New Approaches to Covering and Packing Problems

Aravind Srinivasan*

Abstract

Covering and packing integer programs model a large family of combinatorial optimization problems. The current-best approximation algorithms for these are an instance of the basic probabilistic method: showing that a certain randomized approach produces a good approximation with positive probability. This approach seems inherently sequential; by employing the *method of alteration* we present the first *RNC* and *NC* approximation algorithms that match the best sequential guarantees. Extending our approach, we get the first *RNC* and *NC* approximation algorithms for certain *multi-criteria* versions of these problems. We also present the first *NC* algorithms for two packing and covering problems that are not subsumed by the above result: finding large independent sets in graphs, and rounding fractional Group Steiner solutions on trees.

1 Introduction.

One way of viewing the covering and packing problems, which occupy a central place in combinatorial optimization, is as follows. Let Z_+ denote the set of non-negative integers. Given a monotone increasing[1] function $f : Z_+^n \rightarrow \{0,1\}$ and a non-negative n-dimensional vector $\vec{w}$, a covering problem is about minimizing $\vec{w} \cdot \vec{x}$ subject to $f(\vec{x}) = 1$. Similarly, given $\vec{w}$ and a monotone decreasing $f : Z_+^n \rightarrow \{0,1\}$, a packing problem is to maximize $\vec{w} \cdot \vec{x}$ subject to $f(\vec{x}) = 1$. Specialized to various "combinatorial" functions f, this framework models many NP-hard problems related to set cover, Steiner trees, hypergraph matching etc. In this work, we present the first *NC*-approximation algorithms for a few classes of these problems, matching the best-known sequential guarantees to within constant or $(1 + o(1))$ factors. Some of these ideas also lead to the first *NC*-approximation algorithms for certain multi-criteria covering and packing problems.

(a). *NC*-approximation algorithms for covering and packing integer programs. Our first family are results are for the covering and packing integer programs: we present the first *RNC*- and *NC*-approximation algorithms for these that match the current-best sequential approximation guarantees [38] to within a constant factor.

For a vector z, let z_i denote its ith component.

DEFINITION 1.1. **([38])** *Given $A \in [0,1]^{m \times n}$, $b \in [1,\infty)^m$ and $w \in [0,1]^n$ with $max_j\ w_j = 1$, a packing (resp. covering) integer program PIP (resp. CIP) seeks to maximize (resp. minimize) $w \cdot x$ subject to $x \in Z_+^n$ and $Ax \leq b$ (resp. $Ax \geq b$). If $A \in \{0,1\}^{m \times n}$, we assume that each entry of b is integral; a PIP may also have constraints of the form "$x_j \leq d_j$". We define $B = \min_i\ b_i$.*

As is well-known and explained in [38], although there are usually no restrictions on A, b and c beyond non-negativity, the above restrictions are without loss of generality.

The current-best approximation guarantees (described in §2) for general PIPs and CIPs are due to [38]. They are based on starting with a linear programming (LP) relaxation of the problem wherein the x_j are allowed to be *reals* instead of integers; appropriate randomized rounding [32, 31] is then shown to work. Concretely, suppose we have a PIP, say, with each x_j required to lie in $\{0,1\}$. Let the optimal solution to the LP relaxation be $\{x_1^*, x_2^*, \ldots, x_n^*\}$, and the optimal LP objective function value be $y^* = \sum_j w_j x_j^*$. For a certain $\lambda \geq 1$, round each x_j independently: to 1 with probability x_j^*/λ and to 0 with probability $1 - x_j^*/\lambda$. It is shown in [38] that a suitable choice of λ ensures, with positive probability, that: (i) all constraints are obeyed, and (ii) the objective function is at least as large as, say, one-third its expected value y^*/λ. (Modifications of this basic idea work for CIPs. We will mainly only discuss the case of PIPs in this section.)

The LP relaxation can be solved in *NC*, losing only a $(1 + (\log(m+n))^{-C})$ factor in the objective function for any desired constant $C > 0$ [25]. Also, randomized rounding schemes almost always have a straightforward *RNC* (if not *NC*) implementation: just randomly round all the variables in parallel. So what is the difficulty in at least developing an *RNC* version of the above algorithm of [38]? The crux of [38] is in

*Bell Laboratories, Lucent Technologies, 600–700 Mountain Avenue, Murray Hill, NJ 07974-0636, USA. E-mail: srin@research.bell-labs.com. URL: http://cm.bell-labs.com/cm/ms/who/srin/index.html

[1] i.e., if $f(\vec{x}) = 1$, then $f(\vec{y}) = 1$ for any $\vec{y} \in Z_+^n$ that coordinate-wise dominates $\vec{x}$

showing via an analysis of correlations, that λ can be chosen much smaller than known before (e.g., in [31]), leading to better approximation guarantees.[2] However, the catch is that the "positive probability" alluded to in the previous paragraph for this λ, can be exponentially small in the input size of the problem in the worst case: an explicit example of this is shown in [38]. Thus, if we just conduct a randomized rounding, we will produce a solution guaranteed in [38] with positive probability, but we are not assured of a (deterministic or randomized) *polynomial-time* algorithm for constructing such a solution. It is then shown in [38] that the method of conditional probabilities can be adapted to the situation and that we can efficiently round the variables one-by-one, achieving the existential bound. (This is one of the very few situations known to us where a probabilistic argument does not yield a good randomized algorithm, but spurs the development of an efficient deterministic algorithm.) This approach seems inherently sequential, and hence the apparent difficulty of even developing an RNC version of it.

We tackle this problem by appealing to the *method of alteration*, a key branch of the probabilistic method: do a random construction, allow the result to not satisfy all our requirements, alter the constructed random structure appropriately, and argue that this achieves our goal (in expectation or with high probability, hopefully). We conduct a randomized rounding with parameters similar to those of [38]; the probability of all constraints being satisfied is positive, but can be tiny. We then apply a parallel *greedy* technique that modifies some variables to enforce the constraints, and argue that the objective function does not change much in the process; our specific greedy approach is critical to making this argument work. This yields our RNC algorithm. We then proceed to derandomize this, by appealing to an "automata-fooling" approach of [30, 20, 26]. The natural choices for these automata have superpolynomially many states; we show how we can work with certain alternative polynomial-sized automata, and how the methods of [20, 26] can then be applied.

Similar results hold for CIPs. Specializing to what is perhaps the most well-known CIP, the set cover problem, we get the following. Suppose we have a set cover instance with the ground set having s elements. All known RNC or NC algorithms for this problem achieve an approximation of $O(\log s)$ [7, 25, 33, 8]. However, it is shown in [38, 36] that if the LP optimum is y^* (which is a lower bound on the optimal integral objective value), then there exists a polynomial-time computable solution of value $O(y^* \ln(s/y^*))$. Note in particular that for instances with say, $y^* = \Theta(s/\text{polylog}(s))$, the approximation ratio is $O(\log \log s)$. Thus, since we parallelize the bounds of [38], we get improved NC-approximations for set cover in situations where y^* is "large" as above. Similar remarks of course hold for all CIPs and PIPs.

(b). Approximating multi-criteria CIPs and PIPs. Considerable recent research has focused on *multi-criteria* (approximation) algorithms in scheduling, network design, routing and other areas: see, e.g., [35, 11, 23]. The motivation is to study how to balance *multiple* objective functions under a given system of constraints, modeling the fact that different participating individuals/organizations may have differing objectives. One abstract framework for this is in the setting of CIPs and PIPs: given the constraints of a CIP/PIP, a set of non-negative vectors $\{\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_\ell\}$, and a feasible solution $\vec{x}^*$ to the LP relaxation of the CIP/PIP's constraints, how good integral solutions $\vec{x}$ exist (and can be computed efficiently) that achieve a "good balance" among the different weight functions $\vec{w}_i$? For instance, given a PIP's constraints, we could ask for (approximating) the smallest $\alpha \geq 1$ such that there exists an integral feasible $\vec{x}$ with $\vec{w}_i \cdot \vec{x} \geq (\vec{w}_i \cdot \vec{x}^*)/\alpha$ for all i. We show in §3 how our alteration method of part (a) helps expand the set of situations here for which good RNC and NC approximation algorithms exist; it is crucial that our greedy algorithm of (a) depends only on the matrix A and vector b of the constraint system, and not on the objective function. The concrete benefit of this is that instead of arguing that all m constraints are satisfied, we need only show that a *single* random variable does not deviate much above its mean. Section 3 shows why earlier approaches such as ours in [38] cannot achieve the bounds we get here.

(c). Finding large independent sets in NC. An *independent set* (IS) in an undirected graph is a subset of the vertices such that there is no edge connecting any two vertices of the subset; an IS S is a *maximal independent set* (MIS) if no IS properly contains S. (Finding a maximum-sized IS is a PIP.) Among the first major derandomization results that do not rely on any complexity-theoretic assumptions, is the NC algorithm of Karp & Wigderson to find an MIS in a given graph [21]. This breakthrough was followed by further MIS algorithms (see, e.g., [1, 24, 16]) and related approaches, that significantly enhanced the derandomization area. However, since an MIS can be much smaller than a maximum-cardinality IS (consider, e.g., a star graph),

[2] Recall the parameter B from Definition 1.1. If, say, $B = 1$ and $A \in \{0,1\}^{m \times n}$ for a given PIP, [31] shows how to construct an integral solution of value $v = \Omega(y^*/\sqrt{m})$; [38] constructs a solution of value $\Omega(v^2)$ in this case.

Goldberg & Spencer studied the problem of finding ISs of *guaranteed size* in parallel [17]. Given a graph $G = (V, E)$, let $n = |V|$ and $m = |E|$. We will denote the degree of vertex v by d_v. Define $T_1(G) = \sum_{v \in V} 1/(d_v + 1)$; the convexity of $x \mapsto 1/(x+1)$ for $x \geq 0$, shows that $T_1(G) \geq T_2(G) \doteq n^2/(2m+n)$. Turán's classical theorem shows that G has an IS of size at least $T_1(G)$ (and hence at least $T_2(G)$) [39]. The work of [17] presents an NC algorithm to find an IS of size at least $T_2(G)$. But there exist n-vertex graphs G with $T_1(G) \gg T_2(G)$: even "$T_1(G) = \Theta(n)$ while $T_2(G) = \Theta(1)$" holds for certain graph families. So, the problem we address is: is there an NC algorithm to find ISs of cardinality (close to) $T_1(G)$?

We answer this in the affirmative by developing an NC-derandomization of an algorithm of Spencer [37]. (The *sequential* derandomization of this algorithm by the method of conditional probabilities is in fact the natural greedy algorithm to find a large IS [13].) Our algorithm finds an IS of size $(1 - o(1))T_1(G)$, where the "$o(1)$" term is $1/(\log n)^{1/2-\epsilon}$, with $0 < \epsilon < 1/2$ being any given constant. We can also show that if we wanted an IS of size at least $(1-\delta)T_1(G)$ for any given *constant* $\delta > 0$, we could use the elegant result of Indyk on min-wise independence [18]. Indyk's bounds do not yield NC algorithms if we want δ to go to zero as n increases: the work performed by an NC algorithm here would be $n^{O(\log(1/\delta))}$, which is superpolynomial if $\delta(n) = o(1)$. Moreover, the bounds of [18] appear to imply a work bound of about $n^{12e\log(98/\delta)}$, [3] which is large even for moderate constants δ; the work bound for our $(1 - o(1))T_1(G)$ result is at most $O(n^7)$.

(d). Rounding fractional solutions to Group Steiner problems. The Group Steiner tree problem on graphs [34, 15], defined in §5, is a covering problem that generalizes the Steiner tree problem on graphs. Its restriction to trees generalizes the set cover problem. The work of [15] presented the first polylogarithmic approximation algorithm for the problem: their approach is to reduce the problem to one on trees, and to then conduct a suitable randomized rounding of an LP relaxation. The work of [9] presented a sequential derandomization of this randomized rounding; we modify the algorithm and develop an NC rounding. A common theme of our results (c) and (d) is the use of small sample spaces that approximate product distributions [29, 12].

Details and proofs omitted here will be presented in the full version.

[3] Throughout, e will denote the base of the natural algorithm

2 Randomized rounding augmented with greedy alteration.

2.1 Preliminaries. Let $\exp(x)$ denote e^x. We first recall the Chernoff-Hoeffding bounds [3, 28]. Let $Z_1, Z_2, \ldots, Z_\ell$ be *independent* random variables, each taking values in $[0,1]$. Let $Z = \sum_i Z_i$ and $\mathbf{E}[Z] = \mu$. Then,

$$\begin{aligned} \Pr[Z \geq \mu(1+\delta)] &\leq G(\mu, \delta) \\ &\doteq (\exp(\delta)/(1+\delta)^{(1+\delta)})^\mu, \ \forall \delta \geq 0; \quad (2.1) \\ \Pr[Z \leq \mu(1-\delta)] &\leq H(\mu, \delta) \\ &\doteq (\exp(-\delta)/(1-\delta)^{(1-\delta)})^\mu \\ &\leq \exp(-\mu\delta^2/2), \ \forall \delta \in (0,1). \quad (2.2) \end{aligned}$$

Motivated by these bounds, the scaling parameter $\lambda \geq 1$ for the randomized rounding, is chosen as follows in [38]. Recall that $y^* = \sum_i w_i x_i^*$ denotes the optimal LP objective value for a PIP/CIP; also recall the parameter B from Definition 1.1. For PIPs, [38] chooses λ to be

$$K_0 \cdot \max\{1, (m/y^*)^{1/B}\} \text{ if } A \in \{0,1\}^{m \times n}, \text{ and} \quad (2.3)$$
$$K_0 \cdot \max\{1, (K_1 m/y^*)^{1/(B-1)}\} \text{ otherwise,} \quad (2.4)$$

where $K_0 > 1$ and $K_1 \geq 1$ are certain constants. Doing a correlational analysis via the FKG inequality, it is shown in [38] that PIPs have integral solutions of value at least $\Omega(y^*/\lambda)$ where λ is as above; as mentioned in §1, [38] also shows a constructive version of this, which seems inherently sequential. Note that the approximation guarantee of $O(\lambda)$ here is better for the case $A \in \{0,1\}^{m \times n}$ than for the general case where $A \in [0,1]^{m \times n}$. It is shown in [22] that these bounds for the case $A \in \{0,1\}^{m \times n}$ can also be generalized to the case of *column restricted PIPs*, a useful class of problems related to, e.g., unsplittable flow. (Also note that the bounds of [38] start getting poor in the case where $A \notin \{0,1\}^{m \times n}$, and where B is greater than, but very close to 1. Better bounds for this case are shown in [10], which also our approach matches in NC. For brevity, we defer this case to the full version of this paper.) For CIPs, [38] chooses λ to be

$$1 + O(\max\{\frac{\ln(mB/y^*)}{B}, \sqrt{\frac{\ln(2\lceil mB/y^* \rceil)}{B}}\}), \quad (2.5)$$

and develops an $(1+O(\lambda-1))$–approximation algorithm.

Section 2.2 presents our "randomized rounding plus alteration"; in §2.3, we will show that this algorithm matches the above-seen approximation bounds of [38] for PIPs and CIPs, in expectation. The algorithm will then be derandomized in §2.4.

2.2 The basic algorithm. Suppose we are given a PIP. For any desired constant $C > 0$, a feasible solution $\{x_1^*, x_2^*, \ldots, x_n^*\}$ to the PIP's LP relaxation with objective function value at least $(1 - (\log(m+n))^{-C})$ times the optimal LP objective value, can be found in NC [25]. ($\log x$ denotes $\log_2 x$ for the rest of this paper.) Given the x_j^*, we choose a suitable $\lambda \geq 1$ (the choice of which will be the heart of our analysis), and do the following randomized rounding: independently for each j, round x_j to $\lceil x_j^*/\lambda \rceil$ with probability $x_j^*/\lambda - \lfloor x_j^*/\lambda \rfloor$, and to $\lfloor x_j^*/\lambda \rfloor$ with probability $1 - x_j^*/\lambda + \lfloor x_j^*/\lambda \rfloor$. This is basically the same as in [31, 38]. The crucial point is that we allow the constraints of the PIP to be violated; we now alter the values x_j to fix the violated constraints, as follows. Let $[k]$ denote the set $\{1, 2, \ldots, k\}$. In each row i of the matrix A, let L_i be a pre-computed list (permutation of $[n]$) $\langle \sigma(i,1), \sigma(i,2), \ldots, \sigma(i,n) \rangle$ such that

$$A_{i,\sigma(i,1)} \geq A_{i,\sigma(i,2)} \geq \cdots \geq A_{i,\sigma(i,n)}. \tag{2.6}$$

Ties in the choice of L_i are broken arbitrarily. Suppose the ith constraint "$(Ax)_i \leq b_i$" has been violated; our process $\mathcal{F}_i$ of enforcing this constraint is as follows. We traverse the variables $\{x_1, x_2, \ldots, x_n\}$ in the permutation order given by L_i, and keep rounding down variables that have been rounded up by the randomized rounding, until the constraint "$(Ax)_i \leq b_i$" is satisfied. (In particular, variables x_j for which x_j^*/λ was an integer, remain unaltered at x_j^*/λ.) Note that this is easily done in NC, by a parallel prefix computation. Also, this is done in parallel for all constraints i that were violated; these parallel threads $\mathcal{F}_1, \ldots, \mathcal{F}_m$ *do not* interact with each other. (For instance, suppose the ith constraint is that $0.8x_2 + x_3 + 0.6x_5 + x_7 + 0.7x_8 \leq 2$, and that randomized rounding set $x_3 := 0$ and rounded-up each of x_2, x_5, x_7 and x_8 to 1. Then, $\mathcal{F}_i$ will reset precisely x_7 and x_2 to 0. Now, some $\mathcal{F}_{i'}$ for $i' \neq i$ may reset x_8 to zero, in which case we could potentially revisit the ith constraint and try to set, say, x_2 back to 1. We do not analyze such possible optimizations; this is what we mean by "the different $\mathcal{F}_i$ do not interact with each other". Also note that the different $\mathcal{F}_i$ may round down the same variable in parallel.) After this alteration, all constraints will be satisfied; this is the algorithm we will analyze for PIPs. The alteration is greedy in the sense that guided by (2.6), the $\mathcal{F}_i$ try to alter as few variables as possible.

The algorithm is basically the same for CIPs, with the following minor modifications. First, the NC algorithm of [25] produces a solution to the LP relaxation with value at most $(1 + (\log(m+n))^{-C})$ times optimal, for any desired constant $C > 0$. In the randomized rounding, since all constraints are of the "$\geq$" type, we scale all variables *up* by $\lambda \geq 1$. More precisely, independently for each j, we round x_j to $\lceil \lambda x_j^* \rceil$ with probability $\lambda x_j^* - \lfloor \lambda x_j^* \rfloor$, and to $\lfloor \lambda x_j^* \rfloor$ with probability $1 - \lambda x_j^* + \lfloor \lambda x_j^* \rfloor$. Furthermore, in the alteration step, the $\mathcal{F}_i$ keep rounding up rounded-down variables in the same "L_i order" as above, to enforce the constraints.

2.3 Analysis of the algorithm. We start with PIPs. Fix a PIP conforming to Definition 1.1. Recall the notation of §2.1 and §2.2. We assume that $x_j^* \in [0,1]$ for each j; as shown in [38], the approximation bounds only get better if $x_j^* > 1$ for some j. (We will prove this in the full version of this paper. The intuition is that rounding, say, 26.3 to 26 or 27 is a much less delicate choice than rounding, say, 0.3 to 0 or 1.)

Suppose we run the "randomized rounding and alteration" of §2.2, with λ as in (2.3, 2.4). Let X_i be the random variable denoting the number of variables altered in row i, by $\mathcal{F}_i$. (Suppose $\mathcal{F}_i$ stops at some index j in the list L_i. Note that $\mathcal{F}_{i'}$, for some $i' \neq i$, may turn some variable $x_{j'}$ from 1 to 0, with j' appearing *after* j in L_i. Such variables j' are *not* counted in calculating X_i.)

Notation. (i) For $k = 1, 2, \ldots$, let $L_{i,k}$ be the sub-list of L_i such that for all $j \in L_{i,k}$, $A_{i,j} \in (2^{-k}, 2^{-k+1}]$. (Note that L_i is the concatenation of $L_{i,1}, L_{i,2}, \ldots$.) (ii) Define $M_{i,k}$ to be the multiset obtained by collecting together the elements of $L_{i,t}$, for all $t \geq k$. (iii) Let Z_i denote the largest k for which some element of $L_{i,k}$ was reset from 1 to 0 by $\mathcal{F}_i$; $Z_i = 0$ iff the ith constraint was not violated. (iv) Call a PIP "Type I" if $A \in \{0,1\}^{m \times n}$, and "Type II" otherwise.

For the rest of this subsection, the x_j will denote the outcome of randomized rounding, i.e., the values of the variables *before* alteration. Good upper-tail bounds for the X_i will be crucial to our analysis, as well as to our derandomization of §2.4: Lemma 2.1 provides such bounds. Parts (a) and (b) of Lemma 2.1 will respectively help handle the cases of "large" Z_i and "small" Z_i. In the statement of the lemma, the function G is from (2.1).

LEMMA 2.1. *Fix a PIP; let $i \in [m]$, and $k, y \geq 1$ be integers. Then:*
(a) $\Pr[Z_i \geq k] \leq G(b_i 2^{k-1}/\lambda, \lambda - 1)$.
(b) *If $\lambda \geq 3$, say, then* $\Pr[(Z_i = k) \wedge (X_i \geq y)] \leq O((e/\lambda)^{b_i + 0.5(\lceil y/k \rceil - 1)})$.
(c) *If the PIP is Type I, then* $\Pr[X_i \geq y] \leq G(b_i/\lambda, \lambda(1 + y/b_i) - 1)$.

Proof. (a) $Z_i \geq k$ implies that $\sum_{j \in M_{i,k}} A_{i,j} x_j > b_i$, i.e., that $\sum_{j \in M_{i,k}} 2^{k-1} A_{i,j} x_j > 2^{k-1} b_i$, since otherwise we will have $Z_i < k$. Since $\mathbf{E}[x_j] = x_j^*/\lambda$, $\mathbf{E}[\sum_j A_{i,j} x_j] \leq b_i/\lambda$; so $\mathbf{E}[\sum_{j \in M_{i,k}} 2^{k-1} A_{i,j} x_j] \leq b_i 2^{k-1}/\lambda$. Also,

$2^{k-1}A_{i,j} \in [0,1]$ for all $j \in M_{i,k}$. Bound (2.1) now completes the proof.
(b) Let $X_{i,\ell}$ denote the number of elements of $L_{i,\ell}$ that are altered by $\mathcal{F}_i$. Now, $X_i = \sum_\ell X_{i,\ell}$; also, if $Z_i = k$, then $X_{i,\ell} = 0$ for $\ell > k$. So, "$(Z_i = k) \wedge (X_i \geq y)$" implies the existence of $\ell \in [k]$ with $X_{i,\ell} \geq \lceil y/k \rceil$. This in turn implies that $\sum_{j \in M_{i,\ell}} A_{i,j}x_j > b_i + (\lceil y/k \rceil - 1)2^{-\ell}$, since resetting any element of $L_{i,\ell}$ from 1 to 0 decreases $\sum_j A_{i,j}x_j$ by more than $2^{-\ell}$. Let $\theta \doteq (\lceil y/k \rceil - 1)$ for notational convenience. So, $\Pr[(Z_i = k) \wedge (X_i \geq y)]$ is at most

$$\sum_{\ell=1}^{k} \Pr[\sum_{j \in M_{i,\ell}} 2^{\ell-1}A_{i,j}x_j > b_i 2^{\ell-1} + \theta/2]. \tag{2.7}$$

We have that $\mathbf{E}[\sum_{j \in M_{i,\ell}} 2^{\ell-1}A_{i,j}x_j] \leq b_i 2^{\ell-1}/\lambda$, and that $2^{\ell-1}A_{i,j} \in [0,1]$ for all $j \in M_{i,\ell}$. Using (2.1) to bound (2.7), we get the bound

$$\sum_{\ell=1}^{k} G(b_i 2^{\ell-1}/\lambda, \lambda(1 + \theta 2^{-\ell}/b_i) - 1) \leq$$
$$\sum_{\ell=1}^{k} \left(\frac{e}{\lambda(1 + \theta 2^{-\ell}/b_i)} \right)^{b_i 2^{\ell-1} + 0.5\theta} \leq$$
$$\sum_{\ell=1}^{k} (e/\lambda)^{b_i 2^{\ell-1} + 0.5\theta} =$$
$$O((e/\lambda)^{b_i + 0.5\theta}).$$

(c) Here, $X_i \geq y$ iff $\sum_j A_{i,j}x_j \geq b_i + y$; we now employ (2.1). This concludes the proof of Lemma 2.1.

Remark. Observe how the greedy nature of $\mathcal{F}_i$ helps much in establishing Lemma 2.1.

THEOREM 2.1. *There are constants $K_2, K_3, K_4, K_5 > 0$ such that the following hold. Fix any PIP and any $i \in [m]$; suppose $\lambda \geq 3$. Define $p = (e/\lambda)^B$ if the PIP is Type I, and $p = (e/\lambda)^{B+1}$ otherwise. Then:*
(a) *For any integer $y \geq 2$, $\Pr[X_i \geq y] \leq K_2 p \cdot \min\{1, K_3 e^{-K_4 y/\log y}\}$.*
(b) $\mathbf{E}[X_i] \leq K_5 p$.

Proof. We omit showing some straightforward calculations in this proof.
(a) If the PIP is Type I, (a stronger version of) part (a) easily follows from part (c) of Lemma 2.1; so suppose the PIP is Type II. Choose t of the form $\log y + \Theta(1)$. We have

$$\Pr[X_i \geq y] = \sum_{k \geq 1} \Pr[(Z_i = k) \wedge (X_i \geq y)]$$
$$\leq \sum_{k \in [t-1]} \Pr[(Z_i = k) \wedge (X_i \geq y)] + \Pr[Z_i \geq t].$$

We apply parts (a) and (b) of Lemma 2.1 to respectively bound the second and first summands in the last expression, to get the claimed bound.
(b) $\Pr[X_i \geq 1] = \Pr[(Ax)_i > b_i] \leq p$, by (2.1); see, e.g., [38]. Also, part (a) shows that

$$\sum_{y \geq 2} \Pr[X_i \geq y] = O(p).$$

We now use the equation $\mathbf{E}[X_i] = \sum_{y \geq 1} \Pr[X_i \geq y]$ to complete the proof of Theorem 2.1.

We are now ready to analyze our *RNC* alteration algorithm for PIPs. The expected value of the objective function is y^*/λ; since $w_j \leq 1$ for all j, the expected reduction in the objective value caused by the alteration, is at most $\sum_{i \in [m]} \mathbf{E}[X_i] \leq K_5 mp$, by part (b) of Theorem 2.1. (This is an overcount since the same altered variable x_j may get counted by several X_i.) Thus, the expected final objective value is at least $y^*/\lambda - K_5 mp$, which is $\Omega(y^*/\lambda)$ if the constants K_0 and K_1 of (2.3, 2.4) are chosen large enough.

The basic analysis is similar for CIPs. With λ as in (2.5), we aim to show that $\mathbf{E}[\sum_i X_i] \leq O(y^*(\lambda - 1))$. So, the expected objective value after alteration will be at most $y^*\lambda + O(y^*(\lambda - 1)) = y^*(1 + O(\lambda - 1))$, matching the result of [38]. The basic proof idea, as for PIPs, is to show that Z_i being "large" is unlikely, and to present a good tail bound for the case where Z_i is "small". (The analog of Theorem 2.1(a) is that $\Pr[X_i \geq y] \leq O(\exp(-\Omega(B(\lambda - 1)^2/\lambda + y/\log y)))$.) The proof is deferred to the full version.

2.4 Derandomization. To derandomize the algorithms of §2.2, we will adapt an "automata-fooling" approach of [20, 26]. Simplifying this for our purposes, we have the following. Suppose we have h finite-state automata $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_h$ with respective state-sets $S_1, S_2, \ldots, S_h$, such that $S_i \cap S_j = \emptyset$ if $i \neq j$. Each S_i is partitioned into $n+1$ layers, numbered $0, 1, \ldots, n$. Layer 0 has a unique state s_i, which is also the start-state of $\mathcal{A}_i$. All transitions are only from one layer to the layer numbered one higher; there are no transitions from layer n. Outgoing arcs from a state are numbered by an integer in the range $\{0, 1, \ldots, 2^d - 1\}$, for some integer d. Given a word $\gamma_1\gamma_2\ldots\gamma_n$ where each γ_i is a d-bit string, each $\mathcal{A}_i$ moves from its start-state s_i to some state in layer n of S_i, in the obvious way. Now suppose we are given a parameter $\epsilon \in (0,1)$. Let $R = r + 2^d + 1/\epsilon$, where $r = \sum_i |S_i|$. Then, [20, 26] presents a parallel algorithm to construct a set $T \subseteq \{0, 1, \ldots, 2^d - 1\}^n$ of size poly(R); this parallel algorithm uses poly(R) processors and runs in polylog(R) time. The key property of T is as follows. Given a state t in layer n of S_i, let $p_1(i,t)$ be the

probability of reaching state t, if we choose $\gamma_1\gamma_2\ldots\gamma_n$ uniformly at random from $\{0,1,\ldots,2^d-1\}^n$; let $p_2(i,t)$ be this probability if $\gamma_1\gamma_2\ldots\gamma_n$ is chosen uniformly at random from T. Then, T has the useful property that

$$\forall(i,t),\ |p_1(i,t)-p_2(i,t)|\leq\epsilon. \tag{2.8}$$

We only show here how to adapt the above framework to derandomize our algorithm for PIPs; the case of CIPs is similar and will be presented in the full version. The basic idea is as follows. Let N denote the input size of a PIP; we have $\max\{m,n\}\leq N\leq O(mn)$. Round up all values $A_{i,j}$ to the nearest non-positive power of 2; a feasible fractional solution is obtained by dividing all the x_j^* by 2. Also, any feasible integral solution for this new system is also feasible for our original instance. Note that we now have

$$\forall(i,k)\ \forall j\in L_{i,k},\ A_{i,j}=2^{-k+1}. \tag{2.9}$$

Next, since those $A_{i,j}$ and w_j that are very small (say, less than $1/N^3$), as well as values x_j^* that are, say, less than $1/N^3$ can be safely omitted from consideration, simple perturbations ensure that for all i,j, the values $A_{i,j}$, w_j and $\mathbf{E}[x_j]$ are rationals with denominator 2^d, where $d=\Theta(\log N)$. Note that the values $\{x_1,x_2,\ldots,x_n\}$ output by the randomized rounding are the only random variables in our RNC algorithm. We can construct x_j by generating a random d-bit integer x'_j, and setting x_j to 1 iff $x'_j<x_j2^d$. Instead, suppose, for some explicitly constructed $T\subseteq\{0,1,\ldots,2^d-1\}^n$ of size poly(N) and for $(x_1,x_2,\ldots,x_n)$ chosen at random from T, we have:
(P1) the expected value of the objective function is α, and
(P2) for each $i\in[m]$, $\mathbf{E}[X_i]=\beta_i$.

Then, the one-paragraph analysis following the proof of Theorem 2.1 shows that the expected value of the objective function (for $(x_1,x_2,\ldots,x_n)$ chosen at random from T) is at least $\alpha-\sum_i\beta_i$. Since T is polynomial-sized, a parallel exhaustive search over T will then help construct a solution of value at least $\alpha-\sum_i\beta_i$ in NC. We aim to use the above automata-fooling approach to achieve

$$\alpha=\Omega(y^*/\lambda);\quad\forall i,\ \beta_i=O(p). \tag{2.10}$$

We will then get a solution of value $\Omega(y^*/\lambda)$ as desired.

Our $m+1$ automata $\mathcal{A}_1,\mathcal{A}_2,\ldots,\mathcal{A}_{m+1}$ are as follows. The invariant for all the $\mathcal{A}_i$ is:

(I): The states in each layer ℓ of each $\mathcal{A}_i$ represent some information after the values of $x_1,x_2,\ldots,x_\ell$ are known; layer 0 is for no information yet being known.

We first describe $\mathcal{A}_{m+1}$, which is the simplest. Since all the w_j are rationals with denominator $2^d=N^{\Theta(1)}$, there is some poly(N)-sized set Λ such that for each ℓ and any of the 2^ℓ possible values of $(x_1,x_2,\ldots,x_\ell)$, $\sum_{j\in[\ell]}w_jx_j\in\Lambda$. Each layer $\ell\in[n]$ of $\mathcal{A}_{m+1}$ has the state-set Λ, with the meaning that $\mathcal{A}_{m+1}$ will be in state $\omega\in\Lambda$ of layer ℓ iff $\sum_{j\in[\ell]}w_jx_j=\omega$. (Thus, there will be arcs from state ω in layer ℓ to state $\omega+w_{\ell+1}$ in layer $\ell+1$ with labels $0,1,\ldots,2^d\mathbf{E}[x_{\ell+1}]-1$; there will also be arcs from state ω in layer ℓ to state ω in layer $\ell+1$ with labels $2^d\mathbf{E}[x_{\ell+1}],2^d\mathbf{E}[x_{\ell+1}]+1,\ldots,2^d-1$.) So, $\mathcal{A}_{m+1}$ has poly(N) many states, and the value of the objective function $\sum_{j\in[n]}w_jx_j$ can be read off from the state in layer n that $\mathcal{A}_{m+1}$ reaches. This would help us handle (P1) above.

The automata $\mathcal{A}_i$, $i\in[m]$, help handle (P2); they are more complicated. Fix $i\in[m]$, and recall the invariant (I). $\mathcal{A}_i$ is supposed to represent the ith constraint, i.e., the ith row of matrix A. What states would be of interest to us? Recall that our analysis crucially uses the variables X_i; so we would like to be able to read off X_i from the final state that $\mathcal{A}_i$ enters. To this end, define $U(i,k,\ell)=\sum_{j\leq\ell:\ j\in L_{i,k}}x_j$; equation (2.9) shows that the value of the tuple $(U(i,1,n),U(i,2,n),\ldots,U(i,d+1,n))$ determines the value of X_i. (The value of this tuple does *not* determine which variables are to be altered, but that is not required by (P2).) So, a natural idea is to define layer ℓ of $\mathcal{A}_i$ to be the set of possible values of $\mathcal{T}(i,\ell)=(U(i,1,\ell),U(i,2,\ell),\ldots,U(i,d+1,\ell))$; it is then easy to design the state-transition rules of $\mathcal{A}_i$. Unfortunately, we can check that $\mathcal{T}(i,\ell)$ takes values from a set $W(i,\ell)$ that can have superpolynomial (in N) size. In such a case, the automata-fooling result cannot be applied directly. How to get around this? Appendix A does the following to resolve this. We first define a random variable $\mathcal{T}'(i,\ell)$ that depends only on $x_1,x_2,\ldots,x_\ell$; $\mathcal{T}'(i,\ell)$ takes values in a superpolynomial-sized set $W'(i,\ell)$. However, we will also show in Appendix A that there is an explicit poly(N)-sized $W''(i,\ell)\subseteq W'(i,\ell)$ such that:
(P3) if $\mathcal{T}'(i,\ell)\notin W''(i,\ell)$, then $\mathcal{T}'(i,\ell+1)\notin W''(i,\ell+1)$;
(P4) in our randomized rounding with the x_j chosen independently, $\Pr[\exists\ell:\ \mathcal{T}'(i,\ell)\in(W'(i,\ell)-W''(i,\ell))]\leq 1/N^3$; and
(P5) if $\mathcal{T}'(i,n)\in W''(i,n)$, then the value of X_i can be read off from $\mathcal{T}'(i,n)$.

Thus, we will be able to get away with just one state "bad$_\ell$" in layer ℓ, to represent all of $W'(i,\ell)-W''(i,\ell)$. More precisely, define the states in layer ℓ of $\mathcal{A}_i$ to be $W''(i,\ell)\cup\{\text{bad}_\ell\}$. If $\mathcal{A}_i$ arrives at state $\omega\in W''(i,\ell)$ at layer ℓ, the meaning is that $\mathcal{T}'(i,\ell)=\omega$ [and, from (P3), that $\mathcal{A}_i$ never passed through a "bad" state]; if $\mathcal{A}_i$ arrives at bad$_\ell$, then $\mathcal{T}'(i,\ell)\in(W'(i,\ell)-W''(i,\ell))$ [and $\mathcal{A}_i$ will henceforth

pass through states $\text{bad}_{\ell+1}, \text{bad}_{\ell+2}, \ldots, \text{bad}_n$]. Finally, as shown in Appendix A, the state-transitions are efficiently constructible for these automata. Now, if $\mathcal{A}_i$ ended at a non-bad state in layer n, we can get the value of X_i from it. But $\mathcal{A}_i$ may not; however, in this case which happens with probability at most N^{-3}, X_i can be at most $n \leq N$, thus negligibly affecting our control over $\mathbf{E}[X_i]$. So we will be able to choose $\epsilon = N^{-c}$ for a suitable constant c, and use (2.8) to construct the set T. Please see Appendix A for the details.

3 Multi-criteria CIPs and PIPs.

As mentioned in application (b) of §1, we will now work with multi-criteria PIPs and CIPs, generalizing our results of §2. The basic setting is as follows. Suppose, as in a PIP, we are given a system of m linear constraints $Ax \leq b$, subject to each x_j being an integer in some given range $[0, d_j]$. Furthermore, instead of just one objective function, suppose we are given a collection of non-negative vectors $\{\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_\ell\}$. The question is: given a feasible solution $\vec{x}^*$ to the LP relaxation of the constraints, how good integral feasible solutions $\vec{x}$ exist (and can be computed/approximated efficiently) that achieve a "good balance" among the different $\vec{w}_i$? For instance, we focus in this section on the case where all the $\vec{w}_i$ have equal importance; so, we could ask for approximating the smallest $\alpha \geq 1$ such that there exists an integral feasible $\vec{x}$ with $\vec{w}_i \cdot \vec{x} \geq (\vec{w}_i \cdot \vec{x}^*)/\alpha$ for all i. Similar questions can be asked for CIPs.

We now show how our algorithm and analysis of §2 help. For simplicity, we consider here the case where all the values $\vec{w}_i \cdot \vec{x}^*$ are of the same "order of magnitude": say, within a multiplicative factor of 2 of each other. It will be easy to see how to extend this to arbitrary values $\vec{w}_i \cdot \vec{x}^*$. Basically, Theorem 3.1 says that we can get essentially the same approximation guarantee of $O(\lambda)$ as in §2, even if we have up to $\exp(C_1 y^*/\lambda)$ objective functions $\vec{w}_i$. (See (2.3, 2.4) for the value of λ.)

THEOREM 3.1. *There are constants $C_1 > 0$, $K_0 > 1$ and $K_1 \geq 1$ such that the following holds. Suppose we are given:*
(a) a PIP's constraints $Ax \leq b$;
(b) a feasible solution $\vec{x}^*$ to the LP relaxation of these constraints, and
(c) a collection of non-negative vectors $\{\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_\ell\}$ such that for some y^*, $\vec{w}_i \cdot \vec{x}^* \in [y^*/2, y^*]$ for all i.

Let λ be as in (2.3, 2.4), and suppose ℓ, the number of given $\vec{w}_i$, is at most $\exp(C_1 y^*/\lambda)$. Then, there exists an integral feasible solution $\vec{z}$ to the constraints in (a), such that $\vec{w}_i \cdot \vec{z} \geq \vec{w}_i \cdot \vec{x}^*/(2\lambda)$ for all i; furthermore, we can compute such a $\vec{z}$ in NC.

Proof. We first present the *RNC* version. As in §2, it can be shown that the worst case is when $x_j^* \in [0, 1]$ for all j. As described in §2, the basic algorithm is to conduct a randomized rounding with $\Pr[x_j = 1] = x_j^*/\lambda$ for each j, and then to conduct our alteration. We use the same notation as in §2. For each $i \in [\ell]$, we have $\mathbf{E}[\vec{w}_i \cdot \vec{x}] = \vec{w}_i \cdot \vec{x}^*/\lambda \geq y^*(2\lambda)$, by assumption (b) of the theorem. Usage of (2.2) shows that for a certain absolute constant $C' > 0$,

$$\Pr[\vec{w}_i \cdot \vec{x} \leq \vec{w}_i \cdot \vec{x}^*/(1.5\lambda)] \leq \exp(-C' y^*/\lambda).$$

So, the probability of existence of some $i \in [\ell]$ for which "$\vec{w}_i \cdot \vec{x} \leq \vec{w}_i \cdot \vec{x}^*/(1.5\lambda)$" holds, is at most

$$\ell \cdot \exp(-C' y^*/\lambda) \leq \exp(-(C' - C_1) y^*/\lambda). \tag{3.11}$$

We may assume that $y^*/\lambda \geq (\ln 2)/C_1$, since otherwise $\ell \leq \exp(C_1 y^*/\lambda)$ implies that $\ell = 1$, leaving us with the "one objective function" case, which we have handled in §2. Therefore we have from (3.11) that

$$\Pr[\exists i \in [\ell] : \ \vec{w}_i \cdot \vec{x} \leq \vec{w}_i \cdot \vec{x}^*/(1.5\lambda)] \leq \exp(-(C' - C_1) \cdot (\ln 2)/C_1). \tag{3.12}$$

Theorem 2.1(b) shows that $\mathbf{E}[\sum_{i \in [m]} X_i] \leq K_5 mp$; by Markov's inequality,

$$\Pr[\sum_{i \in [m]} X_i \geq K_5 C_2 mp] \leq C_2^{-1}$$

for any $C_2 \geq 1$. Thus if $C_1 < C'$ and C_2 is a large enough constant such that the sum of C_2^{-1} and (3.12) is less than, and bounded away, from 1, then with positive constant probability, we have: (i) for all $i \in [\ell]$, $\vec{w}_i \cdot \vec{x} > \vec{w}_i \cdot \vec{x}^*/(1.5\lambda)$ and (ii) the total number of altered variables is at most $K_5 C_2 mp$. This implies that for all $i \in [\ell]$, the value of $\vec{w}_i \cdot \vec{x}$ after alteration is at least $\vec{w}_i \cdot \vec{x}^*/(1.5\lambda) - K_5 C_2 mp$, which can be ensured to be at least $\vec{w}_i \cdot \vec{x}^*/(2\lambda)$ by taking K_0 and K_1 large enough, using the definition of p from Theorem 2.1, and using the fact that $\vec{w}_i \cdot \vec{x}^* \geq y^*/2$ for all i.

The derandomization is very similar to that of §2.4; we now have ℓ automata, one for each $\vec{w}_i$, instead of the single automaton $\mathcal{A}_{m+1}$ there. The automata $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m$ remain the same. Thus we have proved Theorem 3.1.

The primary reason why the above works is that the tail bound on $\sum_{i \in [m]} X_i$ works in place of approaches such as ours in [38] which try to handle the very low (in some cases exponentially small) probability of satisfying *all* the m constraints $Ax \leq b$.

Similar results hold for CIPs. We shall also show in the full version that by using the FKG inequality [14], we can get somewhat better bounds on ℓ than the bound $\exp(C_1 y^*/\lambda)$ above, if we only require *RNC* algorithms.

4 Approaching the Turán bound in NC.

To find a large IS in a given graph, we derandomize the following algorithm of Spencer [37]: randomly permute the vertices, and add a vertex v to the IS iff no neighbor of v precedes v in the random order. It can be seen that the expected size of an IS produced here is $T_1(G)$. Letting $\ell = \lceil 3 \log_2 n \rceil$, we first observe that this expectation analysis changes very little if each v picks a label $X(v)$ uniformly at random and independently from $\{0, 1, \ldots, 2^\ell - 1\}$, and gets added to the IS iff $X(v) < X(w)$ for all neighbors w of v. So we focus on finding a "good" set of these labels in NC; our basic approach is as follows. Number the vertices from 1 to n, and write each $X(v)$ in binary as $x_{v,1}x_{v,2}\cdots x_{v,\ell}$; for $1 \leq i \leq \ell$, define the vector $Y(i) = (x_{1,i}, x_{2,i}, \ldots, x_{n,i})$. We aim to find "good" choices for $Y(1), Y(2), \ldots, Y(\ell)$ one-by-one, using an "approximate method of conditional probabilities" due to [2]. If the maximum degree Δ of the graph is at most $O(\log n)$, we would be able to show that it suffices to pick each $Y(i)$ from a suitable polynomial-sized *small-bias sample space*. (See [29] for the important notion of small-bias spaces; specifically, we use $O(\log n)$-wise $n^{-\psi}$-biased sample spaces for a large enough constant ψ.) If $\Delta \gg \log n$, we adapt certain large-deviation results [5] to the setting of small-bias spaces, and show how to implement an approximate method of conditional probabilities here.

5 Rounding Group Steiner solutions.

Group Steiner Tree is a network design problem motivated by VLSI design [34]. Given an undirected graph $G = (V, E)$, a collection of subsets $S_1, S_2, \ldots, S_k$ of V, and a cost $c_f \geq 0$ for each edge $f \in E$, the problem is to construct a minimum-cost tree in G that spans at least one vertex from each S_i. The case $k = 1$ corresponds to the Steiner tree problem; we can model set cover even with G being a star graph here.

Let $n = |V|$ and $N = \max_i |S_i|$. We start by sketching the $O(\log n \log\log n \log N \log k)$–approximation algorithm of [15] for this problem. First, the results of Bartal [4] are used to appropriately reduce the problem to the case where G is a *tree* G', with an $O((\log n) \cdot \log\log n)$ factor loss in the approximation bound. One can also specialize to the rooted version, where a particular vertex r of G' must be included in the constructed tree. So we need to construct a min-cost tree T contained in the rooted tree G', such that r is included and at least one vertex of each S_i is spanned. We can also assume w.l.o.g. that all nodes in $\bigcup_i S_i$ are leaves of G' (please see [15] for the proofs). The problem can be written as an integer linear program, with an indicator variable $x(f)$ for including edge f in the tree T; the objective is to minimize $\sum_f c_f x(f)$. In the LP relaxation, we allow each $x(f)$ to lie in $[0, 1]$; interpreting $x(f)$ as the capacity of edge f, the LP is to choose these capacities such that for each *individual* S_i, the capacities can support one unit of flow from r to S_i. (It is *not* required that the capacities be able to support a unit flow *simultaneously* for all the S_i.) Let $\{x^*(f) : f \in E\}$ denote an optimal LP solution; let $y^* = \sum_f c_f x^*(f)$. The randomized rounding of [15] is as follows. Each edge f incident with the root r is chosen with probability $x^*(f)$. Every other edge f is chosen with probability $x^*(f)/x^*(g)$, where g is the "parent edge" of f. (W.l.o.g., $x^*(f) \leq x^*(g)$ holds.) All these choices are made independently. Finally, we choose the connected component of the chosen edges that includes r. Briefly, the analysis of [15] is as follows. First, the expected cost of the chosen tree is y^*. Next, it is shown via Janson's inequality [19] that for each given i, the probability that at least one vertex of S_i is covered by the above process, is $\Omega(1/\log N)$. With these two properties, it is argued in [15] that if we repeat our rounding $C \log N \log k$ times for a suitably large constant C and include the subgraphs chosen in these iterations, we get a desired tree of cost $O(y^* \log N \log k)$ with probability at least $1/2$.

This algorithm for trees G' is given a sequential derandomization in [9]. We provide an NC derandomization, as follows. Scale *down* all $x^*(f)$ for the parent edges f of all leaves, by $C' \log N$ for a certain constant C', and imagine running one iteration of the above randomized rounding. The scaling down enables us to prove, by a truncated inclusion-exclusion argument, that each given S_i is hit with probability $\Omega(1/\log N)$; also, the expected cost of the tree chosen is at most y^*. The truncated inclusion-exclusion argument, as well as a "depth-shrinking" approach of [15], let us show that we can choose the random variables for the randomized rounding from a suitable polynomial-sized space: one generating almost $O(\log N)$-wise independent random variables [12]. Thus, we can choose a "good" seed for the random process in NC, by an exhaustive search of this space. We then show that repeating this process $O(\log N \log k)$ times suitably as above, solves the problem.

Acknowledgements. We thank Moses Charikar, Chandra Chekuri, Magnús Halldórsson, Goran Konjevod, Seffi Naor, Jaikumar Radhakrishnan, R. Ravi and K. V. Subrahmanyam for helpful discussions.

References

[1] N. Alon, L. Babai, and A. Itai, *A fast and simple randomized parallel algorithm for the maximal independent set problem*, Journal of Algorithms, 7 (1986), pp. 567–583.

[2] N. Alon and M. Naor, *Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions*, Algorithmica, 16 (1996), pp. 434–449.

[3] N. Alon and J. H. Spencer, *The Probabilistic Method*, John Wiley & Sons, Inc., New York, 1992.

[4] Y. Bartal, *On approximating arbitrary metrics by tree metrics*, in Proc. ACM Symposium on Theory of Computing, pp. 161–168, 1998.

[5] M. Bellare and J. Rompel, *Randomness-efficient oblivious sampling*, in Proc. IEEE Symposium on Foundations of Computer Science, pp. 276–287, 1994.

[6] B. Berger and J. Rompel, *Simulating ($\log^c n$)-wise independence in NC*, Journal of the ACM, 38 (1991), pp. 1026–1046.

[7] B. Berger, J. Rompel, and P. W. Shor, *Efficient NC algorithms for set cover with applications to learning and geometry*, Journal of Computer and System Sciences, 49 (1994), pp. 454–477.

[8] A. Broder, M. Charikar, and M. Mitzenmacher, *A derandomization using min-wise independent permutations*, in Proc. International Workshop on Randomization and Approximation Techniques in Computer Science, pp. 15–24, 1998.

[9] M. Charikar, C. Chekuri, A. Goel, and S. Guha, *Rounding via trees: deterministic approximation algorithms for Group Steiner trees and k-median*, in Proc. ACM Symposium on Theory of Computing, pp. 114–123, 1998.

[10] D. Cook, V. Faber, M. V. Marathe, A. Srinivasan and Y. J. Sussmann, *Low-bandwidth routing and electrical power networks*, in Proc. International Colloquium on Automata, Languages, and Programming, pp. 604–615, 1998.

[11] F. Ergün, R. Sinha and L. Zhang, *QoS routing with performance-dependent costs*, in Proc. IEEE Conference on Computer Communications, pp. 137–146, 2000.

[12] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković, *Efficient approximations for product distributions*, Random Structures & Algorithms, 13 (1998), pp. 1–16.

[13] P. Erdős, *On the graph theorem of Turán* (in Hungarian), Mat. Lapok., 21 (1970), pp. 249–251.

[14] C. M. Fortuin, J. Ginibre, and P. N. Kasteleyn, *Correlational inequalities for partially ordered sets*, Communications of Mathematical Physics, 22 (1971), pp. 89–103.

[15] N. Garg, G. Konjevod, and R. Ravi, *A polylogarithmic approximation algorithm for the Group Steiner tree problem*, in Proc. ACM-SIAM Symposium on Discrete Algorithms, pp. 253–259, 1998.

[16] M. Goldberg and T. Spencer, *Constructing a maximal independent set in parallel*, SIAM J. Disc. Math., 2 (1989), pp. 322–328.

[17] M. Goldberg and T. Spencer, *An efficient parallel algorithm that finds independent sets of guaranteed size*, SIAM J. Disc. Math., 6 (1993), pp. 443–459.

[18] P. Indyk, *A small approximately min-wise independent family of hash functions*, in Proc. ACM-SIAM Symposium on Discrete Algorithms, pp. 454–456, 1999.

[19] S. Janson, T. Łuczak, and A. Ruciński, *An exponential bound for the probability of nonexistence of a specified subgraph in a random graph*, in Random Graphs '87 (M. Karoński, J. Jaworski and A. Ruciński, eds.), John Wiley & Sons, Chichester, pp. 73–87, 1990.

[20] D. R. Karger and D. Koller, *(De)randomized construction of small sample spaces in NC*, Journal of Computer and System Sciences, 55 (1997), pp. 402–413.

[21] R. M. Karp and A. Wigderson, *A fast parallel algorithm for the maximal independent set problem*, Journal of the ACM, 32 (1985), pp. 762–773.

[22] S. G. Kolliopoulos and C. Stein, *Approximating disjoint-path problems using greedy algorithms and packing integer programs*, in Proc. MPS Conference on Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science 1412, Springer-Verlag, pp. 153–168, 1998.

[23] J. Könemann and R. Ravi, *A matter of degree: improved approximation algorithms for degree-bounded minimum spanning trees*, in Proc. ACM Symposium on Theory of Computing, pp. 537–546, 2000.

[24] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, SIAM J. Comput., 15 (1986), pp. 1036–1053.

[25] M. Luby and N. Nisan, *A parallel approximation algorithm for positive linear programming*, in Proc. ACM Symposium on Theory of Computing, pp. 448–457, 1993.

[26] S. Mahajan, E. A. Ramos, and K. V. Subrahmanyam, *Solving some discrepancy problems in NC*, in Proc. Annual Conference on Foundations of Software Technology & Theoretical Computer Science, Lecture Notes in Computer Science 1346, Springer-Verlag, pp. 22–36, 1997.

[27] R. Motwani, J. Naor, and M. Naor, *The probabilistic method yields deterministic parallel algorithms*, J. Comput. Syst. Sci., 49 (1994), pp. 478–516.

[28] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[29] J. Naor and M. Naor, *Small–bias probability spaces: efficient constructions and applications*, SIAM J. Comput., 22 (1993), pp. 838–856.

[30] N. Nisan, *Pseudorandom generators for space-bounded computation*, Combinatorica, 12 (1992), pp. 449–461.

[31] P. Raghavan, *Probabilistic construction of deterministic algorithms: approximating packing integer programs*, Journal of Computer and System Sciences, 37 (1988), pp. 130–143.

[32] P. Raghavan and C. D. Thompson, *Randomized rounding: a technique for provably good algorithms and algorithmic proofs*, Combinatorica, 7 (1987), pp. 365–374.

[33] S. Rajagopalan and V. V. Vazirani, *Primal-dual RNC approximation algorithms for (multi)-set (multi)-cover and covering integer programs*, in Proc. IEEE Symposium on Foundations of Computer Science, pp. 322–

331, 1993.
[34] G. Reich and P. Widmayer, *Beyond Steiner's problem: a VLSI oriented generalization*, in Proc. Graph Theoretic Concepts of Computer Science, Lecture Notes in Computer Science 411, Springer-Verlag, pp. 196–210, 1990.
[35] D. B. Shmoys and É. Tardos, *An approximation algorithm for the generalized assignment problem*, Math. Programming A, 62 (1993), pp. 461–474.
[36] P. Slavík, *A tight analysis of the greedy algorithm for set cover*, in Proc. ACM Symposium on Theory of Computing, pp. 435–441, 1996.
[37] J. H. Spencer, *The probabilistic lens: Sperner, Turán and Brégman revisited*, in A Tribute to Paul Erdős (A. Baker, B. Bollobás, A. Hajnal, editors), Cambridge University Press, pp. 391–396, 1990.
[38] A. Srinivasan, *Improved approximation guarantees for packing and covering integer programs*, SIAM J. Comput., 29 (1999), pp. 648–670.
[39] P. Turán, *On the theory of graphs*, Colloq. Math., 3 (1954), pp. 19–30.

Appendix

A Shrinking the automata to poly(N) size.

Let $C_0 > 0$ be a sufficiently large constant. We will first ensure that $b_i \leq C_0 \log N$ for all i, in the given PIP: if $b_i > C_0 \log N$ for some i, just divide the constraint "$(Ax)_i \leq b_i$" by $b_i/(C_0 \log N)$. (This will change the value of the parameter λ that we choose in our randomized rounding, iff B was greater than $C_0 \log N$ before this step. But even in this case, note from (2.3, 2.4) that we will have $\lambda = \Theta(1)$ before and after this step; this is because we can assume that $y^* \geq 1/2$ w.l.o.g. by a simple argument that will be shown in the full version. So, this step of ensuring that $b_i \leq C_0 \log N$ for all i, changes our approximation guarantee by at most a multiplicative constant factor.) Define $k_0 = \lceil \log(C_0 \log N) \rceil$, and $U'(i,\ell) = \sum_{j \leq \ell:\ j \in M_{i,k_0}} x_j$. Please see §2.3 for the meaning of M_{i,k_0}, and §2.4 for the meaning of $U(i,k,\ell)$. We will work with the tuples $\mathcal{T}'(i,\ell)$ defined to be

$$(U(i,1,\ell), U(i,2,\ell), \ldots, U(i,k_0-1,\ell), U'(i,\ell)).$$

As mentioned in §2.4, $\mathcal{T}'(i,\ell)$ depends only on $x_1, x_2, \ldots, x_\ell$, but takes values in a set $W'(i,\ell)$ that could be superpolynomial-sized. The following lemma will help us define the useful poly(N)-sized $W''(i,\ell) \subseteq W'(i,\ell)$ alluded to in §2.4:

LEMMA A.1. *Suppose the constant C_0 is large enough. Consider our randomized rounding with the x_j chosen independently, and fix any i. Then, the probability of existence of an $\ell \in [n]$ for which the event "$(Q1) \vee (Q2)$" holds is at most $1/N^3$. Here, $(Q1) \equiv (U'(i,\ell) > b_i)$, and (Q2) is the event that there exists $k \in [k_0 - 1]$ such that $U(i,k,\ell) > b_i 2^{k-1} + C_0 2^{k-1} \log N \log\log N$.*

Proof. If $U'(i,\ell) > b_i$, then $U'(i,n) > b_i$; i.e., $Z_i \geq k_0$. Lemma 2.1(a) shows that for large enough C_0,

$$\Pr[Z_i \geq k_0] \leq 1/(2N^3). \tag{1.13}$$

Fix $k \in [k_0 - 1]$. If $U(i,k,\ell) > b_i 2^{k-1} + C_0 2^{k-1} \log N \log\log N$, then $U(i,k,n) > b_i 2^{k-1} + C_0 2^{k-1} \log N \log\log N$, so $X_i \geq C_0 \log N \log\log N$. If C_0 is large enough, then Theorem 2.1(a) shows that $\Pr[X_i \geq C_0 \log N \log\log N] \leq (2k_0 N^3)^{-1}$. Summing this over all $k \in [k_0 - 1]$ and adding with (1.13) completes the proof of Lemma A.1.

Next, since all the $A_{i,j}$ are rationals with denominator $2^d = N^{\Theta(1)}$, there is some poly(N)-sized set Λ' such that for all i and ℓ, $U'(i,\ell) \in \Lambda'$. Define $W''(i,\ell)$ to be the set of all $(v_1, v_2, \ldots, v_{k_0-1}, v') \in (Z_+^{k_0-1} \times \Lambda')$ such that

$$(v' \leq b_i) \wedge (\forall k,\ v_k \leq b_i 2^{k-1} + C_0 2^{k-1} \log N \log\log N)$$

holds.

Since all the $U(i,k,\ell)$ and $U'(i,\ell)$ are non-decreasing when viewed as functions of only ℓ, it is easy to check that property (P3) of §2.4 holds. (P4) follows from Lemma A.1. (P5) is true since "$U'(i,n) \leq b_i$" implies that $Z_i \leq k_0 - 1$; hence, the value of X_i can be inferred just by knowing the values of $U(i,k,n)$ for all $k \in [k_0 - 1]$. Finally, to see why the crucial bound $|W''(i,\ell)| \leq$ poly(N) holds, recall from our preprocessing above that $b_i \leq C_0 \log N$; so, $|W''(i,\ell)|$ is at most

$$|\Lambda'| \cdot \prod_{k \in [k_0-1]} (1 + b_i 2^{k-1} + C_0 2^{k-1} \log N \log\log N);$$

i.e., $|W''(i,\ell)| \leq |\Lambda'| \cdot (\log N)^{O(\log\log N)} =$ poly(N).

Thus, we will be able to construct the reduced polynomial-sized automaton $\mathcal{A}_i$ as sketched in §2.4. If $\mathcal{A}_i$ enters a bad state (which happens with probability at most $1/N^3$), we simply set all the variables to 0. Suppose we run the automata-fooling algorithm of [20, 26] on $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{m+1}$. Recall the parameters α and β_i from (P1) and (P2). Then, the following three facts: (i) condition (2.8), (ii) for some constant c', $|\mathcal{A}_i| \leq N^{c'}$ for all i, and (iii) even on entering a bad state in $\mathcal{A}_i$, which happens with probability at most $1/N^3$, the value of X_i is at most n, together guarantee that $\alpha \geq y^*/\lambda - N^{c'}\epsilon$ and $\beta_i \leq O(p + N^{c'}\epsilon + 1/N^2)$. Thus, choosing $\epsilon = N^{-c}$ for a large enough constant c will ensure that (2.10) holds.