Approximability of Sparse Integer Programs

David Pritchard

September 13, 2009

Abstract

The main focus of this paper is a pair of new approximation algorithms for sparse integer programs. First, for covering integer programs $\{\min cx : Ax \geq b, \mathbf{0} \leq x \leq d\}$ where A has at most k nonzeroes per row, we give a k-approximation algorithm. (We assume A, b, c, d are nonnegative.) For any $k \geq 2$ and $\epsilon > 0$, if $\mathsf{P} \neq \mathsf{NP}$ this ratio cannot be improved to $k - 1 - \epsilon$, and under the unique games conjecture this ratio cannot be improved to $k - \epsilon$. One key idea is to replace individual constraints by others that have better rounding properties but the same nonnegative integral solutions; another critical ingredient is knapsack-cover inequalities. Second, for packing integer programs $\{\max cx : Ax \leq b, \mathbf{0} \leq x \leq d\}$ where A has at most k nonzeroes per column, we give a $2^k k^2$-approximation algorithm. This is the first polynomial-time approximation algorithm for this problem with approximation ratio depending only on k, for any $k > 1$. Our approach starts from iterated LP relaxation, and then uses probabilistic and greedy methods to recover a feasible solution.

Note added after publication: This version includes subsequent developments: a $O(k^2)$ approximation for the latter problem using the iterated rounding framework, and several literature reference updates including a $O(k)$-approximation for the same problem by Bansal et al.

1 Introduction and Prior Work

In this paper we investigate the following problem: what is the best possible approximation ratio for integer programs where the constraint matrix is sparse? To put this in context we recall a famous result of Lenstra [35]: integer programs with a constant number of variables or a constant number of constraints can be solved in polynomial time. Our investigations analogously ask what is possible if the constraints each involve at most k variables, or if the variables each appear in at most k constraints.

Rather than consider the full class of all integer programs, we consider only packing and covering problems. One sensible reason for this is that *every* integer program can be rewritten (possibly with additional variables) in such a way that each constraint contains at most 3 variables and each variable appears in at most 3 constraints, if mixed positive and negative coefficients are allowed. Aside from this, packing programs and covering programs represent a substantial portion of the literature on integer programs; and sparse programs of this type are interesting in their own right as "multiple-knapsack" problems where each item affects a bounded number of knapsacks, or each knapsack is affected by a bounded number of items.

We use CIP (resp. PIP) as short for *covering* (resp. *packing*) *integer program*, which is any integer program of the form $\{\min cx : Ax \geq b, \mathbf{0} \leq x \leq d\}$ (resp. $\{\max cx : Ax \leq b, \mathbf{0} \leq x \leq d\}$) with A, b, c, d nonnegative and rational. Note that CIPs are sometimes called *multiset multicover* when A and b are integral. We call constraints $x \leq d$ *multiplicity constraints* (also known as *capacity constraints*). We allow for entries of d to be infinite, and without loss of generality, all finite entries of d are integral. An integer program with constraint matrix A is *k-row-sparse*, or *k-RS*, if each row of A has at most k entries; we define *k-column-sparse (k-CS)* similarly. As a rule of thumb we ignore the case $k = 1$, since such problems trivially admit fully polynomial-time approximation schemes (FPTAS's) or poly-time algorithms. The symbol $\mathbf{0}$ denotes the all-zero vector, and similarly for $\mathbf{1}$. For covering problems an *α-approximation algorithm* is one that always returns a solution with objective value at most α times optimal; for packing, the objective value is at least $1/\alpha$ times optimal. We use n to denote the number of variables and m the number of constraints (i.e. the number of rows of A).

1.1 k-Row-Sparse Covering IPs: Previous and New Results

The special case of 2-RS CIP where A, b, c, d are 0-1 is the same as Min Vertex Cover, which is APX-hard. More generally, 0-1 k-RS CIP is the same as k-Bounded Hypergraph Min Vertex Cover (a.k.a. Set Cover with maximum frequency k) which is not approximable to $k-1-\epsilon$ for any fixed $\epsilon > 0$ unless P=NP [12] ($k-\epsilon$ under the unique games conjecture [27]). This special case is known to admit a matching positive result: set cover with maximum frequency k can be k-approximated by direct rounding of the naive LP [20] or local ratio/primal-dual methods [4].

The following results are known for other special cases of k-RS CIP with multiplicity constraints: Hochbaum [17] gave a k-approximation in the special case that A is 0-1; Hochbaum et al. [22] and Bar-Yehuda & Rawitz [5] gave pseudopolynomial 2-approximation algorithms for the case that $k = 2$ and d is finite. For the special case $d = \mathbf{1}$, Carr et al. [8, §2.6] gave a k-approximation, and Fujito & Yabuta [14] gave a primal-dual k-approximation. Moreover [8, 14] claim a k-approximation for general d, but there seems to have been some oversights as the papers do not provide full proofs and their methods alone seem to be insufficient for general d. Briefly, [8] claims $d = \mathbf{1}$ holds without loss of generality, which is true for some of the applications in [8], but seems false for k-RS CIPs; [14] sketches a reduction that seems faulty, and omits many crucial details for lack of space. Our first main result, given in Section 2, is a simple and correct proof of the same claim.

Theorem 1. *There is a polynomial time k-approximation algorithm for k-RS CIPs with multiplicity constraints.*

Our approach is to first consider the special case that there are no multiplicity constraints (i.e. $d_j = +\infty$ for all j); we then extend to the case of finite d via *knapsack-cover inequalities*, using linear programming (LP) techniques from Carr et al. [8]. A $(k+1)$-approximation algorithm is relatively easy to obtain using LP rounding; in order get the tighter ratio k, we replace constraints by other "$\mathbb{Z}_+$-equivalent" constraints (see Definition 5) with better rounding properties. The algorithm requires a polynomial-time linear programming subroutine.

Independently of our work, recent work of Koufogiannakis & Young [31, 34, 32, 33] also gives a full and correct proof of Theorem 1. Their primal-iterative approach works for a broad generalization of k-RS CIPs and runs in low-degree strongly polynomial time. Our approach has the generic advantage of giving new ideas that can be used in conjunction with other LP-based methods, and the specific advantage of giving integrality gap bounds. See Section 2.2 for more details.

1.2 k-Column-Sparse Packing IPs: Previous and New Results

So far, no constant-factor approximation is known for k-CS PIPs, except in special cases. If every entry of b is $\Omega(\log m)$ then randomized rounding provides a constant-factor approximation. *Demand matching* is the special case of 2-CS PIP where (i) in each column of A all nonzero values in that column are equal to one another and (ii) no two columns have their nonzeroes in the same two rows. Shepherd & Vetta [40] showed demand matching is APX-hard but admits a $(\frac{11}{2} - \sqrt{5})$-approximation algorithm when $d = \mathbf{1}$; their approach also gives a $\frac{7}{2}$-approximation for 2-CS PIP instances satisfying (i). Results of Chekuri et al. [11] yield a $11.542k$-approximation algorithm for k-CS PIP instances satisfying (i) and such that the maximum entry of A is less than the minimum entry of b.

The special case of k-CS PIP where A, b are 0-1 is the same as *min-weight k-set packing*, *hypergraph matching with edges of size $\le k$*, and *strong independent sets in hypergraphs with degree at most k*. The best approximation ratio known for this problem is $(k+1)/2 + \epsilon$ [6] for general weights, and $k/2 + \epsilon$ when $c = \mathbf{1}$ [23]. For $k = 3$ a 2-approximation in the weighted case is known due to Lau and Chan citeCL09. The best lower bound is due to Hazan et al. [19], who showed $\Omega(k/\ln k)$-inapproximability unless P=NP, even for $c = \mathbf{1}$.

Our second main result, given in Section 3, is the following result.

Theorem 2. *There is a polynomial time $2^k(k^2 - 2k + 1) + 1$-approximation algorithm for k-CS PIPs with multiplicity constraints.*

Our methodology begins by using *iterated LP relaxation* [41] to find an integral solution with super-optimal value, but violating some constraints in an additively-bounded way. Then we use a combination of probabilistic and greedy methods to recover a high-weight feasible solution. An extension of this methodology gives improved results in two special cases: we get a 4-approximation when $k = 2$, and we get a $(W+k)/(W-k)$-approximation when the program's *width*, defined as $W := \min_{i,j:A_{ij} \neq 0} \frac{b_i}{A_{ij}}$ satisfies $W > k$. These results also require a polynomial-time linear programming subroutine.

Note added after publication: Within several months of the initial release of this paper on the arXiv [39], a flurry of improvements to Theorem 2 were discovered. First, Chekuri, Ene and Korula (personal communication) obtained an $O(k2^k)$ algorithm using randomized rounding instead of iterated rounding. Then, the same group obtained a significant improvement of $O(k^2)$, which was independently matched by the author and D. Chakrabarty. The idea (which was the same for both groups) is a natural modification of the iterated rounding framework, and we present it in Section 3.2. Finally, Bansal, Korula, and Nagarajan [3] gave a simple and elegant $O(k)$-approximation algorithm based on randomized rounding with a careful sort of alteration.

1.3 Other Related Work

Srinivasan [42, 43] showed that k-CS CIPs admit a $O(\log k)$-approximation. Kolliopoulos and Young [29] extended this result to handle multiplicity constraints. There is a matching hardness result: it is NP-hard to approximate k-Set Cover, which is the special case where A, b, c are 0-1, better than $\ln k - O(\ln \ln k)$ for any $k \geq 3$ [44]. Hence for k-CS CIP the best possible approximation ratio is $\Theta(\log k)$. A $(k + \epsilon)$-approximation algorithm can be obtained by separately applying an approximation scheme to the knapsack problem corresponding to each constraint. Hochbaum [21] showed 2-CS CIPs are NP-hard to optimize and gave a bicriteria approximation algorithm. Although 0-1 2-CS CIP is Edge Cover which lies in P, 2-CS CIP in general is NP-hard to $(17/16 - \epsilon)$-approximate, due to methods from [10], even if A has 2 *equal* nonzeroes per column and d is 0-1 or d is all-$+\infty$. See Appendix A for details.

The special case of 2-RS PIP where A, b, c are 0-1 is the same as Max Independent Set, which is not approximable within $n/2^{\log^{3/4+\epsilon} n}$ unless $\mathsf{NP} \subset \mathsf{BPTIME}(2^{\log^{O(1)} n})$ [26]. On the other hand, n-approximation of any packing problem is easy to accomplish by looking at the best singleton-support solution. A slightly better n/t-approximation, for any fixed t, can be accomplished by exhaustively guessing the t most profitable variables in the optimal solution, and then solving the resulting t-dimensional integer program to optimality via Lenstra's result [35].

1.3.1 k-Dimensional Knapsack

Integer programming with a constant number of constraints *in addition to* nonnegativity constraints is NP-hard: $\{\max cx : Ax \leq b, \mathbf{0} \leq x\}$ where A has k rows is the *k-dimensional unbounded knapsack* problem. For any fixed k this problem has a pseudopolynomial-time algorithm and a PTAS (both still hold even with multiplicity constraints), but no FPTAS for any constant $k \geq 2$ unless P=NP (this was originally shown for $d = \mathbf{1}$ around 1980 (see [25]) and subsequently even for infinite or arbitrary d in [37]). One simple PTAS is as follows: guess the t most profitable items (counting multiplicity) in the knapsack, then solve the natural LP relaxation of the residual problem and round all variables down to the nearest integers. Since in any extreme point solution there will be at most k fractional values (corresponding to at most k tight constraints) this gives a $t/(t + k)$-approximation algorithm. Covering integer programs $\min\{cx : Ax \geq b, 0 \leq x \leq d\}$ where A has k rows behave similarly. An LP relaxation with $1 + \epsilon$ integrality gap for k-dimensional knapsack can be obtained using disjunctive programming as described by Bienstock [7].

1.3.2 Semimodular Optimization

The results mentioned here all assume $d = \mathbf{1}$. Interestingly they all developed in mid-2009 (excepting Koufogiannakis and Young's arXiv report in late 2008). (*This section was added after publication.*)

For minimizing a submodular objective subject to k-row sparse covering constraints, three k-approximations were recently published, one in the framework of Koufogiannakis & Young [31, 34, 32, 33], one by Iwata and Nagano [24], and one by Goel et al. [16].

For maximizing a monotone submodular function subject to k-column sparse packing constraints, Bansal et al. [3] gave a $O(k)$-approximation algorithm.

It is not clear whether anyone has considered the problem of submodular k-CS CIPs. Some work has been done already on submodular set cover, for example see Wolsey [45]. One might hope to use results of Kolliopoulos & Young [29] (see also the references therein including work by Srinivasan) to get an $O(\ln k)$ approximation for this problem.

1.4 Summary

We summarize the existing and new results in Table 1. Note that in all four cases, the strongest known lower bounds are obtained even in the special case that A, b, c, d are 0-1. Note also that these bounds are nearly-matching except for k-CS PIPs.

	k-Column-Sparse		k-Row-Sparse	
	lower bound	upper bound	lower bound	upper bound
Packing	$\Omega(k/\ln k)$	$(\mathbf{k^2-2k+1)2^k+1}$	$n^{1-o(1)}$	ϵn
Covering	$\ln k - O(\ln\ln k)$	$O(\ln k)$	$k-\epsilon$	$\mathbf{k}$

Table 1: The landscape of approximability of sparse integer programs. Our main results are in boldface.

2 k-Approximation for k-Row-Sparse CIPs

By scaling rows and clipping coefficients that are too high, there is no loss of generality in the following definition.

Definition 1. *A* k-RS CIP *is an integer program* $\{\min cx : Ax \geq \mathbf{1}, \mathbf{0} \leq x \leq d\}$ *where A is k-RS and all entries of A are at most 1.*

To begin with, we focus on the case $d_j = +\infty$ for all j, which we will call *unbounded k-RS CIP*, since it already illustrates the essence of our new technique. Motivated by LP rounding methods, we make the following definition, in which x is a vector-valued variable and α is a vector of real coefficients. Throughout, we assume coefficients are nonnegative. When we apply $\lfloor\cdot\rfloor$ to vectors we mean the component-wise floor.

Definition 2. *A constraint $\alpha x \geq 1$ is* ρ-roundable *for some $\rho > 1$ if for all nonnegative real x, $(\alpha x \geq 1)$ implies $(\alpha\lfloor\rho x\rfloor \geq 1)$.*

Note that ρ-roundability implies ρ'-roundability for $\rho' > \rho$. The relevance of this property is explained by the following proposition.

Proposition 3. *If every constraint in an unbounded covering integer program is ρ-roundable, then there is a ρ-approximation algorithm for the program.*

Proof. Let x^* be an optimal solution to the program's linear relaxation. Then cx^* is a lower bound on the cost of any optimal solution. Thus, $\lfloor\rho x^*\rfloor$ is a feasible solution with cost at most ρ times optimal. $\square$

Another simple observation helps us get started.

Proposition 4. *The constraint $\alpha x \geq 1$ is $(1 + \sum_i \alpha_i)$-roundable.*

Proof. Let $\rho = (1 + \sum_i \alpha_i)$. Since $\lfloor t \rfloor > t - 1$ for any t, if $\alpha x \geq 1$ for a nonnegative x, then

$$\alpha \lfloor \rho x \rfloor \geq \sum_i \alpha_i(\rho x_i - 1) = \rho \sum_i \alpha_i x_i - \sum_i \alpha_i \geq \rho \cdot 1 - (\rho - 1) = 1,$$

as needed. $\square$

Now consider an unbounded k-RS CIP. Since each constraint has at most k coefficients, each less than 1, it follows from Proposition 4 that every constraint in these programs is $(k+1)$-roundable, and so such programs admit a $(k+1)$-approximation algorithm by Proposition 3. It is also clear that we can tighten the approximation ratio to k for programs where the sum of the coefficients in every constraint (row) is at most $k-1$. What we will now do is show that rows with sum in $(k-1, k]$ can be replaced by other rows which are k-roundable.

Definition 5. *Two constraints $\alpha x \geq 1$ and $\alpha' x \geq 1$ are* $\mathbb{Z}_+$-equivalent *if for all nonnegative integral x, $(\alpha x \geq 1) \Leftrightarrow (\alpha' x \geq 1)$.*

In other words, $\alpha x \geq 1$ and $\alpha' x \geq 1$ are $\mathbb{Z}_+$-equivalent if $\alpha x \geq 1$ is valid for $\{x : x \geq 0, \alpha' x \geq 1\}$ *and* $\alpha' x \geq 1$ is valid for $\{x : x \geq 0, \alpha x \geq 1\}$.

Proposition 6. *Every constraint $\alpha x \geq 1$ with at most k nonzero coefficients is $\mathbb{Z}_+$-equivalent to a k-roundable constraint.*

Before proving Proposition 6, let us illustrate its use.

Theorem 3. *There is a polynomial time k-approximation algorithm for unbounded k-RS CIPs.*

Proof. Using Proposition 6 we replace each constraint with a $\mathbb{Z}_+$-equivalent k-roundable one. The resulting IP has the same set of feasible solutions and the same objective function. Therefore, Proposition 3 yields a k-approximately optimal solution. $\square$

With the framework set up, we begin the technical part: a lemma, then the proof of Proposition 6.

Lemma 7. *For any positive integers k and v, the constraint $\sum_{i=1}^{k-1} x_i + \frac{1}{v} x_k \geq 1$ is k-roundable.*

Proof. Let $\alpha x \geq 1$ denote the constraint. If x satisfies the constraint, then the maximum of x_1, x_2, ..., x_{k-1} and $\frac{1}{v} x_k$ must be at least $1/k$. If $x_i \geq 1/k$ for some $i \neq k$ then $\lfloor k x_i \rfloor \geq 1$ and so $\alpha \lfloor kx \rfloor \geq 1$ as needed. Otherwise x_k must be at least v/k and so $\lfloor k x_k \rfloor \geq v$ which implies $\alpha \lfloor kx \rfloor \geq 1$ as needed. $\square$

Proof of Proposition 6. If the sum of coefficients in the constraint is $k-1$ or less, we are done by Proposition 4, hence we assume the sum is at greater than $k-1$. Without loss of generality (by renaming) such a constraint is of the form

$$\sum_{i=1}^{k} x_i \alpha_i \geq 1 \tag{1}$$

where $\mathbf{0} < \alpha \leq \mathbf{1}$, $k - 1 < \sum_i \alpha_i \leq k$, and the α_i's are nonincreasing in i.

Define the *support* of x to be $\mathrm{supp}(x) := \{i \mid x_i > 0\}$. Now $\alpha_{k-1} + \alpha_k > 1$ since $k - 1 < \sum_{i<k-1} \alpha_i + \alpha_{k-1} + \alpha_k \leq \alpha_{k-1} + \alpha_k + (k-2)$. Since the α_i are nonincreasing, $\alpha_i + \alpha_j > 1$ for any $i < k, j \leq k$; more generally, any integral $x \geq 0$ with $|\mathrm{supp}(x)| \geq 2$ must satisfy $\alpha x \geq 1$. To express the set of *all* feasible integral solutions, let $t = \max\{0\} \cup \{i \mid \alpha_i = 1\}$, let e_i denote the ith unit basis vector, and let $v = \lceil 1/\alpha_k \rceil$. Then it is not hard to see that the nonnegative integral solution set to constraint (1) is the disjoint union

$$\{x \mid x \geq 0, |\mathrm{supp}(x)| \geq 2\} \uplus \{z e_i \mid 1 \leq i \leq t, z \geq 1\} \uplus \{z e_i \mid t < i < k, z \geq 2\} \uplus \{z e_k \mid z \geq v\}. \tag{2}$$

The special case $t = k$ (i.e. $\alpha_1 = \alpha_2 = \cdots = \alpha_k = 1$) is already k-roundable by Lemma 7, so assume $t < k$. Consider the constraint

$$\sum_{i=1}^{t} x_i + \sum_{i=t+1}^{k-1} \frac{v-1}{v} x_i + \frac{1}{v} x_k \geq 1. \tag{3}$$

Every integral $x \geq 0$ with $|\operatorname{supp}(x)| \geq 2$ satisfies constraint (3). By also considering the cases $|\operatorname{supp}(x)| \in \{0, 1\}$, it is easy to check that constraint (3) has precisely Equation (2) as its set of feasible solutions, i.e. constraint (3) is $\mathbb{Z}_+$-equivalent to $\alpha x \geq 1$. If $t < k - 1$, the sum of the coefficients of constraint (3) is $k - 1$ or less, so it is k-roundable by Proposition 4. If $t = k - 1$, constraint (3) is k-roundable by Lemma 7. Thus in either case we have what we wanted. $\square$

2.1 Multiplicity Constraints

We next obtain approximation guarantee k even with multiplicity constraints $x \leq d$. For this we use *knapsack-cover inequalities.* These inequalities represent residual covering problems when a set of variables is taken at maximum multiplicity. Wolsey [45] studied inequalities like this for 0-1 problems to get a primal-dual approximation algorithm for submodular set cover. The LP we use is most like what appears in Carr et al. [8] and Kolliopoulos & Young [29], but we first replace each row with a k-roundable one.

Specifically, given a CIP $\{\min cx \mid Ax \geq \mathbf{1}, \mathbf{0} \leq x \leq d\}$ with A, d nonnegative, we now define the knapsack cover LP. Note that we allow d to contain some entries equal to $+\infty$. For a subset F of $\operatorname{supp}(A_i)$ such that $\sum_{j \in F} A_{ij} d_j < 1$, define $A_{ij}^{(F)} = \min\{A_{ij}, 1 - \sum_{j \in F} A_{ij} d_j\}$. Following [8, 29] we define the *knapsack cover LP* for our problem to be

$$\text{KC-LP} = \Big\{\min cx : \mathbf{0} \leq x \leq d; \qquad \forall i, \forall F \subset \operatorname{supp}(A_i) \text{ s.t. } \sum_{j \in F} A_{ij} d_j < 1 : \sum_{j \notin F} A_{ij}^{(F)} x_j \geq 1 - \sum_{j \in F} A_{ij} d_j \Big\}.$$

Theorem 1. *There is a polynomial time k-approximation algorithm for k-RS CIPs.*

Proof. Using Proposition 6, we assume all rows of A are k-roundable. Let x^* be the optimal solution to KC-LP. Define $\widehat{x} = \min\{d, \lfloor kx^* \rfloor\}$, where min denotes the component-wise minimum. We claim that $\widehat{x}$ is a feasible solution to the CIP, which will complete the proof. In other words, we want to show for each row i that $A_i \widehat{x} \geq 1$.

Fix any row i and define $F = \{j \in \operatorname{supp}(A_i) \mid x_j^* \geq d_j/k\}$, i.e. F is those variables in the constraint that were rounded to their maximum multiplicity. If $F = \varnothing$ then, by the k-roundability of $A_i x \geq 1$, we have that $A_i \widehat{x} = A_i \lfloor kx^* \rfloor \geq 1$ as needed. So assume $F \neq \varnothing$.

If $\sum_{j \in F} A_{ij} d_j \geq 1$ then the constraint $A_i \widehat{x} \geq 1$ is satisfied; consider otherwise. Since $\lfloor kx_j^* \rfloor > kx_j^* - 1$ for $j \notin F$, since x^* satisfies the knapsack cover constraint for i and F, and since $A_{ij}^{(F)} \leq 1 - \sum_{j \in F} A_{ij} d_j$ for each j, we have

$$\begin{aligned} \sum_{j \notin F} A_{ij}^{(F)} \widehat{x}_j &\geq k \sum_{j \notin F} A_{ij}^{(F)} x_j^* - \sum_{j \notin F} A_{ij}^{(F)} \\ &\geq k\Big(1 - \sum_{j \in F} A_{ij} d_j\Big) - \Big|\{j : j \in \operatorname{supp}(A_i) \backslash F\}\Big| \Big(1 - \sum_{j \in F} A_{ij} d_j\Big). \end{aligned}$$

Since $F \neq \varnothing$ and $|\operatorname{supp}(A_i)| \leq k$, this gives $\sum_{j \notin F} A_{ij}^{(F)} \widehat{x}_j \geq 1 - \sum_{j \in F} A_{ij} d_j$. Rearranging, and using the facts $(\forall j : A_{ij} \geq A_{ij}^{(F)})$ and $(\forall j \in F : d_j = \widehat{x}_j)$, we deduce $A_i \widehat{x} \geq 1$, as needed.

For fixed k, we may solve KC-LP explicitly, since it has polynomially many constraints. For general k, we follow the ellipsoid algorithm-based approach of [8, 29]: rather than solve KC-LP in polynomial time, we obtain a solution x^* which is optimal for a modified KC-LP having not all knapsack-cover constraints, but at least all those for the the m specific (i, F) pairs (depending on x^*) used in our proof; thus we still get a k-approximation in polynomial time. $\square$

2.2 Integrality Gap Bounds

For a particular LP relaxation of a covering integer program, the *integrality gap* is the ratio of the cost of the optimal integral solution to cost of the optimal LP solution. For a given linear program $\mathcal{L}$ let $\Gamma(\mathcal{L})$ denote its integrality gap; note $\Gamma(\mathcal{L}) \geq 1$. See [9] for an alternate characterization of the integrality gap. For packing integer programs there are analogous definitions of the integrality gap.

Integrality gaps are studied in their own right (e.g. [1] where the integrality gap of the *bidirected cut relaxation* turns out to exactly equal a type of *coding advantage*) but also because they are connected to the following common concept in the literature on approximation algorithms. An *LP-relative α-approximation algorithm*, with respect to a particular covering LP relaxation $\mathcal{L}$ of an integer program $\mathcal{P}$, is an algorithm that always returns a solution of value at most $\alpha \cdot \mathrm{OPT}(\mathcal{L})$. Such algorithms are automatically α-approximation algorithms for $\mathcal{P}$ since $\mathrm{OPT}(\mathcal{P}) \geq \mathrm{OPT}(\mathcal{L})$. A large proportion of the approximation algorithms obtained by LP methodology are LP-relative (often with respect to the "simplest" or most "natural" LP relaxations). This is related to integrality gaps because of the following: an $\mathcal{L}$-relative α-approximation algorithm gives a constructive proof that $\Gamma(\mathcal{L}) \leq \alpha$. Conversely, if one can prove a lower bound $\Gamma(\mathcal{L}) \geq \alpha_0$, this is evidence that naive LP-based approximation methods cannot give an α-approximation for any $\alpha < \alpha_0$. (In such cases one can still, in principle, either use a different LP as was done in [38, 13] for the edge dominating set problem, or abandon LP-relative approximation as was done in [2] for the demand interval packing problem.)

Unbounded problems. In discussing integrality gaps for k-RS CIP problems, we say that the *naive LP relaxation* of $\{\min cx \mid x \text{ integral}, Ax \geq b, \mathbf{0} \leq x \leq d\}$ is the LP obtained by removing the restriction of integrality. Earlier, we made the assumption that $A_{ij} \leq b_i$ for all i, j; let us call this the *clipping assumption*. The clipping assumption is without loss of generality for the purposes of approximation guarantees, however, it affects the integrality gap of the naive LP for unbounded k-RS CIP, as we now illustrate. Without the clipping assumption, the integrality gap of k-RS CIP problems can be unbounded as a function of k; indeed for any integer $M \geq 1$ the simple covering problem $\{\min x_1 \mid [M]x_1 \geq 1, 0 \leq x\}$ has integrality gap M. With the clipping assumption, our discussion of roundability shows in unbounded instances the integrality gap of the naive LP is at most $k+1$, since the roundability framework gives an LP-relative approximation algorithm.

Bounded problems. Even under the clipping assumption, k-RS CIPs with *multiplicity constraints* can have large integrality gaps — e.g. $\{\min x_2 \mid [{}^{M}_{M}]x \geq M+1, 0 \leq x, x_1 \leq 1\}$ has integrality gap M. For bounded instances, the knapsack-cover inequalities represent a natural generalization of the clipping assumption, namely, we perform a sort of clipping even considering that any subset of the variables are chosen to their maximum extent. (Note also that if $d_i = +\infty$ for all i, then KC-LP is just the naive LP with the clipping assumption.) Our methods show that KC-LP has integrality gap at most $k+1$ on k-RS CIP instances (since our algorithms are LP-relative). Our methods also show that KC-LP has integrality gap at most k *after* we replace each row with a k-roundable one (Proposition 6), but this result is a little esoteric.

We are actually unaware of any k-RS CIP instance with $k > 1$ where the integrality gap of KC-LP (without applying Proposition 6) is greater than k; resolving whether such an instance exists would be interesting. Some special cases are understood, e.g. Koufogiannakis and Young [33] give a primal-dual k-approximation for k-CS PIP in the case A is 0-1, also known as hypergraph b-matching.

3 Column-Sparse Packing Integer Programs

In this section we give an approximation algorithm for k-column-sparse packing integer programs with approximation ratio $2^k(k^2-2k+1)+1$, and better results for $k=2$. The results hold even in the presence of multiplicity constraints $x \leq d$. Broadly speaking, our approach is rooted in the demand matching algorithm of Shepherd & Vetta [40]; their path-augmenting algorithm can be viewed as a restricted form of *iterated relaxation*, which is the main tool in our new approach. Iterated relaxation yields a superoptimal solution that violates some constraints, and with probabilistic rounding and greedy ideas we are able to obtain a feasible solution while retaining at least a constant fraction of the weight.

By scaling rows and eliminating variables whose coefficients are too high, there is no loss of generality in the following definition.

Definition 8. *A k-*CS PIP *is an integer program $\{\max cx : Ax \leq \mathbf{1}, \mathbf{0} \leq x \leq d\}$ where A is k-CS and all entries of A are at most 1.*

We begin this section by explaining a simpler version of our new mechanism; this simpler version gives a $2^k(k^2-k+1)$-approximation algorithm for k-CS PIP in the special case $d = \mathbf{1}$.

By analogy with the demand matching problem and hypergraphic matching, it is natural to think of the rows of the constraint matrix A as indexed by a *vertex set* V and the columns as indexed by a *hyperedge set* E. Specifically, define a vertex for each row, let A_{ve} denote the entry of A at row v and column e, and for each column define its corresponding hyperedge e to be $\{v \mid A_{ve} > 0\}$; the resulting hypergraph may not be simple. We define the term *endpoint* to mean a pair (v, e) such that $A_{ve} > 0$.

The following intermediate result of iterated rounding is key for our approach. For a k-CS PIP $\mathcal{P}$ let $\mathcal{L}(\mathcal{P})$ denote its linear relaxation $\{\max cx \mid Ax \leq \mathbf{1}, \mathbf{0} \leq x \leq d\}$. Our iterated rounding algorithm computes a set S of *special* endpoints; for such a set we let $A_{S\to 0}$ denote the matrix obtained from A by zeroing out the entries corresponding to each special endpoint.

Lemma 9. *Given a k-CS PIP $\mathcal{P}$ with $d = \mathbf{1}$, we can in polynomial time find $y \in \{0,1\}^E$ and S such that*

(a) $cy \geq \mathrm{OPT}(\mathcal{L}(\mathcal{P}))$

(b) $\forall v \in V$, *we have* $|\{e : (v,e) \in S\}| \leq k$

(c) $A_{S\to 0}y \leq \mathbf{1}$.

Proof of Lemma 9. First, we give a sketch. Since $\mathcal{P}$ is k-column sparse, every hyperedge has size at most k. Let x^* be an extreme optimal solution to $\mathcal{L}(\mathcal{P})$. The crux of our approach deals with the case that x^* has no integral values: then x^* is a *basic feasible solution* all of whose tight constraints correspond to vertices, so the number of vertices is greater than or equal to the number of hyperedges. Thus by double-counting the average vertex degree is at most k, so some vertex has degree at most k. In other words there is some constraint which contains at most k nonzero variables, which allows iterated rounding to take place.

Since y is a 0-1 vector we can alternatively view it as a subset Y of E. With this convention, we now give pseudocode for our iterated rounding algorithm, IteratedSolver.

```
IteratedSolver($A, c$)
1: Set $S = Y = N = \varnothing, V' = V, E' = E$
2: loop
3:     Let $x^*$ be an extreme optimum of
           $\{\max cx \mid x \in [0,1]^E; A_{S\to 0}x \leq \mathbf{1}; \forall e \in Y : x_e = 1; \forall e \in N : x_e = 0\}$
4:     For each $e \in E'$ with $x^*_e = 0$, add $e$ to $N$, delete $e$ from $E'$
5:     For each $e \in E'$ with $x^*_e = 1$, add $e$ to $Y$, delete $e$ from $E'$
6:     If $E' = \varnothing$, terminate and return $S$ and $y$, the characteristic vector of $Y$
7:     for each vertex $v \in V'$ with degree less than or equal to $k$ in $(V', E')$ do
8:         Mark each endpoint $\{(v, e) \mid e \in E'\}$ special, delete $v$ from $V'$ (but leave $E'$ unchanged)
```

Now we explain the pseudocode. The sets Y, N are disjoint subsets of E, and $E' = E\backslash Y\backslash N$. When e leaves E', the value of x_e is fixed at 0 or 1. After deleting a vertex v from V', it will not be possible to later violate the constraint corresponding to v. Hence the linear program effectively only has variables for E' and constraints for V'. As remarked previously, since x^* is a basic feasible solution the average vertex degree is at most k each time Step 7 is reached, so $|V'|$ decreases in each iteration, and the algorithm has polynomial running time. (In fact, it is not hard to show that there are at most $O(k \log |V|)$ iterations.)

The algorithm has the property that cx^* does not decrease from one iteration to the next; since $x^* = y$ at termination, property (a) holds. Properties (b) and (c) can be seen immediately from the definition of the algorithm. ☐

Next, we show the kind of rounding which takes the output of IteratedSolver to a feasible solution.

Theorem 4. *There is a polynomial time $2^k(k^2-k+1)$-approximation algorithm for k-CS PIPs with $d = \mathbf{1}$.*

Proof. After running IteratedSolver, suppose we find a subset Z of Y with the property that if any $e, f \in Z$ intersect at a vertex v, neither (v,e) nor (v,f) is special. Then from Lemma 9(c) and the fact that entries of A are at most 1, it follows that Z is a feasible solution to $\mathcal{P}$ (the original k-CS PIP). In the rest of the proof, we show there exists such a set with at least a constant fraction of Y's profit.

To accomplish this we have each vertex $v \in V$ independently declare "special" or "non-special," each with probability 1/2. We say that the eth column is *accepted* if (1) for every endpoint $(v,e) \in S$ the vertex v declares special and (2) for every endpoint $(v,e) \notin S$ the vertex v declares non-special. It follows from the k-column-sparseness of A that each column is accepted with probability at least $1/2^k$.

Let $Y^a \subset Y$ denote the set of accepted columns, and $V^s \subset V$ denote the set of vertices that declared special. So $E[c(Y^a)] \geq c(Y)/2^k$. We need the following claim.

Claim 10. *If $Z \subset Y^a$ has the property that every two hyperedges e_1, e_2 in Z satisfy $e_1 \cap e_2 \cap V_s = \varnothing$, then Z is a feasible solution to $\mathcal{P}$.*

Proof. Let v be any vertex. If $v \in V^s$, there is at most one hyperedge $e \in Z$ such that $v \in e$. If $v \notin V^s$, the definition of accepted means that each e with $v \in e \in Z$ has $(v,e) \notin S$; and since $A_{S\to 0}y \leq \mathbf{1}$, the sum of A_{ve} over these hyperedges is at most 1. In either case Z satisfies the constraint corresponding to v. □

Another way of stating Claim 10 is that whenever Z is a matching on the induced subhypergraph $(V, Y^a)[V^s]$, Z is a feasible solution to $\mathcal{P}$. (Note, we do not discard "empty" hyperedges in this view — hyperedges disjoint from V_s can be freely added to any matching.) Consider the greedy algorithm for finding such a matching: we iteratively select the maximum-weight hyperedge that does not intersect any previously selected hyperedges on V^s. Since the subhypergraph has hyperedges of size at most k and degree at most k, it is easy to see that each chosen hyperedge precludes at most $k(k-1)$ other hyperedges for future selection. Thus the greedy algorithm outputs a set Z with cost at least $c(Y^a)/(k(k-1)+1)$, which is at least $c(Y)/2^k(k^2-k+1)$ in expectation. Using the fact that $c(Y) \geq \mathrm{OPT}(\mathcal{L}(\mathcal{P})) \geq \mathrm{OPT}(\mathcal{P})$, we are done. □

3.1 Proof of Main Results

Our strongest results are obtained by using a slightly more refined iterated rounding algorithm, shown in IteratedSolverRefined. As before S denotes a set of special endpoints. The new algorithm has two new features. First, we get up to $k-1$ special endpoints per vertex instead of k, but we generate one exceptional solution (a matching M, which we identify with its characteristic vector m) in order to do so. Second, we compute a solution x_0 which essentially reduces the general case to the special case $d = \mathbf{1}$. The new algorithm's main properties are as follows.

Lemma 11. *Given a k-CS PIP $\mathcal{P}$,* IteratedSolverRefined *computes y, m, x_0 and S such that*

(a) $c(x_0 + y + m) \geq \mathrm{OPT}(\mathcal{L}(\mathcal{P}))$

(b) $\forall v \in V$, *we have* $|\{e : (v,e) \in S\}| < k$

(c) $Ax_0 + A_{S\to 0}y \leq \mathbf{1}$

(d) m *is feasible for* $\mathcal{P}$.

Proof. We prove only parts that do not follow the lines of the proof of Lemma 9. The main invariant is that $c(x_0 + m + x^*)$ does not decrease between iterations. First, when reaching Step 10, the average degree in (V', E') is at most k, so $|V'| + |E'|$ indeed drops in each iteration. Note that for each fixed v, the degree of v with respect to E' is monotonically nonincreasing over time. Furthermore, note that when we add an hyperedge e to M, the degree of e's endpoints drop from k to $k-1$. It follows that the hyperedges in M are disjoint, so property (d) holds. □

```
IteratedSolverRefined$(A, c, d)$
 1: Let $x^*$ be an extreme optimum of $\{\max cx \mid \mathbf{0} \le x \le d, Ax \le \mathbf{1}\}$
 2: Let $x_0 = \lfloor x^* \rfloor$
 3: Let $N = \{e \in E \mid x^*_e$ integral$\}$
 4: Set $S = Y = M = \varnothing, E' = E \backslash N, V' = V$
 5: loop
 6:     Let $x^*$ be an extreme optimum of
            $\{\max cx \mid x \in [0,1]^E; Ax_0 + A_{S\to 0}x \le \mathbf{1}; \forall e \in Y : x_e = 1; \forall e \in M \cup N : x_e = 0\}$
 7:     For each $e \in E'$ with $x^*_e = 0$, add $e$ to $N$, delete $e$ from $E'$
 8:     For each $e \in E'$ with $x^*_e = 1$, add $e$ to $Y$, delete $e$ from $E'$
 9:     If $E' = \varnothing$, terminate and return $S$, $x_0$ and $y$, $m$, the characteristic vectors of $Y$, $M$
10:     if in $(V', E')$ every vertex has degree exactly $k$ then
11:         Pick any $e \in E'$, add $e$ to $M$, and delete $e$ from $E'$
12:     else
13:         for each vertex $v \in V'$ with degree less than $k$ in $(V', E')$ do
14:             Mark each endpoint $\{(v, e) \mid e \in E'\}$ special, delete $v$ from $V'$ (but leave $E'$ unchanged)
```

Theorem 2. *There is a polynomial time $2^k(k^2 - 2k + 1) + 1$-approximation algorithm for k-CS PIPs.*

Proof. Similarly to the proof of Theorem 4, each vertex independently declares "special" or "non-special" with probability 1/2 and we define Y^a as before. Define the vector x_0^a by

$$(x_0^a)_e = \begin{cases} 0 & \text{if any } v \in e \text{ is special;} \\ (x_0)_e & \text{otherwise.} \end{cases}$$

Then whenever $Z \subset Y^a$ is a matching on the induced subhypergraph $(V, Y^a)[V^s]$, we have that $z + x_0^a$ is a feasible solution for $\mathcal{P}$. The greedy routine deals with a hypergraph with hyperedges of size at most k and degree at most $k - 1$, hence the greedy routine gives a feasible solution $z + x_0^a$ with $cz \ge y^a/(k^2 - 2k + 1)$. Further, $E[cy^a] \ge cy/2^k$ and $E[cx_0^a] \ge cx_0/2^k$. It is then straightforward to see that the most profitable of $z + x_0^a$ and m yields a $2^k(k^2 - 2k + 1) + 1$-approximately optimal solution, in expectation. ☐

Theorem 5. *There is a deterministic polynomial time 4-approximation algorithm for 2-CS PIPs, and a randomized $6 - \sqrt{5} \approx 3.764$-approximation algorithm when (V, E) has no parallel edges and $d = \mathbf{1}$.*

(Sketch). We modify IteratedSolverRefined slightly. First, we need the observation that E' as initialized in Step 4 has at most one cycle in each connected component; this holds since otherwise there are more fractional variables than tight vertex constraints in that component, which contradicts the fact that x^* is a basic feasible solution. Second, we define M to contain one edge from each cycle in (V, E'), and run the algorithm from Step 5 onwards but using $E'' := E' \backslash M$ in place of E'. Since E'' is acyclic, the condition in Step 10 is never true. Note that Y is a forest and each vertex has at most one special endpoint. We may WOLOG assume each edge $e \in Y$ has at least one special endpoint, since otherwise we can reset $y_e = 0$ and $(x_0)_e = 1$ while retaining all properties from Lemma 11.

To complete the proof, we adapt rounding techniques applied by Shepherd & Vetta [40] to *demand matching*, which is 2-CS PIP where (i) in each column of A all nonzero values in that column are equal to one another and (ii) no two columns have their nonzeroes in the same two rows.

To get the first result, we use a simple colouring argument as in [40, Thm. 4.1] which shows that Y can be decomposed into two feasible solutions $y = y_1 + y_2$. Hence the most profitable of x_0, m, y_1, y_2 is a 4-approximately optimal solution.

For the second result, we instead apply a probabilistic technique from [40, §4.3]. They define a distribution over subsets of Y; let Z be the random variable indicating the subset. (As usual z is its characteristic vector, and similarly with x_0 for X_0, which is valid since $x_0 \leq d = \mathbf{1}$). Let $p = \frac{1}{20}(5 + \sqrt{5})$. Say that an edge uv is *compatible* with Z if Z neither contains an edge with a special endpoint at u, nor at v. The distribution has the properties that Z is always feasible for the PIP, $\Pr[e \in Z] = p$ for all $e \in Y$, and $\Pr[e \text{ compatible with } Z] \geq p$ for all $e \notin Y$. Finally, let x'_0 denote the result of zeroing out all coordinates in x_0 not compatible with Z. Then $z + x'_0$ is a feasible solution, and $\mathrm{E}[c(z + x'_0)] \geq pc(y + x_0)$. It follows that the better solution of $z + x'_0$ and m is a $1 + 1/p = 6 - \sqrt{5}$-approximately optimal solution. □

3.2 Getting Approximation Ratio $2k^2 + 2$

(*This section was added after publication.*) As mentioned in the introduction, two groups independently found that the framework in this paper could give an $O(k^2)$-approximation algorithm. Both groups came up with the same approach, and in fact the approach can be seen as related to the colouring arguments of Shepherd & Vetta [40].

It is not too hard to see that if we get a LP-relative α-approximation for the case $d = \mathbf{1}$, we also get an LP-relative $(\alpha+1)$-approximation for general d. To see this, suppose we have some arbitrary d; let $\mathcal{L}$ denote the linear relaxation of this integer program. Let x be an optimal solution to $\mathcal{L}$. Now x can be rounded down to give an integral solution $\lfloor x \rfloor$ of value $c \cdot \lfloor x \rfloor$. Let $\langle x \rangle := x - \lfloor x \rfloor$ denote the fractional part of x. Now consider the *residual* problem obtained by using the same A, b but with right-hand-size $d' := \lceil \langle x \rangle x \rceil$ (so $d'_i = 1$ if x_i is fractional, and $d'_i = 0$ if x_i is integral). (We may now discard all i with $d'_i = 0$.) Let $\mathcal{L}'$ denote the linear relaxation of the residual problem. It is straightforward to verify that $\langle x \rangle$ is feasible for $\mathcal{L}'$ and so $\mathrm{OPT}(\mathcal{L}') \geq c \cdot \langle x \rangle$. The α-approximation gives us an integral feasible solution x' for the new problem with value $c \cdot x' \geq \mathrm{OPT}(\mathcal{L}')/\alpha$. Thus the better of x' and $\lfloor x \rfloor$, by a standard averaging argument, has value

$$\max\{c \cdot \lfloor x \rfloor, c \cdot x'\} \geq \max\{c \cdot \lfloor x \rfloor, c \cdot \langle x \rangle/\alpha\} \geq \frac{c \cdot \lfloor x \rfloor + c \cdot \langle x \rangle}{\alpha + 1} = c \cdot x/(\alpha+1) = \mathrm{OPT}(\mathcal{L})/(\alpha+1).$$

So the better solution is an $\alpha + 1$-approximation to the original problem.

So from now on in this section we assume $d = \mathbf{1}$. Run the iterated rounding algorithm, giving a subset Y of variables set to 1 and a set S of special endpoints with at most v special endpoints at each vertex. We call two variables $e, f \in Y$ *in conflict* if they have a common vertex v and at least one of (v, e) or (v, f) is special. Now it is not hard to see directly from Lemma 9(c) that any subset $Y' \subset Y$ so that no two variables of Y' are in conflict, is feasible. Hence it suffices to show that there is such a subset Y' with $c(Y') \geq c(Y)/(2k+1)$. To obtain this, in turn, it suffices to show that we can partition Y into $2k+1$ sets $Y_1, Y_2, \ldots, Y_{2k+1}$ such that each Y_i contains no in-conflict pair. In other words, we are looking for a $2k+1$ colouring of Y so that any similarly-coloured edges are not in conflict.

To find our desired colouring, we create a *conflict digraph* which has node set Y and an arc (*directed edge*) from e to f whenever e, f are in conflict at a vertex (constraint) v and (v, e) is special. Rewording, there is an arc (e, f) if and only if there is a v such that $v \in e, v \in f$ and (v, e) is special. (If (v, f) is also special, there is also an arc (f, e) by exchanging e, f in this definition.) The key observation is that each node $f \in Y$ has indegree bounded by k^2, i.e. there are at most k^2 choices of e such that (e, f) is an arc: to see this note f contains at most k vertices since we are dealing with k-CS programs, and each vertex $w \in f$ is special for at most k choices of e. Now we use the following lemma.

Lemma 12. *A digraph with maximum indegree d has a $2d + 1$-colouring.*

Proof. We use induction on the number of nodes in the graph. The average indegree is at most d, and the average indegree equals the average outdegree. Hence some node n has outdegree at most the average, which is k. In total, this node has at most $2d$ neighbours. By induction there is a $(2d+1)$-colouring when we delete n, then we can extend it to the whole digraph by assigning n any colour not used by its neighbours. □

Now the conflict digraph has maximum indegree k^2 and so has a $2k^2 + 1$ colouring, as needed. (It is easy to verify all steps can be performed by a polynomial-time algorithm.)

3.3 Improvements For High Width

The *width* W of an integer program is $1/(\max_{ij} A_{ij}/b_i)$. Note that without loss of generality, $W \geq 1$. If we normalize $b = \mathbf{1}$ by row scaling as in the rest of this paper, then a program has width $\geq W$ iff every entry of A is at most $1/W$.

In many settings better approximation can be obtained as W increases. For example in k-RS CIPs with $b = \mathbf{1}$, the sum of each row of A is at most k/W, so Propositions 3 and 4 give a $(1 + k/W)$-approximation algorithm. Srinivasan [42, 43] gave a $(1+\ln(1+k)/W)$-approximation algorithm for unbounded k-CS CIPs. Using *grouping and scaling* techniques introduced by Kolliopoulos and Stein [28], Chekuri et al. [11] showed that no-bottleneck demand multicommodity flow in a tree admits a $(1+O(1/\sqrt{W}))$-approximation algorithm, and gave general sufficient conditions for a problem to admit a $(1 + O(1/\sqrt{W}))$-approximation algorithm. Along the same vein, using iterated rounding, Könemann et al. [30] obtained a $(1+O(1/W))$-approximation algorithm for ordinary multicommodity flow in a tree, and general sufficient conditions for a problem to admit a $(1 + O(1/W))$-approximation algorithm [30].

Whereas Theorem 2 gives an approximation which is exponential in k, the following result gives a significant improvement for high width. It treats a somewhat specialized case, but its main technique — using an LP to guide the reduction of an additively-violating solution to a feasible solution — is interesting and may have other applications.

Theorem 6. *There is a polynomial time $(1+k/W)/(1-k/W)$-approximation algorithm to solve k-column-sparse PIPs with $k/W < 1$.*

To make Theorem 6 more concrete, notice this implies a $1+O(k/W)$-approximation for $W > 1.01k$. This is tight in the sense that for any fixed $k \geq 4$, $1 + o(1/W)$-approximation is NP-hard, by results from [15, 30] on approximating multicommodity flows in trees. The previous general frameworks give approximation guarantees — [11] gives $1 + O(k/\sqrt{W})$ for $W = \Omega(k^2)$ and [30] gives $(W + k)/(W + k - k^2)$ — but both of these ratios are beaten by Theorem 6.

Note added after publication: The results of this section appear to have been essentially superseded (at least asymptotically) by the nice techniques of Bansal et al. [3], since their methods can be extended (using Chernoff-type bounds) to get an $O(k^{\lfloor 1/W \rfloor})$ approximation for k-CS PIPs.

Proof of Theorem 6. Observe that the output of IteratedSolverRefined satisfies $x_0 + y + m \leq d$; define $\widehat{x} := x_0 + y + m$. Then using Lemma 11, the fact that M is a matching, and the fact that all entries of A are at most $1/W$, we have that

$$A\widehat{x} \leq (1 + k/W)\mathbf{1}. \tag{4}$$

Define $\mathcal{V}(x) \subset V$ by $\mathcal{V}(x) := \{v \in V \mid \sum_e A_{ve}x_e > 1\}$, e.g. the set of violated constraints in $Ax \leq \mathbf{1}$.

We want to reduce $\widehat{x}$ so that no constraints are violated. In order to do this we employ a linear program. Let $\chi(S)$ denote the characteristic vector of S. Our LP, which takes a parameter $\widehat{x}$, is

$$LP(\widehat{x}) : \max\{cx \mid \mathbf{0} \leq x \leq \widehat{x}, Ax \leq \mathbf{1} - \frac{k}{W}\chi(\mathcal{V}(\widehat{x}))\}.$$

This LP is similar to the natural linear relaxation of $\{x \in \mathcal{P} \mid x \leq \widehat{x}\}$ except for the k/W term. We can utilize this LP in an iterated rounding approach, described by the following pseudocode.

```
IteratedReducer
1: while $\mathcal{V}(\widehat{x}) \neq \varnothing$ do
2:     Let $x^*$ be an extreme optimum of $LP(\widehat{x})$
3:     Let $\widehat{x} = \lceil x^* \rceil$
```

We claim that this algorithm terminates, and that the value of $c\widehat{x}$ upon termination is at least

$$\frac{1 - k/W}{1 + k/W}(x_0 + y + m) \geq \frac{1 - k/W}{1 + k/W}\,\mathrm{OPT}(\mathcal{L}(\mathcal{P})),$$

where the last inequality follows from Lemma 11. Once we show these facts, we are done. As an initial remark, note that each coordinate of $\widehat{x}$ is monotonically nonincreasing, and so $\mathcal{V}(\widehat{x})$ is also monotonically nonincreasing.

Observe that the LP in the first iteration has $\frac{1-k/W}{1+k/W}(x_0 + y + m)$ as a feasible solution, by (4). Next, note that x which is feasible for the LP in one iteration is also feasible for the LP in the next iteration since $\mathcal{V}(\widehat{x})$ is monotonically nonincreasing; hence the value of cx^* does not decrease between iterations.

To show the algorithm terminates, we will show that $\mathcal{V}(\widehat{x})$ loses at least one vertex per iteration. Note first that if $v \notin \mathcal{V}(\widehat{x})$, the constraint $A_v x \leq 1$ is already implied by the constraint $x \leq \widehat{x}$. Hence $LP(\widehat{x})$ may be viewed as having only $|\mathcal{V}(\widehat{x})|$ many constraints other than the box constraints $0 \leq x \leq \widehat{x}$. Then x, a basic feasible solution to $LP(\widehat{x})$, must have at most $|\mathcal{V}(\widehat{x})|$ non-integral variables (hyperedges). In particular, using the fact that every hyperedge contains at most k vertices, there exists some vertex $v \in \mathcal{V}(\widehat{x})$ such that v meets at most k of the non-integral variables in x^*. Thus (using the fact that all entries of A are at most $1/W$) we have $A_v\lceil x^*\rceil < A_v x^* + k(1/W) \leq 1$ — so $v \notin \mathcal{V}(\lceil x^*\rceil)$, and $\mathcal{V}(\widehat{x})$ is strictly smaller in the next iteration, as needed. $\square$

4 Open Problems

There remain a few gaps in the types of problems we studied:

- it would be natural to determine whether submodular k-CS CIPs have a $O(\ln k)$ approximation (see Section 1.3.2)
- the approximability of k-CS CIPs is known so far only to be between $\Omega(k/\ln k)$ and $O(k)$; improving this would be significant, since even for the 0-1 case (hypergraph matching) no $o(k)$ approximation is known
- is there any general reason why the special 0-1 cases are essentially as hard as the general cases?

Although 2-RS IPs are very hard to optimize (at least as hard as Max Independent Set), the problem of finding a *feasible* solution to a 2-RS IP is still interesting. Hochbaum et al. [22] gave a pseudopolynomial-time 2-SAT-based feasibility algorithm for 2-RS IPs with finite upper and lower bounds on variables. They asked if there is a pseudopolynomial-time feasibility algorithm when the bounds are replaced by just the requirement of nonnegativity, which is still open as far as we know. It is strongly NP-hard to determine if IPs of the form $\{x \geq 0 \mid Ax = b\}$ are feasible when A is 2-CS [21], e.g. by a reduction from 3-Partition; but for IPs where each variable appears at most twice *including* in upper/lower bounds, it appears all that is known is NP-completeness (for example, via the *unbounded knapsack problem* [36]).

Acknowledgement.

We would like to thank Glencora Borradaile, Christina Boucher, Deeparnab Chakrabarty, Stephane Durocher, Jochen Könemann and Christos Koufogiannakis for helpful discussions, and the ESA referees for useful feedback.

References

[1] A. Agarwal and M. Charikar. On the advantage of network coding for improving network throughput. In *Proc. IEEE Information Theory Workshop*, pages 105–109, 2004.

[2] N. Bansal, Z. Friggstad, R. Khandekar, and M. R. Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. In *Proc. 20th SODA*, pages 702–709, 2009.

[3] N. Bansal, N. Korula, and V. Nagarajan. On k-column sparse packing programs. arXiv:0908.2256, 2009.

[4] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *J. Algorithms*, 2:198–203, 1981.

[5] R. Bar-Yehuda and D. Rawitz. Efficient algorithms for integer programs with two variables per constraint. *Algorithmica*, 29(4):595–609, 2001.

[6] P. Berman. A $d/2$ approximation for maximum weight independent set in d-claw free graphs. *Nordic J. of Computing*, 7(3):178–184, 2000. Preliminary version appeared in *Proc. 7th SWAT*, pages 214–219, 2000.

[7] D. Bienstock. Approximate formulations for 0-1 knapsack sets. *Oper. Res. Lett.*, 36(3):317–320, 2008.

[8] R. D. Carr, L. Fleischer, V. J. Leung, and C. A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proc. 11th SODA*, pages 106–115, 2000.

[9] R. D. Carr and S. Vempala. Randomized metarounding. *Random Struct. Algorithms*, 20(3):343–352, 2002. Preliminary version appeared in *Proc. 32nd STOC*, pages 58–62, 2000.

[10] D. Chakrabarty and G. Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP. In *Proc. 49th FOCS*, pages 687–696, 2008.

[11] C. Chekuri, M. Mydlarz, and F. B. Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms*, 3(3):27, 2007. Preliminary version appeared in *Proc. 30th ICALP*, pages 410–425, 2003.

[12] I. Dinur, V. Guruswami, S. Khot, and O. Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. *SIAM J. Comput.*, 34(5):1129–1146, 2005. Preliminary version appeared in *Proc. 35th STOC*, pages 595–601, 2003.

[13] T. Fujito and H. Nagamochi. A 2-approximation algorithm for the minimum weight edge dominating set problem. *Discrete Appl. Math.*, 118(3):199–207, 2002.

[14] T. Fujito and T. Yabuta. Submodular integer cover and its application to production planning. In *Proc. 2nd WAOA*, volume 3351, pages 154–166, 2004.

[15] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, May 1997. Preliminary version appeared in *Proc. 20th ICALP*, pages 64–75, 1993.

[16] G. Goel, C. Karande, P. Tripathi, and L. Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. In *Proc. 50th FOCS*, 2009. To appear.

[17] N. G. Hall and D. S. Hochbaum. A fast approximation algorithm for the multicovering problem. *Discrete Appl. Math.*, 15(1):35–40, 1986.

[18] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[19] E. Hazan, S. Safra, and O. Schwartz. On the complexity of approximating k-set packing. *Comput. Complex.*, 15(1):20–39, 2006. Preliminary versions appeared in *Proc. 6th APPROX*, pages 83–97, 2003 and ECCC-TR03-020, 2003.

[20] D. S. Hochbaum. Approximation algorithms for set covering and vertex cover problems. *SIAM J. Comput.*, 11:555–556, 1982.

[21] D. S. Hochbaum. Monotonizing linear programs with up to two nonzeroes per column. *Oper. Res. Lett.*, 32(1):49–58, 2004.

[22] D. S. Hochbaum, N. Megiddo, J. S. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Math. Program.*, 62(1):69–83, 1993.

[23] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM J. Discret. Math.*, 2(1):68–72, 1989.

[24] S. Iwata and K. Nagano. Submodular function minimization under covering constraints. In *Proc. 50th FOCS*, 2009. To appear.

[25] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[26] S. Khot and A. K. Ponnuswami. Better inapproximability results for MaxClique, Chromatic Number and Min-3Lin-Deletion. In *Proc. 33rd ICALP*, pages 226–237, 2006.

[27] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2 - \epsilon$. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008. Preliminary version appeared in *Proc. 18th CCC*, pages 379–386, 2003.

[28] S. G. Kolliopoulos and C. Stein. Improved approximation algorithms for unsplittable flow problems. In *Proc. 38th FOCS*, pages 426–436, 1997.

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

[30] J. Könemann, O. Parekh, and D. Pritchard. Max-weight integral multicommodity flow in spiders and high-capacity trees. In *Proc. 6th WAOA*, pages 1–14, 2008. Full version "Integral Multicommodity Flow in Trees: Using Covers to Pack" in preparation.

[31] C. Koufogiannakis and N. E. Young. Flooding overcomes small covering constraints. arXiv:0807.0644, 2008.

[32] C. Koufogiannakis and N. E. Young. Distributed and parallel algorithms for weighted vertex cover and other covering problems. In *Proc. 28th PODC*, pages 171–179, 2009.

[33] C. Koufogiannakis and N. E. Young. Distributed fractional packing and maximum weighted b-matching via tail-recursive duality. In *Proc. 23rd DISC*, 2009. To appear.

[34] C. Koufogiannakis and N. E. Young. Greedy δ-approximation algorithm for covering with arbitrary constraints and submodular cost. In *Proc. 36th ICALP*, pages 634–652, 2009.

[35] H. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

[36] G. S. Lueker. Two NP-complete problems in nonnegative integer programming. Technical Report 178, Computer Science Laboratory, Princeton University, 1975.

[37] M. J. Magazine and M.-S. Chern. A note on approximation schemes for multidimensional knapsack problems. *Math. of Oper. Research*, 9(2):244–247, 1984.

[38] O. Parekh. *Polyhedral Techniques for Graphic Covering Problems.* PhD thesis, Carnegie Mellon University, 2002.

[39] D. Pritchard. Approximability of sparse integer programs. In *Proc. 17th ESA*, pages 83–94, 2009. Preliminary & post-publication versions appear at arXiv:0904.0859.

[40] F. B. Shepherd and A. Vetta. The demand-matching problem. *Mathematics of Operations Research*, 32(3):563–578, 2007. Preliminary version appeared in *Proc. 9th IPCO*, pages 457-474, 2002.

[41] M. Singh. *Iterative Methods in Combinatorial Optimization.* PhD thesis, Carnegie Mellon University, 2008.

[42] A. Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM J. Comput.*, 29(2):648–670, 1999. Preliminary version appeared in *Proc. 27th STOC*, pages 268–276, 1995.

[43] A. Srinivasan. An extension of the Lovász Local Lemma, and its applications to integer programming. *SIAM J. Comput.*, 36(3):609–634, 2006. Preliminary version appeared in *Proc. 7th SODA*, pages 6–15, 1996.

[44] L. Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proc. 33rd STOC*, pages 453–461, 2001.

[45] L. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

A Hardness of Column-Restricted 2-CS CIP

Theorem 7. *It is* NP*-hard to approximate 2-CS CIPs of the form* $\{\min cx \mid Ax \geq b, x \text{ is } 0\text{-}1\}$ *and* $\{\min cx \mid Ax \geq b, x \geq 0, x \text{ integral}\}$ *within ratio* $17/16 - \epsilon$ *even if the nonzeroes of every column of* A *are equal and* A *is of the block form* $\left[\begin{smallmatrix} A_1 \\ A_2 \end{smallmatrix}\right]$ *where each* A_i *is 1-CS.*

Proof. Our proof is a modification of a hardness proof from [10] for a budgeted allocation problem — we thank Deeparnab Chakrabarty for mentioning this to us. We focus on the version where x is 0-1; for the other version, no major modifications to the proof are needed. The specific problem described in the statement of the theorem is easily seen equivalent to the following problem, which we call *demand edge cover in bipartite multigraphs*: given a bipartite multigraph (V, E) where each vertex v has a demand b_v and each edge e has a cost c_e and value d_e, find a minimum-cost set E' of edges so that for each vertex v its demand is satisfied, meaning that $\sum_{e \in E' \cap \delta(v)} d_e \geq b_v$. Our construction also has the property that $c_e = d_e$ for each edge — so from now on we denote both by the variable d_e.

The proof uses a reduction from Max-3-Lin(2), which is the following optimization problem: given a collection $\{x_i\}_i$ of 0-1 variables and a family of three-variable modulo-2 equalities called *clauses* (for example, $x_1 + x_2 + x_3 \equiv 1 \pmod 2$), find an assignment of values to the variables which satisfies the maximum number of clauses. Håstad [18] showed that for any $\epsilon > 0$, it is NP-hard to distinguish between the two cases that (1) a $(1 - \epsilon)$ fraction of clauses can be satisfied and (2) at most a $(1/2 + \epsilon)$ fraction of clauses can be satisfied.

Given an instance of Max-3-Lin(2) we construct an instance of demand edge cover as follows. For each variable x_i there are three vertices "x_i", "$x_i = 0$" and "$x_i = 1$"; these vertices have b-value $4\deg(x_i)$ where $\deg(x_i)$ denotes the number of clauses containing x_i. For each clause there are four vertices labelled by the four assignments to its variables that do *not* satisfy it; for example for the clause $x_1 + x_2 + x_3 \equiv 1 \pmod 2$ we would introduce four vertices, one of which would be named "$x_1 = 0, x_2 = 0, x_3 = 0$." These vertices have b-value equal to 3. Each vertex "$x_i = C$" is connected to "x_i" by an edge with d-value $4\deg(x_i)$; each vertex v of the form "$x_{i_1} = C_1, x_{i_2} = C_2, x_{i_3} = C_3$" is incident to a total of nine edges each with d-value 1: three of these edges go to "$x_{i_j} = C_j$" for each $j = 1, 2, 3$. The construction is illustrated in Figure 1.

Let m denote the total number of clauses; so $\sum_i \deg(x_i) = 3m$. We claim that the optimal solution to this demand edge cover instance has cost $24m + 3t$ where t is the least possible number of unsatisfied clauses for the underlying Max-3-Lin(2) instance. If we can show this then we are done since Håstad's result shows we cannot distinguish whether the optimal cost is $\geq 24m + 3m(1/2 - \epsilon)$ or $\leq 24m + 3(\epsilon m)$.

Let x^* denote a solution to the Max-3-Lin(2) instance with t unsatisfied clauses; we show how to obtain a demand edge cover E' of cost $24m + 3t$. We include in E' the edge between "x_i" and "$x_i = x_i^*$" for each i; this has total cost $\sum_i 4\deg(x_i) = 12m$. For each satisfied clause $x_i + x_j + x_k \equiv C \pmod 2$, we include in E' all three edges between "$x_i = 1 - x_i^*$" and "$x_i = 1 - x_i^*, x_j = x_j^*, x_k = x_k^*$" and similarly for j, k, and one of each of the parallel triples between "$x_i = 1 - x_i^*, x_j = 1 - x_j^*, x_k = 1 - x_k^*$" and its neighbours; this has cost 12 for that clause. For each unsatisfied clause $x_i + x_j + x_k \equiv C \pmod 2$, we include in E' any three edges incident to "$x_i = x_i^*, x_j = x_j^*, x_k = x_k^*$," as well as the following twelve edges: the nodes

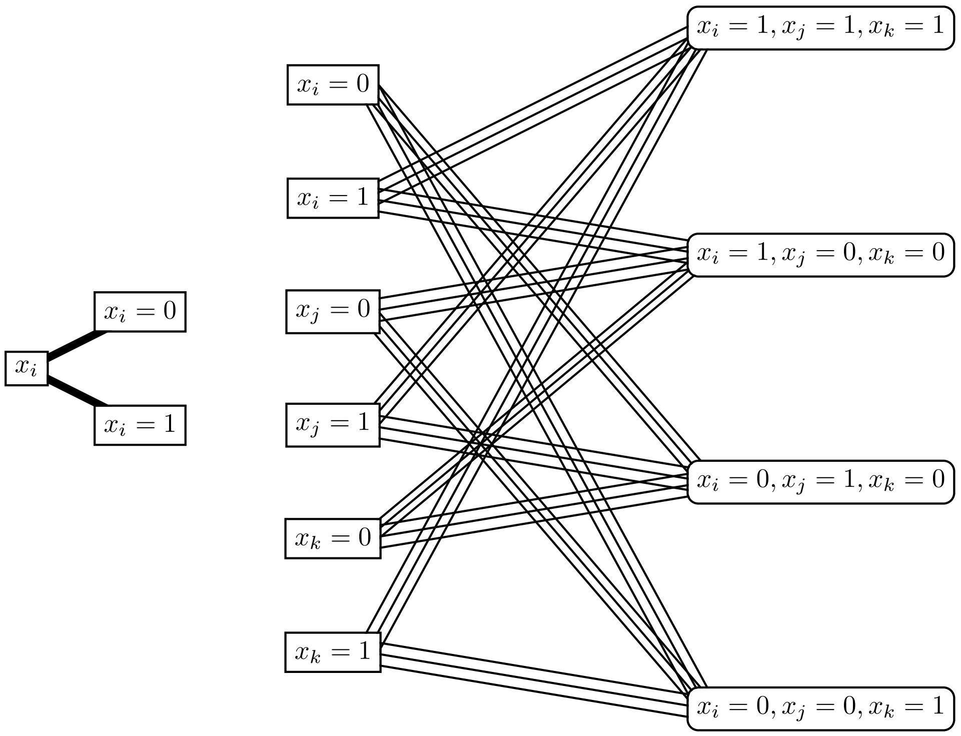
x_i
$x_i = 0$
$x_i = 1$
$x_i = 0$
$x_i = 1$
$x_j = 0$
$x_j = 1$
$x_k = 0$
$x_k = 1$
$x_i = 1, x_j = 1, x_k = 1$
$x_i = 1, x_j = 0, x_k = 0$
$x_i = 0, x_j = 1, x_k = 0$
$x_i = 0, x_j = 0, x_k = 1$

Figure 1: Left: the gadget constructed for each variable x_i. The vertices shown as rectangles have b-value $4\deg(x_i)$; the thick edges have d-value and cost $4\deg(x_i)$. Right: the gadget constructed for the clause $x_i + x_j + x_k \equiv 0 \pmod 2$. The vertices shown as rounded boxes have b-value 3; the thin edges each have unit d-value and cost.

"$x_i = 1 - x_i^*$" (symmetrically for j, k) and "$x_i = 1 - x_i^*, x_j = 1 - x_j^*, x_k = x_k^*$" (symmetrically for j, k) induce a 6-cycle of parallel triples, and we take two edges out of each triple; this has cost 15 for that clause. It is not hard to see that this solution is feasible — perhaps the trickiest vertices to check are those of the form "$x_i = 1 - x_i^*$," which are covered by 4 edges for each clause containing them. The total cost is $c(E') = 12m + 12(m - t) + 15t = 24m + 3t$.

To finish the proof we show the following.

Claim 13. *Given a feasible edge cover E', we can find a solution x^* such that t, the number of unsatisfied clauses for x^*, satisfies $24m + 3t \leq c(E')$.*

Proof. First we claim it is without loss of generality that for each i, E' contains exactly one of the edges incident to "x_i". Clearly at least one of these two edges lies in E'; if both do, then remove one (say, the edge between "x_i" and "$x_i = 0$") and add to E' any subset of the other $6\deg(x_i)$ other edges incident to "$x_i = 0$" so that the total number of edges incident on "$x_i = 0$" in E' becomes at least $4\deg(x_i)$. The removed edge has d-value $4\deg(x_i)$ and all other incident edges have d-value 1, so clearly the solution is still feasible and the cost has not increased.

Define x^* so that for each i, E' contains the edge between "x_i" and "$x_i = x_i^*$." Let E'' denote the edges of E' incident on clause vertices (i.e. the edges of E' with unit d-value). For $F \subset E''$ their *left-contribution*, denoted $\ell(F)$, is the number of them incident on vertices of the form "$x_i = 1 - x_i^*$." Note that $\ell(F) \leq |F|$ for any F. Furthermore for each unsatisfied clause, all edges incident on its vertex "$x_i = x_i^*, x_j = x_j^*, x_k = x_k^*$"

have zero left-contribution, but E' contains at least three of these edges. Thus the edges of E'' incident on that clause's vertices have $\ell(F) \leq |F| - 3$. Finally, consider $\ell(E'')$. Each edge of E'' is in the gadget for a particular clause, and it follows that $\ell(E'') \leq |E''| - 3t$ where t is the number of unsatisfied clauses for x^*. However, E'' needs to have $4\deg(x_i)$ edges incident on each "$x_i = 1 - x_i^*$" so $\ell(E'') \geq \sum_i 4\deg(x_i) = 12m$. Thus $|E''| \geq 12m+3t$ and considering the edges incident on the vertices "x_i" we see that $c(E') \geq 24m+3t$. □

This completes the proof of the reduction. □