Approximating Sparse Covering Integer Programs Online

Anupam Gupta
Computer Science Department, Carnegie Mellon University
email: anupamg@cs.cmu.edu

Viswanath Nagarajan
IBM T.J. Watson Research Center
email: viswanath@us.ibm.com

A *covering integer program* (CIP) is a mathematical program of the form:

$$\min\{c^\top \mathbf{x} \mid A\mathbf{x} \geq \mathbf{1},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{u},\ \mathbf{x} \in \mathbb{Z}^n\}$$

where $A \in R_{\geq 0}^{m \times n}, c, u \in \mathbb{R}_{\geq 0}^n$. In the online setting, the constraints (i.e., the rows of the constraint matrix A) arrive over time, and the algorithm can only increase the coordinates of $\mathbf{x}$ to maintain feasibility. As an intermediate step, we consider solving the *covering linear program* (CLP) online, where the requirement $\mathbf{x} \in \mathbb{Z}^n$ is replaced by $\mathbf{x} \in \mathbb{R}^n$.

Our main results are (a) an $O(\log k)$-competitive online algorithm for solving the CLP, and (b) an $O(\log k \cdot \log \ell)$-competitive randomized online algorithm for solving the CIP. Here $k \leq n$ and $\ell \leq m$ respectively denote the maximum number of non-zero entries in any row and column of the constraint matrix A. Our algorithm is based on the online primal-dual paradigm, where a novel ingredient is to allow dual variables to increase and decrease throughout the course of the algorithm. By a result of Feige and Korman, this result is the best possible for polynomial-time online algorithms, even in the special case of set cover (where $A \in \{0,1\}^{m \times n}$ and $c, u \in \{0,1\}^n$).

Key words: covering integer programs; linear programming; online algorithms

MSC2000 Subject Classification: Primary: 68W07; Secondary: 90C05

OR/MS subject classification: Primary: Analysis of algorithms; Secondary: Linear Programming

---

1. Introduction Covering Integer Programs (CIPs) have long been studied as a very general framework which captures a wide variety of natural problems. CIPs are mathematical programs of the following form:

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^n c_i x_i && && \text{(IP1)} \\ \text{subject to:} \quad & \textstyle\sum_{i=1}^n a_{ij} x_i \geq 1 && \forall j \in [m], && (1.1) \\ & 0 \leq x_i \leq u_i && \forall i \in [n], && (1.2) \\ & x \in \mathbb{Z}^n. && && (1.3) \end{aligned}$$

All the entries a_{ij}, c_i, and u_i are non-negative. Throughout the paper, we use the notation $[n] := \{1, 2, \ldots, n\}$ and $[m] := \{1, 2, \ldots, m\}$. The *constraint matrix* is denoted $A = (a_{ij})_{i \in [n], j \in [m]}$. We define k to be the *row sparsity* of A, i.e., the maximum number of non-zeroes in any constraint $j \in [m]$. For each row $j \in [m]$ let $T_j \subseteq [n]$ denote its non-zero columns; we say that the variables indexed by T_j "appear in" constraint j. Let ℓ denote the *column sparsity* of A, i.e., the maximum number of constraints that any variable $i \in [n]$ appears in. Dropping the integrality constraint (1.3) gives us a covering linear program (CLP).

In this paper we study the *online* version of these problems, where the constraints $j \in [m]$ arrive over time, and we are required to maintain a monotone (i.e., non-decreasing) feasible solution $\mathbf{x}$ at each point in time. Our main results are (a) an $O(\log k)$-competitive algorithm for solving CLPs online, and (b) an $O(\log k \cdot \log \ell)$-competitive randomized online algorithm for CIPs. In settings where $k \ll n$ or $\ell \ll m$ our results give a significant improvement over the previous best bounds of $O(\log n)$ for CLPs [8], and $O(\log n \cdot \log m)$ for CIPs that can be inferred by rounding these LP solutions. Indeed, the underlying constraint matrix in many applications is *sparse*, and hence it is of interest to obtain such stronger guarantees in terms of sparsity parameters. Analyzing performance guarantees for covering/packing integer programs in terms of row (k) and column (ℓ) sparsity has received much attention in the offline setting, e.g. [15, 17, 11, 14, 6]. This paper obtains tight bounds in terms of these parameters for *online* covering integer programs.

Our Techniques. Our algorithms use the online primal-dual framework of Buchbinder and Naor [7]. To solve the covering LP, we give an algorithm which monotonically raises the primal, but *both raises and lowers the dual variables* over the course of its execution. This is unlike typical applications of the online primal-dual

approach, where both primal and dual variables are only increased. This approach of lowering duals is crucial for our bound of $O(\log k)$, since otherwise there is an $\Omega(\log n)$ primal-dual gap even for $k = 1$.

The algorithm for covering IP solves the LP relaxation and then rounds it. It is well-known that the natural LP relaxation is too weak: so we extend our online CLP algorithm to also handle Knapsack Cover (KC) inequalities from [9]. This step has an $O(\log k)$-competitive ratio. Then, to obtain an integer solution, we adapt the method of *randomized rounding with alterations* to the online setting. We note that direct randomized rounding as in [1] results in a worse $O(\log m)$ overhead.

Related Work. The powerful online primal-dual framework has been used to give algorithms for set cover [1], graph connectivity and cut problems [2], caching [19, 4, 5], packing/covering IPs [8], and many more problems. This framework usually consists of two steps: obtaining a fractional solution (to an LP relaxation) online, and rounding the fractional solution online to an integral solution. See the monograph of Buchbinder and Naor [7] for a lucid survey of this area.

In most applications of this framework, the fractional online algorithm raises both primal and dual variables monotonically, and the competitive ratio is given by the primal to dual ratio. For CLPs, Buchbinder and Naor [8] showed that if we increase dual variables monotonically, the primal-dual gap can be $\Omega(\log \frac{a_{max}}{a_{min}})$. In order to obtain an $O(\log n)$-competitive ratio, they used a *guess-and-double* framework [8, Theorem 4.1] that changes duals in a partly non-monotone manner as follows:

> *The algorithm proceeds in phases, where each phase r corresponds to the primal value being roughly 2^r. Within a phase the primal and dual are raised monotonically. But the algorithm resets duals to zero at the beginning of each phase—this is the only form of dual reduction.*

For the special case of fractional set cover (where $A \in \{0,1\}^{m \times n}$), they get an improved $O(\log k)$-competitive ratio using this guess-and-double framework [8, Section 5.1]. However, we show in Section 2.2 that such dual update processes do not extend to obtain an $o(\log n)$ ratio for general CLPs. Our algorithm instead reduces the dual variables in a more continuous manner throughout the algorithm, which leads to an $O(\log k)$-competitive ratio for general CLPs.

Other online algorithms: Koufogiannakis and Young [13] gave a k-competitive deterministic online algorithm for CIPs based on a greedy approach; their result holds for a more general class of constraints and for submodular objectives. Our $O(\log k \log \ell)$ approximation is incomparable to this result. Feige and Korman [12] showed that no randomized *polynomial-time* online algorithm can achieve a competitive ratio better than $O(\log k \log \ell)$, even for set cover (where all entries are in $\{0,1\}$).

Offline algorithms. CLPs can be solved optimally offline in polynomial time. For CIPs in the absence of variable upper bounds, randomized rounding gives an $O(\log m)$-approximation ratio. Srinivasan [15] gave an improved algorithm using the FKG inequality, where the approximation ratio depended on the optimal LP value. Later, Srinivasan [16] also used the method of alterations in context of CIPs and gave an RNC algorithm achieving the bounds of [15]. An $O(\log \ell)$-approximation algorithm for CIPs (no upper bounds) was obtained in [17] using the Lovász Local Lemma. Using KC-inequalities and the algorithm from [17], Kolliopoulos and Young [11] gave an $O(\log \ell)$-approximation algorithm for CIPs with variable upper bounds. Our algorithm matches this $O(\log \ell)$ loss even in the online setting. Finally, the knapsack-cover (KC) inequalities were introduced by Carr et al. [9] to reduce the integrality gap for CIPs. These were used in [11, 10], and also in an online context by [5] for the generalized caching problem.

Outline. In Section 2 we provide a brief introduction to the online primal-dual approach, and discuss why previously used updates are not adequate to obtain a competitive ratio for CLPs that is independent of n. In Section 3 we present an $O(\log k)$-competitive online algorithm for a special case of covering linear programs, where there are no variable upper-bounds. This introduces the main ideas in our primal-dual updates. Then, in Section 4.1 we extend this algorithm to general CLPs with additional knapsack-cover inequalities. Finally, in Section 4.2 we give an online rounding algorithm that converts fractional solutions to integral solutions for covering integer programs.

2. The Online Primal-Dual Approach

In this section, we first recall the online primal-dual approach as applied to the unweighted fractional set cover problem. Then we discuss why previously known primal-dual updates are insufficient to obtain an $O(\log k)$ bound for general covering linear programs.

2.1 Unweighted fractional set cover Consider the special case of CLPs where all entries a_{ij} are $\{0,1\}$, all costs c_i are unit and there are no variable upper-bounds. That is,

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{n} x_i \\ \text{subject to:} \quad & \textstyle\sum_{i \in T_j} x_i \geq 1 \qquad \forall j \in [m], \\ & x \geq \mathbf{0} \end{aligned}$$

Above, $T_j \subseteq [n]$ is some subset for each constraint $j \in [m]$. Notice that this is precisely the fractional set cover problem (sets correspond to variables and elements correspond to constraints) where all sets have unit cost. The dual linear program is the following packing problem:

$$\begin{aligned} \max \quad & \textstyle\sum_{j=1}^{m} y_j \\ \text{subject to:} \quad & \textstyle\sum_{j: i \in T_j} y_j \leq 1 \qquad \forall i \in [n], \\ & y \geq \mathbf{0} \end{aligned}$$

We now apply the online primal-dual method to obtain an $O(\log n)$-competitive online algorithm for the fractional set cover problem, where constraints $\sum_{i \in T_j} x_i \geq 1$ arrive over time and the variables x may only be increased. We will maintain a solution pair consisting of primal x and dual y. Initially, each primal variable $x_i = \frac{1}{n}$, and each dual variable $y_j = 0$. When the j^{th} constraint arrives, perform the following updates repeatedly as long as $\sum_{i \in T_j} x_i < 1$:

$$x_i^{new} \leftarrow 2 \cdot x_i^{old}, \quad \forall i \in T_j \qquad \text{and} \qquad y_j^{new} \leftarrow y_j^{old} + 1.$$

Notice that the primal solution x is always feasible after the updates, and both primal and dual variables are only increased. We will now show that:

P1. In each update step, the increase in the primal cost is at most the increase in the dual objective.

P2. The dual solution is $\beta = 1 + \log_2 n$ approximately feasible, i.e. dual solution y/β is feasible.

For property (P1), note that the primal increase in any update is $\sum_{i \in T_j} \left(x_i^{new} - x_i^{old}\right) = \sum_{i \in T_j} x_i^{old} < 1$, where the last inequality is the condition required to perform an update; and the dual increase in any update is one. To see property (P2), notice that the updates ensure that

$$x_i \;=\; \frac{1}{n} \cdot 2^{\sum_{j: i \in T_j} y_j}, \qquad \forall\, i \in [n].$$

Moreover, no primal variable is increased once its value is at least one: this implies that $\max_{i \in [n]} x_i \leq 2$. Thus $\sum_{j: i \in T_j} y_j \leq 1 + \log_2 n = \beta$ for all $i \in [n]$, which proves (P2).

We can now prove the $O(\log n)$-competitive ratio. Let Opt denote the optimal primal cost; note that $\mathsf{Opt} \geq 1$. Since the initial primal cost is one, using (P1), the final primal cost is $\sum_{i=1}^{n} x_i \leq 1 + \sum_{j=1}^{m} y_j \leq \mathsf{Opt} + \sum_{j=1}^{m} y_j$. By weak duality, any feasible dual solution provides a lower bound to Opt; so by (P2), $\sum_{j=1}^{m} y_j \leq \beta \cdot \mathsf{Opt}$. Thus the cost of the online solution is at most $(1 + \beta)\mathsf{Opt}$.

2.2 Limitations of existing primal-dual updates For general covering linear programs, Buchbinder and Naor [8, Lemma 3.1] showed that if we maintain monotone duals (as in the above fractional set cover example) then the primal-dual gap may be as large as $\Omega(\log \frac{a_{max}}{a_{min}})$. In order to get around this issue and obtain an $O(\log n)$ competitive ratio for general CLPs, [8, Theorem 4.1] used a guess-and-double framework which uses duals in a partly non-monotone manner. However, as we show below, this scheme does not suffice to obtain any primal-dual gap independent of n, even when $k = 1$.

The *guess-and-double* scheme proceeds in phases, and within each phase it maintains monotone primal as well as dual variables. But when the phase changes, the scheme resets all dual values to zero and starts afresh; this is the only allowed dual reduction. To maintain an approximately feasible dual and bounded primal-dual ratio, this scheme is allowed to change phases (and reset duals) only when the primal cost increases by (say)

a factor of two. Upon arrival of the first constraint ($\sum_{i\in T_1} a_{i1}x_i \geq 1$), the scheme produces a lower bound $\alpha_1 = \min_{i\in T_1} c_i/a_{i1}$ on the optimal value and begins its first phase. In the r^{th} phase it is assumed that α_r is the optimal value until the primal cost exceeds α_r; at this point the scheme sets $\alpha_{r+1} = 2\cdot\alpha_r$ and enters phase $r+1$. A competitive ratio of $O(\beta)$ is proven via this scheme by showing that after each phase r, the total primal cost is at most β times the total dual value (added over all phases up to r).

LEMMA 2.1 *Any online algorithm using the guess-and-double framework for covering LPs (even with $k=1$) incurs an unbounded primal to dual ratio.*

PROOF. It suffices to show that for every $\rho > 2$, there exists an instance of the online covering LP with $k=1$ where any algorithm using the guess-and-double framework incurs a primal to dual ratio of $\Omega(\rho)$. Our instances will have all costs being one, so the primal objective is just $\sum_{i=1}^{n} x_i$. Since $k=1$, each constraint will be of the form $x_i \geq b$ for some $i \in [n]$ and $b > 0$. The first constraint is $x_1 \geq \rho^{\rho+2}$. So $\alpha_1 = \rho^{\rho+2}$ in the guess-and-double scheme. In each phase r, constraints appear for a completely new set of variables $x_{r,1}, x_{r,2}, \ldots$ as follows. Initialize $j \leftarrow 1$.

> **Sequence** $I(r,j)$ **:** Constraints of the form $x_{r,j} \geq \rho^h$ with dual variable $y_{r,j}(h)$ appear for $h = 1, 2, \ldots$, until the first time that the algorithm sets dual value $y_{r,j}(h) < \rho^{h-1}$.

At this point we move on to the next variable $x_{r,j+1}$, i.e. set $j \leftarrow j+1$ and repeat the sequence $I(r,j)$. Also, the entire phase r ends when the sum of variables in this phase exceeds α_r, at which point we abort the current sequence $I(r,j)$ and enter phase $r+1$ with $\alpha_{r+1} = 2\cdot\alpha_r$.

Suppose q variables are used in phase r. Let $h_1, \ldots, h_q$ denote the number of constraints produced in $I(r,1), \ldots, I(r,q)$ respectively. Note that the dual variables in this phase are $\bigcup_{j\in[q]}\{y_{r,j}(h) : 1 \leq h \leq h_j\}$, dual constraints are $\sum_h y_{r,j}(h)/\rho^h \leq 1$ for all $j \in [q]$, and dual objective is $\sum_{j\in[q]}\sum_h y_{r,j}(h)$.

Claim 2.1 *For all $j \in [q]$, $h_j \leq \rho + 1$.*

PROOF. Fix any $j \in [q]$; the dual constraint corresponding to variable $x_{r,j}$ reads $\sum_h y_{r,j}(h)/\rho^h \leq 1$. By definition of the sequence $I(r,j)$, for all $1 \leq h < h_j$ the dual value $y_{r,j}(h) \geq \rho^{h-1}$. Note that duals in a single phase are monotone– so at the end of sequence $I(r,j)$ we have:

$$1 \quad \geq \quad \sum_h y_{r,j}(h)/\rho^h \quad \geq \quad \sum_{h=1}^{h_j-1} \rho^{h-1}/\rho^h \quad = \quad \frac{h_j - 1}{\rho}$$

The first inequality is the dual constraint for $x_{r,j}$ and the second uses the lower bound on the first $h_j - 1$ dual values. □

Note that the primal increase in any completed phase r is:

$$P(r) \quad \geq \quad \max\left\{\sum_{j\in[q]} \rho^{h_j},\ \alpha_r\right\} \tag{2.4}$$

The first term above is due to the fact that we see new variables $\{x_{r,1}, \cdots, x_{r,q}\}$ in phase r, and constraints $x_{r,j} \geq \rho^{h_j}$ for $j \in [q]$. The second term is by the definition of when a phase ends.

The next claim shows that the dual increase can only be a small fraction of the primal.

Claim 2.2 *The total dual increase in any completed phase r is at most $\frac{4}{\rho}\cdot P(r)$.*

PROOF. Consider any primal variable $x_{r,j}$, and its dual constraint $\sum_{h=1}^{h_j} y_{r,j}(h)/\rho^h \leq 1$. Clearly the maximum dual value achievable from these dual variables $\sum_{h=1}^{h_j} y_{r,j}(h) \leq \rho^{h_j}$.

Now consider $j \leq q-1$; the sequence $I(r,j)$ was ended due to $y_{r,j}(h_j) < \rho^{h_j-1}$. Also by the dual constraint, $y_{r,j}(h) \leq \rho^h$ for all $1 \leq h \leq h_j - 1$. Thus:

$$\sum_{h=1}^{h_j} y_{r,j}(h) \quad \leq \quad \rho^{h_j-1} + \sum_{h=1}^{h_j-1} \rho^h \quad \leq \quad \rho^{h_j-1}\cdot\left(1 + \frac{1}{1-1/\rho}\right) \quad \leq \quad 3\cdot\rho^{h_j-1},$$

where the last inequality uses $\rho \geq 2$. We now obtain that the total dual value in phase r:

$$\begin{aligned}\sum_{j=1}^{q}\sum_{h=1}^{h_j} y_{r,j}(h) \quad &\leq \quad \rho^{h_q} + \sum_{j=1}^{q-1}\sum_{h=1}^{h_j} y_{r,j}(h) \quad \leq \quad \rho^{h_q} + \sum_{j=1}^{q-1} 3\cdot\rho^{h_j-1} \\ &\leq_{(2.4)} \quad \rho^{h_q} + \frac{3}{\rho}\cdot P(r) \quad \leq_{\text{Claim 2.1}} \quad \rho^{\rho+1} + \frac{3}{\rho}\cdot P(r) \\ &\leq \quad \frac{\alpha_r}{\rho} + \frac{3}{\rho}\cdot P(r) \quad \leq_{(2.4)} \quad \frac{4}{\rho}\cdot P(r).\end{aligned}$$

The second-last inequality is because $\alpha_r \geq \alpha_1 = \rho^{\rho+2}$. This completes the proof. □

Using Claim 2.2, it follows that for the input sequence constructed above, the total dual value accrued $\sum_r \left(\sum_{j,h} y_{r,j}(h)\right)$ is at most $4/\rho$ times the primal cost $\sum_r P(r)$. □

This lemma shows that using just the dual reductions allowed within a guess-and-double framework is insufficient to prove a primal-dual ratio independent of n. Instead, the online algorithms presented in the following sections perform more sophisticated dual reduction steps to get $O(\log k)$-competitiveness.

3. An Algorithm for a Special Class for Covering LPs In this section, we consider CLPs without upper bounds on the variables:

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{n} c_i x_i \\ \text{subject to:} \quad & \textstyle\sum_{i=1}^{n} a_{ij} x_i \geq 1 \qquad \forall j \in [m], \\ & x \geq \mathbf{0} \end{aligned}$$

and give an $O(\log k)$-competitive deterministic online algorithm for solving such LPs, where k is an upper bound on the row-sparsity of $A = (a_{ij})$. The constraints are indexed in the order in which they arrive. The dual is the packing linear program:

$$\begin{aligned} \max \quad & \textstyle\sum_{j=1}^{m} y_j \\ \text{subject to:} \quad & \textstyle\sum_{j=1}^{m} a_{ij} y_j \leq c_i \qquad \forall i \in [n], \\ & y \geq \mathbf{0} \end{aligned}$$

We assume that c_i's are strictly positive for all i, else we can drop all constraints containing variable i.

In the online algorithm, we want a solution pair (x, y), where we monotonically increase the value of x, but the dual variables (that are only used in the analysis) can move up or down as needed. We want a feasible primal, and an approximately feasible dual. The primal update step is the following:

Algorithm 3.1 Online CLP without box constraints.

When constraint h (i.e., $\sum_i a_{ih} x_i \geq 1$) arrives,

1: define $d_{ih} = \frac{c_i}{a_{ih}}$ for all $i \in [n]$, and $d_{m(h)} = \min_i d_{ih} = \min_{i \in T_h} d_{ih}$.
2: **while** $\sum_i a_{ih} x_i < 1$ **do**
3: update the x's by

$$x_i^{new} \leftarrow \left(1 + \frac{d_{m(h)}}{d_{ih}}\right) x_i^{old} \quad + \quad \frac{1}{k \cdot a_{ih}} \frac{d_{m(h)}}{d_{ih}}, \qquad \forall i \in T_h.$$

4: **end while**
5: let t_h be the number of times the update step is performed for constraint h.

As stated, the algorithm assumes we know k, but this is not required. We can start with the estimate $k = 2$ and increase it any time we see a constraint with more variables than our current estimate. Since this estimate for k only increases over time, the analysis below will go through unchanged. (We can assume that k is a power of 2, which makes $\log k$ an integer. Moreover, we assume that $k \geq 2$.)

LEMMA 3.1 *Upon arrival of primal constraint h, the number of primal updates $t_h \leq 2 \log k$.*

PROOF. Fix some h, and consider the value i^* for which $d_{i^*h} = d_{m(h)}$. In each iteration of the while-loop, the variable $x_{i^*} \leftarrow 2x_{i^*} + 1/(k \cdot a_{i^*h})$; hence after t iterations, its value will be at least $(2^t - 1)/(k \cdot a_{i^*h})$. So if we do $2\log k$ updates, this variable alone will satisfy the h^{th} constraint. □

LEMMA 3.2 *Upon arrival of primal constraint h, the total increase in the primal cost is at most $2\,t_h\,d_{m(h)}$.*

PROOF. Consider a single update step that modifies primal variables from x^{old} to x^{new}. In this step, the increase in each variable $i \in T_h$ is $\frac{d_{m(h)}}{d_{ih}} \cdot x_i^{old} + \frac{1}{k \cdot a_{ih}} \frac{d_{m(h)}}{d_{ih}}$. So the increase in the primal objective is:

$$\sum_{i \in T_h} c_i \cdot \left[\frac{d_{m(h)}}{d_{ih}} \cdot x_i^{old} + \frac{1}{k \cdot a_{ih}} \frac{d_{m(h)}}{d_{ih}} \right] = d_{m(h)} \sum_{i \in T_h} a_{ih} \cdot x_i^{old} + d_{m(h)} \cdot \frac{|T_h|}{k} \leq 2 \cdot d_{m(h)}$$

The inequality uses $|T_h| \leq k$ and $\sum_{i \in T_h} a_{ih} \cdot x_i^{old} \leq 1$ which is the reason an update was performed. The lemma now follows since t_h is the number of update steps. □

To show approximate optimality, we want to change the dual variables so that the dual increase is (approximately) the primal increase, and so that the dual remains (approximately) feasible.

As noted in Section 2.2, we cannot achieve an $O(\log k)$-competitive ratio for CLPs using previous approaches of maintaining monotone duals or via the guess-and-double framework. The difficult case (as in the instances of Lemma 2.1) is when variables have exponentially increasing coefficients in different constraints. Indeed, [8] showed that monotone duals suffice to obtain an $O(\log \frac{a_{max}}{a_{min}})$-competitive ratio, and the guess-and-double framework leads to an $O(\log k)$-competitive ratio for the special case of CLP when all $a_{ij} \in \{0, 1\}$.

Upon arrival of a primal constraint, to ensure adequate dual increase we raise the newly arriving dual variable; and to maintain approximate dual feasibility we also decrease the "first few" dual variables in each dual constraint where the new dual variable appears.

For the h^{th} primal constraint, let $d_{ih}, d_{m(h)}, t_h$ be given by the primal update process.

(a) Set $y_h \leftarrow d_{m(h)} \cdot t_h$.
(b) For each $i \in T_h$, do the following for dual constraint $\sum_j a_{ij} y_j \leq c_i$:
  (i) If $\sum_{j<h} a_{ij} y_j \leq (10 \log k)\, c_i$, do nothing; else
  (ii) Let $k_i < h$ be the largest index such that $\sum_{j \leq k_i} a_{ij} y_j \leq (5 \log k)\, c_i$.
  Let $P_i = \{j \leq k_i \mid T_j \ni i\}$ be the indices of these first few dual variables that are active in the i^{th} dual constraint. For all $j \in P_i$,

$$y_j^{new} \leftarrow \left(1 - \frac{d_{m(h)}}{d_{ih}}\right) \cdot y_j^{old}.$$

Observe that the dual update process starts each dual variable y_j off at some value $d_{m(j)} t_j$ and subsequently *only decreases* this dual variable, and that the dual variables remain non-negative.

LEMMA 3.3 *When primal constraint h arrives, the left-hand-side of each dual constraint i increases due to the variable y_h by $a_{ih} \cdot d_{m(h)} \cdot t_h \leq (2 \log k)\, c_i$.*

PROOF. We set the initial value of the dual variable y_h to $d_{m(h)} \cdot t_h$. By Lemma 3.1, $t_h \leq 2 \log k$. By definition, $d_{m(h)} \leq c_i / a_{ih}$. Hence, for any $i \in T_h$, the increase in the left-hand-side of dual constraint i is at most $a_{ih} \cdot (2 \log k)\, (c_i / a_{ih}) = (2 \log k)\, c_i$. For the remaining $i' \in [n] \setminus T_h$, there is no increase in dual constraint i'. This proves the lemma. □

LEMMA 3.4 *When primal constraint h arrives, if the dual update reaches step b(ii) for some $i \in T_h$, then k_i is well-defined and the set P_i is non-empty; moreover, $\frac{\sum_{j \in P_i} a_{ij} y_j}{c_i \log k} \in [3, 5]$.*

PROOF. For each $j < h$ we have $y_j \leq 2 \log k \cdot d_{m(j)}$, since dual variable y_j was initialized to $t_j d_{m(j)} \leq 2 \log k \cdot d_{m(j)}$ (by Lemma 3.1) and subsequently never increased. So $a_{ij} \cdot y_j \leq 2 \log k \cdot d_{m(j)} \cdot a_{ij} \leq 2 \log k \cdot c_i$, using $d_{m(j)} \leq d_{ij} = c_i / a_{ij}$. If the dual update reaches step b(ii) then we have $\sum_{j<h} a_{ij} y_j > (10 \log k)\, c_i$, but each term $j < h$ contributes at most $2 \log k \cdot c_i$. So k_i is well-defined, and P_i is non-empty. Moreover, by the choice of k_i, we have $\sum_{j \leq k_i + 1} a_{ij} y_j > (5 \log k)\, c_i$, so $\sum_{j \leq k_i} a_{ij} y_j > (5 \log k)\, c_i - a_{i,k_i+1} \cdot y_{k_i+1} \geq (3 \log k) \cdot c_i$. Thus $3 \log k \cdot c_i \leq \sum_{j \in P_i} a_{ij} y_j \leq 5 \log k \cdot c_i$ as claimed. □

LEMMA 3.5 *When primal constraint h arrives, after any dual update step, each dual constraint i satisfies $\sum_j a_{ij} y_j \le (12 \log k)\, c_i$. Hence the dual solution is always $(12 \log k)$-approximately feasible.*

PROOF. Consider the dual update process when the primal constraint h arrives, and look at any dual constraint $i \in T_h$ (the other dual constraints are unaffected). If case b(i) happens, then by Lemma 3.3 the left-hand-side of the constraint will be at most $(12 \log k)\, c_i$. Else, case b(ii) happens. Each y_j for $j \in P_i$ decreases by $y_j \cdot d_{m(h)}/d_{ih}$, and so the decrease in $\sum_{j \in P_i} a_{ij} y_j$ is at least $\sum_{j \in P_i} a_{ij} y_j \cdot (d_{m(h)}/d_{ih})$. Using Lemma 3.4, this is at least

$$\frac{d_{m(h)}}{d_{ih}} \cdot c_i\,(3 \log k) \quad = \quad \frac{d_{m(h)}}{c_i/a_{ih}} \cdot c_i\,(3 \log k) \quad = \quad d_{m(h)} \cdot a_{ih} \cdot (3 \log k).$$

But since the increase due to y_h is at most $a_{ih} \cdot d_{m(h)}\, t_h \le a_{ih} \cdot d_{m(h)} \cdot (2 \log k)$, there is no net increase in the left-hand-side, so it remains at most $(12 \log k)\, c_i$. □

LEMMA 3.6 *When primal constraint h arrives, the net increase in the dual value is at least $\frac{1}{2}\, d_{m(h)} \cdot t_h$.*

PROOF. The increase in the dual value due to y_h itself is $d_{m(h)} \cdot t_h$. What about the decrease in the other y_j's? These decreases could happen due to any of the k dual constraints $i \in T_h$, so let us focus on one such dual constraint i, which reads $\sum_{j : T_j \ni i} a_{ij} y_j \le c_i$. Now for $j < h$, define $\gamma_{ij} := \frac{y_j}{t_j\, d_{ij}}$. Since y_j was initially set to $t_j\, d_{m(j)} \le t_j\, d_{ij}$ and subsequently never increased, we know that at this point in time,

$$\gamma_{ij} \quad \le \quad \frac{d_{m(j)}}{d_{ij}} \quad \le \quad 1. \tag{3.5}$$

The following claim, whose proof appears after this lemma, helps us bound the total dual decrease.

Claim 3.1 *If we are in case b(ii) of the dual update, then $\sum_{j \in P_i} \frac{\gamma_{ij} t_j}{a_{ij}} \le \frac{1}{2k} \cdot \frac{1}{a_{ih}}$.*

Using this claim, we bound the loss in dual value caused by dual constraint i:

$$\begin{aligned} \sum_{j \in P_i} \frac{d_{m(h)}}{d_{ih}} \cdot y_j \quad &= \quad \frac{d_{m(h)}}{d_{ih}} \cdot \sum_{j \in P_i} \gamma_{ij} \cdot t_j\, d_{ij} \quad = \quad \frac{d_{m(h)}}{c_i/a_{ih}} \cdot \sum_{j \in P_i} \gamma_{ij} \cdot t_j\, (c_i/a_{ij}) \\ &= \quad d_{m(h)}\, a_{ih} \cdot \sum_{j \in P_i} \gamma_{ij} \cdot \frac{t_j}{a_{ij}} \quad \le_{\text{(Claim 3.1)}} \quad d_{m(h)}\, a_{ih} \cdot \frac{1}{2k} \cdot \frac{1}{a_{ih}} \quad = \quad \frac{d_{m(h)}}{2k}\,. \end{aligned}$$

Summing over the $|T_j| \le k$ dual constraints affected, the total decrease is at most $\frac{1}{2} d_{m(h)} \le \frac{1}{2} d_{m(h)} t_h$ (since there is no decrease when $t_h = 0$). Subtracting from the increase of $d_{m(h)} \cdot t_h$ gives a net increase of at least $\frac{1}{2} d_{m(h)} t_h$, proving the lemma. □

PROOF OF CLAIM 3.1. Consider the primal constraints j such that $T_j \ni i$: when they arrived, the value of primal variable x_i may have increased. The first few among the constraints j such that $T_j \ni i$ lie in the set P_i: when $j \in P_i$ arrived, we added at least $\frac{1}{k \cdot a_{ij}} \frac{d_{m(j)}}{d_{ij}}$ to x_i's value[1], and did so t_j times. Hence the value of x_i after seeing the constraints in P_i is at least $\sum_{j \in P_i} \frac{d_{m(j)} t_j}{k \cdot a_{ij} \cdot d_{ij}} \ge \sum_{j \in P_i} \frac{\gamma_{ij} t_j}{k \cdot a_{ij}}$, using (3.5).

If χ_i is the value of x_i after seeing the constraints in P_i, and χ_i' is its value after seeing the rest of the constraints in $Q_i := (\{j < h \mid T_j \ni i\} \setminus P_i)$. Then, by the multiplicative part of the primal update,

$$\frac{\chi_i'}{\chi_i} \;\ge\; \prod_{j \in Q_i} \left(1 + \frac{d_{m(j)}}{d_{ij}}\right)^{t_j} \;\ge_{(3.5)}\; \prod_{j \in Q_i} (1 + \gamma_{ij})^{t_j} \;\ge_{(\gamma_{ij} \le 1)}\; e^{\frac{1}{2} \sum_{j \in Q_i} \gamma_{ij} t_j} \;\ge\; 2k^2. \tag{3.6}$$

The last inequality uses the fact that $k \ge 2$, and that:

$$\sum_{j \in Q_i} \gamma_{ij} t_j \;=\; \sum_{j \in Q_i} y_j/d_{ij} \;=\; \sum_{j \in Q_i} \frac{y_j \cdot a_{ij}}{c_i} \;=\; \frac{1}{c_i} \left(\sum_{j < h} a_{ij} y_j - \sum_{j \in P_i} a_{ij} y_j \right) \;>\; 5 \log k,$$

where the inequality is because we are in case b(ii) and $\sum_{j \in P_i} a_{ij} y_j \le (5 \log k) \cdot c_i$ by Lemma 3.4.

[1] More precisely, x_i increased by at least $\frac{1}{k_j \cdot a_{ij}} \frac{d_{m(j)}}{d_{ij}}$ where $k_j \le k$ was the estimate of the row-sparsity at the arrival of constraint j, and k is the current row-sparsity estimate.

Finally, when doing the primal/dual update steps for constraint h, the value of x_i just before this must have been $\chi'_i < 1/a_{ih}$ (otherwise constraint h would have already been satisfied just by variable x_i). Note that χ_i is at least $\sum_{j \in P_i} \frac{\gamma_{ij} t_j}{k \cdot a_{ij}}$, by the first calculations. And $\chi'_i/\chi_i \geq 2k^2$ by (3.6). Putting these together gives

$$\sum_{j \in P_i} \frac{\gamma_{ij} t_j}{k \cdot a_{ij}} \quad \leq \quad \frac{1}{2k^2} \cdot \frac{1}{a_{ih}},$$

and hence the claim. □

Lemma 3.6 and Lemma 3.2 imply that the dual increase is at least 1/4 times the primal increase, and Lemma 3.5 implies that we have an $O(\log k)$-approximately feasible dual, implying the following theorem:

THEOREM 3.1 *Algorithm 3.1 is an $O(\log k)$-competitive online algorithm for covering linear programs without upper-bound constraints, where k is the row-sparsity of the constraint matrix.*

4. The Online Algorithm for CIPs We now want to solve CLPs with variable upper bounds, en route to solving general covering integer programs of the form (IP1). However, it is well-known that when we have variable upper-bounds, the natural LP relaxation of CIPs has a large integrality gap.[2] Hence, Carr et al. [9] suggested adding the *knapsack cover* (KC) inequalities—defined below—to reduce the integrality gap significantly. In this section, we first show how to extend Algorithm 3.1 to get an $O(\log k)$-competitive algorithm for the natural CLP relaxation (with upper bounds) where we also satisfy some suitable KC inequalities. Next, we round (in an online fashion) such a fractional solution to get a randomized $O(\log \ell \cdot \log k)$-competitive online algorithm for general k-row-sparse and ℓ-column-sparse CIPs.

Knapsack Cover Inequalities. Given a CIP of the form (IP1), the KC-inequalities for a particular covering constraint $\sum_{i \in [n]} a_{ij} x_i \geq 1$ are defined as follows: for *any* subset $H \subseteq [n]$ of variables, the maximum possible contribution of the variables in H to the constraint is $a_j(H) := \sum_{i \in H} a_{ij} u_i$, and if $a_j(H) < 1$ then at least a contribution of $1 - a_j(H)$ must come from variables $[n] \setminus H$. Moreover, in any *integral solution* $\mathbf{x}$, since each positive variable x_i is at least one, we get the inequality:

$$\sum_{i \in [n] \setminus H} \min\{a_{ij},\, 1 - a_j(H)\} \cdot x_i \quad \geq \quad 1 - a_j(H) \tag{4.7}$$

Since (4.7) is not necessarily true for *fractional* solutions satisfying $\sum_{i \in [n]} a_{ij} x_i \geq 1$, we add such additional constraints to the LP, for each original constraint j and $H \subseteq [n]$ where $a_j(H) < 1$. There are exponentially many such KC-inequalities, and it is not known how to separate exactly over these in poly-time[3]. But as in previous works [9, 11, 5], the randomized rounding algorithm just needs us to enforce one specific KC-inequality for each constraint j—namely for the set $H := \{i \in [n] \mid x_i \geq \tau \cdot u_i\}$ with some suitable threshold $\tau > 0$. We call this the "special" KC-inequality for constraint j.

4.1 Fractional Solution with Upper Bounds and KC-inequalities In extending Algorithm 3.1 from the previous section to also handle "box constraints" (those of the form $0 \leq x_i \leq u_i$), and the associated KC-inequalities, the high-level idea is to create a "wrapper" procedure around Algorithm 3.1 which ensures these new inequalities: when a constraint $\sum_{i \in T_j} a_{ij} x_i \geq 1$ arrives, we start to apply the primal update step from Algorithm 3.1. Now if some variable x_p gets "close" to its upper bound u_p, we then set $x_p = u_p$, and feed the new inequality $\sum_{i \in T_j \setminus p} a_{ij} x_i \geq 1 - a_{pj} u_p$ (or rather, a knapsack cover version of it) to Algorithm 3.1, and continue. Implementing this idea needs a little more work. For the rest of the discussion, $\tau \in (0, \frac{1}{2})$ is a threshold that will be fixed later.

Suppose we want a solution to:

$$(IP) \;\; \min \left\{ \sum_{i=1}^{n} c_i x_i \;\; \middle| \;\; \sum_{i \in S_j} a_{ij} x_i \geq 1 \;\; \forall j \in [m],\; 0 \leq x_i \leq u_i, x_i \in \mathbb{Z} \;\; \forall i \in [n], \right\}$$

[2] The trivial CIP $\min\{x_1 \mid M x_1 \geq 1\}$ has integrality gap M, no upper bounds needed. However, if we truncate the a_{ij}s to be at most 1 (which is the right-hand-side value), and we have no upper bound constraints, this gap disappears. Introducing upper bounds brings back large integrality gaps, e.g. $\min\{x_1 | x_1 + (1-\epsilon)x_2 \geq 1,\, 0 \leq x_1, x_2 \leq 1\}$ which has an integrality gap of $1/\epsilon$.

[3] KC-inequalities can be separated in pseudo-polynomial time via a dynamic program for the knapsack problem.

where we use $S_j \subseteq [n]$ to denote the non-zero columns of each constraint $j \in [m]$. Recall that the row-sparsity $\max_{j\in[m]} |S_j| \leq k$. The natural LP relaxation is:

$$(P) \qquad \min \left\{ \sum_{i=1}^{n} c_i x_i \quad | \quad \sum_{i\in S_j} a_{ij} x_i \geq 1 \ \ \forall j \in [m], \quad 0 \leq x_i \leq u_i \ \ \forall i \in [n] \right\}$$

Algorithm 4.1 will find online a feasible fractional solution to this LP relaxation (P), that also satisfies some additional KC-inequalities. This algorithm maintains a vector $\mathbf{x} \in \mathbb{R}^n$ that need not be feasible for the covering constraints in (P). However $\mathbf{x}$ implicitly defines the "real solution" $\overline{\mathbf{x}} \in \mathbb{R}^n$ as follows:

$$\overline{x}_i = \begin{cases} x_i & \text{if } x_i < \tau u_i \\ u_i & \text{otherwise} \end{cases}, \qquad \forall i \in [n]$$

The solution $\overline{\mathbf{x}}$ to (P) is constructed by solving a related covering linear program (P') *without variable upper-bounds*, having row-sparsity at most k, where the constraints are defined by Algorithm 4.1.

$$(P') \qquad \min \left\{ \sum_{i=1}^{n} c_i x_i \quad | \quad \sum_{i\in T_h} \alpha_{ih} x_i \geq 1 \ \ \forall h \in [m'], \quad x_i \geq 0 \ \ \forall i \in [n] \right\}$$

At the beginning of the algorithm, $h = 0$. When the j^{th} constraint for (IP), namely $\sum_{i\in S_j} a_{ij} x_i \geq 1$, arrives online, the algorithm generates (potentially several) constraints for (P') based on it. Claim 4.2 shows that these constraints are all valid for (IP), so the optimal solution to (P') is at most opt_{IP}.

Algorithm 4.1 Online covering with box constraints and KC-inequalities.

When constraint j (i.e., $\sum_{i\in S_j} a_{ij} \cdot x_i \geq 1$) arrives for (P),

1: set $h \leftarrow h+1$, $t_h \leftarrow 0$, $F_j \leftarrow \{i \in S_j : x_i \geq \tau u_i\}$, $T_h \leftarrow S_j \setminus F_j$.
2: set $b \leftarrow 1 - \sum_{i\in F_j} a_{ij} u_i$, and $\alpha_{ih} \leftarrow \min\left\{1, \frac{a_{ij}}{b}\right\}$, $\forall i \in T_h$, and $\alpha_{ih} = 0$, $\forall i \notin T_h$.
3: if $b > 0$ then generate constraint $\sum_{i\in T_h} \alpha_{ih} x_i \geq 1$ for (P'); else **halt**.
// if $b \leq 0$ then constraint j to (P) is satisfied by $\overline{\mathbf{x}}$.
4: **while** $(\sum_{i\in T_h} \alpha_{ih} \cdot x_i < 1)$ **do**
5: *// start primal-update process for h^{th} constraint $(\sum_{i\in T_h} \alpha_{ih} \cdot x_i \geq 1)$ to (P').*
6: if $T_h = \emptyset$, return INFEASIBLE.
7: define $d_{ih} := \frac{c_i}{\alpha_{ih}}$ for all $i \in [n]$, and $d_{m(h)} := \min_i d_{ih} := \min_{i\in T_h} d_{ih}$.
8: define $\delta \leq 1$ to be the maximum value in $(0, 1]$ so that:

$$\max_{i\in T_h} \left\{ \frac{1}{u_i} \left[\left(1 + \delta \cdot \frac{d_{m(h)}}{d_{ih}}\right) x_i \ + \ \frac{\delta}{k \cdot \alpha_{ih}} \frac{d_{m(h)}}{d_{ih}} \right] \right\} \quad \leq \quad \tau$$

9: perform an update step *for constraint h* as:

$$x_i^{new} \leftarrow \left(1 + \delta \cdot \frac{d_{m(h)}}{d_{ih}}\right) x_i^{old} \ + \ \frac{\delta}{k \cdot \alpha_{ih}} \frac{d_{m(h)}}{d_{ih}}, \qquad \forall i \in T_h.$$

10: set $t_h \leftarrow t_h + \delta$.
11: let $F'_h \leftarrow \{i \in T_h : x_i = \tau u_i\}$ and $F_j \leftarrow F_j \bigcup F'_h$. *//note that $i \in F_j \iff \overline{x}_i = u_i$ and $i \in S_j$.*
12: **if** $(F'_h \neq \emptyset)$ **then**
13: *// new constraint $h+1$ to (P') is generated; constraint h is deemed to be satisfied if $h+1$ is.*
14: set $h \leftarrow h+1$, $t_h \leftarrow 0$, and $T_h \leftarrow S_j \setminus F_j$.
15: set $b \leftarrow 1 - \sum_{i\in F_j} a_{ij} u_i$, $\alpha_{ih} = \min\left\{1, \frac{a_{ij}}{b}\right\}$, $\forall i \in T_h$ and $\alpha_{ih} = 0$, $\forall i \notin T_h$.
16: if $b > 0$ generate constraint $\sum_{i\in T_h} \alpha_{ih} x_i \geq 1$ for (P'); else **halt**.
// if $b \leq 0$ then constraint j to (P) is satisfied by $\overline{\mathbf{x}}$.
17: **end if**
18: **end while** *// constraint j to (P) is now satisfied.*

The following theorem summarizes Algorithm 4.1, and will be used to perform the online rounding in the next subsection.

THEOREM 4.1 *Let $\mathbf{x}^{(j)}$ and $\overline{\mathbf{x}}^{(j)}$ denote the solution vectors $\mathbf{x}$ and $\overline{\mathbf{x}}$ immediately after the j^{th} constraint to (P) has been satisfied in Algorithm 4.1. Then, the following conditions are satisfied.*

(i) The solutions $\mathbf{x}$ *and* $\overline{\mathbf{x}}$ *are non-decreasing over time.*

(ii) Solution $\overline{\mathbf{x}}^{(j)}$ *satisfies the first* j *constraints of* (P)*.*

(iii) For each $j \in [m]$ *let* $H_j = \{i \in [n] \mid x_i^{(j)} \geq \tau \cdot u_i\}$ *and* $a_j(H_j) = \sum_{r \in H_j} a_{rj} u_r$*. Then solution* $\mathbf{x}^{(j)}$ *satisfies the KC-inequality corresponding to constraint* j *with the set* H_j*, i.e., if* $a_j(H_j) < 1$ *then:*

$$\sum_{i \in S_j \setminus H_j} \min\{a_{ij},\, 1 - a_j(H_j)\} \cdot x_i^{(j)} \quad \geq \quad 1 - a_j(H_j).$$

(iv) The cost $\sum_{i=1}^{n} c_i \cdot x_i = O(\log k) \cdot \mathsf{opt}_{IP}$*.*

Again, the value of row-sparsity k is not required in advance—it suffices to use the current estimate as in Algorithm 3.1. The rest of this subsection proves Theorem 4.1.

By construction, $\mathbf{x}$ and $\overline{\mathbf{x}}$ are non-decreasing over the run of the algorithm, which is property (i).

Next we prove property (ii), that $\overline{\mathbf{x}}$ is feasible to (P). Clearly $\overline{\mathbf{x}} \in [0, \mathbf{u}]$. Upon arrival of the j^{th} constraint to (P), solution $\mathbf{x}$ is increased until the condition in line 4 of Algorithm 4.1 is satisfied. The following claim then implies that $\overline{\mathbf{x}}$ satisfies constraint j to (P).

Claim 4.1 *Consider Algorithm 4.1 after the arrival of constraint* j *to* (P)*. At any point in the while-loop, let* $\mathbf{v}$ *denote the current fractional solution and* h *index the current constraint (*$\sum_i \alpha_{ih} \cdot x_i \geq 1$*) to* (P')*. If* $\sum_i \alpha_{ih} \cdot v_i \geq 1$ *then* $\overline{\mathbf{v}}$ *satisfies constraint* j *to* (P)*, i.e.* $\sum_i a_{ij} \cdot \overline{v}_i \geq 1$*.*

Proof. Recall that the j^{th} constraint to (P) is $\sum_{i \in S_j} a_{ij} \cdot x_i \geq 1$. At any point in the while-loop, the current constraint h is of the form $\sum_i \alpha_{ih} \cdot x_i \geq 1$, where

$$\alpha_{ih} = \begin{cases} \min\{1,\, \frac{a_{ij}}{b}\} & i \in S_j \setminus F_j \\ 0 & \text{otherwise} \end{cases}$$

Above, $F_j = \{i \in S_j : w_i \geq \tau \cdot u_i\}$ with $\mathbf{w}$ being the fractional solution at the time constraint h was generated; and $b = 1 - \sum_{i \in F_j} a_{ij} \cdot u_i > 0$. Since the solution is non-decreasing, the current fractional solution $\mathbf{v} \geq \mathbf{w}$. So $\overline{v}_i = \overline{w}_i = u_i$ for all $i \in F_j$. Hence:

$$\sum_{i \in S_j} a_{ij} \cdot \overline{v}_i \;=\; \sum_{i \in F_j} a_{ij} \cdot u_i + \sum_{i \in S_j \setminus F_j} a_{ij} \cdot \overline{v}_i \;\geq\; a_j(F_j) + \sum_{i \in S_j \setminus F_j} a_{ij} \cdot v_i \;\geq\; 1.$$

Above we use $a_j(F_j) = \sum_{i \in F_j} a_{ij} \cdot u_i$, and the last inequality is by:

$$\sum_{i \in S_j \setminus F_j} a_{ij} \cdot v_i \;\geq\; b \cdot \sum_{i \in S_j \setminus F_j} \alpha_{ih} \cdot v_i = b \cdot \sum_i \alpha_{ih} \cdot v_i \;\geq\; b \;=\; 1 - a_j(F_j).$$

The last inequality is because $\sum_i \alpha_{ih} \cdot v_i \geq 1$. □

To see property (iii), note that the termination condition of the while loop captures this very KC inequality since $T_h = \{i \in S_j : x_i < \tau \cdot u_i\} = S_j \setminus F_j$ and $b = 1 - \sum_{i \in F_j} a_{ij} u_i = 1 - a_j(F_j)$ at all times.

Finally, for the main property (iv), we use a primal-dual analysis as in Section 3: we will show how to maintain an $O(\log k)$-approximately feasible dual $\mathbf{y}$ for (P'), so that $\mathbf{c} \cdot \mathbf{x}$ is at most $O(1)$ times the dual objective $\sum_{h \in [m']} y_h$. This would imply $\mathbf{c} \cdot \mathbf{x} \leq O(\log k) \mathsf{opt}_{P'}$. The following claim can then be used to relate $\mathsf{opt}_{P'}$ and opt_{IP}.

Claim 4.2 *The optimal value* $\mathsf{opt}_{P'}$ *of LP* (P') *is at most* opt_{IP}*, the optimal value of* (IP)*.*

Proof. We will show that every inequality in (P') can be obtained as a KC-inequality generated for (IP). Indeed, consider the h^{th} constraint $\sum_{i \in T_h} \alpha_{ih} x_i \geq 1$ added to (P'), say due to the j^{th} constraint $\sum_{i \in S_j} a_{ij} \cdot x_i \geq 1$ of (IP). Here $T_h = S_j \setminus F_j$ for some $F_j \subseteq S_j$, and $\alpha_{ih} = \min\left\{1, \frac{a_{ij}}{b}\right\}$ for $i \in T_h$ with $b = 1 - \sum_{r \in F_j} a_{rj} \cdot u_r > 0$. In other words, the h^{th} constraint to (P') reads

$$\sum_{i \in S_j \setminus F_j} \min\left\{1 - \sum_{r \in F_j} a_{rj} \cdot u_r,\ \ a_{ij}\right\} \cdot x_i \quad \geq \quad 1 - \sum_{r \in F_j} a_{rj} \cdot u_r,$$

which is the KC-inequality from the j^{th} constraint of (IP) with fixed set F_j. Now since all KC-inequalities are valid for any integral solution to (IP), the original claim follows. □

Now we show how to maintain the approximate dual solution for (P'), and bound the cost of the primal update in terms of this dual cost. Note that (P') is a covering linear program without variable upper-bounds, and has row sparsity at most k. So we can apply the analysis from Section 3. However, there is a slight difference in the primal-update step, namely the presence of an extra factor $\delta \in (0, 1]$ in line 9 of Algorithm 4.1; this is needed to ensure variable upper-bounds in (P). This modification in the update step also requires a small change in the following analysis.

The dual of (P') is:

$$(D') \qquad \max\left\{\sum_{h=1}^{m'} y_h \quad | \quad \sum_{h: T_h \ni i} \alpha_{ih} \cdot y_h \leq c_i \ \ \forall i \in [n], \quad y_h \geq 0 \ \ \forall h \in [m']\right\}$$

The dual update process is similar to that in Section 3. When constraint h to (P') is deemed satisfied in line 13 or line 4, update dual $\mathbf{y}$ as follows:

Let $d_{ih}, d_{m(h)}, t_h$ be as defined in Algorithm 4.1.

(a) Set $y_h \leftarrow d_{m(h)} \cdot t_h$.

(b) For each dual constraint i such that $i \in T_h$ (i.e., $\sum_l \alpha_{il} y_l \leq c_i$), do the following:

(i) If $\sum_{l<h} \alpha_{il} y_l \leq (10 \log k)\, c_i$, do nothing; else

(ii) Let $k_i < h$ be the largest index such that $\sum_{l \leq k_i} \alpha_{il} y_l \leq (5 \log k)\, c_i$;
Let $P_i = \{l \leq k_i \mid T_l \ni i\}$ be the indices of these first few dual variables active in dual constraint i, and for all $l \in P_i$ set:

$$y_l^{new} \leftarrow \left(1 - \min\{1,\, t_h\} \cdot \frac{d_{m(h)}}{d_{ih}}\right) \cdot y_l^{old}.$$

The only difference from Section 3 is to change $\left(1 - \frac{d_{m(h)}}{d_{ih}}\right)$ to $\left(1 - \min\{1, t_h\} \frac{d_{m(h)}}{d_{ih}}\right)$. This is because of the modification to the primal update step (involving the additional factor δ), due to which t_h could be much smaller than one. Here too, each dual variable y_h starts off at $d_{m(h)} t_h$, and only decreases thereafter.

In the rest of the proof, we omit details that are repeated from Section 3, and only point out differences, if any. The next six lemmas are similar to the corresponding ones in Section 3.

LEMMA 4.1 *For any constraint h to (P'), the value $t_h \leq 2 \log k$.*

PROOF. Observe that after each primal update step corresponding to constraint h, t_h increases by a value at most one. Each time that t_h increases by 1, the process behaves exactly as in Algorithm 3.1; so by Lemma 3.1, the number of such increases is strictly less than $2 \log k$ (which is an integer since we assumed k is a power of 2). Also, the first time that t_h increases by $\delta < 1$, the algorithm adds at least one variable to F'_h, fixes t_h and moves on to a new constraint $h + 1$. □

LEMMA 4.2 *The total increase in $\sum_{i \in [n]} c_i \cdot x_i$ due to updates for constraint h to (P') is at most $2\, t_h\, d_{m(h)}$.*

LEMMA 4.3 *In the dual update for constraint h to (P'), variable y_h increases the left-hand-side of each dual constraint i by $\alpha_{ih} \cdot d_{m(h)} \cdot t_h \leq (2 \log k) \cdot c_i$.*

LEMMA 4.4 *If the dual update for constraint h to (P') reaches step b(ii), then k_i is well-defined and the set P_i is non-empty; moreover, $\frac{\sum_{l \in P_i} \alpha_{il} y_l}{c_i \log k} \in [3, 5]$.*

LEMMA 4.5 *After the dual update step for constraint h to (P'), each dual constraint i satisfies $\sum_l \alpha_{il} y_l \leq (12 \log k)\, c_i$. Hence the dual solution is $(12 \log k)$-approximately feasible.*

PROOF. As in the proof of Lemma 3.5, consider the update due to constraint h to (P') and the i^{th} dual constraint for some $i \in T_h$. If we are in case b(i), Lemma 4.3 implies that $\sum_l \alpha_{il} y_l \leq (10 \log k) c_i + (2 \log k) c_i$. For case b(ii), the decrease in the left-hand-side $\sum_{l \in P_i} \alpha_{il} y_l$ of constraint i is at least $\min\{1, t_h\} \cdot \sum_{l \in P_i} \alpha_{il} y_l \cdot (d_{m(h)}/d_{ih})$. By Lemma 4.4, the sum $\sum_{l \in P_i} \alpha_{il} y_l \geq c_i \, (3 \log k)$ and hence the reduction in the left-hand-side of dual constraint i is at least

$$\min\{3 \log k, \, t_h\} \cdot \frac{d_{m(h)}}{d_{ih}} \cdot c_i \quad = \quad d_{m(h)} \cdot \alpha_{ih} \cdot \min\{3 \log k, \, t_h\} \quad \geq \quad d_{m(h)} \cdot \alpha_{ih} \cdot t_h.$$

The inequality uses Lemma 4.1. Combined with Lemma 4.3 it follows that there is no net increase in the left-hand-side. Hence we can maintain the invariant that $\sum_l \alpha_{il} y_l$ is at most $(12 \log k) \, c_i$. □

LEMMA 4.6 *The net increase in dual value after updates for constraint h to (P') is at least $\frac{1}{2} \, d_{m(h)} \cdot t_h$.*

PROOF. The increase in the dual value due to y_h is $d_{m(h)} \cdot t_h$. As in Lemma 3.6, let us bound the decrease in the other y_l's. Consider any of the k dual constraints $i \in T_h$. Again define $\gamma_{il} := \frac{y_l}{t_l \, d_{il}}$ for $l < h$; since y_l started off at $t_l \cdot d_{m(l)}$ and never increased, we have $\gamma_{il} \leq d_{m(l)}/d_{il} \leq 1$. Exactly as in Claim 3.1,

Claim 4.3 *If we are in case b(ii) of the dual update, then $\sum_{l \in P_i} \frac{\gamma_{il} t_l}{\alpha_{il}} \leq \frac{1}{2k} \cdot \frac{1}{\alpha_{ih}}$.*

Using Claim 4.3 (and calculations as in Lemma 3.6), the decrease in dual objective due to constraint i is:

$$\min\{1, \, t_h\} \cdot \sum_{l \in P_i} \frac{d_{m(h)}}{d_{ih}} \cdot y_l \quad \leq \quad \frac{1}{2k} \, d_{m(h)} \cdot \min\{1, \, t_h\} \quad \leq \quad \frac{1}{2k} \, d_{m(h)} \cdot t_h.$$

Since there are $|T_h| \leq k$ dual constraints we have to consider, the total decrease is at most $\frac{1}{2} d_{m(h)} \, t_h$. Subtracting this from the total increase of $d_{m(h)} \cdot t_h$ gives the lemma. □

Comparing Lemma 4.6 with Lemma 4.2, while handling the h^{th} constraint in (P') the increase in the dual objective function is at least $1/4$ of the increase in the primal objective function $\mathbf{c} \cdot \mathbf{x}$. And Lemma 4.5 implies that $\mathbf{y}$ is an $O(\log k)$-approximately feasible dual to (P'). Hence:

$$\mathbf{c} \cdot \mathbf{x} \quad \leq \quad 4(\mathbf{1} \cdot \mathbf{y}) \quad \leq_{weak \ duality} \quad O(\log k) \cdot \mathsf{opt}_{P'} \quad \leq_{Claim \ 4.2} \quad O(\log k) \cdot \mathsf{opt}_{IP}.$$

This completes the proof of property (iv) in Theorem 4.1.

4.2 Online Rounding We now complete the algorithm for CIPs by showing how to round the online fractional solution generated by Theorem 4.1, also in an online fashion. This rounding algorithm does randomized rounding on the incremental change as in [1], but to get a smaller $O(\log \ell)$ loss instead $O(\log m)$, we use the method of randomized rounding with alterations [3, 16]. Recall $\ell \leq m$ is the column-sparsity of the constraint matrix A, namely, the maximum number of constraints that any variable x_i participates in. Recall that the $O(\log \ell)$ bound for offline CIPs given by [17, 11] uses a derandomization of the Lovász Local Lemma via pessimistic estimators, and it is not clear how to implement this in the online setting.

Given that the constraints of a CIP arrive online, we run Algorithm 4.1 to maintain vectors $\mathbf{x}$ and $\overline{\mathbf{x}}$ with properties guaranteed by Theorem 4.1. We set the threshold τ in Algorithm 4.1 to $\frac{1}{8} \cdot \frac{1}{\log \ell}$, where ℓ is the column-sparsity of the original constraint matrix A. We note that the column-sparsity of the matrix with additional KC-inequalities (corresponding to CLP (P') in Subsection 4.1) may be larger; however, the analysis here works directly with the original covering constraints and so we only incur an $O(\log \ell)$-factor loss.

Before any constraints arrive, we pick a uniformly random value $\rho_i \in [0, 1]$ for each variable $i \in [n]$—this is the only randomness used by the algorithm. We maintain the online fractional solution $\mathbf{x}$ from Algorithm 4.1 and a random integer solution $\mathbf{Z} \in \mathbb{Z}^n_{\geq 0}$ (not necessarily feasible) that corresponds to independently rounding each variable. For some constraints, the rounded solution $\mathbf{Z}$ may not be feasible (nor does it maintain monotonicity), and in such cases we perform an additional alteration step to obtain the actual (feasible) integral solution $\mathbf{X} \geq \mathbf{Z}$. Let $\mathbf{X}^{(j)}$ denote this solution immediately after primal constraint j has been satisfied. We start off with $\mathbf{X}^{(0)} = \mathbf{0}$. When the j^{th} constraint arrives and the fractional x_i values have been increased in response to this, we do the following.

(i) Define the "rounded unaltered" solution:

$$Z_i = \begin{cases} 0 & \text{if } x_i < \tau\rho_i \\ \lceil x_i/\tau \rceil & \text{if } \tau\rho_i \le x_i < \tau u_i \\ u_i & \text{if } x_i \ge \tau u_i \end{cases}, \qquad \forall i \in [n].$$

Observe that this rounding ensures that $Z_i \in \{0, 1, \ldots, u_i\}$ for all $i \in [n]$.

(ii) *Maintain monotonicity.* Define:

$$X_i^{new} = \max\left\{X_i^{(j-1)},\ Z_i\right\}, \qquad \forall i \in [n].$$

(iii) *Perform potential alterations.* If we are unlucky and the arriving constraint j is not satisfied by $\mathbf{X}^{new}$, we increase $\mathbf{X}^{new}$ to cover this constraint j as follows. Let $H_j := \{i \in [n] \mid x_i^{(j)} \ge \tau \cdot u_i\}$ be the frozen variables in the fractional solution; note that $Z_i = u_i$ for all $i \in H_j$, so these variables cannot be increased. Recall that $a_j(H_j) := \sum_{r \in H_j} a_{rj} \cdot u_r$. Since constraint j is not satisfied, $a_j(H_j) < 1$. The algorithm now performs the following alteration for constraint j. Consider the residual constraint on variables $[n] \setminus H_j$ after applying the KC-inequality on H_j, i.e.

$$\sum_{i \in [n] \setminus H_j} \min\{a_{ij},\, 1 - a_j(H_j)\} \cdot x_i \quad \ge \quad 1 - a_j(H_j).$$

Set $\overline{a}_{ij} = \min\left\{1,\ \frac{a_{ij}}{1-a_j(H_j)}\right\}$ for all $i \in [n] \setminus H_j$. Consider the following minimum knapsack problem:

$$\begin{aligned} \min \quad & \textstyle\sum_{i \in [n] \setminus H_j} c_i \cdot w_i & (IP_K) \\ \text{subject to:} \quad & \textstyle\sum_{i \in [n] \setminus H_j} \overline{a}_{ij} \cdot w_i \ge 1 & \\ & 0 \le w_i \le u_i, \qquad \forall i \in [n] \setminus H_j & \\ & w_i \in \mathbb{Z}, \qquad \forall i \in [n] \setminus H_j & \end{aligned}$$

Note that there is only one covering constraint in this problem. Let W denote an approximately optimal integral solution to IP_K obtained using the polynomial time approximation scheme (PTAS) for the minimum knapsack problem [18]. It is clear that W satisfies the residual constraint j on variables $[n] \setminus H_j$. Define $\mathbf{X}^{(j)}$ as follows.

$$X_i^{(j)} = \begin{cases} X_i^{new} & \text{for } i \in H_i \\ \max\{X_i^{new},\, W_i\} & \text{for } i \in [n] \setminus H_j \end{cases}$$

This completes the description of the algorithm. By construction, it only increases variables and always maintains a feasible integral solution to the constraints so far. It only remains to bound the expected cost.

Remark: This algorithm does not require knowledge of the final column-sparsity ℓ in advance. At each step, we use the current value of ℓ. Notice that this only affects τ and the definition of $\mathbf{Z}$. However, for fixed values of x_i and ρ_i (any $i \in [n]$) the value of Z_i is non-decreasing with ℓ. Since both ℓ and $\mathbf{x}$ are non-decreasing, vector $\mathbf{Z}$ remains monotone over time. We also require a slightly more general version of Theorem 4.1 where we have multiple thresholds $\tau_1 \le \tau_2 \le \cdots \le \tau_m$ and replace τ by τ_j for each $j \in [m]$ in condition (iii). This extension is straightforward and can be verified directly by setting $\tau := \tau_j$ when constraint j arrives in Algorithm 4.1.

Bounding cost of Z. Consider the rounding algorithm immediately after all m constraints have been satisfied. If $x_i/\tau \in [0, 1]$, then $\mathbb{E}[Z_i] = \Pr[\rho_i \le x_i/\tau] = x_i/\tau$. If $x_i/\tau \ge 1$, then $Z_i \le \lceil x_i/\tau \rceil \le 2x_i/\tau$ with probability 1. Hence:

$$\mathbb{E}\left[\sum_{i=1}^{n} c_i \cdot Z_i\right] \quad \le \quad (2/\tau)\sum_{i=1}^{n} c_i x_i \quad = \quad O(\log k \cdot \log \ell) \cdot \mathtt{opt}_{IP},$$

where we use $1/\tau = O(\log \ell)$, and Theorem 4.1(iv) to bound $\sum_i c_i x_i$.

Bounding cost of the difference $(\mathbf{X} - \mathbf{Z})$. To account for $\mathbf{X} - \mathbf{Z}$, we need to bound the expected cost of all alterations. In the following, let ℓ_j, k_j and τ_j denote the respective values of ℓ, k and τ at the arrival time of constraint j. When j is clear from context we will drop the subscript.

Recall that $H_j := \{i \in [n] \mid x_i^{(j)} \geq \tau_j \cdot u_i\}$ are the frozen variables in the fractional solution after handling constraint j, and note $Z_i = u_i$ for $i \in H_j$. Define $A_j := \{i \in [n] \mid x_i^{(j)} < \tau_j\}$. Note that the random choices $\{\rho_i\}$ only play a role in the values of $\{Z_i \mid i \in A_j\}$, since all variables in $[n] \setminus A_j$ are deterministically set to $Z_i = \min\left\{\lceil x_i^{(j)}/\tau_j \rceil, u_i\right\}$. Let $\mathcal{E}_j$ denote the event that an alteration was performed for constraint j. The event $\mathcal{E}_j$ occurs exactly when $\sum_{i\in[n]} a_{ij} \cdot X_i^{new} < 1$. Since variables $r \in H_j$ have $X_r^{new} = Z_r = u_r$ with probability 1, event $\mathcal{E}_j$ is equivalent to the following two conditions: (i) $a_j(H_j) < 1$ (which is a deterministic condition), and (ii) $\sum_{i\in[n]\setminus H_j} a_{ij} \cdot X_i^{new} < 1 - a_j(H_j)$.

LEMMA 4.7 *The probability of an alteration for constraint j is $\Pr[\mathcal{E}_j] \leq \frac{1}{\ell_j^2}$.*

PROOF. Let $b = 1 - a_j(H_j)$; for $\mathcal{E}_j$ to occur we must have $b > 0$. Set $\overline{a}_{ij} = \min\{a_{ij}/b, 1\}$ for $i \in [n] \setminus H_j$. Now since $\mathbf{Z} \leq \mathbf{X}$ and both are integer-valued,

$$\Pr[\mathcal{E}_j] = \Pr\left[\sum_{i\in[n]\setminus H_j} a_{ij} \cdot X_i^{new} < b\right] \leq \Pr\left[\sum_{i\in[n]\setminus H_j} a_{ij} \cdot Z_i < b\right] = \Pr\left[\sum_{i\in[n]\setminus H_j} \overline{a}_{ij} \cdot Z_i < 1\right] .$$

Theorem 4.1(iii) guarantees that $\sum_{i\in[n]\setminus H_j} \overline{a}_{ij} \cdot x_i^{(j)} \geq 1$. Among $i \in [n] \setminus H_j$, we have

- for $i \in [n] \setminus (H_j \cup A_j)$, $Z_i = \lceil x_i^{(j)}/\tau \rceil$ deterministically, and
- for $i \in A_j$, $Z_i \in \{0, 1\}$ with $\mathbb{E}[Z_i] = x_i^{(j)}/\tau$ independently.

So $\mathbb{E}\left[\sum_{i\in[n]\setminus H_j} \overline{a}_{ij} \cdot Z_i\right] \geq \frac{1}{\tau} = 8 \log \ell_j$. Using a Chernoff bound on the $[0, 1]$-valued independent random variables $\{\overline{a}_{ij} \cdot Z_i\}$, the probability of their sum being less than one is at most $1/\ell_j^2$. □

LEMMA 4.8 *Conditioned on $\mathcal{E}_j$, the cost of incrementing $\boldsymbol{X}^{new}$ to $\boldsymbol{X}^{(j)}$ is at most $36 \sum_{i\in S_j} c_i \cdot x_i^{(j)}$; here $S_j \subseteq [n]$ are the non-zero columns in constraint j.*

PROOF. The fractional solution $\mathbf{x}^{(j)}$ satisfies the KC inequality for set H_j, by Theorem 4.1(iii). In particular, setting $w'_i = x_i^{(j)}$ for $i \in S_j \setminus H_j$ (and zero otherwise) gives a feasible *fractional* solution to the LP relaxation of the minimum knapsack subproblem (IP_K). It now suffices to show that the optimal integral solution to (IP_K) costs at most $18 \sum_{i\in S_j} c_i \cdot w'_i$: since our algorithm uses a PTAS for (IP_K) in the alteration step, the lemma would follow. We note that the LP relaxation to minimum knapsack (IP_K) has unbounded integrality gap; however, using the fact that the fractional solution $\mathbf{w}' \leq \mathbf{u}/2$, we will show that the optimal integral solution to (IP_K) has cost at most $O(1)$ times that of $\mathbf{w}'$.

To upper bound the optimal cost of (IP_K) we give a rounding algorithm that obtains an integral solution $\mathbf{W}'$ from $\mathbf{w}'$ with only a factor 18 increase in cost. Set $W'_i \sim \text{Binom}(u_i, 2w'_i/u_i)$ for all $i \in [n] \setminus H_j$—this definition is valid since $w'_i \leq u_i/2$. Recall that $\text{Binom}(t, p)$ for integer $t \geq 1$ and $p \in [0, 1]$ is the sum of t independent Bernoulli random variables (each with probability p). Clearly $\mathbf{W}'$ always satisfies the upper bounds u_i and has expected cost $2\,\mathbf{c} \cdot \mathbf{w}'$. Moreover, each W'_i is a binomial random variable and $\overline{a}_{ij} \leq 1$, so $\sum_i \overline{a}_{ij} \cdot W'_i$ can be viewed as a sum of independent $[0, 1]$-valued random variables. The expectation $\mathbb{E}\left[\sum_i \overline{a}_{ij} \cdot W'_i\right] \geq 2$, so a Chernoff bound gives $\Pr\left[\sum_i \overline{a}_{ij} \cdot W'_i < 1\right] \leq 8/9$. Using Markov's inequality, $\Pr\left[\mathbf{c} \cdot \mathbf{W}' > 18\,\mathbf{c} \cdot \mathbf{w}'\right] < 1/9$. So with positive probability, $\mathbf{W}'$ satisfies (IP_K), i.e. $\sum_i \overline{a}_{ij} \cdot W'_i \geq 1$, and costs at most $18\,\mathbf{c} \cdot \mathbf{w}'$. This shows that the optimal cost of (IP_K) is at most this cost. □

Thus the total expected cost of alterations after m constraints is:

$$\sum_{j=1}^{m} \Pr[\mathcal{E}_j] \cdot 36 \sum_{i\in S_j} c_i \cdot x_i^{(j)} \;\leq\; 36 \sum_{j=1}^{m} \frac{1}{\ell_j^2} \cdot \sum_{i\in S_j} c_i \cdot x_i^{(j)} \;\leq\; 36 \sum_{i=1}^{n} c_i \cdot x_i^{(m)} \left(\sum_{j:i\in S_j} \frac{1}{\ell_j^2}\right)$$
$$\leq\; 36 \sum_{i=1}^{n} c_i \cdot x_i^{(m)} \left(\frac{1}{1^2} + \frac{1}{2^2} + \cdots\right) \;\leq\; 6\pi^2 \sum_{i=1}^{n} c_i \cdot x_i^{(m)}.$$

The second inequality uses the monotonicity of the fractional solution $\mathbf{x}$, and the third inequality uses that for any $i \in [n]$, the value ℓ_j is at least q upon arrival of the q^{th} constraint containing variable i.

Combining the expected cost of $O(1)\,\mathbf{c}\cdot\mathbf{x}$ for the alterations with the expected cost of $O(\log \ell)\cdot(\mathbf{c}\cdot\mathbf{x})$ for the initial rounding, and Theorem 4.1(iv), we obtain the main result of this section:

THEOREM 4.2 *There is an $O(\log k \cdot \log \ell)$-competitive randomized online algorithm for covering integer programs with row-sparsity k and column-sparsity ℓ.*

Again, we note that the algorithm does not assume knowledge of the eventual k or ℓ values; it works with the current values after each constraint. Furthermore, the algorithm clearly does not need the entire cost function in advance: it suffices to know the cost coefficient c_i of each variable i at the arrival time of the first constraint that contains i.

5. Conclusion In this paper, we gave online algorithms for covering linear and integer programs; the competitive ratios we prove are best possible in terms of their row and column sparsity. Our algorithm was based on the online primal-dual approach, where we both increased and decreased dual variables over the course of the algorithm. It would be interesting to find other applications of the online primal-dual approach that require maintaining non-monotone duals. Another interesting future direction is to design algorithms for covering LP/IPs with competitive ratios in terms of other natural parameters of the input.

Acknowledgment Anupam Gupta was supported in part by NSF awards CCF-0964474 and CCF-1016799.

References

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

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

[3] Noga Alon and Joel Spencer. *The Probabilistic Method*. Wiley-Interscience, New York, 2008.

[4] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *J. ACM*, 59(4):19, 2012.

[5] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. *SIAM J. Comput.*, 41(2):391–414, 2012.

[6] Nikhil Bansal, Nitish Korula, Viswanath Nagarajan, and Aravind Srinivasan. Solving packing integer programs via randomized rounding with alterations. *Theory of Computing*, 8(24):533–565, 2012.

[7] Niv Buchbinder and Joseph (Seffi) Naor. The design of competitive online algorithms via a primal-dual approach. *Found. Trends Theor. Comput. Sci.*, 3(2-3):93–263, 2007.

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

[9] Robert D. Carr, Lisa K. Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*, pages 106–115, 2000.

[10] Deeparnab Chakrabarty, Elyot Grant, and Jochen Könemann. On column-restricted and priority covering integer programs. In *Proceedings of the Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 355–368, 2010.

[11] Stavros G. Kolliopoulos and Neal E. Young. Approximation algorithms for covering/packing integer programs. *J. Comput. Syst. Sci.*, 71(4):495–505, 2005.

[12] Simon Korman. On the use of randomness in the online set cover problem. *M.Sc. thesis, Weizmann Institute of Science*, 2005.

[13] Christos Koufogiannakis and Neal E. Young. Greedy δ-approximation algorithm for covering with arbitrary constraints and submodular cost. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 634–652, 2009.

[14] David Pritchard and Deeparnab Chakrabarty. Approximability of sparse integer programs. *Algorithmica*, 61(1):75–93, 2011.

[15] Aravind Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM J. Comput.*, 29(2):648–670, 1999.

[16] Aravind Srinivasan. New approaches to covering and packing problems. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*, pages 567–576, 2001.

[17] Aravind Srinivasan. An extension of the Lovász Local Lemma, and its applications to integer programming. *SIAM J. Comput.*, 36(3):609–634, 2006.

[18] Vijay Vazirani. *Approximation Algorithms*. Springer, 2001.

[19] Neal E. Young. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994.