On Column-restricted and Priority Covering Integer Programs *

Deeparnab Chakrabarty Elyot Grant Jochen Könemann

University of Waterloo, Waterloo, ON, Canada N2L 3G1

Abstract

In a column-restricted covering integer program (CCIP), all the non-zero entries of any column of the constraint matrix are equal. Such programs capture capacitated versions of covering problems. In this paper, we study the approximability of CCIPs, in particular, their relation to the integrality gaps of the underlying 0,1-CIP.

If the underlying 0,1-CIP has an integrality gap $O(\gamma)$, and assuming that the integrality gap of the *priority version* of the 0,1-CIP is $O(\omega)$, we give a factor $O(\gamma+\omega)$ approximation algorithm for the CCIP. Priority versions of 0,1-CIPs (PCIPs) naturally capture *quality of service* type constraints in a covering problem.

We investigate priority versions of the line (PLC) and the (rooted) tree cover (PTC) problems. Apart from being natural objects to study, these problems fall in a class of fundamental geometric covering problems. We bound the integrality of certain classes of this PCIP by a constant. Algorithmically, we give a polytime exact algorithm for PLC, show that the PTC problem is APX-hard, and give a factor 2-approximation algorithm for it.

1 Introduction

In a *0,1-covering integer program* (0,1-CIP, in short), we are given a constraint matrix $A \in \{0,1\}^{m\times n}$, demands $b \in \mathbb{Z}^m_+$, non-negative costs $c \in \mathbb{Z}^n_+$, and upper bounds $d \in \mathbb{Z}^n_+$, and the goal is to solve the following integer linear program (which we denote by $\texttt{Cov}(A,b,c,d)$).

$$\min\{c^T x : Ax \geq b, 0 \leq x \leq d, x \text{ integer}\}.$$

Problems that can be expressed as 0,1-CIPs are essentially equivalent to set multi-cover problems, where sets correspond to columns and elements correspond to rows. This directly implies that 0,1-CIPs are rather well understood in terms of approximability: the class admits efficient $O(\log n)$ approximation algorithms and this is best possible unless NP = P. Nevertheless, in many cases one can get better approximations by exploiting the structure of matrix A. For example, it is well known that whenever A is *totally unimodular* (TU)(e.g., see [19]), the canonical LP relaxation of a 0,1-CIP is integral; hence, the existence of efficient algorithms for solving linear programs immediately yields fast exact algorithms for such 0,1-CIPs as well.

While a number of general techniques have been developed for obtaining improved approximation algorithms for structured $0,1$-CIPs, not much is known for structured non-$0,1$ CIP instances. In this paper, we attempt to mitigate this problem, by studying the class of *column-restricted covering integer programs* (CCIPs), where all the non-zero entries of any column of

*Supported by NSERC grant no. 288340 and by an Early Research Award. Emails: `deeparnab@gmail.com`, `elyot@uwaterloo.ca`, `jochen@uwaterloo.ca`

the constraint matrix are equal. Such CIPs arise naturally out of $0,1$-CIPs, and the main focus of this paper is to understand how the structure of the underlying 0,1-CIP can be used to derive improved approximation algorithms for CCIPs.

Column-Restricted Covering IPs (CCIPs): Given a 0,1-covering problem $\texttt{Cov}(A, b, c, d)$ and a supply vector $s \in \mathbb{Z}_+^n$, the corresponding CCIP is obtained as follows. Let $A[s]$ be the matrix obtained by replacing all the 1's in the jth column by s_j; that is, $A[s]_{ij} = A_{ij}s_j$ for all $1 \leq i \leq m, 1 \leq j \leq n$. The column-restricted covering problem is given by the following integer program.

$$\min\{c^T x \,:\, A[s]x \geq b, 0 \leq x \leq d, x \text{ integer}\}. \qquad (\texttt{Cov}(A[s], b, c, d))$$

CCIPs naturally capture *capacitated* versions of 0,1-covering problems. To illustrate this we use the following 0,1-covering problem called the tree covering problem. The input is a tree $T = (V, E)$ rooted at a vertex $r \in V$, a set of *segments* $\mathcal{S} \subseteq \{(u, v) : u \text{ is a child of } v\}$, non-negative costs c_j for all $j \in \mathcal{S}$, and demands $b_e \in \mathbb{Z}_+$ for all $e \in E$. An edge e is contained in a segment $j = (u, v)$ if e lies on the unique u, v-path in T. The goal is to find a minimum-cost subset C of segments such that each edge $e \in E$ is contained in at least b_e segments of C. When T is just a line, we call the above problem, the *line cover* (LC) problem. In this example, the constraint matrix A has a row for each edge of the tree and a column for each segment in $\mathcal{S}$. It is not too hard to show that this matrix is TU and thus these can be solved exactly in polynomial time.

In the above tree cover problem, suppose each segment $j \in \mathcal{S}$ also has a capacity supply s_j associated with it, and call an edge e covered by a collection of segments C iff the total supply of the segments containing e exceeds the demand of e. The problem of finding the minimum cost subset of segments covering every edge is precisely the column-restricted tree cover problem. The column-restricted line cover problem encodes the minimum knapsack problem and is thus NP-hard.

For general CIPs, the best known approximation algorithm, due to Kolliopoulos and Young [16], has a performance guarantee of $O(1+\log\alpha)$, where α, called the *dilation* of the instance, denotes the maximum number of non-zero entries in any column of the constraint matrix. Nothing better is known for the special case of CCIPs unless one aims for *bicriteria* results where solutions violate the upper bound constraints $x \leq d$ (see Section 1.1 for more details).

In this paper, our main aim is to understand how the approximability of a given CCIP instance is determined by the structure of the underlying $0,1$-CIP. In particular, if a $0,1$-CIP has a constant integrality gap, under what circumstances can one get constant factor approximation for the corresponding CCIP? We make some steps toward finding an answer to this question.

In our main result, we show that there is a constant factor approximation algorithm for CCIP if *two* induced $0,1$-CIPs have constant integrality gap. The first is the underlying original 0,1-CIP. The second is a *priority* version of the 0,1-CIP (PCIP, in short), whose constraint matrix is derived from that of the 0,1-CIP as follows.

Priority versions of Covering IPs (PCIPs): Given a 0,1-covering problem $\texttt{Cov}(A, b, c, d)$, a priority supply vector $s \in \mathbb{Z}_+^n$, and a priority demand vector $\pi \in \mathbb{Z}_+^m$, the corresponding PCIP is as follows. Define $A[s,\pi]$ to be the following 0,1 matrix

$$A[s,\pi]_{ij} = \begin{cases} 1 & : \quad A_{ij} = 1 \text{ and } s_j \geq \pi_i \\ 0 & : \quad \text{otherwise,} \end{cases} \tag{1}$$

Thus, a column j covers row i, only if its priority supply is higher than the priority demand of row i. The priority covering problem is now as follows.

$$\min\{c^T x \,:\, A[s,\pi]x \geq \mathbb{1}, 0 \leq x \leq d, x \text{ integer}\}. \qquad (\texttt{Cov}(A[s,\pi],\mathbb{1},c))$$

We believe that priority covering problems are interesting in their own right, and they arise quite naturally in covering applications where one wants to model *quality of service* (QoS) or priority restrictions. For instance, in the tree cover problem defined above, suppose each segment j has a *quality of service* (QoS) or priority supply s_j associated with it and suppose each edge e has a QoS or priority demand π_e associated with it. We say that a segment j covers e iff j contains e *and* the priority supply of j exceeds the priority demand of e. The goal is to find a minimum cost subset of segments that covers every edge. This is the priority tree cover problem.

Besides being a natural covering problem to study, we show that the priority tree cover problem is a special case of a classical geometric covering problem: that of finding a minimum cost cover of points by axis-parallel rectangles in 3 dimensions. Finding a constant factor approximation algorithm for this problem, even when the rectangles have uniform cost, is a long standing open problem.

We show that although the tree cover is polynomial time solvable, the priority tree cover problem is APX-hard. We complement this with a factor 2 approximation for the problem. Furthermore, we present constant upper bounds for the integrality gap of this PCIP in a number of special cases, implying constant upper bounds on the corresponding CCIPs in these special cases. We refer the reader to Section 1.2 for a formal statement of our results, which we give after summarizing works related to our paper.

1.1 Related work

There is a rich and long line of work ([10, 12, 18, 20, 21]) on approximation algorithms for CIPs, of which we state the most relevant to our work. Assuming no upper bounds on the variables, Srinivasan [20] gave a $O(1 + \log \alpha)$-approximation to the problem (where α is the dilation as before). Later on, Kolliopoulos and Young [16] obtained the same approximation factor, respecting the upper bounds. However, these algorithms didn't give any better results when special structure of the constraint matrix was known. On the hardness side, Trevisan [22] showed that it is NP-hard to obtain a $(\log \alpha - O(\log \log \alpha))$-approximation algorithm even for 0,1-CIPs.

The most relevant work to this paper is that of Kolliopoulos [13]. The author studies CCIPs which satisfy a rather strong assumption, called the *no bottleneck assumption*, that the supply of any column is smaller than the demand of any row. Kolliopoulos [13] shows that if one is allowed to violate the upper bounds by a multiplicative constant, then the integrality gap of the CCIP is within a constant factor of that of the original 0,1-CIP[1]. As the author notes such a violation is necessary; otherwise the CCIP has unbounded integrality gap. If one is not allowed to violated upper bounds, nothing better than the result of [16] is known for the special case of CCIPs.

[1]Such a result is implicit in the paper; the author only states a $O(\log \alpha)$ integrality gap.

Our work on CCIPs parallels a large body of work on column-restricted *packing* integer programs (CPIPs). Assuming the *no-bottleneck assumption*, Kolliopoulos and Stein [15] show that CPIPs can be approximated asymptotically as well as the corresponding 0,1-PIPs. Chekuri et al. [7] subsequently improve the constants in the result from [15]. These results imply constant factor approximations for the column-restricted tree *packing* problem under the no-bottleneck assumption. Without the no-bottleneck assumption, however, only polylogarithmic approximation is known for the problem [6].

The only work on priority versions of covering problems that we are aware of is due to Charikar, Naor and Schieber [5] who studied the priority Steiner tree and forest problems in the context of QoS management in a network multicasting application. Charikar et al. present a $O(\log n)$-approximation algorithm for the problem, and Chuzhoy et al. [9] later show that no efficient $o(\log\log n)$ approximation algorithm can exist unless NP $\subseteq$ DTIME$(n^{\log\log\log n})$ (n is the number of vertices).

To the best of our knowledge, the column-restricted or priority versions of the line and tree cover problem have not been studied. The best known approximation algorithm known for both is the $O(\log n)$ factor implied by the results of [16] stated above. However, upon completion of our work, Nitish Korula [17] pointed out to us that a 4-approximation for column-restricted line cover is implicit in a result of Bar-Noy et al. [2]. We remark that their algorithm is not LP-based, although our general result on CCIPs is.

1.2 Technical Contributions and Formal Statement of Results

Given a 0,1-CIP $\texttt{Cov}(A, b, c, d)$, we obtain its *canonical LP relaxation* by removing the integrality constraint. The *integrality gap* of the CIP is defined as the supremum of the ratio of optimal IP value to optimal LP value, taken over all non-negative integral vectors b, c, and d. The integrality gap of an IP captures how much the integrality constraint affects the optimum, and is an indicator of the *strength* of a linear programming formulation.

CCIPs: Suppose the CCIP is $\texttt{Cov}(A[s], b, c, d)$. We make the following two assumptions about the integrality gaps of the 0,1 covering programs, both the original 0,1-CIP and the priority version of the 0,1-CIP.

Assumption 1. *The integrality gap of the original 0,1-CIP is $\gamma \geq 1$. Specifically, for any non-negative integral vectors b, c, and d, if the canonical LP relaxation to the CIP has a fractional solution x, then one can find in polynomial time an integral feasible solution to the CIP of cost at most $\gamma \cdot c^T x$. We stress here that the entries of b, c, d could be 0 as well as ∞.*

Assumption 2. *The integrality gap of the PCIP is $\omega \geq 1$. Specifically, for any non-negative integral vectors s, π, c, if the canonical LP relaxation to the PCIP has a fractional solution x, then one can find in polynomial time, an integral feasible solution to the PCIP of cost at most $\omega \cdot c^T x$.*

We give an LP-based approximation algorithm for solving CCIPs. Since the canonical LP relaxation of a CCIP can have unbounded integrality gap, we strengthen it by adding a set of valid constraints called the *knapsack cover constraints*. We show that the integrality gap of this strengthened LP is $O(\gamma + \omega)$, and can be used to give a polynomial time approximation algorithm.

Theorem 1. *Under Assumptions 1 and 2, there is a* $(24\gamma + 8\omega)$*-approximation algorithm for column-restricted CIPs.*

Knapsack cover constraints to strengthen LP relaxations were introduced in [1, 11, 23]; Carr et al. [4] were the first to employ them in the design approximation algorithms. The paper of Kolliopoulos and Young [16] also use these to get their result on general CIPs.

The main technique in the design of algorithms for column-restricted problems is *grouping-and-scaling* developed by Kolliopoulos and Stein [14, 15] for packing problems, and later used by Kolliopoulos [13] in the covering context. In this technique, the *columns* of the matrix are divided into groups of 'close-by' supply values; in a single group, the supply values are then scaled to be the same; for a single group, the integrality gap of the original 0,1-CIP is invoked to get an integral solution for that group; the final solution is a 'union' of the solutions over all groups.

There are two issues in applying the technique to the new strengthened LP relaxation of our problem. Firstly, although the original constraint matrix is column-restricted, the new constraint matrix with the knapsack cover constraints is not. Secondly, unless additional assumptions are made, the current grouping-and-scaling analysis doesn't give a handle on the degree of violation of the upper bound constraints. This is the reason why Kolliopoulos [13] needs the strong no-bottleneck assumption.

We get around the first difficulty by grouping the *rows* as well, into those that get most of their coverage from columns not affected by the knapsack constraints, and the remainder. On the first group of rows, we apply a subtle modification to the vanilla grouping-and-scaling analysis and obtain a $O(\gamma)$ approximate feasible solution satisfying these rows; we then show that one can treat the remainder of the rows as a PCIP and get a $O(\omega)$ approximate feasible solution satisfying them, using Assumption 2. Combining the two gives the $O(\gamma + \omega)$ factor. The full details are given in Section 2.

We stress here that apart from the integrality gap assumptions on the 0,1-CIPs, we do not make any other assumption (like the no-bottleneck assumption). In fact, we can use the modified analysis of the grouping-and-scaling technique to get a similar result as [13] for approximating CCIPs violating the upper-bound constraints, under a *weaker* assumption than the no-bottleneck assumption. The no-bottleneck assumption states that the supply of *any* column is less than the demand of *any* row. In particular, even though a column has entry 0 on a certain row, its supply needs to be less than the demand of that row. We show that if we weaken the no-bottleneck assumption to assuming that the supply of a column j is less than the demand of any row i only if $A[s]_{ij}$ is positive, a similar result can be obtained via our modified analysis.

Theorem 2. *Under assumption 1 and assuming* $A_{ij}s_j \le b_i$*, for all* i, j*, given a fractional solution* x *to the canonical LP relaxation of* $\texttt{Cov}(A[s], b, c, d)$*, one can find an integral solution* $x^{\texttt{int}}$ *whose cost* $c \cdot x^{\texttt{int}} \le 10\gamma(c \cdot x)$ *and* $x^{\texttt{int}} \le 10d$*.*

Priority Covering Problems. In the following, we use PLC and PTC to refer to the priority versions of the line cover and tree cover problems, respectively. Recall that the constraint matrices for line and tree cover problems are totally unimodular, and the integrality of the corresponding 0,1-covering problems is therefore 1 in both case. It is interesting to note that the 0,1-coefficient matrices for PLC and PTC are not totally unimodular in general. The following integrality gap bound is obtained via a primal-dual algorithm.

Theorem 3. *The canonical LP for priority line cover has an integrality gap of at least* $3/2$ *and at most* 2.

In the case of tree cover, we obtain constant upper bounds on the integrality gap for the case $c = \mathbb{1}$, that is, for the minimum cardinality version of the problem. We believe that the PCIP for the tree cover problem with general costs also has a constant integrality gap. On the negative side, we can show an integrality gap of at least $\frac{e}{e-1}$.

Theorem 4. *The canonical LP for* unweighted *PTC has an integrality gap of at most* 6.

We obtain the upper bound by taking a given PTC instance and a fractional solution to its canonical LP, and decomposing it into a collection of PLC instances with corresponding fractional solutions, with the following two properties. First, the total cost of the fractional solutions of the PLC instances is within a constant of the cost of the fractional solution of the PTC instance. Second, union of integral solutions to the PLC instances gives an integral solution to the PTC instance. The upper bound follows from Theorem 3. Using Theorem 1, we get the following as an immediate corollary.

Corollary 1. *There are* $O(1)$*-approximation algorithms for column-restricted line cover and the cardinality version of the column-restricted tree cover.*

We also obtain the following combinatorial results.

Theorem 5. *There is a polynomial-time exact algorithm for PLC.*

Theorem 6. *PTC is APX-hard, even when all the costs are unit.*

Theorem 7. *There is an efficient* 2*-approximation algorithm for PTC.*

The algorithm for PLC is a non-trivial dynamic programming approach that makes use of various structural observations about the optimal solution. The approximation algorithm for PTC is obtained via a similar decomposition used to prove Theorem 4.

We end by noting some interesting connections between the priority tree covering problem and set covering problems in computational geometry. The *rectangle cover* problem in 3-dimensions is the following: given a collection of points P in $\mathbb{R}^3$, and a collection C of axis-parallel rectangles with costs, find a minimum cost collection of rectangles that covers every point. We believe studying the PTC problem could give new insights into the rectangle cover problem.

Theorem 8. *The priority tree covering problem is a special case of the rectangle cover problem in* 3*-dimensions.*

2 General Framework for Column Restricted CIPs

In this section we prove Theorem 1. Our goal is to round a solution to a LP relaxation of $\texttt{Cov}(A[s], b, c, d)$ into an approximate integral solution. We strengthen the following canonical LP relaxation of the CCIP

$$\min\{c^T x \ : \ A[s]x \geq b, 0 \leq x \leq d, x \geq 0\}$$

by adding valid *knapsack cover* constraints. In the following we use $\mathcal{C}$ for the set of columns and $\mathcal{R}$ for the set of rows of A.

2.1 Strengthening the canonical LP Relaxation

Let $F \subset \mathcal{C}$ be a subset of the columns in the column restricted CIP $\texttt{Cov}(A[s], b, c, d)$. For all rows $i \in \mathcal{R}$, define $b_i^F = \max\{0, b_i - \sum_{j \in F} A[s]_{ij} d_j\}$ to be the residual demand of row i w.r.t. F. Define matrix $A^F[s]$ by letting

$$A^F[s]_{ij} = \begin{cases} \min\{A[s]_{ij}, b_i^F\} & : \quad j \in \mathcal{C} \setminus F \\ 0 & : \quad j \in F, \end{cases} \tag{2}$$

for all $i \in \mathcal{C}$ and for all $j \in \mathcal{R}$. The following *Knapsack-Cover* (KC) inequality

$$\sum_{j \in \mathcal{C}} A^F[s]_{ij} x_j \geq b_i^F$$

is valid for the set of all integer solutions x for $\texttt{Cov}(A[s], b, c, d)$. Adding the set of all KC inequalities yields the following stronger LP formulation CIP. We note that the LP is not column-restricted, in that, different values appear on the same column of the new constraint matrix.

$$\begin{aligned} \texttt{opt}_P := \min \quad & \sum_{j \in \mathcal{C}} c_j x_j && && \text{(P)} \\ \text{s.t.} \quad & \sum_{j \in \mathcal{C}} A^F[s]_{ij} x_j \geq b_i^F && \forall F \subseteq \mathcal{C}, \forall i \in \mathcal{R} && (3) \\ & 0 \leq x_j \leq d_j && \forall j \in \mathcal{C} && \end{aligned}$$

It is not known whether (P) can be solved in polynomial time. For $\alpha \in (0, 1)$, call a vector x^* α-relaxed if its cost is at most $\texttt{opt}_P$, and if it satisfies (3) for $F = \{j \in \mathcal{C} \,:\, x_j^* \geq \alpha d_j\}$. An α-relaxed solution to (P) can be computed efficiently for any α. To see this note that one can check whether a candidate solution satisfies (3) for a set F; we are done if it does, and otherwise we have found an inequality of (P) that is violated, and we can make progress via the ellipsoid method. Details can be found in [4] and [16].

We fix an $\alpha \in (0, 1)$, specifying its precise value later. Compute an α-relaxed solution, x^*, for (P), and let $F = \{j \in \mathcal{C} \,:\, x_j^* \geq \alpha d_j\}$. Define $\bar{x}$ as, $\bar{x}_j = x_j^*$ if $j \in \mathcal{C} \setminus F$, and $\bar{x}_j = 0$, otherwise. Since x^* is an α-relaxed solution, we get that $\bar{x}$ is a feasible fractional solution to the *residual* CIP, $\texttt{Cov}(A^F[s], b^F, c, \alpha d)$. In the next subsection, our goal will be to obtain an *integral* feasible solution to the covering problem $\texttt{Cov}(A^F[s], b^F, c, d)$ using $\bar{x}$. The next lemma shows how this implies an approximation to our original CIP.

Lemma 1. *If there exists an integral feasible solution,* $x^{\texttt{int}}$*, to* $\texttt{Cov}(A^F[s], b^F, c, d)$ *with* $c^T x^{\texttt{int}} \leq \beta \cdot c^T \bar{x}$*, then there exists a* $\max\{1/\alpha, \beta\}$*-factor approximation to* $\texttt{Cov}(A[s], b, c, d)$*.*

Proof. Define

$$z_j = \begin{cases} d_j & : \quad j \in F \\ x_j^{\texttt{int}} & : \quad j \in \mathcal{C} \setminus F, \end{cases} \tag{4}$$

Observe that $z \leq d$. z is a feasible integral solution to $\texttt{Cov}(A[s], b, c, d)$ since for any $i \in \mathcal{R}$,

$$\sum_{j \in \mathcal{C}} A[s]_{ij} z_j = \sum_{j \in F} A[s]_{ij} d_j + \sum_{j \in \mathcal{C} \setminus F} A[s]_{ij} x_j^{\texttt{int}} \geq (b_i - b_i^F) + \sum_{j \in \mathcal{C} \setminus F} A^F[s]_{ij} x_j^{\texttt{int}} \geq b_i$$

where the first inequality follows from the definition of b_i^F and since $A[s]_{ij} \geq A^F[s]_{ij}$, the second inequality follows since $x^{\texttt{int}}$ is a feasible solution to $\texttt{Cov}(A^F[s], b^F, c, d)$.
Furthermore,

$$c^T z = \sum_{j \in F} c_j d_j + \sum_{j \in \mathcal{C} \setminus F} c_j x_j^{\texttt{int}} \leq \frac{1}{\alpha} \sum_{j \in F} c_j x_j^* + \beta \sum_{j \in \mathcal{C} \setminus F} c_j x_j^* \leq \max\{\frac{1}{\alpha}, \beta\} \texttt{opt}_P$$

where the first inequality follows from the definition of F and the second from the assumption in the theorem statement. □

2.2 Solving the Residual Problem

In this section we use a feasible fractional solution $\bar{x}$ of $\texttt{Cov}(A^F[s], b^F, c, \alpha d)$, to obtain an *integral* feasible solution $x^{\texttt{int}}$ to the covering problem $\texttt{Cov}(A^F[s], b^F, c, d)$, with $c^T x^{\texttt{int}} \leq \beta c^T \bar{x}$ for $\beta = 24\gamma + 8\omega$. Fix $\alpha = 1/24$.

***Converting to Powers of* 2.** For ease of exposition, we first modify the input to the residual problem $\texttt{Cov}(A^F[s], b^F, c, d)$ so that all entries of are powers of 2. For every $i \in \mathcal{R}$, let $\bar{b}_i$ denote the smallest power of 2 larger than b_i^F. For every column $j \in \mathcal{C}$, let $\bar{s}_j$ denote the largest power of 2 smaller than s_j.

Lemma 2. *$y = 4\bar{x}$ is feasible for $\texttt{Cov}(A^F[\bar{s}], \bar{b}, c, 4\alpha d)$.*

Proof. Focus on row $i \in \mathcal{R}$. We have

$$\sum_{j \in \mathcal{C}} A^F[\bar{s}]_{ij} y_j \geq 2 \cdot \sum_{j \in \mathcal{C}} A^F[s]_{ij} \bar{x}_j \geq 2 b_i^F \geq \bar{b}_i,$$

where the first inequality uses the fact that $s_j \leq 2\bar{s}_j$ for all $j \in \mathcal{C}$, the second inequality uses the fact that $\bar{x}$ is feasible for $\texttt{Cov}(A^F[s], b^F, c, \alpha d)$, and the third follows from the definition of $\bar{b}_i$. □

Partitioning the rows. We call $\bar{b}_i$ the residual demand of row i. For a row i, a column $j \in \mathcal{C}$ is *i-large* if the supply of j is at least the residual demand of row i; it is *i-small* otherwise. Formally,

$$\begin{aligned} \mathcal{L}_i &= \{j \in \mathcal{C} : A_{ij} = 1, \bar{s}_j \geq \bar{b}_i\} \quad \text{is the set of } i\text{-large columns} \\ \mathcal{S}_i &= \{j \in \mathcal{C} : A_{ij} = 1, \bar{s}_j < \bar{b}_i\} \quad \text{is the set of } i\text{-small columns} \end{aligned}$$

Recall the definition from (2), $A^F[\bar{s}]_{ij} = \min(A[\bar{s}]_{ij}, b_i^F)$. Therefore, $A^F[\bar{s}]_{ij} = A_{ij} b_i^F$ for all $j \in \mathcal{L}_i$ since $\bar{s}_j \geq \bar{b}_i \geq b_i^F$; and $A^F[\bar{s}]_{ij} = A_{ij}\bar{s}_j$ for all $j \in \mathcal{S}_i$, since being powers of 2, $\bar{s}_j < \bar{b}_i$ implies, $\bar{s}_j \leq \bar{b}_i/2 \leq b_i^F$.

We now partition the rows into large and small depending on which columns most of their coverage comes from. Formally, call a row $i \in \mathcal{R}$ *large* if

$$\sum_{j \in \mathcal{S}_i} A^F[\bar{s}]_{ij} y_j \leq \sum_{j \in \mathcal{L}_i} A^F[\bar{s}]_{ij} y_j,$$

and small otherwise. Note that Lemma 2 together with the fact that each column in row i's support is either small or large implies,

$$\text{For a large row } i, \quad \sum_{j \in \mathcal{L}_i} A^F[\bar{s}]_{ij} y_j \geq \bar{b}_i/2, \quad \text{For a small row } i, \quad \sum_{j \in \mathcal{S}_i} A^F[\bar{s}]_{ij} y_j \geq \bar{b}_i/2$$

Let $\mathcal{R}_L$ and $\mathcal{R}_S$ be the set of large and small rows.

In the following, we address small and large rows separately. We compute a pair of integral solutions $x^{\texttt{int},\mathcal{S}}$ and $x^{\texttt{int},\mathcal{L}}$ that are feasible for the small and large rows, respectively. We then obtain $x^{\texttt{int}}$ by letting

$$x_j^{\texttt{int}} = \max\{x_j^{\texttt{int},\mathcal{S}}, x_j^{\texttt{int},\mathcal{L}}\}, \tag{5}$$

for all $j \in \mathcal{C}$.

2.2.1 Small rows.

For these rows we use the grouping-and-scaling technique a la [7, 13, 14, 15]. However, as mentioned in the introduction, we use a modified analysis that bypasses the no-bottleneck assumptions made by earlier works.

Lemma 3. *We can find an integral solution $x^{\texttt{int},\mathcal{S}}$ such that*

a) $x_j^{\texttt{int},\mathcal{S}} \le d_j$ for all j,

b) $\sum_{j\in\mathcal{C}} c_j x_j^{\texttt{int},\mathcal{S}} \le 24\gamma \sum_{j\in\mathcal{C}} c_j \bar{x}_j$, and

c) for every small row $i \in \mathcal{R}_S$, $\sum_{j\in\mathcal{C}} A^F[s]_{ij} x_j^{\texttt{int},\mathcal{S}} \ge b_i^F$.

Proof. The complete proof is slightly technical and hence we start with a sketch. Since the rows are small, for any row i, we can zero out the entries that are larger than $\bar{b}_i$, and still $2y$ will be a feasible solution. Note that, now in each row, the entries are $< \bar{b}_i$, and thus are at most $\bar{b}_i/2$ (everything being powers of 2). We stress that it could be that $\bar{b}_i$ of some row is less than the entry in some other row, that is, we don't have the no-bottleneck assumption. However, when a particular row i is fixed, $\bar{b}_i$ is at least any entry of the matrix in the ith row. Our modified analysis of grouping and scaling then makes the proof go through.

We *group* the columns into classes that have s_j as the same power of 2, and for each row i we let $\bar{b}_i^{(t)}$ be the contribution of the class t columns towards the demand of row i. The columns of class t, the small rows, and the demands $\bar{b}_i^{(t)}$ form a CIP where all non-zero entries of the matrix are the same power of 2. We scale both the constraint matrix and $\bar{b}_i^{(t)}$ down by that power of 2 to get a 0,1-CIP, and using assumption 1, we get an integral solution to this 0,1-CIP. Our final integral solution is obtained by concatenating all these integral solutions over all classes.

Till now the algorithm is the standard grouping-and-scaling algorithm. The difference lies in our analysis in proving that this integral solution is feasible for the original CCIP. Originally the no-bottleneck assumption was used to prove this. However, we show since the column values in different classes are geometrically decreasing, the weaker assumption of $\bar{b}_i$ being at least any entry in the ith row is enough to make the analysis go through. We now get into the full proof.

Step 1: Grouping the columns. Let $\bar{s}_{min}$ and $\bar{s}_{max}$ be the smallest and largest supply among the columns in $\mathcal{C} \setminus F$. Since all $\bar{s}_j$ are powers of 2, we introduce the shorthand, $\bar{s}^{(t)}$ for the supply $\bar{s}_{\max}/2^t$. We say that a column j is in *class* $t \ge 0$, if $\bar{s}_j = \bar{s}^{(t)}$, and we let

$$\mathcal{C}^{(t)} := \{j \in \mathcal{C} \setminus F \,:\, \bar{s}_j = \bar{s}^{(t)}\}$$

be the set of class t supplies.

Step 2: Disregarding i-large columns of a small row i. Fix a small row $i \in \mathcal{R}_S$. We now identify the columns j that are i-small. To do so, define $t_i := \log(\bar{s}_{max}/\bar{b}_i)+1$. Observe that any column j in class $\mathcal{C}^{(t)}$ for $t \geq t_i$ are i-small. This is because $\bar{s}_j = s_{max}/2^t \leq s_{max}/2^{t_i} = \bar{b}_i/2 < \bar{b}_i$. Define

$$\bar{b}_i^{(t)} = \begin{cases} 2\sum_{j\in\mathcal{C}^{(t)}} A^F[\bar{s}]_{ij} y_j & : \quad t \geq t_i \\ 0 & : \quad \text{otherwise} \end{cases}$$

as the contribution of the class t, i-small columns to the demand of row i, multiplied by 2. Note that by definition of small rows, these columns contribute to more than 1/2 of the demand, and so

$$\sum_{t\geq t_i} \bar{b}_i^{(t)} \geq \bar{b}_i. \tag{6}$$

Henceforth, we will consider only the contributions of the small i-columns of a small row i.

Step 3: Scaling and getting the integral solution. Fix a class t of columns and scale down by $\bar{s}^{(t)}$ to get a $\{0,1\}$-constraint matrix. (Recall entries of the columns in a class t are all $\bar{s}^{(t)}$.) This will enable us to apply assumption 1 and get a integral solution corresponding to these columns. The final integral solution will be the concatenation of the integral solutions over the various classes.

The constants in the next claim are carefully chosen for the calculations to work out later.

Claim 1. *For any $t \geq 0$ and for all $i \in \mathcal{R}_S$, $6 \cdot \sum_{j\in\mathcal{C}^{(t)}} A_{ij} y_j \geq \lfloor 3\bar{b}_i^{(t)}/\bar{s}^{(t)} \rfloor$.*

Proof. The claim is trivially true for rows i with $t_i > t$ as $\bar{b}_i^{(t)} = 0$ in this case. Consider a row i with $t_i \leq t$. Since any column $j \in \mathcal{C}^{(t)}$ is i-small, we get $A^F[\bar{s}]_{ij} = A_{ij}\bar{s}_j = A_{ij}\bar{s}^{(t)}$. Using the definition of $\bar{b}_i$, we obtain

$$6 \cdot \sum_{j\in\mathcal{C}^{(t)}} A_{ij}\bar{s}^{(t)} y_j = 3\bar{b}_i^{(t)}.$$

Dividing both sides by $\bar{s}^{(t)}$ and taking the floor on the right-hand side yields the claim. □

Since $\alpha = 1/24$ and $\bar{x}$ is a feasible solution to $\texttt{Cov}(A^F[s], b^F, c, d/24)$, we get that $6y_j = 24 \cdot \bar{x}_j \leq d_j$ for all $j \in \mathcal{C} \setminus F$. Thus, the above claim shows that $6y$ is a feasible fractional solution for $\texttt{Cov}(A^{(t)}, \lfloor 3\bar{b}^{(t)}/\bar{s}^{(t)} \rfloor, c^{(t)}, d^{(t)})$, where $A^{(t)}$ is the submatrix of A defined by the columns in $\mathcal{C}^{(t)}$, and $c^{(t)}$ and $d^{(t)}$ are the sub-vectors of c and d, respectively, that are induced by $\mathcal{C}^{(t)}$. Using Assumption 1, we therefore conclude that there is an integral vector $x^{\texttt{int},\mathcal{S},t}$ such that

$$\begin{aligned} x_j^{\texttt{int},\mathcal{S},t} &\leq d_j \quad \text{for all } j \in \mathcal{C}^{(t)}, \text{ and} && (7) \\ \sum_{j\in\mathcal{C}^{(t)}} A_{ij}^{(t)} x_j^{\texttt{int},\mathcal{S},t} &\geq \left\lfloor \frac{3\bar{b}_i^{(t)}}{\bar{s}^{(t)}} \right\rfloor \quad \text{for all } i \in \mathcal{R}_S, \text{ and} && (8) \\ \sum_{j\in\mathcal{C}^{(t)}} c_j x_j^{\texttt{int},\mathcal{S},t} &\leq 6\gamma \cdot \sum_{j\in\mathcal{C}^{(t)}} c_j y_j && (9) \end{aligned}$$

We obtain integral solution $x^{\texttt{int},\mathcal{S}}$ by letting $x_j^{\texttt{int},\mathcal{S}} = x_j^{\texttt{int},\mathcal{S},t}$ if $j \in \mathcal{C}^{(t)}$. Thus $x_j^{\texttt{int},\mathcal{S}} \leq d_j$ for all $j \in \mathcal{C}$, and we get,

$$\sum_{j\in\mathcal{C}} c_j x_j^{\texttt{int},\mathcal{S}} = \sum_{t\geq 0}\sum_{j\in\mathcal{C}^{(t)}} c_j x_j^{\texttt{int},\mathcal{S},t} \leq 6\gamma \cdot \sum_{t\geq 0}\sum_{j\in\mathcal{C}^{(t)}} c_j y_j = 24\gamma \cdot \sum_{j\in\mathcal{C}} c_j \bar{x}_j. \tag{10}$$

Thus we have established parts (a) and (b) of the lemma. It remains to show that $x^{\texttt{int},\mathcal{S}}$ is feasible for the set of small rows.

Step 4: Putting them all together: scaling back. Once again, fix a small row $i \in \mathcal{R}_S$. The following inequality takes only contribution of the i-small columns. We later show this suffices.

$$\begin{aligned}\sum_{j\in\mathcal{C}} A^F[s]_{ij} x_j^{\texttt{int},\mathcal{S}} \geq & \sum_{j\in\mathcal{C}:\ j \text{ is } i\text{-small}} A_{ij} s_j x_j^{\texttt{int},\mathcal{S}} \\ = & \sum_{t\geq t_i}\sum_{j\in\mathcal{C}^{(t)}} A_{ij}^{(t)} s_j x_j^{\texttt{int},\mathcal{S}} \geq \sum_{t\geq t_i}\sum_{j\in\mathcal{C}^{(t)}} A_{ij}^{(t)} \bar{s}^{(t)} x_j^{\texttt{int},\mathcal{S},t} \end{aligned} \tag{11}$$

The first inequality follows since $A^F[s]_{ij} = A_{ij}s_j$ for i-small columns, the equality follows from the definition of t_i, and the final inequality uses the fact that $s_j \geq \bar{s}^{(t)}$ for $j \in \mathcal{C}^{(t)}$. The following claim along with (11) proves feasibility of row i. This is the part where our analysis slightly differs from the standard grouping-and-scaling analysis.

Claim 2. *For any small row $i \in \mathcal{R}_S$,*

$$\sum_{t\geq t_i}\sum_{j\in\mathcal{C}^{(t)}} A_{ij}^{(t)} \bar{s}^{(t)} x_j^{\texttt{int},\mathcal{S},t} \geq b_i^F.$$

Proof. In this proof, the choice of the constant 3 on the right-hand side of the inequality in Claim 1 will become clear. Let $S_i = \{t \geq t_i : 3\bar{b}_i^{(t)} < \bar{s}^{(t)}\}$ be the set of i-small classes t whose fractional supply $\bar{b}_i^{(t)}$ is small compared to its integral supply $\bar{s}^{(t)}$. We now show that for any small row i, the columns in the classes not in S_i suffice to satisfy its demand. Note that

$$\sum_{t\notin S_i, t\geq t_i} \bar{b}_i^{(t)} = \sum_{t\geq t_i} \bar{b}_i^{(t)} - \sum_{t\in S_i} \bar{b}_i^{(t)} \geq \sum_{t\geq t_i} \bar{b}_i^{(t)} - \frac{1}{3}\sum_{t\in S_i} \bar{s}^{(t)} \tag{12}$$

which follows from the definition of S_i. Furthermore, from (6) we know that for a small row, $\sum_{t\geq t_i} \bar{b}_i^{(t)} \geq \bar{b}_i$. Also, since $\bar{s}^{(t)}$ form a geometric series, we get that $\sum_{t\in S_i} \bar{s}^{(t)} \leq \sum_{t\geq t_i} \bar{s}^{(t)} \leq 2\bar{s}^{(t_i)}$. Putting this in (12) we get

$$\sum_{t\notin S_i, t\geq t_i} \bar{b}_i^{(t)} \geq \bar{b}_i - \frac{1}{3}\sum_{t\geq t_i} \bar{s}^{(t)} \geq \bar{b}_i - \frac{2}{3}\bar{s}^{(t_i)} = \frac{2}{3}\bar{b}_i, \tag{13}$$

where the final equality follows from the definition of t_i which implies that $\bar{s}^{(t_i)} = \bar{b}_i/2$.

Moreover, for $t \notin S_i$, we know that $\lfloor 3\bar{b}_i^t/\bar{s}^{(t)} \rfloor \geq \frac{3}{2}\bar{b}_i^t/\bar{s}^{(t)}$ since $\lfloor a \rfloor \geq a/2$ if $a > 1$. Therefore, using inequality (8) in (11), we get

$$\begin{aligned}\sum_{j\in\mathcal{C}} A^F[s]_{ij} x_j^{\texttt{int},\mathcal{S}} \geq \sum_{t\geq t_i}\sum_{j\in\mathcal{C}^{(t)}} A_{ij}^{(t)} \bar{s}^{(t)} x_j^{\texttt{int},\mathcal{S},t} & \geq \sum_{t\notin S_i, t\geq t_i} \bar{s}^{(t)} \left\lfloor \frac{3\bar{b}_i^{(t)}}{\bar{s}^{(t)}} \right\rfloor \\ & \geq \frac{3}{2}\sum_{t\notin S_i, t\geq t_i} \bar{b}_i^{(t)} \\ & \geq \bar{b}_i \geq b_i^F, \end{aligned}$$

where the second-last inequality uses (13), and the last uses the definition of $\bar{b}_i$. This completes the proof of the lemma. ∎

∎

2.2.2 Large rows.

The large rows can be showed to be a PCIP problem and thus Assumption 2 can be invoked to get an analogous lemma to Lemma 3.

Lemma 4. *We can find an integral solution* $x^{\texttt{int},\mathcal{L}}$ *such that*

a) $x_j^{\texttt{int},\mathcal{L}} \le 1$ *for all* j,

b) $\sum_{j\in\mathcal{C}} c_j x_j^{\texttt{int},\mathcal{S}} \le 8\omega \sum_{j\in\mathcal{C}} c_j \bar{x}_j$, *and*

c) for every large row $i \in \mathcal{R}_L$, $\sum_{j\in\mathcal{C}} A^F[s]_{ij} x_j^{\texttt{int},\mathcal{S}} \ge b_i^F$.

Proof. Let $i \in \mathcal{R}_L$ be a large row, and recall that $\mathcal{L}_i$ is the set of i-large columns in $\mathcal{C}$. We have

$$\sum_{j\in\mathcal{L}_i} A^F[s]_{ij} y_j = \sum_{j\in\mathcal{L}_i} A_{ij}\bar{b}_i y_j \ge \bar{b}_i/2,$$

and hence

$$2\sum_{j\in\mathcal{L}_i} A_{ij} y_j \ge 1. \tag{14}$$

Let $A^{\mathcal{R}}$ be the minor of A induced by the large rows. Consider the priority cover problem $\texttt{Cov}(A^{\mathcal{R}}[\bar{s},\bar{b}], \mathbb{1}, c)$. From the definition of $\mathcal{L}_i$, it follows $2y$ is a feasible fractional solution to the priority cover problem.

Using Assumption 2, we conclude that there is an integral solution $x^{\texttt{int},\mathcal{L}}$ such that $\sum_{j\in\mathcal{C}} c_j x_j^{\texttt{int},\mathcal{L}} \le 2\omega \sum_{j\in\mathcal{C}} c_j y_j = 8\omega \sum_{j\in\mathcal{C}} c_j \bar{x}_j$, and $\sum_{j\in\mathcal{C}} A_{ij}^{\mathcal{R}} x_j^{\texttt{int},\mathcal{L}} \ge 1$, for all large rows $i \in \mathcal{R}_L$.

Fix a large row i. Since $A^F[s]_{ij} = b_i^F$ for all i-large columns $\mathcal{L}_i$, we get

$$\sum_{j\in\mathcal{C}} A^F[s]_{ij} x_j^{\texttt{int},\mathcal{L}} \ge \sum_{j\in\mathcal{L}_i} A_{ij} b_i^F x_j^{\texttt{int},\mathcal{L}} = b_i^F \sum_{j\in C} A_{ij}^{\mathcal{R}} x_j^{\texttt{int},\mathcal{L}} \ge b_i^F$$

This completes the proof of the lemma. □

Proof of Theorem 1 Let $x^{\texttt{int},\mathcal{S}}$ and $x^{\texttt{int},\mathcal{L}}$ be as satisfying the conditions of Lemma 3 and 4, respectively. Define $x^{\texttt{int}}$ as $x_j^{\texttt{int}} = \max\{x_j^{\texttt{int},\mathcal{S}}, x_j^{\texttt{int},\mathcal{L}}\}$. We have

a) $x_j^{\texttt{int}} \le d_j$ since both $x_j^{\texttt{int},\mathcal{S}} \le d_j$ and $x_j^{\texttt{int},\mathcal{L}} \le 1 \le d_j$.

b) For any row i, $\sum_{j\in\mathcal{C}} A^F[s]_{ij} x_j^{\texttt{int}} \ge b_i^F$ since the inequality is true with $x^{\texttt{int}}$ replaced by $x^{\texttt{int},\mathcal{S}}$ for small rows, and $x^{\texttt{int}}$ by $x^{\texttt{int},\mathcal{L}}$ for large rows.

c) $\sum_{j\in\mathcal{C}} c_j x_j^{\texttt{int}} \le \sum_{j\in\mathcal{C}} c_j x_j^{\texttt{int},\mathcal{S}} + \sum_{j\in\mathcal{C}} c_j x_j^{\texttt{int},\mathcal{L}} \le (24\gamma + 8\omega) \sum_{j\in\mathcal{C}} c_j \bar{x}_j$.

Thus, $x^{\texttt{int}}$ is a feasible integral solution to $\texttt{Cov}(A^F[s], b^F, c, d)$ with cost bounded as $\sum_{j\in\mathcal{C}} c_j x_j^{\texttt{int}} \le (24\gamma + 8\omega)\sum_{j\in\mathcal{C}} c_j \bar{x}_j$. Noting that $\alpha = 1/24$, the proof of the theorem follows from Lemma 1. □.

2.3 CCIPs with violation of upper-bounds: Proof of Theorem 2

In this section we prove Theorem 2 that we restate here. In the proof, we will indicate how we modify the analysis of grouping-and-scaling that allows us to replace the no-bottleneck assumption with a weaker one.

Theorem 9. *(Theorem 2) Under assumption 1 and assuming $A_{ij}s_j \le b_i$, for all i,j, given a fractional solution x to the canonical LP relaxation of $\mathtt{Cov}(A[s], b, c, d)$, one can find an integral solution $x^{\mathtt{int}}$ whose cost $c \cdot x^{\mathtt{int}} \le 10\gamma(c \cdot x)$ and $x^{\mathtt{int}} \le 10d$.*

Proof. Let x be a feasible solution to $A[s]x \ge b, x \ge 0$. We construct an integral solution $x^{\mathtt{int}}$ such that $A[s]x^{\mathtt{int}} \ge b$ and $c^T x^{\mathtt{int}} \le 10\gamma c^T x$. Let s_{max} and s_{min} be the largest and smallest s_j's.

Grouping: Let $\mathcal{C}^{(t)} := \{j : 2^{-(t+1)}s_{max} < s_j \le 2^{-t}s_{max}\}$ for $t = 0, 1, \ldots T$ where $T = \log(\frac{s_{max}}{s_{min}})$. Let $b_i^t := \sum_{j \in \mathcal{C}^{(t)}} A_{ij}s_j x_j$. Note that $\sum_{t=0}^{T} b_i^t \ge b_i$. Let $m_i^t := \min_{j \in \mathcal{C}^{(t)}: A_{ij} \neq 0} s_j A_{ij}$, that is, m_i^t is the smallest non-zero entry of the ith row of A in the columns of $\mathcal{C}^{(t)}$. Note that $m_i^t > 2^{-(t+1)}s_{max}$. Let m_i be the largest entry of row i. The assumption $A_{ij}s_j \le b_i$ implies $m_i \le b_i$.

Scaling: Let y^t be a vector with $y_j^t = 10x_j$ for $j \in \mathcal{C}^{(t)}$, 0 elsewhere. Note that $\sum_t c^T y^t = 10c^T x$ and $y_i^t \le 10d_i$ for any i. Let $\hat{s}^t$ be a vector with $\hat{s}_j^t = 2^{-(t+1)}s_{max}$ for $j \in \mathcal{C}^{(t)}$, 0 otherwise. Since for all $j \in \mathcal{C}^{(t)}$, $\hat{s}_j^t \ge s_j/2$, for all rows i we have

$$\sum_{j \in \mathcal{C}^{(t)}} A_{ij}\hat{s}_j^t y_j^t \ge 5 \sum_{j \in \mathcal{C}^{(t)}} A_{ij}s_j x_j = 5b_i^t$$

Therefore since $m_i^t \ge 2^{-(t+1)}s_{max}$, we get

$$\sum_{j \in \mathcal{C}^{(t)}} A_{ij}y_j^t \ge \frac{5b_i^t}{2^{-(t+1)}s_{max}} \ge \frac{5b_i^t}{m_i^t} \ge \left\lfloor \frac{5b_i^t}{m_i^t} \right\rfloor$$

If we define an integral vector a^t to be $a_i^t := \lfloor \frac{5b_i^t}{m_i^t} \rfloor$, we see that $Ay^t \ge a^t$. Using assumption 1, there exists an integral solution z^t such that $Az^t \ge a^t$, and $c^T z^t \le \gamma(c^T y^t)$, and $z_i^t \le 10d_i$.

Scaling back: Now fix a row i, and look at

$$\sum_{j \in \mathcal{C}^{(t)}} A_{ij}s_j z_j^t \ge \sum_{j \in \mathcal{C}^{(t)}} A_{ij}m_i^t z_j^t = m_i^t \sum_{j \in \mathcal{C}^{(t)}} A_{ij}z_j^t \ge m_i^t \lfloor \frac{5b_i^t}{m_i^t} \rfloor$$

where the first inequality follows since m_i^t is the minimum entry in the ith row in the columns of $\mathcal{C}^{(t)}$. This is where our analysis slightly differs from the previous analyses of grouping and scaling, where instead of multiplying the RHS by m_i^t, the RHS was multiplied by $2^{-t}s_{max}$. This subtle observation leads us to make a weaker assumption than the no-bottleneck assumption.

Getting the final integral solution:
Define $x^{\mathtt{int}} := \sum_{t=0}^{T} z^t$. Note that $c^T x^{\mathtt{int}} = \sum_t c^T z^t \le \gamma \sum_t c^T y^t = 10\gamma(c^T x)$ and $x^{\mathtt{int}} \le 10d$.

Fix a row i and look at the ith entry of $A[s]x^{\mathtt{int}}$.

$$\sum_{t=0}^{T} \sum_{j \in \mathcal{C}^{(t)}} A_{ij}s_j z_j^t \ge \sum_{t=0}^{T} \lfloor \frac{5b_i^t}{m_i^t} \rfloor m_i^t \tag{15}$$

Let $S_i := \{t : 5b_i^t < m_i^t\}$. Note that

$$\sum_{t \in S_i} b_i^t < \frac{1}{5} \sum_{t \in S_i} m_i^t \leq 3m_i/5$$

the second inequality following from Claim 3 below. This gives us

$$\sum_{t \notin S_i} b_i^t > \sum_{t=0}^{T} b_i^t - 3m_i/5 \geq b_i - 3m_i/5$$

For $t \notin S_i$, we have the floor in the inequality (15) at least 1. So we can use the relation $\lfloor x \rfloor \geq x/2$ for $x \geq 1$. Thus, using $m_i \leq b_i$, we have

$$Ax^{\texttt{int}} \geq \sum_{t \notin S_i} \frac{5b_i^t}{2} \geq 5b_i/2 - 3m_i/2 \geq b_i$$

Claim 3. $\sum_{t=0}^{T} m_i^t \leq 3m_i$.

Proof. Note that the non-zero m_i^t decreases as t goes from 0 to T. Also, for any $t < t'$, we have $m_i^t > 2^{-(t+1)} s_{max}$ and $m_i^{t'} \leq 2^{-t'} s_{max}$. Thus, $m_i^{t'} \leq m_i^t \cdot 2^{-(t'-t-1)}$. Since the largest m_i^t can be at most m_i, $\sum_{t=0}^{T} m_i^t \leq m_i + m_i + m_i/2 + m_i/4 + ... \leq 3m_i$. ☐

☐

3 Priority line cover

We first show that the integrality gap of the canonical linear programming relaxation of PLC is at least 3/2 and at most 2. Subsequently, we present an exact combinatorial algorithm for the problem.

3.1 Canonical LP relaxation: Integrality gap

We start with the canonical LP relaxation for PLC and its dual in Figure 1.

$$\min \Big\{ \sum_{j \in \mathcal{S}} c_j x_j : \quad x \in R_+^{\mathcal{S}} \quad \text{(Primal)}$$
$$\sum_{j \in \mathcal{S}: j \text{ covers } e} x_j \geq 1, \quad \forall e \in E \Big\}$$

$$\max \Big\{ \sum_{e \in E} y_e : \quad y \in R_+^{E} \quad \text{(Dual)}$$
$$\sum_{e \in E: j \text{ covers } e} y_e \leq c_j, \quad \forall j \in \mathcal{S} \Big\}$$

Figure 1: The PLC canonical LP relaxation and its dual.

The following example shows that the integrality gap of (Primal) is at least 3/2.

Example 1. Figure 1 shows a line of odd length k; odd numbered edges have demand 1, and even numbered edges have a demand of 2. Paths are shown as lines above the line graph, and are also numbered. Odd numbered paths have a supply of 2, and even numbered ones have a supply of 1. Dashed lines indicate edges spanned but not covered. All paths have cost 1. Note that a fractional solution is obtained by letting $x_p = 2/3$ for paths 2 and k, and $x_p = 1/3$ otherwise.

The cost of this solution is $(k+3)/3$, while the best integral solutions takes all odd-numbered paths, and has cost $(k+1)/2$. As k tends to ∞, the ratio between the integral and fractional optimum tends to 3/2. As an aside, we found the above integrality gap instance by translating a known integrality-gap instance of the tree-augmentation problem in caterpillar graphs; see [8].

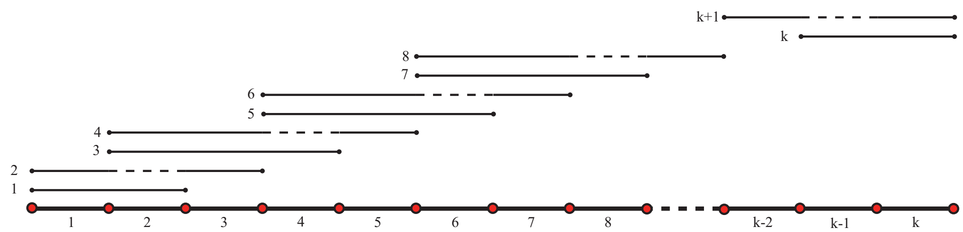


Figure 2: Integrality Gap for PLC

We now show that the integrality gap of the canonical LP for PLC is bounded by 2. We describe a simple primal-dual algorithm that constructs a feasible line cover solution and a feasible dual solution, and the cost of the former is at most twice the value of the dual solution.

The algorithm maintains a set of segments Q. Call an edge e *unsatisfied* if no segment in Q covers e. Let U be the set of unsatisfied edges. Initially Q is the empty set and $U = E$. We grow duals y_e on certain edges, as specified below. We let E_+ denote the edges with positive y_e; we call such edges, *positive* edges. Initially E_+ is empty. Call a segment j *tight* if $\sum_{e \in j : j \textit{ covers } e} y_e = c_j$. We use the terminology an edge e is larger than f, if $\pi_e \geq \pi_f$.

Primal-Dual Algorithm

1. While U is not empty do

   - Breaking ties arbitrarily, pick the largest edge e in U.
   - Increase y_e till some segment becomes tight. Note that each such segment must contain e. Let $j_l(e)$ and $j_r(e)$ be the tight segments that have the smallest left-end-point and the largest right-end-point, respectively. Since e is chosen to be the largest uncovered edge, any unsatisfied edge contained in the two segments $j_l(e)$ or $j_r(e)$ is also covered. We say e is responsible for $j_l(e)$ and $j_r(e)$.

     Add $j_l(e), j_r(e)$ to Q. Add e to E_+. Remove all the unsatisfied edges contained in either $j_l(e)$ or $j_r(e)$ from U.

2. **Reverse Delete:** Scan the segments j in Q in the reverse order in which they were added, and delete j if its deletion doesn't lead to uncovered edges.

It is clear that the final set Q is feasible. It is also clear that y forms a feasible dual. The factor 2-approximation follows from the following lemma by a standard relaxed complementary slackness argument, and this finishes the proof of Theorem 3.

Lemma 5. *Any edge $e \in E_+$ is covered by at most two segments in Q.*

Proof. Suppose there is an edge $e \in E_+$ covered by three segments j_1, j_2 and j_3. Observe that one of the segments, say j_2, must be completely contained in $j_1 \cup j_3$. Since j_2 is not deleted from Q, there must be an edge $f \in j_2$ such that j_2 is the only segment in Q covering f. Since j_1 and j_3 don't cover f, but one of them, say j_1 contains it, this implies $\pi_f > \sup_{j_1} \geq \pi_e$. That is, f is larger than e.

If f is the edge responsible for j_2, then since j_2 contains e, e wouldn't be in E_+. Since f is larger than e, there must be a segment j in Q added before j_2 that covers f. In the reverse delete order, j_2 is processed before j. This contradicts that j_2 is the only segment in Q covering f. □

Lemma 6. $\sum_{j \in Q} c_j \leq 2 \sum_{e \in E} y_e$.

Proof. Since each $s \in Q$ satisfies $\sum_{e \in j : j \text{ } covers \text{ } e} y_e = c_j$, we get

$$\sum_{j \in Q} c_j = \sum_{j \in Q} \sum_{e \in j : j \text{ } covers \text{ } e} y_e = \sum_{e \in E} y_e \cdot |\{j \in Q : j \text{ } covers \text{ } e\}| \leq 2 \sum_{e \in E} y_e$$

□

3.2 An Exact Algorithm for PLC

We first describe the sketch of the algorithm; the full proof starts from Section 3.2.1. A segment j covers only a subset of edges it contains. We call a contiguous interval of edges covered by j, a *valley* of j. The uncovered edges form *mountains*. Thus a segment can be thought of as forming a series of valleys and mountains.

Given a solution $S \subseteq \mathcal{S}$ to the PLC (or even a PTC) instance, we say that segment $j \in S$ is *needed* for edge e if j is the unique segment in S that covers e. We let $E_{S,j}$ be the set of edges that need segment j. We say a solution is *valley-minimal* if it satisfies the following two properties: (a) If a segment j is needed for edge e that lies in the valley v of j, then no higher supply segment of S intersects this valley v, and (b) every segment j is needed for its last and first edges. We show that an optimum solution can be assumed to be valley-minimal, and thus it suffices to find the minimum cost valley-minimal solution.

The crucial observation follows from properties (a) and (b) above. The valley-minimality of solution S implies that there is a unique segment $j \in S$ that covers the first edge of the line. At a very high level, we may now use j to decompose the given instance into a set of *smaller* instances. For this we first observe that each of the remaining segments in $S \setminus \{j\}$ is either fully contained in the strict interior of segment j, or it is disjoint from j, and lies to the right of it. The set of all segments that are disjoint from j form a feasible solution for the smaller PLC instance induced by the portion of the original line instance to the right of j. On the other hand, we show how to reduce the problem of finding an optimal solution for the part of the line contained in j to a single shortest-path computation in an auxiliary digraph. Each of the arcs in this digraph once again corresponds to a smaller sub-instance of the original PLC instance, and its cost is that of its optimal solution. The algorithm follows by dynamic programming.

3.2.1 Valley-Minimal Solutions

As mentioned above, it helps to think of supplies and demands as *heights*. In the case of PLC, the demands of the edges in E form a terrain, and each segment $j \in \mathcal{S}$ corresponds to a straight line at height s_j. Segment j then covers edge e if e lies in the segment's *shadow*, that is, the height of e is smaller than the height of the segment.

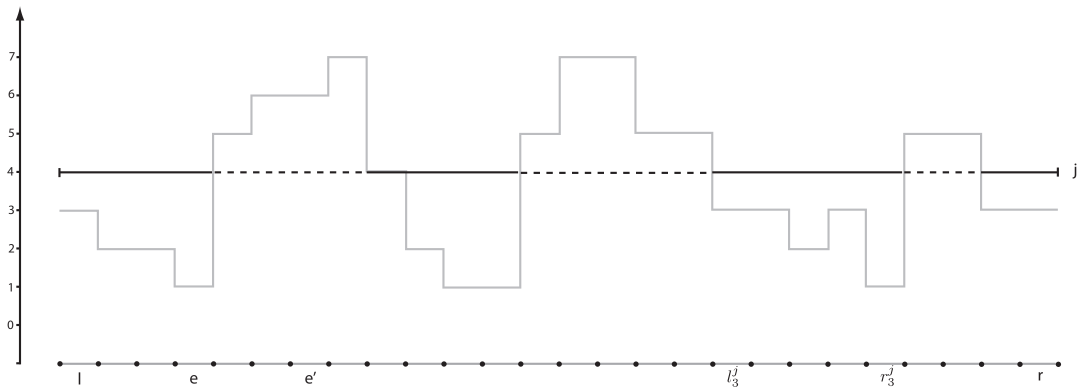


Figure 3: The figure shows a segment j, and the terrain induced by the edges of E that it contains. The terrain partitions j into valleys and mountains. Valleys are indicated by solid parts of j, and mountains are shown as dashed lines.

Figure 3 illustrates this with path P and its edges. The light gray terrain indicates the demands of the edges. The segment j shown in the picture covers the edges in $[l, r]$ that lie in its shadow; e.g., j covers edge e but not e'. The terrain partitions j naturally into *valleys* – contiguous sub-intervals of $[l, r]$ that are in the shadow of j, and *mountains* – those sub-intervals that are contained in $[l, r]$ and consist entirely of edges that are not covered by j. The parts of j that correspond to mountains are indicated by dashed lines, and valleys are depicted by solid lines. In the following, we let $[l_k^j, r_k^j]$ be the interval corresponding to the kth valley of j.

In the following, we will assume that the set of segments $\mathcal{S}$ in the given PLC/PTC instance is *segment-complete*; i.e., if $\mathcal{S}$ contains the segment j then it also contains all proper sub-segments. For example, if a PLC instance contains segment j corresponding to interval $[l^j, r^j]$, then it also contains segments corresponding to intervals $[l, r]$ for all $l^j \leq l \leq r \leq r^j$. This assumption is w.l.o.g. as we can always add a *dummy* sub-segment j' for any such interval $[l, r]$ with the same supply and cost as j. Any minimal solution clearly uses at most one of j and j', and if j' is used, then replacing it with j does not affect feasibility.

Let $S \subset \mathcal{S}$ be an inclusion-wise minimal solution for the given instance, and let $j \in S$ be any one of its segments. We say that j is *needed* for edge $e \in E$ if j covers e, and if there is no other segment in S that covers e; let $E_{S,j}$ be the set of edges that need j, and hence $E_{S,j} \neq \varnothing$ for all $j \in S$. Thus, if j is needed for e, then e is in one of j's valleys; we let val_e^j be that valley.

A solution $S \subseteq \mathcal{S}$ is *valley-minimal* if

[M1] for all $j \in S$ and for all $e \in E_{S,j}$, no segment of higher supply in S covers any of the edges in val_e^j, and

[M2] each segment is needed for its first and last edge.

We obtain the following observation.

Lemma 7. *Given a feasible instance of PLC/PTC, there exists an optimum feasible solution that is valley-minimal.*

Proof. First, it is not too hard to see that we can always obtain an optimal solution that satisfies [M2]. If S is an optimum solution with a segment j, and j is not needed for its first or last edge e, then we may clearly replace j by the sub-segment $j - e$. This does not increase the solutions cost, using the segment-completeness.

Assume, for the sake of contradiction that S violates [M1]. For a solution $S \subseteq \mathcal{S}$, say that (j, j', e) is a *violating triple* if $j, j' \in S$, j' has higher supply than j, j is needed for e, and j' covers some edge in val_e^j. Choose a solution S with the smallest number of violating triples and let (j, j', e) be one such triple. Since j is needed for e, edge e is not contained in j', and hence j' is either fully contained in the interval $(e, n]$ or fully contained in the interval $[1, e)$. Using the segment-completeness assumption, we may replace j' by the sub-segment j'' obtained by removing the prefix consisting of edges in val_e^j; remove j'' if it is empty. The resulting set of segments has cost at most that of S, and the number of violating triples is smaller; a contradiction. □

In the next subsection, we show how we can compute the minimum cost valley-minimal solution for PLC instances in polynomial time using dynamic programming.

3.2.2 Computing valley-minimal solutions

Given $1 \leq l \leq r \leq n$, we obtain the sub-instance *induced* by interval $[l, r]$ by restricting the line $[1, n]$ to this interval, and by keeping only segments that are fully contained in $[l, r]$. Observe that the valley-completeness assumption implies that any such sub-instance is feasible. We begin by making a crucial observation that will allow us to decompose a given PLC instance into *smaller* instances. Let S be a valley-minimal solution for the sub-instance induced by $[l, r]$, and note that [M2] implies that S contains a unique segment j that covers the first edge $(l, l+1)$. Suppose that $E_{S,j} = \{e_1, \ldots, e_k\}$ is the set of edges within $[l, r]$ that need segment j. Abusing notation slightly, we let $\text{val}_i^j = [l_i^j, r_i^j]$ be the valley of j around edge e_i; thus we clearly have

$$E_{S,j} \subseteq \text{val}_1^j \cup \ldots \cup \text{val}_k^j. \tag{16}$$

Note that segment j may have valleys that entirely consist of edges that do not need j; accordingly, such valleys are not part of the list on the right-hand side of (16). Using property [M2], however, we may assume that val_1^j and val_k^j are the first and last valley, respectively, of segment j. We obtain the following observation, where we let $l_{k+1}^j = r + 1$.

Observation 1. *We may assume, for all $1 \leq i \leq k$, if $j' \in S$ contains $e \in (r_i^j, l_{i+1}^j)$, then j' is fully contained in (r_i^j, l_{i+1}^j).*

Proof. Consider first a segment $j' \in S$ with supply bigger than s_j. In this case [M1] implies that j' must have an empty intersection with the valleys $\text{val}_1^j, \ldots, \text{val}_k^j$, and the observation follows.

On the other hand if segment j' has supply at most s_j, then since j' must be needed for some edge e, j must not contain e implying j' must have its right end-point in $(r_k^j, r]$. Replacing j' by its intersection with $(r_k^j, r]$ completes the observation. □

We now let $\texttt{OPT}_{l,r}$ be a minimum cost valley-minimal feasible solution for the sub-instance induced by interval $[l, r]$, and we let $\texttt{opt}_{l,r}$ be its cost. Clearly, $\texttt{OPT}_{n,n}$ consists of the minimum cost segment in $\mathcal{S}$ that covers edge n, and $\texttt{OPT}_{1,n}$ is the optimum solution we want to obtain. Suppose that we know $\texttt{OPT}_{l',r'}$ for all $l < l' \leq r' < r$. The high level idea is the following. The

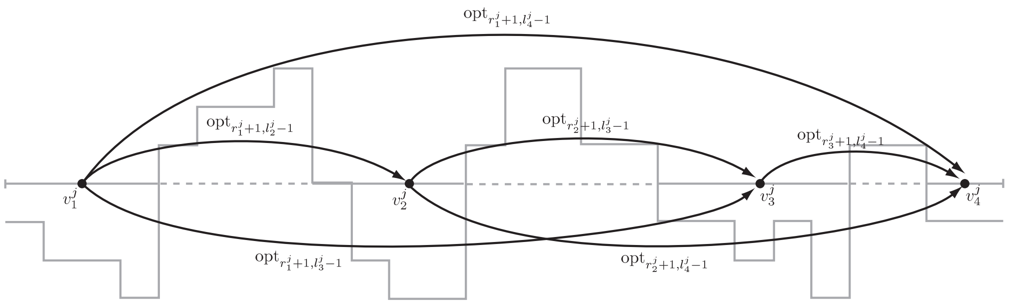


Figure 4: The part of digraph G_l corresponding to segment $j \in \mathcal{S}_l$.

algorithm guesses the first segment j in $\texttt{OPT}_{l,r}$. Suppose that $r' \leq r$ is the rightmost edge covered by j. Observation 1 allows us to partition the remaining segments in $\texttt{OPT}_{l,r}$ into two parts:

Part 1 Segments that contain edges in $(r', r]$. None of these segments can contain any of the edges in $[l, r']$ by the observation.

Part 2 Segments that contain edges in $[l, r']$. Once again, the observation implies that such segments must be fully contained in (l, r').

The first part's solution is obtained since it is a smaller subproblem, the second part is obtained via a shortest-path computation. We now elaborate and give the complete algorithm.

Let $\mathcal{S}_l$ be the segments in $\mathcal{S}$ with leftmost endpoint l. We construct a digraph G_ℓ as follows. Consider a segment $j \in \mathcal{S}_l$, and let

$$[l_1^j, r_1^j], \ldots, [l_k^j, r_k^j]$$

be the set of its valleys. We add a node v_q^j for each valley $1 \leq q \leq k$ of j to G_ℓ. We also add an arc $(v_q^j, v_{q'}^j)$ for all $1 \leq q < q' \leq k$. A shortest path corresponding to the solution $\texttt{OPT}_{l,r}$ will use arc $(v_q^j, v_{q'}^j)$ if

(i) j is the leftmost segment in $\texttt{OPT}_{l,r}$, and

(ii) val_q^j and $\mathrm{val}_{q'}^j$ are two consecutive valleys of j that contain edges that need j.

Observation 1 then states that $\texttt{OPT}_{l,r}$ uses segments that are entirely contained in $(r_q^j, l_{q'}^j)$ to cover $(r_q^j, l_{q'}^j)$. An optimum set of such segments is given by $\texttt{OPT}_{r_q^j+1, l_{q'}^j-1}$, and we therefore give arc $(v_q^j, v_{q'}^j)$ cost $\texttt{opt}_{r_q^j+1, l_{q'}^j-1}$. Figure 4 shows the part of G_ℓ for the segment s from Figure 3.

We add a source node s_l and arcs (s_l, v_1^j) of cost c_j for each of the segments $j \in \mathcal{S}_l$. A shortest path uses such an arc if j is the unique segment starting at l in the corresponding optimum solution. We also add a sink node t_r and add an arc (v_k^j, t_r) for all $j \in \mathcal{S}_l$ of cost $\texttt{opt}_{r_k^j+1,r}$ indicating the optimum PLC for the sub-interval $[r_k^j + 1, r]$. Note that if $r_k^j = r$, then this arc is a loop of cost 0 and can be discarded.

It follows from the above construction that $\texttt{opt}_{l,r}$ is equal to the cost of a shortest s_l, t_r-path in G_l. Each of the shortest-path computations can clearly be done in polynomial time, and hence $\texttt{opt}_{l,r}$ can be obtained via dynamic programming, in polynomial time. This yields the following restatement of Theorem 5.

Theorem 10. *The cost* $\texttt{opt}_{1,n}$ *of an optimum solution for a given PLC instance can be computed in polynomial time.*

4 Priority tree cover

We first give a proof of Theorem 6, and show that rooted PTC is APX-hard, even if all segments have unit cost. Subsequently, we present a 2-approximation algorithm for the problem, by reducing it to an auxiliary instance of the tree augmentation problem. Then, we prove Theorem 4, and show that the integrality gap of the canonical LP formulation of unweighted PTC is bounded by 6. Finally, we prove the connection between PTC and the rectangle cover problem.

4.1 APX-hardness

We prove APX-hardness of PTC via a reduction from the minimum vertex cover problem in bounded degree graphs. The latter problem is known to be APX-hard [3]. Given a bounded degree graph $G(V, E)$, with n vertices and $m = O(n)$ edges, let the edges be arbitrarily numbered $\{1, 2, \dots, m\}$.

The tree in our instance has a broom structure: it has a *handle* which is a path of m edges $(e_1, \dots, e_m)$ given by vertices $\{x_0, x_1, \dots, x_m\}$, and it has n *bristles* where each bristle corresponds to a particular vertex $v \in V$ and is a path of length $deg(v)$. The edge e_i in the handle for $1 \leq i \leq m$, corresponds to the edge numbered i in the graph G. The bristle corresponding to vertex v is a path $(f_1^v, f_2^v, \dots, f_{deg(v)}^v)$ given by the vertices $\{x_m, y_1^v, y_2^v, \dots, y_{deg(v)}^v\}$. The root of the tree is x_0, the end point of the handle. Thus the tree has $m + \sum_v deg(v) = 3m$ edges.

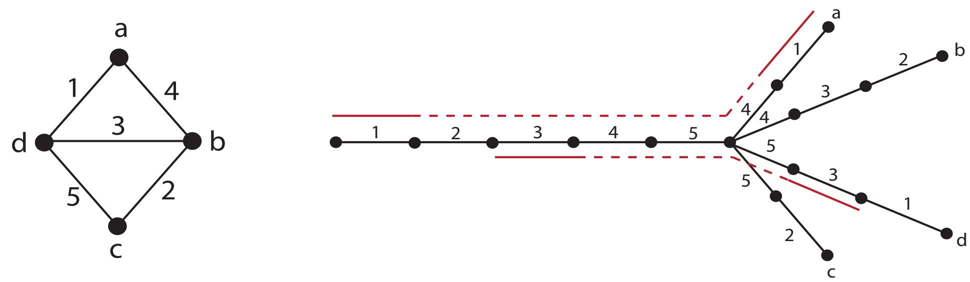
We now describe the priority demands of these tree edges. The demand of edge e_i is i. Consider the edges in G incident on v in the decreasing order of their numbers. Suppose they are $(i_1 > i_2 > \cdots > i_{deg(v)})$. The demands of the edge f_j^v is i_j. Thus, for a particular bristle corresponding to a vertex v, the demands decrease as we go from f_1^v to $f_{deg(v)}^v$, and these demands correspond to the numbers of edges incident on v.

Now we describe the segments. All segments have unit cost. We have two kinds of segments: edge segments and vertex segments. For every edge $i = (v, w)$ in E, there are two edge segments s_v^i and s_w^i. Segments s_v^i contains all edges e_i to e_m and edges f_1^v to f_j^v, where edge i is the jth edge in the descending order of neighbors of v in G. The supply of segment s_v^i is i, and thus by construction, we see that s_v^i only spans edge e_i and f_j^v. That completes the description of edge segments. For every vertex v, there is a vertex segment t_v that covers all the edges in the bristle corresponding to vertex v. That completes the description of the PTC instance. Look at figure 5 for an illustration of the reduction.

The following lemma along with the APX-hardness of the vertex cover problem in bounded degree graphs, and the fact that in the latter any vertex cover is of size $\Omega(n)$, leads to the APX-hardness of the PTC problem.

Lemma 8. *The optimum PTC of the above instance is* $m+k$*, where* k *is the size of the optimum vertex cover of* G*.*

Proof. Firstly note that we may assume that in any optimal PTC, for any edge $i = (v, w)$, we will have exactly one of s_v^i or s_w^i in the solution. We need to have one since these are the only two segments that cover edge e_i in the tree. Instead of picking both, we can remove one, say s_w^i, from the solution and pick the corresponding vertex segment t_w instead, at no increase of cost. Therefore, there are exactly m edge segments picked in any optimal PTC solution.



(i) (ii)

Figure 5: (i) shows an instance of the vertex cover problem, and (ii) is the corresponding PTC instance. The numbers on the edges are the priority demands corresponding to the edge numbers in the graph. Figure (ii) also shows two segments: s_a^1 and s_d^3, having supplies 1 and 3 respectively. Dashed line means that these segments do not have enough supply to cover the edges.

Now note that these m edge segments uniquely correspond to an orientation of the edges in G; if for edge $i = (v, w)$, s_v^i is chosen in the solution, the edge (v, w) is oriented from w to v. In this orientation, if there is a *sink* (a vertex with all edges incident to it) v, then note that all the edges in the bristle corresponding to v have also been covered. Thus, the number of vertex segments required to cover the remaining edges of the tree, is precisely the number of *non-sinks* in this orientation. In particular, the optimal PTC corresponds to the orientation that minimizes the number of non-sinks.

The proof is complete by noting that non-sinks form a vertex cover; this is because each edge is oriented away from some non-sink, and is thus incident to it. Furthermore, given a vertex cover, there exists an orientation with precisely these vertices as non-sinks. Orient the edges towards the complement of the vertex cover (the independent set) - the complement is precisely the set of sinks, and thus the vertex cover is precisely the set of non-sinks. □

Proof of Theorem 6. Suppose the degrees of G are all B, a constant. Note that the vertex cover of this graph is at least $m/B = n/2$. The APX-hardness implies that it is NP-hard to distinguish between the case when the vertex cover is $c_1 n$ or $c_2 n$ where $c_2 > c_1 \geq 1/2$ are certain constants.

The above lemma therefore implies it is NP-hard to distinguish between the cases when the optimum of a PTC is $m + c_1 n = (c_1 + B/2)n$ and when the optimum is $m + c_2 n = (c_2 + B/2)n$. Since B, c_1, c_2 are constants, we get the APX-hardness.

(For the interested reader: the APX-hardness of vertex cover of bounded degree graphs by Berman and Karpinski [3] gives $B = 4$, $c_1 = 78/152$ and $c_2 = 79/152$, showing it is NP-hard to approximate to a factor better than 1.002.) □

4.2 An approximation algorithm for PTC

The crucial idea is the following. Given an optimum solution $S^* \subseteq \mathcal{S}$, we can partition the edge-set E of T into disjoint sets $E_1, \ldots, E_p$, and partition two copies of S^* into $S_1, \ldots, S_p$, such that E_i is a path in T for each i, and S_i is a priority line cover for the path E_i. Once again, we assume without loss of generality that the instance is segment-complete.

In particular, we prove the following lemma. Let $\hat{E}_{S^*,j}$ be the set of edges e such that j

is the segment with the highest supply, among all segments in S^* that cover e. Note that the union of all $\hat{E}_{S^*,j}$, over all $j \in S^*$, partitions E. Also note that for each edge e, there is a unique segment j such that $e \in \hat{E}_{S^*,j}$. If there were two, we could replace one of the segments by a sub-segment and still stay feasible. We call the segment j *responsible* for e.

Lemma 9. *Given an optimal solution $S^* \subseteq \mathcal{S}$ to a PTC instance with tree $T = (V, E)$, there is a partition*

$$E_1 \cup \ldots \cup E_p = E,$$

where each E_i is the edge set of a path in T such that for all $j \in S^*$, $\hat{E}_{S^*,j} \cap E_i \neq \varnothing$ for at most two $i \in \{1, \ldots, p\}$.

Using this, we describe the 2-approximation algorithm which proves Theorem 7.

Proof of Theorem 7. For any two vertices t (top) and b (bottom) of the tree T, such that t is an ancestor of b, let P_{tb} be the unique path from b to t. Note that P_{tb}, together with the restrictions of the segments in $\mathcal{S}$to P_{tb}, defines an instance of PLC. Therefore, for each pair t and b, we can compute the optimal solution to the corresponding PLC instance; let the cost of this solution be c'_{tb}. Create an instance of the 0,1-tree cover problem with T and segments $\mathcal{S}' := \{(t, b) : t \text{ is an ancestor of } b\}$ with costs c'_{tb}. Solve the 0,1-tree cover instance exactly (recall we are in the rooted version) and for the segments (t, b) in $\mathcal{S}'$ returned, return the solution of the corresponding PLC instance of cost c'_{tb}. We now use Lemma 9 to obtain a solution to the 0,1-tree cover problem $(T, \mathcal{S}')$ of cost at most 2 times the cost of S^*. This will prove the theorem.

For each E_i, let t_i and b_i be the end points of E_i with t_i being the ancestor of b_i. Since E_i's partition the edges, the segments $(t_i, b_i) : i = 1, \ldots, p$ is a feasible 0,1-tree cover for $(T, \mathcal{S}')$. Define $S_i := \{j \in S^* : e \in E_i \cap \hat{E}_{S^*,j}\}$ to be the set of segments responsible for the edges in E_i. By definition, S_i is a PLC for E_i. Thus, the cost of the segments in S_i is at least $c'_{t_i b_i}$. Furthermore, Lemma 9 implies that the total cost of the segments in S_i is at most twice the cost of segments in S^*. Therefore, the cost of the feasible solution to the cover problem in $(T, \mathcal{S}')$ is at most twice the cost of segments in S^*.

□

Proof of Lemma 9. We give an algorithm to compute the decomposition. Let e be any of the edges incident to the root of T, and let $j_1 \in S^*$ be the highest-supply segment covering e. We then let E_1 be the edges of the path in T corresponding to j_1. Removing E_1 from T yields sub-trees $T_1, \ldots, T_q$. For each tree T_i we repeat the above steps, and let

$$E_1, \ldots, E_p \tag{17}$$

be the final partition; let $j_i \in S^*$ be the segment corresponding to edge-set E_i. Note that for $q < q'$, $\hat{E}_{S^*,j_q} \cap E_{q'}$ is empty. This is because $E_{q'}$ is a subset of edges which are not in $j_{q'-1}, \ldots, j_1$.

Consider a segment $j \in S$, and let $1 \leq i \leq p$ be smallest such that $\hat{E}_{S^*,j} \cap E_i \neq \varnothing$, and assume that $\hat{E}_{S^*,j} \cap E_q \neq \varnothing$ for some $i < q \leq p$; choose q smallest with this property. We claim that $j_q = j$, and hence for all $q < q' \leq p$ we have $\hat{E}_{S^*,j} \cap E_{q'} = \emptyset$. Thus, $\hat{E}_{S^*,j}$ has non-empty intersection only with E_i and E_q.

Let $e \in E_{S^*,j} \cap E_i$, and let $f \in E_{S^*,j} \cap E_q$ be two edges in different parts of the partition such that j is responsible for both. As both e and f are edges on j, and since $i < q$, it follows that f is a descendant of e in tree T. Let g be the topmost edge of E_q; clearly, g is on the e, f-path in T. By the decomposition algorithm, segment j_q is the highest-supply segment covering edge g. As j contains g, this means that the supply of j_q is at least that of j. Finally, since f is on j_q, j_q covers f as well. But this means that $j_q = j$ as j is responsible for f. □

4.3 Canonical LP relaxation of PTC: Integrality Gap

In this section, we prove Theorem 4, by showing that the canonical LP relaxation of unweighted PTC is at most 6. Recall the PTC LP.

$$\min \quad \left\{ \sum_{s \in \mathcal{S}} c_s x_s : \quad \forall e \in E : \sum_{s:s \ covers \ e} x_s \geq 1; \quad x_s \geq 0, \forall s \in \mathcal{S} \right\} \tag{18}$$

Proof of Theorem 4. The idea of the proof is the following: as in the factor 2-approximation for PTC, we decompose the edge set of the tree into disjoint sets $E_1, \cdots, E_p$, such that each E_i induces a path. We will abuse notation and refer to the E_i's as paths. Furthermore, we take any feasible solution x of (18) and obtain p fractional solutions $x^{(1)}, \ldots, x^{(p)}$ such that $x^{(i)}$ is a feasible fractional solution to (Primal) for the PLC instance on the path E_i. We will guarantee that

$$\sum_{i=1}^{p} \sum_{j \in \mathcal{S}} x_j^{(i)} \leq 3 \sum_{j \in \mathcal{S}} x_j.$$

The theorem then follows from Theorem 3.

Unlike in the argument used in the previous section where the decomposition into paths depended on S^*, the decomposition into disjoint paths that we use here is universal. Each path E_i will end at a unique leaf, and p in (17) will now be the number of leaves of T. Let E_1 be *any* path from the root to a leaf. Delete E_1 from the tree to get a series of sub-trees. Recursively, obtain E_2 to E_p. We call a path E_i a *child* of E_q, if the starting point of E_i lies on E_q.

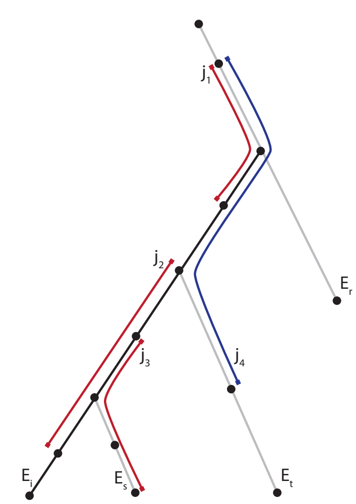


Figure 6: The figure shows a fragment E_i, its parent E_r, and two children E_s and E_t. The segments j_1, j_2, and j_3 are local for E_i, and segment j_4 is global. In particular, j_4 is an i,t-global segment.

Let x be any feasible fractional solution of (18) and let S^* be the support of x, that is, $S^* = \{j : x_j > 0\}$. Fix a path E_i and say that a segment $j \in S^*$ *intersects* E_i if j covers an edge in E_i A segment j that intersects E_i is called *local* for E_i if either the first or the last edge covered by j lies in E_i. A segment j that intersects E_i is called *global* for E_i, otherwise. Figure 6 illustrates this.

Let j be a global segment for E_i, and let e be the first edge contained in j *after* E_i. If $e \in E_q$, we call s an *iq-global* segment. Observe that E_q is a child of E_i. Thus an *iq*-global segment *enters* E_i and *exits* via E_q. Note that *iq*-global segments, over all q such that E_q is a child of E_i, partition all global segments for E_i. Also note that an *iq*-global segment could also be a $i'q'$-global segment for some other i', q'.

Now we are ready to define the fractional solution $x^{(i)}$ that will be feasible for (Primal) for the PLC instance on E_i. Firstly for all segments j that are local for E_i, let $x_j^{(i)} = x_j$. Next, we take care of segments that are global for E_i. For each child E_q of E_i, order all the iq-global segments in non-increasing order of supply: $\{j_1, \ldots, j_r\}$. Let l be such that

$$x_{j_1} + \cdots + x_{j_l} \leq 1 \text{ and } x_{j_1} + \cdots + x_{j_l} + x_{j_{l+1}} > 1$$

If no such l exists, then $l = r$. Define $x_{j_k}^{(i)} = x_{j_k}$ for $1 \leq k \leq l$. If $l < r$, then let $x_{j_{l+1}}^{(i)} = 1 - \sum_{k=1}^{l} x_{j_k}^{(i)}$.

Claim 4. *$x^{(i)}$ is feasible for* (Primal) *for the PLC instance on E_i.*

Proof. Pick any edge $e \in E_i$. Look at all segments $j \in S^*$ that cover e. These segments are either local for e or global for e. If j is local, there is a corresponding segment in the support $x^{(i)}$ of the same value. Furthermore for any q,

$$\sum_{j:j \text{ is } iq\text{-global}, j \text{ covers } e} x_j^{(i)} \geq \min\{1, \sum_{j:j \text{ is } iq\text{-global}, j \text{ covers } e} x_j\}$$

In any case, e is covered by $x^{(i)}$ at least to the extent it is covered by x, which implies $x^{(i)}$ is feasible. ☐

Lemma 10. $\sum_{i=1}^{p} \sum_{j \in \mathcal{S}} x_j^{(i)} \leq 3 \sum_{j \in \mathcal{S}} x_j$

Proof. Each segment $j \in S^*$ is local for at most two paths E_i and E_q. Thus the contribution to the LHS by local segments for some path E_i is exactly $2 \sum_{j \in \mathcal{S}} x_j$.

Furthermore, for every parent-child pair E_i and E_q that induces an iq-global segment for E_i, we increase the LHS by at most 1. The number of such pairs is at most the number of leaves in T. The proof is complete by noting that $\sum_{j \in \mathcal{S}} x_j$ is at least the number of leaves in T. ☐

To complete the proof of the theorem, note that from Theorem 3 we know there exists for each E_i, a set of segments S_i such that $|S_i| \leq 2 \sum_{j \in \mathcal{S}} x_j^{(i)}$. The union of all such S_i forms a valid PTC of cardinality at most $6 \sum_{j \in \mathcal{S}} x_j$. ☐

4.4 Priority Tree Cover and Geometric Covering Problems

In this section, we show that the PTC problem is a special case of covering a set of points in 3-dimension by axis-parallel rectangles (cuboids). In particular we prove Theorem 8. We go in two steps. We first define a problem, that we call 2-Priority Line Cover and show that the PTC problem is a special case of 2-PLC. Subsequently, we show 2-PLC is a special case of 3-dimensional rectangle cover. We start with a definition of 2-PLC.

2**-Priority Line Cover (2-PLC).** The input is a line $T = (V, E)$, and a collection of segments $\mathcal{S} \subseteq V \times V$ with costs c_j for each $j \in \mathcal{S}$. Furthermore, each segment j has a priority supply vector in *two* dimensions, denoted as (s_j^1, s_j^2), and each edge e has a priority demand vector in two dimensions, denoted as (π_e^1, π_e^2). A segment j covers e iff j contains e and $s_j^i \geq \pi_e^i$ for both $i = 1, 2$. The goal is to find the minimum cost collection of segments that cover every edge.

It is easy to see that PLC is a special case of 2-PLC. Somewhat surprisingly, PTC is a special case of 2-PLC as well.

Lemma 11. *Any instance of PTC can be encoded as an instance of 2-PLC with the same solution set.*

Proof. Given a rooted tree $T = (V, E)$, we perform two different depth first traversals to get two different orderings on the edges E. One such ordering will define the line of the 2-PLC instance, the other will define the first coordinates of the priority demand vectors of the edges.

In a depth first traversal of a tree, at every step we move from a vertex to one of its children, if any. Our two different traversals will be defined by two different choices of moving to a child-vertex. For every vertex v of the tree, consider a total order σ_v on its children. One such order that is convenient to keep in mind is the following; given a drawing of the tree, the total order of the children is from left to right. Let σ_v^R be the *opposite* total order. The two depth first traversals are obtained by running with σ_v's and σ_v^R's, respectively. Figure 7 illustrates the two orders with the ordering σ_v at every vertex v being from left-to-right, and σ_v^R being from right-to-left.

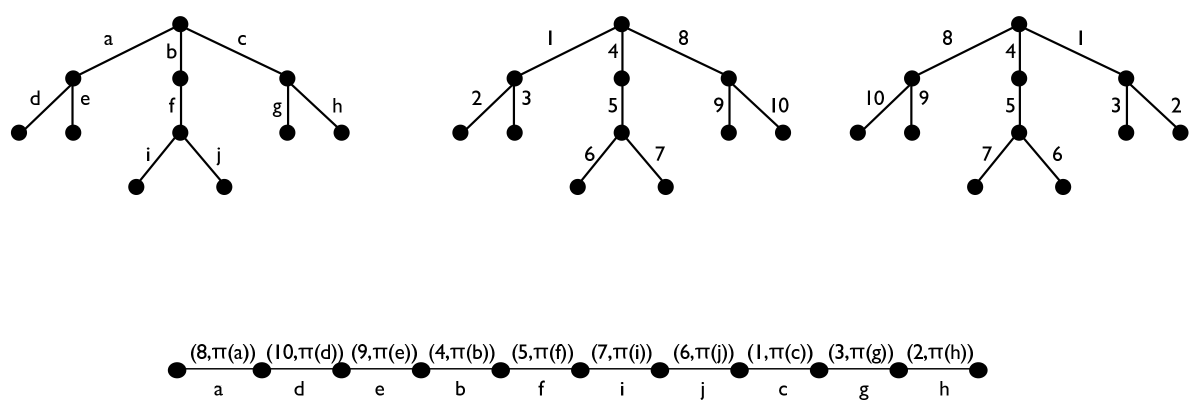


Figure 7: The left most tree is the original tree, the second and third are the two depth first traversals. The line below shows the line in the 2-PLC instance.

Let the two traversals return orderings μ and μ^R on the edges of the tree. The crucial observation is the following: for any vertex v, let $(v_1, \ldots, v_k)$ be the children in the σ_v order; then $\mu(v, v_1) < \mu(v, v_2) < \cdots < \mu(v, v_k)$, and thus, $\mu^R(v, v_1) > \cdots > \mu^R(v, v_k)$.

Now we are ready to describe the 2-PLC instance. The line is defined by the edges of the tree ordered w.r.t. μ. That is, the order of the edges is $(e_1, \ldots, e_m)$ such that $\mu(e_1) < \mu(e_2) < \cdots < \mu(e_m)$. The priority demand vector of an edge e of the tree is $(\mu^R(e), \pi_e)$. Consider a segment $j = (u, v)$ such that u is a descendant of v in the PTC instance. We identify two specific tree edges contained in j: the parent-edge (u, u') of u, and the edge (v, v') between node v and its unique child v' that is on the u, v-path in T. By the depth-first property, we get $\mu(v, v') \leq \mu(u, u')$. The corresponding segment in the 2-PLC instance, also denoted as j, contains all the edges from $\mu(v, v')$ to $\mu(u, u')$. The priority supply vector of j is $(\mu^R(u, u'), s_j)$.

Claim 5. *For any segment j, the set of edges covered by j in the 2-PLC instance is precisely the set of edges covered in the PTC instance.*

Proof. Let e be an edge covered by j in the PTC instance. Since e is contained in the path from u to v in the tree, by property of depth first traversals we get, $\mu(v,v') \leq \mu(e) \leq \mu(u,u')$ and $\mu^R(e) \leq \mu^R(u,u')$. The first pair of inequalities implies e lies in the segment j in the 2-PLC instance, the second implies that $\pi^1_e \leq s^1_j$. Since e is covered by j in the PTC, we also get $\pi^2_e = \pi_e \leq s_j = s^2_j$. Thus, e is covered by j in the 2-PLC instance.

Let e be an edge covered by j in the 2-PLC instance. Since e lies in j, we conclude $\mu(v,v') \leq \mu(e) \leq \mu(u,u')$. This implies either (a) e lies on the path from u to v in the tree, or, (b) there is a node w on the u,v-path in the tree, and a child z of w that is not on this path such that e is contained in the subtree defined by edge (w,z).

Note, that in case (b) the depth-first traversal for order σ visits edge (z,w) *before* edge (u,u'). This implies that the second dfs traversal for order σ^R visits (z,w) *after* (u,u'). Since (z,w) is visited before e in both traversals, we must therefore have $\mu^R(e) > \mu^R(u,u')$, and this implies $s^1_j < \pi^1(e)$ which is impossible since j covers e. Thus, case (b) is not possible, and e lies on the path fro u to v on the tree. Furthermore, we have $s_j = s^2_j \geq \pi^2_e = \pi_e$, and so j covers e in the PTC instance as well. □

□

Now we show that 2-PLC is a special case of 3-dimensional rectangle cover. This is not to hard to see. We assume the edges of the line are numbered $(1,2,\ldots,m)$. For edge e numbered e_i, we associate a point in 3 dimensions with coordinates (i,π^1_e,π^2_e). For each segment $j=(a,b)$, we have a rectangle associated. In fact, these rectangles have are unbounded in the negative y and z coordinates. The other 4 bounding half-spaces are $x \geq a$, $x \leq b$, $y \leq s^1(j)$ and $z \leq s^2(j)$. It is not too hard to see a rectangle corresponding to a segment j contains a point corresponding to an edge e iff j covers e in the 2-PLC instance. This completes the proof of Theorem 8.

5 Concluding Remarks

In this paper we studied column restricted covering integer programs. In particular, we studied the relationship between CCIPs and the underlying 0,1-CIPs. We conjecture that the approximability of a CCIP should be asymptotically within a constant factor of the integrality gap of the original 0,1-CIP. We couldn't show this; however, if the integrality gap of a PCIP is shown to be within a constant of the integrality gap of the 0,1-CIP, then we will be done. At this point, we don't even know how to prove that PCIPs of special 0,1-CIPS, those whose constraint matrices are totally unimodular, have constant integrality gap. Resolving the case of PTC is an important step in this direction, and hopefully in resolving our conjecture regarding CCIPs.

References

[1] E. Balas. Facets of the knapsack polytope. *Math. Programming*, 8:146–164, 1975.

[2] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48(5):1069–1090, 2001.

[3] P. Berman and M. Karpinski. On some tighter inapproximability results. In *Proceedings, International Colloquium on Automata, Languages and Processing*, pages 200–209, 1999.

[4] R. D. Carr, L. K. Fleischer, V. J. Leung, and C. A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 106–115, 2000.

[5] M. Charikar, J. Naor, and B. Schieber. Resource optimization in qos multicast routing of real-time multimedia. *IEEE/ACM Trans. Netw.*, 12(2):340–348, 2004.

[6] C. Chekuri, A. Ene, and N. Korula. Unsplittable flow in paths and trees and column-restricted packing integer programs. In *Proceedings, International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, page (*to appear*), 2009.

[7] C. Chekuri, M. Mydlarz, and F. B. Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Alg.*, 3(3), 2007.

[8] J. Cheriyan, H. Karloff, R. Khandekar, and J. Könemann. On the integrality ratio for tree augmentation. *Operations Research Letters*, 36(4):399–401, 2008.

[9] J. Chuzhoy, A. Gupta, J. Naor, and A. Sinha. On the approximability of some network design problems. *ACM Trans. Alg.*, 4(2), 2008.

[10] G. Dobson. Worst-case analysis of greedy heuristics for integer programming with non-negative data. *Math. Oper. Res.*, 7(4):515–531, 1982.

[11] P. Hammer, E. Johnson, and U. Peled. Facets of regular 0-1 polytopes. *Math. Programming*, 8:179–206, 1975.

[12] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.

[13] S. G. Kolliopoulos. Approximating covering integer programs with multiplicity constraints. *Discrete Appl. Math.*, 129(2-3):461–473, 2003.

[14] S. G. Kolliopoulos and C. Stein. Approximation algorithms for single-source unsplittable flow. *SIAM Journal on Computing*, 31(3):919–946, 2001.

[15] S. G. Kolliopoulos and C. Stein. Approximating disjoint-path problems using packing integer programs. *Math. Programming*, 99(1):63–87, 2004.

[16] S. G. Kolliopoulos and N. E. Young. Approximation algorithms for covering/packing integer programs. *J. Comput. System Sci.*, 71(4):495–505, 2005.

[17] Nitish Korula. private communication, 2009.

[18] S. Rajagopalan and V. V. Vazirani. Primal-dual RNC approximation algorithms for (multi)set (multi)cover and covering integer programs. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, 1993.

[19] A. Schrijver. *Combinatorial optimization.* Springer, New York, 2003.

[20] A. Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM Journal on Computing*, 29(2):648–670, 1999.

[21] A. Srinivasan. An extension of the lovász local lemma, and its applications to integer programming. *SIAM Journal on Computing*, 36(3):609–634, 2006.

[22] L. Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings, ACM Symposium on Theory of Computing*, pages 453–461, 2001.

[23] L. Wolsey. Facets for a linear inequality in 0-1 variables. *Math. Programming*, 8:168–175, 1975.