# On Removing Randomness from a Parallel Algorithm for Minimum Cuts

(Extended Abstract)

Michael Luby [*] Joseph Naor [†] Moni Naor[‡]

TR-93-007

February 1993

## Abstract

The weighted minimum cut problem in a graph is a fundamental problem in combinatorial optimization. Recently, Karger suggested a randomized parallel algorithm for this problem. We show that a similar algorithm can be implemented using only $O(\log^2 n)$ random bits. We also show that our result holds for computing minimum weight $k$-cuts, where $k$ is fixed.

[*]ICSI, UC Berkeley. E-mail: luby@icsi.berkeley.edu. Research supported in part by NSF Grant CCR-9016468 and grant No. 89-00312 from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel

[†]Computer Science Department, Technion University. E-mail: naor@cs.technion.ac.il. Part of this work was done while the author was visiting the International Computer Science Institute at Berkeley, CA.

[‡]Department of Applied Mathematics, Weizmann Institute of Science. E-mail: naor@wisdom.weizmann.ac.il. This work was done while the author was at the IBM Almaden Research Center.

## 1 Introduction

The minimum cut problem is the following: partition the vertices of a graph into two disjoint sets so as to minimize the number of edges in the cut, i.e., edges adjacent to vertices that are in different sets. The graph may be weighted, in which case we want to minimize the weight of the edges in the cut. This problem has received much attention in the literature in the last 40 years. It is a fundamental problem in combinatorial optimization and has numerous applications, e.g., network design and reliability, sequencing and scheduling, location theory, partitioning problems, and heuristics for solving integer programming problems. (See [PQ]). The parallel complexity, however, remained unresolved. Recently, Karger [Ka] proposed a randomized algorithm for computing the minimum cut in a graph. This placed the problem in the complexity class RNC.

Let $G = (V, E)$ be a graph (or multigraph) where $|V| = n$ and $|E| = m$. The purpose of this paper is to show that a randomized parallel algorithm for computing the minimum cut (which is similar to Karger's algorithm), can be implemented using only $O(\log^2 n)$ random bits. This is in contrast to Karger's algorithm which requires a polynomial (in $n$) number of random bits. We view our algorithm as a step towards obtaining a deterministic algorithm for the problem. Alternatively, one can view random bits as a resource (such as time and space), to be used as sparingly as possible, and our result reduces the use of this resource over the algorithm suggested by Karger [Ka]. Reducing the number of random bits needed in computation is a line that has been explored by many researchers in recent years.

### 1.1 Previous work

The straightforward way to find the minimum cut in a graph is by comparing all $\{s, t\}$-minimum cuts for all possible choices of vertices $s$ and $t$. (In an $\{s, t\}$-cut we require that $s$ and $t$ be in different sets). Each $\{s, t\}$-minimum cut can be found by computing the maximum flow from $s$ to $t$. In a classic paper, Gomory and Hu [GH1] have shown that the minimum cut can be determined by computing an equivalent flow-tree which requires considering only $n - 1$ $\{s, t\}$ minimum cuts chosen appropriately. Recently, Hao and Orlin [HO] have shown how to pipeline the computation of the $n - 1$ minimum cuts so that the total running time is no more than a single maximum flow computation. The running time of the current best algorithm for maximum flow is slightly more than $O(mn)$. This was first achieved by Goldberg and Tarjan [GT] and the complexity of their algorithm is $O(mn \log(n^2/m)$. Their result was later improved by [Al, CH, CHM, KRT, PW]

For many years, since the paper of Gomory and Hu appeared, the only approach to the minimum cut problem was via maximum flows. This has changed in the last years, and several papers have shown that a minimum cut can be solved more efficiently than maximum flow. Nagamochi and Ibaraki [NI] gave an algorithm that runs in $O(mn + n^2 \log n)$ time, and this is currently the best deterministic algorithm. Very recently, Karger and Stein [KS] suggested a sequential version of Karger's parallel algorithm that runs in time $O(n^2\text{polylog}(n))$, and finds the minimum cut with probability $1 - \frac{1}{\text{poly(n)}}$. In the unweighted case, there are several efficient algorithms for computing the minimum cut. An $O(mn)$ algorithm was discovered by Podderyugin [Po], and independently by Matula [Ma]. Recently, Gabow [Ga] showed how to compute the minimum cut in $O(\ell n \log(n^2/m))$, where $\ell$ denotes the cardinality of the minimum cut.

In the parallel context, Goldschlager, Shaw and Staples [GSS] proved that for a given pair of vertices, $s$ and $t$, finding the minimum $\{s,t\}$ cut is P-complete. Karp, Upfal and Wigderson [KUW], and Mulmuley, Vazirani and Vazirami [MVV] placed the maximum matching problem in RNC. This implies, using a standard reduction of flow to matching, that the unweighted minimum cut problem is also in RNC. As mentioned earlier, very recently, Karger [Ka] showed that the minimum cut can be computed in RNC in the weighted case as well. The running time is $O(\log^3 n)$ time and the number of processors is $O(n^2 \log^4 n)$. (See [KS]).

## 1.2 Our results

The basic operation in our algorithm is that of *contraction*: when an edge is contracted, the two vertices adjacent to it are contracted and the set of edges connecting the two vertices is deleted; the set of edges leaving the contracted vertex is the union of the set of edges leaving each of the two vertices. Edges are never merged. The following proposition is easy to verify.

**Proposition 1.1** *Let $G = (V, E)$ be an unweighted multigraph, and let $e \in E$ be an edge such that there exists a minimum cut $\mathcal{C}$, where $e \notin \mathcal{C}$. Suppose edge $e$ is contracted. Then, cut $\mathcal{C}$ remains a minimum cut in $G$.*

In Karger's algorithm, we repeatedly choose an edge in random and contract it, until only two vertices remain. (This process can be simulated in NC). Karger proved that with probability at least $1/n^2$, the cut defined by the remaining edges is a minimum cut.

Our algorithm is divided into stages. At a given stage we contract each edge with probability which is proportional to its weight divided by the weight of the minimum cut. The random choices are pairwise independent. We show that with probability greater than a constant, when a stage finishes, the weight of the minimum cut does not change, and the number of vertices decreases by a constant. This implies that after $O(\log n)$ stages, the algorithm terminates with a minimum cut with probability which is at least polynomially small. Each stage in the algorithm can be implemented using $O(\log n)$ random bits; independence, however, is required between stages and the total number of random bits needed is $O(\log^2 n)$. We use the standard method of two point sampling [CG] to amplify the probability of success to a constant with only a constant factor increase in the number of random bits. The precise time and processor bounds are determined in Section 4.3.

We also consider multiway cuts, or $k$-cuts. Here we are required to partition the vertices into $k$ disjoint sets so as to minimize the number of edges in the cut, i.e., edges adjacent to vertices that are in different sets. If the graph is weighted, we want to minimize the weight of the edges in the cut. This problem is NP-complete for arbitrary $k$ [GJ]. Hochbaum and Goldschmidt [GH2] have shown that for fixed $k$, the minimum $k$-cut can be computed in polynomial time. It follows from Karger's results [Ka] that this problem is also in RNC. We show that for fixed $k$ we can compute in RNC a minimum weight $k$-cut using only $O(\log^2 n)$ random bits.

The paper is organized as follows. In Section 2, the algorithm for computing the minimum cut is introduced and analyzed. In Section 3, multiway cuts are considered. In Section 4 we consider implementation issues.

## 2 The basic algorithm and its analysis

Let $G = (V, E)$ be a multigraph where the number of vertices is denoted by $n$. We assume that: (i) The weight of the minimum cut, denoted by $\ell$, is known. (ii) If $G$ is weighted, then each edge $e$, which has weight $w_e \leq \ell$, is replaced by $w_e$ multiple edges. If $w_e > \ell$, then $e$ is contracted before running the algorithm. Clearly, $e$ does not participate in the minimum cut. We justify our assumptions in Section 4.

The basic algorithm is divided into stages and we stop when there are only two vertices left in the graph. At each stage we do the following:

1. Each edge **chooses** itself with probability $1/(c \cdot \ell)$ where $c$ is a constant. The choices are pairwise independent.

2. Each chosen edge **contracts** its endpoints.

We analyze a single stage of the algorithm, and give two different proofs (Theorems 2.2 and 2.3) that with probability greater than a constant, the number of vertices decreases by a constant factor (in a single stage). Suppose the algorithm is run $r = O(\log n)$ stages, (or until we are left with two vertices), each time using independent random bits. By Theorem 2.3, we stop with two vertices representing the minimum cut with probability at least $1/16^r$. We summarize this in the next theorem.

**Theorem 2.1** *There exists a constant $d$ such that the basic algorithm terminates with a minimum weight cut with probability greater than $1/n^d$.*

In Section 4 we show that the number of random bits needed to implement a stage is $O(\log n)$. Since we require independence between stages, the total number of random bits used by the basic algorithm is $O(\log^2 n)$.

For the sake of the analysis we fix a particular minimum cut $\mathcal{C}$. We define the following events: (The complement of event $A$ is denoted by $\overline{A}$).

- $A_e$: The event that edge $e$ is not chosen.

- $B_e$: The event that edge $e$ is chosen and none of the edges belonging to the cut $\mathcal{C}$ are chosen.

- $B_v$: The event defined by $\cup_e B_e$, where the union is taken over all edges $e \notin \mathcal{C}$ that are adjacent to $v$.

### 2.1 The first analysis

We begin with a graph theoretic lemma.

**Lemma 2.1** *Let $\alpha < 1$. The graph $G$ contains at least $n - \frac{2}{\alpha}$ vertices, where each vertex is adjacent to at least $(1-\alpha)\ell$ edges that are not in the cut $\mathcal{C}$.*

**Proof:** Let $U \subseteq V$ denote the set of vertices where each vertex is adjacent to at least $\alpha\ell$ edges in the cut $\mathcal{C}$. Then,

$$2\ell \geq |U| \cdot \alpha \cdot \ell$$

and hence,

$$|U| \leq \frac{2}{\alpha}$$

Since the degree of each vertex is at least $\ell$, the lemma follows. $\square$

**Lemma 2.2** *For an edge $e \notin \mathcal{C}$,*

$$\text{Prob}[B_e] \geq \frac{c-1}{c^2\ell}$$

**Proof:**

$$\text{Prob}\,[B_e] = \text{Prob}\left[\left(\bigcap_{f\in\mathcal{C}} A_f\right) \bigcap \overline{A_e}\right] = \text{Prob}\left[\left(\bigcap_{f\in\mathcal{C}} A_f\right)\middle|\overline{A_e}\right] \cdot \text{Prob}\left[\overline{A_e}\right] =$$

$$\left(1 - \text{Prob}\left[\left(\bigcup_{f\in\mathcal{C}} \overline{A_f}\right)\middle|\overline{A_e}\right]\right) \cdot \text{Prob}\left[\overline{A_e}\right] \geq \left(1 - \sum_{f\in\mathcal{C}} \text{Prob}\left[\overline{A_f}|\overline{A_e}\right]\right) \cdot \text{Prob}\left[\overline{A_e}\right] =$$

$$\left(1 - \sum_{f\in\mathcal{C}} \text{Prob}\left[\overline{A_f}\right]\right) \cdot \text{Prob}\left[\overline{A_e}\right] = \left(1 - \frac{1}{c}\right) \cdot \frac{1}{c\cdot\ell} = \frac{c-1}{c^2\cdot\ell}$$

$\square$

**Lemma 2.3** *For any pair of edges $e_1, e_2 \notin \mathcal{C}$,*

$$\text{Prob}[B_{e_1} \cap B_{e_2}] \leq \frac{1}{c^2\ell^2}$$

**Proof:**

$$\text{Prob}\,[B_{e_1} \cap B_{e_2}] = \text{Prob}\left[\overline{A_{e_1}} \cap \overline{A_{e_2}} \bigcap_{f\in\mathcal{C}} A_f\right] =$$

$$\left(\text{Prob}\left[\left(\bigcap_{f\in\mathcal{C}} A_f\right)\middle|\left(\overline{A_{e_1}} \cap \overline{A_{e_2}}\right)\right]\right) \cdot \text{Prob}\left[\overline{A_{e_1}} \cap \overline{A_{e_2}}\right] \leq \text{Prob}\left[\overline{A_{e_1}} \cap \overline{A_{e_2}}\right] = \frac{1}{c^2\ell^2}$$

$\square$

We denote by $W$ the subset of $V$ for which Lemma 2.1 holds.

**Lemma 2.4** *Let $v \in W$. Then,*

$$\text{Prob}\,[B_v] \geq \frac{(1-\alpha)(2c-3+\alpha)}{2c^2}$$

**Proof:** For a vertex $v \in W$, we define $E_v$ to be a subset of edges adjacent to $v$, that do not belong to $\mathcal{C}$, such that $|E_v| = \ell(1-\alpha)$. By the inclusion-exclusion formula,

$$\text{Prob}\left[B_v\right] \geq \text{Prob}\left[\bigcup_{e \in E_v} B_e\right] \geq \sum_{e \in E_v} \text{Prob}\left[B_e\right] - \sum_{e_1, e_2 \in E_v} \text{Prob}\left[B_{e_1} \cap B_{e_2}\right] \geq$$
$$\frac{\ell(1-\alpha)(c-1)}{c^2\ell} - \binom{\ell(1-\alpha)}{2} \cdot \frac{1}{c^2\ell^2} \geq \frac{(1-\alpha)(c-1)}{c^2} - \frac{(1-\alpha)^2}{2c^2} =$$
$$\frac{(1-\alpha)(2c-3+\alpha)}{2c^2}$$

□

**Theorem 2.2** *The probability that the number of vertices at the end of a stage is less than $63n/64$ and no edge belonging to the cut $\mathcal{C}$ is contracted is at least $\frac{1}{28}$.*

**Proof:** We choose $\alpha = 0.1$ and $c = 2.9$. By Lemma 2.1, $|W| \geq n - 20$; we assume that $n \geq 97$. (If $n$ is smaller, we can run any sequential algorithm to find the minimum cut in the graph.) By Lemma 2.4, for $v \in W$,

$$\text{Prob}[B_v] \geq \frac{1}{7}$$

Let $X_v$ be a random variable defined to be the indicator variable of event $B_v$. (If $v \notin W$, then $X_v = 0$). Clearly,

$$E\left[\sum_{v \in V} X_v\right] \geq \frac{(n-20)}{7}$$

We define the random variable $Y$ to be the number of vertices at the end of a stage. It is easy to see that $Y \leq n - \sum_{v \in V} X_v/2$. Hence,

$$E[Y] \leq n - E\left[\frac{\sum_{v \in V} X_v}{2}\right] \leq n - \frac{n-20}{14} \leq \frac{13n}{14} + 2$$

By Markov's inequality,

$$\text{Prob}[Y \geq \lambda E[Y]] \leq \frac{1}{\lambda}$$

Setting $\lambda = \frac{28}{27}$, we get that

$$\text{Prob}\left[Y \geq \left(\frac{13n}{14} + 2\right) \cdot \frac{28}{27} \cdot n\right] \leq \frac{27}{28}$$

For $n \geq 97$,

$$\frac{26n}{27} + \frac{56}{27} \leq \frac{63n}{64}$$

Hence, we get that with probability at least $\frac{1}{28}$, the number of vertices at the end of a stage is at most $\frac{63n}{64}$. □

## 2.2 The second analysis

**Lemma 2.5** *The probability that none of the edges in $\mathcal{C}$ are contracted in a stage is at least $1 - \frac{1}{c}$.*

**Proof:** We want to estimate the probability of the event $\bigcap_{e \in \mathcal{C}} A_e$.

$$\text{Prob}\left[\bigcap_{e\in\mathcal{C}} A_e\right] = 1 - \text{Prob}\left[\bigcup_{e\in\mathcal{C}} \overline{A_e}\right] \geq 1 - \sum_{e\in\mathcal{C}} \text{Prob}\left[\overline{A_e}\right] \geq 1 - \frac{1}{c}$$

□

**Lemma 2.6** *If $c \geq 4$, then for every four vertices $v_1, v_2, v_3, v_4 \in V$, the probability that at least one of $\{v_1, v_2, v_3, v_4\}$ is adjacent to a chosen edge is at least $\frac{3}{2\cdot c}$.*

**Proof:** Since the cardinality of the minimum cut is $\ell$, the degree of each vertex is at least $\ell$. Therefore, there are at least $2\ell$ different edges adjacent to the set $\{v_1, v_2, v_3, v_4\}$. Let $E_v$ denote a subset of cardinality $2\ell$ of the edges adjacent to these vertices. By the inclusion-exclusion formula,

$$\text{Prob}\left[\bigcup_{e\in E_v} \overline{A_e}\right] \geq \sum_{e\in E_v} \text{Prob}\left[\overline{A_e}\right] - \sum_{e_1,e_2\in E_v} \text{Prob}\left[\overline{A_{e_1}} \cap \overline{A_{e_2}}\right]$$

Since the events $A_e$, where $e \in E_v$, are pairwise independent, we have that for any $e_1, e_2 \in E_v$, $\text{Prob}\left[\overline{A_{e_1}} \cap \overline{A_{e_2}}\right] = \text{Prob}\left[\overline{A_{e_1}}\right] \cdot \text{Prob}\left[\overline{A_{e_2}}\right]$. Substituting in the above,

$$\text{Prob}\left[\bigcup_{e\in E_v} \overline{A_e}\right] \geq \frac{|E_v|}{c\cdot\ell} - \frac{|E_v|\left(|E_v|-1\right)}{2\cdot c^2\cdot\ell^2} \geq \frac{2}{c} - \frac{2}{c^2} \geq \frac{3}{2\cdot c}$$

If $c \geq 4$. □

**Theorem 2.3** *The probability that the number of vertices at the end of a stage is less than $127n/128$, and that no edge belonging to $\mathcal{C}$ is contracted, is at least $\frac{1}{16}$.*

**Proof:** Partition the vertices into $n/4$ sets of four vertices each. Let $X$ be a random variable denoting the number of sets such that at least one vertex in the set is adjacent to some chosen edge at the end of a stage. It follows from Lemma 2.6 that $E[X]$ is at least $\frac{3}{2c}\cdot\frac{n}{4}$. We also know that $X \leq \frac{n}{4}$. Since,

$$\text{Prob}\left[X > \frac{1}{4c}\cdot\frac{n}{4}\right]\cdot\frac{n}{4} + \text{Prob}\left[X \leq \frac{1}{4c}\cdot\frac{n}{4}\right]\cdot\frac{n}{4c\cdot 4} \geq E[X] \geq \frac{3n}{8c}$$

We have that

$$\text{Prob}\left[X > \frac{n}{16c}\right] > \frac{5}{4c}$$

By Lemma 2.5, the probability that no edge belonging to $\mathcal{C}$ is chosen is at least $1 - \frac{1}{c}$. Hence, the probability that the two events intersect is at least $\frac{5}{4c} - \frac{1}{c} = \frac{1}{4c}$. If $X \geq \frac{n}{16c}$, we have that this many vertices have some adjacent edge chosen, and the number of vertices at the end of the stage is at most

$$n - \frac{n}{16c}\cdot\frac{1}{2} = n\left(1 - \frac{1}{32c}\right)$$

Fixing $c$ to be 4 gives us the desired result. □

## 3 Multiway cuts

In this section we show that the algorithm presented in the previous section and its analysis can be extended to compute minimum $k$-cuts where $k$ is fixed. Recall that a $k$-cut is a partition of the vertices into $k$ disjoint sets so as to minimize the weight of the edges in the cut. Let $\ell$ denote the weight of the minimum $k$-cut and assume again that $\ell$ is known. The algorithm is the same as the one presented in Section 2, except that we stop when there are $k$ vertices left in the graph. Fix a particular minimum $k$-cut $\mathcal{C}$. We prove that, with probability at least a constant, the number of vertices decreases by a constant factor, and no edge in $\mathcal{C}$ is chosen in a single stage.

Consider the vertices sorted by their degrees, $v_1, v_2, \ldots, v_n$, where the degree of vertex $v_i$ is $d_i$, and $d_i \leq d_{i+1}$. As in Section 2, we define the events $A_e$, $B_e$, and $B_v$ with respect to the $k$-cut $\mathcal{C}$. We generalize Lemma 2.1 for $k$-cuts.

**Lemma 3.1** *Let $\alpha < 1$. The graph $G$ contains at least $n - k\left(1 + \frac{2}{\alpha}\right)$ vertices, where each vertex is adjacent to at least $\frac{(1-\alpha)}{k}\ell$ edges that are not in the $k$-cut $\mathcal{C}$.*

**Proof:** We first show that,

$$\ell \leq \sum_{i=1}^{k-1} d_i$$

This follows by considering the $k$-cut obtained by placing each of the vertices $v_i$, $1 \leq i \leq k-1$, in a separate component, and placing the remaining vertices, $v_k, \ldots, v_n$ in the $k$th component. The cardinality of this $k$-cut is $\sum_{i=1}^{k-1} d_i$. Hence, for all $i$, $k \leq i \leq n$,

$$d_i \geq \frac{\ell}{k-1}$$

Let $U \subseteq V$ denote the set of vertices where each vertex is adjacent to at least $\alpha\ell/(k-1)$ edges in the $k$-cut $\mathcal{C}$. Then,

$$2\ell \geq |U| \cdot \frac{\alpha \cdot \ell}{k}$$

and hence,

$$|U| \leq \frac{2k}{\alpha}$$

Since, for all $i$, $k \leq i \leq n$, $d_i \geq \ell/k$, the graph contains at least $n - k\left(1 + \frac{2}{\alpha}\right)$ vertices, where each vertex is adjacent to at least $\frac{(1-\alpha)}{k} \cdot \ell$ edges that are not in the $k$-cut $\mathcal{C}$. □

Notice that Lemmas 2.2 and 2.3 hold for $k$-cuts too. That is,

**Lemma 3.2** *For an edge $e \notin \mathcal{C}$,*

$$\text{Prob}[B_e] \geq \frac{c-1}{c^2\ell}$$

**Lemma 3.3** *For any pair of edges $e_1, e_2 \notin \mathcal{C}$,*

$$\text{Prob}[B_{e_1} \cap B_{e_2}] \leq \frac{1}{c^2\ell^2}$$

However, Lemma 2.4 has to be slightly modified. We denote by $W$ the subset of $V$ for which Lemma 3.1 holds.

**Lemma 3.4** *Let $v \in W$.*

$$\text{Prob}\,[B_v] \geq \frac{(1-\alpha)(2kc - 2k - 1 + \alpha)}{2c^2k^2}$$

**Proof:** For a vertex $v \in W$, we define $E_v$ to be a subset of edges adjacent to $v$, that do not belong to $\mathcal{C}$, such that $|E_v| = \ell(1-\alpha)/k$. By the inclusion-exclusion formula,

$$\text{Prob}\,[B_v] \geq \text{Prob}\left[\bigcup_{e \in E_v} B_e\right] \geq \sum_{e \in E_v} \text{Prob}\,[B_e] - \sum_{e_1, e_2 \in E_v} \text{Prob}\,[B_{e_1} \cap B_{e_2}] \geq$$

$$\frac{\ell(1-\alpha)(c-1)}{kc^2\ell} - \binom{\frac{\ell(1-\alpha)}{k}}{2} \cdot \frac{1}{c^2\ell^2} \geq \frac{(1-\alpha)(c-1)}{kc^2} - \frac{(1-\alpha)^2}{2c^2k^2} =$$

$$\frac{(1-\alpha)(2kc - 2k - 1 + \alpha)}{2c^2k^2}$$

□

**Theorem 3.1** *The probability that the number of vertices at the end of a stage is less than $\frac{n(22k-2)}{22k-1}$ and that no edge belonging to $\mathcal{C}$ is contracted is at least $\frac{1}{22k}$.*

**Proof:** Let $X_v$ be a random variable defined to be the indicator variable of event $B_v$. (If $v \notin W$, then $X_v = 0$). The probability that event $B_v$ occurs is maximized when $c = (2k + 1 - \alpha)/k$. Choosing $\alpha = 0.1$, we get that for $v \in W$,

$$\text{Prob}[B_v] \geq \frac{0.9}{4k + 1.8}$$

By Lemma 3.1,

$$|W| \geq n - k\left(1 + \frac{2}{\alpha}\right) = n - 21k$$

(We assume that $n \geq 98k$. Otherwise, we can run any sequential algorithm to find the minimum $k$-cut in the graph.) Hence,

$$E\left[\sum_{v \in V} X_v\right] \geq \frac{9(n - 21k)}{40k + 18}$$

We define the random variable $Y$ as the number of vertices at the end of a stage. It is easy to see that $Y \leq n - \sum_{v \in V} X_v/2$. Thus,

$$E[Y] \leq n - E\left[\frac{\sum_{v \in V} X_v}{2}\right] = n - \frac{9(n - 21k)}{80k + 36} \leq \frac{n(11k - 1)}{11k} + 2$$

By Markov's inequality,

$$\text{Prob}[Y \geq \lambda E[Y]] \leq \frac{1}{\lambda}$$

Setting $\lambda = \frac{22k}{22k-1}$, we get that

$$\text{Prob}\left[Y \geq \left(\frac{n(11k-1)}{11k} + 2\right) \cdot \frac{22k}{22k-1}\right] \leq \frac{22k-1}{22k}$$

For $n \geq 98k$,

$$\frac{n(22k-2)}{22k-1} + \frac{44k}{22k-1} \leq \frac{n(40k-1)}{40k}$$

Hence, with probability at least $\frac{1}{22k}$, the number of vertices at the end of a stage is at most $\frac{n(40k-1)}{40k}$. $\square$

## 4 Implementation

We now justify the assumptions made in Section 2. Also, there are a few implementation details that need to be taken care of. By Theorem 2.1, the probability of success of the basic algorithm is $\Omega(1/n^d)$, where $d$ is a constant. We run $O(n^d)$ copies of the basic algorithm simultaneously to increase the probability of success to 1/2 and henceforth refer to these simultaneous runs as the algorithm.

The implementation issues for multiway cuts are similar and are therefore omitted from this abstract.

### 4.1 Finding the weight of the minimum cut

In the algorithm, we assumed that $\ell$, the cardinality, or weight, of the minimum cut, is known. In the unweighted case, $\ell$ is upper bounded by the minimum degree. In the weighted case, however, the feasible range of $\ell$ is much larger. We find the value of $\ell$ up to a multiplicative factor of 2. This is sufficient for us, since it may change the probability of success of a single stage of the basic algorithm by at most a constant factor.

Let $w_1, w_2, \ldots, w_m$ be the sequence of edge weights in sorted order and let the set $B$ be defined as follows:

$$B = \{p \cdot w_i | 1 \leq i \leq m,\ 1 \leq p \leq i\}$$

Let $w$ be the weight of the heaviest edge in the minimum cut and let $k$ denote the number of edges in the cut. We claim that the set $B$ contains at least one element such that the ratio between that element and $\ell$ is at most 2. This follows by observing that,

$$\exists i,\ 1 \leq i < k,\ w \cdot i \leq \ell \leq w \cdot (i+1)$$

The cardinality of the set $B$ is $O(m^2)$. We can therefore run the algorithm simultaneously for all the $|B|$ possible values of $\ell$. The output is the minimum cut obtained among all simultaneous runs of the algorithm.

### 4.2 The number of random bits

The next issue we elaborate on is the number of random bits used by the algorithm. In the basic algorithm we claimed that each edge $e$ is replaced by $w_e$ parallel edges. This is of course impractical if we want to maintain reasonable running times. Instead, in the basic algorithm, we choose to contract edge $e$ with probability $w_e/(c\ell)$, where the choices are pairwise independent. We claim that this can only increase the probability of success. For an edge $e$, let the random variable $y_i$, $1 \leq i \leq w_e$, be the indicator of the event that the $i$th parallel edge (substituted for $e$) is contracted by the basic algorithm in a particular stage. By the inclusion-exclusion formula,

$$\frac{w_e}{c\ell} - \frac{w_e^2}{c^2\ell^2} \leq \text{Prob}\left[\sum_{i=1}^{w_e} y_i > 0\right] \leq \frac{w_e}{c\ell}$$

Notice that the ratio between the lower and upper bound is at most 2, if $c > 2$. The following two observations guarantee that Theorem 2.3 still holds with the above change in the probability of contracting edges: (i) $1 - \frac{1}{c}$ remains a lower bound on the probability of *not* contracting an edge belonging to the minimum cut $\mathcal{C}$ (Lemma 2.5); (ii) for any edge $e \in E$, the probability of contracting it has increased, and therefore, Lemma 2.6 is still valid.

Another important issue is that since the probabilities may be very small, generating the events requires too many random bits. To overcome this problem, we set a threshold

$$\Delta = \frac{\ell}{mn^d \log n}$$

If the weight of edge $e$ is less than $\Delta$, then in the basic algorithm, the probability of contracting $e$ is set to zero. We claim that this cannot decrease the probability of success by much. In our analysis of the algorithm, we defined a successful stage as one in which: (i) An edge belonging to the minimum cut $\mathcal{C}$ was not contracted. (ii) The number of vertices decreased by a constant factor. Let $E' \subset E$ be the set of edges with weight less than $\Delta$. Setting the probability of contracting an edge $e \in E'$ to zero can only help (i). As for (ii), for $e \in E'$,

$$\text{Prob[edge } e \text{ is contracted in a stage]} \leq \frac{1}{cmn^d \log n}$$

The probability that an edge belonging to $E'$ is contracted in any of the stages in the basic algorithm is at most $r/(cn^d \log n)$, since $|E'| \leq m$ and $r$ is the number of stages. Setting $c \geq 2r/\log n$ guarantees that the probability of success of the basic algorithm is at least $1/2n^d$.

The minimum weight of an edge now is $\Delta$ and the maximum weight is $\ell$. (Recall that edges with weight greater than $\ell$ were contracted before running the algorithm). The ratio between them is bounded by a polynomial in $n$ and $m$. We set the weight of each edge $e \in E - E'$ to be $\lfloor w(e)/\Delta \rfloor$. We conclude with the next proposition that follows from [AS, pp. 228-232].

**Proposition 4.1** *The number of random bits needed to generate $m$ $\{0,1\}$ random variables is $O(\log n)$ if: (i) the variables are pairwise independent; (ii) the probability that each variable is "1" (or "0") is not less than $\frac{1}{\text{poly}(n)}$.*

The last issue we address is how to generate the random bits for the many simultaneous runs of the basic algorithm. Recall that there are two reasons for simultaneous runs. The first one is to decrease the probability of error. However, to decrease the probability of error to a constant it suffices to use the method of two point sampling of Chor and Goldreich [CG]. Generating random bits using this method guarantees that runs of the basic algorithm are pairwise independent. The number of random bits needed for implementing two point sampling remains $O(\log^2 n)$. The second reason for simultaneous runs is testing all the values in the set $B$. We note that the same random bits can be used for all such runs.

### 4.3 Processor and time bounds

Our parallel model of computation is the CRCW PRAM. A stage in the basic algorithm can be implemented using $O(m)$ processors and $O(\log n)$ time, implying that the time bound for the algorithm is $O(\log^2 n)$. The total number of processors is $O(n^d \cdot m^2 \cdot m) = O(n^d \cdot m^3)$ to account for all the simulataneous runs.

## References

[Al] N. Alon, Generating pseudo-random permutations and maximum flow algorithms, Information Processing Letters, Vol. 35, pp. 201–204, 1990.

[AS] N. Alon and J. Spencer, **The probabilistic method**, John Wiley and Sons Inc., New York, 1992.

[CG] B. Chor and O. Goldreich, On the power of two point sampling, Journal of Complexity, Vol. 5, pp. 96-106, 1989.

[CH] J. Cheriyan and T. Hagerup, A randomized maximum flow algorithm, Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, 1988, pp. 118-123.

[CHM] J. Cheriyan, T. Hagerup and S. N. Maheshwari, Can a maximum flow be computed in $o(mn)$ time?, Proceedings of International Colloquium on Automata, Languages and Programming, 1990.

[Ga] H. N. Gabow, Applications of a Poset Representation to Edge Connectivity and Graph Rigidity, Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1991, pp. 812-821.

[GH1] R.E. Gomory, T.C. Hu, Multi-Terminal Network Flows, Siam J. Appl. Math., Vol. 9, pp. 551-560, 1961.

[GH2] O. Goldschmidt and D. S. Hochbaum, Polynomial algorithm for the $k$-cut problem, Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, 1988, pp. 444-451.

[GJ] M. R. Garey and D. S. Johnson, **Computers and intractability - a guide to the theory of NP-completeness**, W. H. Freeman, San Francisco, 1979.

[GSS] L. Goldschlager, R. Shaw and J. Staples, The maximum flow problem is log space complete for P, Theoretical Computer Science, Vol. 21, pp. 105-111, 1982.

[GT] A. Goldberg and R. E. Tarjan, A new approach to the maximum flow problem, Journal of the ACM, Vol. 35, pp. 921-940, 1988.

[HO] J. Hao and J. B. Orlin, A faster algorithm for finding the minimum cut in a graph, Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, 1992, pp. 165-174.

[Ka] D. R. Karger, Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm, Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms, 1993.

[KRT] V. King, S. Rao and R. Tarjan, A faster deterministic maximum flow algorithm, Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, 1992, pp. 157-164.

[KS] D. R. Karger and C. Stein, An $\tilde{O}(n^2)$ algorithm for minimum cuts, To appear in: Proceeding of the 25th ACM Annual Symposium on Theory of Computing, 1993.

[KUW] R. M. Karp, E. Upfal and A. Wigderson, Constructing a perfect matching is in random NC, Combinatorica, Vol. 6, pp. 35-48, 1986.

[Ma] D. W. Matula, Determining the edge connectivity in $O(nm)$, Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science, Los Angeles, CA, 1987, pp. 249-251.

[MVV] K. Mulmuley, U. V. Vazirani and V. V. Vazirani, Matching is as easy as matrix inversion, Combinatorica, Vol. 7, pp. 105-113 (1987).

[NI] H. Nagamochi and T. Ibaraki, Computing edge-connectivity in multigraphs and capacitated graphs, Siam Journal on Discrete Math, Vol. 5, pp. 54-66 (1992).

[Po] V.D. Podderyugin, An Algorithm for Finding the Edge Connectivity of Graphs, Vopr. Kibern., No. 2, 136, 1973.

[PQ] J. C. Picard and M. Querayne, Selected applications of minimum cuts in networks, INFOR, Vol. 20, pp. 394-422, 1982.

[PW] S. Phillips and J. Westbrook, Online load balancing and network flow, To appear in: Proceeding of the 25th ACM Annual Symposium on Theory of Computing, 1993.