Minimizing Randomness in Minimum Spanning Tree, Parallel Connectivity, and Set Maxima Algorithms*

Seth Pettie and Vijaya Ramachandran
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
seth@cs.utexas.edu, vlr@cs.utexas.edu

TR-01-25

July 13, 2001

Abstract

There are several fundamental problems whose deterministic complexity remains unresolved, but for which there exist *randomized* algorithms whose complexity is equal to known lower bounds. Among such problems are the minimum spanning tree problem, the set maxima problem, the problem of computing connected components and (minimum) spanning trees in parallel, and the problem of performing sensitivity analysis on shortest path trees and minimum spanning trees. However, while each of these problems has a randomized algorithm whose performance meets a known lower bound, all of these randomized algorithms use a number of random bits which is linear in the number of operations they perform.

We address the issue of reducing the number of random bits used in these randomized algorithms. For each of the problems listed above, we present randomized algorithms that have optimal performance but use only a polylogarithmic number of random bits; for some of the problems our optimal algorithms use only $\log^* n$ random bits. Our results represent an exponential savings in the amount of randomness used to achieve the same optimal performance as in the earlier algorithms. Our techniques are general and could likely be applied to other problems.

*This work was supported by Texas Advanced Research Program Grant 003658-0029-1999 and NSF Grant CCR-9988160. Seth Pettie was also supported by an MCD Fellowship.

1 Introduction

For many fundamental algorithmic problems there is a discrepancy between what we know of their deterministic complexity versus their established randomized complexity. In this paper we examine some well-studied problems fitting this description: the minimum spanning tree problem, the parallel connectivity and parallel minimum spanning tree problems, the set maxima problem, and the sensitivity analysis problem on shortest paths trees and minimum spanning trees. We propose new algorithms for these problems which have optimal performance but which use significantly fewer random bits than earlier algorithms.

Randomized algorithms are frequently simpler, and, in the abstract, faster than their deterministic counterparts (e.g. testing primality [Rab80], median finding [BFP+72, FR75], computing minimum spanning trees [KKT95]). However, they make use of a commodity that is scarce in reality: a stream of perfectly random bits. In practice a stream of "random" bits is produced by a pseudo-random function which has been *seeded* with a little non-determinism[1], say the least significant digits from the computer's clock. It is not clear if the seed is truly random, and even less clear how good the commonly used pseudo-random functions are. Bach [Bac91] studied a few number theoretic algorithms under the assumption of a truly random seed and a commonly used pseudo-random function, and showed them to have a good probability of success, though not as good as guaranteed by using totally random bits. Karloff and Raghavan [KR93] assumed the same model and showed that Quicksort can have much poorer performance than predicted. Other peculiarities of certain pseudo-random functions were noted in [FLW92] for Monte Carlo simulations and in [HR96, Hsu97] for parallel implementations of basic graph algorithms. Although debunking commonly used pseudo-random functions is worthwhile, the lesson here is to design algorithms which demand much less randomness, and more important, to analyze them in a realistic abstract environment.

There has been considerable amount of work on derandomizing randomized algorithms. A common technique to reducing randomness is to use k-wise independent random variables rather than totally independent ones. The generation of k-wise independent and approximately k-wise independent random variables has been well-studied [Jof74, CG89, NN93, EGL+98, CRS00]. Several results are known on derandomizing randomized algorithms that use k-wise independence to obtain deterministic algorithms (see [KW85, Lub86, ABI86, Lub93, BR91]). Very recently Klivans and Spielman [KS01] gave a randomness-efficient method for testing if a multivariate polynomial is identically zero. In all of these algorithms a reduction in randomness is traded for an acceptable increase in the running time[2].

1.1 Our Results

In this paper we address the issue of reducing randomness but with an additional twist: *we require our algorithms to perform an optimal number of operations (to within a constant factor), and we focus on reducing randomness subject to this primary goal.* In the limit we would like to use no random bits at all, and obtain optimal deterministic algorithms. But in the absence of this ultimate result, it is a worthwhile goal to reduce our dependence on such a scarce resource as randomness.

[1] We note that there are now web servers (e.g. HotBits, *http://www.fourmilab.ch/hotbits/*) which will provide a modest number of bits upon request. The bits, apparently random, are derived by measuring some unpredictable physical process, such as the decay of a radioactive substance.

[2] One exception is an intermediate result in [Lub86] where a parallel MIS algorithm that uses a logarithmic number of random bits is given that has the same resource bounds as the original algorithm that used a linear number of random bits, but neither of these algorithms perform optimal work.

We propose optimal algorithms using a reduced number of random bits for all the problems given in the Introduction. In our algorithms, at most a polylogarithmic-sized random seed is assumed, and methods for deriving random variables from that seed are analyzed explicitly. Our results represent an exponential reduction over earlier optimal algorithms in the number of random bits used.

We describe our results below. A summary is given in Table 1. Our results are obtained using two main techniques – (1) using the properties of an arbitrary k-wise independent sampler, and (2) re-using random bits.

1.1.1 Parallel MST & Connectivity

The best deterministic parallel MST and connectivity algorithms [CV86, CV91, CHL99] run in logarithmic-time yet they all use *super*linear work. There are somewhat simpler logarithmic-time linear expected work randomized MST and connectivity algorithms [Gaz91, CKT96, HZ96, PR99, HZ01], but each uses a linear number of random bits.

We present a new randomized MST algorithm which requires only a pairwise independent sampler. Our parallel implementation, which runs on the EREW PRAM [KR92], takes expected linear work using $O(\log^3 n \log^* n)$ random bits. Our sampling approach differs from [KKT95] and previous parallel MST algorithms [CKT96, PR99]; it is conceptually simpler but not as easily parallelizable, resulting in a running time of $O(\log^2 n \log^* n)$.

A simpler version of our parallel MST algorithm also solves parallel connected components with the same resources, improving upon [HZ96, HZ01, Gaz91] in terms of the number of random bits used.

1.1.2 Set Maxima and Local Sorting

In the *set maxima* problem we are given a set system $(\chi, \mathcal{S})$ where χ is a set of n totally ordered elements and $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a collection of subsets of χ, and asked to determine the maximum element in each set $S_1, \ldots, S_m$. The goal is to obtain the solution with an algorithm that uses the minimum number of comparisons between elements in χ.

This intriguing problem seems to have been introduced by Graham, Yao and Yao [GYY80] who noted the trivial solution – just sort χ – and gave a simple $O(n + m2^m)$ time algorithm, which is optimal for very small m. A bound of Fredman appears in the same paper showing that an instance of set maxima can have no more than $\binom{m+n-1}{n-1}$ distinct solutions; this was later shown to be asymptotically tight in [GKKS93]. Liberatore [Lib98] has shown the set maxima problem to be precisely the problem of verifying the optimal base of an arbitrary matroid, and Karger [Kar93] has demonstrated the usefulness of set maxima in actually finding an optimal base. Many other concrete problems are instances of set maxima (or are reducible to it). These include verifying a partial order [KMK89], sensitivity analysis (including verification) of minimum spanning trees and shortest path trees [Tar82, Kom85], and orienting the edges of an undirected, node-weighted graph from the lesser to greater endpoint. This last problem was dubbed *local sorting* by Goddard et al. [GKKS93].

Besides the simple set maxima algorithm of [GYY80] and the trivial algorithm, there are really only two results to speak of for the general set maxima problem. Bar-noy et al. [BNMN92] gave a deterministic algorithm that uses $O(n)$ expected comparisons when the $m = n$ sets are chosen *randomly*. Goddard et al. gave an elegant randomized algorithm for set maxima which makes an optimal $O(n \log \frac{m+n}{n})$ expected comparisons using the same number of random bits.

We apply our k-wise independence result to improve Goddard et al.'s [GKKS93] local sorting and general set maxima algorithms as follows. We give an optimal local sorting algorithm which uses $O(\log n \log\log\log n)$ random bits, and an optimal set maxima algorithm which uses $O(\log n \log^{(3)} n\, 2^{O(\beta(m,n))})$ random bits, where[3] $\beta(m,n) = \log^* n - \log^* \frac{m}{n}$. Both algorithms make an expected $O(n \log \frac{m+n}{n})$ comparisons, which is optimal [GYY80, GKKS93].

1.1.3 Reusing Random Bits

Using the simple technique of re-using random bits, we obtain a dramatic reduction in the number of random bits used to find a minimum spanning tree and to perform sensitivity analysis on MSTs and shortest path trees. Sensitivity analysis subsumes the simpler MST/SSSP verification problem. For each of these problems there exist optimal deterministic algorithms *with unknown complexities* (see [PR00] for MST and [DRT92] for MST/SSSP sensitivity analysis), deterministic algorithms which take time $O(m\alpha(m,n))$, where $\alpha(m,n)$ is Tarjan's inverse-Ackermann function (see [Cha00a] for MST and [Tar82] for MST/SSSP sensitivity analysis) and expected linear-time algorithms which use a linear number of random bits [KKT95, DRT92, GKKS93]. For both problems we give expected linear-time algorithms which use just $\log^* n$ random bits.

1.2 Organization

The rest of the paper is organized as follows. In Section 2 we give a fairly general lemma on the expected behavior of a k-wise independent sampler and show how pairwise independent sampling can be performed efficiently on the EREW PRAM. In Section 3 we give a new parallel MST/connectivity algorithm requiring only pairwise independence. In Section 4 we observe that any 4-wise independent sampler works in a previous set maxima algorithm, and we give improved set maxima and local sorting algorithms which reduce the required number of random bits to polylogarithmic. Finally, in Section 5 we use the technique of re-using random bits to give a simple linear expected time MST algorithm (based on the optimal MST algorithm in [PR00]) and an expected linear-time algorithm for sensitivity analysis of MST and shortest path trees, both of which use only $\log^* n$ random bits.

2 Limited Independence Sampling

In this section we establish a fairly general result on k-wise independent sampling which suggests that $O(1)$-wise independence is nearly as good in situations common to many randomized sorting-type algorithms. The situation is this: we have a set of elements from a total order χ and wish to find an element on the cheap whose rank is close to some desired rank t. If we have an abundance of randomness, we can simply select each element of χ independently with probability p and take the rank tp sampled element as a decent approximation of the actual rank t element. A tradeoff between the efficiency and accuracy of this scheme can be had by manipulating the sampling probability p.

We show that by using just pairwise independence the expected rank (w.r.t. χ) of the rank t sampled element is $O(\frac{t}{p}\log n)$, and using a $2k$-wise independence, $k > 1$, its expectation is $O(\frac{t}{p})$. There is then a natural tradeoff between k and the concentration of the distribution of the t^{th} sampled element around its mean.

The following Lemma is just an extension of Chebyshev's inequality for 0/1 random variables. A more complex proof of this result appears in [SSS95]; our proof is elementary. We note that a similar, though incorrect, lemma appears in [Cha00c, p. 424].

[3] This is the same β function as the one defined in Fredman & Tarjan's minimum spanning tree algorithm [FT87]

Problem	Deterministic Bound	Probabilistic Bound	
		Best previous	This paper
Parallel $\mathcal{NC}$ graph connectivity (work)	$O(m\alpha(m,n))$ [CV91]	$O(m)$ $O(m)$ random bits [HZ01] for EREW [Gaz91] for CRCW	$O(m)$ (EREW) $o(\log^{3+\epsilon} n)$ random bits
Parallel $\mathcal{NC}$ minimum spanning trees (work)	$O(m \log^{(3)} n)$ [CV86]	$O(m)$ $O(m)$ random bits [PR99] for EREW [CKT96] for CRCW	$O(m)$ (EREW) $o(\log^{3+\epsilon} n)$ random bits
Local sorting (comparisons)	$O(n \log n)$ (trivial)	$O(n \log \frac{m+n}{n})$ $O(n \log \frac{m+n}{n})$ random bits [GKKS93]	$O(n \log \frac{m+n}{n})$ $o(\log^{1+\epsilon} n)$ random bits
Set maxima (comparisons)	$O(n \log n)$ (trivial)	$O(n \log \frac{m+n}{n})$ $O(n \log \frac{m+n}{n})$ random bits [GKKS93]	$O(n \log \frac{m+n}{n})$ $o(\log^{1+\epsilon} n)$ random bits
Minimum spanning trees (time)	$O(m\alpha(m,n))$ [Cha00a] $O(\text{Optimal}(m,n))$ [PR00]	$O(m)$ $O(m)$ random bits [KKT95]	$O(m)$ $O(\log^* n)$ random bits
MST/SSSP sensitivity analysis (time)	$O(m\alpha(m,n))$ [Tar82] $O(\text{Optimal}(m,n))$ [DRT92]	$O(m)$ $O(m)$ random bits [DRT92]	$O(m)$ $O(\log^* n)$ random bits

Abbreviations: EREW and CRCW are, resp., the Exclusive Read Exclusive Write and Concurrent Read Concurrent Write parallel RAMs. Every EREW algorithm is a CRCW algorithm. Optimal(m,n) is the decision-tree complexity of the respective problem. We denote by ϵ an arbitrarily small constant. With the exception of Set Maxima, $m = |E|$ is the number of edges and $n = |V|$ is the number of vertices; for Set Maxima m is the number of sets and n the number of elements.

Table 1. Summary of our results.

Lemma 2.1 *Let $X_1, \ldots, X_n$ be $2k$-wise independent $0/1$ random variables, each with mean μ. For $X = \sum_i X_i$ we have $\mu_X = \mu n$ and*

$$\Pr\left[|X - \mu_X| \geq t\right] < \left(\frac{4k\mu_X}{t^2}\right)^k$$

Proof: Along the lines of Chebyshev's inequality we have $\Pr[|X - \mu_X| \geq t] = \Pr[(X - \mu_X)^{2k} \geq t^{2k}]$ which is $\leq \frac{\mathbb{E}(X-\mu_X)^{2k}}{t^{2k}}$ by Markov's inequality. The numerator can be expanded into an expression of the form $\mathbb{E}\sum\prod(X_i - \mu)$ — the expectation of a sum of products. By identifying duplicate factors in each product we can simplify them to be of the form $\prod_i (X_i - \mu)^{a_i}$ where $\sum_i a_i = 2k$. Notice that because the X_i's are $2k$-wise independent the factors of each product are also independent. We may then rewrite the numerator in the form $\sum\prod\mathbb{E}(X_i - \mu)^{a_i}$ — a sum of products of expectations. Observe that in any term, if some $a_i = 1$ then $\mathbb{E}(X_i - \mu)^{a_i} = 0$ and the term disappears. We bound the numerator by first bounding a single term then bounding the number of non-zero terms. For the first, note that $\mathbb{E}(X_i - \mu)^{a_i} = \mu(1-\mu)[(1-\mu)^{a_i - 1} - (-\mu)^{a_i - 1}] \in (-\mu, \mu)$, hence each term is bounded by μ^k.

Bounding the number of non-zero terms is equivalent to a balls-and-bins problem: how many ways are there to put $2k$ balls in n bins (order counts!) such that all bins have zero or ≥ 2 balls? Let N be the number of non-zero terms, we have that

$$N \leq \sum_{i=1}^{k} \binom{n}{i} \binom{2k-i-1}{i-1} \frac{(2k)!}{2^k}$$

Here i represents the number of non-empty bins, $\binom{n}{i}$ the number of ways of selecting such bins, and $\binom{2k-i-1}{i-1}$ the number of ways to distribute the $2k - 2i$ balls still unaccounted for. Each such distribution of balls-to-bins can be realized by a number of distinct orderings, which is certainly no more than $(2k)!/2^k$. Simplifying the above expression,

$$\begin{aligned}
N &\leq \sum_{i=1}^{k} \frac{n^i}{i!} \cdot \frac{(2k-i-1)^{i-1}}{(i-1)!} \cdot \frac{(2k)!}{2^k} \\
&\leq \tfrac{4}{3} \cdot \frac{n^k}{k!} \cdot \frac{(2k)^k}{k!} \cdot \frac{(2k)!}{2^k} \qquad \{i^{\text{th}} \text{ term} \leq \tfrac{1}{4} \cdot (i+1)^{\text{th}} \text{ term.}\} \\
&\leq \tfrac{4}{3} \cdot \frac{1 + \frac{1}{24k-1}}{\sqrt{k\pi}} \cdot \left(\frac{ne}{k}\right)^k \cdot \left(\frac{2ke}{k}\right)^k \cdot \left(\frac{2k^2}{e^2}\right)^k \qquad \{\text{Stirling's approximation}\} \\
&< (4nk)^k
\end{aligned}$$

We conclude that $\mathbb{E}(X - \mu_X)^{2k} < (4\mu_X k)^k$

This bound is reasonably tight. For a lower bound consider just those terms with exactly k distinct factors (each repeated twice).

$$\begin{aligned}
N &\geq \binom{n}{k} \cdot \frac{(2k)!}{2^k} \\
&\geq \frac{n^k}{k!} \cdot \frac{(2k)!}{4^k} \qquad \{k \leq n/2\} \\
&\geq n^k \left(\frac{e}{k}\right)^k \left(\frac{4k^2}{e^2}\right)^k 4^{-k}
\end{aligned}$$

$$\geq \left(\frac{nk}{e}\right)^k$$

We conclude, using the fact that $E(X_i-\mu)^2 = \mu(1-\mu)$, that $\mathbb{E}(X-\mu_X)^{2k} = \Omega\left((nk\mu(1-\mu)e^{-1})^k\right)$ □

The main lemma of this section is given below.

Lemma 2.2 *Let χ be a set of totally ordered elements and χ_s be a subset of χ derived by sampling each element with probability p using a $2k$-wise independent sampler. Let Y be the number of unsampled elements less than* $\min \chi_s$. *Then*

$$\mathbb{E}(Y) \leq \begin{cases} \frac{4\ln(np)+O(1)}{p} & \text{for } k = 1 \\ \\ \frac{21}{p} & \text{for } k > 1 \end{cases}$$

and

$$\Pr[Y \geq \ell] \leq \min_{\nu \leq k} \left\{ \left(\frac{4\nu}{p\ell}\right)^\nu \right\}$$

Proof: Let $X_i = 1$ if the element of χ with rank i is sampled, and 0 otherwise. So $\mathbb{E}(X_i) = p$ and for any distinct indices $i_1, \ldots, i_{2k}$, $X_{i_1}, \ldots, X_{i_{2k}}$ are independent.

Let $S_\ell = \sum_{i=1}^{\ell} X_i$ count the number of ones in $X_1, \ldots, X_\ell$. We have that $\mathbb{E}(S_\ell) = p\ell$ and

$$\Pr[Y \geq \ell] = \Pr[S_\ell = 0] \leq \Pr[|S_\ell - \mathbb{E}(S_\ell)| \geq p\ell]$$

Using Lemma 2.1 we can bound $\Pr[Y \geq \ell]$ as follows.

$$\begin{aligned} \Pr[|S_\ell - \mathbb{E}(S_\ell)| \geq p\ell] &\leq \frac{\mathbb{E}(S_\ell - E(S_\ell))^{2k}}{(p\ell)^{2k}} \\ &< \left(\frac{4p\ell k}{(p\ell)^2}\right)^k \qquad \{ \text{ Lemma 2.1} \} \\ &= \left(\frac{4k}{p\ell}\right)^k \end{aligned}$$

The second part of the Lemma follows from the simple observation that any $2k$-wise independent distribution is also 2ν-wise independent for $\nu \leq k$.

To bound $\mathbb{E}(Y)$ we use a variation on a familiar identity. For a random variable Z taking on values from the naturals it is easy to show that $E(Z) = \sum_{i=1}^{\infty} \Pr[Z \geq i]$ (a similar expression can be used for real Z). Plugging our best bound on $\Pr[Y \geq \ell]$ into this identity gives a weak bound on the expectation of Y. We have $\mathbb{E}(Y) \leq (4kp^{-1})^k \cdot \sum_i i^{-k}$. Before we give a tighter analysis, consider the following bound on $\mathbb{E}(Z)$ for any natural r.v. Z.

$$\begin{aligned} \mathbb{E}(Z) = \sum_{i=1}^{\infty} i \cdot \Pr[Z = i] &\leq \delta + \sum_{i=\delta+1}^{\infty} \Pr[Z \geq i] \\ &\leq \delta + \Delta \sum_{i=0}^{\infty} \Pr[Z \geq \delta + 1 + \Delta \cdot i] \end{aligned}$$

We will now bound $\mathbb{E}(Y)$ using this inequality. Assume w.l.o.g. that k is 1 or 2. Letting $\delta = \Delta = \frac{\beta}{p}$, we have that

$$\begin{aligned}
\mathbb{E}(Y) &\leq \frac{\beta}{p} + \frac{\beta}{p}\sum_{i=0}^{\infty} \Pr[Y \geq \tfrac{\beta(i+1)}{p}] \\
&\leq \frac{\beta}{p}\left(1 + \left(\frac{4k}{\beta}\right)^k \sum_{i=1}^{\lceil np/\beta \rceil} i^{-k}\right) \\
\{\text{For } k=2 \text{ and } \beta = 10\} &\leq \frac{21}{p} \\
\{\text{For } k=1 \text{ and } \beta = 1\} &\leq \frac{4\ln(pn) + O(1)}{p}
\end{aligned}$$

□

We omit the proof of the following Lemma; it is similar to that of Lemma 2.2.

Lemma 2.3 *Let χ be a set of totally ordered elements and χ_s be a subset of χ derived by sampling each element with probability p using a $2k$-wise independent sampler. Let x_t be the element of χ_s with rank t and let Y_t be the number of elements in χ less than x_t. Then*

$$\mathbb{E}(Y_t) = \begin{cases} O(tp^{-1}\log(np)) & \text{for } k = 1 \\ O(tp^{-1}) & \text{for } k > 1 \end{cases}$$

2.1 Pairwise Independent Sampling on the EREW PRAM

In Section 3 we need a method for generating a set of sampled elements in linear time *in the size of the sample*. Furthermore, we would like it to work on the EREW PRAM, which is a much more realistic model than the CRCW PRAM. We solve both of these problems using Joffe's method for generating k-wise independent variables, given below.

Lemma 2.4 *(Joffe [Jof74]) Let q be prime, $a_0, a_1, \ldots, a_{k-1}$ be chosen uniformly at random from $\mathbb{Z}_q$, and $X(i) = \sum_{j=0}^{k-1} a_j \cdot i^j \pmod q$. Then $X(0), \ldots, X(q-1)$ are uniformly distributed over $\mathbb{Z}_q$ and k-wise independent.*

That is, for generating pairwise independent variables we require two random coefficients, a_0 and a_1. We assume that m (the number of edges) is prime and that all edges are given a unique ID in $\mathbb{Z}_m$; if m is composite we find a prime $q > m$ and include $q - m$ dummy edges. We sample the edges with probability (about) p as follows. If $X(i) = a_1 i + a_0 \pmod m \in [0..\lceil pm \rceil - 1]$ then edge i is sampled; otherwise it is not. Evaluating the polynomial X on m points is too expensive because the number of sampled elements could be sublinear in m. Under the assumption that $a_1 \neq 0$ we can generate the sampled graph by generating all solutions to $i = (j - a_0)a_1^{-1} \pmod m$ for $j \in [0..\lceil pm \rceil - 1]$. This leads us to the following scheme for assigning processors to sampled edges. It takes work linear in the size of the sample, usually $O(pm)$.

We assume an EREW PRAM with P processors, each of which knows m, a_0, a_1, a_1^{-1}, and its unique processor ID.

If $a_1 = 0$ and $a_0 \geq \lceil pm \rceil$ then $X(\cdot) = a_0$ and no edges are sampled.

If $a_1 = 0$ and $a_0 < \lceil pm \rceil$ then all edges are sampled. Processor k is assigned edges $\lceil \frac{m}{P} \rceil k$ through $\lceil \frac{m}{P} \rceil (k+1) - 1$.

If $a_1 \neq 0$, then processor k is assigned edges with IDs of the form $(j - a_0)a_1^{-1} \pmod m$, for $\lceil \frac{mp}{P} \rceil k \leq j < \lceil \frac{mp}{P} \rceil (k+1)$.

Notice that with Joffe's pairwise independent sampler, assigning EREW processors to sampled edges is quite easy, whereas using his 3-wise independent sampler would be more cumbersome. We would need to resolve the inevitable conflicts that occur when more than one processor attempts to claim the same edge.

2.2 Finding a Prime in Parallel

Joffe's [Jof74] method for generating pairwise independent variables relies on having a known prime. Since m is a relatively small number (w.r.t. the number of processors), we can find the first prime greater than m very easily. Baker and Harman (see [BS96, p. 225]) showed that if p_n is the n^{th} prime, then $p_n - p_{n-1} \leq n^{.535+o(1)}$. We use this bound to find the smallest prime not less than m.

Lemma 2.5 *Let q be the smallest prime such that $q \geq m$. Then with probability at least $1 - m^{-2c+1}$, q can be found on the EREW PRAM with $O(\log m)$ time, $c \log^2 m$ random bits, and $b(m) \cdot c \log m$ processors, where $b(m) = m^{.535+o(1)}$.*

Proof: We run the Miller-Rabin [Mil76, Rab80] primality test $c \log m$ times on each integer in $[m \,..\, m + b(m)]$, reusing the same random bits for each number tested. The probability that Miller-Rabin reports the wrong answer for any of the numbers is $\leq b(m) \left(\frac{1}{4}\right)^{c \log m} \leq m^{-(2c-1)}$. Each test uses $\log m$ random bits and takes time $O(\log m)$, hence finding the first prime $\geq m$ takes $c \log^2 m$ random bits and $O(\log m)$ time using $b(m) \cdot c \log m$ processors. □

If a better bound on $b(m)$ is established then finding the next prime $\geq m$ can be made deterministic. Consider Cramér's conjecture.

Conjecture 2.1 *(Cramér) Let p_n be the n-th prime. Then $p_n - p_{n-1} = O(\log^2 n)$.*

If this conjecture were true we could employ $O(\sqrt{m} \log m)$ processors to find the first prime $\geq m$ in $O(\log m)$ time deterministically.

3 A Parallel MST/Connectivity Algorithm

3.1 Overview

In this section we present a new randomized MST algorithm which uses the same type of approach as the one in Karger, Klein and Tarjan [KKT95]. Our algorithm has two desirable attributes. It, like the KKT algorithm, is based on 'Borůvka steps' and is therefore parallelizable. Second, we show that our algorithm can tolerate a lower quality random sampler. In particular we use a pairwise independent sampler in lieu of total independence. This allows us to reduce the number of random bits used to polylogarithmic in the input size.

We assume, in this section and in Section 5, a familiarity with the minimum spanning tree problem. See [CLR90] or [Tar82] for a description of MST.

3.2 An Alternate Randomized MST Algorithm

A traditional *Borůvka step* identifies and contracts at least half of the unidentified MST edges, reducing the number of vertices by at least a factor of two. If implemented in the usual way, each Borůvka step takes linear time and Borůvka's algorithm takes $O(m \log n)$ time. Below we define inductively an *approximate* Borůvka step, based upon a normal Borůvka step in a sampled graph.

Definition 3.1 *In the i^{th} approximate Borůvka step all edges are sampled with a fixed probability. An edge is* **eligible** *to participate in this step if it is sampled, it is not a self-loop, and it was not* **tainted** *in the first $i-1$ approximate Borůvka steps – see Definition 3.2.*

Definition 3.2 *In an approximate Borůvka step each vertex selects and contracts its minimum weight incident edge which is eligible by Definition 3.1. An unsampled edge (u, v) becomes* **tainted** *if it is lighter than either one of the edges selected by u or v.*

Let $G_0 = G$ be the original graph and let G_i be the graph after i approximate Borůvka steps. If for some vertex v no edge incident on v is sampled, we let v choose an imaginary infinite weight edge (v, ∞), thus tainting all unsampled edges incident on v. This guarantees that after $\log n$ approximate Borůvka steps the graph will contract to a single vertex.

i^{th} Approximate Borůvka Step:

1. Let G_s be derived from G_{i-1} by randomly sampling each edge with probability $p(i)$ and removing self-loops.
2. For each vertex v let e_v be the least weight *untainted* edge incident on v which appears in G_s, and let $F = \{e_v : v \in V(G_s)\}$
3. Let G_i be derived from G_{i-1} by contracting all edges in F.

After $\log n$ approximate Borůvka steps we have constructed an approximate MST (composed of all edges contracted in Step 3). We then employ a linear-time MST verification algorithm (see [DRT92, Kin97, Kom85]) to filter out those edges never tainted, then reduce the number of vertices in the graph by performing a few *exact* Borůvka steps. Repeated iterations — approximate Borůvka steps, filtering, and exact Borůvka steps — will eventually reduce the graph to a single vertex. All edges identified in the exact Borůvka steps belong to the MST. The efficiency of this algorithm depends upon how we implement the sampling and ensuring that (most of the time) the number of edges in the graph is reduced by a constant factor in each iteration. Before addressing these matters of efficiency we should prove correctness. Lemma 3.1, proved below, implies that edges never tainted after the $\log n^{\text{th}}$ approximate Borůvka step are indeed not in the MST.

Definition 3.3 *Let C be a subgraph of G and $U \subseteq E(G)$. A subgraph C is contractible w.r.t. U if for any $e_1 = (u_1, v_1), e_2 = (u_2, v_2) \in U$ where $u_1, u_2 \in C$, there exists a path in C between u_1 and u_2 consisting of edges with weight less than* $\max\{w(e_1), w(e_2)\}$.

Lemma 3.1 *Let C_v denote the subgraph of G contracted to form $v \in V(G_i)$. Then C_v is contractible w.r.t. all edges still untainted after the i^{th} approximate Borůvka step.*

Proof: Note that C_v is the union of some set $\{C_{v_1}, \ldots, C_{v_j}\}$ where $\{v_1, \ldots, v_j\} \subseteq V(G_{i-1})$. We inductively assume that the $\{C_{v_j}\}$ are contractible w.r.t. untainted edges. Consider first the case