Solving Some Discrepancy Problems in NC *

Sanjeev Mahajan†
LSI Logic
Milpitas, CA 95035, U.S.A.
msanjeev@lsil.com

Edgar A. Ramos‡
Max-Planck-Institut für Informatik
Im Stadtwald, 66123 Saarbrücken, Germany
ramos@mpi-sb.mpg.de

K. V. Subrahmanyam§
SPIC Mathematical Institute
92.GN Chetty Road, T. Nagar Madras, India. 600 017
kv@smi.ernet.in

*A preliminary version of this work appeared in: Proceedings of the 17th Conference on Foundations of Software Technology and Theoretical Computer Science, Kharagpur, India. Lecture Notes on Computer Science, Vol. 1346, pages 22–36. Springer 1997.

†Work performed while at the Max-Planck-Institut für Informatik, Germany.

‡The work by this author was started while at DIMACS/Rutgers University, New Brunswick, NJ, USA, supported by DIMACS postdoctoral fellowship.

§Work performed while visiting the Max-Planck-Institut für Informatik, Germany.

Abstract

We show that several discrepancy-like problems can be solved in NC nearly achieving the *discrepancies* guaranteed by a probabilistic analysis and achievable sequentially. For example, we describe an NC algorithm that given a set system $(X, \mathcal{S})$, where X is a ground set and $\mathcal{S} \subseteq 2^X$, computes a set $R \subseteq X$ so that for each $S \in \mathcal{S}$ the *discrepancy* $||R \cap S| - |\overline{R} \cap S||$ is $O(\sqrt{|S| \log |\mathcal{S}|})$. Whereas previous NC algorithms could only achieve discrepancies $O(\sqrt{|S|^{1+\epsilon} \log |\mathcal{S}|})$ with $\epsilon > 0$, ours matches the probabilistic bound within a multiplicative factor $1+o(1)$. Other problems whose NC solution we improve are lattice approximation, ϵ-approximations of range spaces with constant VC-exponent, sampling in geometric configuration spaces, approximation of integer linear programs, and edge coloring of graphs.

Key Words: Discrepancy, lattice approximation, parallel algorithms, derandomization, geometric sampling.

1 Introduction

Problem and Previous Work. Discrepancy is an important concept in combinatorics, see e.g. [2, 6], and theoretical computer science, see e.g. [31, 27, 11]. It attempts to capture the idea of a *good sample* from a set. The simplest example, the *set discrepancy problem*, considers a set system $(X, \mathcal{S})$ where X is a ground set and $\mathcal{S} \subseteq 2^X$ is a family of subsets of X, and asks for a subset $R \subseteq X$ such that for each $S \in \mathcal{S}$ the difference $||R \cap S| - |\overline{R} \cap S||$, called the *discrepancy*, is small. Using Chernoff-Hoeffding bounds [12, 18, 32, 31], it is found that a random sample $R \subseteq X$, with each $x \in X$ taken into R independently with probability $1/2$, is with nonzero probability a low discrepancy set: for each $S \in \mathcal{S}$, $||R \cap S| - |\overline{R} \cap S|| = O(\sqrt{|S| \log |\mathcal{S}|})$. In [31] using the method of *conditional probabilities* this was derandomized to obtain a deterministic sequential algorithm computing such a sample R. In parallel, several approaches have been used (k-wise independence combined with the method of conditional probabilities and relaxed to biased spaces [8, 27, 29, 9]). However, so far these efforts to compute a sample in parallel have resulted only in discrepancies $O(\sqrt{|S|^{1+\epsilon} \log |\mathcal{S}|})$.

Results. In this paper, we describe NC algorithms (specifically, the algorithms run in $O(\log^2 n)$ time using $O(n^C)$ processors for some constant C in the EREW PRAM model[1]) that achieve the probabilistic bounds (achievable sequentially) within a multiplicative factor $1 + o(1)$. The technique we use is to model random sampling by *randomized finite automatons*[2] (RFA's) and then *fool* these automata with a probability distribution of polynomial size support. The approach is not new; in fact, Karger and Koller [20] show how to fool such automata via the lattice approximation problem, using a solution for that problem developed in [27]. However, they apparently did not realize that the lattice approximation problem can itself be modelled by RFAs and, as a result this and other discrepancy-like problems can be solved in parallel, nearly achieving the probabilistic bounds. We also describe how the work of Nisan [30] of fooling RFAs via pseudo-random generators also fits the same general approach.

We consider a sample R from X with each $x_j \in X$ selected into R independently with probability p_j. The *goodness* of the sample is determined by a polynomial number (in $|X|$) of random variables $c_i = \sum_{x_j \in X} a_{ij} q_j$ with coefficients a_{ij} in $[0, 1]$ and $q_j = 1$ iff $x_j \in R$ (the indicators for R). More precisely, R is good if for each i, $|c_i - \mu_i| \leq \lambda_i$, where $\mu_i = \sum_{x_j \in X} a_{ij} p_j$ is the expected value of c_i, and λ_i is a deviation guaranteed by probabilistic (Chernoff-Hoeffding) bounds. Each coefficient a_{ij} is restricted to have $O(\log |X|)$ bits so that the number of possible values of c_i is polynomial in $|X|$. This observation allows us to regard each random variable c_i as being computed by a RFA of polynomial size (there is one RFA for each i). A key point, which perhaps explains why our observations had not been noticed before, is that it is sufficient to fool the *individual* transition probabilities of the RFAs *simultaneously*, rather than the *joint* transition probabilities, since the probability of obtaining a bad sample is bounded by the sum of the probabilities that each individual constraint does not hold. Although limited, this framework includes the lattice approximation problem, the discrepancy problem, and sampling problems in computational geometry. Also, since the lattice approximation problem can be used to obtain approximate solutions to integer linear programs [32, 31], this leads to improved results for this problem in the parallel context. As a result, we automatically improve on the recent work in [3].

[1] In the CRCW PRAM model, the time can be reduced to $O(\log n)$ using approximate counting: It is possible to count with relative error $1/\log^c n$ in $O(1)$ time using a polynomial number of processors [1, 14].

[2] Finite automata in which transitions from a state to its immediate successor occurs with a certain probability.

Our improvement also translates to the derandomization in [27] of an algorithm for graph edge coloring by Karloff and Shmoys [22].[3]

Contents of the paper. We first state the Chernoff-Hoeffding bounds used in this paper. In Sect. 2, we state and model the lattice approximation problem by RFAs; in Sect. 3, we present the techniques for fooling RFAs and the resulting algorithm for the lattice approximation problem; in Sect. 4, we consider the discrepancy problem and its application to solving the lattice approximation problem; in Sect. 5, we present two applications to computational geometry; finally, in Sect. 6, we briefly mention the applications to approximating integer linear programs and to edge coloring of graphs. Finally, in the Appendix, we include some computations omitted in the main body of the paper.

Chernoff-Hoeffding Bounds. For independent random variables $X_1, \ldots, X_n$ in $[0,1]$, $X = \sum_{i=1}^n X_i$ and $\mu = \mathbf{E}[X]$, let $\lambda(\mu, x)$ denote the absolute deviation for which, $\Pr\{|X-\mu| > \lambda(\mu,x)\} < x$. A bound for $\lambda(\mu, x)$ is obtained using the Chernoff-Hoeffding bounds [12, 18, 31, 2]:

$$\lambda(\mu,x) = \begin{cases} \Theta(\sqrt{\mu \log(1/x)}) & \text{if } \mu \geq c\log(1/x) \\ \Theta\left(\frac{\log(1/x)}{\log(\log(1/x)/\mu)}\right) & \text{otherwise,} \end{cases} \tag{1}$$

where c is a constant. We define, likewise, $\lambda_k(\mu, x)$, when X is the sum of k-wise independent random variables $X_1, \ldots, X_n$ with values in $[0,1]$. We have the following bounds from [7, 33]:

$$\lambda_k(\mu,x) = \begin{cases} \Theta(\sqrt{k\mu}(1/x)^{1/k}) & \text{if } \mu \geq k \\ \Theta(k(1/x)^{1/k}) & \text{otherwise.} \end{cases} \tag{2}$$

In this paper the case k=2 will be frequently used. In this case the Chebychev's inequality gives:

$$\lambda_2(\mu,x) = \Theta\left(\sqrt{\frac{\mu}{x}}\right) \tag{3}$$

2 Lattice Approximation

In the *lattice approximation problem* we are given an $m \times n$ matrix A with $a_{ij} \in [0,1]$, an $n \times 1$ vector p with $p_j \in [0,1]$, and we are to compute an $n \times 1$ vector q with $q_j \in \{0,1\}$, *a lattice vector*, that achieves small *discrepancies* $\Delta_i = \left|\sum_{j=1}^n a_{ij}(p_j - q_j)\right|$.

2.1 Randomized Rounding

Raghavan's [31] solution to the lattice approximation problem is to set each q_j to 1 with probability p_j, independently of all others, a procedure called *randomized rounding*. Let $\mu_i = \sum_{j=1}^n a_{ij}p_j$. The Chernoff-Hoeffding bounds guarantee that, for each i, $\Delta_i > \lambda(\mu_i, 1/m)$ holds with probability less than $1/m$; therefore, with nonzero probability, for all i, $\Delta_i \leq \lambda(\mu_i, 1/m)$ (m is the number of equations). For $\mu_i = \Omega(\log m)$ (which will be the case most of the time), this is $\Theta(\sqrt{\mu_i \log m})$.

Raghavan [31] converted this probabilistic existence argument into a deterministic algorithm through the so called *method of conditional probabilities.* (achieving the discrepancies guaranteed by the Chernoff-Hoeffding bounds). A parallel version by Motwani *et al* [27] used polynomial size

[3] We thank an anonymous referee for pointing out this application.

spaces with limited independence, together with a bit-by-bit rounding approach. Unfortunately, under the requirement that the algorithm be in NC, the best discrepancies obtained are $\Delta_i = O(\sqrt{\mu_i^{1+\epsilon}\log m})$.

Using the Chernoff-Hoeffding bounds for arbitrary p_j's and the construction of k-wise independent probability spaces in [19], it is possible to avoid the bit-by-bit rounding and obtain a faster and simpler algorithm (checking all the points in the probability space in a straightforward manner), though with worse bounds for the discrepancies obtained and the number of processors used. This algorithm, with $k = 2$, turns out to be useful as a part of our main algorithm. To simplify later expressions, we assume that m is polynomial in n, so that $\log(n+m) = O(\log n)$ (the resulting work bound is polynomial in n only if such is the case).

Lemma 2.1 *A lattice vector with discrepancies $\Delta_i = O(\sqrt{\mu_i}m^{1/k})$ can be computed in $O(\log(m+n)) = O(\log n)$ time using $O(mn^{k+1})$ processors in the EREW PRAM model.*

The lemma is verified as follows. First, the Chernoff-Hoeffding bound for k-wise independence guarantees the existence of a lattice vector q such that $\Delta_i = O(\sqrt{\mu_i}m^{1/k})$ (assuming that $k \leq \mu$ is a constant). Second, use the construction in [19] of a k-wise independent probability space D of size $O(n^k)$.[4] All points in D can be checked in parallel using time $\log(m+n)$ and total work $O(mn^{k+1})$.

2.2 Modelling Rounding with Levelled RFAs

Limiting the Precision. In order to derandomize the rounding procedure while getting closer to the probabilistic bound, it is useful to model it with RFAs. Specifically, the idea is to have one RFA for each of the m equations so that in the i-th RFA, states correspond to the different partial sums $\sum_{j=1}^{l} a_{ij}q_j$, $l = 0,\ldots,n$ and $q_j \in \{0,1\}$. For this to be useful, the number of states must be polynomial. Fortunately, as observed in [27], the fractional part of the coefficients a_{ij} (and so the partial sums) can be truncated without a significant increment in the discrepancies. Also, it will be useful later to limit the precision of the probabilities p_j. More precisely, these parameters can be truncated to $L' = \lceil\log(3n/\hat{\epsilon})\rceil$ fractional bits while increasing the discrepancy by at most $\hat{\epsilon}$: Letting $\tilde{a}_{ij}$ and $\tilde{p}_j$ be the corresponding truncated numbers, the discrepancy $|\sum_j(a_{ij}q_j - a_{ij}p_j)|$ with respect to the original parameters can be upper bounded by

$$\left|\sum_j(a_{ij}q_j - \tilde{a}_{ij}q_j)\right| + \left|\sum_j(\tilde{a}_{ij}q_j - \tilde{a}_{ij}\tilde{p}_j)\right| + \left|\sum_j(\tilde{a}_{ij}\tilde{p}_j - \tilde{a}_{ij}p_j)\right| + \left|\sum_j(\tilde{a}_{ij}p_j - a_{ij}p_j)\right|$$
$$\leq \sum_j|a_{ij} - \tilde{a}_{ij}| + \tilde{\Delta}_i + \sum_j|\tilde{p}_j - p_j| + \sum_j|\tilde{a}_{ij} - a_{ij}| \leq \tilde{\Delta}_i + \hat{\epsilon},$$

where $\tilde{\Delta}_i$ is the discrepancy achieved for the truncated parameters. Furthermore, for the integer part of the partial sums, $L'' = \lceil\log n\rceil$ bits suffice. If $1/\hat{\epsilon}$ is polynomially bounded, then so is the number of states needed in the RFAs. W.l.o.g. we assume that $\hat{\epsilon} = O(1)$ is sufficient and so $L = L' + L'' = 2\log n + O(1)$ bits are sufficient to represent the different possible sums.

[4] To be able to use such construction, we need the assumption that the p_j can be limited to $\log n$ bits. This is a reasonable assumption as discussed in the next subsection. Otherwise, we could use the construction in [21], which has size $O(n^{2k})$.

Levelled RFAs. Thus, the rounding procedure can be modelled with m *levelled* RFAs. The i-th RFA, M_i, consists of $n+1$ *levels* of states $N_{i,0}, \ldots, N_{i,n}$, so that in $N_{i,j}$ there is a state $\langle i,j,r\rangle$ for each number r with L bits (L'' of them are integer bits, the rest are fractional). The transitions in M_i are between consecutive levels $N_{i,j-1}$ and $N_{i,j}$ in the natural way: $\langle i,j-1,r\rangle$ is connected to $\langle i,j,r\rangle$ under $q_j = 0$, and $\langle i,j-1,r\rangle$ is connected to $\langle i,j,r+a_{ij}\rangle$ under $q_j = 1$. The only state $s_i = \langle i,0,0\rangle$ in $N_{i,0}$ is the start state of M_i. A state $\langle i,n,r\rangle$ in the last level $N_{i,n}$ is *accepting* if r is within a specified deviation λ_i from μ_i, that is, if $|r-\mu_i| \le \lambda_i$. Let R_i denote the set of *rejecting* states in $N_{i,n}$. For two states s and t in some M_i and a string w, $s \overset{w}{\to} t$ denotes that starting at s the string w leads to t, and $[s \overset{w}{\to} t]$ is an indicator equal to 1 if $s \overset{w}{\to} t$ holds and equal to 0 otherwise. Let D be a probability distribution on Σ_l, the set of all 0/1 strings of length l. For $w \in \Sigma_l$, $\mathrm{Pr}_D\{w\}$ denotes the probability of w in D, and $\mathrm{Pr}_D\{st\}$ denotes the probability of $s \overset{w}{\to} t$ when w is chosen at random according to D. Then

$$\mathrm{Pr}_D\{st\} = \sum_{w\in\Sigma_l} [s \overset{w}{\to} t] \cdot \mathrm{Pr}_D\{w\}.$$

Basic Approach. Let F_n be the fully independent distribution on Σ_n according to the specified bit probabilities p_j. Suppose that we can construct in polynomial time a distribution D_n on Σ_n with polynomial size support such that for each i,

$$\sum_{r\in R_i} |\mathrm{Pr}_{D_n}\{s_i r\} - \mathrm{Pr}_{F_n}\{s_i r\}| \le \epsilon.$$

Then $\sum_{r\in R_i} \mathrm{Pr}_{D_n}\{s_i r\} \le \epsilon + \sum_{r\in R_i} \mathrm{Pr}_{F_n}\{s_i r\}$, and if we set $\lambda_i = \lambda(\mu_i, 1/2m)$ the right hand side of this equation is at most $\epsilon + \frac{1}{2m}$. Thus, summing over all i, $\sum_i \sum_{r\in R_i} \mathrm{Pr}_{D_n}\{s_i r\} < m\epsilon + 1/2$. For $\epsilon = \frac{1}{2m}$, this is at most 1. That is, there is at least one event in D_n that gives a lattice vector solution almost as good as that guaranteed by the probabilistic bound under F_n. As a result, we obtain discrepancies within a multiplicative factor $1+o(1)$: $\lambda(\mu_i, 1/2m)$ rather than $\lambda(\mu_i, 1/m)$. (We could get even closer to the probabilistic bound by further reducing the error in the approximation at the expense of a greater amount of work.) Thus, derandomizing the rounding procedure becomes a problem of *fooling* a set of levelled RFAs, which is discussed in the next section.

3 Fooling Levelled RFAs in Parallel

Techniques to fool RFAs are found in the work of Nisan [30] in the context of derandomizing space bounded machines, and in the work of Karger and Koller [20] in the context of parallel derandomization. Karger and Koller's approach is stronger in that it achieves relative error in the transition probabilities, while Nisan's approach achieves absolute error. On the other hand, Nisan's approach has the advantage of a compact representation, but that is not important for our purposes. So far it has gone unnoticed that these techniques are precisely what is needed to nearly achieve the probabilistic bounds for the lattice approximation problem in parallel. We present these two approaches in a unified manner for the particular case of levelled RFAs, which results in somewhat better processor bounds than if general RFAs are considered. (The processor bounds however are still quite large).

3.1 General Approach

The goal is to construct a distribution D_n on Σ_n that fools each RFA M_i. We emphasize that we can fool simultaneously the individual transition probabilities of all the RFAs, $\Pr_{F_n}\{s_i \to r_i\}$ for all i, but cannot fool the joint transition probabilities $\Pr_{F_n}\{s_1 \to r_1, \ldots, s_m \to r_m\}$. Let E_0 be an integer parameter which will correspond to the (approximate) size of D_n, and let $W = \lceil \log E_0 \rceil$.

Algorithm. As in [30, 20], D_n is determined by a divide and conquer approach in which the generic procedure $\mathtt{fool}(l, l')$ constructs a distribution that fools the transition probabilities between levels l and l' in *all* the RFAs. $\mathtt{fool}(l, l')$ works as follows: It computes, using $\mathtt{fool}(l, l'')$ and $\mathtt{fool}(l'', l')$ recursively, distributions D_1 and D_2, each of size at most $E_0(1 + o(1))$, that fool the transitions between states in levels l and $l'' = \lfloor (l + l')/2 \rfloor$, and between states in levels l'' and l; $\mathtt{reduce}(D_1 \times D_2)$ then combines D_1 and D_2 into a distribution D of size at most $E_0(1 + o(1))$ that fools the transitions between states in levels l and l' in all the RFAs. In the bottom of the recursion we use a 0/1 distribution F_1 with support of size E_0 implemented by W unbiased bits,[5] which preserves the transition probabilities exactly.

$\mathtt{fool}(l, l')$
1. if $l = l'$ then return F_1
2. $l'' = \lfloor (l + l')/2 \rfloor$
3. $D_1 = \mathtt{fool}(l, l'')$
4. $D_2 = \mathtt{fool}(l'', l')$
5. return $\mathtt{reduce}(D_1 \times D_2)$

Reduce. Let $\tilde{D} = D_1 \times D_2$ be the product distribution with support $\mathrm{supp}(\tilde{D}) = \{w_1 w_2 : w_i \in \mathrm{supp}(D_i)\}$ and $\Pr_{\tilde{D}}\{w_1 w_2\} = \Pr_{D_1}\{w_1\}\Pr_{D_2}\{w_2\}$. A randomized version of the combining is, as in [20]: Retain each $w \in \tilde{D}$ with certain probability $q(w)$ into $\mathrm{supp}(D)$ with $\Pr_D\{w\} = \Pr_{\tilde{D}}\{w\}/q(w)$. Thus, for all states, s, t, the transition probabilities are preserved in expectation:

$$\mathbf{E}[\Pr_D\{st\}] = \sum_w [s \xrightarrow{w} t] \frac{\Pr_{\tilde{D}}\{w\}}{q(w)} q(w) = \Pr_{\tilde{D}}\{st\}. \tag{4}$$

This selection also implies that the expected size of $\mathrm{supp}(D)$ is $\sum_w q(w)$. We will bound this by our desired value $E_0(1 + o(1))$ and formulate these conditions as a randomized rounding problem. This is exactly the approach of Karger and Koller [20]; but they missed the fact that the lattice approximation problem itself can be modelled by RFAs and as a result the probabilistic bound can be nearly achieved.

Next, we describe and analyze deterministic procedures to obtain a distribution D of size at most $E_0(1+o(1))$ such that for all states s, t the difference $|\Pr_D\{st\} - \Pr_{\tilde{D}}\{st\}|$ is small. We distinguish two cases according to whether we aim for absolute or relative error in the approximation. These

[5] That is, we assume that W unbiased bits are sufficient to implement each probability p_j. A valid assumption since E_0 will be polynomial in n and m. On the other hand, fewer unbiased bits could be sufficient to implement p_j, for example when $p_j = 1/2$. In this case, it is possible to save a polylog factor in the final processor bound of the algorithm by grouping together several bits for which a distribution with support of size E_0 can be constructed. Since the gain is small, we neglect this possibility. Also, for simplicity, we assume that necessary operations on W bits can be performed in constant time.

cases correspond to the work Nisan [30] and of Karger and Koller [20] respectively.[6] Our aim is a unified and self contained presentation adapted to our situation, emphasising how new instances of the lattice approximation problem appear naturally in solving the original instance. We describe the deterministic procedure at the j-th level of the recursion, but to keep notation simple we use the notation D_1, D_2, $\tilde{D}$ and D introduced above, rather than using an additional index indicating the level.

An important observation is that if the transitions from $s_{i,l} = \langle i, l, 0\rangle$ are fooled then the transitions from the other states $\langle i, l, r\rangle$, $r \neq 0$, in $N_{i,l}$ are automatically fooled as well. This is because a string w induces a transition from $\langle i, l, 0\rangle$ to $\langle i, l', \delta\rangle$ iff it induces a transition from $\langle i, l, r\rangle$ to $\langle i, l', r+\delta\rangle$. Therefore, in the following arguments, and in the algorithm, we only need to make sure that transitions from $s_{i,l}$ are fooled. This leads to some savings in the number of processors needed.

3.2 Absolute Error

D should fool the RFAs in the sense that, for each $s = s_{i,l} = \langle i, l, 0\rangle$,

$$\sum_{t \in N_{i,l'}} |\Pr_D\{st\} - \Pr_{F_h}\{st\}| \leq \epsilon_j,$$

where $h = l' - l$ and ϵ_j is an upper bound on the absolute error accumulated up to the j-th recursion level.

Accumulation of Error. Let us assume that D, obtained from $\tilde{D}$, satisfies

$$\sum_{t \in N_{i,l'}} |\Pr_{\tilde{D}}\{st\} - \Pr_D\{st\}| \leq \tilde{\epsilon} \tag{5}$$

for each $s = s_{i,l}$. Since (see App. A)

$$\sum_{t \in N_{i,l'}} |\Pr_{\tilde{D}}\{st\} - \Pr_{F_h}\{st\}| \leq 2\epsilon_{j-1},$$

then $\epsilon_j \leq 2\epsilon_{j-1} + \tilde{\epsilon}$, and so $\epsilon_j \leq (2^j - 1)\tilde{\epsilon}$. Let $d = \lceil \log n \rceil$ be the last level. In order to achieve final error $\epsilon_d \leq \epsilon$, we choose $\tilde{\epsilon} = \epsilon/n$.

Computing D from $\tilde{D}$. At each stage of the algorithm the partial distribution D constructed will be uniform on its support. If $|\mathrm{supp}(\tilde{D})|$ is less than E_0, then $D = \tilde{D}$, nothing needs to be done. Otherwise, D is obtained from $\tilde{D}$ as follows. We have the following equations for every pair of states $s = s_{i,l}$ and $t \in N_{i,l'}$:

$$\sum_{w \in \mathrm{supp}(\tilde{D})} [s \xrightarrow{w} t] \Pr_{\tilde{D}}\{w\} = \Pr_{\tilde{D}}\{st\}; \tag{6}$$

and there is also the normalization condition:

$$\sum_{w \in \mathrm{supp}(\tilde{D})} \Pr_{\tilde{D}}\{w\} = 1 \tag{7}$$

[6] For most of our applications, absolute error suffices. See Sect. 5 for an application in which relative error seems to be needed.

Multiplying each of these equations by E_0 and setting $q(w) = E_0 \Pr_{\tilde{D}}\{w\} = E_0/|\mathrm{supp}(\tilde{D})|$, we obtain the following equations:

$$\begin{aligned}
\sum_{w \in \mathrm{supp}(\tilde{D})} [s \xrightarrow{w} t] q(w) &= \Pr_{\tilde{D}}\{st\} E_0 \qquad \text{for each } M_i,\ s = s_{i,l} \text{ and } t \in N_{i,l'} && (8)\\
\sum_{w \in \mathrm{supp}(\tilde{D})} q(w) &= E_0. && (9)
\end{aligned}$$

The key idea is to regard the above equations as defining a lattice approximation problem. A lattice vector approximately satisfying these equations, gives us the desired probability space D: The support of the space D will be precisely the support of this lattice vector, and the elements in the support will be assigned probability $1/|\mathrm{supp}(D)|$.[7] As already indicated, a solution to this lattice approximation problem is to retain each element $w \in \mathrm{supp}(\tilde{D})$ in D with probability $q(w) = E_0/|\mathrm{supp}(\tilde{D})|$. But rather than choosing the w's independently, we choose them using a 2-wise independent probability space.

Let $\eta = 2^L$ be the number of states in a level, and $N = m\eta$ be the number of pairs i, t. So the lattice approximation problem in Eqns. (8-9) has $N + 1$ equations. Using the Chernoff-Hoeffding bounds, there exists a lattice vector (whose support is identified with $\mathrm{supp}(D)$ in the sequel) such that for all states s, t the following holds with non zero probability:

$$\begin{aligned}
\left| \sum_{w \in \mathrm{supp}(D)} [s \xrightarrow{w} t] - \Pr_{\tilde{D}}\{st\} E_0 \right| &\leq \lambda_2(\Pr_{\tilde{D}}\{st\} E_0, \Pr_{\tilde{D}}\{st\}/(m+1))\\
\left| \sum_{w \in \mathrm{supp}(D)} 1 - E_0 \right| &\leq \lambda_2(E_0, 1/(m+1)),
\end{aligned}$$

The equations hold with non zero probability since the error probability as given by the union bound is less than $\sum_{i=1}^{m} \sum_{t \in N_{i,l'}} \Pr_{\tilde{D}}\{st\}/(m+1) + 1/(m+1) < 1$. Dividing the above equations by E_0 we obtain,

$$\begin{aligned}
\left| \sum_{w \in \mathrm{supp}(D)} \frac{[s \xrightarrow{w} t]}{|\mathrm{supp}(D)|} \frac{|\mathrm{supp}(D)|}{E_0} - \Pr_{\tilde{D}}\{st\} \right| &\leq \frac{\lambda_2(\Pr_{\tilde{D}}\{st\} E_0, \Pr_{\tilde{D}}\{st\}/(m+1))}{E_0}\\
\left| \sum_{w \in \mathrm{supp}(D)} \frac{1}{E_0} - 1 \right| &\leq \frac{\lambda_2(E_0, 1/(m+1))}{E_0} \quad ,
\end{aligned}$$

Then, substituting γ for $|\mathrm{supp}(D)|/E_0$,

$$\begin{aligned}
|\Pr_D\{st\}\gamma - \Pr_{\tilde{D}}\{st\}| &\leq \frac{\lambda_2(\Pr_{\tilde{D}}\{st\} E_0, \Pr_{\tilde{D}}\{st\}/(m+1))}{E_0}\\
|\gamma - 1| &\leq \frac{\lambda_2(E_0, 1/(m+1))}{E_0}.
\end{aligned}$$

[7] To satisfy Eqn. (4) exactly we need to assign each element retained in D a probability $\Pr_{\tilde{D}}\{w\}/q(w)$. However D may not satisfy the requirements of being a probability distribution under such an assignment and so we normalize the probability of every element to $1/|\mathrm{supp}(D)|$. Still, D defined in this way approximates $\tilde{D}$ well.

So, for all $s = s_{i,l}$ and $t \in N_{i,l'}$, the following holds with nonzero probability

$$
\begin{aligned}
|\mathrm{Pr}_{\tilde{D}}\{st\} - \mathrm{Pr}_D\{st\}| &\leq |\mathrm{Pr}_{\tilde{D}}\{st\} - \mathrm{Pr}_D\{st\}\gamma| + \mathrm{Pr}_D\{st\}|\gamma - 1| \\
&\leq \frac{\lambda_2(\mathrm{Pr}_{\tilde{D}}\{st\}E_0, \mathrm{Pr}_{\tilde{D}}\{st\}/(m+1))}{E_0} + \frac{\lambda_2(E_0, 1/(m+1))}{E_0}.
\end{aligned}
$$

In order to achieve the error bound between $\tilde{D}$ and D expressed by Eqn. (5), it is sufficient that $|\mathrm{Pr}_{\tilde{D}}\{st\} - \mathrm{Pr}_D\{st\}| \leq \tilde{\epsilon}/\eta = \epsilon/n\eta$.

Choice of E_0. Since the w's are selected with probability $q(w)$ from a 2-wise independent probability distribution, using the estimate for $\lambda_2(\mu, x)$ in equation (3), we obtain that $|\mathrm{Pr}_{\tilde{D}}\{st\} - \mathrm{Pr}_D\{st\}| \leq Cm^{1/2}/\sqrt{E_0}$. So we need that $Cm^{1/2}/\sqrt{E_0} \leq \epsilon/n\eta$. We then choose E_0 so that

$$
E_0 \geq C\frac{n^2\eta^2 m}{\epsilon^2}. \tag{10}
$$

3.3 Relative Error

In this case, D should fool the RFAs in the sense that for each $s = s_{i,l}$ and $t \in N_{i,l'}$,

$$
\left|\frac{\mathrm{Pr}_D\{st\}}{\mathrm{Pr}_{F_h}\{st\}} - 1\right| \leq \delta_j,
$$

where δ_j is the relative error accumulated up to the j-th recursion level. To achieve this, the distribution D is allowed to be non uniform on its support (a probability distribution uniform on a support of polynomial size cannot have events with very small probability). The probabilities $q(w)$ with which elements in $\tilde{D}$ are retained into D are also non uniform. As in the absolute error case, we set up a lattice approximation problem. The support of D will be precisely the support of a solution to it. Instead of assigning each element in the support of D a probability $\mathrm{Pr}_{\tilde{D}}\{w\}/q(w)$ as required to satisfy Eqn. (4), we normalize it by $\gamma = \sum_{w \in \mathrm{supp}(D)} \mathrm{Pr}_{\tilde{D}}\{w\}/q(w)$; that is we set $\mathrm{Pr}_D\{w\} = \mathrm{Pr}_{\tilde{D}}\{w\}/q(w)\gamma$.

Accumulation of Error. Let us assume that D, obtained from $\tilde{D}$, satisfies

$$
\left|\frac{\mathrm{Pr}_D\{st\}}{\mathrm{Pr}_{\tilde{D}}\{st\}} - 1\right| \leq \tilde{\delta}, \tag{11}
$$

for each $s = s_{i,l}$ and $t \in N_{i,l'}$. Since (see Appendix A)

$$
\left|\frac{\mathrm{Pr}_D\{st\}}{\mathrm{Pr}_{F_h}\{st\}} - 1\right| + 1 \leq (1 + \delta_{j-1})^2(1 + \tilde{\delta}),
$$

we have that $\delta_j \leq (1+\delta_{j-1})^2(1+\tilde{\delta}) - 1$, and so $\delta_d \leq (1+\tilde{\delta})^n - 1 \leq 2n\tilde{\delta}$ for $\tilde{\delta} \leq 1/2n$. Accordingly, we choose $\tilde{\delta} = \delta/2n$ to achieve total relative error δ.

Choice of $q(w)$. If $|\text{supp}(\tilde{D})|$ is less than E_0 then $D = \tilde{D}$, nothing needs to be done. Otherwise, we proceed as follows. Let $\Delta = E_0/(N+1)$. We rewrite Eqns. (6) and (7) as

$$\sum_{w \in \text{supp}(\tilde{D})} \frac{[s \xrightarrow{w} t]\Pr_{\tilde{D}}\{w\}\Delta}{q(w)\Pr_{\tilde{D}}\{st\}} q(w) \quad = \quad \Delta \tag{12}$$

$$\sum_{w \in \text{supp}(\tilde{D})} \frac{\Pr_{\tilde{D}}\{w\}\Delta}{q(w)} q(w) \quad = \quad \Delta. \tag{13}$$

The probabilities $q(w)$ are chosen as small as possible (to reduce the size of the support). However each coefficient in this system of equations should be at most 1, so that these equations constitute a lattice approximation problem. Therefore, as in [20], we choose:

$$q(w) = \max\left(\max_{s,t} \frac{[s \xrightarrow{w} t]\Pr_{\tilde{D}}\{w\}\Delta}{\Pr_{\tilde{D}}\{st\}}, \Pr_{\tilde{D}}\{w\}\Delta\right).$$

If the maximum is greater than 1, $q(w)$ is set to 1 (such a w is always retained into distribution D). Note that as a result of this case, we are actually not able to enforce that all coefficients in the equations be at most 1. This turns out to be only a minor technical difficulty because those strings with $q(w) = 1$ do not contribute to deviations from the mean. See App. B for a detailed analysis. Using summation as an upper bound for maximum, we obtain the following upper bound for $\sum_w q(w)$:

$$\sum_w \left(\sum_{st} \frac{[s \xrightarrow{w} t]\Pr_{\tilde{D}}\{w\}\Delta}{\Pr_{\tilde{D}}\{st\}} + \Pr_{\tilde{D}}\{w\}\Delta\right) = \sum_{st}\sum_w \frac{[s \xrightarrow{w} t]\Pr_{\tilde{D}}\{w\}\Delta}{\Pr_{\tilde{D}}\{st\}} + \Delta = (N+1)\Delta.$$

That is, when the w's are selected with probability $q(w)$, the expected size of $\text{supp}(D)$ is at most $(N+1)\Delta = E_0$, as desired.

Computing D from $\tilde{D}$. As we proceeded in the absolute error case, a lattice vector approximating Eqns. (12) and (13) is obtained and the support of D is defined to be the support of the lattice vector so obtained. As in that case, the w's are selected from a 2-wise independent probability space. Thus, it follows that with non zero probability there exists a lattice vector (whose support is identified with $\text{supp}(D)$ in the sequel) such that for for each M_i, $s = s_{i,l}$ and $t \in N_{i,l'}$ (see App. B):

$$\left|\sum_{w \in \text{supp}(D)} \frac{[s \xrightarrow{w} t]\Pr_{\tilde{D}}\{w\}\Delta}{q(w)\Pr_{\tilde{D}}\{st\}} - \Delta\right| \quad \leq \quad \lambda_2(\Delta, 1/(N+1))$$

$$\left|\sum_{w \in \text{supp}(D)} \frac{\Pr_{\tilde{D}}\{w\}\Delta}{q(w)} - \Delta\right| \quad \leq \quad \lambda_2(\Delta, 1/(N+1)).$$

Recalling that in the distribution D, $\Pr_D\{w\} = \Pr_{\tilde{D}}\{w\}/q(w)\gamma$, we rewrite the above equations as

$$\left|(\Pr_D\{st\}/\Pr_{\tilde{D}}\{st\})\gamma\Delta - \Delta\right| \leq \lambda_2(\Delta, 1/(N+1))$$

$$|\gamma\Delta - \Delta| \leq \lambda_2(\Delta, 1/(N+1))$$

Setting $\rho = \Pr_D\{st\}/\Pr_{\tilde{D}}\{st\}$ and dividing by Δ, we get

$$|\rho\gamma - 1| \le \frac{\lambda_2(\Delta, 1/(N+1))}{\Delta} \quad \text{and} \quad |\gamma - 1| \le \frac{\lambda_2(\Delta, 1/(N+1))}{\Delta}.$$

Now,

$$\left|\frac{\Pr_D\{st\}}{\Pr_{\tilde{D}}\{st\}} - 1\right| = |\rho - 1| \le \rho|1-\gamma| + |\rho\gamma - 1| \quad \le \quad \frac{\lambda_2(\Delta, 1/(N+1))}{\Delta}(1+\rho)$$

Requiring that $\lambda_2(\Delta, 1/(N+1))/\Delta \le 1/4$, this last inequality then implies that $\rho \le 2$. Then, substituting this upper bound for ρ in the right hand side we obtain

$$|\rho - 1| \le 3\frac{\lambda_2(\Delta, 1/(N+1))}{\Delta}$$

Choice of Δ and E_0. We need $3\lambda_2(\Delta, 1/(N+1))/\Delta \le \tilde{\delta}$. Since $\tilde{\delta} = \frac{\delta}{2n}$ and $E_0 = (N+1)\Delta$ from Eqn. (3) we obtain

$$E_0 \ge C\frac{n^2(\eta m)^2}{\delta^2} \tag{14}$$

3.4 Processor and Time Bounds

A variation of the algorithm in Lemma 2.1 is used for `reduce`. The recurrence for the number of processors used by $\texttt{fool}(l, l')$ is $W(h) \le 2W(h/2) + Cf(E_0^2)E_0 m$, where $f(n) = n$ is the size of a 2-wise independent probability space on n variables. So the total number of processors is $O(f(E_0^2)E_0 mn)$. Using Eqns. (10) and (14), we finally obtain the following. See App. C for details.

Theorem 3.1 *In the EREW PRAM model, a levelled RFA can be fooled with absolute error ϵ in $O(\log^2 n)$ time using $O(n^7\eta^6 m^4/\epsilon^6)$ processors, and with relative error δ in $O(\log^2 n)$ time using $O(n^{11}\eta^{10}m^{11}/\delta^{10})$ processors.*

For the lattice approximation problem, it is sufficient to use either absolute error with $\epsilon = 1/2m$, or relative error with $\delta = 1$. Thus, we obtain the following.

Theorem 3.2 *In the EREW PRAM model, the lattice approximation problem can be solved deterministically in the EREW PRAM model, resulting in discrepancies within a multiplicative factor $1 + o(1)$ of the probabilistic bound, using $O(\log^2 n)$ time and $O(n^7\eta^6 m^{10})$ processors.*

3.5 Using k-wise Independent Probability Spaces

In the calculations above, we used 2-wise independent sample spaces at each level of the recursion. Is there any gain in using k-wise independent sample spaces instead? Using the bounds in Eqn. (2), we obtain the following conditions: $E_0 \ge C\frac{n^2\eta^2 m^{2/k}}{\epsilon^2}$ in the absolute error case and $E_0 \ge C\frac{n^2(\eta m)^{1+2/k}}{\delta^2}$ in the relative error case. While this gives better bounds on E_0 the bound on the number of processors blows up significantly. This is because a k-wise independent probability space on E_0^2 variables has size $\Omega(E_0^k)$. So the choice of $k = 2$ seems to be optimal. Further details are given in App. C.

4 Discrepancy

4.1 Problem

The particular case of the lattice approximation problem in which each a_{ij} is 0 or 1 and each p_j is 1/2 corresponds to the well-known *set discrepancy* problem. It is usually stated as follows. We are given a set system $(X, \mathcal{S})$ where X is a *ground set* and $\mathcal{S}$ is a collection of subsets of X, $n = |X|$ and $m = |\mathcal{S}|$, and we are to compute a subset R from X, such that for each $S \in \mathcal{S}$, the *discrepancies* $||R \cap S| - |\overline{R} \cap S||$ are small. Let R be a sample from X with each $x \in X$ selected into R independently with probability 1/2. Then the Chernoff-Hoeffding bound for full independence, Eqn. (1), guarantees that with nonzero probability for each $S \in \mathcal{S}$:

$$\Big||R \cap S| - |\overline{R} \cap S|\Big| \leq \lambda(|S|/2, 1/m) = \Theta(\sqrt{|S| \log m}).$$

Generalizations and variations of the discrepancy problem have been extensively studied in combinatorics and combinatorial geometry (where $\mathcal{S}$ is determined from $X \subseteq \mathrm{I\!R}^d$ by specific geometric objects) [6, 2]. Computationally, it has also been object of extensive research [27, 29, 8]. Because of its importance, we consider in detail the work and time requirements for its solution in NC. Also, it is shown in [27] that an algorithm for the discrepancy problem can be used to solve the more general lattice approximation problem. As a result, if we are willing to loose a $\log n$ factor in the running time, and a constant factor in the value of discrepancy achieved, this leads to a substantial savings in the number of processors needed (though still much higher than the work performed sequentially).

4.2 Algorithm

The algorithm is just the specialization of the lattice approximation algorithm of Sect. 2. The RFAs effectively work as counters that for each $S \in \mathcal{S}$ store the number of elements of S that have been selected into R. Thus $\eta = n + 1$. The threshold λ_S that determines the rejecting states of M_S is set to $\lambda(|S|/2, 1/2m) = \Theta(\sqrt{|S| \log m})$, so that even after an absolute error less than $1/2m$ per RFA, or a relative error less than 1, still there is a good set with nonzero probability. This choice of λ_S results in a discrepancy that is larger than the probabilistic bound (which is achievable sequentially) by only a factor $1 + o(1)$. It is possible to do somewhat better than in the general lattice approximation algorithm as far as the number of processors required (see argument in App. C).

Theorem 4.1 *The discrepancy problem can be solved deterministically in the EREW PRAM model in $O(\log n \log(n + m)) = O(\log^2 n)$ time using $O(n^7 m^{10} \log^6 n)$ processors.*

This is still much greater than the work $O(nm)$ that is needed to compute a good discrepancy set sequentially. For $m = n$, the previous best parallel algorithm of Chari *et al*, [9] computes a set of discrepancy $O(\sqrt{|S|^{1+\epsilon} \log n})$ in $O(\log n)$ time using $O(n^{1+4/\epsilon})$ processors (for $\epsilon = 1/4$, that is $O(n^{17})$ processors, the same bound as ours).

4.3 Lattice Approximation Via Discrepancy

The algorithm for the lattice approximation problem in [27] is obtained by a reduction to the discrepancy problem. The resulting lattice approximation algorithm achieves discrepancies a constant factor larger, while it has essentially the same work bound as the discrepancy algorithm and a running time larger by a factor $\log n$. The reduction uses as an intermediate step, for the purpose of analysis, the *vector balancing* problem. This problem is a lattice approximation problem in which each $p_j = 1/2$. Our improvement also translates to this algorithm. As a result, we obtain the following.

Theorem 4.2 *In the EREW PRAM model, the lattice approximation problem can be solved deterministically, resulting in discrepancies within a multiplicative factor $O(1)$ from the probabilistic bound, for $\mu_i \geq \log m$, in $O(L \log^2 n) = O(\log^3 n)$ time using $O(n^7 m^{10} \log^6 n)$ processors.*

For completeness, the analysis is given in App. D (though it does follow closely the analysis in [27], one needs to verify that it is not dependent on discrepancies of the form $O(\sqrt{\mu^{1+\epsilon} \log m})$ as obtained there).

5 Sampling in Computational Geometry

Randomized algorithms have been very successful in computational geometry [13, 28] and, as a result, there has been interest in their derandomization. For this, two concepts capturing the characteristics of a sample have been developed: *approximations of range spaces* and *sampling in configuration spaces.* In both cases, our approach improves on previous NC constructions.

5.1 Approximations of Range Spaces

A *range space* is a set system $(X, \mathcal{R})$ consisting of a ground set X, $n = |X|$, and a set $\mathcal{R}$ of subsets of X called *ranges.* A subset $A \subseteq X$ is called an ϵ-approximation for $(X, \mathcal{R})$ if for each $R \in \mathcal{R}$, $||A \cap R|/|A| - |R|/|X|| \leq \epsilon$. For $Y \subseteq X$, the restriction $\mathcal{R}_{|Y}$ is the set $\{Y \cap R : R \in \mathcal{R}\}$. $(X, \mathcal{R})$ is said to have *bounded VC-exponent* if there is a constant d such that for any $Y \subseteq X$, $|\mathcal{R}_{|Y}| = O(|Y|^d)$. For $(X, \mathcal{R})$ with bounded VC-exponent, a random sample of size $O(r^2 \log r)$, where the multiplicative constant depends on d, is a $(1/r)$-approximation with nonzero probability [34, 2]. Sequentially, the method of conditional probabilities leads to a polynomial time algorithm for constructing these approximations with optimal size (matching the probabilistic bound). With a constant size loss in the size, they can be constructed in $O(nr^C)$ time, for some constant C that depends on $(X, \mathcal{R})$ [24, 11]. Furthermore, for some range spaces that here we just call *linearizable*, and for $r \leq n^\epsilon$, some $\epsilon > 0$ depending on the range space, the construction can be performed in $O(n \log r)$ time [25]. In parallel (NC), however, only size $O(r^{2+\delta})$ has been achieved using k-wise independent probability spaces [15, 16, 17]. There is a close relation with the discrepancy problem. In fact, when the random sample R is of size $|X|/2$, the low discrepancy and approximation properties are (almost) equivalent. From the definition, it is clear that the same approach used for the discrepancy problem can be used to compute an approximation of optimal size in parallel. Taking advantage of the good behavior of approximations under partitioning and iteration [24], the running times of the algorithms can be improved as stated in the following theorem, with only a constant factor loss in the size (details omitted here). The results for the CRCW PRAM model in [16, 17] can be similarly improved.

Theorem 5.1 *A $(1/r)$-approximation of size $O(r^2 \log r)$ of a range space $(X, \mathcal{R})$, $|X| = n$, can be computed deterministically in the EREW PRAM model in $O(\log n + \log^2 r)$ time using $O(nr^C)$ work, for some $C > 0$. If $(X, \mathcal{R})$ is linearizable, then for $r \leq n^\epsilon$, for some $0 < \epsilon < 1$, the construction can be performed in $O(\log n \log r)$ time using $O(n \log r)$ work.*

5.2 Sampling in Geometric Configuration Spaces

Configuration spaces [13, 10, 28, 26] provide a general framework for geometric sampling. A *configuration space* is a 4-tuple $(X, \mathcal{T}, \mathrm{trig}, \mathrm{kill})$ where: X is a finite set of objects, $n = |X|$; $\mathcal{T}$ is a mapping that assigns to each $S \subseteq X$ a set $\mathcal{T}(S)$ called the *regions* determined by S, let $\mathcal{R}(X) = \cup_{S \subseteq X} \mathcal{T}(S)$; trig is a mapping $\mathcal{R}(X) \to 2^X$ indicating for each $\sigma \in \mathcal{R}(X)$ the set of objects in X that *trigger* σ; kill is a mapping $\mathcal{R}(X) \to 2^X$ indicating for each $\sigma \in \mathcal{R}(X)$ the set of objects in X that *kill* σ. We are interested in configuration spaces that satisfy the following *axioms*: (i) $d = \max\{|\mathrm{trig}(\sigma)| : \sigma \in \mathcal{R}\}$ is a constant, called the *dimension* of the configuration space; furthermore, for $S \subseteq X$ with $|S| \leq d$, the number of regions determined by S is at most a constant number E. (ii) For all $S \subseteq X$ and $\sigma \in \mathcal{R}(X)$, $\sigma \in \mathcal{T}(S)$ iff $\mathrm{trig}(\sigma) \subseteq S$ and $S \cap \mathrm{kill}(\sigma) = \emptyset$. The following sampling theorem is the basis for many geometric algorithms [13].[8]

Theorem 5.2 *Let $(X, \mathcal{T}, \mathrm{trig}, \mathrm{kill})$ be a configuration space, with $n = |X|$, satisfying axioms (i) and (ii), and for an integer $1 \leq r \leq n$ let R be a sample from X with each element of X taken into R independently with probability $p = r/n$. Then: $\mathbf{E}\left[\sum_{\sigma \in \mathcal{T}(R)} \exp\left(\frac{r}{2n}|\mathrm{kill}(\sigma)|\right)\right] \leq 2^{d+1} f(r/2)$ where $f(r)$ is an upper bound for $\mathbf{E}[|\mathcal{T}(R)|]$. It follows that with nonzero probability: (1) For all $\sigma \in \mathcal{T}(R)$: $|\mathrm{kill}(\sigma)| \leq C\frac{n}{r}\log r$, and (2) For all integer $j \geq 0$: $\sum_{\sigma \in \mathcal{T}(R)} |\mathrm{kill}(\sigma)|^j \leq C\left(\frac{n}{r}\right)^j f(r/2)$.*

Sequentially, a sample as guaranteed by the sampling theorem can be computed in polynomial time (using the method of conditional probabilities). Through the use of a $(1/r)$-approximation, the time can be reduced to $O(nr^C)$, and for *linearizable* configuration spaces, for $r \leq n^\epsilon$, to $O(n \log r)$. In parallel (NC), k-wise independence can only guarantee part (2) of the theorem for $j = O(k)$ (but not part (1)) [4, 5]. Modelling the sampling with levelled RFAs, and fooling them with *relative* error, we can construct in parallel a sample as guaranteed by the sampling theorem, except for a constant multiplicative factor. Relative error is needed because of the exponential weighting that makes even small probability events relevant. We obtain the following (details omitted here).

Theorem 5.3 *A sample as guaranteed by the sampling theorem can be computed deterministically in the EREW PRAM model in $O(\log n + \log^2 r)$ time using $O(nr^C)$ work; and in the case of a linearizable configuration space and $r \leq n^\epsilon$, in $O(\log n \log r)$ time using $O(n \log r)$ work.*

6 Other Applications

6.1 Approximation of Integer Linear Programs

An NC algorithm for approximating positive linear programs was proposed in [23]. To solve positive integer linear programs approximately in NC, Alon and Srinivasan [3] propose mimicking the approach of Raghavan [31] - first solve the program without integrality constraints, approximately, using [23], and then use the NC lattice approximation algorithm of [27] as a rounding black box to obtain an integral solution. However the second step introduces an additional error since [27] only guarantees $O(\sqrt{\mu_i^{1+\epsilon}\log m})$ discrepancy sets. In [3] the error introduced as a result of using the lattice approximation algorithm of [27] was corrected partially in some cases. Our algorithm essentially reduces the error introduced by lattice approximation to the minimum possible.

[8] This is not exactly the form stated there, but it follows using the same proofs.

6.2 Edge Coloring of Graphs

Let $G = (V, E)$ be an undirected graph whose maximal degree is Δ. A *legal edge coloring* is an assignment of colors to the edges such that two edges incident to the same vertex cannot have the same color. Vizing's theorem states that G can be edge colored with $\Delta + 1$ colors, and it implies a polynomial time sequential algorithm to find such coloring. The best deterministic parallel algorithm is the derandomization in [27] of an algorithm in [22]. It uses a discrepancy algorithm and produces a coloring with $\Delta + O(\sqrt{\Delta^{1+\epsilon}})$ colors. For $\Delta = \Omega(\log n)$, substituting there our discrepancy algorithm produces a coloring with $\Delta + O(\sqrt{\Delta \log n})$ colors.

References

[1] M. Ajtai. Approximate counting with uniform constant depth circuits. In J.-Y. Cai, editor. *Advances in Computational Complexity Theory,* DIMACS Series in Discrete Mathematics and Theoreical Computer Science. 1–20. American Mathematical Society, 1993.

[2] N. Alon and J. Spencer. The Probabilistic Method. *Wiley–Interscience*, 1992.

[3] N. Alon and A. Srinivasan. Improved parallel approximation of a class of integer programming problems. *Algorithmica* **17** (1997) 449–462.

[4] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci.*, 1994, 683–694.

[5] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Computing faces in segment and simplex arrangements. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, 1995, 672–682.

[6] J. Beck and W. Chen. Irregularities of distribution. Cambridge University Press, 1987.

[7] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci.*, 1994, 276–287.

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

[9] S. Chari, P. Rohatgi and A. Srinivasan. Improved Algorithms via Approximations of Probability Distributions. In *Proc. ACM Sympos. Theory Comput.* (1994) 584–592.

[10] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica* **10** (1990) 229–249.

[11] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 281–290, 1993.

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

[13] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, **4** (1989) 387–421.

[14] T. Goldberg and U. Zwick. Optimal deterministic approximate parallel prefix sums and their applications. In *Proc. 4th IEEE Israel Symp. on Theory of Computing and Systems*, 220–228, 1995.

[15] M.T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, 73–82, 1993.

[16] M.T. Goodrich. Fixed-dimensional parallel linear programming via relative ϵ-approximations. In *Proc. 7th ACM-SIAM Sympos. Discr. Alg. (SODA)*, 1996, 132–141.

[17] M.T. Goodrich and E.A. Ramos. Bounded independence derandomization of geometric partitioning with applications to parallel fixed-dimensional linear programming. To appear in *Discrete and Comput. Geom.*

[18] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statist. Assoc. J.* **58** (1963) 13–30.

[19] A. Joffe. On a set of almost deterministic k-independent random variables. *Annals of Probability* **2** (1974) 161–162.

[20] D.R. Karger and D. Koller. (De)randomized constructions of small sample spaces in NC. In *Proc. 35th Annu. IEEE Sympos. Foundations Comput. Sci.*, 1994, 252–263.

[21] H.J. Karloff and Y. Mansour. On construction of k-wise independent random variables. In *Proc. 26th Annu. ACM Sympos. Theory Comput.*, 1994, 564–573.

[22] H.J. Karloff and D.B. Shmoys. Efficient parallel algorithms for edge coloring problems. Ĵ. Algorithms **8** (1987) 39–52.

[23] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, 1993, 448–457.

[24] J. Matoušek. Approximations and optimal geometric divide-and-conquer. In *Proc. 23rd Annu. ACM Sympos. Theory Comput.*, 1991, 505–511. Also in *J. Comput. Syst. Sci.* **50** (1995) 203–208.

[25] J. Matoušek. Efficient partition trees *Discrete Comput. Geom.* **8** (1992) 315–334.

[26] J. Matoušek. Derandomization in computational geometry. Available in the web site: http:// www.ms.mff.cuni.cz/ acad/kam/matousek/ Earlier version appeared in *J. Algorithms.*

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

[28] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1993.

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

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

[31] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *J. Comput. Syst. Sci.* **37** (1988) 130–143.

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

[33] J.P. Schmidt, A. Siegel and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.* **8** (1995) 223-250.

[34] V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.* **16** (1971) 264–280.

A Accumulation of Error

A.1 Absolute Error

Let $h' = l'' - l$ and $h'' = l' - l''$, then

$$\begin{aligned}
\sum_{t \in N_{i,l'}} \left|\mathrm{Pr}_{\tilde{D}}\{st\} - \mathrm{Pr}_{F_h}\{st\}\right| &= \sum_{t \in N_{i,l'}} \left| \sum_{r \in N_{i,l''}} \mathrm{Pr}_{D_1}\{sr\}\mathrm{Pr}_{D_2}\{rt\} - \sum_{r \in N_{i,l''}} \mathrm{Pr}_{F_{h'}}\{sr\}\mathrm{Pr}_{F_{h''}}\{rt\} \right| \\
&\leq \frac{1}{2} \sum_{t \in N_{i,l'}} \sum_{r \in N_{i,l''}} \Big| (\mathrm{Pr}_{D_1}\{sr\} - \mathrm{Pr}_{F_{h'}}\{sr\})(\mathrm{Pr}_{D_2}\{rt\} + \mathrm{Pr}_{F_{h''}}\{rt\}) + \\
&\qquad (\mathrm{Pr}_{D_1}\{sr\} + \mathrm{Pr}_{F_{h'}}\{sr\})(\mathrm{Pr}_{D_2}\{rt\} - \mathrm{Pr}_{F_{h''}}\{rt\}) \Big| \\
&\leq \sum_{r \in N_{i,l''}} \left|\mathrm{Pr}_{D_1}\{sr\} - \mathrm{Pr}_{F_{h'}}\{sr\}\right| \frac{1}{2} \sum_{t \in N_{i,l'}} (\mathrm{Pr}_{D_2}\{rt\} + \mathrm{Pr}_{F_{h''}}\{rt\}) + \\
&\quad + \sum_{t \in N_{i,l'}} \left|\mathrm{Pr}_{D_2}\{rt\} - \mathrm{Pr}_{F_{h''}}\{rt\}\right| \frac{1}{2} \sum_{r \in N_{i,l''}} (\mathrm{Pr}_{D_1}\{sr\} + \mathrm{Pr}_{F_{h'}}\{sr\}) \\
&\leq 2\epsilon_{k-1}.
\end{aligned}$$

A.2 Relative Error

To verify the claim notice that, using the inequality $|ab - 1| \leq (1 + |a-1|)(1 + |b-1|) - 1$ and $h' = l'' - l, h'' = l' - l''$,

$$\left| \frac{\mathrm{Pr}_D\{st\}}{\mathrm{Pr}_{F_h}\{st\}} - 1 \right| \leq \left(1 + \left| \frac{\mathrm{Pr}_D\{st\}}{\mathrm{Pr}_{\tilde{D}}\{st\}} - 1 \right| \right) \left(1 + \left| \frac{\mathrm{Pr}_{\tilde{D}}\{st\}}{\mathrm{Pr}_{F_h}\{st\}} - 1 \right| \right) - 1,$$

and that

$$\begin{aligned}
\left| \frac{\mathrm{Pr}_{\tilde{D}}\{st\}}{\mathrm{Pr}_{F_h}\{st\}} - 1 \right| &= \frac{1}{\mathrm{Pr}_{F_h}\{st\}} \left|\mathrm{Pr}_{\tilde{D}}\{st\} - \mathrm{Pr}_{F_h}\{st\}\right| \\
&= \frac{1}{\mathrm{Pr}_{F_h}\{st\}} \left| \sum_{r \in N_{i,l''}} \Big(\mathrm{Pr}_{D_1}\{sr\}\mathrm{Pr}_{D_2}\{rt\} - \mathrm{Pr}_{F_{h'}}\{sr\}\mathrm{Pr}_{F_{h''}}\{rt\}\Big) \right| \\
&\leq \frac{1}{\mathrm{Pr}_{F_h}\{st\}} \sum_{r \in N_{i,l''}} \left| \frac{\mathrm{Pr}_{D_1}\{sr\}}{\mathrm{Pr}_{F_{h'}}\{sr\}} \frac{\mathrm{Pr}_{D_2}\{rt\}}{\mathrm{Pr}_{F_{h''}}\{rt\}} - 1 \right| \mathrm{Pr}_{F_{h'}}\{sr\}\mathrm{Pr}_{F_{h''}}\{rt\} \\
&\leq \frac{1}{\mathrm{Pr}_{F_h}\{st\}} \sum_{r \in N_{i,l''}} \Big((1 + \delta_{k-1})^2 - 1\Big) \mathrm{Pr}_{F_{h'}}\{sr\}\mathrm{Pr}_{F_{h''}}\{rt\} \\
&= (1 + \delta_{k-1})^2 - 1.
\end{aligned}$$

B Existence of a Lattice Vector in the Relative Error Case

Let S be the 2-wise probability space used by `reduce`. Each point in the sample space S corresponds to a distribution D as described in Sect. 3.3. We abuse notation and identify a point in S

with the distribution D it gives rise to. We need to show that there is a point D in S such that for all s, t pairs,

$$\left|\sum_{w \in \text{supp}(D)} \frac{[s \xrightarrow{w} t]\Pr_{\tilde{D}}\{w\}\Delta}{q(w)\Pr_{\tilde{D}}\{st\}} - \Delta\right| \leq \lambda_2(\Delta, 1/(N+1))$$

$$\left|\sum_{w \in \text{supp}(D)} \frac{\Pr_{\tilde{D}}\{w\}\Delta}{q(w)} - \Delta\right| \leq \lambda_2(\Delta, 1/(N+1))$$

The proof is a straightforward application of Chebychev's inequality. Some care has to be taken for w's selected with probability $q(w) = 1$. Such w's are in the support of every distribution obtained from S.

Fix any pair of states s, t. Let W_1 be the set of w's in $\text{supp}(\tilde{D})$ such that $q(w) = 1$, and let W_2 be $\text{supp}(\tilde{D}) - W_1$. We define a random variable X_w for every $w \in \text{supp}(\tilde{D})$. For $w \in W_1$, X_w takes the value $\frac{[s \xrightarrow{w} t]\Pr_{\tilde{D}}\{w\}\Delta}{\Pr_{\tilde{D}}\{st\}}$ with probability $q(w) = 1$ (note that in this case the variable is not really random). For $w \in W_2$, X_w takes the value $\frac{[s \xrightarrow{w} t]\Pr_{\tilde{D}}\{w\}\Delta}{q(w)\Pr_{\tilde{D}}\{st\}}$ with probability $q(w)$ and a value 0 with probability $1 - q(w)$. From the definition of $q(w)$ $X_w, w \in W_2$ is a $[0, 1]$ random variable. It is easy to see that random variable $X = \sum_{w \in \text{supp}(\tilde{D})} X_w$, has expectation Δ. Now $X = \sum_{w \in W_1} X_w + \sum_{w \in W_2} X_w$. The contribution to X from $w \in W_1$ is always $\sum_{w \in W_1} \frac{[s \xrightarrow{w} t]\Pr_{\tilde{D}}\{w\}\Delta}{\Pr_{\tilde{D}}\{st\}}$, since these w's are in the support of every sample point in S. So, the probability that X deviates from its mean Δ by an amount λ is exactly the probability that $\sum_{w \in W_2} X_w$ deviates from its mean by λ. Since these are $[0, 1]$ random variables, Chebychev's inequality applies. Let X_2 denote $\sum_{w \in W_2} X_w$. It is clear that $E[X_2]$ is at most Δ, and so we get

$$\begin{aligned}
\Pr_{D \in S}\{|X - \Delta| \geq \lambda_2(\Delta, 1/(N+1))\} &= \Pr_{D \in S}\{|X_2 - E[X_2]| \geq \lambda_2(\Delta, 1/(N+1))\} \\
&\leq \Pr_{D \in S}\{|X_2 - E[X_2]| \geq \lambda_2(E[X_2], 1/(N+1))\} \\
&< 1/(N+1)
\end{aligned}$$

In other words,

$$\Pr\left\{\left|\sum_{w \in \text{supp}(D)} \frac{[s \xrightarrow{w} t]\Pr_{\tilde{D}}\{w\}\Delta}{q(w)\Pr_{\tilde{D}}\{st\}} - \Delta\right| \geq \lambda_2(\Delta, 1/(N+1))\right\} \leq 1/(N+1)$$

A similar argument shows that

$$\Pr\left\{\left|\sum_{w \in \text{supp}(D)} \frac{\Pr_{\tilde{D}}\{w\}\Delta}{q(w)} - \Delta\right| \geq \lambda_2(\Delta, 1/(N+1))\right\} \leq 1/(N+1)$$

Since there are at most N+1 equations, there is a sample point satisfying all the equations.

C Processor and Time Bounds

C.1 Lattice Approximation

Let us consider how `reduce` obtains D from $\tilde{D} = D_1 \times D_2$. The probabilities $\Pr_{\tilde{D}}\{st\}$ that we aim to preserve are computed in time $O(\log(n+m))$ and using $O(E_0^2 m)$ processors (for each w in the support of $\tilde{D}$, which is of size $O(E_0^2)$, and each of the m RFAs M_i, determine the state $t \in N_{i,l'}$ that is reached from $s_{i,l}$; collect those that reach the same state and add up the corresponding probabilities). Let P be the k-wise independent probability space used in the reduction. Each $p \in \mathrm{supp}(P)$, which corresponds to a probability distribution on $\mathrm{supp}(\tilde{D})$, is tested to determine a good one (which is guaranteed to exist by the computations of the previous subsection). Thus, for each $p \in P$ and each $i = 1, \ldots, m$ the following is done: For each $w \in \mathrm{supp}(\tilde{D})$, determine the state $t \in N_{i,l'}$ that is reached from $s_{i,l}$, and then add for each t all $\Pr_p\{w\}$ for w that lead to t. This gives all the probabilities $\Pr_p\{st\}$, and from this information we can determine a good p. The amount of work performed is then

$$E_0^2 \cdot m + f(E_0^2) \cdot E_0 \cdot m$$

where $f(E_0^2)$ is the size of P, E_0 is the size of $\mathrm{supp}(p)$ for $p \in P$ and m is the number of RFAs. The second term, $f(E_0^2)E_0 m$, dominates. The time required is $O(\log(n+m)) = O(\log n)$.

The number of processors needed is dominated by those needed at the bottom level, where there are $O(n)$ merges. So the total number of processors needed is is $O(f(E_0^2)E_0 mn)$. Since the size of a k-wise, k even, independent probability space for x variables is $f(x) = O(x^{ck})$, where $c \geq 1/2$, we conclude from the expressions we obtained for E_0, that the best choice is $k = 2$.

For the case of absolute error, in which the distributions are uniform, $f(x) = O(x)$ is achievable: Use hash functions to generate P, following the approach of Nisan [30]. Let H be a 2-wise independent family of hash functions $h : E_0 \to E_0$. The size of H is E_0^2. P is generated from H as follows: For $h \in H$, let $p_h \in P$ be the uniform distribution with $\mathrm{supp}(p_h) = \{wh(w) : w \in E_0\}$. The 2-wise independence of H implies the 2-wise independence of P and, obviously, the size of P is also E_0^2.[9] This results in a number of processors $O(E_0^3 mn)$. Finally, replacing the condition for E_0, this is at most proportional to

$$\left(\frac{n^2\eta^2 m}{\epsilon^2}\right)^3 mn = \frac{n^7\eta^6 m^4}{\epsilon^6}.$$

For the case of relative error, with $k = 2$, we have $f(x) = O(x^2)$ using the construction in [19]. For this, it is important to note that the probabilities $q(w)$ can be truncated to $\log n$ bits (by the same argument used for the p_j's). So we obtain the following upper bound for the number of processors

$$\left(\frac{n^2\eta^2 m^2}{\delta^2}\right)^5 mn = \frac{n^{11}\eta^{10} m^{11}}{\delta^{10}}.$$

Since the number of levels is $O(\log n)$, the time required is $O(\log n \log(n+m)) = O(\log^2 n)$.

C.2 Discrepancy

Let us consider the absolute error case. Consider a reduction at level j of the recursion of the procedure `fool`, that is, of depth $h = d - k$ (depth 0 at the top). We have assumed that the number of states in a level of each RFA M_i is $\eta = n$. A better bound is $\eta_h = n/2^h$. Also,

[9]For the reader familiar with Nisan's construction, we point out that in his construction all the subproblems generated at the same level of recursion of `fool` would use the same good hash function h. This is important there to obtain a compact representation of the final pseudorandom strings, but here choosing a good h independently in each subproblem results in less work.

we have chosen $\tilde{\epsilon} = \epsilon/n$ equal for each level. A better choice is $\tilde{\epsilon}_h = \epsilon/2^h \log n$, while still the total accumulated error is ϵ (the total error is $\sum_h \tilde{\epsilon}_h 2^h$). Now E_0 also depends on the depth: $E_h = \eta_h^2 m/\tilde{\epsilon}_h^2$ (using 2-wise independence). Thus, the total number of processors needed at depth h is bounded by

$$\left(\frac{\eta_h^2 m}{\tilde{\epsilon}_h^2}\right)^3 \cdot m \cdot 2^h = \left(\frac{n^2 m \log^2 n}{\epsilon^2}\right)^3 m 2^h.$$

The total is dominated by the bottom level, that is by $(n^2 m \log^2 n/\epsilon^2)^3 mn$. Replacing $\epsilon = 1/2m$, we obtain the upper bound $n^7 m^{10} \log^6 n$.

D Lattice Approximation Via Discrepancy

D.1 Reduction from Vector Balancing to Discrepancy

Let us assume that each a_{ij} has L fractional bits, and let $a_{ij}^{(k)}$ be the k-th most significant one. That is, $a_{ij} = \sum_{k=0}^{L} a_{ij}^{(k)} 2^{-k}$. Also let $\mu_i^{(k)} = \frac{1}{2}\sum_{j=1}^{n} a_{ij}^{(k)}$, so that $\mu_i = \sum_{k=0}^{L} \mu_i^{(k)} 2^{-k}$. Note that $\mu_i^{(k)} \leq \mu_i 2^k$. The reduction is to transform the vector balancing problem with the $m \times n$ matrix A of coefficients a_{ij} into the discrepancy problem with an $m(L+1) \times n$ matrix A' obtained by writing in column the $m(L+1)$ bits of a_{ij}. The claim is that the solution to the discrepancy problem is a solution to the vector balancing problem with only a constant factor loss. Let q be a solution to the discrepancy problem. Then

$$\Delta_i^{(k)} = \left|\mu_i^{(k)} - \sum_{j=1}^{n} a_{ij}^{(k)} q_j\right| \leq C\sqrt{\mu_i^{(k)} \log(mL)} \leq C\sqrt{\mu_i \log(mL)} 2^{k/2}.$$

Then

$$\begin{aligned}
\left|\mu_i - \sum_{j=1}^{n} a_{ij} q_j\right| &= \left|\sum_{k=0}^{L} \mu_i^{(k)} 2^{-k} - \sum_{j=1}^{n}\sum_{k=0}^{L} a_{ij}^{(k)} 2^{-k} q_j\right| \leq \sum_{k=0}^{L}\left|\mu_i^{(k)} - \sum_{j=1}^{n} a_{ij}^{(k)} q_j\right| 2^{-k} \\
&\leq \sum_{k=0}^{L} C\sqrt{\mu_i \log(mL)} 2^{k/2} 2^{-k} \leq C\sqrt{\mu_i \log(mL)} \sum_{k=0}^{L} 2^{-k/2} \leq \alpha C\sqrt{\mu_i(\log m + \log L)} \\
&\leq \alpha\left(1 + \frac{\log L}{\log m}\right)^{1/2} C\sqrt{\mu_i \log m},
\end{aligned}$$

where $\alpha = \sum_{k=0}^{L} 2^{-k/2} = O(1)$.

D.2 Reduction from Lattice Approximation to Vector Balancing

The reduction uses *bit-by-bit* randomized rounding. Let us assume that each p_j has L fractional bits, and let $p_j^{(k)}$ be the k-th most significant one. That is, $p_j = \sum_{k=0}^{L} p_j^{(k)} 2^{-k}$. The bit-by-bit rounding consists of L stages. Let $p_j^{\{k\}}$ be the rounded version of p_j at the beginning of the k-th stage, so $p_j^{\{0\}} = p_j$ and $p_j^{\{L\}} = q_j$, the resulting lattice vector ($p_j^{\{k\}}$ has $L-k$ fractional bits, in particular q_j is 0 or 1). In the k-stage, the $(L-k)$-th significant bit of $p_j^{\{k\}}$, denoted $p_j^{[L-k]}$, is rounded: if nonzero then *round up* or *round down* with equal probability, that is,

$$p_j^{\{k+1\}} = p_j^{\{k\}} + \frac{1}{2^{L-k-1}}\left(q_j^{[L-k]} - \frac{1}{2} p_j^{[L-k]}\right),$$

where $q_j^{[L-k]}$ is 0 or 1 with equal probability if $p_j^{[L-k]} = 1$ and 0 otherwise. It is argued in [27] that this is equivalent to the original randomized rounding. In the deterministic version, $q_j^{[L-k]}$ is the solution to the vector balancing problem with matrix A and vector $p_j^{[L-k]}$. Let $\mu_i^{[L-k]} = \sum_{j=1}^m a_{ij} p_j^{[L-k]}$. The solution to the vector balancing problem satisfies

$$\Delta_i^{[L-k]} = \left| \sum_{j=1}^m a_{ij} q_j^{[L-k]} - \mu_i^{[L-k]} \right| \leq C \sqrt{\mu_i^{[L-k]} \log m}.$$

Let $\mu_i^{\{k\}} = \sum_{j=1}^m a_{ij} p_j^{\{k\}}$ and note that $\mu_i^{[L-k]} \leq 2^{L-k} \mu_i^{\{k\}}$. So

$$\Delta_i^{\{k+1\}} = \left| \mu_i^{\{k+1\}} - \mu_i^{\{k\}} \right| = \frac{1}{2^{L-k-1}} \Delta_i^{[L-k]} \leq \frac{2C}{2^{L-k}} \sqrt{\mu_i^{[L-k]} \log m} \leq \frac{2C}{2^{(L-k)/2}} \sqrt{\mu_i^{\{k\}} \log m}.$$

Assuming that $\mu_i \geq \log m$, we claim that $\left| \mu_i^{\{k\}} - \mu_i \right| \leq \alpha \sqrt{\mu_i \log m}$. This is verified inductively. First, using the induction hypothesis:

$$\Delta_i^{\{k+1\}} \leq 2C \frac{\sqrt{(\mu_i + \alpha \sqrt{\mu_i \log m}) \log m}}{2^{(L-k)/2}} \leq 2C(1+\alpha)^{1/2} \frac{\sqrt{\mu_i \log m}}{2^{(L-k)/2}}.$$

Then, the induction step:

$$\left| \mu_i^{\{k+1\}} - \mu_i \right| \leq \sum_{r=1}^{k+1} \Delta_i^{\{r\}} \leq \sum_{r=1}^{k+1} 2C(1+\alpha)^{1/2} \frac{\sqrt{\mu_i \log m}}{2^{(L-r)/2}} \leq \alpha \sqrt{\mu_i \log m},$$

as long as $\alpha \geq 2C(1+\alpha)^{1/2} \sum_{r=0}^{\infty} \frac{1}{2^{r/2}}$.