Online Set Cover with Set Requests*

Kshipra Bhawalkar[1], Sreenivas Gollapudi[2], and Debmalya Panigrahi[3]

1 Google Inc., Mountain View, CA, U.S.
kshipra@google.com
2 Microsoft Research Search Labs, Mountain View, CA, U.S.
sreenig@microsoft.com
3 Duke University, Durham, NC, U.S.
debmalya@cs.duke.edu

Abstract

We consider a generic online allocation problem that generalizes the classical online set cover framework by considering requests comprising a set of elements rather than a single element. This problem has multiple applications in cloud computing, crowd sourcing, facility planning, etc. Formally, it is an online covering problem where each online step comprises an offline covering problem. In addition, the covering sets are capacitated, leading to packing constraints. We give a randomized algorithm for this problem that has a nearly tight competitive ratio in both objectives: overall cost and maximum capacity violation. Our main technical tool is an online algorithm for packing/covering LPs with nested constraints, which may be of interest in other applications as well.

1998 ACM Subject Classification F.2.2 Non-numerical Algorithms and Problems

Keywords and phrases Online Algorithms, Set Cover

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2014.64

1 Introduction

In recent years, significant research has been conducted in online allocation problems (see [1] and [8] for a comprehensive discussion on online algorithms), often motivated by inherently online modern applications such as internet advertising, crowd sourcing, scheduling in the cloud, etc. We continue this research effort in this paper by considering a generic allocation problem that is motivated by various real-world applications and generalizes the well-studied online set cover framework. In the online set cover problem [2], a collection of subsets (of given costs) of a universe of elements are given offline and elements from the universe arrive online. At any time, the algorithm must maintain a monotonically increasing (over time) collection of subsets of minimum cost that cover all the elements that have arrived thus far. In the capacitated version, every set also has a given capacity which represents the maximum number of elements it can cover. In this paper, we consider a natural generalization of this problem, where instead of a single new element, a subset of elements arrives in each online step. Note that this generalization is meaningful only in the capacitated situation since the elements arriving in the same online step use up only one unit of capacity of the covering sets. In the uncapacitated (i.e., infinite capacity) scenario, the elements arriving in a single step can be thought of as arriving sequentially.

* Part of this work was done when all the authors were at Microsoft Research.

17th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'14) / 18th Int'l Workshop on Randomization and Computation (RANDOM'14).
Editors: Klaus Jansen, José Rolim, Nikhil Devanur, and Cristopher Moore; pp. 64–79
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

To formally describe our problem, we need to introduce some notation and terminology. Departing from the usual set cover notation, we think of every element as a resource and every covering set as a facility that provides some subset of resources. This ties the notation to natural applications of the problem and helps us distinguish between the request sets (that arrive online) and the covering sets (that are offline and called facilities now). Let U be the set of n different resources (such as goods and services) and $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$ be a set of facilities, each of which can provide some subset $S_j \subseteq U$ of resources. Each facility S_j also has an associated cost c_j and capacity t_j. The above are given offline. There are k *requests* that arrive online. In each online step, a request $R_i \subseteq U$ arrives, and has to be satisfied by assigning a subset of facilities to it that can cumulatively provide all the resources requested, i.e. by using a subset of facilities $\mathbf{T}_i \subseteq \mathbf{S}$ such that $R_i \subseteq \cup_{T \in \mathbf{T}_i} T$. The capacity of a facility is the maximum number of requests it can serve, and the ratio of the number of requests served by a facility to its capacity is called its *congestion*. The goal is to minimize the sum of costs of the facilities purchased by the algorithm. We call this the Cover-SetReq problem. Our focus, in this paper, will be to design an online algorithm for the Cover-SetReq problem.

Our work was motivated by various applications of the above general framework in emerging domains. We give a couple of motivating examples below:

- **Distributed Computing:** In distributed computing environments such as cloud computing and crowd sourcing, each computing unit (e.g., a human or a server) provides a subset of computing resources and has a maximum capacity. The goal is to minimize cost while allocating each arriving task to a subset of computing units that have adequate resources to solve it.
- **Facility Planning:** The goal is to minimize the cost of facilities (each of which can provide a subset of services and has a maximum capacity) to serve service requests that grow over time as new customers are added.
- **Subscription Markets.** In addition to traditional products, the internet has emerged as the principal medium for the sale of services based on information and data management including access to data sets and computing resources (see, e.g., [7, 13, 14]). Examples include the Windows Azure Marketplace[1], Amazon Web Services[2], etc. These services are typically sold as *subscriptions* comprising one or more resources that come as a bundle with an usage limit. The consumer objective is to satisfy their data/computing needs which arrive over time at minimum cost by buying an optimal set of subscriptions.

Our main result is a polynomial-time online algorithm for the Cover-SetReq problem. To state its competitive ratio, let us use an equivalent (up to a constant factor in the competitive ratio, by a standard doubling search approach) description of the Cover-SetReq problem, where in addition to the input described above, a cost bound $\mathbf{C}$ is given offline with the guarantee that there exists a feasible solution, i.e. a solution that does not use more than the capacity of any facility and has total cost at most $\mathbf{C}$. Then, an online algorithm for the Cover-SetReq problem is said to have a bi-criteria competitive ratio of (α, β) if its total cost is at most $\alpha\mathbf{C}$ and for every facility, the number of requests that it is used to satisfy is at most β times its capacity (i.e., its congestion is at most β). Our main theorem obtains poly-logarithmic factors for both α and β.

[1] `http://datamarket.azure.com`
[2] `http://aws.amazon.com`

$$
\begin{aligned}
\textstyle\sum_{j:j\in[m]} c_j x_j &\le \mathbf{C} && && (1)\\
\textstyle\sum_{j:u\in S_j} y_{ij} &\ge 1 && \forall u \in R_i, \forall i \in [k] && (2)\\
y_{ij} &\le 2x_j && \forall i \in [k], \forall j \in [m] && (3)\\
\textstyle\sum_{i:i\in[k]} y_{ij} &\le x_j t_j && \forall j \in [m] && (4)\\
0 \le y_{ij} &\le 1 && \forall i \in [k], \forall j \in [m] && (5)\\
0 \le x_j &\le 1 && \forall j \in [m] && (6)
\end{aligned}
$$

Figure 1 Linear program for the Cover-SetReq problem.

▶ **Theorem 1.** *There is a randomized online algorithm for the* Cover-SetReq *problem that has a competitive ratio of* (α, β) *where* $\alpha = O(\log(mn)\log(kmn))$ *and* $\beta = O(\log n(\log m + \log\log n)\log(kmn))$.

We note that this theorem is nearly tight since there are logarithmic lower bounds for both α and β: (1) there is a (randomized) lower bound of $\Omega(\log m \log n)$ [2, 12] for the competitive ratio of the online set cover problem, which holds for the cost objective of the Cover-SetReq problem, and (2) there is a lower bound of $O(\log m)$ [3] for the online restricted assignment problem, which holds for the congestion objective of the Cover-SetReq problem.

We remark that some applications motivate a version of the Cover-SetReq problem with soft capacities, i.e. where multiple copies of a facility can be used in the solution. Clearly, our algorithm has a poly-logarithmic competitive ratio for this problem as well. However, this problem can be solved using an alternative (simpler) technique:

- linearize the cost of all copies of a facility other than its first copy by losing a factor of 2 (see Jain and Vazirani [11])
- reduce the mixed LP to a covering LP of exponential size (but with an efficient separation oracle) by eliminating precedence packing constraints
- obtain a fractional solution to the covering problem using a standard template given by Buchbinder and Naor [9] (see also Gupta and Nagarajan [10])
- obtain an integer solution using our randomized rounding procedure.

The details of these steps appear in the appendix. The second step fails for the Cover-SetReq problem, i.e., when we have hard capacities.

Our Techniques. First, we define an LP for the Cover-SetReq problem (in Figure 1). Let x_j denote whether facility S_j is opened and y_{ij} indicate whether facility S_j is used to serve request R_i. We enforce that each resource in every request is served (i.e. $\sum_{j:u\in S_j} y_{ij} \ge 1$) and to ensure a bounded integrality gap, that $y_{ij} \le 2x_j$ (the factor 2 is for technical reasons). In addition, the total cost is bounded ($\sum_{j\in[m]} c_j \le \mathbf{C}$) and the congestion on every facility S_j is bounded ($\sum_{i\in[n]} y_{ij} \le x_j t_j$).

As mentioned earlier, we will obtain a fractional solution to this LP (which will violate some of the constraints) using combinatorial techniques and then round this fractional solution online. This recipe was suggested originally by Alon *et al.* [2] for the online set cover problem and has since been used extensively for online algorithms (see the survey by Buchbinder and Naor [9] for more details). The online rounding algorithm is the easier of the two steps and (somewhat delicately) combines rounding techniques for the online [5] and offline [16] set cover problems.

Obtaining a fractional algorithm turns out to be much more challenging. Since the Cover-SetReq problem is represented by a mixed packing covering LP, following Azar *et al.* [5], for each request, we use a sequence of multiplicative updates on a prefix of facilities with $x_j < 1$ ordered by a carefully chosen function that represents the derivative of the overall potential of the solution. However, unlike in [5], since requests contain multiple resources, the prefix is not unique, rather it depends on the resource being considered. Moreover, it is not immediate as to how we can compare between two facilities, one with many resources but higher cost and another with fewer resources but lower cost. To complicate matters further, at any stage of the multiplicative weights update process, different resources are at various stages of being served: some have been completely served, some only partially served, while others have not been served at all. The resources that have been fully served should cease to influence the ordering since the facilities providing these resources are no longer contributing to serving these resources.

Since each online step is an offline set cover problem, we inherit its greedy property and order facilities by the potential increase *per resource* that each facility provides (call it the *scaled cost*). To address the issue of some resources having already been completely served, we make these prefix orderings *dynamic*: once a resource has been completely served, it is not included in defining scaled costs thereafter. Moreover, since each resource only appears in some of the facilities, we introduce the notion of a *resource specific* prefix ordering, which is a subsequence of the overall prefix ordering.

For the fully open facilities (i.e., $x_j = 1$), we need to ensure that the maximum congestion is small. For this purpose, we follow a technique introduced by Aspnes *et al* [3] (see also [6, 4]) for online load balancing, where a greedy algorithm on an exponential potential function of the machine loads is used. Our main technical contribution is a procedure that co-ordinates between the greedy selection of facilities in prefixes, multiplicative weight updates on these multiple prefixes, and greedy assignment of requests to facilities according to an exponential potential function of their congestion for fully open facilities.

Roadmap. The next section presents the online algorithm, whose competitive ratio is derived in two parts: the analysis of the fractional solution is in section 3 and the analysis of the randomized rounding procedure to convert the fractional solution into an integer one is in section 4. In the appendix, we present a simpler algorithm for the soft-capacitated version of the Cover-SetReq problem (section A).

2 Description of the Algorithm

The algorithm has three phases: (a) an offline pre-processing phase, (b) an online phase that produces a fractional solution, and (c) an online rounding phase that produces an integer solution from the fractional solution. The last two phases are interleaved. Recall that we are given a bound on the cost $\mathbf{C}$ and the number of requests k in advance. Let OPT denote a solution that has congestion at most 1 on every facility and total cost at most $\mathbf{C}$.

The Offline Pre-processing Phase. First, we discard all facilities S_j with $c_j > \mathbf{C}$ from $\mathbf{S}$. Clearly, none of these facilities were being used by OPT. From now on, m will denote the size of $\mathbf{S}$ after this step. Next, we divide the cost of each facility by $\frac{\mathbf{C}}{m}$. After this scaling, the total cost of OPT is at most m. For any facility $S_j \in \mathbf{S}$, if $c_j < \frac{1}{k}$, we increase c_j to $\frac{1}{k}$. After this transformation, the total cost of OPT is at most $\left(1 + \frac{1}{k}\right) m < 2m$.

Let $x_j^{(i)}$ denote the value of variable x_j at the end of the updates for request R_i. Note that the non-decreasing property of x_j requires that $x_j^{(i)} \geq x_j^{(i-1)}$. We say that facility S_j is *fully open* if $x_j = 1$, and *partially open* otherwise. We initialize x_j to $x_j^{(0)} = \frac{1}{m}$ for all facilities $S_j \in \mathbf{S}$. Therefore, initially, all facilities are partially open.

Online Updates to the Fractional Solution. Suppose a new request R_i arrives online. Any resource $u \in R_i$ is said to be *satisfied* if $\sum_{j:u \in S_j} y_{ij} \geq 1$. Clearly, R_i is satisfied when all resources in R_i are satisfied. We start by setting $x_j^{(i)} = x_j^{(i-1)}$ (required by monotonicity of the fractional solution). We increase the value of $x_j^{(i)}$ on selected facilities S_j in small increments over multiple *rounds* and make corresponding increments in y_{ij}. Each round, in turn, consists of multiple *iterations*.

Let $\overline{R}_i$ denote the set of resources in R_i that are not yet satisfied at the beginning of the round, i.e., $\overline{R}_i = \{u \in R_i : \sum_{j:u \in S_j} y_{ij} < 1\}$. The increments in the values of $x_j^{(i)}$ and y_{ij} in any particular round are based on defining a sequence of facilities containing u (called the *prefix* for u and denoted $\mathbf{P}_i(u)$) for each individual resource $u \in \overline{R}_i$. For some of the resources $u \in \overline{R}_i$, we will also define an additional facility in $\mathbf{S} \setminus \mathbf{P}_i(u)$ as the *boundary facility* for u, and denote the index of this facility by $p_i(u)$. Let $\widehat{\mathbf{P}}_i(u) = \mathbf{P}_i(u) \cup S_{p_i(u)}$; we call this the *closed prefix* of u.

To describe the update rule of the fractional variables and the construction of the prefixes, we need some additional notation. For every facility S_j, we partition requests into those that arrive before S_j is fully open (denote this set $R_0(j)$) and those that arrive after (denote this set $R_1(j)$). For the request that was being served when the facility became fully open, we consider the part of the request that arrived while $x_j < 1$ in $R_0(j)$ and the rest of the request in $R_1(j)$. The *virtual congestion* (denoted $\tilde{L}_j$) of a facility S_j is defined as

$$\tilde{L}_j = \begin{cases} x_j & \text{if } x_j < 1 \\ 1 + \sum_{i:R_i \in R_1(j)} \frac{y_{ij}}{t_j} & \text{if } x_j = 1. \end{cases}$$

Now, we define a function (A is a constant that we will fix later)

$$\psi_j = \begin{cases} \frac{c_j}{t_j} & \text{if } x_j < 1 \\ \frac{c_j A^{\tilde{L}_j}(A-1)}{t_j} & \text{if } x_j = 1. \end{cases}$$

The updates for all facilities in prefix $\mathbf{P}_i(u)$ and the boundary facility $S_{p_i(u)}$ are collectively called an iteration for resource u, and the iterations for all resources in $\overline{R}_i$ constitute a round for request R_i. The update rule for a round is given in Algorithm 1, where N is a discretization parameter that we set to kmn^2. One important point to note is that if a partially open facility S_j belongs to k_j closed prefixes, then the value of $x_j^{(i)}$ increases in multiplicative update steps k_j times in a single round.

Definition of the Prefixes. We initialize the prefix $\mathbf{P}_i(u)$ to the empty sequence for every resource $u \in \overline{R}_i$. The prefixes are populated in a sequence of steps, where in each step, we add a carefully selected facility to some of the prefixes. To describe a step, we need some additional notation. Let $\overline{\overline{R}}_i$ denote the set of resources in $\overline{R}_i$ whose prefix has not been fully defined yet. Clearly, $\overline{\overline{R}}_i$ equals $\overline{R}_i$ at the beginning of a round. Further, let $\mathbf{S}(i)$ denote the collection of facilities in $\mathbf{S}$ that overlap $\overline{\overline{R}}_i$ and have not been used in a previous step (i.e. is not part of any prefix currently). Initially, $\mathbf{S}(i) = \{S_j \in \mathbf{S} : S_j \cap \overline{R}_i \neq \emptyset\}$.

For any facility $S_j \in \mathbf{S}(i)$, let its *scaled cost* be $\phi_j = \frac{\psi_j}{|S_j \cap \overline{\overline{R}}_i|}$.

Algorithm 1 A Single Round of the Fractional Algorithm

- $\overline{R}_i = \{u \in R_i : \sum_{j:u\in S_j} y_{ij} < 1\}$.
- Create closed prefixes $\widehat{\mathbf{P}}_i(u)$ simultaneously for all resources $u \in \overline{R}_i$.
- For every facility S_j: initialize $\Delta x_j = \Delta y_{ij} = 0$.
- For every resource $u \in \overline{R}_i$: for every partially open facility $S_j \in \widehat{\mathbf{P}}_i(u)$, we increase Δx_j by $\frac{x_j^{(i)}}{c_j N}$ (sequentially, in arbitrary order over the closed prefixes $\widehat{\mathbf{P}}_i(u)$ for all resources $u \in \overline{R}_i$).
- For every facility S_j: if S_j is partially open, we set $\Delta y_{ij} = \min\left((\Delta x_j)t_j, 2(x_j^{(i)} + \Delta x_j) - y_{ij}\right)$; if S_j is fully open, we set $\Delta y_{ij} = \frac{1}{\psi_j N}$.
- For every facility S_j: increase $x_j^{(i)}$ by Δx_j and y_{ij} by Δy_{ij}.

In each step, the algorithm performs the following operations:

1. Find facility $S_j \in \mathbf{S}(i)$ that has the least value of ϕ_j; let us denote its index by j^*.
2. Remove S_{j^*} from $\mathbf{S}(i)$.
3. Let $\mathbf{x}(u) = \sum_{j:S_j\in\mathbf{P}_i(u)} x_j^{(i)} + x_{j^*}^{(i)}$. For each resource $u \in S_{j^*} \cap \overline{\overline{R}}_i$, if $\mathbf{x}(u) < 1$, then we add S_{j^*} to the prefix $\mathbf{P}_i(u)$. Otherwise, if $\mathbf{x}(u) \geq 1$, then we define S_{j^*} as the boundary facility for resource u, i.e., $p_i(u) = j^*$ and remove u from $\overline{\overline{R}}_i$.
4. Re-define $\mathbf{S}(i)$ (since $\overline{\overline{R}}_i$ might have changed) and re-compute ϕ_j for all facilities $S_j \in \mathbf{S}(i)$ (even if a facility continues to be in $\mathbf{S}(i)$, its scaled cost might have changed since $\overline{\overline{R}}_i$ has changed).

Note that it might so happen that for a resource $u \in R_i$, even after including all facilities containing u in the prefix $\mathbf{P}_i(u)$, $\sum_{j:S_j\in\mathbf{P}_i(u)} x_j^{(i)} < 1$. In this case, the boundary facility for u is undefined, and its closed prefix is identical to its prefix.

Online Randomized Rounding. There are two decisions that the integer algorithm must make on receiving a new request R_i. First, it needs to decide which set of facilities it wants to open. Since decisions are irrevocable in the online model, the open facilities form a monotonically growing set over time. Next, the algorithm must decide which of the open facilities it will use to satisfy request R_i. 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 above.

To simplify the analysis later, we will consider two copies of each facility: 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 the congestion. First, we define a randomized process that controls the opening of blue copies of facilities in the integer algorithm. Let $\mathbf{S}_o(i)$ denote the set of facilities whose blue copies are open after request R_i has been satisfied, and $X_j^{(i)}$ be an indicator random variable whose value is 1 if facility $i \in \mathbf{S}_o(i)$ and 0 otherwise. Let $x_j^{(i)}$ be the value of variable x_j in the fractional solution after request R_i has been completely assigned (fractionally). For a parameter $\alpha = \Theta(\log(kmn))$, the integer algorithm maintains the following invariant for every facility S_j and request R_i:

$$\mathbb{P}[X_j^{(i)} = 1] = \min(\alpha \cdot x_j^{(i)}, 1), \tag{7}$$

using the rule for opening facilities in Algorithm 2. Next, we need to use the open facilities to satisfy request R_i. Let Y_{ij} be the indicator variable for facility S_j being used to serve request R_i. Define

$$z_{ij} = \begin{cases} 0 & \text{if } X_j^{(i)} = 0 \\ \frac{y_{ij}}{2x_j^{(i)}} & \text{if } X_j^{(i)} = 1 \text{ and } x_j^{(i)} < \frac{1}{\alpha} \\ \alpha \cdot y_{ij} & \text{otherwise.} \end{cases}$$

The assignment rule for request R_i is given in Algorithm 2.

Algorithm 2 Satisfying a Single Request R_i in the Integer Algorithm

---

Opening Facilities:

- For every facility S_j whose blue copy is not already open, open it with probability $\min\left(\frac{\alpha(x_j^{(i)} - x_j^{(i-1)})}{1-\alpha \cdot x_j^{(i-1)}}, 1\right)$. (Eqn. 7 is satisfied by this rule using conditional probabilities.)

Satisfying Request R_i:

- For every open facility S_j, we set $Y_{ij} = 1$ independently with probability z_{ij}.
- For every resource $u \in R_i$ such that no facility containing u was selected in the previous step, set $Y_{ij} = 1$ for the red copy of *any* facility S_j such that $u \in S_j$, after opening the facility if necessary.

---

3 Analysis of the Fractional Algorithm

We note that the fractional solution maintains the invariant $\sum_{R_i \in R_0(j)} \frac{y_{ij}}{t_j} \leq x_j$ for every facility S_j. This invariant ensures that the actual congestion of any facility is always at most its virtual congestion (denoted $\tilde{L}_j$; see section 2 for its formal definition). Therefore, it suffices to bound the total cost and the maximum virtual congestion on the facilities. For this purpose, we design a potential function that combines these two objectives: $\gamma_j = c_j x_j A^{\tilde{L}_j / x_j}$ for some $A \in (1, 2)$ that we will fix later. Note that we can rewrite the potential function as

$$\gamma_j = \begin{cases} A c_j x_j & \text{if } x_j < 1 \\ c_j A^{\tilde{L}_j} & \text{if } x_j = 1. \end{cases}$$

The potential function is continuous and monotonically non-decreasing. We define the overall potential $\Gamma = \sum_{j: S_j \in \mathbf{S}} \gamma_j$.

The next lemma bounds the potential function at the end of the pre-processing step.

▶ **Lemma 2.** *At the end of the pre-processing step, $\Gamma \leq m$.*

Proof. There are m partially open facilities, the cost of each of which is at most m. Since we initialize $x_j^{(0)} = 1/m$ for all the m facilities, the lemma follows. ◀

Next we will bound the increase in potential due to online updates to the fractional solution. Recall that for any request R_i, there are several rounds, each comprising multiple iterations, one for every resource in $\overline{R}_i$. Our general plan is the following: we will first bound the increase in potential in a *single iteration* and then bound the total number of iterations performed by the algorithm (overall, for all requests and for all rounds corresponding to a request).

Increase in Potential in a Single Iteration. First, note that a facility S_j might belong to multiple closed prefixes in a single round. Therefore, the value of x_j for partially open facilities S_j and that of $\tilde{L}_j$ for fully open facilities changes from one iteration to another in the same round. To reconcile this inconsistency, we bound the increase of these variables in a single round in the next lemma.

▶ **Lemma 3.** *For any partially open facility S_j, the value of $x_j^{(i)}$ can increase by a multiplicative factor of at most e in a single round. Similarly, for any fully open facility S_j, the value of $A^{\tilde{L}_j}$ can increase by a multiplicative factor of at most 2 in a single round.*

Proof. First, consider a partially open facility S_j. Since there are at most n iterations in a round, the multiplicative factor by which the value of $x_j^{(i)}$ increases in a single round is at most

$$\left(1 + \frac{1}{Nc_j}\right)^n \leq \left(1 + \frac{k}{N}\right)^n \leq e,$$

where the first inequality follows from the fact that $c_j \geq \frac{1}{k}$ for all facilities S_j and the second inequality holds since $N \geq nk$.

Next, consider a fully open facility S_j with virtual congestion $\tilde{L}_j$ at the beginning of the round. The multiplicative factor by which $A^{\tilde{L}_j}$ increases in a single round is at most

$$A^{\frac{\Delta y_{ij}}{t_j}} - 1 = (1 + (A-1))^{\frac{\Delta y_{ij}}{t_j}} - 1 \leq 2(A-1)\frac{\Delta y_{ij}}{t_j} \leq \frac{2(A-1)n}{c_j A^{\tilde{L}_j}(A-1)N} \leq \frac{2nk}{AN} \leq 2,$$

where the first inequality uses the fact that for any $y \geq x \geq 0$,

$$\left(1 + \frac{1}{x}\right)^{1/y} \leq e^{x/y} \leq 1 + \frac{2x}{y} \tag{8}$$

(we call this *local linearization*); the second inequality holds since virtual congestion, and therefore ψ_j, is non-decreasing and there are at most n iterations in a round; the third inequality uses $c_j \geq \frac{1}{k}$ for all facilities S_j and $\tilde{L}_j \geq 1$ for any fully open facility S_j; and the last inequality follows from $N \geq nk$. ◀

The next lemma bounds the increase in potential of the fractional solution in a single iteration.

▶ **Lemma 4.** *The increase in potential in a single iteration for any resource $u \in \overline{R}_i$ is at most $\frac{10A}{N}$.*

Proof. Note that the increase in potential in an iteration can be attributed to two possible sources: increase in cost for partially open facilities in the closed prefix $\widehat{\mathbf{P}}_i(u)$ and increase in virtual congestion of the boundary facility $S_{p_i(u)}$.

First, we bound the increase in cost. Recall that at the beginning of the round,

$$\sum_{j: S_j \in \widehat{\mathbf{P}}_i(u)} x_j^{(i)} = \left(\sum_{j: S_j \in \mathbf{P}_i(u)} x_j^{(i)}\right) + x_{p_i(u)}^{(i)} \leq 1 + 1 = 2.$$

However, the value of $x_j^{(i)}$ increases over the various iterations in the round, and therefore, it is possible that $\sum_{j: S_j \in \widehat{\mathbf{P}}_i(u)} x_j^{(i)} > 2$ at the beginning of the iteration for resource u. Nevertheless, by Lemma 3, we can claim that $\sum_{j: S_j \in \widehat{\mathbf{P}}_i(u)} x_j^{(i)} \leq 2e < 6$.

The increase in potential due to increments in $x_j^{(i)}$ for all partially open facilities $S_j \in \widehat{\mathbf{P}}_i(u)$ is

$$A \sum_{j:S_j \in \widehat{\mathbf{P}}_i(u)} \frac{c_j x_j^{(i)}}{N c_j} = \frac{A}{N} \sum_{j:S_j \in \widehat{\mathbf{P}}_i(u)} x_j^{(i)} < \frac{6A}{N},$$

where the inequality follows from the observation above.

Next, we consider the increase in potential due to the increase in virtual congestion of facility $S_{p_i(u)}$, if it is fully open. If the virtual congestion before the iteration was $\tilde{L}_{p_i(u)}$, then the increase in potential is

$$c_{p_i(u)} A^{\tilde{L}_{p_i(u)}} \left(A^{\frac{\Delta y_{ip_i(u)}}{t_{p_i(u)}}} - 1 \right) \leq 2 \frac{\psi_{p_i(u)} t_{p_i(u)}}{(A-1)} \cdot \frac{2(A-1)\Delta y_{ip_i(u)}}{t_{p_i(u)}} = \frac{4}{N} < \frac{4A}{N},$$

where the first inequality uses local linearization (see Eqn. 8) and Lemma 3. ◀

Total Number of Iterations. Recall that OPT is a a feasible integer solution with cost at most $\mathbf{C}$ and congestion at most 1 on each facility. Let OPT(R_i) denote the facilities used by OPT to satisfy request R_i. An iteration for resource $u \in \overline{R_i}$ is in one of the following two categories:

1. At least one facility in OPT(R_i) is in the prefix $\mathbf{P}_i(u)$.
2. No facility in OPT(R_i) is in the prefix $\mathbf{P}_i(u)$.

The number of iteration of the first category is bounded by the next lemma.

▶ **Lemma 5.** *The total number of iterations of the first category is $O(Nm \log m)$.*

Proof. Let S_{j^*} be a facility in OPT. The number of iterations where S_{j^*} is in the prefix is $O(Nc_{j^*} \log m)$ since:

- x_{j^*} is initialized to $\frac{1}{m}$ in the pre-processing phase.
- $x_{j^*} < 1$ before the last round where x_{j^*} increases. Therefore, by Lemma 3, $x_{j^*} < e$ at the end of the round.
- x_{j^*} increases by a multiplicative factor of $\left(1 + \frac{1}{Nc_{j^*}}\right)$ in every iteration where it belongs to the prefix.

The lemma follows by summing over all facilities $S_{j^*} \in$ OPT. ◀

Now, we focus on iterations of the second category. We partition rounds into ones where $\overline{R_i}$ changes (we call these *dynamic* rounds) and ones where $\overline{R_i}$ does not change (we call these *static* rounds). The number of iterations in dynamic rounds is bounded by the next lemma.

▶ **Lemma 6.** *The total number of iterations in dynamic rounds is $O(N)$.*

Proof. Since $\overline{R_i}$ changes, i.e., loses a resource in any dynamic round, a single request R_i can have at most $|R_i| \leq n$ dynamic rounds. Since there are at most n iterations in each round and at most k requests overall, the lemma follows from $N > kn^2$. ◀

Now, we focus on counting the number of iterations of the second category in static rounds. Recall that for any partially open facility S_j, we set $y_{ij} = \min(2x_j^{(i)}, t_j(x_j^{(i)} - x_j^{(i-1)}))$ at the end of the round. Let $\overline{T}$ be the collection of partially open facilities such that $y_{ij} = 2x_j^{(i)}$ and $T = \mathbf{S} \setminus \overline{T}$ be all the remaining facilities. The next lemma lower bounds the contribution of facilities in T in any iteration of the second category in a static round.

▶ **Lemma 7.** *For any static round and any iteration of the second category for resource $u \in \overline{R}_i$, it holds that*

$$\sum_{j:S_j \in T \cap \widehat{\mathbf{P}}_i(u)} x_j^{(i)} \geq 1/2.$$

Proof. Suppose for some resource u, $\sum_{j:S_j \in T \cap \widehat{\mathbf{P}}_i(u)} x_j^{(i)} < 1/2$. Note that since the iteration for u is of the second category, the boundary facility must be defined and $\sum_{j \in \widehat{\mathbf{P}}_i(u)} x_j^{(i)} \geq 1$. We conclude that $\sum_{j \in \overline{T} \cap \widehat{\mathbf{P}}_i(u)} x_j^{(i)} \geq 1/2$. For every facility $S_j \in \overline{T}$, $y_{ij} = 2x_j^{(i)}$. However, in that case, $\sum_{j \in \overline{T} \cap \widehat{\mathbf{P}}_i(u)} y_{ij} \geq 1$ and the resource u is satisfied at the end of this round. In other words, the round is dynamic, which is a contradiction. ◀

For a resource u, we refer to $\sum_{j:u \in S_j} y_{ij}$ as its *coverage*. We will show in the next lemma that the increase in coverage on resources in $\overline{R}_i$, averaged over the iterations of the second category in a static round, is large. Before stating the lemma, we need to introduce some notation. For any facility S_j, let us partition $\overline{R}_i \cap S_j$ as follows: U_j contains resources that have S_j in their prefix, V_j contains resources that have S_j as the boundary facility, and W_j contains the rest of the resources. Note that prefixes of resources in W_j are filled first, followed by those in V_j, and finally those in U_j.

Now, consider a facility $S_{j^*} \in \mathrm{OPT}(R_i)$. Let $\overline{B}$ be the set of resources in $S_{j^*} \cap \overline{R}_i$ that have iterations of the second category; let $\overline{b} = |\overline{B}|$.

▶ **Lemma 8.** *The total increase in coverage on resources of $\overline{B}$ in a single static round is at least $\frac{\overline{b} \cdot |S_{j^*} \cap \overline{R}_i|}{2N\psi_{j^*}}$.*

Proof. Let $u \in \overline{B}$ and D_u denote the set of resources of $S_{j^*} \cap \overline{R}_i$ whose closed prefixes were filled with or after $\widehat{\mathbf{P}}_i(u)$. Clearly, $D_u \subseteq \overline{\overline{R}}_i$ for all steps of constructing the prefix till the step that filled $\widehat{\mathbf{P}}_i(u)$. Since S_{j^*} was not inserted in $\mathbf{P}_i(u)$, therefore the scaled cost ϕ_j for every facility S_j in the closed prefix of u satisfies

$$\phi_j = \frac{\psi_j}{|U_j \cup V_j|} \leq \frac{\psi_{j^*}}{|D_u|}.$$

For any such facility $S_j \in T \cap \widehat{\mathbf{P}}_i(u)$, the total increase of y_{ij} is at least $\frac{x_j |U_j \cup V_j|}{N\psi_j} \geq \frac{x_j |D_u|}{N\psi_{j^*}}$. Summing over all such facilities S_j and using Lemma 7, we can conclude that the total increase in coverage on u is at least $\frac{|D_u|}{2N\psi_{j^*}}$.

Now, let us order the resources $u \in \overline{B}$ in which $\widehat{\mathbf{P}}_i(u)$ got filled. For the first u in this order, $D_u = S_{j^*} \cap \overline{R}_i$. Each subsequent D_u loses precisely one resource, the one whose prefix was just filled. For the last u in in the order, $D_u = U_{j^*} \cup \{u\}$. Note that the iterations for resources in U_{j^*} are in the first category. Thus, $\overline{B} \subseteq V_{j^*} \cup W_{j^*}$. Let $|U_{j^*}| = p$ and $|V_{j^*} \cup W_{j^*}| = q$. Then,

$$|U_{j^*} \cup V_{j^*} \cup W_{j^*}| = |S_{j^*} \cap \overline{R}_i| = p + q.$$

Adding up the increases in coverage obtained from the above expression,

$$\sum_{u \in \overline{B}} \frac{|D_u|}{2N\psi_{j^*}} = \sum_{i=p+1}^{p+q} \frac{i}{2N\psi_{j^*}} \geq \quad \frac{p(p+q)}{2N\psi_{j^*}} = \frac{\overline{b} \cdot |S_{j^*} \cap \overline{R}_i|}{2N\psi_{j^*}}.$$

◀

We now consider two subcases:

- (a) static rounds where S_{j^*} is partially open, and
- (b) static rounds where S_{j^*} is fully open.

The advantage with subcase (a) is that the value of ψ_{j^*} in the previous lemma depends only on the facility S_{j^*} and not on the state of the algorithm.

▶ **Lemma 9.** *Let S_{j^*} be a facility in* OPT(R_i)*. Then, the number of iterations of the second category in static rounds for resources in $S_{j^*} \cap R_i$, where S_{j^*} is partially open, is $2\left(\frac{c_{j^*}}{t_{j^*}}\right) N \ln n$.*

Proof. Let $z_{j^*} = \sum_{u \in S_{j^*} \cap R_i} \max\left(1 - \sum_{j: u \in S_j} y_{ij}, 0\right)$. By Lemma 8, the decrease in z_{j^*} in any static round comprising $\bar{b}$ iterations is at least $\frac{\bar{b} \cdot |S_{j^*} \cap \overline{R}_i|}{2N(c_{j^*}/t_{j^*})}$. Since z_{j^*} decreases from at most n to 0, it follows that the total number of iterations is

$$\int_{z_{j^*}=n}^{0} 2N \cdot \frac{c_{j^*}}{t_{j^*}} \cdot \frac{dz_{j^*}}{z_{j^*}} = 2N\left(\frac{c_{j^*}}{t_{j^*}}\right) \ln n. \qquad ◀$$

The next corollary follows by summing over all requests R_i and facilities S_{j^*} in OPT.

▶ **Corollary 10.** *The total number of iterations of the second category for resources u in static rounds for requests R_i such that there exists a partially open facility S_{j^*} that is used to satisfy R_i in* OPT *and contains u is at most $O(Nm \log n)$.*

We are left with subcase (b), i.e., when facility S_{j^*} is fully open. Let $\mathbf{L}_{j^*}$ be the virtual congestion on facility S_{j^*} at the end of the algorithm. Then, at any intermediate stage of the algorithm when S_{j^*} was fully open,

$$\psi_{j^*} \le \frac{c_{j^*} A^{\mathbf{L}_{j^*}}(A-1)}{t_{j^*}}.$$

Using the same logic as Lemma 9, we obtain the next lemma.

▶ **Lemma 11.** *Let S_{j^*} be a facility in* OPT(R_i)*. Then, the number of iterations of the second category in static rounds for resources in $S_{j^*} \cap R_i$, where S_{j^*} is fully open, is $2\left(\frac{c_{j^*} A^{\mathbf{L}_{j^*}}(A-1)}{t_{j^*}}\right) N \ln n$.*

Now, note that the congestion on S_{j^*} in OPT is at most 1, i.e. the number of requests served by facility S_{j^*} is at most t_{j^*}. Therefore, we obtain the next corollary.

▶ **Corollary 12.** *The total number of iterations of the second category for resources u in static rounds for requests R_i such that there exists a fully open facility S_{j^*} that is used to satisfy R_i in* OPT *and contains u is at most $2N \ln n(A-1) \sum_{j^*: S_{j^*} \in \text{OPT}} c_{j^*} A^{\mathbf{L}_{j^*}}$.*

Now, we add up all the bounds that we have obtained on the increase of the potential to obtain the next lemma.

▶ **Lemma 13.** *At the end of the algorithm, the final potential $\Gamma_f = O(m \log(mn))$.*

Proof. By summing up over the individual bounds on the number of iterations in the various categories,

$$\Gamma_f = O(m \log(mn)) + 20A(\ln n)(A-1) \sum_{j: S_j \in \text{OPT}} c_j A^{\mathbf{L}_{j^*}} \le O(m \log(mn)) + \frac{1}{2}\Gamma_f,$$

by choosing $A = 1 + \frac{1}{80 \ln n}$. The lemma follows. ◀

The next corollary follows from the definition of the potential function Γ.

▶ **Corollary 14.** *The total cost of the fractional solution is $O(m\log(mn))$ and the maximum congestion on a facility is $O(\log n(\log m + \log\log n))$.*

4 Analysis of Online Randomized Rounding

Recall that the fractional solution maintains the following invariant for any facility S_j and any request R_i:

$$y_{ij} \leq 2x_j^{(i)}. \tag{9}$$

First, we consider red copies of facilities.

▶ **Lemma 15.** *The probability that the red copy of any facility is opened is at most $e^{-\Omega(\alpha)}$.*

Proof. We first consider the scenario where for a resource u, no facility $S_j \in \mathbf{S}(u)$ is opened in the integer solution, i.e., $\sum_{j:u\in S_j} X_j^{(i)} = 0$. Since $y_{ij} \leq 2x_j^{(i)}$ (Eqn. 9) and $\sum_{j:S_j\in\mathbf{S}(u)} y_{ij} \geq 1$, it follows that $\sum_{j:u\in S_j} \alpha x_j^{(i)} \geq \frac{\alpha}{2}$. Therefore, the probability that $\sum_{j:u\in S_j} X_j^{(i)} = 0$ is at most $\prod_{j:u\in S_j}\left(1-\alpha x_j^{(i)}\right) = e^{-\Omega(\alpha)}$ by Eqn. 7.

Next, consider the scenario where $\sum_{j:u\in S_j} X_j^{(i)} \geq 1$ but $\sum_{j:u\in S_j} Y_{ij} = 0$, i.e., even though facilities that contain resource u are open, none of them have been assigned to request R_i. Let $\mathbf{A}_i$ and $\mathbf{B}_i$ respectively denote the set of facilities S_j with $x_j^{(i)} < \frac{1}{\alpha}$ and those with $x_j^{(i)} \geq \frac{1}{\alpha}$. Clearly, all facilities in $\mathbf{B}_i$ are open in the integer solution and some subset of facilities in $\mathbf{A}_i$ is open. We consider two subcases. First, suppose $\sum_{j:S_j\in\mathbf{B}_i,u\in S_j} y_{ij} \geq 1/2$. Then,

$$\sum_{j:S_j\in\mathbf{B}_i,u\in S_j} = \sum_{j:S_j\in\mathbf{B}_i\cap\mathbf{S}(u)} \alpha y_{ij} \geq \frac{\alpha}{2}.$$

Therefore, the probability of $\sum_{j:u\in S_j} Y_{ij} = 0$ is at most $\prod_{j:S_j\in\mathbf{B}_i,u\in S_j}(1-z_{ij}) = e^{-\Omega(\alpha)}$.

Finally, suppose $\sum_{j:S_j\in\mathbf{B}_i,u\in S_j} y_{ij} < 1/2$. Then, $\sum_{j:S_j\in\mathbf{A}_i\cap\mathbf{S}(u)} y_{ij} \geq 1/2$.

In this case, we first estimate the expectation and bound the probability of deviation of random variables z_{ij}. We have

$$\mathbb{E}\left[\sum_{j:S_j\in\mathbf{A}_i,u\in S_j} z_{ij}\right] = \sum_{j:S_j\in\mathbf{A}_i,u\in S_j}\left(\frac{y_{ij}}{2x_j^{(i)}}\right)\mathbb{P}\left[X_j^{(i)}=1\right]$$
$$= \sum_{j:S_j\in\mathbf{A}_i,u\in S_j}\left(\frac{y_{ij}}{2x_j^{(i)}}\right)\alpha x_j^{(i)} = \sum_{j:S_j\in\mathbf{A}_i,u\in S_j}\frac{y_{ij}\alpha}{2} \geq \sum_{j:S_j\in\mathbf{A}_i,u\in S_j}\frac{\alpha}{4}.$$

Since $y_{ij} \leq 2x_j^{(i)}$, we can use Chernoff bounds (see, e.g., [15]) to claim that with probability $1-e^{-\Omega(\alpha)}$,

$$\sum_{j:S_j\in\mathbf{A}_i\cap\mathbf{S}(u)} z_{ij} = \Omega(\alpha). \tag{10}$$

On the other hand, if Eqn. 10 holds, then the probability of $\sum_{j:u\in S_j} Y_{ij} = 0$ is

$$\prod_{j:S_j\in\mathbf{A}_i,u\in S_j}(1-z_{ij}) = e^{-\Omega(\alpha)}.$$ ◀

We choose $\alpha = \Theta(\log(knm))$ and use linearity of expectation over all requests and resources to conclude that the red copies of the facilities can be ignored by incurring an additive $O(1)$ loss in the approximation ratio.

We will now bound the expected cost and congestion of the blue copies of facilities.

▶ **Lemma 16.** *The expected total cost of blue copies of facilities in the integer solution is at most α times the cost of the fractional solution.*

Proof. The proof is an immediate consequence of Eqn. 7 using linearity of expectation. ◀

▶ **Lemma 17.** *With probability $1 - o(1)$, the congestion on every facility in the integer solution is $O(\alpha)$ times their virtual congestion in the fractional solution.*

Proof. We split the congestion on a facility S_j in the integer solution into its congestion from requests in $R_0(j)$ (before S_j is fully open in the fractional solution) and $R_1(j)$ (after S_j is fully open in the fractional solution). By linearity of expectation, the expected congestion due to requests in $R_1(j)$ is at most $\alpha \sum_{i:R_i \in R_1(j)} \frac{y_{ij}}{t_j}$.

On the other hand, the expected congestion due to requests in $R_0(j)$ is at most

$$\sum_{i:R_i \in R_0(j)} \frac{y_{ij}}{2x_j^{(i)} t_j} \leq \sum_{i:R_i \in R_0(j)} \frac{x_j^{(i)} - x_j^{(i-1)}}{2x_j^{(i)}} \leq \int_{1/m}^{1} \frac{dw}{w} = \ln m = O(\alpha),$$

where the last bound follows from the choice of $\alpha = \Theta(\log knm)$. ◀

Using standard techniques (bounding the maximum possible congestion if the above lemma fails, and therefore obtaining a bound on its contribution to the expectation), we can convert the high probability bound on the maximum congestion in the above lemma to the same bound (up to constants) on the expectation of the maximum congestion.

5 Conclusion and Future Work

We have given an algorithm for a generic online covering problem where each individual request comprises a set of elements. The competitive ratio of our algorithm is poly-logarithmic in the input parameters. While such dependence on the number of elements and subsets in the set system is matched by existing lower bounds, it is not clear whether our dependence on the number of requests is necessary. We leave the resolution of this dependence as an open question. Our problem represents a nesting of online and offline covering problems. An intriguing open problem is to obtain a formal algorithmic framework for packing/covering LPs that are revealed online in stages where each stage is an offline packing/covering LP.

Acknowledgement. We thank an anonymous reviewer for suggesting the alternative (and simpler) technique for the Cover-SetReq problem with soft capacities. D. Panigrahi is supported in part by startup funds from Duke University.

References

1 Susanne Albers. Online algorithms: a survey. *Math. Program.*, 97(1-2):3–26, 2003.
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 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.

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

5 Yossi Azar, Umang Bhaskar, Lisa K. Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *SODA*, 2013.

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

7 M. Balazinska, B. Howe, and D. Suciu. Data markets in the cloud: An opportunity for the database community. *PVLDB*, 4(12):1482–1485, 2011.

8 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, New York, NY, USA, 1998.

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

10 Anupam Gupta and Viswanath Nagarajan. Approximating sparse covering integer programs online. In *ICALP (1)*, pages 436–448, 2012.

11 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

12 Simon Korman. On the use of randomization in the online set cover problem. *M.S. thesis, Weizmann Institute of Science*, 2005.

13 Paraschos Koutris, Prasang Upadhyaya, Magdalena Balazinska, Bill Howe, and Dan Suciu. Querymarket demonstration: Pricing for online data markets. *PVLDB*, 5(12):1962–1965, 2012.

14 Chao Li and Gerome Miklau. Pricing aggregate queries in a data marketplace. In *WebDB*, pages 19–24, 2012.

15 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.

16 Prabhakar Raghavan and Clark D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

A A simpler algorithm for the Cover-SetReq problem with soft capacities

Here we describe a simpler algorithm for the Cover-SetReq problem with soft capacities that follows from previous work. The algorithm follows by reducing the linear program for the Cover-SetReq problem with soft capacities which has mixed packing and covering constraints to one with just covering constraints. An online solution for covering program can be constructed using existing techniques [9, 10].

Note that the integer programming formulation of the Cover-SetReq problem with soft capacities is as follows:

$$
\textbf{(P1)}\ \text{Minimize} \sum_{j=1}^{m} c_j x_j \qquad \text{subject to}
$$

$$
\begin{aligned}
\sum_{j: u \in S_j} y_{ij} &\geq 1 & \forall i \in [k], u \in R_i \\
y_{ij} &\leq x_j & \forall i \in [k], j \in [m] \\
\sum_{i=1}^{k} y_{ij} &\leq x_j t_j & \forall j \in [m] \\
y_{ij} \in \{0,1\}, x_j &\in \mathbb{N} & \forall i \in [k], j \in [m]
\end{aligned}
$$

First we will show that the same problem can be solved using the following formulation while losing only a constant factor of 2 in the objective. This is based on an observation for Jain and Vazirani [11] along with some further ideas to obtain a covering LP.

$$\textbf{(P2)}\ \text{Minimize}\ \sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{k}\sum_{j=1}^{m} \frac{c_j}{t_j}\cdot y_{ij} \qquad \text{subject to}$$
$$\sum_{j\in J} x_j + \sum_{j\in\{j:u\in S_j\}\setminus J} y_{ij} \geq 1 \quad \forall i\in[k], u\in R_i, J\subseteq\{j: u\in S_j\}$$
$$y_{ij}\geq 0, x_j \geq 0 \qquad \forall i\in[k], j\in[m]$$

▶ **Lemma 18.** *The program (P2) is a linear relaxation of (P1) with a factor* 2 *loss in objective. In particular,*

- $\text{OPT}(P2) \leq 2\cdot\text{OPT}(P1)$ *where* $\text{OPT}(P)$ *denotes the value of the optimal feasible solution.*
- *Any feasible solution* (x', y') *of (P2) can be mapped to a solution of the program (P1) with the same value of the objective provided it satisfies* $y_{ij}\leq x_j$ *for all* $1\leq i\leq k, 1\leq j\leq m$.

Proof. Consider the optimal feasible solution (x, y) of (P1). Define (x', y') as follows:

$$x'_j = min\{1, x_j\} \qquad \forall j\in[m]$$
$$y'_{ij} = y_{ij} \quad \forall j\in[m], i\in[k]$$

First we will show that $y'_{ij}\leq x'_j$. Since (x, y) is an optimal feasible solution, $y_{ij}\leq 1$. Then by the definition of x'_j, $y'_{ij} = y_{ij}\leq min\{1, x_j\} = x'_j$. It then follows that the first constraint in the LP (P2) holds since y_{ij} satisfy the first inequality in the program (P1). Next we bound the objective. Trivially, $\sum_{j=1}^{m} c_j x'_j \leq \sum_{j=1}^{m} c_j x_j = \text{OPT}(P1)$. Finally,

$$\sum_{j=1}^{m}\sum_{i=1}^{k}\frac{c_j}{t_j}y'_{ij} = \sum_{j=1}^{m} c_j\sum_{i=1}^{k} y_{ij}/t_j \leq \sum_{j=1}^{m} c_j x_j = \text{OPT}(P2).$$

It then follows that

$$\text{OPT}(P1)\leq \sum_{j=1}^{m} c_j x'_j + \sum_{j=1}^{m}\sum_{i=1}^{k}\frac{c_j}{t_j}y_{ij} \leq 2\text{OPT}(P1).$$

Next consider a feasible solution (x', y') of (P2) with $y_{ij}\leq x_j$ for all $1\leq i\leq k, 1\leq j\leq m$. Construct a solution (x, y) of (P1) as follows:

$$\forall 1\leq j\leq m \quad x_j = x'_j + \textstyle\sum_{i=1}^{k}\frac{y'_{ij}}{t_j}$$
$$\forall 1\leq j\leq m, 1\leq i\leq k \qquad y_{ij} = y'_{ij}$$

The first constraint of (P1) is obviously true. Since $y'_{ij}\leq x'_j$, it follows that $y_{ij}\leq x'_j\leq x_j$. Moreover,

$$\sum_{i=1}^{k} y_{ij}/t_j = \sum_{i=1}^{k} y'_{ij}/t_j \leq x_j.$$

Finally,

$$\sum_{j=1}^{m} c_j x_j = \sum_{j=1}^{m} c_j\left(x'_j + \sum_{i=1}^{k} y'_{ij}\right) = \sum_{j=1}^{m} c_j x'_j + \sum_{j=1}^{m}\sum_{k=1}^{n}\frac{c_j}{t_j}x_j. \qquad ◀$$

Finally note that the requirement $y'_{ij} \leq x'_j$ on a solution (x', y') of the program (P2) is without loss of generality. Any solution that violates this constraint can be fixed by lowering the value of y'_{ij} to x'_j. This still maintains all of the constraints while lowering the objective value.

We have thus obtained a covering linear program, any solution to which can be mapped to a feasible solution of the original IP with only a factor 2 loss in the objective. It is possible to construct a solution to program (P2) in an online manner using the techniques of Buchbinder and Naor [9]. (See also Gupta and Nagarajan [10].) The resulting solution can then be rounded using our randomized rounding procedure.