minimize energy consumption in large data centers, such as those used by Google and Amazon (see [12, 29] for the practical significance of this problem). This problem has been studied earlier in both the offline [19, 29] and online [17, 27, 28] models. While near-optimal algorithms were known in the offline model, we give the first near-optimal online algorithm for this problem.

Capacity Constrained Facility Location (CCFL). We are given offline a set of facilities $\mathcal{F}$ with opening cost c_i and capacity u_i for each facility i in $\mathcal{F}$. Clients arrive online, and each client j in $\mathcal{C}$ has an assignment cost a_{ij} and a demand p_{ij} on being assigned to facility i. As each client arrives, we must determine whether to open new facilities by paying their opening cost, and then assign the client to an open facility. The congestion of a facility is the ratio of the sum of the demands of clients assigned to the facility, to the capacity of the facility. Our goal is to assign clients to facilities to minimize the sum of opening costs and assignment costs subject to the constraint that the congestion on any facility is at most 1.

This problem generalizes online (non-metric) facility location [1] (when all capacities are ∞) and online unrelated machine scheduling [5] (when all costs are 0 and all capacities are identical). Both the offline [35] and online [1, 20] versions of the uncapacitated facility location have been studied earlier. Capacitated facility location has been studied in the offline model both with soft capacities (allows multiple facilities at a location) and hard capacities (at most one facility per location) [37, 43].

Note that if we map facilities to machines and clients to jobs, then the UMSC problem is a special case of the CCFL problem where all assignment costs are 0 and all capacities are **L**.

1.1 Our Results.

We give polylogarithmic competitive ratios for the problems discussed. Our results are the first sublinear guarantees for these problems. We also give lower bounds to show that our results are nearly tight.

For OMPC, we show the following result in Section 2.

THEOREM 1.1. *There is a deterministic $O(\log m \log(d\rho\kappa))$-competitive algorithm for the OMPC problem, where m is the number of packing constraints, d is the maximum number of variables in any constraint, and ρ (resp., κ) is the ratio of the maximum to the minimum non-zero packing (resp., covering) coefficient respectively.*

If all coefficients are either 0 or 1, this yields an $O(\log m \log d)$-competitive algorithm. We also show a strong lower bound (details deferred to the full version of the paper) which implies that our algorithm is nearly optimal.

THEOREM 1.2. *There is a lower bound of $\Omega(\log m \log(d/\log m))$ on the competitive ratio on any deterministic algorithm for the OMPC problem.*

Next, we consider the UMSC and CCFL problems. We give a randomized online algorithm with a polylogarithmic bicriteria* competitive ratio for the CCFL problem, which implies an identical result for the UMSC problem as well (Section 3). Let α be the minimum sum of opening and assignment costs of a schedule with congestion 1.

THEOREM 1.3. *There is a randomized online algorithm for the CCFL problem that has a bicriteria competitive ratio of $(O(\log(mn)\log m), O(\log(m\alpha)\log m))$.*

If the cost of an optimal solution α is given in advance, then our algorithm is $(O(\log(mn)\log m), O(\log m))$-competitive. As noted earlier, UMSC (and therefore CCFL) generalizes online set cover and online unrelated machine scheduling, which respectively have lower

*An online randomized algorithm for the CCFL problem with a bicriteria competitive ratio of (α, β) produces a solution with congestion at most β and expected cost at most α times the cost of an optimal solution with congestion 1.

bounds of $\Omega(\log m \log n)$ (assuming BPP $\neq$ NP) [3] and $\Omega(\log m)$ [5]. Thus, our algorithm is nearly-optimal in its competitive ratio.

On the other hand, if the optimal cost α is not given, then we give a lower bound on the competitive ratio of any deterministic algorithm for UMSC, even for fractional solutions (details deferred to the full version of the paper). This lower bound on deterministic fractional solutions is relevant to our technique since our algorithm first produces a deterministic fractional solution, and then uses randomization to round it to an integer solution.

THEOREM 1.4. *No deterministic online algorithm for the UMSC problem can obtain a fractional solution with makespan $o(\log \alpha)\mathbf{L}$ and startup cost within a polylogarithmic factor of the optimal.*

Previous work in online machine scheduling also considered the objective of minimizing the *sum* of the makespan and startup costs [16, 17] when no target makespan is given. Our bicriteria algorithm for CCFL also yields an algorithm that minimizes the sum of makespan, startup/opening, and assignment costs, called the *total cost*. Let β^* be the total cost of the optimal schedule. We start with an estimate β to the optimal total cost, and run our algorithm for CCFL with all capacities and the optimal cost α set to β. Whenever the algorithm fails, i.e. α is too small, we double β. Since the algorithm is $(O(\log(mn)\log m), O(\log m))$-competitive, when $\beta = \beta^*$ the algorithm succeeds, giving an assignment of total cost $O(\log(mn)\log m)\beta^*$. The previous failures at most double the cost of the solution obtained.

THEOREM 1.5. *There is an $O(\log(mn)\log m)$-competitive randomized online algorithm for CCFL with the objective of minimizing the sum of opening costs, assignment costs, and makespan.*

Since this problem generalizes online set cover, the competitive ratio obtained is nearly optimal.

1.2 Our Techniques. Our key technical contribution in this paper is an extension of the multiplicative weight updates method to LPs with both packing and covering constraints. For these LPs, we replace the packing constraints with a potential function that is exponential in the maximum violation of any packing constraint. When a covering constraint arrives, we use multiplicative updates to increase the values of the variables to satisfy the covering constraint. The update to each variable is inversely proportional to the potential function. Our work is the first to use an exponential potential function to drive multiplicative updates that yield provably good competitive ratios for *online* algorithms.

For OMPC, we replace the packing constraints $\mathbf{Px} \leq \lambda$ by the potential function $\Phi := \log \sum_k e^{(\mathbf{Px})_k}$, where $(\mathbf{Px})_k$ is the value of the kth packing constraint. For any variable x_j, define rate_j as the rate of change of Φ with respect to x_j. In every update, the increment to x_j is inversely proportional to rate_j. Section 2 describes the complete algorithm. Our analysis follows the primal-dual schema, although we need to account for the nonlinearity of Φ. A key step in our analysis is to upper bound rate_j.

Multiplicative weight updates are used in [14] to obtain $O(\log n)$-competitive fractional solutions for covering LPs when the constraints arrive online. In [14], the cost is a simple linear function of the variables. The update to each variable is inversely proportional to the sensitivity of the cost function relative to the variable, given by the variable's coefficient in the cost function. In our problem, however, the cost is the maximum violation of any packing constraint. The cost function is thus nonlinear, and since its sensitivity relative to a variable changes, it is not apparent how to extend the techniques from [14]. The algorithm in [14] is also extended to the case when box constraints of the form $x_i \leq u_i$ are given offline. These are inviolable constraints that must be satisfied by any solution, and are dealt with in [14] by simply ceasing to increment variables when they reach their upper bound. The rest of the algorithm remains unchanged, though the analysis differs somewhat. In contrast, our packing constraints consist of multiple variables and our objective is to minimize the maximum violation of any packing constraint. Thus in our case, there is no natural value where we can cease incrementing variables.

A large body of work uses Lagrangean-relaxation techniques to obtain approximate algorithms for solving LPs offline, e.g., [39, 42]. In these papers, the constraints in the LP are replaced by an exponential penalty function. In each update, the update vector for the variables minimizes the change in the penalty function. In this sense, the updates in these offline algorithms are greedy. Since the constraints are available offline, this gives ϵ-approximate solutions. In our case, since covering constraints arrive online, greedy algorithms perform very poorly, and we must use different techniques. We use an exponential penalty function similar to offline algorithms. However, instead of a greedy strategy as used in [39, 42], we hedge our bets and increment all variables that appear in the covering constraint. The increment to each variable is inversely proportional to its contribution to the penalty function.

For CCFL, our algorithm has two steps, both of which are performed in any online round. We first obtain a fractional assignment of clients to facilities, and then use a randomized rounding scheme to obtain an integer solution. For the fractional solution, we design a somewhat more complicated potential function that combines the two objectives: for each facility i, it is linear in the opening cost c_i and exponential in the congestion l_i. Our algorithm creates a dynamic list of facilities in increasing order of its change in potential if the current job were assigned to it, and then uses multiplicative updates to assign larger fractions of the client to facilities that appear earlier in the order. The analysis for this algorithm involves proving a bound on the value of the cumulative potential function over all the facilities. We then adapt randomized rounding techniques used for offline machine scheduling with startup costs [29] and online set cover [15] to round the fractional solution and obtain an integral assignment of jobs to machines.

While a reduction exists from uncapacitated nonmetric facility location to set cover [31] (and this reduction can be made online), such a reduction from CCFL to UMSC seems more challenging. The addition of demands complicates things since we can no longer assign each client to an open facility of minimum assignment cost as in [31], but must consider the demand placed on the facility as well. Instead of obtaining such a reduction, we give an algorithm that works directly with assignment costs.

1.3 Related Work. Multiplicative updates are used in a wide variety of contexts. They are used in both offline approximation algorithms for packing and covering problems [11, 18, 21, 24, 25, 26, 30, 32, 33, 39, 41, 42], and online algorithms for problems with only packing or only covering constraints such as set cover [15], caching [9], paging [8], ad allocations [13], and network optimization [1, 2]. Multiplicative weight updates also have a long history in learning theory; these results are surveyed in [4].

The (offline) UMSC problem was first studied in [19, 29]. They give offline algorithms for the problem using different techniques. For the online problem with identical machines, [17, 28] give $O(1)$-competitive algorithms where the objective is the sum of the makespan and startup cost. These are extended to the case where machines have speed either 1 or s, with more general startup costs, by Imreh [27]. We give the first results for the online setting with unrelated machines.

Many variants of the machine scheduling (or *load balancing*) problem have been extensively studied. The best known of these is perhaps offline minimum makespan scheduling, for which [34] give a 2-approximation. In the online setting, Graham [22, 23] showed that the natural greedy heuristic achieves a competitive ratio of $2 - 1/m$ for m identical machines. The competitive ratio of this problem has been subsequently improved in a series of results (see, e.g., [10] and subsequent improvements). For the more general restricted assignment problem where the processing time of each job j on any machine is either some value p_j or ∞, an online algorithm having competitive ratio $O(\log m)$ was designed by Azar, Naor and Rom [7]. This algorithm was later generalized to the unrelated machines scenario by Aspnes *et al* [5] with the same competitive ratio. Various other models and objectives have been considered for the machine scheduling problem; for a comprehensive survey, see [6] and [40].

2 Algorithm for the OMPC Problem

We now give an algorithm for the OMPC problem and prove Theorem 1.1. A mixed packing and covering LP consists of packing constraints $\mathbf{Px} \leq \mathbf{1}$ and covering constraints $\mathbf{Cx} \geq \mathbf{1}$. Our problem is to obtain a solution $\mathbf{x}$ for (2.1) that minimizes λ, the maximum amount by which any packing constraint is violated.

$$\begin{aligned} (2.1) \qquad \min \quad & \lambda \\ \text{s.t.} \quad & \mathbf{Cx} \geq \mathbf{1}, \\ & \mathbf{Px} \leq \lambda, \\ & \mathbf{x}, \lambda \geq \mathbf{0}. \end{aligned}$$

Packing constraints are given to us initially, and covering constraints $\mathbf{Cx} \geq \mathbf{1}$ are revealed one at a time. Our algorithm assigns fractional values to the variables. The variable values can be increased, but not decreased.

For a vector $\mathbf{v}$, both v_i and $(v)_i$ denote its ith component. The set $\{1, 2, \ldots, n\}$ is denoted by $[n]$. The number of variables, packing constraints, and covering constraints in the LP are n, m, and m_c respectively. Define $\rho = \max_{k,j} p_{kj} / \min_{k,j:p_{kj}>0} p_{kj}$ and $\kappa = \max_{i,j} c_{ij} / \min_{i,j:c_{ij}>0} c_{ij}$. κ is used only in the analysis of the algorithm; we do not need to know its value during execution. Define $\kappa_1 = \max_j c_{1j}$, i.e., κ_1 is the maximum coefficient in the first covering constraint to arrive. We use d to denote the maximum number of variables in any constraint, and d_1 for the maximum number of variables in any packing constraint, and the first covering constraint. Define $\mu = 1 + \frac{1}{3\ln(em)}$, and $\sigma = e^2 \ln(\mu d^2 \rho \kappa)$ (e is the base of the natural logarithm.) OPT is the value of the optimal solution to (2.1). For our analysis, we consider the dual of (2.1):

$$\begin{aligned} (2.2) \qquad \max \quad & \sum_i y_i \\ \text{s.t.} \quad & \mathbf{C^T y} \;\le\; \mathbf{P^T z}\,, \\ & \sum_{k=1}^{m} z_k \;\le\; 1\,, \\ & \mathbf{y},\, \mathbf{z} \;\ge\; \mathbf{0}\,. \end{aligned}$$

We assume that $\frac{1}{4\sigma} \le \text{OPT} \le \frac{1}{2\sigma}$. This can be achieved by dividing the matrix of packing coefficients $\mathbf{P}$ by a parameter Γ that satisfies $2\text{OPT} \ge \frac{\Gamma}{2\sigma} \ge \text{OPT}$. Without scaling $\mathbf{P}$, our algorithm obtains an *additive* $O(\log m)$ term in the competitive ratio. Scaling the matrix $\mathbf{P}$ allows us to convert the additive term into a multiplicative term for the competitive ratio. The lower bound on Γ ensures $\frac{\text{OPT}}{\Gamma} \le \frac{1}{2\sigma}$, and is used to obtain meaningful bounds on the competitive ratio. Without this estimate Γ, we can use a "doubling procedure" commonly used in online algorithms, which increases the competitive ratio by a factor of 4. This doubling procedure is discussed in detail in Section A.

Given vector $\mathbf{x}$, let $\lambda(\mathbf{x}) = \max_{k\in[m]}(\mathbf{Px})_k$. Our goal is to obtain $\mathbf{x}$ that is feasible for (2.1) and minimizes $\lambda(\mathbf{x})$. Since $\lambda(\mathbf{x})$ is the maximum of the packing constraints, it is non-differentiable. We therefore define the potential function

$$\Phi(\mathbf{x}) \;:=\; \ln\left(\sum_{k\in[m]} \exp(\mathbf{Px})_k\right).$$

$\Phi(\mathbf{x})$ is a differentiable estimate of $\lambda(\mathbf{x})$, since $\max_k(\mathbf{Px})_k \le \Phi(\mathbf{x}) \le \max_k(\mathbf{Px})_k + \ln m$. In our algorithm, the multiplicative update to each variable x_j is inversely proportional to the rate of change of $\Phi(\mathbf{x})$ relative to x_j, given by

$$\begin{aligned} (2.3) \qquad \text{rate}_j(\mathbf{x}) &= \frac{\partial \Phi(\mathbf{x})}{\partial x_j} \\ &= \frac{\sum_{k\in[m]} p_{kj} \exp(\mathbf{Px})_k}{\sum_{k\in[m]} \exp(\mathbf{Px})_k}\,. \end{aligned}$$

For covering constraint i, define

$$(2.4) \qquad \epsilon_i(\mathbf{x}) = (\mu - 1) \min_{j:c_{ij}>0} \text{rate}_j(\mathbf{x})/c_{ij}\,,$$

so that for all $j \in [n]$, $\epsilon_i(\mathbf{x})c_{ij}/\text{rate}_j(\mathbf{x}) \le \mu - 1$.

Our algorithm is as follows. When the first covering constraint arrives, initialize $x_j \leftarrow 1/(d_1^2\rho\kappa_1)$ for all $j \in [n]$. Each covering constraint is assigned a new dual variable y_i when it arrives. For covering constraint i, while $(\mathbf{Cx})_i < 1$, we execute the following steps:

- Let $\mathbf{x}^l$ be the current value of $\mathbf{x}$. Increase each x_j to $x_j\left(1 + \epsilon_i(\mathbf{x}^l)\frac{c_{ij}}{\text{rate}_j(\mathbf{x}^l)}\right)$.
- Increment dual variable y_i by $e\epsilon_i(\mathbf{x}^l)$.

A single iteration of the while loop is a *phase*, indexed by l. The first phase is phase 0. The value of the variables before they are incremented in phase l is $\mathbf{x}^l$. $\mathbf{x}^0$ denotes the values after initialization. For covering constraint i, L_i is the set of indices of the phases executed from its arrival until $(\mathbf{Cx})_i \ge 1$, and $L = \cup_i L_i$.

Lemma 2.1 follows from the initialization for variables x_j.

LEMMA 2.1. *For the variables as initialized,* $\lambda(\mathbf{x}^0) \le \text{OPT}$*, and hence* $\Phi(\mathbf{x}^0) \le \text{OPT} + \ln m$.

Proof. Let x_j^* be the values for the variables in an optimal solution. After the first covering constraint is received, $1 \le \sum_j c_{1j}x_j^* \le \max_r c_{1r}\sum_j x_j^*$. Since the first covering constraint has at most d_1 variables, there exists variable $x_b^* \ge 1/(d_1 \max_r c_{1r})$, and hence

$$\begin{aligned} \text{OPT} \;=\; \max_{k\in[m]}(\mathbf{Px}^*)_k \;&\ge\; \min_{k,j:p_{kj}>0} p_{kj}x_b^* \\ &\ge\; \min_{k,j:p_{kj}>0} p_{kj}/(d_1 \max_r c_{1r}) \\ &=\; \min_{k,j:p_{kj}>0} p_{kj}/(d_1\kappa_1)\,. \end{aligned}$$

Using $\rho = \max_{k,j} p_{kj} / \min_{k,j:p_{kj}>0} p_{kj}$,

$$(2.5) \qquad \text{OPT} \;\ge\; \frac{\max_{k,j:p_{kj}>0} p_{kj}}{d_1\rho\kappa_1}\,.$$

Our algorithm initializes $x_j^0 = 1/(d_1^2\rho\kappa_1)$, and hence

$$\begin{aligned} (2.6) \qquad \lambda(\mathbf{x}^0) \;&=\; \max_{k\in[m]}(\mathbf{Px}^0)_k \;\le\; \frac{d_1 \max_{k,j} p_{kj}}{d_1^2\rho\kappa_1} \\ &=\; \frac{\max_{k,j} p_{kj}}{d_1\rho\kappa_1} \overset{(2.5)}{\le} \text{OPT}\,, \end{aligned}$$

where the first inequality is because any packing constraint has at most d_1 variables. Thus, $\Phi(\mathbf{x}^0) \le \lambda(\mathbf{x}^0) + \ln m \le \text{OPT} + \ln m$, proving the lemma.

COROLLARY 2.1. *In any phase* l, $\lambda(\mathbf{x}^l) \le 3\ln(em)$.

Proof. By (2.6), $\lambda(\mathbf{x}^0) \leq 1/(2\sigma) \leq 1$. Thus the lemma is satisfied for the first phase. For any phase $l > 0$, the algorithm would have failed at the end of phase $l-1$ if $\lambda(\mathbf{x}) \geq 3\ln(em)$. Since the algorithm did not fail in phase $l-1$, in any phase l, $\lambda(\mathbf{x}^l) \leq 3\ln(em)$.

Lemma 2.1 thus bounds the initial value of the primal objective. Our proof of the competitive ratio follows from a primal-dual analysis. We show in Lemma 2.2 that the dual objective is an upper bound on the *increase* in the primal objective. Corollary 2.2 and Lemma 2.5 show how the dual variables maintained by the algorithm can be scaled down to obtain feasible dual values. We use these results together with weak duality in Theorem 2.1 to prove the bound on the competitive ratio.

Lemma 2.2 follows from our choice of multiplicative updates.

LEMMA 2.2. *The increase in $\sum_i y_i$ is an upper bound on the increase in $\Phi(\mathbf{x})$ in every phase.*

To prove this lemma, we will use the following technical lemma.

LEMMA 2.3. *Given $\mathbf{x}'$ and $\mathbf{x}''$ with $\lambda(\mathbf{x}') \leq 3\ln(em)$ and $x'_j \leq x''_j \leq \mu x'_j$ where $\mu = 1 + \frac{1}{3\ln(em)}$,*

$$\text{rate}_j(\mathbf{x}'') \leq e\,\text{rate}_j(\mathbf{x}').$$

Proof. By definition of $\text{rate}_j(\mathbf{x})$ in (2.3), and since $x'_j \leq x''_j \leq \mu x'_j$,

$$\begin{aligned}(2.7)\qquad \text{rate}_j(\mathbf{x}'') &= \frac{\sum_{k\in[m]} p_{kj}\exp(\mathbf{P}\mathbf{x}'')_k}{\sum_{k\in[m]}\exp(\mathbf{P}\mathbf{x}'')_k} \\ &\leq \frac{\sum_{k\in[m]} p_{kj}\exp(\mu(\mathbf{P}\mathbf{x}')_k)}{\sum_{k\in[m]}\exp(\mathbf{P}\mathbf{x}')_k}.\end{aligned}$$

Since $\lambda(\mathbf{x}') \leq 3\ln(em)$, $\forall k$, $(\mathbf{P}\mathbf{x}')_k \leq 3\ln(em)$, and hence $\mu(\mathbf{P}\mathbf{x}')_k = (\mathbf{P}\mathbf{x}')_k + (\mathbf{P}\mathbf{x}')_k/(3\ln(em)) \leq (\mathbf{P}\mathbf{x}')_k + 1$. Substituting $(\mathbf{P}\mathbf{x}')_k + 1$ for $\mu(\mathbf{P}\mathbf{x}')_k$ in (2.7) yields

$$\begin{aligned}\text{rate}_j(\mathbf{x}'') &\leq \frac{\sum_{k\in[m]} p_{kj}\exp((\mathbf{P}\mathbf{x}')_k + 1)}{\sum_{k\in[m]}\exp(\mathbf{P}\mathbf{x}')_k} \\ &= e\,\frac{\sum_{k\in[m]} p_{kj}\exp(\mathbf{P}\mathbf{x}')_k}{\sum_{k\in[m]}\exp(\mathbf{P}\mathbf{x}')_k} \\ &= e\ \text{rate}_j(\mathbf{x}'),\end{aligned}$$

proving the lemma.

We are now ready to prove Lemma 2.2.

Proof of Lemma 2.2. Let Φ^l and Φ^{l+1} denote the values of $\Phi(\mathbf{x})$ before and after the variables are incremented in phase l, respectively. We will show that $\Phi^{l+1} - \Phi^l \leq e\epsilon_i(\mathbf{x}^l)$, which is the increase in $\sum_i y_i$ in phase l.

Let $\mathbf{x}^l$ and $\mathbf{x}^{l+1}$ be the values of $\mathbf{x}$ before and after being incremented in phase l. For each x_j, let $g_j(t) := x_j^l + (x_j^{l+1} - x_j^l)t$ for $0 \leq t \leq 1$. Note that $g_j(0) = x_j^l$ and $g_j(1) = x_j^{l+1}$. Define $\mathbf{g}(t) = (g_1(t), g_2(t), \ldots, g_m(t))$. With some abuse of notation, any function of $\mathbf{x}$, say $h(\mathbf{x})$, can be viewed as a function of t, with $h(t) := h(\mathbf{g}(t))$. Thus, the functions $\Phi(\mathbf{x})$ and $\text{rate}_j(\mathbf{x})$ can be written as functions of t: $\Phi(t) = \ln\sum_{k\in[m]}\exp(\mathbf{P}\mathbf{g}(t))_k$, and

$$\begin{aligned}\text{rate}_j(t) = \text{rate}_j(\mathbf{g}(t)) &= \frac{\partial\Phi(t)}{\partial g_j(t)} \\ &= \frac{\sum_{k\in[m]}\tilde{p}_{kj}\exp(\mathbf{P}\mathbf{g}(t))_k}{\exp(\Phi(t))}.\end{aligned}$$

We use these alternate expressions in the remainder of the proof. By the chain rule,

$$\frac{d\Phi(t)}{dt} = \sum_{j=1}^{n}\frac{\partial\Phi(t)}{\partial g_j(t)}\frac{dg_j(t)}{dt},$$

and hence,

$$\begin{aligned}(2.8)\qquad \Phi^{l+1} - \Phi^l &= \int_{t=0}^{1}\frac{d\Phi(t)}{dt}dt \\ &= \sum_{j=1}^{n}\int_{t=0}^{1}\text{rate}_j(t)\frac{dg_j(t)}{dt}dt.\end{aligned}$$

In a phase, each variable is incremented by at most a factor of μ. Therefore $\mathbf{x}^{l+1} \leq \mu\mathbf{x}^l$. Then by Corollary 2.1, $\lambda(\mathbf{x}^l) \leq 3\ln(em)$ in any phase l. Thus $\text{rate}_j(t) \leq e\,\text{rate}_j(0)$ for $0 \leq t \leq 1$ by Lemma 2.3. Hence

$$\begin{aligned}\Phi^{l+1} - \Phi^l &\leq e\sum_{j=1}^{n}\text{rate}_j(\mathbf{x}^l)\int_{t=0}^{1}\frac{dg_j(t)}{dt}dt \\ &= e\sum_{j=1}^{n}\text{rate}_j(\mathbf{x}^l)(x_j^{l+1} - x_j^l).\end{aligned}$$

Since in phase l each variable x_j gets multiplied by $1 + \epsilon_i(\mathbf{x}^l)\frac{c_{ij}}{\text{rate}_j(\mathbf{x}^l)}$,

$$\begin{aligned}\Phi^{l+1} - \Phi^l &\leq e\epsilon_i(\mathbf{x}^l)\sum_{j=1}^{n}\text{rate}_j(\mathbf{x}^l)\frac{c_{ij}x_j^l}{\text{rate}_j(\mathbf{x}^l)}\\ &= e\epsilon_i(\mathbf{x}^l)\sum_{j=1}^{n}c_{ij}x_j^l\\ &\leq e\epsilon_i(\mathbf{x}^l)\end{aligned}$$

where the last inequality follows since, on entering the for loop, $(\mathbf{Cx})_i < 1$. Since $e\epsilon_i(\mathbf{x}^l)$ is the increase in the dual objective, this proves the lemma.

The following results obtain bounds on the infeasibility of the constraints in the dual linear program (2.2).

LEMMA 2.4. *For any $j \in [n]$,*

$$(\mathbf{C^T y})_j \leq \sigma \max_{l\in L}\text{rate}_j(\mathbf{x}^l)\,.$$

Proof. In any phase l, x_j is multiplied by $1 + c_{ij}\epsilon_i(\mathbf{x}^l)/\text{rate}_j(\mathbf{x}^l) \geq \exp\left(\frac{c_{ij}\epsilon_i(\mathbf{x}^l)}{e\,\text{rate}_j(\mathbf{x}^l)}\right)$, where the inequality uses $1 + a \geq \exp(a/e)$ for $0 \leq a \leq 1$. The initial value of x_j is $1/(d_1^2\rho\kappa_1)$, and finally $x_j \leq \mu/\min_{i:c_{ij}>0} c_{ij}$. Thus over all phases,

$$\sum_{i\in[m_c]}\sum_{l\in L_i} c_{ij}\epsilon_i(\mathbf{x}^l) \leq e\ln\left(\frac{\mu d_1^2\rho\kappa_1}{\min_{i:c_{ij}>0} c_{ij}}\right)\times\max_{l\in L}\text{rate}_j(\mathbf{x}^l)\,. \tag{2.9}$$

From the algorithm, the dual variable y_i is exactly $e\sum_{l\in L_i}\epsilon_i(\mathbf{x}^l)$. Hence the expression on the left in (2.9) is $(\mathbf{C^T y})_j/e$. Since $\sigma \geq e^2\ln\frac{\mu d_1^2\rho\kappa_1}{\min_{i:c_{ij}>0} c_{ij}}$, the lemma follows.

From Lemma 2.4, if we set dual variables z_k so that $\sum_{k\in[m]} p_{kj}z_k$ is at least $\max_{l\in L}\text{rate}_j(\mathbf{x}^l)$, then the dual constraint $(\mathbf{C^T y})_j \leq (\mathbf{P^T z})_j$ is violated by a factor of at most σ. By definition of $\text{rate}_j(\mathbf{x})$,

$$\begin{aligned}\max_{l\in L}\text{rate}_j(\mathbf{x}^l) &= \max_{l\in L}\frac{\sum_{k\in[m]} p_{kj}\exp(\mathbf{Px}^l)_k}{\sum_{k\in[m]}\exp(\mathbf{Px}^l)_k}\\ &\leq \sum_{k\in[m]} p_{kj}\max_{l\in L}\frac{\exp(\mathbf{Px}^l)_k}{\sum_{k\in[m]}\exp(\mathbf{Px}^l)_k}\,.\end{aligned}$$

We now define our dual variables z_k as

$$z_k := \max_{l\in L}\frac{\exp(\mathbf{Px}^l)_k}{\sum_{k\in[m]}\exp(\mathbf{Px}^l)_k}\,.$$

Corollary 2.2 follows from Lemma 2.4 and by definition of z_k.

COROLLARY 2.2. *For any $j \in [n]$,*

$$(\mathbf{C^T y})_j \leq \sigma(\mathbf{P^T z})_j\,.$$

LEMMA 2.5. $\sum_{k\in[m]} z_k \leq \ln(em) + \max_{l\in L}\lambda(\mathbf{x}^l)$.

Proof. For each packing constraint k, define

$$\phi(k) := \arg\max_l \frac{\exp((\mathbf{Px}^l)_k)}{\sum_{k\in[m]}\exp(\mathbf{Px}^l)_k}\,.$$

Thus z_k attains its value in phase $\phi(k)$. We index the packing constraints so that if $k < k'$, then $\phi(k) < \phi(k')$. To prove Lemma 2.5 we make use of the following bound on the sum of a sequence that we prove in Section A.

LEMMA 2.6. *For $m \in \mathbb{Z}_+$ and $a_1, a_2, \ldots, a_m \in \mathbb{R}_{\geq 0}$ with $a_1 > 0$, $\sum_{i\in[m]}\frac{a_i}{\sum_{j\leq i} a_j} \leq 1 + \ln\frac{\sum_{i=1}^m a_i}{a_1}$.*

Let $a_k := \exp((\mathbf{Px}^{\phi(k)})_k)$. We claim that $z_k \leq a_k/\sum_{l\leq k} a_l$. Given this claim, Lemma 2.5 follows from Lemma 2.6 since $a_1 = \exp((\mathbf{Px}^{\phi(1)})_1) \geq 1$ and hence

$$\begin{aligned}1 + \ln\frac{\sum_{i\in[m]} a_i}{a_1} &\leq 1 + \ln\sum_{i\in[m]} a_i\\ &\leq 1 + \ln m + (\mathbf{Px}^{\phi(k)})_k\\ &\leq \ln(em) + \max_l \lambda(\mathbf{x}^l)\,.\end{aligned}$$

To show that $z_k \leq a_k/\sum_{l\leq k} a_l$, observe that the denominator of z_k is given by $\sum_{l\in[m]}\exp((\mathbf{Px}^{\phi(k)})_l) \geq \sum_{l\leq k}\exp((\mathbf{Px}^{\phi(k)})_l)$. Fix k. For each $l < k$, $\phi(l) < \phi(k)$. Since variables are non-decreasing, $\exp((\mathbf{Px}^{\phi(l)})_l) \leq \exp((\mathbf{Px}^{\phi(k)})_l)$. Hence,

$$\sum_{l\in[m]}\exp((\mathbf{Px}^{\phi(k)})_l) \geq \sum_{l\leq k}\exp((\mathbf{Px}^{\phi(l)})_l) = \sum_{l\leq k} a_l\,.$$

The denominator of z_k is thus at least $\sum_{l\leq k} a_l$. Since the numerator of z_k is a_k, the claim follows.

We now use the previous lemmas to prove the bound on the competitive ratio of our algorithm.

THEOREM 2.1. *The algorithm is* $8\sigma \ln(em)$*-competitive.*

Proof. The dual objective $\sum_i y_i$ is an upper bound on the total increase in $\Phi(\mathbf{x})$ in each phase, and initially, $\Phi(\mathbf{x}^0) \leq \ln m + \text{OPT}$. Since $\Phi(\mathbf{x})$ is an upper bound on $\lambda(\mathbf{x})$, $\lambda(\mathbf{x}) \leq \sum_i y_i + \ln m + \text{OPT}$. Define $\nu := \ln(em) + \max_l \lambda(\mathbf{x}^l)$. The dual is a maximization problem, and Corollary 2.2 and Lemma 2.5 show that $\mathbf{z}/\nu$ and $\mathbf{y}/(\sigma\nu)$ are feasible values for the dual variables. By weak duality, any feasible dual solution is a lower bound on the optimal primal solution, and hence $\sum_i y_i/(\sigma\nu) \leq \text{OPT}$. For $\lambda(\mathbf{x})$, we thus obtain

$$\lambda(\mathbf{x}) \;\leq\; \sigma\nu OPT + OPT + \ln m\,. \tag{2.10}$$

We first show that $\lambda(\mathbf{x}) \leq 3\ln(em)$, and then obtain the bound on the competitive ratio. Since $\text{OPT} \leq \frac{1}{2\sigma}$, substituting in (2.10),

$$\lambda(\mathbf{x}) \;\leq\; \frac{\nu}{2} + \frac{1}{2\sigma} + \ln m\,,$$

and substituting $\nu = \ln(em) + \lambda(\mathbf{x})$ yields $\lambda(\mathbf{x}) \leq 3\ln(em)$.

For the competitive ratio, since $\lambda(\mathbf{x}) \leq 3\ln(em)$, hence $\nu = \ln(em) + \lambda(\mathbf{x}) \leq 4\ln(em)$. Further, $\ln m \leq 4\sigma \ln m\text{OPT}$. Substituting in (2.10) yields

$$\begin{aligned}\lambda(\mathbf{x}) \;&\leq\; 4\sigma\ln(em)\text{OPT} + \text{OPT} + 4\sigma\ln m\text{OPT}\\ &\leq\; 8\sigma\ln(em)OPT\,.\end{aligned}$$

3 Algorithm for the CCFL Problem

In this section, we give an algorithm for the CCFL problem and prove Theorem 1.3. In describing the algorithm, we assume that the number of clients n and the optimal cost α are given offline. If n is not known offline, each client estimates n by assuming that it is the last client; this incurs *additive* factors of $O(\log\log n)$ in the makespan and $O(\log n \log(mn))$ in the cost. If α is not known offline, then we can guess the value of α using standard techniques; this incurs *multiplicative* factors of $O(\log(m\alpha))$ in the makespan and $O(1)$ in the cost. We also assume that all facilities have capacity 1, which can be achieved by scaling each demand by the capacity of the corresponding facility.

As described earlier, our algorithm produces a fractional solution which is then rounded online to obtain an integer solution.

Minimize $\sum_{i\in\mathcal{F}} c_i x_i \;+\; \sum_{j\in\mathcal{C}}\sum_{i\in\mathcal{F}} a_{ij}y_{ij}$ subject to:

$$\sum_{j\in\mathcal{C}} p_{ij}y_{ij} \;\leq\; x_i \quad \forall i \in \mathcal{F} \tag{3.11}$$

$$y_{ij} \;\leq\; x_i \quad \forall\, i \in \mathcal{F},\ j \in \mathcal{C} \tag{3.12}$$

$$\sum_{i\in\mathcal{F}} y_{ij} \;\geq\; 1 \quad \forall\, j \in \mathcal{C} \tag{3.13}$$

$$x_i, y_{ij} \;\in\; \{0,1\} \quad \forall i \in \mathcal{F},\ j \in \mathcal{C} \tag{3.14}$$

Figure 1: The Integer Scheduling LP (ISLP) for the CCFL problem

3.1 Fractional Algorithm.

An integer LP formulation of CCFL (called *integer scheduling LP* or ISLP) is given in Figure 1. The variable x_i is 1 if and only if facility i is open, and y_{ij} is 1 if and only if client j is assigned to facility i. In the fractional relaxation (FSLP), these variables are constrained to be in the range $[0,1]$ instead of Eqn. (3.14). Note that we have both covering (3.13) and packing (3.11) constraints in these relaxations.

We will now describe the online updates to the fractional solution to maintain feasibility for FSLP on receiving a new client j. This involves updating the values of y_{ij} (the fraction of client j assigned to facility i) so as to satisfy (3.13), and corresponding updates to the values of x_i if (3.11) or (3.12) is violated. In fact, we relax the constraints in FSLP in two ways. Let facility 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. First, for technical reasons, we relax (3.11) and (3.12) to

$$\sum_{j\in\mathcal{C}} p_{ij}y_{ij} \;\leq\; 9x_i \quad \forall\, i \in \mathcal{F} \tag{3.15}$$

$$y_{ij} \;\leq\; 2x_i \quad \forall\, i \in \mathcal{F},\ j \in \mathcal{C} \tag{3.16}$$

Further, we enforce (3.15) only if $x_i < 1$, i.e. if facility i is not fully open. The congestion on a fully open facility will be bounded separately in the analysis. We call this the *relaxed fractional scheduling LP* or RFSLP.

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

Pre-processing. First, we discard all facilities with opening cost greater than α (recall that α is the optimal cost) and redefine m to be the number of remaining facilities. Next, we multiply all costs (opening and assignment) by m/α so that the new optimum is m. Further, for every facility i with $c_i \leq 1$, we increase c_i to 1. Finally, we initialize x_i for all facilities i to $1/m$. At the end of the pre-processing phase, we have the

following properties:

- The optimal cost is between m and $2m$.
- The opening cost of every facility is at least 1 and at most m.
- Every facility i has $x_i = 1/m$.
- $\sum_{i\in\mathcal{F}} x_i = 1$.

The Algorithm. To describe the online updates, we need the following definitions. We define the *virtual cost* of client j on facility i as

$$\eta_i(j) = \begin{cases} c_i A^{\ell_i - 1} p_{ij} + a_{ij}, & \text{if facility } i \text{ is fully open,} \\ & \text{i.e., } x_i = 1 \\ c_i p_{ij} + a_{ij}, & \text{otherwise} \end{cases}$$

where A is a constant that we will fix later, and ℓ_i represents the congestion on facility i, i.e.

$$\ell_i = \sum_{j\in\mathcal{C}} p_{ij} y_{ij}.$$

Let $M(j)$ denote an ordering of facilities in non-decreasing order of virtual cost $\eta_i(j)$ for client j. Let $P(j)$ denote the maximal prefix of $M(j)$ such that $\sum_{i\in P(j)} x_i < 1$. (Note that $P(j)$ may be empty.) Let $k(j)$ denote the first facility in $M(j)$ that is not in $P(j)$. (Since $\sum_{i\in\mathcal{F}} x_i = 1$ after the pre-processing phase and the values of x_i's are non-decreasing during the course of this algorithm, $P(j) \neq M(j)$ and therefore, $k(j)$ is well-defined.)

We now describe the online updates. Suppose client j arrives online. We increase the values x_i's and y_{ij}'s in multiple iterations until $\sum_{i\in\mathcal{F}} y_{ij} \geq 1$, i.e. Eqn. (3.13) is satisfied. To describe a single iteration, let us first set up some notation. Let Δx_i be the total increase in the value of x_i over all iterations for client j. We say that the *effective capacity* created on facility i for client j is

$$\min\left(2x_i, \frac{9\ \Delta x_i}{p_{ij} + (a_{ij}/c_i))}\right).$$

Note that the effective capacity created by the increase in x_i gives a value of y_{ij} that preserves the feasibility of the fractional solution for RFSLP.

Now, we are ready to define a single iteration (called an *algorithmic step*) of the updates. Let $\delta x_i = x_i/c_i n$ for facility i. We increase x_i (and correspondingly Δx_i) by δx_i for each facility $i \in P(j)$.[†] Further, for each facility $i \in P(j)$, we set the value of y_{ij} to the effective capacity created on facility i for client j. For facility $k(j)$, we have two cases:

- $x_{k(j)} < 1$ **(i.e. facility $k(j)$ is partially open).** We increase $x_{k(j)}$ (and correspondingly $\Delta x_{k(j)}$) by $\delta x_{k(j)}$ for facility $k(j)$; further, we set the value of $y_{k(j)j}$ to the effective capacity created on facility $k(j)$ for client j. We call this an algorithmic step of **type A**.
- $x_{k(j)} = 1$ **(i.e. facility $k(j)$ is fully open).** We keep the value of $x_{k(j)}$ unchanged at 1 but *increase* $y_{k(j)j}$ by $9/\eta_{k(j)}(j)n$. We call this an algorithmic step of **type B**.

As noted earlier, the above algorithmic steps are repeatedly undertaken until $\sum_{i\in\mathcal{F}} y_{ij} \geq 1$.

Analysis. Our goal is to show the following bounds on the maximum congestion and cost of the fractional assignment.

LEMMA 3.1. *The fractional assignment produced by the online algorithm satisfies*

$$\sum_{j\in\mathcal{C}} y_{ij} p_{ij} = O(\log m)$$

for each facility i, and

$$\sum_{i\in\mathcal{F}} c_i x_i + \sum_{j\in\mathcal{C}} \sum_{i\in\mathcal{F}} a_{ij} y_{ij} = O(m \log m).$$

We introduce a potential function ϕ_i for facility i defined as

$$\phi_i = \begin{cases} c_i A^{\ell_i - 1} + \frac{1}{9} \sum_{j\in\mathcal{C}} a_{ij} y_{ij}, & \text{if facility } i \text{ is fully open,} \\ & \text{i.e., } x_i = 1 \\ c_i x_i + \frac{1}{9} \sum_{j\in\mathcal{C}} a_{ij} y_{ij}, & \text{otherwise.} \end{cases}$$

The cumulative potential function $\phi = \sum_{i\in\mathcal{F}} \phi_i$. Our goal will be to show that $\phi = O(m \log m)$. Note that the potential function ϕ_i may not be monotonically increasing at $x_i = 1$; however, the following fact ensures that proving a bound on the final value of ϕ implies Lemma 3.1.

FACT 3.1. *For any facility i, let the final value of the potential function ϕ_i be T. Then, the total increase in the value of ϕ_i during the course of the algorithm is $O(T)$.*

Proof. If facility i is partially open at the end, then the total increase in the value of ϕ_i is at most T since ϕ_i is non-decreasing. On the other hand, if facility i is fully open at the end, then the decrease in the value of ϕ_i

[†] This introduces a minor technical issue since the value of x_i can exceed 1, but this can be easily dealt with by capping the value of x_i at 1. We will ignore this issue to avoid unnecessary notational complexity.

when facility i is fully opened (i.e. when x_i reaches 1) is at most

$$\left(1 - \frac{1}{A}\right) c_i \leq (A-1)T,$$

since $T \geq c_i/A$. Hence, the total increase in the value of ϕ_i is at most $A \cdot T$.

We prove the bound on ϕ in three steps. First, we bound the increase of ϕ in the pre-processing phase (Lemma 3.2); next, we bound the increase of ϕ in each algorithmic step of either type A or type B (Lemma 3.3); and finally, we bound the total number of algorithmic steps.

Pre-processing

LEMMA 3.2. *At the end of the pre-processing phase, $\phi \leq m$.*

Proof. Note that each facility i has $c_i \leq m$ and $x_i = 1/m$ after pre-processing.

Single Algorithmic Step

LEMMA 3.3. *For any constant $1 < A < 19/18$, the increase in potential in a single algorithmic step of either type A or type B is at most $4/n$.*

We prove Lemma 3.3 by considering algorithmic steps of type A and type B separately.

LEMMA 3.4. *The increase in potential in a single algorithmic step of type A is at most $4/n$.*

Proof. The total increase in potential in an algorithmic step of type A (due to increase in the value of x_i for facilities $i \in P(j) \cup \{k(j)\}$) is at most

$$\begin{aligned}
&\sum_{i \in P(j) \cup \{k(j)\}} \left(c_i \ \delta x_i + \frac{a_{ij}}{9} \left(\frac{9 \ \delta x_i}{p_{ij} + a_{ij}/c_i} \right) \right) \\
&\leq \sum_{i \in P(j) \cup \{k(j)\}} \left(c_i + \frac{a_{ij}}{a_{ij}/c_i} \right) \left(\frac{x_i}{c_i n} \right) \\
&= \sum_{i \in P(j) \cup \{k(j)\}} \frac{2x_i}{n} \\
&< 4/n\,.
\end{aligned}$$

In the first inequality, we drop the term p_{ij} from the denominator and replace $\delta x_i = x_i/c_i n$.

LEMMA 3.5. *For any constant $1 < A < 19/18$, the increase in potential in a single algorithmic step of type B is at most $3/n$.*

Proof. The total increase in potential for facilities $i \in P(j)$ due to an algorithmic step of type B is at most

$$\begin{aligned}
&\sum_{i \in P(j)} \left(c_i \ \delta x_i + \frac{a_{ij}}{9} \left(\frac{9 \ \delta x_i}{p_{ij} + a_{ij}/c_i} \right) \right) \\
&\leq \sum_{i \in P(j)\}} \left(c_i + \frac{a_{ij}}{a_{ij}/c_i} \right) \left(\frac{x_i}{c_i n} \right) \\
&= \sum_{i \in P(j)\}} \frac{2x_i}{n} \\
&< 2/n.
\end{aligned}$$

In the first inequality, we drop the term p_{ij} from the denominator and replace $\delta x_i = x_i/c_i n$.

The other source of increase in potential is the assignment of a fraction of client j to facility $k(j)$. Let

$$B = \eta_{k(j)}(j)/p_{k_{(j)}j} = (c_{k(j)} A^{\ell_{k(j)}-1}) + (a_{k(j)j}/p_{k(j)j}).$$

Then, the congestion on facility $k(j)$ increases by $\delta\ell = 5/Bn$. The resulting increase of potential $\phi_{k(j)}$ is

$$\begin{aligned}
&c_{k(j)}(A^{\ell_{k(j)}-1+\delta\ell} - A^{\ell_{k(j)}-1}) + \frac{a_{k(j)j}}{9} \left(\frac{\delta\ell}{p_{k(j)j}} \right) \\
&= c_{k(j)} A^{\ell_{k(j)}-1}(A^{\delta\ell} - 1) + \delta\ell \left(\frac{a_{k(j)j}}{9 p_{k(j)j}} \right) \\
&= c_{k(j)} A^{\ell_{k(j)}-1} \left((1 + (A-1))^{9/Bn} - 1 \right) \\
&\qquad + \left(\frac{9}{Bn} \right) \left(\frac{a_{k(j)j}}{9 p_{k(j)j}} \right) \\
&< c_{k(j)} A^{\ell_{k(j)}-1} \left(\frac{18(A-1)}{Bn} \right) + \left(\frac{1}{Bn} \right) \left(\frac{a_{k(j)j}}{p_{k(j)j}} \right) \\
&< \left(c_{k(j)} A^{\ell_{k(j)}-1} + a_{k(j)j}/p_{k(j)j} \right) \left(\frac{1}{Bn} \right) \\
&= \frac{1}{n}.
\end{aligned}$$

The penultimate inequality follows from the property that $(1+x)^{1/y} < e^{x/y} < 1 + 2x/y$, for any $y \geq x > 0$.

This completes the proof of Lemma 3.3.

Number of Algorithmic Steps

We classify the algorithmic steps according to a fixed optimal offline (integer) assignment that we call OPT. Suppose OPT assigns client j to facility $\text{OPT}(j)$, and let M_{OPT} denote the facilities that are fully open in the optimal offline schedule. The three categories are:

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

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

We now bound the total increase in potential due to algorithmic steps in each of the three categories above. (Lemma 3.6 for category 1, Lemma 3.8 for category 2, and Lemma 3.9 for category 3.)

LEMMA 3.6. *The total increase in potential due to all algorithmic steps in the first category is $O(m \log m)$.*

Proof. In any algorithmic 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 each facility i in the pre-processing phase, there are at most $\sum_{i \in M_{\text{OPT}}} c_i n \log m = O(mn \log m)$ algorithmic steps of the first category. The lemma now follows from Lemma 3.3.

For the second and third categories, we need the following property.

LEMMA 3.7. *The total number of algorithmic steps (of either type A or type B) in the second and third categories for a client j is at most $2\eta_{\text{OPT}(j)}(j)n/9$.*

Proof. We want to show that after at most $2\eta_{\text{OPT}(j)}(j)n/9$ algorithmic steps in the second or third category for client j, we will have $\sum_{i \in \mathcal{F}} y_{ij} \geq 1$. To show this, we will use the fact that

$$\eta_i(j) \leq \eta_{\text{OPT}(j)}(j) \tag{3.17}$$

for every facility $i \in P(j) \cup \{k(j)\}$.

First, note that each algorithmic step of type B creates an effective capacity of $9/\eta_{k(j)}(j)n$ in facility $k(j)$. By Eqn. (3.17), the total effective capacity created by each such algorithmic step is at least $9/\eta_{\text{OPT}(j)}(j)n$.

Now, we will show that for any algorithmic step of type A, either the total effective capacity created is at least $9/\eta_{k(j)}(j)n$ *or* it is the last algorithmic step for client j. Let $R(j)$ be the set of facilities $i \in P(j) \cup \{k(j)\}$ such that

$$2x_i < \frac{9\ \Delta x_i}{p_{ij} + (a_{ij}/c_i)}.$$

If $\sum_{i \in R(j)} x_i \geq 1/2$, then the total effective capacity created for client j is at least

$$\sum_{i \in R(j)} 2x_i \geq 1,$$

and therefore, the current step is the last algorithmic step for client j. Otherwise, the total effective capacity created by the current algorithmic step on facilities $i \in (P(j) \cup \{k(j)\}) \setminus R(j)$ is

$$\begin{aligned}
& \sum_{i \in (P(j) \cup \{k(j)\}) \setminus R(j)} \frac{9\ \delta x_i}{p_{ij} + (a_{ij}/c_i)} \\
= & \sum_{i \in (P(j) \cup \{k(j)\}) \setminus R(j)} \frac{9x_i}{(c_i p_{ij} + a_{ij})n} \\
= & \sum_{i \in (P(j) \cup \{k(j)\}) \setminus R(j)} \frac{9x_i}{\eta_i(j)n} \quad \text{(type A)} \\
\geq & \sum_{i \in (P(j) \cup \{k(j)\}) \setminus R(j)} \frac{9x_i}{\eta_{\text{OPT}(j)}(j)n} \quad \text{(Eqn. (3.17))} \\
= & \frac{9}{(\eta_{\text{OPT}(j)}(j)n} \sum_{i \in (P(j) \cup \{k(j)\}) \setminus R(j)} x_i \\
> & \frac{9}{2\eta_{\text{OPT}(j)}(j)n}.
\end{aligned}$$

The last inequality follows from the fact that $\sum_{i \in (P(j) \cup \{k(j)\} \setminus R(j)} x_i > 1/2$ when $\sum_{i \in R(j)} x_i < 1/2$ since $\sum_{i \in P(j) \cup \{k(j)\}} x_i \geq 1$.

LEMMA 3.8. *The total increase in potential due to all algorithmic steps in the second category is $O(m)$.*

Proof. Note that for algorithmic steps in the second category for client j, the virtual cost

$$\eta_{\text{OPT}(j)}(j) = c_{\text{OPT}(j)} p_{\text{OPT}(j)j} + a_{\text{OPT}(j)j}.$$

By Lemmas 3.3 and 3.7, the total increase in potential due to algorithmic steps in the second category for client j is at most

$$\begin{aligned}
\frac{4}{n} \times \frac{2\eta_{\text{OPT}(j)}(j)n}{9} &= \frac{8\eta_{\text{OPT}(j)}(j)}{9} \\
&= \frac{8(c_{\text{OPT}(j)} p_{\text{OPT}(j)j} + a_{\text{OPT}(j)j})}{9}.
\end{aligned}$$

To complete the proof, note that

$$\sum_{j \in \mathcal{C}} c_{\text{OPT}(j)} p_{ij} + a_{\text{OPT}(j)j} = O(m). \tag{3.18}$$

LEMMA 3.9. *The total increase in potential due to all algorithmic steps in the third category is at most $m + (8/9)\sum_{i \in \mathcal{F}_A} c_i a^{L_i - 1}$, where L_i is the final congestion on facility i in the assignment produced by the algorithm and $\mathcal{F}_A$ is the set of facilities that are fully opened by the algorithm.*

Proof. Since the congestion of facilities is monotonically non-decreasing,

$$\eta_{\text{OPT}(j)j} \leq c_{\text{OPT}(j)} A^{L_{\text{OPT}(j)} - 1} p_{\text{OPT}(j)j} + a_{\text{OPT}(j)j}.$$

for every algorithmic step in the third category. Therefore, summing over all clients, the total increase in potential due to algorithmic steps in the third category is at most (using Lemmas 3.3 and 3.7)

$$\begin{aligned}
& \frac{8}{9}\sum_{j\in\mathcal{C}}\left(c_{\mathrm{OPT}(j)}A^{L_{\mathrm{OPT}(j)}-1}p_{\mathrm{OPT}(j)j}+a_{\mathrm{OPT}(j)j}\right)\\
< \quad & m+\frac{8}{9}\sum_{i\in M_{\mathrm{OPT}}\cap\mathcal{F}_A} c_iA^{L_i-1}\sum_{j:\mathrm{OPT}(j)=i} p_{ij}\\
& (\text{since } \sum_{j\in\mathcal{C}} a_{\mathrm{OPT}(j)j}\le m)\\
\le \quad & m+\frac{8}{9}\sum_{i\in M_{\mathrm{OPT}}\cap\mathcal{F}_A} c_iA^{L_i-1}\\
& (\text{since } \sum_{j:\mathrm{OPT}(j)=i} p_{ij}\le 1\ )\\
\le \quad & m+\frac{8}{9}\sum_{i\in\mathcal{F}_A} c_iA^{L_i-1}.
\end{aligned}$$

Finally, we bound the total potential ϕ; this immediately yields Lemma 3.1.

LEMMA 3.10. *The online fractional algorithm produces a schedule that satisfies $\phi = O(m\log m)$.*

Proof. Note that the final value of ϕ is the total increase in ϕ over all the algorithmic steps for all the clients minus the decrease in ϕ_i's when x_i's reach 1. Thus, the final value of ϕ is at most the total increase in ϕ over all the algorithmic steps. By Lemmas 3.2, 3.6, 3.8 and 3.9, we conclude that

$$\begin{aligned}
\phi \quad &\le \quad O(m\log m)+(8/9)\phi\\
\Rightarrow \quad \phi \quad &= \quad O(m\log m).
\end{aligned}$$

3.2 Online Randomized Rounding. In this section, we give an online randomized rounding scheme for the fractional solution produced by the algorithm.

The Algorithm. For each facility i, we select (offline) a number r_i uniformly at random and independently from $[0,1]$. In response to a new client j arriving online, the algorithm updates the assignment in three steps:

- **Fractional step.** The fractional solution is updated (via multiple algorithmic steps) as described in the fractional algorithm. Let $x_i(j)$ be the value of x_i after this update.
- **Activation step.** Each facility i that satisfies $5x_i(j)\ln(mn) \ge r_i$ (and is not already open) is opened. Let $M_{(j)}$ denote the set of open facilities after this step.
- **Assignment step.** Let

$$z_{ij} = \begin{cases} \frac{y_{ij}}{2x_i(j)} & \text{if } x_i(j) < \frac{1}{5\ln(mn)}\\ y_{ij} & \text{otherwise}\end{cases}$$

and

$$q_{ij} = \frac{z_{ij}}{\sum_{i\in\mathcal{F}_A(j)} z_{ij}}$$

for any facility $i\in\mathcal{F}_A(j)$. We assign client j to a facility $i\in\mathcal{F}_A(j)$ with probability q_{ij}.

Analysis. The next lemma is an immediate consequence of the fact that facility i is open in the integer solution with probability $\min(5x_i\ln(mn),1)$.

LEMMA 3.11. *The total opening cost of all facilities opened in the integer schedule is $O(m\log m\log(mn))$ in expectation.*

Proof. The expected opening cost of facility i in the integer schedule is at most $5c_ix_i\ln(mn)$; the lemma now follows from Lemma 3.1 and linearity of expectation.

To bound the total assignment cost and the maximum congestion of the integer solution, we first bound the probabilities q_{ij}. (This lemma also shows that with high probability, $\mathcal{F}_A(j)$ is non-empty even if j is the first client.)

LEMMA 3.12. *With probability at least $1-1/m$, $q_{ij}\le z_{ij}$ for all facilities $i\in\mathcal{F}$ and clients $j\in\mathcal{C}$.*

Proof. We show that $\sum_{i\in\mathcal{F}_A(j)} z_{ij}\ge 1$ with probability at least $1-1/mn$ for any client j; the lemma then follows using the union bound over all clients. Let $\mathcal{F}_1(j)$ denote the set of facilities i with $x_i(j)\ge 1/5\ln(mn)$, and let $\mathcal{F}_2(j)=\mathcal{F}\setminus\mathcal{F}_1(j)$. We have

$$\begin{aligned}
\sum_{i\in\mathcal{F}_A(j)} z_{ij} \quad &< \quad \sum_{i\in\mathcal{F}} y_{ij}\\
\text{if and only if } \sum_{i\in\mathcal{F}_2(j)\cap\mathcal{F}_A(j)} z_{ij} \quad &< \quad \sum_{i\in\mathcal{F}_2(j)} y_{ij},
\end{aligned}$$

since $z_{ij}=y_{ij}$ for each facility $i\in\mathcal{F}_1(j)$.

Define a random variable Z_{ij} for any facility $i\in\mathcal{F}_2(j)$ as

$$Z_{ij} = \begin{cases} 1 & \text{with probability } \frac{5y_{ij}\ln(mn)}{2}\\ 0 & \text{otherwise.}\end{cases}$$

Note that for any facility $i\in\mathcal{F}_2(j)$,

$$\begin{aligned}
\mathbb{E}[Z_{ij}] \quad &= \quad \frac{5y_{ij}\ln(mn)}{2}\\
&= \quad \frac{y_{ij}}{2x_i(j)}\times 5x_i(j)\ln(mn)\\
&= \quad z_{ij}\times\mathbb{P}[i\in\mathcal{F}_A(j)]. \qquad (3.19)
\end{aligned}$$

Note that $z_{ij} \in [0,1]$ since $y_{ij} \leq 2x_i(j)$ and $Z_{ij} \in \{0,1\}$ by definition. Therefore,

$$\begin{aligned}
& \mathbb{P}\left[\sum_{i\in\mathcal{F}_A(j)} z_{ij} < \sum_{i\in\mathcal{F}} y_{ij}\right] \\
= \; & \mathbb{P}\left[\sum_{i\in\mathcal{F}_2(j)\cap\mathcal{F}_A(j)} z_{ij} < \sum_{i\in\mathcal{F}_2(j)} y_{ij}\right] \\
\leq \; & \mathbb{P}\left[\sum_{i\in\mathcal{F}_2(j)} Z_{ij} < \sum_{i\in\mathcal{F}_2(j)} y_{ij}\right] \\
& \text{(by Eqn. (3.19) and since } z_{ij} \in [0,1]) \\
< \; & \frac{1}{mn},
\end{aligned}$$

by Chernoff bounds (cf. e.g. [38]).

Now, we bound the total assignment cost.

LEMMA 3.13. *The total assignment cost of all clients in the integer assignment is $O(m \log m \log(mn))$ in expectation.*

Proof. We will first prove that the expected assignment cost is $O\left(\sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} y_{ij}a_{ij}\ln(mn)\right)$, conditioned on the event that $q_{ij} \leq z_{ij}$ for all facilities $i \in \mathcal{F}$ and clients $j \in \mathcal{C}$. If $x_i(j) \geq 1/5\ln(mn)$, then the probability that client j is assigned to facility i is

$$q_{ij} \leq z_{ij} = y_{ij}.$$

On the other hand, if $x_i(j) < 1/5\ln(mn)$, this probability is at most

$$z_{ij} \times 5x_i(j)\ln(mn) = (5/2)y_{ij}\ln(mn).$$

To remove the conditioning and complete the proof, we use Lemma 3.12 and note that the maximum assignment cost of any assignment is mn.

Finally, we bound the makespan.

LEMMA 3.14. *The maximum congestion on any facility is $O(\log m)$ with probability $1 - 2/\sqrt{m}$.*

Proof. First, we prove that the congestion of any facility i is $O(\log m + \sum_{j\in\mathcal{C}} y_{ij}p_{ij})$ with probability at least $1 - 1/m^{3/2}$ conditioned on the event that $q_{ij} \leq z_{ij}$ for all facilities $i \in \mathcal{F}$ and clients $j \in \mathcal{C}$. Note that the congestion on facility i due to client j is p_{ij} with probability at most z_{ij}. We have

$$\sum_{j\in\mathcal{C}} z_{ij}p_{ij} \leq \sum_{j\in\mathcal{C}:x_i(j)<1} \frac{y_{ij}p_{ij}}{2x_i(j)} + \sum_{j\in\mathcal{C}} y_{ij}p_{ij}.$$

Let client j' immediately precede client j in the online order; if j is the first client, $x_i(j') = 1/m$. Then,

$$\begin{aligned}
& \sum_{j\in\mathcal{C}:x_i(j)<1} \frac{y_{ij}p_{ij}}{2x_{ij}} \\
\leq \; & (9/2) \sum_{j\in\mathcal{C}:x_i(j)<1} \frac{x_i(j) - x_i(j')}{x_i(j)} \\
\leq \; & (9/2) \sum_{j\in\mathcal{C}:x_i(j)<1} \int_{w=x_i(j')}^{x_i(j)} \frac{dw}{w} \\
\leq \; & (9/2) \int_{1/m}^{1} \frac{dw}{w} \\
\leq \; & (9/2)\ln m.
\end{aligned}$$

The first inequality follows from the fractional algorithm which assigns a fraction $y_{ij} \leq 5(x_i(j) - x_i(j'))/p_{ij}$ of client j to facility i. Since $y_{ij}p_{ij} \leq p_{ij} \leq 1$ for all clients j, it follows using Chernoff bounds that the congestion on machine i is $O(\log m + \sum_{j\in\mathcal{C}} y_{ij}p_{ij})$ with probability at least $1 - 1/m^{3/2}$.

Using the union bound over all facilities and Lemma 3.12, we can now claim that with probability at least $1 - 2/\sqrt{m}$, the congestion on every facility is $O(\log m + \sum_{j\in\mathcal{C}} y_{ij}p_{ij})$. The lemma now follows from Lemma 3.1.

Finally, we note that Lemmas 3.11, 3.13, and 3.14 imply Theorem 1.3.

4 Conclusion

In this paper, we have presented an algorithm for solving a mixed packing and covering LP, where the packing constraints are given offline and the covering constraints arrive online. The solution produced by our algorithm satisfies each covering constraint exactly, and violates each packing constraint by at most a polylogarithmic factor. We also observed that such violations are necessary and near optimal from information theoretic considerations. We employed the ideas developed in this generic algorithm to solve two canonical problems represented by mixed packing and covering LPs: the unrelated machine scheduling problem with startup costs and the more general capacity constrained facility location problem.

Our work opens up multiple directions of future research. One concrete question relates to the unrelated machine scheduling problem with startup costs. Recently, algorithms have been proposed for offline versions of this problem that can handle cost functions that are more general than startup costs considered in this work [36]. A technical challenge would be to extend our algorithmic framework to handle these more gen-

eral cost functions. Another compelling direction of research would be to consider alternative input models for mixed packing and covering LPs. This paper only considers the model where packing constraints are available offline and covering constraints arrive online. It is feasible that some natural online optimization problems that can be expressed as mixed LPs do not conform to this model, and therefore require new ideas beyond those developed in this paper.

References

[1] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006.

[2] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009.

[3] Noga Alon, Yossi Azar, and Shai Gutner. Admission control to minimize rejections and online set cover with repetitions. *ACM Transactions on Algorithms*, 6(1), 2009.

[4] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta algorithm and applications. Technical report, 2005.

[5] James Aspnes, Yossi Azar, Amos Fiat, Serge A. Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 44(3):486–504, 1997.

[6] Yossi Azar. On-line load balancing. In *Online Algorithms*, pages 178–195, 1996.

[7] Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line assignments. *J. Algorithms*, 18(2):221–237, 1995.

[8] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. In *FOCS*, pages 507–517, 2007.

[9] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. In *STOC*, pages 235–244, 2008.

[10] Yair Bartal, Amos Fiat, Howard J. Karloff, and Rakesh Vohra. New algorithms for an ancient scheduling problem. *J. Comput. Syst. Sci.*, 51(3):359–366, 1995.

[11] Daniel Bienstock and Garud Iyengar. Approximating fractional packings and coverings in o(1/epsilon) iterations. *SIAM J. Comput.*, 35(4):825–854, 2006.

[12] Ken Birman, Gregory Chockler, and Robbert van Renesse. Toward a cloud computing research agenda. *SIGACT News*, 40(2):68–80, 2009.

[13] Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, pages 253–264, 2007.

[14] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing problems. In *ESA*, pages 689–701, 2005.

[15] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009.

[16] György Dósa and Yong He. Better online algorithms for scheduling with machine cost. *SIAM J. Comput.*, 33(5):1035–1051, 2004.

[17] György Dósa and Zhiyi Tan. New upper and lower bounds for online scheduling with machine cost. *Discrete Optimization*, 7(3):125–135, 2010.

[18] Lisa Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discrete Math.*, 13(4):505–520, 2000.

[19] Lisa Fleischer. Data center scheduling, generalized flows, and submodularity. In *ANALCO*, pages 56–65, 2010.

[20] Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

[21] Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37(2):630–652, 2007.

[22] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Tech. Journal*, 45:1563–1581, 1966.

[23] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.

[24] Michael D. Grigoriadis and Leonid G. Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM Journal on Optimization*, 4(1):86–107, 1994.

[25] Michael D. Grigoriadis and Leonid G. Khachiyan. An exponential-function reduction method for block-angular convex programs. *Networks*, 26(2):59–68, 1995.

[26] Michael D. Grigoriadis and Leonid G. Khachiyan. Approximate minimum-cost multicommodity flows in $\tilde{o}(\epsilon^{-2}knm)$ time. *Math. Program.*, 75:477–482, 1996.

[27] Csanád Imreh. Online scheduling with general machine cost functions. *Discrete Applied Mathematics*, 157(9):2070–2077, 2009.

[28] Csanád Imreh and John Noga. Scheduling with machine cost. In *RANDOM-APPROX*, pages 168–176, 1999.

[29] Samir Khuller, Jian Li, and Barna Saha. Energy efficient scheduling via partial shutdown. In *SODA*, pages 1360–1372, 2010.

[30] Philip N. Klein, Serge A. Plotkin, Clifford Stein, and Éva Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM J. Comput.*, 23(3):466–487, 1994.

[31] Antoon Kolen and Arie Tamir. Covering problems. In P.B. Mirchandani and R.L. Francis, editors, *Discrete Location Theory*. John Wiley, 1991.

[32] Christos Koufogiannakis and Neal E. Young. Beating simplex for fractional packing and covering linear programs. In *FOCS*, pages 494–504, 2007.

[33] Frank Thomson Leighton, Fillia Makedon, Serge A. Plotkin, Clifford Stein, Éva Stein, and Spyros Tragoudas. Fast approximation algorithms for multicommodity flow problems. *J. Comput. Syst. Sci.*, 50(2):228–243, 1995.
[34] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.
[35] Retsef Levi, David B. Shmoys, and Chaitanya Swamy. Lp-based approximation algorithms for capacitated facility location. In *IPCO*, pages 206–218, 2004.
[36] Jian Li and Samir Khuller. Generalized machine activation problems. In *SODA*, pages 80–94, 2011.
[37] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation algorithms for metric facility location problems. *SIAM J. Comput.*, 36(2):411–432, 2006.
[38] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.
[39] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. In *FOCS*, pages 495–504, 1991.
[40] Jiri Sgall. On-line scheduling. In *Online Algorithms*, pages 196–231, 1996.
[41] Farhad Shahrokhi and David W. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, 1990.
[42] Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *FOCS*, pages 538–546, 2001.
[43] Jiawei Zhang, Bo Chen, and Yinyu Ye. A multiexchange local search algorithm for the capacitated facility location problem. *Math. Oper. Res.*, 30(2):389–403, 2005.

A Additional Details of the OMPC Algorithm

In this section, we will give the missing details of our OMPC algorithm.

A.1 Proof of Lemma 2.6. Lemma A.1 is a technical result used in the proof of Lemma 2.6.

LEMMA A.1. *Let $y := \sum_{i=1}^{n} r_i$, where $0 < r_i \leq 1$ for $i \in [n]$, and $\prod_{i=1}^{n} r_i = P$. Then y is minimized when $r_i = P^{1/n}\ \forall i$, and the minimum value is $nP^{1/n}$.*

Proof. The proof is by induction on n. For $n = 1$, the lemma is obviously true. Let $\gamma_k(P)$ be the minimum value of the sum of k variables, when the product of the variables is P. Then $\gamma_n(P) = \min_{0<r_1\leq 1}\{r_1 + \gamma_{n-1}(P/r_1)\}$. By the inductive hypothesis, $\gamma_{n-1}(P/r_1) = (n-1)(P/r_1)^{1/(n-1)}$. Hence

$$\gamma_n(P) \;=\; \min_{0<r_1\leq 1}\left\{r_1 + (n-1)\left(\frac{P}{r_1}\right)^{1/(n-1)}\right\}.$$

We will show that the expression on the right is minimized when $r_1 = P^{1/n}$. Then by the inductive hypothesis, each of the other variables is $P^{1/n}$ as well, completing the proof.

Let $z = r_1 + (n-1)\left(\frac{P}{r_1}\right)^{1/(n-1)}$. Then

$$\frac{dz}{dr_1} \;=\; 1 - \frac{1}{n-1} r_1^{-n/(n-1)} (n-1) P^{1/(n-1)},$$

and setting $dz/dr_1 = 0$, we obtain $r_1 = P^{1/n}$. Further, $d^2z/dr_1^2 \geq 0\ \forall r_1 \geq 0$. Hence, the point $r_1 = P^{1/n}$ is a minimum. This completes the proof.

Proof of Lemma 2.6. For $n = 1$, the statement is trivially true. For $n \geq 2$, define $b_i = \sum_{j\leq i} a_j$. Then $a_i = b_i - b_{i-1}$ for $i \geq 2$, and hence

$$\text{(A.1)}\qquad \sum_{i\in[n]} \frac{a_i}{\sum_{j\leq i} a_j} \;=\; 1 + \sum_{i=2}^{n} \frac{b_i - b_{i-1}}{b_i} \;=\; 1 + \sum_{i=2}^{n}\left(1 - \frac{b_{i-1}}{b_i}\right).$$

Let $r_i = \frac{b_i}{b_{i+1}}$, and let $y = \sum_{i=2}^{n}(1 - \frac{b_{i-1}}{b_i})$. Then

$$y = \sum_{i=1}^{n-1}(1 - r_i) = (n-1) - \sum_{i=1}^{n-1} r_i.$$

Since each $r_i \leq 1$ and $\prod_{i=1}^{n-1} r_i = b_1/b_n$, we have (by Lemma A.1)

$$\text{(A.2)}\quad y \;\leq\; (n-1) - (n-1)\left(\frac{b_1}{b_n}\right)^{1/(n-1)}.$$

Let $c = \frac{b_1}{b_n}$ and $z = (n-1) - (n-1)c^{1/(n-1)}$. Differentiating z w.r.t. n,

$$\frac{\partial z}{\partial n} = 1 - c^{1/(n-1)} + \frac{c^{1/(n-1)}}{n-1}\ln c$$

and again,

$$\partial^2 z/\partial n^2 = -c^{1/(n-1)}\ln^2 c/(n-1)^3 < 0.$$

Hence, z is maximized when

$$(n-1) - (n-1)c^{1/(n-1)} = c^{1/(n-1)}\ln\frac{1}{c}.$$

Substituting the expression on the left in this equality in Eqn. (A.2) gives us

$$y \leq c^{1/(n-1)}\ln\frac{1}{c},$$

and since

$$c = \frac{b_1}{b_n} \leq 1,$$

we have

$$y \leq \ln \frac{1}{c} = \ln \frac{b_n}{b_1} = \ln \frac{\sum_{i=1}^{n} a_i}{a_1}. \tag{A.3}$$

Then from Eqns. (A.1) and (A.3), and by the definition of y,

$$\sum_{i \in [n]} \frac{a_i}{\sum_{j \leq i} a_j} \;\leq\; 1 + \ln \frac{\sum_{i=1}^{n} a_i}{a_1}.$$

A.2 Bounding OPT**.** We now discuss our doubling procedure that allows us to bound OPT as required by our algorithm. In our discussion, we use $\tilde{\mathbf{P}}$ for the matrix of packing coefficients without scaling, and OPT for the optimal solution to (2.1) with the unscaled packing constraints. We initially set $\Gamma = \max_{k,j:\tilde{p}_{kj}>0} \tilde{p}_{kj}/(d_1\rho\kappa_1)$ and use this value to scale the packing constraints. We run our algorithm with the scaled packing constraints. If during the execution of the algorithm $\lambda(\mathbf{x})$ exceeds $3\ln(em)$, we double Γ, scale the packing constraints by the new value of Γ and restart the algorithm. We repeat this each time $\lambda(\mathbf{x})$ exceeds $3\ln(em)$.

Call each execution of the algorithm a *trial*. Each trial τ has distinct primal and dual variables $(\lambda(\tau), \mathbf{x}(\tau))$ and $(\mathbf{y}(\tau), \mathbf{z}(\tau))$ that are initialized at the start of the trial and increase as the trial proceeds. At the start of the trial, each $x_j(\tau)$ is initialized to $x_j^0(\tau) = 1/(d_1^2\rho\kappa_1)$. If a trial fails, we double the value of Γ and proceed with the next trial with new primal and dual variables. Thus in every trial, $\Gamma \geq \max_{k,j:\tilde{p}_{kj}>0} \tilde{p}_{kj}/(d_1\rho\kappa_1)$.

Our final solution $(\mathbf{x}, \lambda)$ is the sum of the values obtained in each trial. Thus, our variables are non-decreasing. Let $\Gamma(\tau)$ be the value of Γ used in trial τ, and $\lambda^f(\tau)$ be the value of the primal $\lambda(\tau)$ when trial τ ends. T is the last trial, i.e., the algorithm does not fail in trial T. Since $\mathbf{x}$ obtained by the algorithm is the sum of $\mathbf{x}(\tau)$ in each trial τ, the value of the primal objective obtained by the algorithm is at most $\sum_{\tau \leq T} \Gamma(\tau)\lambda^f(\tau)$. Then

THEOREM A.1. *The value of the primal objective* $\sum_{\tau \leq T} \Gamma(\tau)\lambda^f(\tau)$ *obtained is* $O(\ln m \ln(d\rho\kappa))\text{OPT}$.

We first show a bound on Γ in any trial.

LEMMA A.2. *In any trial,* $\Gamma \leq 4\sigma\text{OPT}$.

Proof. Initially, $\Gamma = \max_{k,j:p_{kj}>0} \tilde{p}_{kj}/(d_1\rho\kappa_1) \leq \text{OPT}$ by (2.5). Hence the lemma is true for the first trial. Theorem 2.1 proves that if $4\sigma\text{OPT} \geq \Gamma \geq 2\sigma\text{OPT}$, then $\lambda(\mathbf{x}) \leq 3\ln(em)$ and the algorithm does not fail. Since Γ is doubled after each failed trial, by Theorem 2.1 some trial with $\Gamma \leq 4\sigma\text{OPT}$ will not fail. Hence, for every trial, $\Gamma \leq 4\sigma\text{OPT}$.

Proof of Theorem A.1. By Corollary 2.1, $\lambda(\tau) \leq 3\ln(em)$ at the start of any phase. Within a phase, each variable gets multiplied by at most a factor of $\mu = 1 + 1/(3\ln(em))$. Hence when trial τ fails, $\lambda^f(\tau) \leq 1 + 3\ln(em) \leq 4\ln(em)$, or $\Gamma(\tau)\lambda^f(\tau) \leq 4\Gamma(\tau)\ln(em)$. Since the value of $\Gamma(\tau)$ doubles after each trial,

$$\begin{aligned} \sum_{\tau \leq T} \Gamma(\tau)\lambda^f(\tau) \;&\leq\; 4\ln(em) \sum_{\tau \leq T} \Gamma(\tau) \\ &=\; 4\ln(em) \sum_{\tau \leq T} 2^{\tau - T}\Gamma(T) \\ &\leq\; 8\ln(em)\Gamma(T)\,. \end{aligned} \tag{A.4}$$

Thus, from Eqn. (A.4) and Lemma A.2,

$$\sum_{\tau \leq T} \lambda^f(\tau) \leq 32\sigma\ln(em)\text{OPT}.$$