Improved parallel approximation of a class of integer programming problems

Noga Alon* Aravind Srinivasan†

Abstract. We present a method to derandomize RNC algorithms, converting them to NC algorithms. Using it, we show how to approximate a class of NP-hard integer programming problems in NC, to within factors better than the current-best NC algorithms (of Berger & Rompel and Motwani, Naor & Naor); in some cases, the approximation factors are as good as the best-known sequential algorithms, due to Raghavan. This class includes problems such as global wire-routing in VLSI gate arrays and a generalization of telephone network planning in SONET rings. Also for a subfamily of the "packing" integer programs, we provide the first NC approximation algorithms; this includes problems such as maximum matchings in hypergraphs, and generalizations. The key to the utility of our method is that it involves sums of *superpolynomially many* terms, which can however be computed in NC; this superpolynomiality is the bottleneck for some earlier approaches, due to Berger & Rompel and Motwani, Naor & Naor.

Keywords. De-randomization, integer programming, parallel algorithms, approximation algorithms, rounding theorems, randomized rounding, linear programming, linear relaxation, combinatorial optimization.

*School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel. Part of this work was done while visiting the Institute for Advanced study, School of Mathematics, Princeton, NJ 08540, USA, supported in part by the Sloan Foundation, grant No. 93-6-6 and by the Fund for Basic Research administered by the Israel Academy of Sciences. E-mail: `noga@math.tau.ac.il`.

†Dept. of Information Systems & Computer Science, National University of Singapore, Singapore 119260, Republic of Singapore. Work done in part at the National University of Singapore, at DIMACS (supported in part by NSF-STC91-19999 and by support from the N.J. Commission on Science and Technology), and at the Institute for Advanced Study, Princeton (supported in part by grant 93-6-6 of the Alfred P. Sloan Foundation). E-mail: `aravind@iscs.nus.sg`.

1 Introduction

Derandomization is the development of general tools to derive efficient *deterministic* algorithms from their randomized counterparts. We present here a method to derandomize a class of randomized parallel algorithms. This tool is then used to derive good NC approximation algorithms for a class of integer programming problems, matching the approximation factors of the best-known RNC algorithms (and, in some cases, the best-known sequential algorithms also) and improving on the guarantees provided by known NC algorithms.

Research in derandomization is motivated by at least three reasons. First, though randomized algorithms perform well empirically, the fact that computers do not use "real" random sources prevents randomized algorithms from having a sound footing; indeed, it has been shown that if randomized algorithms such as randomized Quicksort are not implemented carefully when used with some existing pseudorandom generators, their expected running times can be high (Karloff & Raghavan [12]). In fact, there have been reports of Monte-Carlo simulations giving quite different results under different random-number generators (Ferrenberg, Landau & Wong [8]), and direct implementations of certain RNC algorithms for list ranking (Hsu [10]) and graph connectivity (Hsu, Ramachandran & Dean [11]) taking longer time than expected due to the pseudorandom nature of computer-generated "random" bits. Second, especially in critical applications, it is preferable to have absolute certainty if possible, rather than probabilistic guarantees. Finally, such research makes progress toward settling the complexity-theoretic question of how much computational power can be provided by randomness. In addition, the useful mathematical tools arising from research in this area have also provided it further impetus.

In this paper, we present a simple tool to convert a class of RNC algorithms to NC algorithms, building on some existing tools for derandomization (Alon & Naor [4], Schmidt, Siegel & Srinivasan [22]). A key property of this tool is that while, as in Luby [14], Berger & Rompel [5] and Motwani, Naor & Naor [16], it uses the method of conditional probabilities in parallel, its structure enables it to handle a conditional estimator that is a sum of *superpolynomially many* terms, which indeed is a bottleneck for the techniques of [5, 16]. The bottleneck arises from the fact that the work of [14, 5, 16] essentially assigns one processor to each term of the conditional estimator, thus giving them the power of handling estimators which have only polynomially many terms. We expect our method to be useful in other contexts too.

The first application of our method is to approximate a class of integer programming (IP) problems–*minimax integer programs*–by solving their linear programming (LP) relaxations (approximately in parallel, via Luby & Nisan [15]) and then employing randomized rounding as in Raghavan & Thompson [21]; our task is to do the rounding in NC.

For any non-negative integer k, let $[k]$ denote the set $\{1, 2, \ldots, k\}$.

Definition 1 *A minimax integer program (MIP) in our case has variables W and $\{x_{i,j} : i \in [\ell], j \in [n_i]\}$, for some integers $\{n_i\}$. Let $N = \sum_{i\in[\ell]} n_i$ and let x denote the N-dimensional vector of the variables $x_{i,j}$ (arranged in any fixed order). An MIP seeks to minimize W, subject to:*
(i) Equality constraints: $\forall i \in [\ell]$ $\sum_{j\in[n_i]} x_{i,j} = 1$;
(ii) a system of linear inequalities $Ax \le \vec{W}$, where $A \in [0,1]^{m\times N}$ and $\vec{W}$ is the m-dimensional vector with the variable W in each component;
(iii) Integrality constraints: $x_{i,j} \in \{0,1\}$ $\forall i, j$, and
(iv) W can be any non-negative real.

To see what problems MIPs model, note, from constraints (i) and (iii) of MIPs, that for all i, any feasible solution will make the set $\{x_{i,j} : j \in [n_i]\}$ have precisely one 1, with all other elements being 0; MIPs thus model many "choice" scenarios. Consider, *e.g.*, global routing in VLSI gate arrays [21]; this can be generalized as follows (Chapter 3 of Raghavan [19]). We are given an undirected graph G with m edges, a set of pairs of vertices $\{(s_i, t_i) : 1 \le i \le \ell\}$, and $\forall i \in [\ell]$, a set P_i of paths in G, each connecting s_i to t_i. The objective is to connect each s_i with t_i using exactly one path from P_i, so that the maximum number of paths which use any edge in G, is minimized. An MIP formulation is obvious, with $x_{i,j}$ being the indicator variable for picking the jth path in P_i. Similarly, the vector-selection problem of [21], and many discrepancy-type problems, are all modeled by MIPs; many MIP instances, *e.g.*, global routing, are NP-hard. This has led to the study of efficient approximation algorithms for MIPs. A useful approach in this regard has been to start with the *linear programming (LP) relaxation* of a given MIP, which lets $x_{i,j} \in [0,1]$ for each i, j, as opposed to the stringent $x_{i,j} \in \{0,1\}$. Thus, such an LP relaxation is a linear program and hence is solvable in polynomial time.

When the optimum C^* of the LP relaxation of a given MIP is at most $O(\log m)$, we present an NC approximation algorithm for the MIP which has a better approximation guarantee than does previous work [5, 16], and matches that of the known sequential method (Raghavan [20]), to within a $(1 + o(1))$ factor. Concretely, we derive NC algorithms that deliver integral feasible solutions with objective function value

$$O(\frac{\log m}{\max\{1, \log((\log m)/C^*)\}}),$$

for families of MIP instances where C^* is $O(\log m)$. However, a better *existential* result–that the integrality gap of the LP relaxation of *sparse* MIPs is better than that proven by [20]–is known (Srinivasan [24]); the results of [20] are the current-best *constructive* approximations. If $C^* = O(\log m)$, we always improve on the approximation factor of [5, 16] by at least a $\log^{\epsilon} m$ factor for some fixed $\epsilon > 0$. This improvement increases with decreasing C^*; *e.g.*, if $C^* = O(1)$, the improvement is $\Theta(\log^{\epsilon} m \log\log m)$.

As another instance of MIPs, consider a generalization of the problem of telephone network planning in bidirectional SONET rings: the "ring loading problem" (Schrijver, Seymour & Winkler [23]). Given a ring and a traffic demand between every pair of vertices, all traffic between them must be routed one way or the other around the ring. For a given set of traffic demands, the problem is to route the traffic so that the maximum traffic on any link is minimized; the generalization to arbitrary networks involves, for every pair of vertices, an allowed set of paths of which exactly one must be chosen. In this case, simple scaling shows that our method delivers better approximation factors than does [5, 16], if the optimum objective function value is within an $O(\log m)$ factor of the maximum traffic demand between any pair of vertices (m is the number of edges in the network).

Our method also applies to the class of *packing integer programs* (PIPs), which model many problems in combinatorial optimization; most of these again are NP-hard. A PIP seeks to maximize $c^T \cdot x$ subject to $Ax \leq b$, where $A \in [0,1]^{m \times n}$, b is an m-vector and c is an n-vector such that *the entries of* b *and* c *are non-negative*, with the integrality constraint $x_j \in \{0, 1, \ldots, d_j\}$ for every entry x_j of x; some of the d_js could also be infinite. Here in fact for a subclass (wherein which each b_i is at most $O(\log(m + n)))$, we derive the first NC approximation algorithms with any "reasonable" performance guarantee; our guarantees again match those of the best-known RNC algorithms (these RNC algorithms are directly got by combining [15] and [21]). However, in terms of *sequential* algorithms, better approximation guarantees are known now (Srinivasan [25, 24]).

We now describe our approximation results for packing; our method works best when each b_i is at most $O(\log(m+n))$. Analogously to MIPs, the LP relaxation of a PIP lets each x_j be a real lying in $[0, d_j]$. Suppose C^* is the optimal objective function value of the LP relaxation of a given PIP; note that C^* is at least as big as the optimal value of the PIP. For PIPs, the work of [21, 20] presents sequential algorithms that deliver integral solutions of value $\Omega(C^*/m^{1/B})$ and $\Omega(C^*/m^{1/(B+1)})$ respectively, if $A \in [0,1]^{m\times n}$ and $A \in \{0,1\}^{m\times n}$. If each b_i is at most $O(\log(m+n))$, we present an NC algorithm that matches this bound. This is the first NC approximation algorithm with any reasonable performance guarantee when the b_is are all $O(\log(m+n))$ (the algorithms of [5, 16] will not necessarily satisfy the constraints, even if $b_i = O(\log(m+n))$ for a few values of i). On the other hand, the results of [5, 16] are generally better when all the b_is grow faster than $\log(m+n)$.

An important class of PIPs is matching problems on hypergraphs. Recall that a hypergraph $H = (V, E)$ is a collection of subsets E (hyperedges) of a finite set V (vertices). A *matching* in H is a collection of hyperedges from E such that no vertex occurs in more than one edge; a basic and well-known NP-hard problem is to find a matching of maximum cardinality in the given hypergraph. A generalization of this notion is that of k-matchings, for integral $k \geq 1$ (see Lovász [13]): here, we allow each vertex to be present in at most k edges in the subcollection. It is easily seen that the k-matching problem can be written as a packing integer program with the right-hand-side constants b_i all equaling k and thus our method applies if $k = O(\log(m+n))$. Thus even for the special (and basic) case of $k = 1$ where we look for a maximum matching, our method yields the first NC approximation algorithms with any reasonable performance guarantee; we present integral feasible solutions of value $\Omega(C^*/\sqrt{m})$, where m is the number of vertices in the hypergraph. This matches the sequential bound of [20, 1]. Similar results hold for k-matching, when $k = O(\log(m+n))$.

Returning to the general packing formulation, our method works even if all the b_is are not $O(\log(m+n))$, via straightforward scaling–dividing constraint i by $b_i/(c\log(m+n))$ so that b_i now equals $c\log(m+n)$, for some desired constant c. Thus if some of the b_is are $O(\log(m+n))$ and some are $\omega(\log(m+n))$ (growing strictly faster than $\log(m+n)$), we still get the first reasonable NC approximation algorithms. However, for such general packing integer programs, we get approximation ratios which are within any desired constant $c > 1$ of the sequential guarantees of [20]; we do not see how to improve this to $(1 + o(1))$.

Thus, the contributions of this work are to present a parallel derandomization technique and to apply it to derive improved NC approximation algorithms for a class of IP problems; some of these are the first NC approximation algorithms for a class of problems. One such important problem is the NP-hard problem of finding maximum matchings in hypergraphs, for which our performance guarantee matches that of the current-best RNC algorithms; we get the first NC algorithms with any reasonable performance guarantee for this and related problems. We also expect that the derandomization tool will be used and extended in the future in other domains.

2 Preliminaries

Notation. We denote "random variable" by "r.v.". For real x and any positive integer r, we define, as usual,

$$\binom{x}{r} \doteq (x(x-1)\cdots(x-r+1))/r!;$$

$\binom{x}{0} \doteq 1$.

We start with a recent tool [22]–a new look at the Chernoff-Hoeffding (CH) bounds for tail probabilities [7, 9]. We define, for $z = (z_1, z_2, \ldots, z_n) \in \Re^n$, a family of symmetric polynomials $S_j(z), j = 0, 1, \ldots, n$, where $S_0(z) \equiv 1$, and for $1 \leq j \leq n$,

$$S_j(z) \doteq \sum_{1 \leq i_1 < i_2 \cdots < i_j \leq n} z_{i_1} z_{i_2} \cdots z_{i_j}.$$

Then, a small extension of a basic theorem of [22] that we will need is

Theorem 1 ([22]) *Given random variables $X_1, \ldots, X_n \in [0,1]$, let $X = \sum_{i=1}^n X_i$ and $\mu = E[X]$. (a) For any $\delta > 0$, any nonempty event Z and any $k \leq \mu(1+\delta)$, $Pr(X \geq \mu(1+\delta)|Z) \leq E[Y_k|Z]$, where*

$$Y_k = S_k(X_1, \ldots, X_n)/\binom{\mu(1+\delta)}{k}.$$

(b) If the X_is are independent and $k = \lceil \mu\delta \rceil$, then $Pr(X \geq \mu(1+\delta)) < E[Y_k] \leq G(\mu, \delta)$, where

$$G(\mu, \delta) = \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu.$$

Proof. Suppose $r_1, r_2, \ldots r_n \in [0,1]$ satisfy $\sum_{i=1}^n r_i \geq a$. Then, a simple algebraic proof is given in [22], for the fact that for any non-negative integer $k \leq a$,

$S_k(r_1, r_2, \ldots, r_n) \geq \binom{a}{k}$. (This is immediate if each r_i is either 0 or 1, and takes a little more work if $r_i \in [0,1]$.) This, then, clearly holds even given the occurrence of any (positive probability) event Z. Hence,

$$Pr(X \geq \mu(1+\delta)|Z) \leq Pr(Y_k \geq 1|Z) \leq E[Y_k|Z],$$

where the second inequality follows from Markov's inequality. A proof of (b) is given in [22]. □

A simple but crucial property of the functions S_k is as follows. Suppose random variables $X_1, X_2, \ldots, X_n$ take on only non-negative values. Then for any integer $k \geq 1$, if we expand $E[S_k(X_1, \ldots, X_n)]$ as a sum of $\binom{n}{k}$ terms by applying linearity of expectation, each term in the sum is *non-negative*. More precisely, focus on a generic term $E[X_{i_1} X_{i_2} \cdots X_{i_k}]$ in the expansion of $E[S_k(X_1, \ldots, X_n)]$. Suppose that for some parameter $\epsilon > 0$, we are able to show that this generic term is within a relative error of $\pm\epsilon$ from the value of such a term, had the X_i been independent (with the same marginals–individual distributions–as the variables X_i on hand). Then, it is easy to see that $E[S_k(X_1, \ldots, X_n)]$ is at most $(1+\epsilon)$ times what it would be in the "independent" case; this follows easily from the fact that the coefficient of $E[X_{i_1} X_{i_2} \cdots X_{i_k}]$ in $E[S_k(X_1, \ldots, X_n)]$ is *non-negative* (one). Such a property is not enjoyed by other tools such as the k-th moment inequality, that are used in derandomization approaches. This is one of the main reasons for the functions S_k helping us, as will be borne out by the analyses of Sections 4 and 5.

Definition 2 *For $x \in (0,1)$, define $D(\mu, x)$ such that $G(\mu, D(\mu, x)) = x$ (D is well-defined).*

The following fact is easily checked; we will primarily be interested in its first case in this paper.

Fact 1 *There is a constant $c > 0$ such that*

$$D(\mu, x) = \Theta(\log(x^{-1})/(\mu \log(\log(x^{-1})/\mu)))$$

if $\mu \leq c \ln(x^{-1})$, and

$$\Theta(\sqrt{\log(x^{-1})/\mu})$$

otherwise.

A simple but very useful fact we will need is that though $S_k(r_1, \ldots, r_n)$ has superpolynomially many (in n) terms if $k = k(n) \to \infty$, it is efficiently computable:

Lemma 1 *For any* $\vec{r} = (r_1, \ldots, r_n)$ *and* $1 \leq k \leq n$, $S_k(r_1, \ldots, r_n)$ *is computable in* NC.

PROOF. The coefficient of z^{n-k} in the polynomial $f_{\vec{r}}(z) \doteq \prod_{i=1}^{n}(z - r_i)$ is $(-1)^k S_k(r_1, \ldots, r_n)$. Evaluating f at $(n+1)$ distinct points and solving a system of linear equations gives us $S_k(r_1, \ldots, r_n)$. □

We also recall a key property of small-bias probability spaces (Naor & Naor [17]; see also Alon, Goldreich, Håstad & Peralta [3], Alon, Bruck, Naor, Naor & Roth [2], and Chari, Rohatgi & Srinivasan [6]): a "d-wise ρ-biased" sample space S for n-bit vectors has the property that if $\vec{X} = (X_1, \ldots, X_n)$ is sampled uniformly at random from S, then

$$\forall I \subseteq [n], |I| \leq d, \;\; \forall b_1, b_2, \ldots b_{|I|} \in \{0,1\} \quad |Pr_{\vec{X} \in S}[\wedge_{i \in I} X_i = b_i] - 2^{-|I|}| \leq \rho. \tag{1}$$

Such spaces of cardinality $O((d \log n/\rho)^2)$, for instance, are constructed explicitly in [3]; these simplify the constructions of [17].

3 LP relaxations and randomized rounding

We quickly recapitulate the idea of randomized rounding now [21]; consider, *e.g.*, the problem of global routing in VLSI, defined in the introduction. Letting r_{ij} be the jth path in P_i and x_{ij} be the indicator variable for path r_{ij} being chosen to join s_i and t_i, we get the minimax integer program

(IP1) Minimize C, such that

$$x_{ij} \in \{0,1\}, \; \forall i, 1 \leq i \leq \ell, \;\; \sum_j x_{ij} = 1 \text{ (selecting exactly one path), and} \tag{2}$$

$$\forall e \in E, \sum_{e \in r_{ij}} x_{ij} \leq C. \tag{3}$$

The LP relaxation (LP1) of (IP1), relaxes each x_{ij} to lie in $[0,1]$. Solve (LP1), let $\{x_{ij}^*\}$ be the values of the variables in the optimal solution, and let C^* be the optimal objective function value. The key *randomized rounding* idea of [21] is, for each i independently of the others, to choose the path r_{ij} with probability x_{ij}^*. (The extension of this idea to general MIPs is obvious: independently for each i, randomly round exactly one $x_{i,j}$ to

1, guided by the "probabilities" $\{x^*_{i,j}\}$.) Now for any edge $e \in E$, Pr(More than $C^*(1 + D(C^*, 1/m))$ paths use e) $< 1/m$ from the definition of D and thus,

$$Pr(\exists e \in E: \text{ More than } C^*(1 + D(C^*, 1/m)) \text{ paths use } e) < m/m = 1; \tag{4}$$

hence, there exists a rounding method with objective function value at most $C^*(1 + D(C^*, 1/m))$. This is derandomized *sequentially* in [20] via the method of conditional probabilities.

What about the parallel setting? Suppose we are given any *positive linear program* (PLP) of input size N–an LP problem where the coefficient matrix, the r.h.s. vector, and the objective function coefficients are all non-negative, with the variables also constrained to lie in nonnegative ranges. Given any $\epsilon > 0$ with $\epsilon = \log^{-O(1)} N$, Luby & Nisan [15] show how to find in NC, a feasible solution to (PLP) (if one exists), at which the objective function value is within a relative error of $\pm\epsilon$ from optimal. Thus for the global routing problem, for instance, by trying out all values $1, 2, \ldots, \ell$ (and a finer range, if C^* is very small: note that $C^* \geq \max_{i \in [\ell]} 1/|P_i|$) for C in parallel, a fractional solution which is at most $(1 + \log^{-\Theta(1)}(m + n))$ the optimal fractional solution, can be found in NC. We need a little work here, to first transform (LP1) to a packing formulation, to apply the algorithm of [15]. Given a candidate value $a \leq \ell$ for C, consider the linear program

(LP1') Maximize $\sum_{i,j} \; x_{i,j}$ subject to:

(i) $0 \leq x_{i,j} \leq 1$ for each i, j;
(ii) $\forall i$, $\sum_j x_{i,j} \leq 1$, and
(iii) $\sum_{e \in r_{i,j}} x_{i,j} \leq a$ for each $e \in E$.

It is easily checked that this is a formulation equivalent to (LP1), if we wish to try out the case $C = a$. In all our IP applications, finding such an approximately good fractional solution is handled similarly.

But the rounding in parallel is trickier, and that is where we apply our method. To our knowledge, the best current method is the *parallel lattice approximation* algorithm of [16]. Given a matrix $A \in [0,1]^{m \times n}$ and a vector $p \in [0,1]^n$, the lattice approximation problem is to find a *lattice point* $q \in \{0,1\}^n$ such that $\| A \cdot (p - q) \|_\infty$ is "small" [20].

Letting $c_i = (Ap)_i$, [16] shows how to find $q \in \{0,1\}^n$ in NC such that for each i,

$$|A \cdot (p - q)| = O(c_i^{1/2+\epsilon}\sqrt{\log m} + \log^{1/(1-2\epsilon)} m),$$

for any fixed $\epsilon > 0$. This is not as good as in the sequential domain. In particular for our problem, we can see that randomized rounding and its sequential derandomization [20] guarantees a solution with value $C_1 = C^*(1 + D(C^*, 1/m))$. Note from Definition 2 and Fact 1 that if $C^* = o(\log m)$, then $C_1 = C^* + O(\log m / \log(\log m / C^*))$; if $C^* = a \log m$, then $C_1 \leq C^*(1 + g(a))$, where g is a positive decreasing function which goes to 0 as $a \to \infty$.

Thus, C_1 is better than the value

$$C^* + O((C^*)^{1/2+\epsilon}\sqrt{\log m} + \log^{1/(1-2\epsilon)} m)$$

guaranteed by [16]; similar remarks hold for all MIPs. We demonstrate our method by showing how to match the sequential approximation guarantee of [20] (to within a $(1 + o(1))$ factor) in parallel, if $C^* = O(\log m)$. However, lattice approximation also has an advantage over our approach as it bounds the deviation of $(Aq)_i$ from $(Ap)_i$ both from above and from below.

4 Approximating minimax integer programs

We now illustrate our method by showing how to achieve an objective function value of $C_3 = C_2(1 + o(1))$ in parallel for global routing, if $C^* = O(\log m)$ ($C_2 \doteq C^*(1 + D(C^*, 1/(2m)))$ here). Note that $C_3 = C_1(1 + o(1))$. We discuss global routing just for concreteness–our results hold for all MIPs.

For the randomized rounding, we may assume that each x_{ij}^* is a rational of the form $a/2^b$, where $b = O(\log(m+n))$; as in [16], it is easily seen that such a perturbation affects C^* little. Thus, for each i, we may partition $R \doteq \{0, 1, \ldots, 2^b - 1\}$ into subsets S_{ij}, with $|S_{ij}| = 2^b x_{ij}^*$; we imagine picking a uniformly random b-bit number y_i independently for each pair (s_i, t_i), and choose path r_{ij} for (s_i, t_i) iff $y_i \in S_{ij}$. For each edge $e \in E$, note that inequality (4) refers to a sum of *independent* random bits, one for each (s_i, t_i) path possibly using e; each such random bit Z_i (corresponding to the (s_i, t_i) path using e) will be one iff y_i lies in some fixed subset of R. Looking at our problem slightly more generally as follows, sheds more light on our other applications as well.

The $O(\log n)$th moment estimator problem. We are given n *independent* r.v.s $y_1, \ldots, y_n$, each of which takes values *uniformly* in $R = \{0, 1, \ldots, 2^b - 1\}$ where $b = O(\log(m+n))$. We are also given, for each $j \in [n]$, a finite set of binary r.v.s $\{z_{jt} : t = 1, 2, \ldots\}$ where z_{jt} is 1 iff y_j lies in some fixed subset R_{jt} of R. Also given are m random variables

$$C_i = \sum_{j=1}^{n} a_{ij} z_{j,f(i,j)}, \;\; i \in [m],$$

where $a_{ij} \in [0,1]$ and f is some arbitrary function. Now given that $E[C_i] = c_i = O(\log(m+n))$ for each i, the problem is to find a setting for the y_i's in NC, such that

$$C_i \leq d_i \doteq c_i(1 + D(c_i, 1/(2m)))(1 + o(1)) \text{ for each } i.$$

Letting $y_j = y_{j,b-1} y_{j,b-2} \cdots y_{j,0}$ for each j, we will show how to set, in stages $s = 0, 1, \ldots, b-1$, the vector

$$v_s = (y_{1,s}, y_{2,s}, \ldots, y_{n,s});$$

since $b = O(\log(m+n))$, the sequentiality in the stages is fine.

For each $i \in [m]$, let $k_i = \lceil c_i D(c_i, 1/(2m)) \rceil$; note, crucially, that

$$k_i = O(\log(m+n)), \text{ since } c_i = O(\log(m+n)). \tag{5}$$

Let B denote the "bad" event $(\exists i \in [m] : C_i \geq d_i)$. By Theorem 1(a),

$$Pr(B|Z) \leq E[X|Z] \text{ for any nonempty event } Z, \tag{6}$$

where X is defined by the *pessimistic estimator*

$$X = \sum_{i=1}^{m} \left(S_{k_i}(a_{i1} z_{1,f(i,1)}, a_{i2} z_{2,f(i,2)}, \ldots, a_{in} z_{n,f(i,n)}) / \binom{d_i}{k_i} \right). \tag{7}$$

Given any positive $\epsilon = (m+n)^{-O(1)}$, we now show how to set the vectors $v_0 := w_0, v_1 := w_1, \ldots, v_{b-1} := w_{b-1}$ in that order, so that the desired inequalities hold, up to some function of ϵ. Let $k = \max_i k_i$. Fix a k-wise $\rho = (2^{-k}\epsilon)$ biased sample space for n-bit vectors, S. Note that $|S| = poly(m,n)$, since $2^k/\epsilon$ is; this is crucially where the $O(\log(m+n))$ bound on each c_i is needed (see (5)). For $t = 0, 1, \ldots, b-1$, let $V_t \doteq (v_t, \ldots, v_{b-1})$, U_t and U_S denote the uniform distributions on $\{0,1\}^{n(b-t)}$ and S respectively, and let $BU_t \doteq U_S \times U_{t+1}$. We will pick $w_0, w_1, \ldots$ such that for $t = -1, 0, \ldots, b-1$,

$$E[X | v_0 = w_0, \ldots, v_t = w_t, \text{ and } V_{t+1} \text{ picked according to } U_{t+1}] \leq (1/2)(1+\epsilon)^{t+1}. \tag{8}$$

From (6), we see that establishing this for $t = b - 1$ will prove the algorithm if $\epsilon = o(1/\log(m+n))$, since $(1/2)(1+\epsilon)^b < 1$. Inequality (8) is established by induction on t, as in [4]; the basis $t = -1$ holds, since by definition of D and Theorem 1, $E[X|V_0$ picked according to $U_0] \leq 1/2$. Assume that (8) is true for t; we now show how to pick $w_{t+1} \in S$ such that it holds for $t+1$. Suppose we pick V_{t+1} according to BU_{t+1}. Focus on some term

$$S_{k_i}(a_{i1}z_{1,f(i,1)}, a_{i2}z_{2,f(i,2)}, \ldots, a_{in}z_{n,f(i,n)})$$

in X, which is the sum of $\binom{n}{k_i}$ sub-terms. Then, from (1), we can see that the expectation of each such sub-term is at most

$$(2^{-k_i} + \rho)/2^{-k_i} \leq (1+\epsilon)$$

times what it would have been, had V_{t+1} been picked according to U_{t+1}. Thus by the induction hypothesis, $E[X|v_0 = w_0, \ldots, v_t = w_t,$ and V_{t+1} picked according to $BU_{t+1}] \leq (1/2)(1+\epsilon)^{t+1}$ and hence, there exists $w \in S$ such that

$$E[X|v_0 = w_0, \ldots, v_t = w_t, v_{t+1} = w, \text{ and } V_{t+2} \text{ picked according to } U_{t+2}] \leq (1/2)(1+\epsilon)^{t+1}.$$

Finding w reduces to computing

$$E[X|v_0 = w_0, \ldots, v_t = w_t, v_{t+1} = w, \text{ and } V_{t+2} \text{ picked according to } U_{t+2}] \tag{9}$$

for each $w \in S$ and picking the w with the smallest conditional expectation; we can search over all $w \in S$ in parallel since $|S| = poly(m,n)$. All this is as in [4].

The key point now is that each term

$$E[S_{k_i}(a_{i1}z_{1,f(i,1)}, \ldots, a_{in}z_{n,f(i,n)})|v_0 = w_0, \ldots, v_t = w_t, v_{t+1} = w, V_{t+2} \text{ picked according to } U_{t+2}] \tag{10}$$

in (9) can be computed efficiently in NC as follows. For each $z_{i,j}$,

$$p_{ij} \doteq Pr(z_{i,j} = 1|v_0 = w_0, \ldots, v_t = w_t, v_{t+1} = w, \text{ and } V_{t+2} \text{ picked according to } U_{t+2})$$

can be computed easily in NC. Now, (10) equals

$$S_{k_i}(a_{i1}p_{1,f(i,1)}, a_{i2}p_{2,f(i,2)}, \ldots, a_{in}p_{n,f(i,n)}),$$

since $v_{t+2}, \ldots v_{b-1}$ are assumed to be picked *independently*. Thus, invoking Lemma 1, (10) and hence (9), can be computed in NC. Therefore, we get

Theorem 2 *The $O(\log n)$th moment estimator problem is in NC.*

Corollary 1 *Given any MIP, let C^* be its fractional optimum. If $C^* = O(\log m)$, then a feasible integral solution with objective function value at most $C^*(1 + D(C^*, 1/(2m)))(1 + o(1))$, can be found in NC.*

Another problem problem modeled by MIPs is a generalization of the problem of telephone network planning in bidirectional SONET rings: the "ring loading problem" (Schrijver, Seymour & Winkler [23]). Given a ring and a traffic demand between every pair of vertices, all traffic between them must be routed one way or the other around the ring. For a given set of traffic demands, the problem is to route the traffic so that the maximum traffic on any link is minimized; the generalization to arbitrary networks involves, for every pair of vertices, an allowed set of paths of which exactly one must be chosen. In this case, suppose f is the maximum traffic demand between any pair of vertices, and m denotes the number of edges in the network. We can formulate an IP for this problem as above, and scale down the inequality corresponding to each edge by a factor of f, to ensure that the coefficient matrix has entries in $[0, 1]$. Thus, our method delivers better approximation factors here than does [16], if the optimum objective function value is within an $O(\log m)$ factor of f.

5 NC approximation of a class of packing integer programs

Applications similar to those of Section 4 hold for *packing integer programs* where in fact for a subclass, we derive the first NC approximation algorithms with any "reasonable" performance guarantee; our guarantees again match those of the sequential algorithms in [20, 18], to within $(1 + o(1))$ factors. They also match the guarantee of the best-known RNC algorithm, which follows directly by combining [15] and [21]; however as pointed out before, there are sequential algorithms now with a better performance guarantee [25]. As in [20], we assume without loss of generality that the entries of c are in $[0, 1]$. We also assume, for simplicity, that $x_i \in \{0, 1\}$ for each i; our results also hold for the general case where $x_i \in \{0, 1, \ldots, d_i\}$, for some non-negative integer d_i (which is possibly infinity). Our method applies when each b_i is at most $O(\log(m+n))$, matching the sequential guarantee.

(If, for instance, the entries of A and c are in $\{0,1\}$ and if each entry of b is the same positive integer k, we get the *simple k-matching problem* on hypergraphs [13], mentioned in the introduction. See Aharoni, Erdős and Linial [1] also.)

Given such an NC algorithm, why can't we use it for arbitrary packing problems (with no required bound on the b_is), by simply scaling down constraint i by a suitable value if $b_i > c\log(m+n)$ for some desired constant c? The answer is that the tail bounds will not be good enough if we scale down: note, from Fact 1, that $D(\mu_1, x) > D(\mu_2, x)$ if $\mu_1 < \mu_2$. Thus, we assume for now that $b_i = O(\log(m+n))$ for each i, and consider the general case in Theorem 4.

The idea of [21, 20] is to solve the LP relaxation of (PIP) as usual, then *scale down* each LP optimal variable by a suitable positive value r so that after a randomized rounding is performed,

Pr(all constraints are satisified and the objective function does not decrease by "too much")

is positive. Let C^* be the value of the LP optimum. If the CH bounds say that for a sum X of independent r.v.s taking values in $[0,1]$ with $E[X] = \mu$, $Pr(X \le E[X](1-F(\mu, x))) < x$ for $0 < x < 1$, then picking $r \ge 1$ such that $r \ge \max_{i\in[m]}(1 + D(b_i/r, \epsilon/(m+1)))$ would ensure, for any $\epsilon > 0$, that

Pr(all constraints satisfied and objective function value $\ge (C^*/r)(1-F(C^*/r, \epsilon/(m+1))))$

is at least $1-\epsilon$; the reasoning is similar to that for global routing. (Such an r can easily be seen to exist, since the function $t \mapsto D(b_i/t, \epsilon/(m+1))$ decreases monotonically to zero, in the interval $t \in (1, \infty)$.)

In the parallel setting, we can find a feasible solution $\{x_i^* : i \in [n]\}$ to within $(1 - \log^{-O(1)}(m+n))$ of C^* via [15], as before. Now suppose $b_i = O(\log(m+n))$, for each i. Since the rounding produced by the approach of [16] can make the r.h.s. of the ith constraint as high as $\log^{1+\Omega(1)} m$ (recall the discussion in Section 3), it will not necessarily satisfy the constraints, even if $b_i = O(\log m)$ for just one i. Let r be such that

$$r \ge \max_{i\in[m]}(1 + D(b_i/r, 1/(m \ln m))).$$

We also scale the values $\{x_i^*\}$ down by r and can assume, as before, that each x_i^*/r is rational with denominator 2^b, where $b = O(\log(m+n))$. We show how to do the rounding in NC now, by posing our problem as follows.

The $O(\log n)$th moment estimator problem for packing. We are given n *independent* r.v.s $y_1, \ldots, y_n$, each of which takes values *uniformly* in $R = \{0, 1, \ldots 2^b - 1\}$ where $b = O(\log(m+n))$. We are also given, for each $j \in [n]$, a binary r.v. z_j, where z_j is 1 iff y_j lies in some fixed subset R_j of R. Also given are $(m+1)$ random variables

$$G_i = \sum_{j=1}^{n} a_{ij} z_j, \ i \in [m], \text{ and } H = \sum_{j=1}^{n} c_j z_j,$$

where $a_{ij}, c_j \in [0,1]$. Now given that $E[G_i] = g_i = O(\log(m+n))$ for each i and that $E[H] = h$, the problem is to find a setting for the y_i's in NC, such that

$$G_i \le h_i \doteq g_i(1 + D(g_i, 1/(m \ln m)))(1 + o(1)) \text{ for each } i, \text{ and } H \ge h - \sqrt{h}(1 - 2/\ln m)^{-0.5}.$$

We first show that such a setting exists. As before, let $k_i = \lceil g_i D(g_i, 1/(m \ln m)) \rceil$ (note, again, that $k_i = O(\log(m+n))$), and $k = \max_{i \in [m]} k_i$. By Chebyshev's inequality,

$$Pr(H \le h - \sqrt{h}(1 - 2/\ln m)^{-0.5}) \le E[(H-h)^2](1 - 2/\ln m)/h.$$

Let A and C denote

$$\sum_{i=1}^{m} S_{k_i}(a_{i1} z_1, \ldots a_{in} z_n) / \binom{h_i}{k_i} \text{ and } (H-h)^2 (1 - 2/\ln m)/h$$

respectively, and B denote the "bad event"

$$(\exists i : G_i \ge h_i, \text{ or } H \le h - \sqrt{h}(1 - 2/\ln m)^{-0.5}).$$

Then for any positive probability event Z,

$$Pr(B|Z) \le E[X|Z], \text{ where } X \doteq A + C;$$

X is our *(pessimistic) conditional estimator* now. From the definition of D, we see that $E[A] \le \sum_{i=1}^{m} (1/(m \ln m)) = 1/\ln m$; the fact $E[(H-h)^2] \le h$ implies that $E[C] \le 1 - 2/\ln m$. Hence, $Pr(B) \le E[X] \le 1/\ln m + 1 - 2/\ln m < 1$ and hence, a "good" setting for the y_i's exists.

How can we find such a good setting? Let $y_{ij}, S, \rho, v_t, U_t, U_S$, and BU_t be as for the $O(\log n)$th moment estimator problem. Then, as before, we can show, by induction on t, $t = -1, 0, \ldots, b-1$, how to pick $v_0 = w_0, v_1 = w_1 \ldots$, such that

$$E[X | v_0 = w_0, \ldots, v_t = w_t, \text{ and } V_{t+1} \text{ picked according to } U_{t+1}] \le E[X] + a(t+1)2^k \rho, \tag{11}$$

for some constant $a > 0$. We omit the details. Thus by picking $\rho = (2ab2^k \ln m)^{-1}$, say, we can ensure that $E[X|v_0 = w_0, \ldots, v_{b-1} = w_{b-1}] < 1$, implying a "good" setting for the y_i's.

Theorem 3 *The $O(\log n)$th moment estimator problem for packing is in NC. Thus, packing integer programs with 0-1 variables and with the r.h.s. constants bounded by $O(\log(m+n))$ can be approximated in NC to within a $(1+o(1))$ factor of the sequential bounds of [20, 18].*

Simple scaling allows us to handle general packing integer programs (without the constraints on the r.h.s. constants) also. For any fixed $c > 1$ and given an arbitrary PIP, suppose we want an NC approximation algorithm which produces a feasible solution that is at least $1/c$ times the sequential guarantee. Fact 1 shows us a function h such that $D((h(c) \log m)/c, 1/(m \ln m)) \leq c - 1$; thus if we have a sum X of independent r.v.s each taking values in $[0,1]$ such that $E[X] = (h(c) \log m)/c$, then $Pr(X \geq h(c) \log m) \leq 1/(m \ln m)$. Thus if we scale down each inequality with r.h.s. $b_i > h(c) \log m$ by $b_i/(h(c) \log m)$, then our above method will produce a feasible solution in NC, which is at least $1/c$ times the sequential guarantee. Thus we can handle general PIPs also, but only to within an arbitrarily small constant factor $c > 1$ of the best-known sequential algorithms, as opposed to an $(1+o(1))$ factor.

Theorem 4 *For any constant $c > 1$, packing integer programs with 0-1 variables can be approximated in NC to within an $1/c$ factor of the sequential guarantee of [20, 18].*

6 Conclusions and Open Problems

We have presented a derandomization technique which provides good parallel approximation algorithms for a family of positive linear programs; it would be interesting to find other such applications. A basic way in which this technique is useful is that it is the first method which allows superpolynomially many terms in the conditional estimator. However, a major limitation of our technique is that it works best only when all terms in the conditional estimator are positive (note that S_k involves a sum of *positive* terms, when its arguments are all positive). This is why we cannot use other useful tools for upper bounding tail probabilities such as the k–th moment inequality. In particular, our technique in its current form, cannot be used for approximating covering integer programs, which seek to minimize $c^T \cdot x$ subject to $Ax \geq b$, where $A \in \Re_+^{m \times n}$, $b \in \Re_+^m$, and $c \in \Re_+^n$, with the entries x_i of x being constrained to be non-negative integers in some range. It is an interesting open problem to come up with good techniques to approximate covering integer programs well in parallel.

Another interesting direction is to come up with NC algorithms that match the improved bounds for minimax, packing and covering integer programs due to [25, 24]. In fact, not even RNC algorithms are known for these improved bounds; furthermore, the result of [24] on minimax integer programs is non-constructive, and does not imply even a sequential (randomized) polynomial-time algorithm, as it stands. These suggest possible first steps to take before looking for NC algorithms for these problems.

A question of broader interest is the utility of de-randomization techniques in practical settings. As in this work, these techniques usually convert a fairly efficient randomized algorithm (which, however, assumes a source of "perfect randomness") into a less efficient deterministic procedure. Thus in a given setting, the cost of this loss of efficiency has to be weighed against the benefit of absolute certainty. The work of [8, 10, 11] mentioned in the introduction, suggests that de-randomization techniques might prove their worth in critical applications.

Acknowledgements. We thank Prabhakar Raghavan for clarifying an issue about randomized rounding, and David Zuckerman for pointing out the work of [8]. We also thank the referee for his/her helpful comments.

References

[1] R. Aharoni, P. Erdős, and N. Linial. Optima of dual integer linear programs. *Combinatorica*, 8:13–20, 1988.

[2] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth. Construction of asymptotically good, low-rate error-correcting codes through pseudo-random graphs. *IEEE Trans. Info. Theory*, 38:509–516, 1992.

[3] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k–wise independent random variables. *Random Structures and Algorithms*, 3(3):289–303, 1992.

[4] N. Alon and M. Naor. Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. Manuscript, 1992. To appear in *Algorithmica*.

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

[6] S. Chari, P. Rohatgi, and A. Srinivasan. Improved algorithms via approximations of probability distributions. In *Proc. ACM Symposium on Theory of Computing*, pages 584–592, 1994.

[7] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[8] A. M. Ferrenberg, D. P. Landau, and Y. J. Wong. Monte Carlo simulations: Hidden errors from "good" random number generators. *Physical Review Letters*, 69(23):3382–3384, 1992.

[9] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.

[10] T.-s. Hsu. *Graph augmentation and related problems: theory and practice.* PhD thesis, Department of Computer Sciences, University of Texas at Austin, October 1993.

[11] T.-s. Hsu, V. Ramachandran, and N. Dean. Parallel implementation of algorithms for finding connected components. In *DIMACS International Algorithm Implementation Challenge*, pages 1–14, 1994.

[12] H. J. Karloff and P. Raghavan. Randomized algorithms and pseudorandom numbers. *Journal of the ACM*, 40(3):454–476, 1993.

[13] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[14] M. Luby. Removing randomness in parallel computation without a processor penalty. *Journal of Computer and System Sciences*, 47(2):250–286, 1993.

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

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

[17] J. Naor and M. Naor. Small–bias probability spaces: efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[18] S. A. Plotkin, D. B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 495–504, 1991.

[19] P. Raghavan. *Randomized Rounding and Discrete Ham–Sandwich Theorems: Provably Good Algorithms for Routing and Packing Problems.* PhD thesis, University of California at Berkeley, July 1986. Also available as Computer Science Department Report UCB/CSD 87/312.

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

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

[22] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8:223–250, 1995.

[23] A. Schrijver, P. Seymour, and P. Winkler. The ring loading problem. In preparation, 1994.

[24] A. Srinivasan. An extension of the Lovász Local Lemma, and its applications to integer programming. *Proc. ACM/SIAM Symposium on Discrete Algorithms*, 1996.

[25] A. Srinivasan. Improved approximations of packing and covering problems. In *Proc. ACM Symposium on Theory of Computing*, pages 268–276, 1995.