Beyond the Quadratic Time Barrier for Network Unreliability

Ruoxu Cen[*]   William He[†]   Jason Li[‡]   Debmalya Panigrahi[§]

Abstract

Karger (STOC 1995) gave the first FPTAS for the network (un)reliability problem, setting in motion research over the next three decades that obtained increasingly faster running times, eventually leading to a $\tilde{O}(n^2)$-time algorithm (Karger, STOC 2020). This represented a natural culmination of this line of work because the algorithmic techniques used can enumerate $\Theta(n^2)$ (near-)minimum cuts. In this paper, we go beyond this quadratic barrier and obtain a faster FPTAS for the network unreliability problem. Our algorithm runs in $m^{1+o(1)} + \tilde{O}(n^{1.5})$ time.

Our main contribution is a new estimator for network unreliability in *very* reliable graphs. These graphs are usually the bottleneck for network unreliability since the disconnection event is elusive. Our estimator is obtained by defining an appropriate importance sampling subroutine on a dual spanning tree packing of the graph. To complement this estimator for very reliable graphs, we use recursive contraction for *moderately* reliable graphs. We show that an interleaving of sparsification and contraction can be used to obtain a better parametrization of the recursive contraction algorithm that yields a faster running time matching the one obtained for the very reliable case.

1 Introduction

In the network unreliability problem, we are given an undirected, unweighted graph $G = (V, E)$ and a failure probability $0 < p < 1$. The goal is to find the probability that the graph disconnects when every edge is deleted independently with probability p. The probability of disconnection is called the unreliability of the graph and denoted $u_G(p)$. Reliability problems naturally arise from the need to keep real-world networks connected under edge failures, and entire books have been devoted to the topic [4, 3]. In particular, the problem of estimating $u_G(p)$ has been dubbed as "...perhaps the most fundamental problem in the study of network reliability"[14].

In 1979, Valiant showed that the network unreliability problem is #P-hard [19], which implies that it is unlikely that a polynomial-time algorithm can *exactly* compute the value of $u_G(p)$. Over the next 15 years, several algorithms were designed to approximate $u_G(p)$ for various special cases such as planar graphs (Karp and Luby [17]), dense graphs (Alon, Frieze, and Welsh [2]), etc. Eventually, in a seminal work, Karger [10] proposed the first fully polynomial-time randomized approximation scheme (FPRAS) for the unreliability problem. For any constant $\varepsilon \in (0, 1)$, this algorithm outputs a $(1 \pm \varepsilon)$-approximation for $u_G(p)$ with high probability (whp)[1] in $\tilde{O}(mn^4)$ time, where n is the number of vertices in the graph and m is the number of edges. This work established a bifurcation of instances of the problem into two cases depending on the value of $u_G(p)$. If $u_G(p)$ is *large* (the unreliable case), then Monte Carlo sampling suffices. On the other hand, if $u_G(p)$ is *small* (the reliable case), then Karger showed that whenever the graph disconnects, it's almost always the case that a *near-minimum* cut fails. In this case, the algorithm uses DNF counting [17] on the (polynomial) set of near-minimum cuts to obtain an estimate of $u_G(p)$. Using the same template, Harris and Srinivasan [8] improved the running time of the algorithm to $n^{3+o(1)}$. They did so by establishing a tighter bound on the number of cuts that contribute to $u_G(p)$ and showing that the instances of DNF counting generated by the unreliability problem are nongeneric and admit faster algorithms than the generic case.

In the last decade, Karger [12, 13, 14] has further improved the running time of the problem to the current best bound of $\tilde{O}(n^2)$. As in prior works, the unreliable case is handled by Monte Carlo sampling. But, for the

---

[*]Department of Computer Science, Duke University. Email: `ruoxu.cen@duke.edu`

[†]Computer Science Department, Carnegie Mellon University. Email: `wrhe@andrew.cmu.edu`. This work was done while the author was an undergraduate student at Duke University.

[‡]Simons Institute for the Theory of Computing, UC Berkeley. Email: `jmli@alumni.cmu.edu`

[§]Department of Computer Science, Duke University. Email: `debmalya@cs.duke.edu`

[1]In this paper, as in the network unreliability literature, a result is said to hold with high probability if it holds with probability $1 - \frac{1}{\text{poly}(n)}$.

reliable case, instead of DNF counting, every edge is failed with some probability $q \geq p$ and all edges that survive are contracted to yield a smaller graph H. On this smaller graph H, the algorithm computes $u_H(p/q)$, which is an unbiased estimator of $u_G(p)$. Now, if the relative variance of $u_H(p/q)$ is bounded by η, then a standard technique of repeating $\tilde{O}(\eta\varepsilon^{-2})$ times and taking the *median-of-averages* yields a $(1 \pm \varepsilon)$-approximation to $u_G(p)$ whp.

The algorithms in [12, 13, 14] differ in the choice of q, the structure of the computation tree, and the bound on relative variance.[2] In [12], q is chosen so that H is expected to be of constant size, and the resulting estimator has a relative variance of $O(n^2)$ and can be computed in $\tilde{O}(n)$ time. This yields a running time of $\tilde{O}(n^3)$. In [13], q is chosen differently, so as to ensure that the expected size of H is a constant fraction of G. The algorithm then proceeds to recursively estimate $u_H(p/q)$, closely resembling the recursive contraction framework of Karger and Stein [16] originally proposed for the minimum cut problem. The bound on the running time of this algorithm was initially established as $\tilde{O}(n^{2.71})$ [13], but a more refined analysis of the relative variance later improved this to $\tilde{O}(n^2)$ [14]. This remained the fastest algorithm for the network unreliability problem prior to our work.

1.1 Our Contribution In this paper, we give a fully polynomial-time randomized approximation scheme (FPRAS) for the network unreliability problem that has a running time of $m^{1+o(1)} + \tilde{O}(n^{1.5})$. We show the following theorem:

THEOREM 1.1. *For any $\varepsilon \in (0, 1)$, there is a randomized Monte Carlo algorithm for the network unreliability problem that runs in $m^{1+o(1)}\varepsilon^{-2} + \tilde{O}(n^{1.5}\varepsilon^{-3})$ time and outputs a $(1 \pm \varepsilon)$-approximation to $u_G(p)$ whp.*

We remark that the dependence on m, which is not explicit in prior bounds because the bounds are $\Omega(n^2)$ and therefore $\Omega(m)$, is necessary because the number of edges may exceed $n^{1.5}$ in general.

The running time bound of $\tilde{O}(n^2)$ obtained by [14] was a natural culmination of prior research on the unreliability problem because all prior algorithms involve explicit or implicit enumeration of a set of potentially $\Omega(n^2)$ (near-)minimum cuts of the graph. To see this, consider the cycle graph on n vertices. This is a sparse graph of n edges, and one would ideally like a subquadratic running time for the unreliability problem. However, the cycle has $\binom{n}{2} = \Theta(n^2)$ minimum cuts, which means that any subquadratic algorithm must refrain from enumerating the minimum cuts of the graph. The prior techniques for the unreliability problem are unsuitable for this purpose. The DNF counting-based algorithms [9, 8] perform explicit enumeration of near-minimum cuts and the DNF formula is already of $\Omega(n^2)$ size. The later algorithms based on recursive contraction [12, 13, 14] do not perform explicit enumeration, but the framework can be used to generate all the (possibly $\Theta(n^2)$) minimum cuts of a graph. Our main conceptual contribution in this work is to overcome this bottleneck of $\Omega(n^2)$ in the running time of unreliability algorithms. We outline the main techniques that we employ for this purpose next.

1.2 Our Techniques As in prior work, in the *unreliable* case, i.e., when $u_G(p)$ is above some threshold $n^{-o(1)}$, we apply naïve Monte Carlo sampling. The algorithm removes each edge independently with probability p and checks whether the remaining graph is connected. Each sample takes $O(m)$ time and the estimator has relative variance $\approx \frac{1}{u_G(p)}$. Thus, $\tilde{O}\left(\frac{1}{u_G(p)}\right) = n^{o(1)}$ samples are sufficient and the running time of the algorithm is $m^{1+o(1)}$.

When $u_G(p)$ is too small for Monte Carlo sampling, Karger [14] proved that the partition function $z_G(p)$, defined as the expected number of failed cuts, approximates $u_G(p)$ up to a factor of $1 + o(1)$. So, it suffices to estimate $z_G(p)$ in this case instead of $u_G(p)$.

Let us first review the recursive contraction algorithm in [14] for estimating $z_G(p)$. The algorithm contracts each edge independently with probability $1 - q$ for some parameter $q > p$ to form a smaller graph H, and estimates $u_H(p/q)$ in this graph recursively. The parameter q is chosen so that $q^{-\lambda} = 2$ (or any constant), where λ is the value of the minimum cut in the graph. This ensures that one step of contraction reduces the number of vertices by a constant fraction in expectation. The relative variance is $q^{-\lambda}(1 + o(1))$. By repeating $q^{-\lambda}$ times, the relative variance is kept at $1 + o(1)$. This yields the recurrence: $T(n) = q^{-\lambda} \cdot T(q^{\lambda/2} \cdot n)$. Solving this recurrence gives $T(n) = O(n^2)$.

In this recurrence, the size bound and relative variance can both be simultaneously tight. To see this, consider a cycle graph with $\frac{\lambda}{2} + 1$ parallel edges where one vertex on the cycle is connected to a leaf vertex with λ parallel

---

[2]The relative variance of a random variable is the ratio of its variance to the square of its expectation.

edges. Suppose p is small enough that the minimum cut dominates unreliability, i.e., $z_G(p) \approx p^\lambda$. The estimator (approximately) returns $(p/q)^\lambda$ with probability q^λ; hence, the relative variance is $\frac{1}{z_G(p)^2} \cdot q^\lambda \cdot \left(\frac{p}{q}\right)^{2\lambda} \approx q^{-\lambda}$. The expected size of the contracted graph is $\approx q^{\lambda/2+1} \cdot n$. This example shows that we cannot hope to unconditionally improve the previous analysis to obtain a sub-quadratic bound.

But, under what conditions can the pathological example above be realized? To answer this question, we partition the reliable case into two subcases depending on whether $u_G(p)$ exceeds $O(n^{-3})$. We call these the *moderately reliable* and *very reliable* cases respectively. Clearly, the example above can be realized in the very reliable case; hence, we need a new algorithm for this case. This algorithm is our main contribution, and we will describe it below. But, before doing so, let us consider the moderately reliable setting. In this case, our main observation is that the two bounds on graph size and relative variance in the recurrence relation for recursive contraction cannot be tight simultaneously. Consider the following two typical cases. In a cycle with $\frac{\lambda}{2}$ parallel edges, the size bound $q^{\lambda/2} \cdot n$ is tight, but the true relative variance is close to 1, which is much smaller than the $q^{-\lambda}$ bound. In contrast, in a graph with a single minimum cut where all other cuts are large (e.g., a dumbbell graph), the relative variance is close to $q^{-\lambda}$ but the size decreases much faster than $q^{\lambda/2} \cdot n$. In general, we interpolate between these two extreme cases. We choose a parameter $\gamma \in \left[\frac{3}{4}\lambda, \lambda\right]$, and prove sharper bounds of $q^{-\gamma}$ on relative variance and $q^{\frac{2\gamma}{3}} \cdot n$ on the size of the recursive instance. Then the recurrence becomes $T(n) = q^{-\gamma} \cdot T\left(q^{\frac{2\gamma}{3}} \cdot n\right)$, which yields a running time of $O(n^{1.5})$ as desired.

Finally, we consider the very reliable case, i.e., when $u_G(p)$ is smaller than $O(n^{-3})$. In this case, we design a new algorithm for estimating $z_G(p)$. When p is small, the contribution of large cuts to the value of $z_G(p)$ decreases. In particular, for $u_G(p) = O(n^{-3})$, we can discard all cuts of value $> 3.5\lambda$. Our goal, then, is to obtain an estimator that performs *importance sampling* on cuts of value $\leq 3.5\lambda$ (i.e., reweights them appropriately) to yield $z_G(p)$. But, how do we sample such cuts? We use a semi-duality between (near-)minimum cuts and maximum spanning tree packings for this purpose. Namely, one can construct λ spanning trees with edge congestion 2, which implies that in a randomly chosen tree, a cut of value $\leq \alpha\lambda$ only has $\leq 2\alpha$ edges, i.e., 2α-*respects* the tree, in expectation. (See Gabow [5] and Karger [11] for prior uses of such duality in minimum cut algorithms.) This allows us to sample 7-respecting cuts [3] from a spanning tree instead of cuts of value $\leq 3.5\lambda$, and redefine the support of the estimator to 7-respecting cuts. There are two main challenges: first, we need to implement the importance sampling subroutine very efficiently, namely calculate each sample and its corresponding weight in $\tilde{O}(1)$ time, and second, we must control the relative variance to $\tilde{O}(n^{1.5})$ since we can only draw $\tilde{O}(n^{1.5})$ samples (note that the support of the distribution is $\approx n^7$). We design a data structure based on orthogonal range queries in $\mathbb{R}^2$ to implement importance sampling efficiently. To control the relative variance, we contract well-connected components of the graph (using a Gomory-Hu tree [6]) and apply the sampling subroutine on this contracted graph to boost the sampling probability of the small cuts.

1.3 Overall Algorithm and Paper Organization We now give an overview of the algorithm. We describe when the algorithm invokes its different components, and give a pointer to the section where each component is described and analyzed.

Our overall algorithm builds a recursive computation tree. There are several base cases of this recursion that we describe first.

- The first base case is determined simply by the number of vertices in the graph. If $n \leq \tilde{O}(\varepsilon^{-2})$, then we run Karger's algorithm [14] that gives an unbiased estimator for $u_G(p)$. The following theorem states the properties of this estimator:

  THEOREM 1.2. ([14]) *Given a graph G with vertex size n, an unbiased estimator of $u_G(p)$ with $O(1)$ relative variance can be computed in $\tilde{O}(n^2)$ time. As a consequence, a $(1\pm\varepsilon)$-approximation to $u_G(p)$ can be computed in time $\tilde{O}(n^2\varepsilon^{-2})$.*

- The second base case is to run Monte Carlo sampling. Intuitively, this is done when the probability of the graph being disconnected is large. There are two subcases for Monte Carlo sampling:

---

[3]We say a cut k-respects a tree if at most k tree edges cross the cut.

– The first subcase is when $p > \theta$ for some threshold θ whose value will be given by Lemma 2.1. In this case, we run a naïve Monte Carlo algorithm to obtain an estimator for $u_G(p)$ (Section 6.1). The properties of the estimator are summarized below:

LEMMA 1.1. *For any $p \geq \theta$, an unbiased estimator of $u_G(p)$ with relative variance $O(1)$ can be computed in time $m^{1+o(1)}$. As a consequence, a $(1 \pm \varepsilon)$-approximation to $u_G(p)$ can be computed in $m^{1+o(1)}\varepsilon^{-2}$ time under the condition that $p \geq \theta$.*

– The second subcase is when $p < \theta$ but $p^\lambda > n^{-0.5}$. In this case, we run a *two-step* Monte Carlo algorithm to obtain an estimator for $u_G(p)$ (Section 6.2). The properties of the estimator are summarized below:

LEMMA 1.2. *For p such that $p < \theta$ and $p^\lambda > n^{-1/2}$, an unbiased estimator of $u_G(p)$ with relative variance $O(1)$ can be computed in $\tilde{O}(m + n^{1.5})$ time. As a consequence, a $(1 \pm \varepsilon)$-approximation to $u_G(p)$ can be computed in $\tilde{O}((m + n^{1.5})\varepsilon^{-2})$ time under the condition that $p < \theta$ and $p^\lambda > n^{-1/2}$.*

- The final base case is the most interesting new contribution of this paper. This is invoked in the highly reliable setting, when $p^\lambda < 4n^{-3}$. In this case, we run an importance sampling algorithm on a spanning tree packing of the graph (Section 3). We prove the following lemma (an estimator X for $u_G(p)$ with relative bias δ satisfies $\mathbb{E}[X] \in (1 \pm \delta)u_G(p)$):

LEMMA 1.3. *For any p such that $p^\lambda \leq O(n^{-3})$, an estimator for $u_G(p)$ with relative bias $O\left(\frac{\log n}{\sqrt{n}}\right)$ and relative variance $O(1)$ can be computed in $m^{1+o(1)} + \tilde{O}(n^{1.5})$ time. As a consequence, a $(1 \pm \varepsilon)$-approximation to $u_G(p)$ can be computed in $(m^{1+o(1)} + \tilde{O}(n^{1.5}))\varepsilon^{-2}$ time under the condition that $p^\lambda \leq O(n^{-3})$.*

We have described the base cases, all of which are non-recursive algorithms. The remaining case is when $4n^{-3} \leq p^\lambda \leq n^{-0.5}$ and $p < \theta$. In this case, we run a step of recursive contraction (Section 4). In earlier works using recursive contraction, the number of edges is trivially bounded by $O(n^2)$ in recursive calls. Since we would like to impose the stricter bound of $\tilde{O}(n^{1.5})$ on the running time, we need the number of edges to also satisfy this bound. But, this may not hold in general in recursive steps. To restore this property, we occasionally interleave calls to a standard sparsification algorithm with the recursive contraction steps. This increases variance – we bound it in Section 5 and use this bound in the analysis of recursive contraction in Section 4. We obtain the following lemma (the relative second moment of a random variable X is defined as $\mathbb{E}[X^2]/(\mathbb{E}[X])^2$):

LEMMA 1.4. *Suppose $4n^{-3} \leq p^\lambda \leq n^{-0.5}$ and $p < \theta$. An estimator X for $u_G(p)$ with relative bias $\leq 0.1\varepsilon$ and relative second moment $\leq \log^{O(1)} n$ can be computed in $m^{1+o(1)} + \tilde{O}(n^{1.5}\varepsilon^{-1})$ time.*

We now show that Theorem 1.1 follows from this lemma:

Proof. [Proof of Theorem 1.1] The base cases are immediate from Theorem 1.2 and Lemmas 1.1 to 1.3. In the first base case, the running time is $\tilde{O}(n^2\varepsilon^{-2}) = \tilde{O}(n^{1.5}\varepsilon^{-3})$ by the assumption $n \leq \tilde{O}(\varepsilon^{-2})$; the running times in other base cases are all bounded by $m^{1+o(1)}\varepsilon^{-2} + \tilde{O}(n^{1.5}\varepsilon^{-3})$. So, we focus on the recursive contraction case. Let X be the estimator output by the recursive contraction algorithm. From Lemma 1.4, we have its relative variance $\eta[X] \leq \tilde{O}(1)$. By standard techniques (see Lemma 2.6), we can run the algorithm $\tilde{O}(\varepsilon^{-2})$ times to get a $(1 \pm \frac{\varepsilon}{2})$-approximation of $\mathbb{E}[X]$ whp. Because $\mathbb{E}[X]$ is a $(1 \pm 0.1\varepsilon)$-approximation of $u_G(p)$, the aggregated estimator is a $(1 \pm \varepsilon)$-approximation of $u_G(p)$. Each run takes $m^{1+o(1)} + \tilde{O}(n^{1.5}\varepsilon^{-1})$ time, so the overall running time is $m^{1+o(1)}\varepsilon^{-2} + \tilde{O}(n^{1.5}\varepsilon^{-3})$. ☐

Finally, we describe how the algorithm decides which case it is in at any node of the computation tree. The first base case can be identified based on the size of the graph. If not in the first base case, we calculate the value of the minimum cut λ in $\tilde{O}(m)$ time [11]. If $p^\lambda < 4n^{-3}$, we are in the third base case. We are left to identify the two Monte Carlo base cases. We need the value of θ for this determination. Unfortunately, we do not know of a way to efficiently calculate θ. Therefore, we distinguish these cases indirectly. We first run the naïve Monte Carlo algorithm and calculate the estimator of $u_G(p)$ given by this algorithm (which is the empirical probability of disconnection). If the value of this estimator is at least $n^{-o(1)}$, then we can conclude that the estimator $(1 \pm \varepsilon)$-approximates $u_G(p)$ whp. If the estimator returns a smaller value, then we are in the case $p < \theta$. In this case, we calculate the value of p^λ and depending on whether it exceeds $n^{-0.5}$, we either run the two-step Monte Carlo algorithm or a step of recursive contraction.

2   Preliminaries

We give some known observations in this section that we use throughout the paper.

Phase Transition. An important observation due to Karger [14] is the so called *phase transition* property of network unreliability. Roughly speaking, this property says that there is a threshold θ on the value of p such that (a) above this threshold, a naïve Monte Carlo algorithm is efficient, and (b) below this threshold, conditioned on the graph getting disconnected, the probability that a single cut fails is close to 1. We state this property precisely below (Lemma 2.1).

Let $\mathcal{C}(G)$ be the family of cuts in G, where a cut is represented by the set of cut edges. Let $z_G(p)$ be the number of failed cuts, and $x_G(p)$ be the number of failed cut pairs. By linearity of expectation,

$$z_G(p) = \sum_{C_i \in \mathcal{C}(G)} p^{|C_i|} \quad \text{and} \quad x_G(p) = \sum_{C_i, C_j \in \mathcal{C}(G), C_i \neq C_j} p^{|C_i \cup C_j|}.$$

When context is clear, we write $z = z_G(p)$ and we omit the index range $\mathcal{C}(G)$ from the sums.

We state the following phase transition lemma:

LEMMA 2.1. *There exists a threshold θ such that*

1. $u_G(\theta) = n^{-O(1/\log\log n)}$.

2. *When $p < \theta$, we have $\frac{x_G(p)}{z_G(p)} \leq \frac{1}{\log n}$; therefore, $\left(1 - \frac{1}{\log n}\right) z_G(p) \leq u_G(p) \leq z_G(p)$.*[4]

In the very reliable case, i.e, $p^\lambda < O(n^{-3})$, we can show a tighter bound for approximating u with z:

LEMMA 2.2. *When $p^\lambda < O(n^{-3})$, we have $\frac{x_G(p)}{z_G(p)} \leq O\left(\frac{\log n}{\sqrt{n}}\right)$; therefore,*

$$\left(1 - O\left(\frac{\log n}{\sqrt{n}}\right)\right) z_G(p) \leq u_G(p) \leq z_G(p).$$

The bounds in above two lemmas easily follow from previously known bounds in [14]. We give proof of the lemmas for completeness in Appendix A.

Random Contraction, Sparsification, and Near-Minimum Cuts. Next, we give some standard results in graph connectivity related to random contractions, counting near-minimum cuts, and graph sparsification. First is a standard bound on the number of near-minimum cuts. Let λ be the value of a minimum cut. Let the value of a cut C_i be denoted c_i. We call a cut C_i d-strong if $c_i \geq d$ and d-weak if $c_i \leq d$.

LEMMA 2.3. (LEMMA 3.2 OF [11]) *The number of $\alpha\lambda$-weak cuts in a graph with minimum cut value λ is at most $n^{\lfloor 2\alpha \rfloor}$ for any $\alpha \geq 1$.*

We also use the following standard sparsification result:

LEMMA 2.4. (COROLLARY 2.4 OF [9]) *Given an unweighted undirected graph G with min-cut value λ and any parameter $\delta \in (0,1)$, there exists $\alpha = O\left(\frac{\log n}{\delta^2 \lambda}\right)$ such that if a subgraph H is formed by picking each edge independently with probability α in G, then the following holds whp: for every cut S, its value in H (denoted $d_H(S)$) and its value in G (denoted $d_G(S)$) are related by $d_H(S) \in [(1-\delta) \cdot \alpha \cdot d_G(S), (1+\delta) \cdot \alpha \cdot d_G(S)]$. Note that this implies that the min-cut value in H is $\lambda_H = O(\log n / \delta^2)$.*

The running time of the sparsification algorithm is $O(m)$.

Finally, we state a bound on the expected number of uncontracted edges after random edge contractions. This was previously used (and proved) in the celebrated linear-time randomized MST algorithm of Karger, Klein, and Tarjan.

LEMMA 2.5. (LEMMA 2.1 OF [15]) *Given an undirected multigraph, if we contract each edge independently with probability π, then the expected number of uncontracted edges is at most n/π.*

---

[4]In this paper, all logarithms are with base 2 unless otherwise mentioned.

Gomory-Hu Tree. Next, we recall the definition of a Gomory-Hu tree [6], which is used in various parts of the paper:

DEFINITION 2.1. *The Gomory-Hu tree [6] of an undirected graph $G = (V, E)$ is a (weighted) tree Y on the same set of vertices V such that for every pair of vertices $s, t \in V$, the minimum (s, t)-cut in Y (which is simply the minimum weight edge on the unique $s - t$ path in Y) is also a minimum (s, t)-cut in G, and has the same value.*

Relative Variance and Relative Bias. The *relative variance* of our estimators will play an important role in our analysis. We define this below and state some standard properties.

DEFINITION 2.2. *The* relative variance *of a random variable X, denoted $\eta[X]$, is defined as the ratio of its variance and its squared expectation, i.e., $\eta[X] = \frac{\mathbb{V}[X]}{\mathbb{E}^2[X]} = \frac{\mathbb{E}[X^2]}{\mathbb{E}^2[X]} - 1$. We also define* relative second moment *of X as $\frac{\mathbb{E}[X^2]}{\mathbb{E}^2[X]} = \eta[X] + 1$.*

Similar to variance, relative variance can be decreased by taking multiple independent samples.

FACT 2.1. (LEMMA I.4 OF [13]) *The average of N independent samples of X has relative variance $\frac{\eta[X]}{N}$.*

This leads to the following property:

LEMMA 2.6. (LEMMA I.2 OF [13]) *Fix any $\varepsilon, \delta \in (0, 1)$. For a random variable X with relative variance $\eta[X]$, the median of $O\left(\log \frac{1}{\delta}\right)$ averages of $O\left(\frac{\eta[X]}{\varepsilon^2}\right)$ independent samples of X is a $(1 \pm \varepsilon)$-approximate estimation of $\mathbb{E}[X]$ with probability $1 - \delta$.*

The next lemma is an important property of relative variance that allows us to compose estimators in a recursive algorithm.

LEMMA 2.7. (LEMMA II.3 OF [13]) *Suppose Y is an unbiased estimator of x with relative second moment V_1, and conditioned on a fixed Y, Z is an unbiased estimator of Y with relative second moment V_2. Then Z is an unbiased estimator for x with relative second moment $V_1 V_2$.*

We also define the *relative bias* of an estimator:

DEFINITION 2.3. *The* relative bias *of an estimator X for a value x is defined as $\frac{|\mathbb{E}[X] - x|}{x}$.*

3 Importance Sampling on a Spanning Tree Packing

As stated earlier, our main new algorithmic contribution is an estimator for $u_G(p)$ in the very reliable case, i.e., $p^\lambda \leq O(n^{-3})$. Our goal in this section is to prove Lemma 1.3, which we restate below:

LEMMA 1.3. *For any p such that $p^\lambda \leq O(n^{-3})$, an estimator for $u_G(p)$ with relative bias $O\left(\frac{\log n}{\sqrt{n}}\right)$ and relative variance $O(1)$ can be computed in $m^{1+o(1)} + \tilde{O}(n^{1.5})$ time. As a consequence, a $(1 \pm \varepsilon)$-approximation to $u_G(p)$ can be computed in $(m^{1+o(1)} + \tilde{O}(n^{1.5}))\varepsilon^{-2}$ time under the condition that $p^\lambda \leq O(n^{-3})$.*

We design an importance sampling algorithm that gives an estimator X of $u_G(p)$ with a relative bias of $O(\log n / \sqrt{n})$ in $m^{1+o(1)} + \tilde{O}(n^{1.5})$ time. Since $n > \tilde{\Omega}(\varepsilon^{-2})$ (otherwise we are in the first base case), the relative bias is at most $\varepsilon/3$. It follows by Lemma 2.6 that $\tilde{O}(\varepsilon^{-2})$ runs of the algorithm gives a $(1 \pm \varepsilon/3)$-approximation of $\mathbb{E}[X]$, which in turn is a $(1 \pm \varepsilon/3)^2 \in (1 \pm \varepsilon)$-approximation of $u_G(p)$.

3.1 Dependence of $z_G(p)$ on Near-Minimum Cuts By Lemma 2.2, since $p^\lambda \leq O(n^{-3})$, we have

$$\left(1 - O\left(\frac{\log n}{\sqrt{n}}\right)\right) z_G(p) \leq u_G(p) \leq z_G(p).$$

This allows us to focus on approximating $z_G(p)$ instead of $u_G(p)$.

Intuitively, when p^λ is small, the value of $z_G(p)$ only depends on the near-minimum cuts because larger cuts scarcely fail. We make this intuition formal below. Let M_k be the number of cuts of value k. Then, $z_G(p)$ is defined as

$$z_G(p) = \sum_{k \geq \lambda} M_k \cdot p^k.$$

By cut counting (Lemma 2.3), we know that $M_k \leq n^{2k/\lambda}$. The assumption $p^\lambda \leq O(n^{-3})$ implies $p^k \leq O(n^{-3k/\lambda})$ for all $k \geq \lambda$. So, p^k decreases much faster than the increase in M_k as k increases; as a consequence, the product $M_k \cdot p^k$ decays rapidly. The next lemma shows that we can truncate the sum $\sum_{k \geq \lambda} M_k \cdot p^k$ at $k = 3.5\lambda$ without significantly distorting its value. To state the lemma, let us define the partial sums:

$$z^{\geq \alpha} = \sum_{C_i : c_i \geq \alpha\lambda} p^{c_i} \qquad \text{and} \qquad z^{\leq \alpha} = \sum_{C_i : c_i \leq \alpha\lambda} p^{c_i}.$$

LEMMA 3.1. *If $p^\lambda \leq O(n^{-3})$, then $z^{\geq 3.5} \leq O(n^{-0.5}) \cdot z_G(p)$, where $z^{\geq 3.5}$ is the expected number of failed cuts of value at least 3.5λ.*

Proof. By Lemma 2.3, there are at most $n^{\lfloor 2\alpha \rfloor}$ $\alpha\lambda$-weak cuts. So for each integer $k \geq 7$, the expected number of failed cuts with cut value in $\left[\frac{k}{2} \cdot \lambda, \frac{k+1}{2} \cdot \lambda \right)$ is at most $n^k \cdot p^{k\lambda/2}$. Therefore,

$$\frac{z^{\geq 3.5}}{p^\lambda} \leq \frac{1}{p^\lambda} \sum_{k \geq 7} n^k \cdot p^{k\lambda/2} \leq n^7 \cdot p^{2.5\lambda} \cdot \left(\sum_{k \geq 0} n^k \cdot p^{k\lambda/2} \right) = O\left(n^7 \cdot p^{2.5\lambda} \right) = O(n^{-0.5})$$

since $p^\lambda = O(n^{-3})$. Therefore $z^{\geq 3.5} = O(n^{-0.5}) \cdot p^\lambda \leq O(n^{-0.5}) \cdot z_G(p)$. □

3.2 Algorithm for Estimating $z_G(p)$ for 3.5λ-weak Cuts Lemma 3.1 implies that for the purpose of getting a $(1 \pm \varepsilon)$-approximation of $z_G(p)$, it suffices to only consider cuts of value $\alpha\lambda$ for $1 \leq \alpha \leq 3.5$. We first give the high level idea for estimating the contribution of these cuts to $z_G(p)$. Suppose we choose a spanning tree of the graph uniformly at random from a collection of $\frac{\lambda}{2}$ edge-disjoint spanning trees. By averaging, we expect to see at most 7 edges from a 3.5λ-weak cut in this randomly chosen tree. We don't know how to sample a 3.5λ-weak cut directly, but we can sample a cut whose projection on the spanning tree contains at most 7 edges. Therefore, we can write an estimator that reweights these cuts appropriately to obtain an unbiased sample of $z_G(p)$ restricted to the 3.5λ-weak cuts. This is the *importance sampling* problem that we solve below.

There are two main challenges in this problem. First, even if we had access to a uniform distribution over these $\tilde{O}(n^7)$ cuts, we do not have enough time to draw sufficiently many samples to pick every cut. Hence, we need a careful sampling and reweighting algorithm that allows us to sample much fewer cuts but still keep the variance under control. Second, we do not have access to a uniform distribution over these cuts. Instead, we can only sample a tree, obtain a random set of 7 or fewer edges in the tree, and define the corresponding cut as our sampled cut. This creates a biased distribution over the cuts themselves, and our reweighting must eliminate this bias, again without increasing variance.

Formally, our algorithm for estimating $z_G(p)$ has the following three steps.

Step 1: Sparsification. First, we apply the sparsification algorithm (Lemma 2.4) with parameter $\delta = \frac{1}{\log n}$ to get a sparsifier H.

Step 2: Tree Packing. Next, we construct a packing of λ_H spanning trees in the sparsifier graph H where each edge appears at most twice:

LEMMA 3.2. *Given an undirected graph G with min-cut value λ, we can construct in $\tilde{O}(\lambda m)$ time a collection $\mathcal{T}$ of λ spanning trees such that every edge appears in at most two trees.*

This lemma is well-known and follows, e.g., from Gabow [5]; we give a short proof based on Gabow's result in the appendix for completeness. Note that since $\lambda_H = O(\log^3 n)$, this algorithm runs in $\tilde{O}(m)$ time.

We say that a cut k-*respects* a tree if there are at most k edges from the cut in the tree. The key property of the spanning tree packing in Lemma 3.2 is that every 3.5λ-weak cut will 7-respect at least one tree in the packing:

LEMMA 3.3. *Fix any $k \in \{2, 3, \ldots, 7\}$. Let $\beta = 16$. It holds whp that for every cut C_i (in G) with $c_i \leq \left(k + 1 - \frac{\beta}{\log n} \right) \cdot \frac{\lambda}{2}$, there exists a tree $T \in \mathcal{T}$ such that $|C_i \cap T| \leq k$.*

Proof. Let $\mathcal{C}_k$ be the set of cuts C_i in G satisfying $c_i \leq \left(k + 1 - \frac{\beta}{\log n} \right) \cdot \frac{\lambda}{2}$. Recall that $\delta = \frac{1}{\log n}$ is the sparsification parameter we used when applying Lemma 2.4. Thus, for every cut C_i in $\mathcal{C}_k$, we have $c_i \leq (k + 1 - \beta\delta) \cdot \frac{\lambda}{2}$. After

sparsification, the value of every cut in $\mathcal{C}_k$ in the sparsifier H is at most $(1+\delta)(k+1-\beta\delta)\cdot\alpha\cdot\frac{\lambda}{2}$ whp. Moreover, $\lambda_H \geq (1-\delta)\alpha\lambda$ whp. Therefore, the value of every cut in $\mathcal{C}_k$ in the sparsifier H is at most:

$$(1+\delta)(k+1-\beta\delta)\cdot\alpha\cdot\frac{\lambda}{2} \leq \frac{(1+\delta)(k+1-\beta\delta)}{1-\delta}\cdot\frac{\lambda_H}{2} < (k+1)\frac{\lambda_H}{2}, \quad \text{for } \beta \geq 2(k+1).$$

Since $k \leq 7$, our choice of $\beta = 16$ ensures the inequality above. This implies that after sparsification, each cut in $\mathcal{C}_k$ has value strictly smaller than $(k+1)\frac{\lambda_H}{2}$ in H. Since there are λ_H trees produced by Lemma 3.2 and every edge can appear at most twice, it follows that every cut in $\mathcal{C}_k$ has strictly less than $k+1$ edges on average across the trees. Therefore, for every cut in $\mathcal{C}_k$, there is at least one tree containing at most k edges from the cut. $\quad\square$

As a consequence of Lemma 3.3, it suffices to calculate the contribution to $z_G(p)$ of all cuts that 7-respect some tree in the tree packing. We call this latter set $\mathcal{C}_7$. Note that the set $\mathcal{C}_7$ includes all 3.5λ-weak cuts in G, but might include other cuts as well. The remainder of the section will design an unbiased estimator of $z_7(p) = \sum_{C_i\in\mathcal{C}_7} p^{c_i}$. Lemma 3.4 shows that the estimator for $z_7(p)$ is also a (biased) estimator of $u_G(p)$ with an overall relative bias of $O\left(\frac{\log n}{\sqrt{n}}\right)$.

LEMMA 3.4. *Assume $p^\lambda < O(n^{-3})$. Then $|z_7(p) - u_G(p)| \leq O\left(\frac{\log n}{\sqrt{n}}\right) u_G(p)$.*

Proof. Lemma 3.3 gives that $z^{\leq 3.5} \leq z_7(p) \leq z_G(p)$. By Lemma 3.1,

$$|z_7(p) - z_G(p)| \leq |z^{\leq 3.5} - z_G(p)| \leq |z^{\geq 3.5}| \leq O(n^{-0.5})\cdot z_G(p).$$

By Lemma 2.2, since $p^\lambda < O(n^{-3})$, we have $|z_G(p) - u_G(p)| \leq O\left(\frac{\log n}{\sqrt{n}}\right) z_G(p)$ and $z_G(p) = O(u_G(p))$. Therefore,

$$|z_7(p) - u_G(p)| \leq |z_7(p) - z_G(p)| + |z_G(p) - u_G(p)| \leq O\left(\frac{\log n}{\sqrt{n}}\right) u_G(p).$$

$\square$

Step 3: Unbiased Estimator for $z_7(p)$ via Importance Sampling. Recall that $z_7(p)$ is a sum of p^{c_i} over the $\tilde{O}(n^7)$ cuts C_i in $\mathcal{C}_7$. This is much fewer than the $O(2^n)$ cuts overall, but we still cannot afford to directly enumerate all of them. Instead, using importance sampling, we obtain an estimator of $z_7(p)$ with small variance.

Our estimator X is defined as follows: $X = \frac{p^{c_i}}{q(C_i)}$ with probability $q(C_i)$, for some distribution $q : \mathcal{C}_7 \to [0,1]$ that we will define later. To minimize the relative variance, we would ideally want $q(C_i) \propto p^{c_i}$ for all $C_i \in \mathcal{C}_7$. But, this is impossible to ensure exactly because the set $\mathcal{C}_7$ is unknown and too large to enumerate. Instead, we use a surrogate distribution q to approximate this ideal distribution. The surrogate distribution q will be tailored so that we can efficiently sample cuts from q, and moreover that given a cut C_i, we can efficiently compute $q(C_i)$.

In Lemma 3.5, we show that X is an unbiased estimator of $z_7(p)$, and also obtain a bound on the relative variance of X as a function of the distribution q:

LEMMA 3.5. *X is an unbiased estimator of $z_7(p)$ and its relative variance $\eta[X]$ satisfies whp*

$$\eta[X] \leq \frac{1}{z_7(p)}\cdot\left(\max_{C_i\in\mathcal{C}_7}\frac{p^{c_i}}{q(C_i)}\right).$$

Proof. The expectation of X is given by

$$\mathbb{E}[X] = \sum_{C_i\in\mathcal{C}_7} q(C_i)\cdot\frac{p^{c_i}}{q(C_i)} = \sum_{C_i\in\mathcal{C}_7} p^{c_i} = z_7(p).$$

The relative variance of X satisfies

$$\begin{aligned}
\eta[X] \quad &< \quad \frac{\mathbb{E}[X^2]}{\mathbb{E}[X]^2} \quad = \quad \frac{1}{(z_7(p))^2}\sum_{C_i\in\mathcal{C}_7} q(C_i)\left(\frac{p^{c_i}}{q(C_i)}\right)^2 \\
&\leq \quad \frac{1}{z_7(p)}\left(\max_{C_i\in\mathcal{C}_7}\frac{p^{c_i}}{q(C_i)}\right)\left(\frac{\sum_{C_i\in\mathcal{C}_7} p^{c_i}}{z_7(p)}\right) \quad = \quad \frac{1}{z_7(p)}\left(\max_{C_i\in\mathcal{C}_7}\frac{p^{c_i}}{q(C_i)}\right).
\end{aligned}$$

$\square$

The distribution q is defined as the mixture of a set of distributions q_j for $j \in \{1, 2, \ldots, 7\}$, where j can be loosely interpreted as the number of edges in the intersection of the sampled cut $C \in \mathcal{C}_7$ and some tree T chosen randomly from the packing $\mathcal{T}$ given by Lemma 3.2. For $j \geq 2$, the distribution q_j is given by the following process: First, we pick a tree T uniformly at random from the packing $\mathcal{T}$ given by Lemma 3.2. Next, pick j edges uniformly at random from T *with replacement*. The cut C is then defined as the unique cut in G that intersects T at precisely the chosen edges. (Note that the number of chosen edges might actually be less than j because the sampling is with replacement.)

We now precisely calculate the values of $q_j(C_i)$ for any cut C_i and any $j \in \{2, 3, \ldots, 7\}$. The following fact is useful for this purpose (we include a proof in the appendix):

FACT 3.1. *Given a universe U of N elements and a set $A \subseteq U$ of size α, if we pick j elements from U uniformly at random with replacement, then the probability that the set of elements picked is precisely A is given by $\frac{\alpha! \, S(j, \alpha)}{N^j}$, where $S(j, \alpha)$ is the Stirling number of the second kind.*

In particular, when $j = 2$, the probability is $\frac{\alpha}{N^j}$ for $\alpha \in \{1, 2\}$ and 0 otherwise.

Using Fact 3.1, we can now infer that for any cut $C_i \in \mathcal{C}_7$ and any $j \in \{2, 3, \ldots, 7\}$, we have

$$(3.1) \qquad q_j(C_i) = \frac{1}{|\mathcal{T}|} \cdot \sum_{T \in \mathcal{T}} \frac{(|C_i \cap T|)! \cdot S(j, |C_i \cap T|)}{(n-1)^j},$$

where $S(j, \alpha)$ is the Stirling number of the second kind.

We will show later that in the sampling process for q_j described above, cuts of value at least 1.5λ contribute at most $\tilde{O}(n^{1.5})$ to the relative variance of the estimator. But, the contribution of 1.5λ-weak cuts to the relative variance of the estimator can be as large as $\Omega(n^2)$. So, we cannot simply repeat the sampling to obtain an estimator with constant relative variance.

Instead, we mix an additional distribution in q that we call q_1. This distribution q_1 amplifies the weight on the small cuts, so that they contribute less to the relative variance. Alternatively said, q_1 dampens the effect on the relative variance of the very large cuts. This distribution is based on a tree packing on a contracted graph.

To define the contracted graph, we construct a Gomory-Hu tree Y (Definition 2.1) of our input graph G. Let τ be the $\sqrt{n}$-th smallest edge weight in Y. We contract all edges of weight at least τ in Y; the sets of vertices that are contracted into single nodes are now contracted in G as well. This results in a graph G' and its Gomory-Hu tree Y', both on $n' \leq \sqrt{n}$ vertices. Next, we repeat the sparsification (Lemma 2.4) and tree packing (Lemma 3.2) steps on G' to obtain a tree packing $\mathcal{T}'$ on a sparsifier H' of G'. By the same argument as for $\mathcal{T}$, we have $|\mathcal{T}'| = O(\log^3 n) = \tilde{O}(1)$. Now we define q_1 as follows: First, pick a tree T' in $\mathcal{T}'$ uniformly at random. Then, choose 2 edges *with replacement* uniformly at random from T'. Finally, define the sampled cut C as the unique cut in G' (and therefore in G) that intersects T' at precisely the chosen edges. (Note that because of sampling with replacement, the number of edges in the intersection can be either 1 or 2.) Using Fact 3.1, we can precisely state the probability $q_1(C_i)$ for any cut $C_i \in \mathcal{C}_7$:

$$(3.2) \qquad q_1(C_i) = \frac{1}{|\mathcal{T}'|} \cdot \sum_{T' \in \mathcal{T}' : |C_i \cap T'| \in \{1, 2\}} \frac{|C_i \cap T'|}{(n'-1)^2}.$$

We have given a set of distributions $q_1, q_2, \ldots, q_7$ for defining the cut C. Finally, we combine these distributions uniformly: namely, we choose an index $j \in \{1, 2, \ldots, 7\}$ uniformly at random and then choose C according to distribution q_j. Thus, $q(C_i) = \frac{1}{7} \sum_{j=1}^{7} q_j(C_i)$ for $C_i \in \mathcal{C}_7$.

Recall that in Lemma 3.5, we bounded the relative variance of the estimator by $\frac{1}{z_7(p)} \cdot \max_{C_i \in \mathcal{C}_7} \frac{p^{c_i}}{q(C_i)}$. We start by obtaining a bound on $\frac{1}{q(C_i)}$ by individually bounding $\frac{1}{q_j(C_i)}$ for every $j \in \{1, 2, \ldots, 7\}$.

LEMMA 3.6. *If a cut C_i 2-respects some tree in $\mathcal{T}'$, then $\frac{1}{q(C_i)} \leq \tilde{O}(n)$. For $j \in \{2, 3, \ldots, 7\}$, if a cut C_i j-respects some tree in $\mathcal{T}$, then $\frac{1}{q(C_i)} \leq \tilde{O}(n^j)$.*

Proof. First, consider a cut C_i that 2-respects a tree $T' \in \mathcal{T}'$. Since $|C_i \cap T'| \in \{1, 2\}$, we have

$$q(C_i) \geq \frac{1}{7} \cdot q_1(C_i) \geq \frac{1}{7 \cdot |\mathcal{T}'| \cdot (n'-1)^2}$$

We have $|\mathcal{T}'| = O(\log^3 n)$ and $n' \le \sqrt{n}$. Hence, $\frac{1}{q(C_i)} \le \tilde{O}(n)$.

For C_i that j-respects $T \in \mathcal{T}$ for some $j \in \{2, 3, \ldots, 7\}$, we have $1 \le |C_i \cap T| \le j$. Note that $S(j, \alpha) \ge 1$ for $1 \le \alpha \le j$. Therefore,

$$q(C_i) \ge \frac{1}{7} \cdot q_j(C_i) \ge \frac{1}{7 \cdot |\mathcal{T}| \cdot (n-1)^j}$$

Since $|\mathcal{T}| = O(\log^3 n)$, we have $\frac{1}{q(C_i)} \le \tilde{O}(n^j)$. □

Next, we bound the expression $\frac{1}{z_7(p)} \cdot \frac{p^{c_i}}{q(C_i)}$ for every cut $C_i \in \mathcal{C}_7$, thereby bounding the relative variance using Lemma 3.5.

LEMMA 3.7. *Assume* $p^\lambda \le O(n^{-3})$. *For every cut* $C_i \in \mathcal{C}_7$, *we have* $\frac{1}{z_7(p)} \cdot \frac{p^{c_i}}{q(C_i)} \le \tilde{O}(n^{1.5})$.

Proof. By Lemma 3.6, $\frac{1}{q(C_i)} = \tilde{O}(n^7)$ for every cut $C_i \in \mathcal{C}_7$. This is quite a loose bound but it already suffices for large cuts, namely when $c_i \ge 3.5\lambda$. In this case, $p^{c_i - \lambda} \le p^{2.5\lambda} \le O(n^{-7.5})$. Therefore,

$$\frac{1}{z_7(p)} \cdot \frac{p^{c_i}}{q(C_i)} \le \frac{p^{c_i - \lambda}}{q(C_i)} \le O(n^{-7.5}) \cdot \tilde{O}(n^7) \le O(1).$$

Next, suppose the value of C_i satisfies $(j - \beta\delta) \cdot \frac{\lambda}{2} < c_i \le (j + 1 - \beta\delta) \cdot \frac{\lambda}{2}$ for some $j \in \{3, 4, \ldots, 7\}$, where $\delta = \frac{1}{\log n}$ and $\beta = 16$. Then, from Lemma 3.3, we know that C_i j-respects some tree T in the packing $\mathcal{T}$. Thus, $\frac{1}{q(C_i)} = \tilde{O}(n^j)$ by Lemma 3.6. Since $p^\lambda \le O(n^{-3})$ and $\frac{j - \beta\delta}{2} > 1$, we get

$$\frac{p^{c_i}}{z_7(p)} \le p^{c_i - \lambda} \le p^{(j - \beta\delta) \cdot \frac{\lambda}{2} - \lambda} = \left(p^\lambda\right)^{\frac{j - \beta\delta}{2} - 1} \le \left(O(n^{-3})\right)^{\frac{j - \beta\delta}{2} - 1} \le O(n^{3 - 1.5j}).$$

So $\frac{p^{c_i}}{z_7(p)} \cdot \frac{1}{q(C_i)} \le \tilde{O}(n^{3 - 0.5j}) \le \tilde{O}(n^{1.5})$ since $j \ge 3$.

Note that the previous cases cover all $C_i \in \mathcal{C}_7$ satisfying $c_i > (3 - \beta\delta) \cdot \frac{\lambda}{2}$. The remaining case is $c_i \le (3 - \beta\delta) \cdot \frac{\lambda}{2}$. By Lemma 3.3, C_i 2-respects some tree T in the packing $\mathcal{T}$, and hence $\frac{1}{q(C_i)} = \tilde{O}(n^2)$ by Lemma 3.6. Now, there are two subcases depending on whether $c_i \ge \tau$ or $c_i < \tau$. (Recall that τ is the value of $\sqrt{n}$th smallest weight edge in the Gomory-Hu tree Y of G.)

Consider $c_i \ge \tau$. In this case, Y (and therefore G) has at least $\sqrt{n}$ cuts of value at most τ. Since $\tau \le c_i < 1.5\lambda$, it follows that all these $\sqrt{n}$ cuts are in $\mathcal{C}_7$. For each such cut C, we have $p^{|C|} \ge p^\tau$. Therefore, $z_7(p) \ge \sqrt{n} \cdot p^\tau$. Hence,

$$\frac{1}{q(C_i)} \cdot \frac{p^{c_i}}{z_7(p)} \le \tilde{O}(n^2) \cdot \frac{p^\tau}{z_7(p)} \le \tilde{O}(n^{1.5}).$$

Next, consider $c_i < \tau$. In this case, we claim that the cut C_i is preserved in G'. Since the edges that are contracted in Y to form Y' are all of value at least τ, it follows that any pair of vertices in G that are contracted to the same node in G' must be τ-connected[5]. On the other hand, every edge in C_i is in a cut of value strictly less than τ, and therefore, the vertices at the ends of the edge are not τ-connected. In particular, note that the minimum cuts in G are also preserved in G', and therefore, the min-cut value of G' is also λ. Therefore, by Lemma 3.3, we can conclude that C_i 2-respects some tree in $\mathcal{T}'$. It follows by Lemma 3.6 that $\frac{1}{q(C_i)} = \tilde{O}(n)$. We can now bound $\frac{1}{q(C_i)} \cdot \frac{p^{c_i}}{z_7(p)} \le \frac{1}{q(C_i)} = \tilde{O}(n)$. □

Finally, we combine the results to give a proof of Lemma 1.3.

Proof. [Proof of Lemma 1.3] We show that the average of multiple samples of our estimator X satisfies the lemma.

By Lemma 3.5, $\mathbb{E}[X] = z_7(p)$. Using Lemma 3.7 and Lemma 3.5, we conclude that the relative variance of our estimator is $\tilde{O}(n^{1.5})$. Therefore, by repeating the sampling algorithm $\tilde{O}(n^{1.5})$ times, and using Lemma 2.7, we obtain an unbiased estimator for $z_7(p)$ with relative variance $O(1)$. By Lemma 3.4, this estimator has relative bias $O\left(\frac{\log n}{\sqrt{n}}\right)$ for $u_G(p)$.

We will show in Theorem 3.1 that the algorithm has preprocessing time $m^{1+o(1)}$ and each sample takes $\tilde{O}(1)$ time. We take $\tilde{O}(n^{1.5})$ samples, so the running time is $m^{1+o(1)} + \tilde{O}(n^{1.5})$. □

---

[5]A pair of vertices s, t are τ-connected if the s-t min cut value is at least τ.

3.3 Running Time of the Algorithm We now show that after $m^{1+o(1)}$ preprocessing time, we can sample the estimator described in the previous section in $\tilde{O}(1)$ time. Since the estimation algorithm is repeated $\tilde{O}(n^{1.5})$ times, this yields an overall running time of $m^{1+o(1)} + \tilde{O}(n^{1.5})$.

Preprocessing of the algorithm. Preprocessing involves the following steps: computing a Gomory-Hu tree (takes $m^{1+o(1)}$ time for an unweighted graph by [1]), sparsification using uniform probabilities (takes $O(m)$ time by Lemma 2.4), tree packing (takes $\tilde{O}(m\lambda_H) = \tilde{O}(m)$ time by Lemma 3.2 since $\lambda_H = \tilde{O}(1)$), and contractions on the Gomory-Hu tree (takes $O(m)$ time using, e.g., breadth-first search). Therefore, preprocessing takes a total of $m^{1+o(1)}$ time.

Data structure for cut queries. We are left to show that we can sample the estimator in $\tilde{O}(1)$ time. For this purpose, we describe a data structure with respect to any graph G and a tree T defined on the vertices V of G. (In general, we do not require T to be a subgraph of G.) First, we define a mapping of $G = (V, E)$ into a set of points in $\mathbb{R}^2$. To define the mapping, we define an order on the vertices V in G. We create an Euler tour of T starting at an arbitrary vertex; call the resulting vertex sequence $\mathcal{E}(T)$. Next, we order vertices by their first appearance in $\mathcal{E}(T)$; we call this order the *preorder* sequence of the vertices and denote it $v_1, v_2, \ldots, v_n$. Next, for every edge $e = (v_i, v_j)$ in G where $i < j$, we map it to the point (i, j) on $\mathbb{R}^2$.

Our data structure, which we denote $\mathbb{S}_{G,T}$, supports 2-dimensional *orthogonal range queries* on the set of points corresponding to the edges in G. Namely, given a pair of intervals U and W in $\mathbb{R}$, the data structure reports the number of points in the rectangle $U \times W$; we denote this count $\mathbb{S}_{G,T}(U, W)$. In particular, suppose U and W correspond to *disjoint* sets of vertices that are contiguous in the preorder sequence and U comes before W in the sense that for every $v_i \in U, v_j \in W$, we have $i < j$. Then, $\mathbb{S}_{G,T}(U, W) = |\{(v_i, v_j) \in E : v_i \in U, v_j \in W\}|$. In other words, it counts the number of edges between the vertex sets U and W. This data structure can be implemented using standard results in computational geometry, e.g., [18]. For a set of m points, the data structure has a preprocessing time of $\tilde{O}(m)$ and supports orthogonal range queries (i.e., reports $\mathbb{S}_{G,T}(U, W)$ for given U, W) in $O(\log^2 m)$ time.

On top of this data structure, we need to maintain some additional information. Recall the Euler tour $\mathcal{E}(T)$ of tree T. Note that every edge is traversed twice in $\mathcal{E}(T)$. We label the traversal of an edge by the number of distinct vertices in V that we have seen so far in the Euler tour. This means that for any i, the Euler tour visits all edges that have a label of i between the first occurrences of v_i and v_{i+1} in $\mathcal{E}(T)$. Conversely, each edge gets two labels i and j and is visited between the first occurrences of v_i, v_{i+1} and v_j, v_{j+1} in $\mathcal{E}(T)$. We denote the two labels for edge e by the (two-element) set $L(e)$ and collectively, $\mathbb{L}_T = \{L(e) : e \in T\}$. Note that we can compute $\mathbb{L}_T$ in $O(n)$ preprocessing time by a single traversal of $\mathcal{E}(T)$.

Let χ be a set of edges in T. There is a unique cut in T such that the cut edges are exactly those in χ. We denote the vertex bipartition of this cut by $C(\chi)$. Since G and T are defined on the same set of vertices, the vertex bipartition $C(\chi)$ also induces a cut in G; overloading notation, we call this cut $C(\chi)$ as well. Note that in particular, if T is a subgraph of G, then $\chi = T \cap C(\chi)$.

Using our data structures $\mathbb{S}_{G,T}$ and $\mathbb{L}_T$, we now show that the following cut query can be answered efficiently: *given an edge set χ in T of constant size, calculate the cut value of the vertex bipartition $C(\chi)$ in G.*

LEMMA 3.8. *Suppose we have the data structures $\mathbb{S}_{G,T}$ and $\mathbb{L}_T$ for a graph G and a spanning tree T of G. Now, for any set χ of a constant number of edges in T, the value of the cut $C(\chi)$ in G can be computed using a constant number of orthogonal range queries on $\mathbb{S}_{G,T}$. Therefore, the total query time for χ is $\tilde{O}(1)$.*

Proof. Let $\mathbb{L}_T(\chi)$ be the multiset of labels $L(e)$ corresponding to the edges $e \in \chi$. Since each edge has two labels and $|\chi| = O(1)$, this multiset has constant size. Next, we sort the labels in $\mathbb{L}_T(\chi)$; denote this sorted sequence $\ell_1, \ell_2, \ldots, \ell_{2|\chi|}$. We append this list with $\ell_0 = 0$ and $\ell_{2|\chi|+1} = n$. For any pair of consecutive labels ℓ_i, ℓ_{i+1} for $i = 0, 1, \ldots, 2|\chi|$ in this appended list, we define a vertex set $V_i = \{v_j : \ell_i < j \le \ell_{i+1}\}$. Note that it is possible that $\ell_i = \ell_{i+1}$, in which case $V_i = \varnothing$. Now, define $V_{\text{odd}} = V_1 \cup V_3 \cup \ldots \cup V_{2|\chi|-1}$ and $V_{\text{even}} = V_0 \cup V_2 \cup \ldots \cup V_{2|\chi|}$. Note that although the sets V_{odd} and V_{even} can be superconstant in size, they can be represented and computed from the labels in $\mathbb{L}_T(\chi)$ in $O(1)$ time by denoting each interval V_i by its two endpoints $\ell_i + 1$ and ℓ_{i+1}.

Finally, we note that the vertex bipartition $C(\chi)$ is precisely given by the sets V_{odd} and V_{even}. To see this, consider the Euler tour $\mathcal{E}(T)$ and mark every edge in χ on it. Now, the vertices switch from one side of $C(\chi)$ to the other side every time we traverse a marked edge. This corresponds to taking the odd and even indexed sets V_i, which is exactly how we defined the sets V_{odd} and V_{even}. Finally, we take each pair of sets V_i, V_j where $V_i \in V_{\text{odd}}$ and $V_j \in V_{\text{even}}$ and run the query $\mathbb{S}_{G,T}(V_i, V_j)$ on the data structure $\mathbb{S}_{G,T}$. Note that this is only $O(1)$

number of queries, and each query takes $O(\log^2 m)$ time by the properties of $\mathbb{S}_{G,T}$. Finally, we return the sum of the values returned by these queries as the number of edges in cut $C(\chi)$ in graph G. □

Specifically, we instantiate the following data structures $\mathbb{S}$ and $\mathbb{L}$ for our purpose. First, we construct $\mathbb{S}_{G,T}$ for every tree $T \in \mathcal{T} \cup \mathcal{T}'$ (with G as the input graph). Note that the data structure requires T to be defined on the same set of vertices as G, which does not hold for the trees $T' \in \mathcal{T}'$. To resolve this, we expand the contracted vertices in G' into connected components in G and connect each component by an arbitrary spanning tree. By doing this, the tree T' is expanded into a tree defined on the vertices V. Moreover, this step will not change the intersection of T' with any cut in G'. Therefore, we can use the expanded tree to build $\mathbb{S}_{G,T'}$ and report the correct cut value of $C(\chi)$ for any χ defined on T'. We also build a data structure $\mathbb{S}_{T_1,T_2}$ for every pair of trees $T_1, T_2 \in \mathcal{T} \cup \mathcal{T}'$. Note that after expansion of the trees in $\mathcal{T}'$, all the trees T_1, T_2 are on the same set of vertices V; thus the data structures are well-defined. In addition to these data structures, we also construct the label sets $\mathbb{L}_T$ for all trees $T \in \mathcal{T} \cup \mathcal{T}'$. Note that the total number of data structures is $\tilde{O}(1)$, which implies that the preprocessing time increases only by a $\tilde{O}(1)$ factor.

Running time of the sampling algorithm. First, we compute the running time for producing a single sample for the estimator X defined in Step 3 of Section 3.2 using the data structures $\mathbb{S}_{G,T}$ and $\mathbb{L}_T$ that we defined above. The first step of the sampling algorithm is to choose a specific distribution q_j from $j \in \{1, 2, \ldots, 7\}$ uniformly at random in $O(1)$ time. Next, the algorithm chooses a tree T uniformly at random from the packing $\mathcal{T}$ (or $\mathcal{T}'$ for q_1) in $\tilde{O}(1)$ time. Third, the algorithm chooses a set χ of j edges from T uniformly at random with replacement. This takes $\tilde{O}(1)$ time since j is a constant.

The only involved step is to calculate the estimator X at this point. Recall that X is defined as $\frac{p^{|C(\chi)|}}{q(C(\chi))}$. First, we compute the cut value $|C(\chi)|$ by querying $\mathbb{S}_{G,T}$ in $\tilde{O}(1)$ time by Lemma 3.8. We are left to compute $q(C(\chi))$, which only requires computing $|C(\chi) \cap T'|$ for every tree $T' \in \mathcal{T} \cup \mathcal{T}'$ (where $C(\chi)$ is defined on G) by using Equations (3.1) and (3.2). (Note that the Stirling numbers in Equation (3.1) can be retrieved from a constant-sized table since $j \leq 7$.) To compute $|C(\chi) \cap T'|$, we use the data structure $\mathbb{S}_{T',T}$ to obtain the cut value of $C(\chi)$ in T', which is $|C(\chi) \cap T'|$ where $C(\chi)$ is defined on graph G. Overall, this requires $\tilde{O}(1)$ queries of the data structures, each of which takes $\tilde{O}(1)$ time by Lemma 3.8.

We have established the following theorem:

THEOREM 3.1. *There is an algorithm that takes $m^{1+o(1)}$ preprocessing time and $\tilde{O}(1)$ query time to produce a sample of the estimator X.*

4 Recursive Contraction

In this section, we design a recursive contraction algorithm that solves the moderately reliable case, that is, $4n^{-3} \leq p^\lambda \leq n^{-0.5}$. Our goal is to show Lemma 1.4, which we restate below:

LEMMA 1.4. *Suppose $4n^{-3} \leq p^\lambda \leq n^{-0.5}$ and $p < \theta$. An estimator X for $u_G(p)$ with relative bias $\leq 0.1\varepsilon$ and relative second moment $\leq \log^{O(1)} n$ can be computed in $m^{1+o(1)} + \tilde{O}(n^{1.5}\varepsilon^{-1})$ time.*

4.1 Description of the Algorithm We use a parameter γ to measure how many near-minimum cuts contribute significantly to $z_G(p)$. Intuitively, if there are very few such cuts, then the average degree is much larger than λ. In this case, random contraction will significantly decrease the size of the graph. On the other hand, if there are many near-minimum cuts, then the variance of the estimator $u_H(p/q)$ will be small. Our definition of γ allows us to smoothly interpolate between these two extreme cases.

We choose γ as follows. First, compute the Gomory-Hu tree Y of G (Definition 2.1). Let S_k be the sum of the k smallest weights among the tree edges. Then,

$$\gamma = \min\left\{\lambda, \frac{3}{4} \cdot \frac{S_k}{k}\right\} \text{ for } k = 2^{-2/3} \cdot n.$$

Set q so that $q^\gamma = 1/2$. The algorithm samples two graphs independently $H_1, H_2 \sim G(q)$ and returns the average of $u_{H_1}(p/q)$ and $u_{H_2}(p/q)$, which are computed recursively. Note that $G(q)$ represents (the distribution of) the random graph generated by contracting each edge in G independently with probability $1 - q$.

It will be important to ensure that $m \leq n^{1.5-\xi}$ for some small enough constant $\xi > 0$ whenever we apply the recursive contraction step. If this condition is violated, we perform two algorithmic steps that restore the condition in expectation. The first step is to reduce the minimum cut of the graph to $\tilde{\lambda} = O(\log^3 n)$ if it is larger. For this purpose, we use the (standard) *sparsification* algorithm in Section 5 with sparsification parameter $\delta = \frac{1}{\log n}$. To control the additional variance caused by this sparsification step, we also branch whenever we sparsify. Namely, instead of creating a single sparsifier, we create two independent sparsifier graphs $\tilde{G}_1$ and $\tilde{G}_2$ from graph G using the same sparsification parameter. We now recurse on these graphs. In particular, if any of the graphs satisfies one of the base cases given below, then we run the corresponding base case algorithm. If not, then we must perform recursive contraction. This means that we draw two independent random samples each from $\tilde{G}(q)$ for any such graph $\tilde{G}$ produced after sparsification. Let $\tilde{m}$ be the number of edges in any graph drawn from $\tilde{G}(q)$. The important property that we we will establish (using Lemma 2.5) is that $\mathbb{E}[\tilde{m}] = O(n\tilde{\lambda}) = \tilde{O}(n)$. Now, the algorithm recurses on this graph. In other words, if this new graph has $\tilde{n}$ vertices, then we check if $\tilde{m} > \tilde{n}^{1.5-\xi}$ and continue the algorithm.

There are three base cases in the recursion:

1. When n is less than some sufficiently large constant (polynomial in ε^{-1}, will be decided in Lemma 4.5), we use Karger's algorithm (Theorem 1.2) to compute an unbiased estimator of $u_G(p)$ with relative variance $O(1)$.

2. When $p \geq \theta$ or $p^\lambda > n^{-0.5}$, we use Monte Carlo sampling (Section 6) to get an unbiased estimator of $u_G(p)$ with relative variance $O(1)$ (Lemmas 1.1 and 1.2).

3. When $p^\lambda < 4n^{-3}$, we use importance sampling on a spanning tree packing (Section 3) to get an estimate of $z_7(p)$, which is a biased estimator for $u_G(p)$ with bias $O\left(\frac{\log n}{\sqrt{n}}\right) \cdot u_G(p)$ and relative variance $O(1)$ (Lemma 1.3).

4.2 Properties of the Parameter γ The results in this section characterize important properties of γ that allow us to simultaneously relate γ to the shrinkage of the graph under random contractions and the variance of the recursive estimator for unreliability.

FACT 4.1. $\frac{3}{4}\lambda \leq \gamma \leq \lambda$.

Proof. Recall that S_k is the sum of the k edges with minimum weight in the Gomory-Hu tree and $\gamma = \min\left\{\lambda, \frac{3}{4} \cdot \frac{S_k}{k}\right\}$ for $k = 2^{-2/3}n$. If $\gamma = \lambda$, the statement is trivial. Next, assume $\gamma = \frac{3}{4} \cdot \frac{S_k}{k} \leq \lambda$. Because each Gomory-Hu tree edge represents a cut in G whose value is at least that of the minimum cut λ, we have $S_k \geq k\lambda$. Hence, $\gamma = \frac{3}{4} \cdot \frac{S_k}{k} \geq \frac{3}{4}\lambda$. $\square$

LEMMA 4.1. *Assume $p^\lambda \geq 4n^{-3}$. Then $z_G(p) \geq p^\gamma$.*

Proof. If $\gamma = \lambda$, then $z_G(p) \geq p^\lambda$ is trivial. Next assume $\gamma = \frac{3}{4} \cdot \frac{S_k}{k}$. Consider the partial sum of the $k = 2^{-2/3} \cdot n$ smallest cuts represented by edges in the Gomory-Hu tree. Let their cut values be $(a_1, \ldots, a_k)$.

$$(4.3) \qquad z_G(p) \geq \sum_{i=1}^{k} p^{a_i} = k\left(\frac{1}{k}\sum_{i=1}^{k} p^{a_i}\right) \geq k \cdot p^{\frac{1}{k}\sum_{i=1}^{k} a_i} \text{(by convexity)} = kp^{S_k/k} = 2^{-2/3} \cdot n \cdot p^{\frac{4}{3}\gamma}.$$

Since $p^\lambda \geq 4n^{-3}$, we have $p^{-\lambda/3} \leq 2^{-2/3} \cdot n$. Therefore, from Equation (4.3), we get

$$z_G(p) \geq p^{\frac{4}{3}\gamma - \frac{1}{3}\lambda} \geq p^{\frac{4}{3}\gamma - \frac{1}{3}\gamma} \text{(since } \gamma \leq \lambda\text{)} = p^\gamma.$$

$\square$

The next two claims are crucial in our analysis of the shrinkage of the graph under random contractions. Lemma 4.2 gives a lower bound on the number of edges in the graph relative to the number of vertices as we randomly contract edges. This can be compared to the bound of $\frac{|E(H)|}{|V(H)|} \geq \frac{1}{2}\lambda$ used in [14]. The latter bound can be derived from the fact that the value of the minimum cut in H, obtained by contracting edges in G, is at

least λ, which is the value of the minimum cut in G. Therefore, the degree of every vertex in H is at least λ. In comparison, our use of the parameter γ allows us to derive a bound of $\frac{|E(H)|}{|V(H)|-1} \geq \frac{2}{3}\gamma$ on the shrinkage of the graph. To compare the two bounds, note that $\frac{2}{3}\gamma \geq \frac{1}{2}\lambda$ since $\gamma \geq \frac{3}{4}\lambda$ by Fact 4.1. Thus, our bound is always (asymptotically) better, which will be crucial in obtaining the better running time bound.

LEMMA 4.2. *Suppose H is formed by contraction from G, and $|V(H)| > 2^{-2/3} \cdot n$. Then*

$$\frac{|E(H)|}{|V(H)|-1} \geq \frac{2}{3}\gamma.$$

Proof. Root the Gomory-Hu tree Y at an arbitrary vertex r. Let $\mathrm{wt}_Y(\cdot)$ denote edge weights in Y. Let $k' = |V(H)| > k = 2^{-2/3} \cdot n$. Let $\phi : V(G) \to V(H)$ be the contraction map that takes each vertex of G to its contracted node in H. Note that the nodes in H correspond to a partition of the vertices in G. Overloading notation, we also use ϕ to map a vertex in $V(G)$ to the subset of this partition that the vertex belongs to.

For any $W \in V(H) \setminus \{\phi(r)\}$, pick an arbitrary vertex $w \in V(G)$ such that $\phi(w) = W$. Since $\phi(w) \neq \phi(r)$, the tree path from r to w in Y contains at least one tree edge (u_W, v_W) (where u_W is the parent of v_W in T) such that $\phi(u_W) \neq W$ and $\phi(v_W) = W$. Let T' be the collection of such edges, i.e., $T' = \{(u_W, v_W) : W \in V(H) \setminus \{\phi(r)\}\}$. Note that for $W \neq W'$, we have $(u_W, v_W) \neq (u_{W'}, v_{W'})$ since $\phi(v_W) = W$ and $\phi(v_{W'}) = W'$. Thus, $|T'| = k' - 1$.

For any $W \in V(H) \setminus \{\phi(r)\}$, the edge (u_W, v_W) represents a minimum (u_W, v_W) cut in G. Note also that the degree cut of W in H (denote its value $\deg_H(W)$) is a (u_W, v_W) cut in G. Thus, $\mathrm{wt}_Y(u_W, v_W) \leq \deg_H(W)$. Taking the sum over edges in T' gives

$$(4.4) \qquad \sum_{e \in T'} \mathrm{wt}_Y(e) \leq \sum_{W \in V(H) \setminus \{\phi(r)\}} \deg_H(W) \leq \sum_{W \in V(H)} \deg_H(W).$$

Recall that $S_{k'-1}$ denotes the the sum of the smallest $k' - 1$ weights of edges in Y. Hence, we have $\sum_{e \in T'} \mathrm{wt}_Y(e) \geq S_{k'-1}$. Now, note that $S_{k'-1}$ is the prefix sum of a nondecreasing sequence, where $k' - 1 \geq k$. Thus, $\frac{S_{k'-1}}{k'-1} \geq \frac{S_k}{k} \geq \frac{4}{3}\gamma$ by the definition of γ. Putting these together, we get

$$(4.5) \qquad \frac{4}{3}\gamma \leq \frac{S_{k'-1}}{k'-1} \leq \frac{1}{k'-1} \cdot \sum_{e \in T'} \mathrm{wt}_Y(e).$$

Combining Equations (4.4) and (4.5), we get

$$\frac{2}{3}\gamma \overset{(4.5)}{\leq} \frac{1/2}{k'-1} \cdot \sum_{e \in T'} \mathrm{wt}_Y(e) \overset{(4.4)}{\leq} \frac{1/2}{k'-1} \cdot \sum_{W \in V(H)} \deg_H(W) = \frac{|E(H)|}{k'-1} \quad \left(\text{since} \sum_{w \in V(H)} \deg_H(w) = 2|E(H)|\right).$$

$\square$

An important consequence of the previous lemma (Lemma 4.2) is that the number of vertices in the graph after a contraction step can be bounded as follows:

LEMMA 4.3. *$H \sim G(q)$ has vertex size at most $(2^{-2/3} + n^{-0.1}) \cdot n$ whp.*

Proof. The random graph $G(q)$ can be modeled by the following continuous-time random process. Initially, the graph is empty. Edges $e \in E$ arrive at times t_e, which are i.i.d. exponential variables with rate 1.

Let us define a graph G_t by contracting all edges in G that arrive before time t. Notice that $G_{-\ln q}$ is exactly $G(q)$. Let n_t and m_t be the number of vertices and edges in G_t respectively. We consider how the vertex size n_t evolves with t. Let $t(k)$ be the first time that $n_t \leq k$. Initially $n_0 = n$ and $t(n) = 0$. Now, n_t decreases by 1 whenever an uncontracted edge arrives. Since there are m_t uncontracted edges at time t, the earliest arriving time follows an exponential distribution of rate m_t. Let $\Delta_k = t(k-1) - t(k)$; note that Δ_k follows the exponential distribution of rate $m_{t(k)}$.

Because of the memoryless property of the exponential distribution and independence of edges, Δ_k for different values of k are independent. By Lemma 4.2, we have $m_{t(k)} \geq \frac{2}{3}\gamma(k-1)$ for any $k > 2^{-2/3} \cdot n$. We couple each

Δ_k with a new random variable Δ'_k, where Δ'_k follows an exponential distribution of rate $\frac{2}{3}\gamma(k-1)$. Thus, Δ_k is stochastically dominated by Δ'_k for any $k > 2^{-2/3} \cdot n$.

The random variables Δ'_k can be coupled with the aforementioned contraction process on a star graph. Let R be a star on n vertices with $n-1$ edges. We use the same notation as above for graph R, but with a superscript of R. Note that during the contraction process, we always have $m_t^R = n_t^R - 1$. Therefore, $\Delta_k^R = t^R(k-1) - t^R(k)$ follows an independent exponential distribution of rate $k - 1$ by the same argument. We can re-define $\Delta'_k = \Delta_k^R / (\frac{2}{3}\gamma)$. The coupling now gives that for all $k > 2^{-2/3} \cdot n$, we have

$$t(k) = t(k) - t(n) = \sum_{i=k+1}^{n} \Delta_i \leq \sum_{i=k+1}^{n} \frac{3}{2\gamma} \cdot \Delta_i^R = \frac{3}{2\gamma} \cdot \left(t^R(k) - t^R(n)\right) = \frac{3}{2\gamma} \cdot t^R(k).$$

This implies that either $n_t \leq 2^{-2/3} \cdot n$, or $n_t \leq n_\tau^R$ where $\tau = \frac{2\gamma}{3} \cdot t$.

Recall that when $t = -\ln q$, n_t is the vertex size of $G(q)$. This corresponds to $\tau = -\frac{2\gamma}{3} \cdot \ln q$ in the contraction process defined for R. That is each edge in R fails with probability $e^{-\tau} = q^{\frac{2\gamma}{3}} = 2^{-2/3}$ since $q^{-\gamma} = 2$. Let X be the number of uncontracted edges in R at time τ. Then by Hoeffding's inequality,

$$\Pr[X \geq (1 + n^{-0.1}) \cdot 2^{-2/3} \cdot (n-1)] \leq \exp(-\frac{1}{3} \cdot (1 + n^{-0.1})^2 \cdot 2^{-2/3} \cdot (n-1)) = \exp(-\Omega(n^{0.8})).$$

So $n_\tau^R = X + 1 \leq (2^{-2/3} + n^{-0.1}) \cdot n$ whp. By coupling, $|V(H)| = n_{-\ln q} \leq (2^{-2/3} + n^{-0.1}) \cdot n$ whp. □

4.3 Bias and Variance of the Estimator First, we bound the bias of the estimator. We start by observing that the contraction steps themselves are unbiased.

LEMMA 4.4. *Assume H is generated by $H \sim G(q)$ for some $q < p$. Then, $u_H(p/q)$ is an unbiased estimator of $u_G(p)$.*

Proof. The lemma follows from the observation that deleting each edge with probability p from G is equivalent to first creating a set of candidate edges by sampling each edge with probability q, and then removing each candidate edge with probability p/q. □

We prove a similar property of unbiasedness of sparsification steps (Lemma 5.1 in Section 5). Given these lemmas, the only source of bias are the base cases. We can now bound overall bias of the estimator as follows:

LEMMA 4.5. *The algorithm outputs an estimate of $u_G(p)$ with bias at most $0.1\varepsilon \cdot u_G(p)$.*

Proof. Let X be the output of the algorithm. Consider the base cases of the recursion. Karger's algorithm (Theorem 1.2) for a small instance and the Monte Carlo sampling algorithms have no bias.

In the last base case of importance sampling, the bias is $O\left(\frac{\log n}{\sqrt{n}}\right) \cdot u_G(p)$ by Lemma 1.3. We can set the constant threshold for the first base case to be $O(\varepsilon^{-2} \log^2 \frac{1}{\varepsilon})$, so that for n larger than the threshold we have $O\left(\frac{\log n}{\sqrt{n}}\right) \leq 0.1\varepsilon$.

We use induction on recursion depth to prove that $(1 - 0.1\varepsilon) \cdot u_G(p) \leq \mathbb{E}[X] \leq (1 + 0.1\varepsilon) \cdot u_G(p)$ holds for the recursive case as well.

Consider an inductive step where the algorithm takes the average of two recursive calls. Let X_1 and X_2 be the output of the two recursive calls, so that $X = \frac{X_1 + X_2}{2}$. This can happen for two reasons: recursive contraction or sparsification. First, we consider the case of contraction. Later, we extend the same analysis to sparsification.

Let H_1 and H_2 be the two recursively generated graphs by contraction. By the inductive hypothesis,

$$|\mathbb{E}[X_i|H_i] - u_{H_i}(p/q)| \leq 0.1\varepsilon \cdot u_{H_i}(p/q).$$

By Lemma 4.4, $\mathbb{E}[u_{H_i}(p/q)] = u_G(p)$. Therefore,

$$|\mathbb{E}[X_i] - u_G(p)| = |\mathbb{E}_{H_i}[\mathbb{E}[X_i|H_i]] - \mathbb{E}_{H_i}[u_{H_i}(p/q)]| \leq \mathbb{E}_{H_i}[|\mathbb{E}[X_i|H_i] - u_{H_i}(p/q)|]$$
$$\leq \mathbb{E}_{H_i}[0.1\varepsilon \cdot u_{H_i}(p/q)] \quad \text{(by inductive hypothesis)} = 0.1\varepsilon \cdot \mathbb{E}_{H_i}[u_{H_i}(p/q)] = 0.1\varepsilon \cdot u_G(p).$$

Thus, $|\mathbb{E}[X] - u_G(p)| \leq \frac{1}{2}\left(|\mathbb{E}[X_1] - u_G(p)| + |\mathbb{E}[X_2] - u_G(p)|\right) \leq 0.1\varepsilon \cdot u_G(p)$, using the previous inequality.

In a sparsification step, the algorithm also takes the average of two recursive calls $X = \frac{X_1 + X_2}{2}$ on the sparsifier graphs $\tilde{G}_1, \tilde{G}_2$. To replicate the proof above, we need to show that $u_{\tilde{G}_1}(q)$ and $u_{\tilde{G}_2}(q)$ are unbiased estimators of $u_G(p)$. We show this in Lemma 5.1 in Section 5. ☐

We now need to bound the relative variance of the estimator. Again, there are two cases depending on whether contraction or sparsification is being used in the current computation node. We first prove the bound on the relative variance due to contraction steps below. The bound for sparsification is very similar, and is established formally in Section 5.

Instead of bounding the relative variance of $u_H(p/q)$, we will bound the relative variance of $z_H(p/q)$ and invoke the following lemma:

LEMMA 4.6. *Assume $p < \theta$. If $z_H(q)$ is an unbiased estimator of $z_G(p)$ with relative variance η, and $u_H(q)$ is an unbiased estimator of $u_G(p)$, then $u_H(q)$ has relative variance at most $\left(1 + O\left(\frac{1}{\log n}\right)\right)\eta + O\left(\frac{1}{\log n}\right)$.*

Proof. Since $p < \theta$, we have $u_G(p) \geq \left(1 - \frac{1}{\log n}\right) \cdot z_G(p)$ by Lemma 2.1. We also have $u_H(q) \leq z_H(q)$ for any $q \in [0,1]$. Since $\mathbb{E}[u_H(q)] = u_G(p)$, the relative variance of $u_H(q)$ can be bounded as follows:

$$\frac{\mathbb{E}[u_H(q)^2]}{(u_G(p))^2} - 1 \leq \frac{1}{\left(1 - \frac{1}{\log n}\right)^2} \cdot \frac{\mathbb{E}[z_H(q)^2]}{(z_G(p))^2} - 1 \leq \left(1 + O\left(\frac{1}{\log n}\right)\right)\eta + O\left(\frac{1}{\log n}\right).$$

☐

We now bound the relative variance of $z_H(p/q)$. For the base case, we will use the following known bound:

LEMMA 4.7. (LEMMA 3.1 OF [14]) *Assume $p < \theta$. Sample $H \sim G(q)$ for $q^{-\lambda} = \Theta(1)$. Then, $u_H(p/q)$ is an unbiased estimator of $u_G(p)$, and $z_H(p/q)$ is an unbiased estimator of $z_G(p)$. Moreover, both estimators have relative second moment upper bounded by*

$$q^{-\lambda} \cdot \left(1 + O\left(\frac{1}{\log p^{-\lambda}}\right) + O\left(\frac{1}{\log n}\right)\right).$$

LEMMA 4.8. *Assume that $4n^{-3} \leq p^\lambda \leq n^{-0.5}$ and $p < \theta$, where θ is given in Lemma 2.1. Then the relative second moment of $z_H(p/q)$ is $2 + O\left(\frac{1}{\log n}\right)$ where $H \sim G(q)$ and $q^\gamma = 1/2$.*

Proof. There are two cases depending on the relative values of γ and λ.

First, suppose $\left(1 + \frac{5}{\log n}\right)\gamma \geq \lambda$. Then,

$$q^{-\lambda} \leq q^{-\gamma(1+5/\log n)} = 2^{1+5/\log n} \leq 2\left(1 + O\left(\frac{1}{\log n}\right)\right) = 2 + O\left(\frac{1}{\log n}\right).$$

By Lemma 4.7, when $p^\lambda \leq n^{-0.5}$, the relative second moment of $z_H(p/q)$ is upper bounded by $q^{-\lambda}\left(1 + O\left(\frac{1}{\log n}\right)\right)$, which is at most $2 + O\left(\frac{1}{\log n}\right)$ by our bound on $q^{-\lambda}$ above.

In the rest of the proof, we assume $\left(1 + \frac{5}{\log n}\right)\gamma < \lambda$. This proof now follows the same template as in Karger's analysis [14]. The main difference is that we use the parameter γ whereas Karger uses λ.

The relative second moment is bounded as:

$$\frac{\mathbb{E}[z_H(p/q)^2]}{(z_G(p))^2} = \frac{1}{(z_G(p))^2} \cdot \mathbb{E}\left[\sum_{C_i, C_j \in \mathcal{C}(H)} \left(\frac{p}{q}\right)^{c_i + c_j}\right] = \frac{1}{(z_G(p))^2} \sum_{C_i, C_j \in \mathcal{C}(G)} q^{|C_i \cup C_j|} \cdot \left(\frac{p}{q}\right)^{c_i + c_j}$$

$$= \frac{1}{(z_G(p))^2}\left(\sum_{C_i} \frac{p^{2c_i}}{q^{c_i}} + \sum_{C_i \neq C_j : |C_i \cap C_j| \leq \gamma} \frac{p^{c_i + c_j}}{q^{|C_i \cap C_j|}} + \sum_{C_i \neq C_j : |C_i \cap C_j| > \gamma} \frac{p^{c_i + c_j}}{q^{|C_i \cap C_j|}}\right).$$

In the above expression, we distinguished between the cases $C_i = C_j$ and $C_i \neq C_j$, and the second case is further split into $|C_i \cap C_j| \leq \gamma$ and $|C_i \cap C_j| > \gamma$. We define:

$$V_1 = \sum_{C_i} \frac{p^{2c_i}}{q^{c_i}}$$

$$V_2 = \sum_{C_i \neq C_j : |C_i \cap C_j| \leq \gamma} \frac{p^{c_i + c_j}}{q^{|C_i \cap C_j|}}$$

$$V_3 = \sum_{C_i \neq C_j : |C_i \cap C_j| > \gamma} \frac{p^{c_i + c_j}}{q^{|C_i \cap C_j|}}.$$

We first bound V_1:

$$V_1 - 2 \sum_{C_i} p^{2c_i} = \sum_{C_i} p^{2c_i} \left(\frac{1}{q^{c_i}} - \frac{1}{q^\gamma} \right) = \frac{p^\gamma}{q^\gamma} \sum_{C_i} p^{c_i} \cdot p^{c_i - \gamma}(q^{-(c_i - \gamma)} - 1) = 2 \cdot p^\gamma \sum_{C_i} p^{c_i} \cdot p^{c_i - \gamma}(q^{-(c_i - \gamma)} - 1)$$

Let us also denote $t_i = \frac{c_i - \gamma}{\gamma}$ and $f(t_i) = (2u)^{t_i} - u^{t_i} = p^{c_i - \gamma}(q^{-(c_i - \gamma)} - 1)$ where $u = p^\gamma$. Therefore,

$$V_1 - 2 \sum_{C_i} p^{2c_i} = 2p^\gamma \sum_{C_i} p^{c_i} \cdot f(t_i).$$

Note that since $p^\lambda \in [n^{-3}, n^{-0.5}]$ and $\gamma \in [\frac{3}{4}\lambda, \lambda]$, we have $u = p^\gamma \in [n^{-3}, n^{-0.375}]$. We now use the following fact, which we prove in the appendix:

FACT 4.2. *For function $f(t) = (2u)^t - u^t$, suppose $u \leq n^{-0.375}$ and $t \geq \frac{5}{\log n}$. Then, for n larger than some constant, we have $\frac{\mathrm{d}f}{\mathrm{d}t} < 0$.*

Thus, f is monotone decreasing when $t \geq \frac{5}{\log n}$. Since $t_i \geq \frac{\lambda - \gamma}{\gamma} \geq \frac{5}{\log n}$, we can upper bound $f(t_i) \leq f\left(\frac{5}{\log n}\right) = u^{5/\log n} \cdot (2^{5/\log n} - 1)$. Combined with $p^\gamma \leq z_G(p)$ by Lemma 4.1, we have

$$V_1 - 2 \sum_{C_i} p^{2c_i} \leq 2 \cdot p^\gamma \cdot z_G(p) \cdot f\left(\frac{5}{\log n}\right) \leq 2 \cdot (z_G(p))^2 \cdot u^{5/\log n} \cdot (2^{5/\log n} - 1)$$

$$\leq 2(z_G(p))^2 \left(n^{-0.375}\right)^{5/\log n} \cdot (2^{5/\log n} - 1) \leq O\left(\frac{1}{\log n}\right) \cdot (z_G(p))^2.$$

Thus,

(4.6)
$$V_1 \leq O\left(\frac{1}{\log n}\right) \cdot (z_G(p))^2 + 2 \sum_{C_i} p^{2c_i}.$$

Next, we bound V_2 as follows:

(4.7)
$$V_2 = \sum_{C_i \neq C_j, |C_i \cap C_j| \leq \gamma} \frac{p^{c_i + c_j}}{q^{|C_i \cap C_j|}} \leq q^{-\gamma} \cdot \sum_{C_i, C_j} p^{c_i + c_j} - q^{-\gamma} \cdot \sum_{C_i} p^{2c_i} = 2(z_G(p))^2 - 2 \cdot \sum_{C_i} p^{2c_i}.$$

Finally, we bound V_3 as follows:

$$V_3 = \sum_{C_i \neq C_j, |C_i \cap C_j| > \gamma} p^{|C_i \cup C_j|} \left(\frac{p}{q}\right)^{|C_i \cap C_j|} \leq \sum_{C_i \neq C_j} p^{|C_i \cup C_j|} \left(\frac{p}{q}\right)^\gamma = q^{-\gamma} \cdot x_G(p) \cdot p^\gamma \leq 2 \cdot x_G(p) \cdot z_G(p).$$

Since $p < \theta$, Lemma 2.1 gives $x_G(p) \leq \frac{1}{\log n} \cdot z_G(p)$. Therefore,

$$(4.8) \qquad V_3 \leq 2 \cdot x_G(p) \cdot z_G(p) \leq \frac{2}{\log n} \cdot (z_G(p))^2.$$

Putting the bounds on V_1, V_2 and V_3 given by Equations (4.6) to (4.8) together, we get

$$\frac{\mathbb{E}[z_H(p/q)^2]}{(z_G(p))^2} \leq \frac{V_1 + V_2 + V_3}{(z_G(p))^2} = 2 + O\left(\frac{1}{\log n}\right).$$

□

Combining Lemmas 4.6 to 4.8, we obtain the following:

COROLLARY 4.1. $u_H(p/q)$ *is an unbiased estimator of* $u_G(p)$ *with relative second moment at most* $2 + O\left(\frac{1}{\log n}\right)$.

We now use induction to bound the second moment of the overall estimator. This requires our bounds on the base cases as well as that established for a recursive contraction step in Lemma 4.8 and the corresponding bound for a sparsification step that we establish in Lemma 5.2 in Section 5.

LEMMA 4.9. *The second moment of the estimator given by the overall algorithm is at most* $\log^{O(1)} n \cdot (u_G(p))^2$ *whp.*

Proof. Let X be the estimator given by the overall algorithm. We use induction on recursion depth to prove $\mathbb{E}[X^2] \leq (\log^K n) \cdot (u_G(p))^2$ for some constant $K \geq 1$.

Consider the base cases of the recursion. In the first case (Karger's algorithm (Theorem 1.2) the relative variance is $O(1)$. In the second case (Monte Carlo sampling), we get an unbiased estimator of $u_G(p)$ with relative variance $O(1)$ by Lemmas 1.1 and 1.2. In the last case (importance sampling on spanning tree packing), we get an estimator of $u_G(p)$ with relative bias 0.1ε (by Lemma 4.5) and relative variance $O(1)$ (by Lemma 1.3). For such an estimator X', we have $\mathbb{E}[X'] \leq (1+0.1\varepsilon)u$ and $\mathbb{E}[X'^2] \leq O(1) \cdot \mathbb{E}[X']^2 \leq O(1) \cdot (1+0.1\varepsilon)^2 (u_G(p))^2 = O(1) \cdot (u_G(p))^2$.

Therefore, the statement of the lemma holds for all the base cases.

Next we consider the inductive step where the algorithm takes average of two recursive calls. Let X_1 and X_2 be the outputs of the two recursive calls, so that $X = \frac{X_1 + X_2}{2}$. Again, we have two cases depending on whether we are in a contraction step or a sparsification step. First, we consider a contraction step. The sparsification step is similar and handled at the end.

Let H_1 and H_2 be the two random contracted graphs. Now, we have

$$\mathbb{E}[X^2] = \mathbb{E}\left[\left(\frac{X_1 + X_2}{2}\right)^2\right] = \mathbb{E}_{H_1, H_2}\left[\frac{1}{4}(\mathbb{E}[X_1^2|H_1] + \mathbb{E}[X_2^2|H_2]) + \frac{1}{2} \cdot \mathbb{E}[X_1|H_1] \cdot \mathbb{E}[X_2|H_2]\right],$$

since X_1, X_2 are respectively independent of H_2, H_1.

By the inductive hypothesis, $\mathbb{E}[X_i^2|H_i] \leq (\log^K n_i) \cdot (u_{H_i}(p/q))^2$ for $i = 1, 2$. By applying Lemma 4.5 on the recursive calls, we have

$$\mathbb{E}[X_i|H_i] \leq (1 + 0.1\varepsilon) \cdot u_{H_i}(p/q).$$

We can now bound the second moment by

$$\mathbb{E}[X^2] \leq \frac{1}{4} \cdot \mathbb{E}_{H_1, H_2}\left[(\log^K n_1) \cdot (u_{H_1}(p/q))^2 + (\log^K n_2) \cdot (u_{H_2}(p/q))^2\right]$$

$$+ \frac{1}{2}(1 + 0.1\varepsilon)^2 \cdot \mathbb{E}_{H_1, H_2}[u_{H_1}(p/q) \cdot u_{H_2}(p/q)]$$

$$\leq \frac{1}{4} \cdot (\log^K n_1) \cdot \mathbb{E}_{H_1}[(u_{H_1}(p/q))^2] + \frac{1}{4} \cdot (\log^K n_2) \cdot \mathbb{E}_{H_2}[(u_{H_2}(p/q))^2]$$

$$(4.9) \qquad + \mathbb{E}_{H_1}[u_{H_1}(p/q)] \cdot \mathbb{E}_{H_2}[u_{H_2}(p/q)],$$

by independence of H_1, H_2 and since $\varepsilon < 1$.

To bound the first term in Equation (4.9), note that by Corollary 4.1, we have

$$\mathbb{E}[(u_{H_i}(p/q))^2] \leq \left(2 + O\left(\frac{1}{\log n}\right)\right)(u_G(p))^2 \text{ for } i = 1, 2.$$

To bound the second term in Equation (4.9), note that $\mathbb{E}[u_{H_i}(p/q)] = u_G(p)$ by Lemma 4.4. Thus,

$$E_{H_1}[u_{H_1}(p/q)] \cdot \mathbb{E}_{H_2}[u_{H_2}(p/q)] = (u_G(p))^2.$$

Putting the two terms together, we get

$$\mathbb{E}[X^2] \leq \frac{1}{4}(\log^K n_1 + \log^K n_2)\left(2 + O\left(\frac{1}{\log n}\right)\right)(u_G(p))^2 + (u_G(p))^2.$$

By Lemma 4.3, $n_i \leq (2^{-2/3} + n^{-0.1}) \cdot n$ whp. Then, for a fixed constant $c \in (0, 1)$, we have $\log^K n_i \leq (\log n - c)^K$ for $i = 1, 2$. Continuing the bound on $\mathbb{E}[X^2]$,

$$\mathbb{E}[X^2] \leq \frac{1}{2} \cdot (\log n - c)^K \left(2 + O\left(\frac{1}{\log n}\right)\right)(u_G(p))^2 + (u_G(p))^2$$

$$\frac{\mathbb{E}[X^2]}{(u_G(p))^2} \leq (\log n - c)^K \left(1 + O\left(\frac{1}{\log n}\right)\right) + 1 \leq \log^K n \cdot \left(1 - \frac{c}{\log n}\right)^K \cdot \left(1 + \frac{c}{\log n}\right)^{O(1)} \leq \log^K n,$$

for a large enough K and n larger than some constant.

For a sparsification step, we can repeat the same argument after replacing H by $\tilde{G}$ and $u_H(p/q)$ by $u_{\tilde{G}}(q)$. By Lemma 5.1, $u_{\tilde{G}}(q)$ is also an unbiased estimator of $u_G(p)$. The bound for relative second moment of $u_{\tilde{G}}(q)$ is $2 + O\left(\frac{1}{\log n}\right)$ by Corollary 5.1, which is as good as Corollary 4.1. □

4.4 Running Time of the Algorithm We first establish the following property enforced by sparsification steps.

LEMMA 4.10. *Assume $m > n^{1.5-\xi}$ and the algorithm executes a sparsification step followed by a contraction step. Let m' be the number of edges in a resulting graph. Then, the following bounds hold in expectation: $m' = \tilde{O}(n)$ and $\log^2 m' \leq 2^{-2/3} \cdot \log^2 m$.*

Proof. Sparsification generates a graph $\tilde{G}$ with min-cut value $\tilde{\lambda} = \tilde{O}(1)$. Then we perform random contraction on $\tilde{G}$ with $q^\gamma = 1/2$, where $\gamma \leq \tilde{\lambda}$. Then $q^{\tilde{\lambda}} \leq q^\gamma = 1/2$ and $\frac{1}{1-q} = O(\tilde{\lambda})$. By Lemma 2.5, the expected edge size of the graph after contraction (which follows distribution $\tilde{G}(q)$) is at most $\frac{n}{1-q} = O(n\tilde{\lambda}) = \tilde{O}(n)$. (Note that n is the vertex size before sparsification.)

The assumption gives $m > n^{1.5-\xi}$. After contraction we have $\mathbb{E}[m'] \leq \tilde{O}(n) \leq \tilde{O}(m^{2/3+\xi})$. Notice that $\log^2(x)$ is concave when $x > e$ and we can apply Jensen's inequality. For $\xi < 0.1$,

$$\mathbb{E}[\log^2 m'] \leq \log^2 \mathbb{E}[m'] \leq \log^2 \tilde{O}(m^{2/3+\xi}) \leq \left(\left(\frac{2}{3} + \xi + o(1)\right)\log m\right)^2 \leq 2^{-2/3} \cdot \log^2 m.$$

□

LEMMA 4.11. *The algorithm runs in $m^{1+o(1)} + \tilde{O}(n^{1.5}\varepsilon^{-1})$ time in expectation.*

Proof. Let $T(n, m)$ be the expected running time of a recursive call on a graph with n vertices and m edges. Recall that there are three types of nodes in the computation tree: base cases, contraction nodes, and sparsification nodes. If the parent of a contraction node is a sparsification node, we call it an *irregular* contraction node; otherwise, the contraction node is called a *regular* contraction node. First, we shortcut all irregular contraction nodes (by making their parent the parent of their children). The running time of all such irregular contraction nodes are accounted for by their parent sparsification nodes. Note that each sparsification node has at most two irregular contraction nodes as children; hence, it accounts for the cost of at most three nodes (itself and its two children

irregular contraction nodes). Because of this transformation, the number of children of a sparsification node can increase to at most 4.

In the rest of the discussion, we assume that the contraction tree has only three types of nodes: sparsification nodes, regular contraction nodes and base cases (which are leaves of the computation tree). In the first base case of $n < O(\varepsilon^{-2} \log^2 \frac{1}{\varepsilon})$, the running time is $\tilde{O}(n^2) = \tilde{O}(n^{1.5}\varepsilon^{-1})$ by Theorem 1.2. The second and third base cases take $m^{1+o(1)} + \tilde{O}(n^{1.5})$ time by Lemmas 1.1 to 1.3. In a sparsification node, the sparsification algorithm in Section 5 takes $O(m)$ time. In a contraction node, constructing a Gomory-Hu tree for unweighted graphs takes time $m^{1+o(1)}$ [1], which dominates all other operations. So the time spent at any node of the computation tree is $m^{1+o(1)} + \tilde{O}(n^{1.5}\varepsilon^{-1})$ in total (including the charge received by a sparsification node from its children irregular contraction nodes).

Lemma 4.3 shows that whp each recursive contraction reduces the vertex size by a factor of $2^{-2/3} + O(n^{-0.1})$. In particular, this holds for a regular contraction node and its children, as well as a sparsification node and its children inherited from an irregular contraction child.

First, we consider regular contraction nodes. Note that these nodes still have at most two children. Moreover, they satisfy $m \le n^{1.5-\xi}$ (which implies $m^{1+o(1)} \le \tilde{O}(n^{1.5})$). Therefore, the recurrence is

$$(4.10) \qquad T(n, m) \le \tilde{O}(n^{1.5}\varepsilon^{-1}) + 2T(2^{-2/3}(1 + n^{-0.1}) \cdot n, m).$$

Now, we consider a non-root sparsification node. This is more complicated because $m > n^{1.5-\xi}$. Recall that the running time incurred at this node (including that inherited from irregular contraction children) is $m^{1+o(1)} + \tilde{O}(n^{1.5}\varepsilon^{-1})$. Of these, the term $\tilde{O}(n^{1.5}\varepsilon^{-1})$ can be handled as in the previous case, namely it appears in the recursion. The other term $m^{1+o(1)}$ is charged to the parent of the sparsification node. Let $\hat{n}, \hat{m}$ respectively represent the number of vertices and edges in the parent node. If the parent is a regular contraction node, then we have $m \le \hat{m} \le \hat{n}^{1.5-\xi}$. In this case, we can charge the $m^{1+o(1)}$ term to the parent's recurence relation Equation (4.10). Otherwise, the parent is a sparsification node and we have $m = \tilde{O}(\hat{n})$ in expectation by Lemma 4.10. So, we can also charge $m^{1+o(1)}$ to $\tilde{O}(\hat{n}^{1.5})$. Finally, note that a sparsification node has at most 4 children. Let m' denote the number of edges in any child of the sparsification node. We can write the following recurrence for a sparsification node:

$$(4.11) \qquad T(n, m) \le \tilde{O}(n^{1.5}\varepsilon^{-1}) + 4T(2^{-2/3}(1 + n^{-0.1}) \cdot n, m'), \quad \text{where } m' \text{ satisfies Lemma 4.10.}$$

For the sake of the master theorem, we define a potential $\rho = n \cdot (1 + n^{-0.1}) \cdot \log^2 m$. Lemma 4.10 measures the progress in $\log^2 m$ for the second recurrence (Equation (4.11)) by $\log^2 m' \le 2^{-2/3} \cdot \log^2 m$. We have

$$T(\rho) = \tilde{O}(\rho^{1.5}\varepsilon^{-1}) + 2T(2^{-2/3}\rho) \text{ or } T(\rho) = \tilde{O}(\rho^{1.5}\varepsilon^{-1}) + 4T(2^{-4/3}\rho)$$

The solution is $T(\rho) = \tilde{O}(\rho^{1.5}\varepsilon^{-1})$, or $T(n, m) = \tilde{O}(n^{1.5}\varepsilon^{-1})$.

Finally, note that the root node takes $m^{1+o(1)} + \tilde{O}(n^{1.5}\varepsilon^{-1})$ time (and unlike non-root nodes, the $m^{1+o(1)}$ term isn't chargeable elsewhere). Therefore, the overall running time is $m^{1+o(1)} + \tilde{O}(n^{1.5}\varepsilon^{-1})$ in expectation. ∎

Lemma 1.4 now follows from Lemmas 4.5, 4.9 and 4.11.

5 Sparsification in Recursive Contraction

This section describes the sparsification step, which is used in recursive contraction at the root of the computation tree, or when the edge size $m > n^{1.5-\xi}$. The goal is to reduce the minimum cut value to $O(\log^3 n) = \tilde{O}(1)$.

We apply the sparsification lemma (Lemma 2.4) with parameter $\delta = O\left(\frac{1}{\log n}\right)$ to obtain a sparsifier graph $\tilde{G}$. The graph $\tilde{G}$ is generated by picking each edge independently with probability $\alpha = \Theta\left(\frac{\log^3 n}{\lambda}\right)$ from G. $\tilde{G}$ has min-cut value $\tilde{\lambda} = O(\log^3 n)$, which is our desired property.

In the rest of this section, we show that there is a value q such that $u_{\tilde{G}}(q)$ is an unbiased estimator of $u_G(p)$ with $O(1)$ relative variance. This would allow us to focus on estimating $u_{\tilde{G}}(q)$ in the recursive contraction algorithm.

Choose q such that $1 - q = \frac{1-p}{\alpha}$. Note that $q < p$ by the following argument: since $\lambda > \Omega(\log^3 n)$, we have $\alpha < 1$, which implies $1 - q > 1 - p$ and $q < p$.

LEMMA 5.1. *Assume $\tilde{G}$ is generated by picking every edge in G independently with probability α. Also, suppose q is chosen such that $1 - q = \frac{1-p}{\alpha}$. Then, $u_{\tilde{G}}(q)$ is an unbiased estimator of $u_G(p)$.*

Proof. Keeping each edge with probability $1 - p$ is equivalent to first choosing it with probability α and then keeping it with probability $1 - q = \frac{1-p}{\alpha}$. □

Our goal now is to bound the relative variance of $u_{\tilde{G}}(q)$. Instead, we will bound the relative variance of $z_{\tilde{G}}(q)$ and invoke Lemma 4.6. We now bound the relative second moment of $z_{\tilde{G}}(q)$:

LEMMA 5.2. *When $n^{-3} \leq p^{\lambda} \leq n^{-0.5}$, we have that $z_{\tilde{G}}(q)$ is an unbiased estimator of $z_G(p)$ with relative second moment at most $2 + O\left(\frac{1}{\log n}\right)$.*

Proof. Since $p^{\lambda} \geq n^{-3}$, we have $1 - p \leq 1 - e^{-3 \ln n / \lambda} \leq O\left(\frac{\log n}{\lambda}\right)$. Similarly, $p^{\lambda} \leq n^{-0.5}$ implies that $1 - p \geq 1 - e^{-0.5 \ln n / \lambda} \geq \Omega\left(\frac{\log n}{\lambda}\right)$. Therefore, denote

$$\tau = 1 - p = \Theta\left(\frac{\log n}{\lambda}\right).$$

Recall from above that we defined q so as to satisfy

$$1 - q = \frac{\tau}{\alpha} = \Theta\left(\frac{\frac{\log n}{\lambda}}{\frac{\log^3 n}{\lambda}}\right) = \Theta\left(\frac{1}{\log^2 n}\right).$$

Let Y_e be the indicator that edge e is picked by the random graph $\tilde{G}$. For any edge e, we have

(5.12) $$\mathbb{E}\left[q^{Y_e}\right] = \alpha q + (1 - \alpha) = 1 - \alpha(1 - q) = p$$

(5.13) $$\mathbb{E}\left[q^{2Y_e}\right] = \alpha q^2 + (1 - \alpha) = 1 - \tau(1 + q) \leq (1 - \tau)(1 - \tau q) = p \cdot (1 - \tau q).$$

We can bound $\mathbb{E}\left[q^{2Y_e}\right]$ in two ways:

(5.14) $$\frac{\mathbb{E}\left[q^{2Y_e}\right]}{p} \leq 1 - \tau q$$

(5.15) $$\frac{\mathbb{E}\left[q^{2Y_e}\right]}{p^2} \leq \frac{1 - \tau q}{p} = \frac{(1 - \tau) + \tau(1 - q)}{1 - \tau} = 1 + \frac{\tau(1 - q)}{1 - \tau} \leq 1 + 2\tau(1 - q) = 1 + O\left(\frac{1}{\lambda \log n}\right).$$

Next we calculate the expectation and relative variance of $z_{\tilde{G}}(q)$. Notice that Y_e's are independent for each edge e. Use $C_i \Delta C_j$ to denote the symmetric difference $(C_i \setminus C_j) \cup (C_j \setminus C_i)$ over two cuts C_i, C_j. Use $\tilde{d}(\cdot)$ to denote the cut value function in $\tilde{G}$. First, we calculate the expectation of $z_{\tilde{G}}(q)$:

$$\mathbb{E}[z_{\tilde{G}}(q)] = \mathbb{E}\left[\sum_{C_i} q^{\tilde{d}(C_i)}\right] = \sum_{C_i} \mathbb{E}\left[q^{\sum_{e \in C_i} Y_e}\right] = \sum_{C_i} \prod_{e \in C_i} \mathbb{E}\left[q^{Y_e}\right] \stackrel{(5.12)}{=} \sum_{C_i} p^{c_i} = z_G(p).$$

Next, we bound the second moment of $z_{\tilde{G}}(q)$

$$\mathbb{E}\left[(z_{\tilde{G}}(q))^2\right] = \mathbb{E}\left[\sum_{C_i} \sum_{C_j} q^{\tilde{d}(C_i) + \tilde{d}(C_j)}\right] = \sum_{C_i} \sum_{C_j} \mathbb{E}\left[q^{\sum_{e \in C_i} Y_e + \sum_{e \in C_j} Y_e}\right]$$

$$= \sum_{C_i} \sum_{C_j} \prod_{e \in C_i \cap C_j} \mathbb{E}\left[q^{2Y_e}\right] \prod_{e \in C_i \Delta C_j} \mathbb{E}\left[q^{Y_e}\right] \stackrel{(5.12)}{=} \sum_{C_i} \sum_{C_j} p^{|C_i \Delta C_j|} \cdot \left(\mathbb{E}\left[q^{2Y_e}\right]\right)^{|C_i \cap C_j|}.$$

We partition this sum into three parts and separately bound their ratios with $(z_G(p))^2$.

For terms with $C_i = C_j$,

$$\frac{\sum_{C_i} p^{|C_i \Delta C_i|} \cdot \left(\mathbb{E}\left[q^{2Y_e}\right]\right)^{|C_i \cap C_i|}}{(z_G(p))^2} = \frac{\sum_{C_i} \left(\mathbb{E}\left[q^{2Y_e}\right]\right)^{c_i}}{(z_G(p))^2} = \frac{\sum_{C_i} p^{c_i} \cdot \left(\frac{\mathbb{E}\left[q^{2Y_e}\right]}{p}\right)^{c_i}}{\sum_{C_i} p^{c_i} \cdot z_G(p)} \leq \max_{C_i} \frac{\left(\frac{\mathbb{E}\left[q^{2Y_e}\right]}{p}\right)^{c_i}}{z_G(p)}$$

$$(5.16) \qquad \overset{(5.14)}{\leq} \frac{(1-\tau q)^{\lambda}}{z_G(p)} \leq \left(\frac{1-\tau q}{p}\right)^{\lambda} \overset{(5.15)}{\leq} \left(1 + O\left(\frac{1}{\lambda \log n}\right)\right)^{\lambda} = 1 + O\left(\frac{1}{\log n}\right).$$

For terms with $|C_i \cap C_j| \leq \lambda$,

$$\frac{\sum_{|C_i \cap C_j| \leq \lambda} p^{|C_i \Delta C_j|} \cdot \left(\mathbb{E}\left[q^{2Y_e}\right]\right)^{|C_i \cap C_j|}}{(z_G(p))^2} = \frac{\sum_{|C_i \cap C_j| \leq \lambda} p^{c_i + c_j} \cdot \left(\frac{\mathbb{E}\left[q^{2Y_e}\right]}{p^2}\right)^{|C_i \cap C_j|}}{(z_G(p))^2}$$

$$\overset{(5.15)}{\leq} \frac{\sum_{|C_i \cap C_j| \leq \lambda} p^{c_i + c_j} \left(1 + O\left(\frac{1}{\lambda \log n}\right)\right)^{\lambda}}{\sum_{C_i, C_j} p^{c_i + c_j}} \leq \left(1 + O\left(\frac{1}{\lambda \log n}\right)\right)^{\lambda} = 1 + O\left(\frac{1}{\log n}\right).$$

For terms with $|C_i \cap C_j| > \lambda$ and $C_i \neq C_j$, we have

$$\frac{\sum_{C_i \neq C_j, |C_i \cap C_j| > \lambda} p^{|C_i \Delta C_j|} \cdot \mathbb{E}\left[q^{2Y_e}\right]^{|C_i \cap C_j|}}{(z_G(p))^2} = \frac{\sum_{C_i \neq C_j, |C_i \cap C_j| > \lambda} p^{|C_i \cup C_j|} \cdot \left(\frac{\mathbb{E}\left[q^{2Y_e}\right]}{p}\right)^{|C_i \cap C_j|}}{(z_G(p))^2}$$

$$\overset{(5.14)}{\leq} \frac{\sum_{C_i \neq C_j, |C_i \cap C_j| > \lambda} p^{|C_i \cup C_j|} \cdot (1-\tau q)^{\lambda}}{(z_G(p))^2} \leq \frac{x_G(p) \cdot (1-\tau q)^{\lambda}}{(z_G(p))^2} \qquad \text{(by definition of } x_G(p)\text{)}.$$

Applying $\frac{x_G(p)}{z_G(p)} \leq \frac{1}{\log n}$ from Lemma 2.1, this is at most

$$\frac{1}{\log n} \cdot \frac{(1-\tau q)^{\lambda}}{z_G(p)} \overset{(5.16)}{\leq} \frac{1}{\log n} \cdot \left(1 + O\left(\frac{1}{\log n}\right)\right) \leq O\left(\frac{1}{\log n}\right).$$

In conclusion, the total relative variance is given by

$$\frac{\mathbb{E}[z_{\tilde{G}}(q)^2]}{(z_G(p))^2} \leq \left(1 + O\left(\frac{1}{\log n}\right)\right) + \left(1 + O\left(\frac{1}{\log n}\right)\right) + O\left(\frac{1}{\log n}\right) = 2 + O\left(\frac{1}{\log n}\right).$$

□

Combining Lemmas 4.6 and 5.2, we obtain the following:

COROLLARY 5.1. $u_{\tilde{G}}(q)$ *is an unbiased estimator of* $u_G(p)$ *with relative second moment at most* $2 + O\left(\frac{1}{\log n}\right)$.

6 Monte Carlo Sampling

Finally, we use Monte Carlo sampling for the unreliable case. We use it in two different ways: naïve Monte Carlo sampling and two-step recursive Monte Carlo sampling. These two algorithms respectively handle the cases $p \geq \theta$ and $n^{-\frac{1}{2\lambda}} < p < \theta$, for $u_G(\theta) = n^{-O(1/\log \log n)}$. (The precise value of θ is the one given in Lemma 2.1.) Note that in conjunction with the previous sections, this covers all possibilities.

6.1 Naïve Monte Carlo Sampling
When $p \geq \theta$, we run a naïve Monte Carlo sampling algorithm. Our goal is to show Lemma 1.1, which we restate below:

LEMMA 1.1. *For any* $p \geq \theta$, *an unbiased estimator of* $u_G(p)$ *with relative variance* $O(1)$ *can be computed in time* $m^{1+o(1)}$. *As a consequence, a* $(1 \pm \varepsilon)$-*approximation to* $u_G(p)$ *can be computed in* $m^{1+o(1)}\varepsilon^{-2}$ *time under the condition that* $p \geq \theta$.

In each round of this algorithm, we run the following sampling process: remove each edge independently with probability p and check whether the graph gets disconnected. The corresponding indicator variable X is a Bernoulli random variable with parameter $u_G(p)$. The expectation, variance, and relative variance of this variable are given below:

LEMMA 6.1. *For a single round of Monte Carlo sampling, the mean, variance, and relative variance of the estimator are given by* $\mathbb{E}[X] = u_G(p)$, $\mathbb{V}[X] = u_G(p)(1 - u_G(p))$ *and* $\eta[X] = \frac{u_G(p)(1-u_G(p))}{(u_G(p))^2} \leq \frac{1}{u_G(p)}$.

By repeated Monte Carlo sampling for $\tilde{O}(1/u_G(p))$ independent rounds, we can reduce the relative variance to $O(1)$ by Fact 2.1. The running time of each round is $O(m)$, so the total running time is $\tilde{O}(m/u_G(p))$. Note that $u_G(p) \geq u_G(\theta) = n^{-o(1)}$; therefore, the running time is $m^{1+o(1)}$.

This completes the proof of Lemma 1.1.

6.2 Two-step Monte Carlo Sampling When $p < \theta$ and $p > n^{-\frac{1}{2\lambda}}$, we use a two-step Monte Carlo sampling algorithm. This follows a sparse sampling technique used by Karger [12]. Our goal is to show Lemma 1.2, which we restate below:

LEMMA 1.2. *For p such that $p < \theta$ and $p^\lambda > n^{-1/2}$, an unbiased estimator of $u_G(p)$ with relative variance $O(1)$ can be computed in $\tilde{O}(m + n^{1.5})$ time. As a consequence, a $(1 \pm \varepsilon)$-approximation to $u_G(p)$ can be computed in $\tilde{O}((m + n^{1.5})\varepsilon^{-2})$ time under the condition that $p < \theta$ and $p^\lambda > n^{-1/2}$.*

To describe this algorithm, we first note that instead of removing each edge with probability p and checking if the graph is disconnected, we can equivalently *contract* every edge with probability $1 - p$ and check if we get more than one vertex. Instead of doing this in one shot, we stage this contraction process out into two steps: in the first step, for some $q > p$, we contract each edge with probability $1 - q$ to form a graph H (i.e., $H \sim G(q)$), and then in the second step, we contract each edge in H with probability $1 - p/q$. Note that the indicator variable for obtaining > 1 vertex at the end of this two step contraction process is an unbiased estimator for $u_H(p/q)$, and since $\mathbb{E}[u_H(p/q)] = u_G(p)$, is also an unbiased estimator of $u_G(p)$.

But, what do we gain in this two-step process? To understand this, we need to bound the running time for each of the two steps. In each step, we bound the running time for a single round of Monte Carlo sampling and also the relative variance of the resulting estimator, which in turn bounds the number of rounds by Lemma 1.1.

For the first step, we use a naïve implementation of Monte Carlo sampling in $O(m)$ time. To ensure efficiency in terms of the number of rounds of sampling, we need to choose q to be large enough such that the relative variance at the end of this step is small. We choose q such that $q^\lambda = 1/2$. Note that if $q^\lambda = 1/2$ and $p < \theta$ (for θ given in Lemma 2.1), then

$$q^\lambda = \frac{1}{2} > n^{-O(1/\log\log n)} = u_G(\theta) \geq u_G(p) \geq p^\lambda;$$

thus $q > p$ as required. Furthermore, since $p \leq \theta$, we can apply Lemma 4.7. Note that $p^\lambda \leq u_G(p) \leq u_G(\theta) = n^{-O(1/\log\log n)}$ (by Lemma 2.1). Thus,

$$O\left(\frac{1}{\log p^{-\lambda}}\right) = O\left(\frac{\log\log n}{\log n}\right) = o(1).$$

Since $q^{-\lambda} = 2$, Lemma 4.7 implies that the relative variance of $u_H(p/q)$ over the randomness of H is at most $2 + o(1) = O(1)$.

Now, we consider the second step of Monte Carlo sampling. Here, we naïvely bound the relative variance of the estimator by the relative variance of the overall estimator, which by Lemma 6.1 is given by $1/u_G(p)$. Now, since $u_G(p) \geq p^\lambda > n^{-1/2}$, we get that the relative variance of the estimator in the second step is at most $\sqrt{n}$. Since we are aiming for a running time of $\tilde{O}(m + n^{3/2})$, we must give an implementation of the second step of Monte Carlo sampling in $\tilde{O}(n)$ time. Crucially, because of edge contractions, the number of edges in H generated after the first step of sampling is at most $\frac{n}{1-q}$ in expectation (by Lemma 2.5). By our choice of q that ensures $q^\lambda = 1/2$, Lemma 2.5 implies that in expectation, H contains $O(n\lambda)$ edges. So, a naïve implementation of Monte Carlo sampling takes $O(n\lambda)$ time per round in expectation. But, this is still not enough since we are aiming for

a running time of $\tilde{O}(n)$ per round. Recall that the probability of contracting an edge in H is $1 - p/q$, which can be bounded as follows (the second inequality uses $p^\lambda > n^{-1/2}$):

$$1 - \frac{p}{q} < 1 - p \le 1 - e^{-0.5 \ln n / \lambda} \le \frac{0.5 \ln n}{\lambda} = O\left(\frac{\log n}{\lambda}\right).$$

Now, since H only has $O(n\lambda)$ edges in expectation, it follows that the expected number of contracted edges in H is $O(n \log n)$.

Instead of iterating over all edges in H, we directly choose the edges to contract and check if they contain a spanning tree over the set of vertices in H. The first step is to determine the number of edges to contract. Note that this is a Binomial random variable with parameters $|E(H)|$ and $1 - p/q$. Once we have selected the number of contracted edges by generating the Binomial random variable, we must then select these edges to contract uniformly at random from the edges in H. We choose edges uniformly at random (with replacement, discarding if we get a duplicate) until we have the desired number of edges. This algorithm requires $\tilde{O}(1)$ time per contracted edge, which is $\tilde{O}(n)$ time in expectation overall.

We now describe our overall algorithm. For each round of the first sampling step that reduces G to H, we use $O(\sqrt{n})$ independent rounds of the second step of Monte Carlo sampling on H. This gives an estimator of $u_H(p/q)$ with relative variance 1, and takes $\tilde{O}(m + n^{1.5})$ time. The overall relative variance of the estimator of $u_G(p)$ is now $O(1)$ by Lemma 2.7. So, we now invoke Lemma 2.6 by repeating this sampling for $\tilde{O}(\varepsilon^{-2})$ rounds to get a $(1 \pm \varepsilon)$-approximate estimate of $u_G(p)$ in $\tilde{O}((m + n^{1.5})\varepsilon^{-2})$ time overall.

This completes the proof of Lemma 1.2.

7 Conclusion

We obtain an algorithm for the network unreliability problem that runs in $m^{1+o(1)} + \tilde{O}(n^{1.5})$ time and improves on the previous best running time of $\tilde{O}(n^2)$. Our main technical contribution is a new algorithm for estimating unreliability in the *very reliable* situation, which is normally the bottleneck for unreliability algorithms. Our algorithm utilizes a carefully defined importance sampling procedure on a collection of cuts defined via a spanning tree packing of the graph. In addition, for the *moderately reliable* setting, we give a new, improved analysis for (a version of) the recursive contraction algorithm. Our algorithm is almost-linear time for dense instances $(m = \Omega(n^{1.5}))$. Obtaining an almost linear running time for all instances is the natural eventual goal. But, this will require new ideas that go beyond the techniques described in this paper which are optimized to obtain the $\tilde{O}(n^{1.5})$ bound.

Our result holds for the *uniform* case where every edge fails with the same probability p. A more general setting is when each edge e fails with a different probability p_e. This can be simulated in the uniform setting by using multiple parallel edges for each edge, and our algorithm works for this simulated graph but at the cost of an increase in the running time because of the dependence on the number of edges m.

The problem of estimating reliability, i.e., the probability that the graph *stays connected* under edge failures, is equivalent to the unreliability problem when we seek exact computation. However, for approximation algorithms, the problems are different. For approximating reliability, the bottleneck graphs are the very unreliable ones, and require an entirely different set of techniques. For the reliability problem, the current best running time is $\tilde{O}\left(\frac{mn^2}{\varepsilon^2(1-p)}\right)$ due to Guo and He [7]. Improving this bound is also an interesting question in the general area of network reliability.

References

[1] Amir Abboud, Robert Krauthgamer, Jason Li, Debmalya Panigrahi, Thatchaphol Saranurak, and Ohad Trabelsi. Breaking the cubic barrier for all-pairs max-flow: Gomory-hu tree in nearly quadratic time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 884–895, 2022.

[2] Noga Alon, Alan M. Frieze, and Dominic Welsh. Polynomial time randomized approximation schemes for tutte-gröthendieck invariants: The dense case. *Random Struct. Algorithms*, 6(4):459–478, 1995.

[3] Sanjay Kumar Chaturvedi. *Network reliability: measures and evaluation.* John Wiley & Sons, 2016.

[4] Charles J Colbourn. *The combinatorics of network reliability.* Oxford University Press, Inc., 1987.

[5] H.N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *Journal of Computer and System Sciences*, 50(2):259–273, 1995.

[6] Ralph E Gomory and Tien Chung Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[7] Heng Guo and Kun He. Tight bounds for popping algorithms. *Random Structures & Algorithms*, 57(2):371–392, 2020.

[8] David G. Harris and Aravind Srinivasan. Improved bounds and algorithms for graph cuts and network reliability. *Random Structures & Algorithms*, 52(1):74–135, 2018.

[9] David R. Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24(2):383–413, 1999.

[10] David R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM Journal on Computing*, 29(2):492–514, 1999.

[11] David R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, 2000.

[12] David R. Karger. A fast and simple unbiased estimator for network (un)reliability. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 635–644, 2016.

[13] David R. Karger. Faster (and still pretty simple) unbiased estimators for network (un)reliability. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 755–766, 2017.

[14] David R. Karger. A phase transition and a quadratic time unbiased estimator for network reliability. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 485–495, New York, NY, USA, 2020. Association for Computing Machinery.

[15] David R. Karger, Philip N. Klein, and Robert E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *J. ACM*, 42(2):321–328, 1995.

[16] David R Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, 1996.

[17] Richard M Karp, Michael Luby, and Neal Madras. Monte-carlo approximation algorithms for enumeration problems. *Journal of algorithms*, 10(3):429–448, 1989.

[18] George S. Lueker. A data structure for orthogonal range queries. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 28–34, 1978.

[19] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

A Additional Proofs

A.1 Proof of Lemma 2.1 To prove this lemma, we use the following lemma shown by Karger [14]:

LEMMA A.1. (THEOREMS 5.1 AND 7.1 OF [14]) *There exist thresholds $0 < s < b < 1$ such that*

1. *When $p < s$, $\frac{x_G(p)}{z_G(p)} \leq (p/s)^{\lambda/2}$.*

2. *$u_G(b) = \frac{1}{2}$, $s^\lambda = \Omega(b^\lambda / \log^2 n)$.*

3. *$z_{G,\alpha}(b) = n^{-O(1/\log \alpha)}$, where $z_{G,\alpha}(b)$ is the expected number of failed $\alpha\lambda$-weak cuts in G when each edge is removed with probability b.*

Recall that $x_G(p)$ is the expected number of failed cut pairs, and $z_G(p)$ is the expected number of failed cuts. A simple inclusion-exclusion on $u_G(p)$ gives

$$z_G(p) - x_G(p) \leq u_G(p) \leq z_G(p).$$

Let s and b be the thresholds provided by Lemma A.1. Set $\theta^\lambda = s^\lambda / \log^2 n$, $\theta < s$. Then, when $p < \theta$,

$$\frac{x_G(p)}{z_G(p)} \leq (p/s)^{\lambda/2} \leq (\theta/s)^{\lambda/2} = \frac{1}{\log n},$$

so the second property holds.

Next we prove the first property. $\theta^\lambda = s^\lambda / \log^2 n = \Omega(b^\lambda / \log^4 n)$.

$$u_G(\theta) \geq \left(1 - \frac{1}{\log n}\right) z_G(\theta) \geq \frac{1}{2} z_{G,\alpha}(\theta)$$

$$= \frac{1}{2} \sum_{C_i : c_i \leq \alpha\lambda} \theta^{c_i} \geq \frac{1}{2} \sum_{C_i : c_i \leq \alpha\lambda} b^{c_i} \left(\frac{\theta}{b}\right)^{\alpha\lambda}$$

$$\geq z_{G,\alpha}(b) \cdot (\Omega(\log^{-4} n))^\alpha \geq n^{-O(1/\log \alpha)} \cdot e^{-O(\alpha \log \log n)}$$

Set $\alpha = O(\log n / (\log \log n)^2)$, then $n^{-O(1/\log \alpha)} = n^{-O(1/\log \log n)}$, and $e^{-O(\alpha \log \log n)} = n^{-O(1/\log \log n)}$. In conclusion $u_G(\theta) = n^{-O(1/\log \log n)}$. So, the first property holds.

A.2 Proof of Lemma 2.2 We first state a simple fact about the ratio p^λ, the probability that a specific min cut fails, and $u_G(p)$, the probability that any cut fails.

LEMMA A.2. (COROLLARY II.2 OF [12]) *For a graph G with n vertices and min cut value λ,*

$$p^\lambda \leq u_G(p) \leq n^2 p^\lambda.$$

By Lemma A.2 and the definition of b in Lemma A.1, we have

$$1/2 = u_G(b) \leq n^2 b^\lambda.$$

Therefore, $s^\lambda = \Omega(b^\lambda / \log^2 n) = \Omega(n^{-2} / \log^2 n)$. When $p^\lambda = O(n^{-3})$, we have

$$(p/s)^{\lambda/2} = \sqrt{p^\lambda / s^\lambda} \leq \sqrt{\frac{O(n^{-3})}{\Omega\left(\frac{n^{-2}}{\log^2 n}\right)}} = O\left(\frac{\log n}{\sqrt{n}}\right).$$

The lemma follows by plugging this bound into the first property of Lemma A.1.

A.3 Proof of Lemma 3.2 We use Gabow's algorithm that packs directionless spanning trees in a directed graph D. A directionless spanning tree of D is a subgraph of D such that if we replace all its edges with undirected edges, we get a tree spanning all vertices of D.

LEMMA A.3. ([5]) *Given a directed graph D with min-cut value λ, we can construct in $O(\lambda m \log(n^2/m))$ time a packing of λ edge-disjoint directionless spanning trees.*

Given an undirected graph G, we construct a directed graph D by replacing every undirected edge by two directed edges in opposite directions. Then the min-cut value in D is the same as the min-cut value λ in G. We apply Lemma A.3 to get λ edge-disjoint directionless spanning trees in D. After that, remove the directions of all edges in the trees to form a tree packing $\mathcal{T}$ in G. Removing direction maps two edges in D into one edge in G. Therefore, each edge of G is used by at most two trees in $\mathcal{T}$.

A.4 Proof of Fact 3.1 Among the N^j possible sampling sequences, we count the number of sequences that form set A. Such a sequence can be generated by first partitioning the j elements into α sets, then choosing a bijection between these sets and elements in A. The first step has $S(j, \alpha)$ possibilities, and the second step has $\alpha!$ choices. So, the total number of sequences that yield A is given by $\alpha! \, S(j, \alpha)$. The first part of the lemma now follows since in uniform sampling with replacement, all N^j sequences are equally likely. For the second part, we note that when $j = 2$, we have $S(j, \alpha) = 1$ for $\alpha \in \{1, 2\}$ and $S(j, \alpha) = 0$ otherwise.

A.5 Proof of Fact 4.2 Since $t \geq \frac{5}{\log n} = \frac{5 \ln 2}{\ln n} > \frac{3}{\ln n}$, we have $2^t - 1 \geq t \ln 2 > \frac{3 \ln 2}{\ln n}$. This implies $\frac{2^t}{2^t - 1} = 1 + \frac{1}{2^t - 1} < 1 + \frac{\ln n}{3 \ln 2}$, and $\frac{2^t \ln 2}{2^t - 1} < \ln 2 + \frac{1}{3} \ln n$. When n is greater than some constant, we have $\frac{2^t \ln 2}{2^t - 1} < 0.375 \ln n$. Therefore,

$$\frac{df}{dt} = (2u)^t \ln(2u) - u^t \ln u = u^t \left(2^t \ln 2 - (2^t - 1) \ln \frac{1}{u}\right) \leq u^t (2^t \ln 2 - (2^t - 1) 0.375 \ln n) < 0.$$