Network Unreliability in Almost-Linear Time

Ruoxu Cen
Duke University
Durham, USA
ruoxu.cen@duke.edu

Jason Li
Carnegie Mellon University
Pittsburgh, USA
jmli@cs.cmu.edu

Debmalya Panigrahi
Duke University
Durham, USA
debmalya@cs.duke.edu

Abstract

The network unreliability problem asks for the probability that a given undirected graph gets disconnected when every edge independently fails with a given probability p. Valiant (1979) showed that this problem is #P-hard; therefore, the best we can hope for are approximation algorithms. In a classic result, Karger (1995) obtained the first FPTAS for this problem by leveraging the fact that when a graph disconnects, it almost always does so at a near-minimum cut, and there are only a small (polynomial) number of near-minimum cuts. Since then, a series of results have obtained progressively faster algorithms to the current bound of $m^{1+o(1)} + \tilde{O}(n^{3/2})$ (Cen, He, Li, and Panigrahi, 2024). In this paper, we obtain an $m^{1+o(1)}$-time algorithm for the network unreliability problem. This is essentially optimal, since we need $O(m)$ time to read the input graph. Our main new ingredient is relating network unreliability to an *ideal* tree packing of spanning trees (Thorup, 2001).

CCS Concepts

• **Theory of computation** → **Graph algorithms analysis**; • **Networks** → **Network reliability**.

Keywords

Network reliability, Importance sampling, Ideal tree packing

ACM Reference Format:
Ruoxu Cen, Jason Li, and Debmalya Panigrahi. 2025. Network Unreliability in Almost-Linear Time. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC '25), June 23–27, 2025, Prague, Czechia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3717823.3718140

1 Introduction

The unreliability of an undirected graph $G = (V, E)$, denoted $u_G(p)$, is the probability that it gets disconnected when every edge independently fails with a given probability p. It measures the robustness of a network to random edge failures (as against worst case failures, measured by the minimum cut). Reliability problems are extensively studied (see the books [4, 6]) and the problem of estimating $u_G(p)$ has been dubbed "the most fundamental" problem in this space [14]. This problem was shown to be #P-hard by Valiant in 1979 [20]. The first fully polynomial-time randomized approximation scheme (FPRAS) was given in a seminal work by Karger [11]. 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. The running time was later improved by Harris and Srinivasan [10] to $n^{3+o(1)}$ using a finer-grained analysis of Karger's algorithmic framework. Further improvements came in a series of works by Karger [12–14] that utilized progressively optimized versions of a recursive random contraction framework to eventually attain a running time of $\tilde{O}(n^2)$. This quadratic bound is a natural threshold because the techniques developed for network unreliability could also (implicitly or explicitly) enumerate the possibly $\Omega(n^2)$ minimum cuts of the graph. The quadratic barrier was recently breached by Cen, He, Li, and Panigrahi [2] who obtained a running time of $m^{1+o(1)} + \tilde{O}(n^{3/2})$ by using an importance sampling technique. The ultimate goal in this line of work, explicitly conjectured by Karger in [14], is to obtain a network unreliability algorithm that runs in (essentially) linear time in the size of the input graph. This last bound is also the best possible, since any network unreliability algorithm must read its entire input.

1.1 Our Result

In this paper, we give a fully polynomial-time randomized approximation scheme (FPRAS) for the network unreliability problem that has a running time of $m^{1+o(1)}$. 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* $m^{1+o(1)}\varepsilon^{-O(1)}$ *time and outputs a* $(1 \pm \varepsilon)$*-approximation to* $u_G(p)$ *whp.*

Up to lower order terms, the above theorem brings the line of research into faster network unrelibility algorithms to a close. One might still hope to improve on the lower order terms: specifically, improve $m^{o(1)}$ to $\text{poly}\log(m)$ in the running time. As we will see, the $m^{o(1)}$ overhead in our running time comes from two sources: The first is the use of maximum flow algorithms, the fastest of which run in $m^{1+o(1)}$ time [5, 21]; this will automatically improve if faster maximum flow algorithms are discovered. The second source is more intrinsic to our method and relates to the amount of variance in the estimator used in our work; it is an interesting question as to whether this variance can be reduced from $m^{o(1)}$ to $\text{poly}\log(m)$ in our framework.

1.2 Our Techniques

Intuitively, we divide into two cases depending on the value of $u_G(p)$: if (roughly) $u_G(p) < 1/n$, we say the graph is reliable, else it is unreliable. Morally, we would like to use Monte Carlo sampling for the unreliable case, i.e., sample edge failure outcomes and use

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

ACM ISBN 979-8-4007-1510-5/25/06
https://doi.org/10.1145/3717823.3718140

an indicator variable for disconnection to estimate $u_G(p)$. But, we cannot do this directly since generating each independent sample takes $\tilde{O}(m)$ time, and we need $\tilde{O}(n)$ samples. Instead, we use the recursive contraction algorithm of Karger and Stein [15] to (intuitively) generate $\tilde{O}(n)$ samples in a correlated manner such that each sample only takes $\tilde{O}(1)$ amortized time. We explain how the recursive contraction algorithm can be used to generate Monte Carlo samples in the next paragraph.

A mechanism for generating a Monte Carlo sample is to randomly order edges, contract a prefix of edges in this order, and ask if the resulting graph is a singleton vertex. If this process is independently repeated multiple times, it produces a set of independent samples. But, now, suppose that instead of creating completely independent random permutations, we create correlations in the following manner: we first select a random set of edges that appear at the head of the permutations for all the samples; then, we create two independent copies and independently sample the next set of edges in these two copies; then, we subdivide further by creating two independent copies of each of the previous copies, and so on. This creates a correlated set of permutations on which we perform contractions to generate samples of the network. The advantage is that this latter process can be implemented very efficiently, because this is exactly the recursive contraction algorithm [15]. Normally, a recursive contraction algorithm would generate $\tilde{O}(n^2)$ samples in $\tilde{O}(n^2)$ time. However, using the assumption that $u_G(p) > 1/n$, we show that $\tilde{O}(n)$ samples generated by this process are sufficient, and that this can be produced in $m^{1+o(1)}$ time.

Next, we consider the reliable case, i.e., when $u_G(p) < 1/n$. Recall that we would like to estimate the probability that at least one cut fails (i.e., all edges are deleted). In the reliable case, this is a rare event and can be well-approximated by the expected number of failed cuts, i.e., the corresponding union bound. Furthermore, the large cuts can be ignored (their failure is unlikely), but we are still left with $\Omega(n^2)$ cuts, e.g., the minimum cuts themselves. To do this in close to $O(m)$ time, we sample a surrogate distribution $q(C)$ that approximates the real distribution $p^{|C|}$. The distribution $q(C)$ is defined via a distribution on spanning trees, from which we sample a spanning tree, and perform a sampling procedure on this tree to define a cut. We describe the spanning tree distribution next.

We use a fractional spanning tree packing that was introduced by Thorup [17, 18]. The total value of the packing equals the min-ratio value of all multiway cuts in the graph, i.e., $\pi := \min \frac{|C|}{k(C)-1}$, where the minimum is taken over all multiway cuts C, and $k(C)$ is the number of sides of C. The packing is also "edge-disjoint" in the sense that each edge is used only in spanning trees that add up to a total value at most 1. This latter property ensures that every edge in the min-ratio cut appears in exactly $1/\pi$ fraction of the trees by value. More generally, the tree packing also ensures that this property holds recursively on the induced subgraph defined on each side of the min-ratio cut, which has a larger cut ratio than π. When everything is settled, any edge e will be in $1/\pi(e)$ fraction of trees for some cut ratio $\pi(e)$. We call $\pi(e)$ the *level* of edge e, and the entire collection of fractional trees is called an *ideal* tree packing.

So, the first step of our sampling is to generate a spanning tree from the ideal tree packing distribution. Next, we need to sample edges in the sampled spanning tree to define the sampled cut. For intuition, let us imagine an idealized scenario where for any cut C, every spanning tree T in the ideal tree packing contains $\frac{|C_i|}{\pi_i}$ edges at level i, where C_i is the set of level-i edges in C. Then, an unbiased estimator for $p^{|C|}$ can be obtained by sampling every edge e in a sampled spanning tree T with probability $p^{\pi(e)}$. The advantage of this process is that we are sampling a small number of edges that appear in the spanning tree, much smaller than the size of the cut, which can be done much more efficiently. This intuition is not precise because edges of a cut are not proportionately distributed across the spanning trees of the ideal tree packing. Nevertheless, we show that the error due to the non-uniformity (i.e., variance) can be bounded.

A final complication is that we need to use an approximation algorithm for ideal tree packing since exact algorithms are too slow for our purpose. This means the levels of edges are now only approximately correct, which leads to dependence between the random sampling process across the different (approximate) levels. We show that this additional error can be controlled for the above arguments to still hold approximately, but in the process, we lose additional sub-polynomial terms in the running time to counter the variance of the estimator.

1.3 Overall Algorithm and Paper Organization

We now give an overview of the algorithm. Our overall algorithm builds a recursive computation tree. There are several base cases that we describe first. The first two base cases also appear in prior work [2, 14]. The interesting (and new) base case is the third one, which is also the main contribution of this paper.

- If $n \le \varepsilon^{-O(1)}$, then any $n^{O(1)}\varepsilon^{-O(1)}$ running time is still $\varepsilon^{-O(1)}$, so we run Karger's algorithm [14] that gives an unbiased estimator for $u_G(p)$. The following theorem states the properties of this estimator (relative variance of a random variable X is $\frac{\mathbb{V}[X]}{(\mathbb{E}[X])^2}$):

  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 when $p \ge \theta$ for some threshold θ whose value is given by Lemma 1.4. The properties of the estimator are summarized below:

  Lemma 1.3 (Lemma 1.3 in [2]). *For any $p \ge \theta$, 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 $p \ge \theta$.*

  We are left to define the value of θ. We set θ such that when $p < \theta$, it is very unlikely that more than one cut fails. To quantify this, define $z_G(p)$ as the expected number of failed cuts $z_G(p) := \sum_{C_i} p^{|C_i|}$ and $x_G(p)$ as the expected number of failed cut pairs $x_G(p) := \sum_{C_i \neq C_j} p^{|C_i \cup C_j|}$. By the inclusion-exclusion principle, $u_G(p) \ge z_G(p) - x_G(p)$. We define θ as follows:

Lemma 1.4 (Phase transition, Lemma 2.1 of [2]). *There exists a threshold θ such that $u_G(\theta) = n^{-\Omega(1/\log\log n)}$ satisfying the following property: When $p < \theta$, we have $\frac{x_G(p)}{z_G(p)} \le \frac{1}{\log n}$, and therefore,*[2]

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

Intuitively, this means that when $p < \theta$, we can use an estimator of $z_G(p)$ as a surrogate for that of $u_G(p)$, although we need to be careful about the $1/\log n$ gap between $z_G(p)$ and $u_G(p)$.

- *The final base case is the most interesting new contribution of this paper.* It is invoked when $u_G(p) < n^{-1-\Omega(1/\log\log n)}$. In this case, we run our new algorithm (Section 3) and 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 1.5. *If $u_G(p) < n^{-1-\Omega(1/\log\log n)}$, an estimator for $u_G(p)$ with relative bias $n^{-\Omega(1)}$ and relative variance ≤ 1 can be computed in $m^{1+o(1)}$ time. As a consequence, under the condition that $u_G(p) < n^{-1-\Omega(1/\log\log n)}$, a $(1 \pm \varepsilon)$-approximation to $u_G(p)$ can be computed in $m^{1+o(1)}\varepsilon^{-O(1)}$ time.*

We have described the base cases, all of which are non-recursive algorithms. The remaining case is when $u_G(p) \ge n^{-1-O(1/\log\log n)}$ and $p < \theta$. In this case, we run a step of recursive contraction (Section 4). As in [2], we interleave recursive contraction calls to a standard sparsification algorithm. However, unlike prior work, we obtain a sharp tradeoff between the rate of progress and the amount of variance in this recursion, by leveraging the parameters given by the ideal tree packing. 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 1.6. *Suppose $u_G(p) \ge n^{-1-O(1/\log\log n)}$ and $p < \theta$. An estimator X for $u_G(p)$ with relative bias $n^{-\Omega(1)}$ and relative second moment $(\log n)^{O(\log\log n)}$ can be computed in $m^{1+o(1)}$ time.*

We now show that Theorem 1.1 follows from these lemmas:

Proof of Theorem 1.1. The base cases are immediate from above. So, we focus on the recursive contraction case. By setting a sufficiently large constant in the exponent of the first base case, we can ensure in the recursive case that the $n^{-\Omega(1)}$ upper bound of relative bias is less than $\frac{\varepsilon}{2}$. Let X be the estimator output by the recursive contraction algorithm. From Lemma 1.6, we have that the relative variance $\eta[X] \le n^{o(1)}$. By standard techniques (see Lemma 2.3), we can run the algorithm $n^{o(1)}\varepsilon^{-2}$ times to get a $(1 \pm \frac{\varepsilon}{2})$-approximation of $\mathbb{E}[X]$ whp. Because $\mathbb{E}[X]$ is a $(1 \pm n^{-\Omega(1)})$-approximation of $u_G(p)$, the aggregated estimator is a $(1 \pm \varepsilon)$-approximation of $u_G(p)$. Each run takes $m^{1+o(1)}\varepsilon^{-O(1)}$ time, so the overall running time is also $m^{1+o(1)}\varepsilon^{-O(1)}$. □

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. To identify the second base case, we use an indirect method as in [2] since we do not know the value of θ. We first run the 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 $p < \theta$.

Next, we need to determine if we are in the third base case. This is more complicated since we do not know the value of $u_G(p)$, and cannot estimate it by Monte Carlo simulation since we are in a reliable case. Instead of detecting the condition on $u_G(p)$ directly, we (approximately) detect the consequence of this condition using two surrogate conditions. The first surrogate condition given in Definition 3.11 is based on a lower bound of $z_G(p)$ (Lemma 3.6). It either decides we are not in the third case, or it allows us to obtain a tight estimator $\tilde{z}$ of $z_G(p)$ with relative bias $O(1/n)$. Since we have $p < \theta$, $\tilde{z}$ is already an estimator of $u_G(p)$ with relative bias $O(1/\log n)$, by Lemma 1.4. Still, we need a second condition to achieve a better relative bias of $n^{-\Omega(1)}$. The second surrogate condition we use is $\tilde{z} < \frac{1}{2n}$, which is a close approximation of the original condition $u_G(p) < n^{-1-\Omega(1/\log\log n)}$. Combined together, the two conditions either decide that we are not in the third base case, or imply that the conclusions of Lemma 1.6 hold.

Roadmap. We set up basic notation and discuss some preliminary facts in Section 2. In Section 3, we give the importance sampling algorithm using an ideal tree packing to handle the reliable case (the third base case) – this is the main contribution of this paper. In Section 4, we give the recursive contraction algorithm for the unreliable case, show the sharp tradeoff between progress and variance in the recursion, and combine it with all the base cases. We note that all missing proofs are deferred to the full version of the paper.

2 Preliminaries

In the network unreliability problem, we are given an undirected multigraph G, and an edge failure probability $p \in (0, 1)$. The goal is to $(1 \pm \varepsilon)$-approximate the unreliability $u_G(p)$, which is the probability that G disconnects when each edge is independently deleted with probability p.

Throughout the paper, we use λ to denote the value of a minimum cut of the graph. Note that $u_G(p) = \Pr[\cup_{\text{cuts } C} C \text{ disconnects}]$, and the probability that a cut C disconnects is given by $p^{|C|}$.

Relative Variance and Bias. We give estimators for $u_G(p)$ (and $z_G(p)$) in this paper. Here, we give (standard) definitions of relative variance and bias of an estimator:

Definition 2.1. 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 2.2 (Lemma I.4 of [13]). *The relative variance of the average of N independent samples of X is $\frac{\eta[X]}{N}$.*

[2] All logarithms are base e unless otherwise stated.

This leads to the following property:

Lemma 2.3 (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 estimate of $\mathbb{E}[X]$ with probability $1-\delta$.*

Next, we define bias and relative bias of an estimator:

Definition 2.4. Let X be an estimator of u. Then, the bias of X is $|\mathbb{E}[X]-u|$, and assuming $u > 0$, its relative bias is $\frac{|\mathbb{E}[X]-u|}{u}$.

k-Strong Components. In the reliable case, it is unlikely that a well-connected part of the graph disconnects; therefore, these parts can be contracted before estimating $u_G(p)$. To formalize this, recall that a k-strong component is a maximal k-edge-connected induced subgraph. The strength k_e of an edge e is the maximum value k such that a k-strong component contains e.

The next lemma shows that 3λ-strong components can be contracted before estimating $u_G(p)$:

Lemma 2.5. *Suppose $p^\lambda \le \varepsilon n^{-1}$. Then, we can contract all 3λ-strong components to form H, such that $u_H(p)$ $(1\pm\varepsilon)$-approximates $u_G(p)$.*

3 The Reliable Case: Importance Sampling on Ideal Tree Packing

This section solves the reliable case (the third base case from Section 1.3), whose main lemma is restated below.

Lemma 1.5. *If $u_G(p) < n^{-1-\Omega(1/\log\log n)}$, an estimator for $u_G(p)$ with relative bias $n^{-\Omega(1)}$ and relative variance ≤ 1 can be computed in $m^{1+o(1)}$ time. As a consequence, under the condition that $u_G(p) < n^{-1-\Omega(1/\log\log n)}$, a $(1\pm\varepsilon)$-approximation to $u_G(p)$ can be computed in $m^{1+o(1)}\varepsilon^{-O(1)}$ time.*

We will first design an importance sampling algorithm for a slightly different goal, which is an estimator for $z_G(p)$ instead of $u_G(p)$.

Lemma 3.1. *If $u_G(p) < n^{-1-\Omega(1/\log\log n)}$, an estimator for $z_G(p)$ with relative bias $O(1/n)$ and relative variance ≤ 1 can be computed in $m^{1+o(1)}$ time.*

We will use Lemma 3.1 to prove Lemma 1.5 in Section 3.5. The rest of this section is devoted to Lemma 3.1.

One complexity here is that the algorithm does not know the value of $u_G(p)$, so it cannot determine if the assumption is true. Instead, we will use an alternative assumption based on tree packings that implies the original assumption $u_G(p) \le n^{-1-\Omega(1/\log\log n)}$.

At a high level, our algorithm uses importance sampling. Note that we need to preferentially sample the near-minimum cuts since they are more likely to fail. Moreover, we need to sample a cut efficiently, e.g., by only picking $O(1)$ edges from the cut. To attain these two properties, we use a fractional spanning tree packing to guide our sampling process. The advantage is that it distributes a near-minimum cut among $\Omega(\lambda)$ spanning trees, creating an $O(1)$-size projection on average in each of them. If we can sample the edges in the projection from a spanning tree, we can recover the near-minimum cut without having to explicitly sample all edges of the cut.

One limitation of the above approach is there is a duality gap of 2 between the tree packing value and min-cut value λ. As a consequence, to sample a near min-cut, we need to pick ≥ 2 tree edges on average, which results in a sampling space of $\Omega(n^2)$. Since we can only afford $o(n^2)$ samples, a naive approach yields only $o(1)$-fraction of these near min-cuts; it is not hard to create examples where the resulting estimator has large error because it misses the critical near min-cuts that determine the value of $u_G(p)$.

To alleviate this concern, consider contracting well-connected vertex sets in the graph. This preserves near min-cuts; so sampling is valid on the contracted graph. Importantly, every vertex pair that remains has a near min-cut separating it; this gives us a different structural tool for evaluating how the sample hits the near min-cuts. More generally, if we do these contractions at multiple recursive levels that preserve near min-cuts of different values, than the resulting graphs allow us a tighter control over the set of cuts that we hit in our sampling process. Since we need the tree packing to guide cut sampling across multiple recursive levels, we need it to evenly partition the cuts for different levels with their corresponding ranges of connectivity. This is achieved by an *ideal tree packing* that we describe next.

3.1 Ideal Tree Packing

Let a *fractional tree packing* be a distribution over spanning trees of a graph. The *load* of each edge is the probability that a tree sampled from this distribution contains that edge. A maximum tree packing is a distribution that minimizes the maximum load of any edge. The *min-ratio cut* of a graph is the partition P of the vertices minimizing the ratio $\frac{d(P)}{|P|-1}$, where $d(P)$ is the cut value P. In other words, $d(P) = |\partial P|$, where ∂P is the set of edges whose endpoints are separated by P. A well-known theorem of Nash-Williams [16] and Tutte [19] states that the maximum load of any edge in a maximum tree packing equals the inverse of the minimum value of the cut ratio defined above.

Thorup's ideal tree packing [17, 18] is a fractional tree packing that is not only (globally) a maximum tree packing, but also maximum in a local sense. The loads $\ell^*(e)$ of edges e in an ideal tree packing are defined by the following *recursive min-ratio cut process.*

Definition 3.2 (Recursive min-ratio cut process). Let P be the min-ratio cut of the input graph, i.e., the vertex partition minimizing $\frac{d(P)}{|P|-1}$. Define $\ell^*(e) := \frac{|P|-1}{d(P)}$ for every edge $e \in \partial P$. Recursively apply this process on the induced subgraph of each non-singleton component of P to set the loads on the other edges.

We call $\ell^*(e)$ the ideal load of edge e. The following lemma ensures that this process uniquely produces the loads of a fractional tree packing.

Lemma 3.3 (Lemma 3 of [18]). *The following hold:*

1. *The ideal loads are unique (although the min-ratio cuts chosen by the recursive min-ratio cut process may not be unique).*
2. *ℓ^* is in the spanning tree polytope, i.e., it is the load of a fractional tree packing.*

Let π^* be the minimum ratio $\frac{d(P)}{|P|-1}$ among all vertex partitions P. The first iteration of the recursive min-ratio cut process described above defines the ideal loads of the edges in the min-ratio cut to be $\frac{1}{\pi^*}$. We have the following relation between π^* and λ.

Fact 3.4 (Lemma 3 of [17]). $\frac{\lambda}{2} < \pi^* \le \lambda$.

A key property of the recursive min-ratio cut process is that the ideal loads are monotone non-increasing in the recursion.

Lemma 3.5 (Lemma 14 of [17]). *Suppose P is a min-ratio cut during the recursive min ratio cut process, $e_1 \in \partial P$, and e_2 is in an induced subgraph of a component of P. Then, $\ell^*(e_1) \ge \ell^*(e_2)$.*

It follows that the ideal loads defined in the first iteration is the maximum throughout the recursion, i.e., $\frac{1}{\pi^*} = \max_e \ell^*(e)$.

Let k_τ be the number of vertices if we contract all edges with $\ell^*(e) < \frac{1}{\tau}$. Clearly, k_τ is monotonically increasing in τ. We give a lower bound on $z_G(p)$ in terms of k_τ:

Lemma 3.6. *$z_G(p) \ge k_\tau p^{2\tau}$ for any $\tau \ge \pi^*$.*

Proof. Form a graph H by contracting all edges e with $\ell^*(e) < \frac{1}{\tau}$. Since $\ell^*(e) = \frac{1}{\pi^*} \ge \frac{1}{\tau}$ for edges e in the global min-ratio cut, H is not a singleton. By definition of k_τ, we have $|V(H)| = k_\tau$.

Next, we claim that if we run the recursive min-ratio cut process in H, the ideal loads of edges in H are the same as their original ideal loads in G. Denote the ideal loads on H by ℓ_H. Whenever the recursive min-ratio cut process on G finds a partition P with cut ratio $\frac{d(P)}{|P|-1} \le \tau$, all edges in ∂P have load $\ell^*(e) \ge \frac{1}{\tau}$ and will not be contracted. So, the cut P is preserved in H with the same cut ratio, and it must be a min-ratio cut in the current induced subgraph. By Lemma 3.3 (1), we can choose the same cut in the process to define ℓ_H. Whenever the recursive min-ratio cut process on G finds a partition P with cut ratio $\frac{d(P)}{|P|-1} > \tau$, the cut edges $e \in \partial P$ have $\ell^*(e) < \frac{1}{\tau}$. Moreover, by Lemma 3.5, all edges in the current induced subgraph have $\ell^*(e) < \frac{1}{\tau}$. So, the whole induced subgraph will be contracted in H. By inductively applying the above two arguments, we can define a recursive min-ratio cut process on H that produces $\ell_H(e) = \ell^*(e)$ for any $e \in E(H)$. By Lemma 3.3 (2), ℓ_H is in the spanning tree polytope. So, $\sum_{e\in E(H)} \ell^*(e) = |V(H)| - 1 = k_\tau - 1$.

Let $d(v)$ be the degree of v, and $\bar{d}$ be the average degree in H. We bound

$$\bar{d} = \frac{1}{k_\tau}\sum_{v\in V(H)} d(v) = \frac{2|E(H)|}{k_\tau} \le \frac{2}{k_\tau}\sum_{e\in E(H)} \ell^*(e)\cdot\tau < 2\tau.$$

Since the degree cut of each vertex $v \in V(H)$ corresponds to a separate cut in G, we have

$$z_G(p) \ge \sum_{v\in V(H)} p^{d(v)} \ge k_\tau p^{\bar{d}} \ge k_\tau p^{2\tau},$$

where the middle inequality follows by convexity of the function $f(x) = p^x$. ◻

Approximate Tree Packing. The recursive algorithm used to define ideal loads above requires many min-ratio cut computations; we don't know how to implement them all in almost-linear time. Instead, we approximate the ideal loads using a *greedy tree packing* algorithm described below. The main purpose is to eventually define surrogate parameters $\tilde{k}_\tau$ that can substitute for the parameters k_τ in Lemma 3.6.

Definition 3.7 (Greedy tree packing). Given a input graph G and a number of rounds k, the greedy algorithm iteratively finds a collection of spanning trees $\{T_1, T_2, \dots, T_k\}$, such that each T_i is a minimum spanning tree on G with respect to edge loads induced by $\{T_1, \dots, T_{i-1}\}$, which we define below. For a collection of spanning trees $\mathcal{T}$, define its induced loads $\ell^{\mathcal{T}}$ by the edge loads of the uniform distribution over trees in $\mathcal{T}$, i.e., $\ell^{\mathcal{T}}(e) = \frac{|\{T\in\mathcal{T}:T\ni e\}|}{|\mathcal{T}|}$. (Specially, for an empty $\mathcal{T}$, $\ell^T(e) = 0$.)

The next lemma asserts that this greedy process approximates an ideal tree packing:

Lemma 3.8. *In an unweighted graph with min-cut value λ, given a parameter $\delta \in (0,1)$, we can find a (multi-)set $\mathcal{T}$ of $6\lambda \ln m/\delta^2$ spanning trees such that*

(1) $|\ell^{\mathcal{T}}(e) - \ell^*(e)| \le 3\delta/\lambda$, $\forall e \in E$, where $\ell^{\mathcal{T}}$ are the edge loads for the uniform distribution over $\mathcal{T}$.
(2) Each tree T is the MST of some weight function ℓ_T satisfying $|\ell_T(e) - \ell^*(e)| \le \delta/\lambda$ for all $e \in E$.

The algorithm runs in time $\tilde{O}(m/\delta^2)$. The trees are implicit, but the loads $\ell^{\mathcal{T}}$ are output explicitly.

We wish to obtain a multiplicative approximation of $\ell^*(e)$ instead of the additive approximation in Lemma 3.8. Moreover, we will later group edges multiplicatively by loads and need to ensure that there are few groups. For both these purposes, we contract all 3λ-strong components in the input graph. The next lemma says that this restricts $\ell^*(e)$ to $(\frac{1}{3\lambda}, \frac{2}{\lambda})$ for all the uncontracted edges:

Lemma 3.9. *After contracting all 3λ-strong components, each edge e satisfies $\ell^*(e) \in (\frac{1}{3\lambda}, \frac{2}{\lambda})$.*

Contracting all 3λ-strong components is without loss of generality due to Lemma 2.5. (If the assumption of Lemma 2.5 does not hold, we can directly switch to the unreliable case since that assumption is implied by the assumption of reliable case.) By applying Lemma 3.8 with parameter $\delta/3$ on the contracted graph given by Lemma 3.9, we get $\ell^{\mathcal{T}}(e) \approx_\delta \ell^*(e)$ for all $e \in E$, where $x \approx_\delta y$ denotes $(1+\delta)^{-1}y \le x \le (1+\delta)y$.

Edge Layering. We set the parameter $\delta := \Theta(1/\log\log n)$ and partition the edges into $O(\frac{1}{\delta}) = O(\log\log n)$ levels based on their ideal load as follows: Let $\pi_i = (1+\delta)^i\lambda/2$; we define $E_i \subseteq E$ to be the edges $e \in E$ with $\ell^*(e) \in [\frac{1}{\pi_i}, \frac{1}{\pi_{i-1}})$. Since $\ell^*(e) \in (\frac{1}{3\lambda}, \frac{2}{\lambda})$ for all edges e, the sets $E_1, E_2, \dots, E_L$ partition E for $L = O(\frac{1}{\delta})$. To avoid clutter, define $E_{>i} = E_{i+1} \cup \dots \cup E_L$. Recall that k_{π_i} is the number of nodes after contracting all edges in $E_{>i}$ (i.e., edges with $\ell^*(e) < \frac{1}{\pi_i}$).

One complication is that the values k_τ are defined on the ideal tree packing, which means that we do not know them exactly. So, we next define approximations $\widetilde{E}_i$ and $\tilde{k}_\tau$ based on our approximate tree packing $\mathcal{T}$. Define $\widetilde{E}_i \subseteq E$ to be the edges $e \in E$ with $\ell^{\mathcal{T}}(e) \in [\frac{1}{\pi_i}, \frac{1}{\pi_{i-1}})$, and let $\tilde{k}_\tau$ be the number of nodes after contracting all edges with $\ell^{\mathcal{T}}(e) < \frac{1}{\tau}$. As earlier, define $\widetilde{E}_{>i} = \widetilde{E}_{i+1} \cup \dots \cup \widetilde{E}_L$, so $\tilde{k}_{\pi_i}$ is the number of nodes after contracting all edges in $\widetilde{E}_{>i}$ (i.e., edges with $\ell^{\mathcal{T}}(e) < \frac{1}{\pi_i}$).

For a spanning tree $T \in \mathcal{T}$, define $\tilde{k}_{\pi_i}(T) = |E(T) \cap \widetilde{E}_i|$; note that since T is a tree, $\tilde{k}_{\pi_i}(T)$ is also the number of nodes in T after contracting the set of edges $E(T) \setminus \widetilde{E}_i$, minus 1. We now show how these parameters k_{π_i}, $\tilde{k}_{\pi_i}$, and $\tilde{k}_{\pi_i}(T)$ are related.

Lemma 3.10. *For every i, we have $k_{\pi_i} \le \tilde{k}_{\pi_{i+1}} \le k_{\pi_{i+2}}$ and $\tilde{k}_{\pi_i}(T) \le \tilde{k}_{\pi_{i+2}} - 1$ for all $T \in \mathcal{T}$.*

3.2 Surrogate Criterion for Deciding Reliable vs Unreliable Cases

Based on the edge layering above, we define the criterion that is used to distinguish between the reliable and unreliable cases. This replaces the condition $u_G(p) < n^{-1-\Omega(1/\log\log n)}$ in Lemma 3.1. The latter condition, in conjunction with Lemmas 1.4 and 3.6, yield a condition on k_τ that, in principle, can be used directly to separate the reliable and unreliable cases. However, since we do not have a sufficiently fast algorithm to compute the parameters k_τ, we will instead use the approximate parameters $\tilde{k}_\tau$ and rely on Lemma 3.10 to relate the $\tilde{k}_\tau$ and k_τ parameters.

Definition 3.11. The algorithm is in the reliable case if for all levels i, $n^{1+30\delta}\tilde{k}_{\pi_i}p^{2\pi_i} \le 1$ holds. Otherwise, it is in the unreliable case. (Note that since we are not in the second base case, we assume $p < \theta$ for both these cases.)

The next two lemmas summarize the relevant implications of this condition being true or false, i.e., in the unreliable and reliable cases respectively.

Recall that $\ell^*(\cdot)$ and $\ell^{\mathcal{T}}(\cdot)$ respectively denote the ideal loads and approximate loads computed by greedy tree packing (as in Lemma 3.8) for all edges. We first extend this notation to set functions on cuts as follows: For any function $\ell(\cdot)$ (such as $\ell^*(\cdot)$, $\ell^{\mathcal{T}}(\cdot)$, etc.) and a cut C, we use the shorthand $\ell(C)$ to denote $\sum_{e\in C}\ell(e)$.

Let us first consider the case when Definition 3.11 is false, i.e., the unreliable case. In this case, we show that $u_G(p) > n^{-1-O(1/\log\log n)}$ holds. This implies that in order to prove Lemma 3.1, it suffices to show the conclusion of the lemma under the surrogate condition Definition 3.11.

Lemma 3.12 (Unreliable Case). *If $n^{1+30\delta}\tilde{k}_{\pi_i}p^{2\pi_i} > 1$ for some $i \ge 0$ and $p < \theta$, then $u_G(p) > n^{-1-O(1/\log\log n)}$.*

We are now left to show that under Definition 3.11, we can prove the conclusion of Lemma 3.1; this will establish Lemma 3.1.

Define $\mathcal{C}$ as the collection of cuts C with $\ell^{\mathcal{T}}(C) \le 2\log n$, and define $z' = z'_G(p) = \sum_{C\in\mathcal{C}} p^{|C|}$. Clearly, $z'_G(p) \le z_G(p)$. The next lemma shows that $z'_G(p)$ is a tight upper bound for $z_G(p)$ when the condition in Definition 3.11 is true.

Lemma 3.13 (Reliable Case). *If $n^{1+30\delta}\tilde{k}_{\pi_i}p^{2\pi_i} \le 1$ for all $i \ge 0$, then $z'_G(p) \ge (1 - O(1/n))z_G(p)$.*

To prove Lemma 3.1, we are left to give an algorithm for an estimator of $z'_G(p)$ and bound its variance. We do this over the next two subsections.

3.3 Cut Sampling Algorithm on Tree Packing

We now give a cut sampling algorithm on the approximate tree packing that we described above. The goal of this algorithm is to design an estimator for $z'_G(p)$.

First, we define a distribution q on the cuts of the graph. The algorithm will sample a cut C according to this distribution q, i.e. each cut C is sampled with probability $q(C)$, and then compute the estimator

$$X = \begin{cases} p^{|C|}/q(C) & \text{if } \ell^{\mathcal{T}}(C) \le 2\log n, \\ 0 & \text{otherwise.} \end{cases}$$

Our high-level plan is to establish the following properties:

(1) After $\tilde{O}(m)$ preprocessing time, we can sample a cut C according to q and compute X in $\tilde{O}(1)$ time.
(2) The relative variance of X is $n^{1+o(1)}$.

If these hold, then by taking average of $n^{1+o(1)}$ samples, we obtain an estimator for $z'_G(p)$ with relative variance $O(1)$ in $\tilde{O}(m)+n^{1+o(1)}$ time by Fact 2.2. (Recall that by Lemma 3.13, this gives an estimator for $z_G(p)$ with relative bias $O(1/n)$, which establishes Lemma 3.1.)

Property 1 essentially follows from prior work [2] (given later in Lemma 3.16). We focus on satisfying Property 2 first. In fact, we show a stronger property. Define the *likelihood ratio* of a cut C as $\rho(C) = \frac{p^{|C|}}{q(C)\cdot z'_G(p)}$, i.e., the ratio of the fractional contribution $p^{|C|}/z'_G(p)$ of cut C to $z'_G(p)$ and the probability $q(C)$ of being sampled. Then,

2'. For each cut $C \in \mathcal{C}$, we have $\rho(C) \le n^{1+o(1)}$.

Property 2' implies property 2 because of the following lemma.

Lemma 3.14. *Suppose $z' = \sum_{C\in\mathcal{C}} p^{|C|}$ for some collection of cuts $\mathcal{C}$, and X is an importance sampling estimator such that $X = p^{|C|}/q(C)$ when $C \in \mathcal{C}$ and $X = 0$ otherwise. Here, $q(C)$ is the probability of sampling C. Then, the relative variance of X is upper bounded by the maximum likelihood ratio $\max_{C\in\mathcal{C}} \dfrac{p^{|C|}}{q(C)\cdot z'}$.*

Sampling Algorithm on Approximate Tree Packing. Our goal in this section is to describe a distribution q on cuts such that the likelihood ratio $\rho(C) = \frac{p^{|C|}}{q(C)\cdot z'_G(p)}$ is $n^{1+o(1)}$ for all cuts C with $\ell^{\mathcal{T}}(C) \le 2\log n$. Note that the cuts C with $\ell^{\mathcal{T}}(C) > 2\log n$ contribute only an $O(1/n)$ fraction of $z_G(p)$ by Lemma 3.13.

The distribution q is defined by a process that samples a random cut. For computational reasons, we first sparsify the tree packing $\mathcal{T}$ into another tree packing $\widetilde{\mathcal{T}}$ by uniformly sampling $\tilde{O}(1)$ trees in $\mathcal{T}$ with replacement. Using the implicit representation of $\mathcal{T}$, we can explicitly recover all trees in $\widetilde{\mathcal{T}}$ in $\tilde{O}(n)$ time.

Now, we uniformly sample a random tree T from the sparsified packing $\widetilde{\mathcal{T}}$, and also uniformly sample random integers $j_1, j_2, \ldots, j_L$ conditioned on $\sum_i j_i \le 8\log n$ and $j_i \le |E(T) \cap \widetilde{E}_i|$ for all $i \in [L]$. Then, we uniformly sample j_i edges from $E(T) \cap \widetilde{E}_i$ (without replacement). The sampled edges uniquely define a vertex bi-partition such that the set of edges in T crossing the bi-partition is exactly the set of sampled edges. We output the cut corresponding to this bi-partition.

The next lemma bounds the likelihood ratio $\frac{p^{|C|}}{q(C)\cdot z'_G(p)}$ of this sampling procedure for the cuts involved in $z'_G(p)$, under the additional assumption that $p^\lambda \ge n^{-1000}$.[3] In Section 3.4, we give a separate algorithm that handles the $p^\lambda < n^{-1000}$ case.

[3]We can use any sufficiently large constant instead of 1000 here.

Lemma 3.15. *Assume that $n^{1+30\delta}\tilde{k}_{\pi_i}p^{2\pi_i} \le 1$ for all i, and that $p^\lambda \ge n^{-1000}$. With high probability, for each cut C with $\ell^{\mathcal{T}}(C) \le 2\log n$, the likelihood ratio of C is $(\log n)^{O(1/\delta)}n^{1+O(\delta)}$.*

Proof Sketch. Consider a fixed cut $C \in \mathcal{C}$, and define $C_i = C \cap \widetilde{E}_i$. Sample a tree $T \in \widetilde{\mathcal{T}}$, and define $j_i^* = |C_i \cap E(T)|$ for all $i \in [L]$. Define $\tilde{k}'_{\pi_i} = \max\{\tilde{k}_{\pi_i}, \sqrt{n}\}$. Using Lemma 3.10, we can show that at least a $\Omega(\delta)$ fraction of trees in $\mathcal{T}$ satisfy the following property:

$$\sum_i j_i^* \log \tilde{k}'_{\pi_{i+2}} \le (1+\delta)\sum_i \ell^{\mathcal{T}}(C_i)\log \tilde{k}'_{\pi_{i+2}}. \tag{3.1}$$

Since the sparsified $\widetilde{\mathcal{T}}$ consists of $\tilde{O}(1)$ many random trees in $\mathcal{T}$, the probability that at least one $T \in \widetilde{\mathcal{T}}$ satisfies (3.1) is at least $1 - n^{-10\log n}$. Since there are only $|\mathcal{T}| \cdot O(n^{8\log n})$ many possible cuts C that can be sampled, by a union bound we conclude that at least one tree $T \in \mathcal{T}$ satisfies (3.1) for all cuts C we need to consider. In this case, we obtain

$$\begin{aligned}\frac{1}{2}\log n \sum_i j_i^* &\le \sum_i j_i^* \log\sqrt{n} \le \sum_i j_i^* \log \tilde{k}'_{\pi_{i+2}} \\ &\le (1+\delta)\sum_i \ell^{\mathcal{T}}(C_i)\log \tilde{k}'_{\pi_{i+2}} \le (1+\delta)\ell^{\mathcal{T}}(C)\log n,\end{aligned}$$

which implies that $\sum_i j_i^* \le 2(1+\delta)\ell^{\mathcal{T}}(C)$. Since $\ell^{\mathcal{T}}(C) \le 2\log n$ and $\delta \le 1$ by assumption, we have $\sum_i j_i^* \le 8\log n$. In particular, it is possible for the algorithm to sample $j_i = j_i^*$ for all i. The probability that this occurs is the inverse of the number of combinations $j_1, j_2, \ldots, j_L$ satisfying $\sum_i j_i \le 8\log n$ and $j_i \le |E(T) \cap \widetilde{E}_i|$ for all $i \in [L]$. The first inequality alone limits the number of combinations to $(8\log n)^{O(L)} = (\log n)^{O(1/\delta)}$. Then, the probability that $j_i = j_i^*$ for all i is $(\log n)^{-O(1/\delta)}$.

Conditioned on $j_i = j_i^*$ for all i, the cut C is sampled precisely when we sample exactly the edges $C_i \cap E(T)$ from $\widetilde{E}_i \cap E(T)$ for each i. The total number of possible samples (over all i) is

$$\prod_i (k_{\pi_i})^{j_i} \le \prod_i (\tilde{k}'_{\pi_{i+2}})^{j_i^*} \overset{(3.1)}{\le} \prod_i (\tilde{k}'_{\pi_{i+2}})^{(1+\delta)\ell^{\mathcal{T}}(C_i)}.$$

Multiplying the probabilities together, the probability of sampling a tree $T \in \widetilde{\mathcal{T}}$ satisfying (3.1), and then sampling $j_i = j_i^*$ for all i, and then sampling exactly the edges $C_i \cap E(T)$ from $\widetilde{E}_i \cap E(T)$ for each i is at least

$$\frac{1}{|\widetilde{\mathcal{T}}|} \cdot (\log n)^{-O(1/\delta)} \cdot \prod_i (\tilde{k}'_{\pi_{i+2}})^{-(1+\delta)\ell^{\mathcal{T}}(C_i)}.$$

This is a lower bound for the probability $q(C)$ of sampling cut C. Note that since $|\widetilde{\mathcal{T}}| = \tilde{O}(1)$, $|\widetilde{\mathcal{T}}|$ can be absorbed into $(\log n)^{O(1/\delta)}$. We abbreviate the term $1/|\widetilde{\mathcal{T}}| \cdot (\log n)^{-O(1/\delta)}$ by $1/\sigma$, and $\sigma = (\log n)^{O(1/\delta)}$.

Finally, we bound the likelihood ratio

$$\rho(C) = \frac{p^{|C|}}{q(C)\cdot z'} \le \frac{1}{z'} \cdot p^{|C|} \cdot \sigma \cdot \prod_i (\tilde{k}'_{\pi_{i+2}})^{(1+\delta)\ell^{\mathcal{T}}(C_i)}.$$

Notice that

$$|C| = \sum_i |C_i| = \sum_i \sum_{e\in C_i} 1 \ge \sum_i \sum_{e \in C_i} \ell^{\mathcal{T}}(e)\pi_{i-1} = \sum_i \ell^{\mathcal{T}}(C_i)\pi_{i-1},$$

so

$$\rho(C) \le \frac{1}{z'} \cdot \prod_i p^{\ell^{\mathcal{T}}(C_i)\pi_{i-1}} \cdot \sigma \cdot \prod_i (\tilde{k}'_{\pi_{i+2}})^{(1+\delta)\ell^{\mathcal{T}}(C_i)}.$$

Recall from Lemma 3.6 that $z \ge k_{\pi_{i+3}}p^{2\pi_{i+3}}$ for all i. We have $k_{\pi_{i+3}} \ge \tilde{k}_{\pi_{i+2}}$ by Lemma 3.10, so we can write

$$\begin{aligned}\frac{1}{z} &= \prod_i z^{-\ell^{\mathcal{T}}(C_i)/\ell^{\mathcal{T}}(C)} \le \prod_i \left(\tilde{k}_{\pi_{i+2}}p^{2\pi_{i+3}}\right)^{-\ell^{\mathcal{T}}(C_i)/\ell^{\mathcal{T}}(C)} \\ &= \prod_i \left(\tilde{k}_{\pi_{i+2}}p^{2(1+\delta)\pi_{i+2}}\right)^{-\ell^{\mathcal{T}}(C_i)/\ell^{\mathcal{T}}(C)},\end{aligned}$$

and

$$\begin{aligned}\rho(C) &\le \sigma \cdot \prod_i \left(\left(\tilde{k}_{\pi_{i+2}}p^{2(1+\delta)\pi_{i+2}}\right)^{-1/\ell^{\mathcal{T}}(C)} p^{\pi_{i-1}}(\tilde{k}'_{\pi_{i+2}})^{1+\delta}\right)^{\ell^{\mathcal{T}}(C_i)} \\ &= \sigma \cdot \prod_i \left(\left(\tilde{k}_{\pi_{i+2}}\right)^{-1/\ell^{\mathcal{T}}(C)} p^{\pi_{i+2}\left(\frac{1}{(1+\delta)^3} - \frac{2(1+\delta)}{\ell^{\mathcal{T}}(C)}\right)}(\tilde{k}'_{\pi_{i+2}})^{1+\delta}\right)^{\ell^{\mathcal{T}}(C_i)}.\end{aligned} \tag{3.2}$$

Recall our assumption $n^{1+30\delta}\tilde{k}_{\pi_i}p^{2\pi_i} \le 1$ for all i. Notice that

$$\begin{aligned}(\tilde{k}'_{\pi_{i+2}})^{1+15\delta} &= \max\{\tilde{k}_{\pi_{i+2}}, \sqrt{n}\}^{1+15\delta} \le \left(\sqrt{n\tilde{k}_{\pi_{i+2}}}\right)^{1+15\delta} \\ &= \sqrt{n^{1+15\delta}(k_{\pi_{i+2}})^{1+15\delta}} \le \sqrt{n^{1+30\delta}k_{\pi_{i+2}}} \le p^{-\pi_{i+2}}.\end{aligned}$$

So, $p^{\pi_{i+2}} \le (\tilde{k}'_{\pi_{i+2}})^{-(1+15\delta)}$.

Let us first assume that $\ell^{\mathcal{T}}(C) \ge 2-\delta$. Using this assumption, we can show that for a large enough constant $K > 0$,

$$p^{\pi_{i+2}\left(\frac{1}{(1+\delta)^3} - \frac{2(1+\delta)}{\ell^{\mathcal{T}}(C)}\right)} \le (\tilde{k}'_{\pi_{i+2}})^{-(1+\delta) + \frac{2+O(\delta)}{\ell^{\mathcal{T}}(C)} - \frac{K\delta}{\ell^{\mathcal{T}}(C)}} p^{\pi_{i+2}\left(-\frac{K\delta}{\ell^{\mathcal{T}}(C)}\right)}$$

Continuing from (3.2), we are left with

$$\begin{aligned}\rho(C) &\le \sigma \prod_i \left((\tilde{k}_{\pi_{i+2}})^{\frac{-1}{\ell^{\mathcal{T}}(C)}} (\tilde{k}'_{\pi_{i+2}})^{\frac{2+O(\delta)}{\ell^{\mathcal{T}}(C)}} (\tilde{k}'_{\pi_{i+2}}p^{\pi_{i+2}})^{\frac{-K\delta}{\ell^{\mathcal{T}}(C)}}\right)^{\ell^{\mathcal{T}}(C_i)} \\ &= \sigma \prod_i \left(\tilde{k}^{-1}_{\pi_{i+2}}(\tilde{k}'_{\pi_{i+2}})^{2+O(\delta)}\right)^{\frac{\ell^{\mathcal{T}}(C_i)}{\ell^{\mathcal{T}}(C)}} \prod_i \left(\tilde{k}'_{\pi_{i+2}}p^{\pi_{i+2}}\right)^{-K\delta\cdot\frac{\ell^{\mathcal{T}}(C_i)}{\ell^{\mathcal{T}}(C)}}.\end{aligned}$$

Next, we use $\tilde{k}'_{\pi_{i+2}} = \max\{\tilde{k}_{\pi_{i+2}}, \sqrt{n}\} \le \sqrt{n\tilde{k}_{\pi_{i+2}}}$ to get an upper bound of $n^{1+O(\delta)}$ for the first product. For the second product, we use $\pi_i \in \Theta(\lambda)$ and $p^\lambda \ge n^{-1000}$ to obtain

$$\prod_i (\tilde{k}'_{\pi_{i+2}}p^{\pi_{i+2}})^{-K\delta\cdot\frac{\ell^{\mathcal{T}}(C_i)}{\ell^{\mathcal{T}}(C)}} \le (n^{-O(1)})^{-K\delta\cdot\sum_i\frac{\ell^{\mathcal{T}}(C_i)}{\ell^{\mathcal{T}}(C)}} = n^{O(\delta)}.$$

We conclude that $\rho(C) \le \sigma n^{1+O(\delta)}$ when $\ell^{\mathcal{T}}(C) \ge 2-\delta$.

To handle the case $\ell^{\mathcal{T}}(C) < 2-\delta$, observe that by Markov's inequality, at least a $\Omega(\delta)$ fraction of trees $T \in \mathcal{T}$ satisfy $|C \cap E(T)| < 2$, which means $|C \cap E(T)| = 1$. Thus, with high probability at least one tree $T \in \widetilde{\mathcal{T}}$ satisfies $|C \cap E(T)| = 1$. Now suppose that the single edge in $C \cap E(T)$ is in $\widetilde{E}_i$. Then, with probability $\frac{1}{\sigma}$ we have $j_i = 1$ and $j_{i'} = 0$ for all $i' \ne i$, and with probability $1/|\widetilde{E}_i \cap E(T)| \ge 1/n$. So overall, the cut C is sampled with probability $1/|\widetilde{\mathcal{T}}| \cdot 1/\sigma \cdot 1/n$. It follows that

$$\rho(C) = \frac{p^{|C|}}{q(C)\cdot z'_G(p)} \le \frac{1}{q(C)} = (\log n)^{O(1/\delta)}\cdot n,$$

concluding the proof. □

Running Time. Recall that we set $\delta = \Theta(1/\log\log n)$, so for any cut C with $2/(1+\delta) < \ell^*(C) \le \log n$, the likelihood ratio $\rho(C)$ is $n^{1+o(1)}$ by Lemma 3.15. By Lemma 3.14, the relative variance of $X = p^{|C|}/q(C)$ is $n^{1+o(1)}$, so the algorithm takes $n^{1+o(1)}$ samples, obtaining an estimator for $z_G(p)$ with relative variance $O(1)$ and relative bias $O(1/n)$.

To obtain the $n^{1+o(1)}$ samples efficiently, the algorithm uses the following lemma from [2]:

Lemma 3.16 (Lemma 3.8 of [2])**.** *Given an unweighted undirected graph $G = (V, E)$ and a tree T spanning V, we can build a data structure $\mathbb{S}_{G,T}$ in $m^{1+o(1)}$ time such that the following query can be answered in $\tilde{O}(1)$ time: Given $\tilde{O}(1)$ edges on T to define a cut C, return its cut value on G.*

In particular, the algorithm initializes data structure $\mathbb{S}_{G,T}$ for each $T \in \widetilde{\mathcal{T}}$ and then, for $n^{1+o(1)}$ iterations, performs the following: sample $T \in \widetilde{\mathcal{T}}$ and $j_1, \ldots, j_L$ and $j_i \le 8\log n$ edges from $E(T) \cap \widetilde{E}_i$ for each i, and then query the cut value using data structure $\mathbb{S}_{G,T}$. To compute $q(C)$, the algorithm needs to figure out the probability that cut C is sampled. This probability can be determined by simple combinatorics after computing $|C \cap \widetilde{E}_i \cap E(T')|$ for all trees $T' \in \widetilde{\mathcal{T}}$. To compute these values in $\tilde{O}(1)$ time per sample, initialize data structure $\mathbb{S}_{\widetilde{E}_i \cap T_1, T_2}$ for each pair of trees $T_1, T_2 \in \widetilde{\mathcal{T}}$ and $i \in [L]$. Then $|C \cap \widetilde{E}_i \cap E(T')|$ equals the output of data structure $\mathbb{S}_{\widetilde{E}_i \cap T', T}$ when queried the sampled edges of T. The overall running time is $m^{1+o(1)}$.

This concludes the analysis of the algorithm. We have established Lemma 3.1 for $p^\lambda \ge n^{-1000}$.

3.4 Very Reliable Case

This section handles the very reliable case $p^\lambda < n^{-1000}$. We start by proving an analog of Lemma 3.13 – note that we are defining $z'_G(p)$ differently from Lemma 3.13 for the very reliable case.

Lemma 3.17. *Assume $p^\lambda < n^{-1000}$. Let $z'_G(p) = \sum_{C:|C|<1.1\lambda} p^{|C|}$. Then $z'_G(p) \ge (1 - O(1/n))\, z_G(p)$.*

As in Section 3.3, the algorithm first samples a cut C according to some distribution q, i.e., each cut C is sampled with probability $q(C)$, and then computes the estimator

$$X = \begin{cases} p^{|C|}/q(C) & \text{if } |C| < 1.1\lambda, \\ 0 & \text{otherwise.} \end{cases}$$

By the same argument as in Section 3.3, Lemma 3.17 implies that the estimator X has relative bias $O(1/n)$ to $z_G(p)$. It suffices to bound the likelihood ratio $\rho(C) = \frac{p^{|C|}}{q(C)\cdot z'_G(p)}$ over all cuts C with $|C| < 1.1\lambda$. These cuts have $\ell^*(C) \le |C|/\pi^* \le 2|C|/\lambda < 2.2$.

Algorithm Description. First, we compute a Gomory-Hu tree [7] of G, which is defined as follows:

Definition 3.18. For an undirected graph $G = (V, E)$, a tree T on the set of vertices V is said to be a Gomory-Hu tree of G if for every pair of vertices $u, v \in V$, there is an (u, v)-mincut in T that has the same bi-partition and the same cut value as an (u, v)-mincut in G.

Being a tree, each edge (u, v) on the Gomory-Hu tree defines a (u, v) min-cut; call these Gomory-Hu tree cuts. There are $n - 1$ distinct Gomory-Hu tree cuts. We order their cut values in increasing order to form a list $L = (c_1, c_2, \ldots, c_{n-1})$. Also define $c_n = \max\{c_{n-1}, 1.1\lambda\}$.

Next, the algorithm partitions the list into $\log\log n$ levels as follows. Let $k_i = \lfloor n^{1-2^{-i}} \rfloor$ for $i = 1, 2, \ldots, \log\log n$. Also define $k_0 = 1$ and $k_{\log\log n+1} = n$. For each level $i = 1, 2, \ldots, \log\log n + 1$, the algorithm contracts all c_{k_i}-connected components in G to form a subgraph G_i. This can be achieved by contracting all edges on the Gomory-Hu tree that correspond to cuts of value $\ge c_{k_i}$. The algorithm computes a greedy tree packing $\mathcal{T}_i$ of $\tilde{O}(\lambda/\varepsilon^2)$ trees on each contracted graph G_i, and then samples $\tilde{O}(1)$ of them to form $\widetilde{\mathcal{T}_i}$. Let ℓ_i be the load of $\mathcal{T}_i$ and let ℓ^*_i be the ideal load of G_i, so that $\ell_i \approx_{0.01} \ell^*_i$.

The distribution q is defined by the following sampling procedure: First, we sample i uniformly from $\{1, 2, \ldots, \log\log n + 1\}$. Then, we uniformly sample a tree $T \in \widetilde{\mathcal{T}_i}$ and two edges in T (with replacement) to define a cut C.

We bound the likelihood ratio in the next lemma:

Lemma 3.19. *Assume $p^\lambda < n^{-1000}$. With high probability, for each cut C with $|C| < 1.1\lambda$, the likelihood ratio of C is $\tilde{O}(n)$.*

We now use the connection between likelihood ratio and relative variance given by Lemma 3.14 to establish the desired bound on the relative variance of the estimator for $z'_G(p)$.

Corollary 3.20. *The relative variance in the very reliable case of the estimator of $z'_G(p)$ is at most $\tilde{O}(n)$.*

Running Time. For the sampled cut C, we can compute its overall sampling probability $q(C)$ using the data structure of Lemma 3.16 similar to the previous section. We need an additional preprocessing step to construct a Gomory-Hu tree, which takes $m^{1+o(1)}$ time [1]. In conclusion, the same running time bound of $m^{1+o(1)}$ continues to hold for the very reliable case.

Now, we have established bounds on the relative variance and running time of estimators of $z'_G(p)$, as defined differently in the two cases – reliable and very reliable. This concludes our proof of Lemma 3.1.

3.5 Using Bootstrapping to Improve Relative Bias

We have established an importance sampling algorithm which outputs an estimator of $z_G(p)$ with relative bias $O(1/n)$ and relative variance ≤ 1. By Lemma 1.4, this is also an estimator of $u_G(p)$ with relative bias $O(1/\log n)$. But, this is not sufficient for establishing Lemma 1.5. It turns out we can improve this bound by viewing the importance sampling algorithm as a bootstrapping step.

Definition 3.21. Assume we are in the reliable case of Definition 3.11. Run the importance sampling algorithm to get an estimator for $z_G(p)$ with relative bias $O(1/n)$ and relative variance ≤ 1 guaranteed by Lemma 3.1. By sampling from this estimator and applying Lemma 2.3, whp we can obtain a 2-approximation of $z_G(p)$, denoted $\tilde{z}$. The algorithm is still in the reliable case if $\tilde{z} < \frac{1}{2n}$. If $\tilde{z} \ge \frac{1}{2n}$, we switch to the unreliable case.

If we switch to the unreliable case according to the above criteria, we merely view the importance sampling algorithm as a preprocessing step to distinguish reliable and unreliable cases. If we are still in the reliable case, we run the importance sampling algorithm again to obtain an estimator, which serves as the output of the algorithm. Since we have a stronger assumption than our surrogate assumption, we can exploit it to guarantee a stronger bound for relative bias.

Lemma 3.22. *If $\tilde{z} \geq \frac{1}{2n}$, $p < \theta$ and $\tilde{z}$ is a 2-approximation of $z_G(p)$, then $u_G(p) \geq n^{-1-O(1/\log\log n)}$.*

Lemma 3.23. *If $\tilde{z} < \frac{1}{2n}$ and z is a 2-approximation of $z_G(p)$, then the estimator output by importance sampling algorithm has $n^{-\Omega(1)}$ relative bias to $u_G(p)$.*

Finally, we combine the results to prove Lemma 1.5. By Lemmas 3.12 and 3.22, we only switch to unreliable case when the assumption of Lemma 1.5 does not hold. The algorithm outputs an importance sampling estimator in the reliable case. Its relative bias is $n^{-\Omega(1)}$ by Lemma 3.23. Its relative variance is ≤ 1 by Lemmas 3.14, 3.15 and 3.19. (The estimator takes average of $n^{1+o(1)}$ samples.) This argument requires the boostrapping step to correctly obtain a 2-approximation, which happens whp.

4 The Unreliable Case: Recursive Contraction

4.1 Description of the Algorithm

The algorithm is recursive. First, we recall the base cases from Section 1.3.

(1) If $n \leq \varepsilon^{-O(1)}$, then we run Karger's algorithm [14] that gives an unbiased estimator for $u_G(p)$.
(2) The second base case is to run Monte Carlo sampling when $p \geq \theta$ for some threshold θ whose value is given by Lemma 1.4.
(3) The final base case is invoked when Definition 3.11 and Definition 3.21 both hold. In this case, we run the importance sampling algorithm in Section 3.

We say the algorithm in the recursive case if we are not in any base case. By Lemmas 1.4, 3.12 and 3.22, we have the following fact, which shows that the recursive case is a moderately unreliable case.

Fact 4.1. *In the recursive case, we have $n^{-1-O(1/\log\log n)} \leq u_G(p) \leq n^{-\Omega(1/\log\log n)}$.*

The random contraction algorithm is defined as follows: We say $H \sim G(q)$ if H is generated from G by contracting each edge independently with probability $1-q$. (We will define the value of q shortly.) We repeat this process twice independently to generate independent samples $H_1, H_2 \sim G(q)$, and recursively obtain estimators X_1, X_2 for $u_{H_1}(p/q)$ and $u_{H_2}(p/q)$ respectively. We return the average of X_1 and X_2 as the estimator for $u_G(p)$.

In the random contraction algorithm above, we use $q = 2^{-1/\gamma}$, where γ is defined next. This parameter is related to the ideal tree packing from Section 3. Recall that $\ell^*(e)$ is the load on edge e in an ideal tree packing. Let γ^* be the maximum threshold such that contracting edges e with $\ell^*(e) \leq 1/\gamma^*$ will result in at most δn nodes for $\delta = \Theta(1/\log\log n)$. Ideally, we would like to set $\gamma = \gamma^*$, but we don't know the ideal tree packing. Instead, we $(1 \pm \delta)$-approximate the load on every edge by a greedy tree packing as described in Definition 3.7. Call the loads induced by greedy tree packing as greedy loads. Now, we take the maximum threshold γ' such that contracting edges with greedy load $\leq 1/\gamma'$ produces at most δn nodes. We set $\gamma = \gamma'$ if $\gamma' \leq \lambda$, the min-cut value; else, we set $\gamma = \lambda$.

To make random contraction efficient, we want to ensure that $m \leq n^{1+o(1)}$ in every recursive step. We would like to use standard cut sparsification to achieve this bound. But, sparsification introduces additional variance, which we also need to control. It turns out that the right choice is to have $m \leq n^{1+\Theta\left(\frac{1}{\sqrt{\log n}}\right)}$ during the recursion. When this is violated, we perform a sparsification step given by the following lemma (similar to the sparsification step in [2]).

Lemma 4.2. *Given a graph G and failure probability p such that the algorithm is in the recursive case, we can generate a sparsifier H with $\lambda_H = \tilde{O}(1)$ and a probability $q < p$, such that $u_H(q)$ is an unbiased estimator of $u_G(p)$ with relative second moment at most $2 + O\left(\frac{1}{\log n}\right)$. The running time is $\tilde{O}(m)$.*

Whenever we perform sparsification, we also reduce the relative variance by branching into two recursive calls H_1, H_2 by making two independent calls to the sparsfication algorithm. We then recursively obtain estimators X_1, X_2 for $u_{H_1}(q')$ and $u_{H_2}(q')$ respectively, and return the average of X_1 and X_2. (The value of q' is given by Lemma 4.2.)

4.2 Bias of the Estimator

Lemma 4.3. *The algorithm outputs an estimator of $u_G(p)$ with relative bias $n^{-\Omega(1)}$.*

Proof. The proof is by induction on vertex size. The first two base cases are unbiased. The last base case of importance sampling has relative bias $\beta = n^{-\Omega(1)}$ by Lemma 1.5.

In both inductive steps of random contraction and sparsification, we output the estimator $X = \frac{X_1+X_2}{2}$, where X_1, X_2 are recursive estimators for u_1, u_2 respectively, and $\mathbb{E}[u_1] = \mathbb{E}[u_2] = u_G(p)$. By the inductive hypothesis, $|\mathbb{E}[X_1] - u_1| \leq \beta u_1$, $|\mathbb{E}[X_2] - u_2| \leq \beta u_2$.

$$\begin{aligned}|\mathbb{E}[X] - u_G(p)| &= 1/2 \cdot |\mathbb{E}[X_1] + \mathbb{E}[X_2] - \mathbb{E}[u_1] - \mathbb{E}[u_2]| \\ &\leq 1/2 \cdot (\mathbb{E}_{H_1}[|\mathbb{E}[X_1] - u_1| \mid H_1] + \mathbb{E}_{H_2}[|\mathbb{E}[X_2] - u_2| \mid H_2]|) \\ &\leq 1/2 \cdot (\mathbb{E}_{H_1}[\beta u_1] + \mathbb{E}_{H_2}[\beta u_2]) \quad = \beta \cdot u_G(p).\end{aligned}$$ □

4.3 Some Important Properties

We are left to bound the relative variance of the estimator and the running time of the algorithm. For this, we first need to establish some properties of the parameter γ that we used in the algorithm. Then, we argue about the rate at which the vertex size decreases when we run random contraction. This is used both in bounding relative variance and the running time of the algorithm.

Properties of γ. First, we give two properties of γ. Recall that $\delta = \Theta(1/\log\log n)$ is the parameter such that greedy tree packing approximates the load on every edge in ideal tree packing up to a factor of $1 \pm \delta$ by Lemma 3.8.

Fact 4.4. $\gamma^* \in (1 \pm \delta)\gamma'$.

Fact 4.5. *When* $u_G(p) \geq n^{-1-O(1/\log\log n)}$, *it holds that* $u_G(p) \geq p^{(1+\delta)\cdot\gamma}$.

Proof. There are two cases depending on whether $\gamma = \lambda$ or $\gamma = \gamma'$. In the first case, we have $u_G(p) \geq p^{\lambda} \geq p^{(1+O(1/\log\log n))\lambda}$.

In the second case, we first note that $u_G(p) \geq n^{-1-O(1/\log\log n)}$ also implies $u_G(p) \geq (\delta n)^{-1-\Theta(1/\log\log n)}$ since

$$\delta^{-1-\Theta(1/\log\log n)} \leq O(\log\log n)^{1+O(1/\log\log n)} \leq n^{O(1/\log\log n)}.$$

Next, recall that k_τ is the number of nodes after contracting all edges e with $\ell^*(e) < 1/\tau$. By Lemma 3.6 and Fact 4.4,

$$u_G(p) \geq k_{\gamma^*} p^{2\gamma^*} \geq \delta n \cdot p^{2(1+\delta)\gamma}.$$

We combine this with $u_G(p) \geq (\delta n)^{-1-\Theta(1/\log\log n)}$, which is equivalent to

$$(u_G(p))^{1+\Theta(1/\log\log n)} \geq (\delta n)^{-1}.$$

Multiplying the two inequalities above, we get

$$(u_G(p))^{2+\Theta(1/\log\log n)} \geq p^{2(1+\delta)\gamma}.$$

Therefore, $(u_G(p))^2 \geq p^{2(1+\delta)\gamma}$, or equivalently, $u_G(p) \geq p^{(1+\delta)\gamma}$. □

Size Decrease. We show that the vertex size decreases sufficiently after a random contraction step.

Lemma 4.6. *Whp, it holds that for every random contraction step on an n-vertex graph, the vertex size after contraction is at most* $\left(\frac{1}{2} + O\left(\frac{1}{\log\log n}\right)\right) n$.

Proof. We first argue about a single random contraction step. Consider the clusters formed if we removed all edges with $\ell^*(e) > 1/\gamma^*$. Each cluster has min-ratio cut value $\geq \gamma^*$. Suppose a cluster has n_i vertices. We use the following claim to bound the cluster size after random contraction:

Claim 4.7. *Suppose G has min-ratio cut* π. *Then* $H \sim G(q)$ *has vertex size at most* $1+(1+\varepsilon)q^{\pi}n$ *with probability at least* $1-\exp(-\Omega(\varepsilon^2 n))$.

Then, by Claim 4.7, the size of the cluster after the random contraction step is at most $1 + (1+\delta)q^{\gamma^*}n_i$. By the definition of γ^*, there are at most δn clusters. Therefore, the number of vertices after contraction is at most $\delta n + (1+\delta)q^{\gamma^*}n$. In the recursive contraction algorithm, we chose q such that $q^{\gamma} = \frac{1}{2}$. We also have that $\gamma \leq \gamma'$, which means $q^{\gamma'} \leq \frac{1}{2}$. By Fact 4.4, we have $\gamma^* \geq (1-\delta)\gamma'$. This implies that

$$q^{\gamma^*} \leq (q^{\gamma'})^{1-\delta} \leq (1/2)^{1-\delta} \leq 1/2 + O\left(\frac{1}{\log\log n}\right).$$

So, the number of vertices is at most $\left(\frac{1}{2} + O\left(\frac{1}{\log\log n}\right)\right) n$ whp.

The failure probability of the bound above in a single random contraction step is $\exp(-\Omega(\delta^2 n))$ by Claim 4.7. So, we can apply a union bound over the polynomially many random contraction steps and conclude the statement of the lemma. □

4.4 Relative Variance of the Estimator

We first bound the relative second moment of a random contraction step. By Lemma 1.4, in the unreliable case, $z_G(p)$ is a sufficiently good approximation to $u_G(p)$, where $z_G(p)$ is the expected number of failed cuts. So, instead of analyzing the relative variance of $u_H(p/q)$, we analyze the relative variance of $z_H(p/q)$. We bound its second moment below (similar to Lemma 4.8 in [2]):

Lemma 4.8. *The relative second moment of* $z_H(p/q)$ *is at most* $2 + O\left(\frac{\log\log n}{\log n}\right)$ *where* $H \sim G(q)$ *and* $q^{\gamma} = 1/2$.

We now use the following lemma from [2]:

Lemma 4.9 (Lemma 4.6 of [2])**.** *Assume* $p < \theta$. *If* $z_H(p')$ *is an unbiased estimator of* $z_G(p)$ *with relative variance* η, *and* $u_H(p')$ *is an unbiased estimator of* $u_G(p)$, *then* $u_H(p')$ *has relative variance at most* $\left(1 + O\left(\frac{1}{\log n}\right)\right)\eta + O\left(\frac{1}{\log n}\right)$.

Combining Lemmas 4.8 and 4.9, we obtain the following:

Corollary 4.10. $u_H(p/q)$ *is an unbiased estimator of* $u_G(p)$ *with relative second moment at most* $2 + O\left(\frac{\log\log n}{\log n}\right)$.

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 in Lemma 4.2.

Lemma 4.11. *The second moment of the estimator given by the overall algorithm is at most* $(\log n)^{O(\log\log n)} \cdot (u_G(p))^2$ *whp.*

Proof. Let X be the estimator given by the overall algorithm. We use induction on recursion depth to prove

$$\mathbb{E}[X^2] \leq (\log n)^{K\log\log n} \cdot (u_G(p))^2$$

for a large constant K to be decided later.

Consider the base cases of the recursion. In the first case (Karger's algorithm) and the second case (Monte Carlo sampling), we get an unbiased estimator of $u_G(p)$ with relative variance at most 1 by Theorem 1.2 and Lemma 1.3. In the last base case, we get an estimator of $u_G(p)$ with relative bias at most $n^{-\Omega(1)} < 0.1$ and relative variance at most 1 by Lemma 1.5. For such an estimator Y, we have $\mathbb{E}[Y] \leq 1.1u_G(p)$ and $\mathbb{E}[Y^2] - \mathbb{E}^2[Y] \leq \mathbb{E}^2[Y]$. The latter can be rewritten as

$$\mathbb{E}[Y^2] \leq 2 \cdot \mathbb{E}^2[Y] \leq 2 \cdot (1.1u_G(p))^2 \leq 3 \cdot (u_G(p))^2.$$

Therefore, the lemma holds for all the base cases.

Next, we consider the inductive step where the algorithm takes the average of two recursive calls. Let X_1 and X_2 be the estimators returned by the two recursive calls; then, $X = \frac{X_1+X_2}{2}$. We have two cases depending on whether we are in a random contraction step or a sparsification step. First, we consider a random contraction step. The sparsification step is similar and discussed at the end.

Let H_1 and H_2 be the graphs resulting from random contraction. Now, we have

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

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

By the inductive hypothesis,

$$\mathbb{E}[X_i^2|H_i] \le (\log n_i)^{K \log\log n}(u_{H_i}(p/q))^2, \text{ for } i = 1, 2.$$

By applying Lemma 4.3 on the recursive calls, we have $\mathbb{E}[X_i|H_i] \le 1.1 u_{H_i}(p/q)$. We can now bound the second moment of X as follows:

$$\begin{aligned}
\mathbb{E}[X^2] \le& \frac{1}{4} \cdot \mathbb{E}\Big[(\log n_1)^{K\log\log n}(u_{H_1}(p/q))^2 \\
&+(\log n_2)^{K\log\log n}(u_{H_2}(p/q))^2\Big] + \frac{1}{2}\mathbb{E}[1.1^2 u_{H_1}(p/q)u_{H_2}(p/q)] \\
\le& \frac{1}{4}\cdot(\log n_1)^{K\log\log n}\cdot\mathbb{E}[(u_{H_1}(p/q))^2] \\
&+\frac{1}{4}\cdot(\log n_2)^{K\log\log n}\cdot\mathbb{E}[(u_{H_2}(p/q))^2] \\
&+\mathbb{E}[u_{H_1}(p/q)]\cdot\mathbb{E}[u_{H_2}(p/q)],
\end{aligned} \tag{4.3}$$

by independence of H_1, H_2.

To bound the first two terms in Equation (4.3), note that by Corollary 4.10, we have

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

To bound the last term in Equation (4.3), note that $\mathbb{E}[u_{H_i}(p/q)] = u_G(p)$ since the random contraction step is unbiased. Thus,

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

Putting all these bounds together, we get

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

Lemma 4.6 implies that $n_i \le 0.9n$ across all steps of recursion whp. Then, $\log n_i \le \log n - 0.1$ in all the recursion step whp. We get

$$\begin{aligned}
\mathbb{E}[X^2] \le& (\log n - 0.1)^{K\log\log n}\left(1 + O\left(\frac{\log\log n}{\log n}\right)\right) \\
&\cdot(u_G(p))^2 + (u_G(p))^2.
\end{aligned}$$

Therefore, by choosing a large enough constant K, we ensure that

$$\begin{aligned}
\frac{\mathbb{E}[X^2]}{(u_G(p))^2} \le& (\log n)^{K\log\log n}\left(1 - \frac{0.1}{\log n}\right)^{K\log\log n} \\
&\cdot\left(1 + O\left(\frac{\log\log n}{\log n}\right)\right) + 1 \le (\log n)^{K\log\log n}.
\end{aligned}$$

For a sparsification step, we can repeat the same argument but define H_1, H_2 as the sparsifiers instead of the contracted graphs. By Lemma 4.2, $u_{H_i}(q')$ is also an unbiased estimator of $u_G(p)$ with relative second moment at most $2 + O\left(\frac{1}{\log n}\right)$. The only caveat is that the vertex size does not decrease in the sparsification step itself. However, the vertex size decreases in the next step since it is a random contraction step. So, we can repeat the same argument by combining two steps (with relative second moment $4 + O(\frac{1}{\log n})$ and 4 branches). □

4.5 Running Time

Lemma 4.12. *The algorithm runs in $m^{1+o(1)}\varepsilon^{-O(1)}$ 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 $n \le \varepsilon^{-O(1)}$), the running time is $\tilde{O}(n^2) = \varepsilon^{-O(1)}$ by Theorem 1.2. The second and third base cases take $m^{1+o(1)}$ time by Lemmas 1.3 and 1.5. In a sparsification node, the sparsification algorithm in Lemma 4.2 takes $O(m)$ time. In a contraction node, randomly contracting edges also takes $O(m)$ time. So, the time spent at any node of the computation tree is $m^{1+o(1)}$ in total (including the charge received by a sparsification node from its children irregular contraction nodes).

Lemma 4.6 shows that whp each recursive contraction reduces the vertex size by a factor of $\frac{1}{2} + O(1/\log\log n)$. 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 $m \le n^{1+o(1)}$ (which implies $m^{1+o(1)} \le n^{1+o(1)}$). Therefore, the recurrence is

$$T(n, m) \le n^{1+o(1)} + 2\cdot T\left(\left(1/2 + O\left(1/\log\log n\right)\right)\cdot n, m\right). \tag{4.4}$$

Now, we consider a non-root sparsification node. This is more complicated because $m > n^{1+\Omega(1/\sqrt{\log n})}$. Recall that the running time incurred at this node (including that inherited from irregular contraction children) is $m^{1+o(1)}$. Consider 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 $m \le \hat{m} \le \hat{n}^{1+O(1/\sqrt{\log n})}$. In this case, we can charge the $m^{1+o(1)}$ term to the parent's recurence relation Equation (4.4). Otherwise, the parent is a sparsification node. We have the following property:

Lemma 4.13. *Assume $m > n^{1+\Omega(1/\sqrt{\log n})}$ 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 hold in expectation: $m' = \tilde{O}(n)$ and $(m')^{1/\sqrt{\log n}} \le 1/2\cdot m^{1/\sqrt{\log n}}$.*

Lemma 4.13 implies that $m = \tilde{O}(\hat{n})$ in expectation. So, we can also charge $m^{1+o(1)}$ to the parent node. 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:

$$T(n,m) \le n^{1+o(1)} + 4 \cdot T\left(\left(1/2 + O\left(1/\log\log n\right)\right) \cdot n, m'\right), \qquad (4.5)$$

where m' satisfies Lemma 4.13.

For the sake of the master theorem, we define a potential $\rho = n \log n \cdot m^{1/\sqrt{\log n}}$. The progress in the first term is

$$n' \le \left(1/2 + O\left(1/\log\log n\right)\right) n, \text{ so}$$
$$n' \log n' \le \left(1/2 + O\left(1/\log\log n\right)\right) n \cdot (\log n - \Omega(1)) \le 1/2 \cdot n \log n.$$

Lemma 4.13 measures the progress in $m^{1/\sqrt{\log n}}$ for the second recurrence (Equation (4.5)) by $m' = \tilde{O}(n)$ and $m'^{1/\sqrt{\log n}} \le 1/2 \cdot m^{\frac{1}{\sqrt{\log n}}}$. We have one of the following recurrence relations:

$$T(\rho) = \rho^{1+o(1)} + 2T(\rho/2)$$
$$T(\rho) = \rho^{1+o(1)} + 4T(\rho/4).$$

In either case, this solves to $T(\rho) = \rho^{1+o(1)}\varepsilon^{-O(1)}$ (including the base cases discussed above). Therefore,

$$T(n,m) = n^{1+o(1)}\varepsilon^{-O(1)}.$$

Since the root node takes $m^{1+o(1)}$ time which isn't chargeable elsewhere, the expected runtime overall is $m^{1+o(1)}\varepsilon^{-O(1)}$. □

So, we have obtained an algorithm that runs in $m^{1+o(1)}\varepsilon^{-O(1)}$ time and obtains an estimator for $u_G(p)$ with relative variance $n^{o(1)}$. We repeat $n^{o(1)}\varepsilon^{-2}$ times to get the $(1 \pm \varepsilon)$-approximation for $u_G(p)$ whp, thereby establishing Theorem 1.1.

5 Conclusion

In this paper, we gave an almost-linear time algorithm to compute the unreliability of an undirected graph. Up to lower order (subpolynomial) improvements, this brings to a close the line of work on designing fast network unreliability algorithms. However, many related problems remain open. The complementary problem of network reliability that estimates the probability that an undirected graph stays connected when every edge fails independently with some probability p, does not have close to linear time algorithms, even in the dense case when $m = \Theta(n^2)$ [8, 9]. Another interesting direction is to consider more complex forms of edge failure, e.g., by allowing limited dependence such as the unreliability problem on hypergraphs where all constituents edges of a hyperedge fail together. For this problem, obtaining a polynomial-time approximation scheme remains open [3].

6 Acknowledgments

RC and DP were supported in part by NSF grants CCF-1955703 and CCF-2329230. Part of this research was conducted at the Simons Institute for the Theory of Computing, UC Berkeley under the aegis of the program on "Data Structures and Optimization for Fast Algorithms". We acknowledge the hospitality and support of the institute. DP would also like to acknowledge the support of Google Research and UC Berkeley for hosting him during his sabbatical from Duke University in 2023-24.

References

[1] Amir Abboud, Jason Li, Debmalya Panigrahi, and Thatchaphol Saranurak. 2023. All-Pairs Max-Flow is no Harder than Single-Pair Max-Flow: Gomory-Hu Trees in Almost-Linear Time. In *Proceedings of the 64th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2204–2212.

[2] Ruoxu Cen, William He, Jason Li, and Debmalya Panigrahi. 2024. Beyond the Quadratic Time Barrier for Network Unreliability. In *Proceedings of the 35th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1542–1567.

[3] Ruoxu Cen, Jason Li, and Debmalya Panigrahi. 2024. Hypergraph Unreliability in Quasi-Polynomial Time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*. 1700–1711.

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

[5] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. 2022. Maximum Flow and Minimum-Cost Flow in Almost-Linear Time. In *Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*. 612–623.

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

[7] Ralph E Gomory and Tien Chung Hu. 1961. Multi-terminal network flows. *J. Soc. Indust. Appl. Math.* 9, 4 (1961), 551–570.

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

[9] Heng Guo and Mark Jerrum. 2019. A Polynomial-Time Approximation Algorithm for All-Terminal Network Reliability. *SIAM J. Comput.* 48, 3 (2019), 964–978.

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

[11] David R. Karger. 1999. A Randomized Fully Polynomial Time Approximation Scheme for the All-Terminal Network Reliability Problem. *SIAM J. Comput.* 29, 2 (1999), 492–514.

[12] David R. Karger. 2016. A Fast and Simple Unbiased Estimator for Network (Un)reliability. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 635–644.

[13] David R. Karger. 2017. Faster (and Still Pretty Simple) Unbiased Estimators for Network (Un)reliability. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 755–766.

[14] David R. Karger. 2020. A Phase Transition and a Quadratic Time Unbiased Estimator for Network Reliability. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing*. 485–495.

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

[16] C St JA Nash-Williams. 1961. Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society* 1, 1 (1961), 445–450.

[17] Mikkel Thorup. 2001. Fully-dynamic min-cut. In *Proceedings of the 33rd Annual ACM Symposium on Theory of computing*. 224–230.

[18] Mikkel Thorup. 2008. Minimum k-way cuts via deterministic greedy tree packing. In *Proceedings of the 40th Annual ACM Symposium on Theory of computing*. 159–166.

[19] W. T. Tutte. 1961. On the Problem of Decomposing a Graph into n Connected Factors. *Journal of the London Mathematical Society* s1-36, 1 (01 1961), 221–230.

[20] Leslie G. Valiant. 1979. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.* 8, 3 (1979), 410–421.

[21] Jan van den Brand, Li Chen, Richard Peng, Rasmus Kyng, Yang P. Liu, Maximilian Probst Gutenberg, Sushant Sachdeva, and Aaron Sidford. 2023. A Deterministic Almost-Linear Time Algorithm for Minimum-Cost Flow. In *Proceedings of the 64th Annual IEEE Symposium on Foundations of Computer Science, FOCS*. 503–514.

Received 2024-11-03; accepted 2025-02-01