58th Annual IEEE Symposium on Foundations of Computer Science

Faster (and still pretty Simple) Unbiased Estimators for Network (Un)reliability

David R. Karger

***Abstract*—Consider the problem of estimating the (un)reliability of an n-vertex graph when edges fail with probability p. We show that the Recursive Contraction Algorithms for minimum cuts, essentially unchanged and running in $n^{2+o(1)}$ time, yields an unbiased estimator of constant relative variance (and thus an FPRAS with the same time bound) whenever $p^c < n^{-2}$. For larger p, we show that reliable graphs—where failures are rare so seemingly harder to find—effectively act like small graphs and can thus be analyzed quickly. Combining these ideas gives us an unbiased estimator for unreliability running in $\tilde{O}(n^{2.78})$ time, an improvement on the previous $\tilde{O}(n^3)$ time bound.**

I. Introduction

In the network reliability problem we are given a graph G and a failure probability p and seek to compute the probability $u_G(p)$ that G becomes disconnected when edges fail independently with probability p.

This problem is ♯P-complete [1], [2]. Karger [3] gave the first *fully polynomial randomized approximation scheme (FPRAS)* for $u_G(p)$, with a running time of roughly $\tilde{O}(n^5)$.[1] Harris and Srinivasan [4] improved the runtime to $n^{3+o(1)}$ at the cost of a more complex algorithm and analysis. Most recently, Karger [5] gave a simple algorithm with running time $O(mn^2)$ which, combined with standard sparsification methods, yielded a runtime of $\tilde{O}(n^3)$. All of the previous work shared a high level structure, applying simple naive Monte Carlo estimation when the min-cut c satisfies $p^c \geq n^{-2}$ and more sophisticated methods when $p^c \leq n^{-2}$.

In this work, we improve both approaches. We give a fast and simple $n^{2+o(1)}$-time algorithm for the case $p^c \leq n^{-2}$. Rather surprisingly, the algorithm for doing so is a familiar one: the Recursive Contraction Algorithm for finding minimum cuts [6]. The algorithm runs almost unchanged, copying and generating contracted graphs recursively. But instead of returning a cut, each leaf node now returns a reliability estimate (for a tiny graph); the average of all the estimates forms an *unbiased estimator* of constant relative variance which yields an FPRAS in a straightforward fashion.

While the Recursive Contraction Algorithm was previously known, our primary contribution is the analytic work needed to demonstrate that it can correctly estimate network reliability. We develop techniques for both upper and lower bounding the reliability of networks; these techniques may yield additional applications in network reliability or in the combinatorics of graph cuts more generally.

With this progress for small p (hopefully a common case in practice, where reliable graph are preferred), we must address the time taken by the naive Monte Carlo estimation algorithm when p is large. Here, our main contribution is to demonstrate that $\tilde{O}(n^3)$ is *not* a fundamental limit. We give an algorithm with runtime $\tilde{O}(n^{2.78})$ for this case. While only a small improvement, this result also relies on interesting new analytic methods. In particular, we define an *effective size* for graphs that can be used to analyze random contraction algorithms; reliable graphs have a smaller effective size because they contract "faster."

A. Related Work

Earlier reliability algorithms were based on cut enumeration. Karger [7] showed that for small p the value $u_G(p)$ is approximately equal to the probability that a *small* cut fails and that there are few such small cuts. One can therefore enumerate them and apply a DNF-counting algorithm [8] to approximate this approximation. Harris and Srinivasan [4] dug much deeper into this analysis, giving better, reliability-dependent bounds on the number of small cuts, faster algorithms for enumerating them, and data-structures for accelerating estimation over them.

Karger [9] showed that explicit cut enumeration is not necessary, demonstrating instead that simply generating

MIT Computer Science and AI Laboratory, Cambridge, MA 02138.
email: karger@mit.edu.
URL: http://people.csail.mit.edu/karger

[1]Technically, we are estimating network *unreliability*. For exact algorithms the two problems are identical, but an approximation to one does not translate to the other. Estimating reliability—i.e. the likelihood of staying connected—is most difficult on unreliable graphs which are unimportant in practice. There is no known FPRAS for this problem. But for reliable graphs an FPRAS for reliability is trivial while estimating unreliability is the harder problem.

0272-5428/17 $31.00 © 2017 IEEE
DOI 10.1109/FOCS.2017.75

a small random contraction of G and estimating *its* reliability is sufficient to estimate $u_G(p)$. In a sense, the random contraction selects a random sample of the cuts whose failure probability, appropriately scaled, provides sufficient information about all the original cuts. In addition to skipping the complexity of cut enumeration algorithms and data structures, this approach yields an *unbiased estimator*—an output whose expectation is exactly equally to $u_G(p)$. Since the *relative variance* of this estimator is small, it can easily be transformed into an FPRAS.

In this work, we take a deeper dive into the analysis of the relative variance of the reliability of random subgraphs. Several of our core lemmas are identical to those of Harris and Srinivasan [4], although our proofs are somewhat different, relying less on detailed numerical arguments and more on combinatorial thought experiments. And our *use* of these lemmas is quite different: instead of applying them to cut enumeration, we apply them to better relative variance bounds. This lets us maintain the algorithmic simplicity of the previous estimator approach [9] while still achieving better runtimes than were previously known.

B. Our Approach

Our approach is based on a technique and a conjecture about it from previous work. We wish to estimate the probability $u_G(p)$ that G becomes disconnected when edges fail with probability p producing a random graph $G(p)$. If $u_G(p)$ is large we can do so by direct experiment, deleting edges and testing with G is connected. This *naive Monte Carlo (NMC)* algorithm requires roughly $1/u_G(p)$ trials for a good estimate.

When $u_G(p)$ is small NMC is too slow. In previous work [5] we derived a natural *unbiased estimator* for this problem. To construct $G(p)$, consider *marking* each edge for possible deletion with probability q, then deleting each marked edge with probability p/q. This deletes each edge with probability $(p/q) \cdot q = p$ as desired. But unmarked edges are never deleted, so the only cuts that can fail are those with all edges marked. Thus, we can contract all unmarked edges before the second deletion step, yielding a graph H distributed as a sample from $G(q)$. The probability $u_G(p)$ that $G(p)$ is disconnected is then the probability $H(p/q)$ is disconnected, namely $u_H(p/q)$, meaning $u_G(p) = E_H[u_H(p/q)]$. In other words, $u_H(p/q)$ is an *unbiased estimator* for $u_G(p)$. This means that $u_G(p)$ can be estimated by repeatedly generating samples H and averaging their reliabilities.

The key to this approach is bounding the *relative variance* of this estimator. If the relative variance is r then combining $O(r)$ samples yields an FPRAS:

Definition I.1. The *relative variance* of random variable X is the ratio of its variance to its squared mean, which is $\frac{E[X^2]}{E[X]^2} - 1$.

The following lemma is a well-known application of the Chebyshev bound.

Lemma I.2. *If X has relative variance r, then the median of $O(\log 1/\delta)$ averages of $O(r/\epsilon^2)$ independent samples of X, totaling $O(r\epsilon^{-2} \log 1/\delta)$ samples, yields an ϵ approximation to $E[X]$ (i.e., a value in the range $(1 \pm \epsilon)E[X]$) with probability $1 - \delta$.*

For NMC, the unbiased estimator is the indicator variable for whether $G(p)$ is disconnected in an experiment; it has relative variance $1/u_G(p)$. For our faster algorithm, we generate H and use $u_H(p/q)$ as our estimator X. In our previous work [5], we showed that if $q^c = 1/n^2$ then the relative variance of $u_H(p/q)$ is $O(n^2)$, meaning that $O(n^2)$ samples suffice to estimate $u_G(p)$. Furthermore, each graph H generated this was is tiny, so computing $u_H(p/q)$ is easy.

That analysis was tight, in that two cliques connected by c edges do have relative variance n^2 for the given q. However, that example more generally has relative variance q^{-c} for *every* value of q. Considering this example suggested it is the worst case and motivated a conjecture that a q^{-c} bound on relative variance holds for *every* G and q. This would suggest using a larger value of q to generate samples, so that fewer samples would be needed.

In this work, we prove a good part of the conjecture. We show that when $p^c \leq n^{-2}$, the relative variance of $G(q)$ is $O(q^{-c} - 1)$. This is sufficient to yield a fast algorithm for this range of p values. Like the previous work, we generate several instances H and average their reliability $u_H(p/q)$ to estimate $u_G(p)$. But leveraging our conjecture, we set $q^c = O(1)$ and generate only a constant number of samples to speed up our algorithm. The samples are large, but we can apply our algorithm recursively to estimate their reliabilities. The resulting algorithm is essentially the Recursive Contraction Algorithm [10], modified to return reliability estimates instead of cuts.

The bulk of this paper is devoted to proving the conjecture. Along the way, we develop analytic machinery and new reliability bounds. We give a *lower bound* for network reliability, showing that when $p^c \leq n^{-2}$ that $u_G(p)$ is close to the sum of cut failure probabilities (i.e., the union bound is tight). To do so, we give new results on cut failures *conditioned* on the failure of other cuts. And to do this, we give a useful generalization of the analysis of the stochastic cut-selection process defined by

the Contraction Algorithm. Much of this work parallels but simplifies that of Harris and Srinivasan [4].

Because our algorithm is so fast for small p, we face a different bottleneck from earlier algorithms: large p where we must revert to NMC. Here we make less progress, but we give a new analysis based on the idea of *effective size* of a graph that lets us improve the runtime to $\tilde{O}(n^{2.78})$. Essentially, in Monte Carlo simulation it is the rare events that are slowest to measure—in this case, that is highly reliable graphs that rarely disconnect. We develop a new analytic method to show that these highly reliable graphs act like smaller graphs from the perspective of contraction algorithms, which permits us to apply our "small p" algorithm even when p exceeds n^{-2}.

II. The Recursive Estimator

Our estimator relies on two lemmas:

Lemma II.1 ([5])**.** *The number of vertices in $G(q)$ is stochastically dominated by a binomial distribution with parameters n and $q^{c/2}$, so has expectation at most $q^{c/2}n$ and is $O(q^{c/2}n)$ with high probability.*

Lemma II.2. *When $p^c \leq n^{-2}$ and $q^c \geq n^{-2}$, the estimator $u_H(p/q)$ has relative variance $O(q^{-c}-1)$.*

This first lemma was proven previously [5] by giving a coupling argument that showed the cycle ends up with more components than any other graph given n, c, and p, then exactly analyzing the cycle. Corollary VI.3 below gives an alternative proof of the first lemma. We prove the second lemma in Section V.

The idea of our construction is to recurse this basic estimator: we generate the above estimator with relative variance $O(q^{-c})$, then recursively estimate the value of the estimator.

A. Composing Unbiased Estimators

Because we're generating unbiased estimators recursively, we need to understand how they compose.

Lemma II.3. *Let X be some quantity and suppose we sample an unbiased estimator Y for X with relative variance r, then sample an unbiased estimator Z for Y with relative variance s. Then Z is an unbiased estimator for X with relative variance $(s+1)(r+1)-1$.*

Proof: Note that $E[Z \mid X] = E_Y[E[Z \mid Y] \mid X] = E_Y[Y \mid X] = X$, meaning Z is also an unbiased estimator for X. For the relative variance, recall that a relative variance of s means that $E[Z^2 \mid Y] \leq (s+1)E[Z \mid Y]^2$. Thus

$$\begin{aligned} E[Z^2 \mid X] &= E_Y[E[Z^2 \mid Y] \mid X] \\ &\leq E_Y[(s+1)E[Z \mid Y]^2 \mid X] \end{aligned}$$

by the relative variance of Z for Y

$$\begin{aligned} &= (s+1)E[Y^2 \mid X] \\ &\leq (s+1)(r+1)E[X]^2 \end{aligned}$$

by the relative variance of Y for X.

■

In other words, composing unbiased estimators multiplies their relative variances. We combine this idea with the fact that repeating unbiased estimators divides their variance:

Lemma II.4. *If X has relative variance r then the average of k samples of X has relative variance r/k.*

B. The Algorithm

For simplicity, we first design and analyze our algorithm under the assumption that the relative variance of the estimator $u_H(p/q)$ is exactly $q^{-c}-1$ for all p. Afterward, we'll show how to tweak both to handle the range limitations and extra constant factor that our proofs produce. Our recursive algorithm is as follows. First, if G has less than some constant number of vertices that will follow from our analysis, we compute $u_G(p)$ exactly. And if $p^c > 1/2$, we simply perform one naive Monte Carlo trial, failing each edge with probability p, and return 0 or 1 as the estimate if the graph is connected or disconnected respectively. Otherwise, we solve the problem recursively. Choose $q = 2^{-1/c}$ such that $q^c = 1/2$ so the relative variance of $u_H(p/q)$ is $q^{-c}-1 = 1$ (note $q > p$ so $u_H(p/q)$ is defined). Generate two independent subgraphs $H_1, H_2 \sim G(q)$, recursively compute the estimators $u_{H_1}(p/q)$ and $u_{H_2}(p/q)$, then average the two results as an estimator for $u_G(p)$.

We now analyze our algorithm inductively. In the base cases of constant n or $p^c > 1/2$ we perform only linear work. If we compute $u_G(p)$ exactly then of course our relative variance is 0; if on the other hand $p^c > 1/2$ we run a naive Monte Carlo step whose relative variance is $1/u_G(p) - 1 \leq 1/p^c - 1 \leq 1$.

Lemma II.1 indicates that the two graphs H_i will have at most $q^{c/2}n = n/\sqrt{2}$ vertices in expectation. For an intuitive analysis of our algorithm, let's assume this is a deterministic bound instead of a probabilistic one. This yields a runtime recurrence $T(n) = O(n^2) +$

$2T(n/\sqrt{2})) = \tilde{O}(n^2)$, which is exactly the same recurrence as for the Recursive Contraction Algorithm.

We can outline a similar recurrence for the relative variance. Suppose we prove that our estimator's relative variance on the smaller graphs is R. That is, for each H_i we get an unbiased estimator X_i for $u_H(p/q)$ with relative variance R. We also know that $u_H(p/q)$ is itself an unbiased estimator for $u_G(p)$ with relative variance $q^{-c} - 1 = 1$. Thus, by Lemma II.3, X_i is an unbiased estimator for $u_G(p)$ with relative variance $(R+1)(1+1)-1 = 2R+1$. But our algorithm performs 2 independent experiments and takes the average, which by Lemma II.4 halves the relative variance to $R+1/2$. In other words, our recursive step adds $1/2$ to the relative variance of our subproblem's estimator.

This yields a recurrence for R_n, the worst-case relative variance of our estimator on any graph of n or fewer vertices. Assuming the H_i have $n/\sqrt{2}$ vertices, we deduce that $R_n \leq R_{n/\sqrt{2}} + 1/2$, which implies that $R_n = O(\log n)$.

We've given intuitive arguments for both the runtime and the relative variance of our estimator, but they both relied on assuming the subproblems have size $n/\sqrt{2}$ deterministically. In fact this is only true with high probability. The randomness involved makes a formal analysis more complicated.

C. Formalizing The Runtime

We formalize the runtime analysis of our recursive algorithm. Note that at depth d of the recursion, the accumulation of sampling steps has produced a graph sampled as $G(q^d)$. Thus, by Lemma II.1, the number of vertices in each graph at this level is stochastically dominated by $B(n, 2^{-d/2})$.

For $d \leq 2\log n$, we conclude that the expected number of vertices in each such graph is $2^{-d/2}n$ while the expected number of edges is the expected *square* of this quantity. But this expected square is just the variance plus the expectation squared. Since the variance of the binomial distribution is also $O(2^{-d/2}n)$, we conclude that the expected number of edges is $O(2^{-d}n^2)$. The work spent at each recursion node is linear in this number of edges. Since there are 2^d recursion nodes at level d, the expected total work at level d is $O(n^2)$. Thus over all $O(\log n)$ levels the work is $O(n^2 \log n)$.

We analyze deeper levels slightly differently. In particular, note that the work done at a *leaf* of the recursion tree is dominated by the work done at its parent, so we can bound the overall work by bounding the work at *non-leaf* nodes. At level $d + \log n^2$, the number of vertices in a subproblem is distributed as $B(n, 2^{-d/2}/n)$. Thus, the probability there are more than k vertices is at most $\binom{n}{k}(2^{-d/2}n)^k \leq (e \cdot 2^{-d/2}/k)^k$. The number of potential recursion nodes at depth $d+\log n^2$ is $2^{d+\log n^2}$. Thus, taking k to be the size at which we terminate the recursion, the expected number of non-leaf nodes at level $d+\log n^2$ is at most $2^{d+\log n^2}(e2^{-d/2}/k)^k = n^2 2^{-\Omega(kd)}$; in other words, it is geometrically decreasing in d. Summing over all $d \geq 1$ yields a bound of $O(n^2)$ additional work, which leaves our overall $O(n^2 \log n)$ bound unchanged.

This expected time bound can be shown to hold with high probability using standard methods.

D. Formalizing the Relative Variance

We now formalize the inductive analysis of the relative variance of our estimator. Let R_n denote its worst-case relative variance on any graph of n or fewer vertices; note that by definition R_n is increasing in n.

Lemma II.1 asserts that the number of vertices in $G(q)$ is bounded by $B(q^{c/2}, n)$. Thus, if $q^c = 1/2$, the expected size is $n/\sqrt{2}$. It follows by a Chernoff bound that $\Pr[B(q^{c/2}, n) \geq 4n/5] = e^{-\delta n}$ for some constant δ. We halt our recursion and solve the problem exactly whenever $n \leq (\ln 2)/\delta$, which ensures that the probability of the "rare" event is at most $1/2$.

We can use this to bound our subproblems' relative variances. We've just argued that H_i has at most $4n/5$ vertices with probability at least $1/2$, in which case the recursive estimator's relative variance is at most $R_{4n/5}$. In the "rare" event that H_i is larger, it has size at most n and thus the estimator's relative variance is at most R_n. Since relative variance is an expectation, we can combine these two cases using conditional expectation and conclude that relative variance for each sub-problem's estimator is at most $(1/2)R_{4n/5} + (1/2)R_n$.

We argued in the previous subsection that (under our current assumption that the relative variance of $u_H(p/q) = q^{-c}-1$) the relative variance of the estimator for G is at most $1/2$ greater than that of the estimator on the subproblems H_i. This yields the following recurrence:

$$\begin{aligned} R_n &\leq (1/2)R_{4n/5} + (1/2)R_n + 1/2 \\ (1/2)R_n &\leq (1/2)R_{4n/5} + 1/2 \\ R_n &\leq R_{4n/5} + 1 \\ &= O(\log n) \end{aligned}$$

Note how we took advantage of the fact that the relative variance is itself an expectation. We don't qualify our bound on the relative variance as "high probability." Rather, we have derived a bound on the *expectation*

that is universally true, although of course there can be cases where the returned value deviates greatly from this expectation.

E. Handling the Restriction on p

Our algorithm relies on a variance bound that we can only prove (below) when $p^c = O(n^{-2})$. While we take this as given in the initial call to the algorithm, we must also consider the recursive calls which arrive with their own p and n. In the event that $p^c > n^2$ in a recursive invocation, we fall back on Naive Monte Carlo, performing n^2 trials of failing edges and testing whether the graph disconnects, and reporting the fraction of failures. One such trial has relative variance $1/p^c \leq n^2$, meaning that n^2 such trials have relative variance $O(1)$ as assumed in the induction. Thus, our analysis of the relative variance of our algorithm remains valid. A single trial takes time m, meaning this algorithm has runtime $O(mn^2)$, which seems problematic for our target runtime. However, we will now argue that with high probability, we will not use this escape hatch on any recursive call whose input size exceeds $O(\log n)$. Thus, the contribution of these Naive Monte Carlo steps is negligible.

The reason is that, as we saw in the runtime analysis, graphs at depth d of the recursion have a number of vertices stochastically dominated by $B(n, q^{dc})$. Thus, they have $n' = O(q^{dc/2}n)$ vertices with high probability until this quantity is $O(\log n)$. At the same time, the probability parameter at this level is $p' = p/q^d$. Thus, with high probability $(n')^2 \cdot (p')^c = O((q^{dc/2}n)^2 \cdot (p^c/q^{dc}) = O(n^2p^c)$. It follows that if initially $p^c \leq P$ for some $P = \Omega(n^{-2})$, then with high probability we will have $p^c \leq n^{-2}$ in all subproblems as well. This implies that with high probability the Naive Monte Carlo escape hatch will not be used on any problem of size exceeding $O(\log n)$. This in turn implies that with high probability we suffer only a polylogarithmic slowdown in our running time compared to the analysis above.

F. Correcting for a Constant Factor

Our analysis above assumed that when $q^c = 1/2$ the relative variance is 1. However, our proof of this conjecture below adds a constant factor, bounding the relative variance by $O(q^{-c} - 1)$.g

To handle this concern, note that the $O(q^{-1} - 1)$ relative variance is at most $k(q^{-c} - 1)$ for some constant k so long as n exceeds some absolute constant. If n is smaller than this constant we can compute the reliability exactly in constant time for relative variance 0. Otherwise, instead of choosing $q^c = 1/2$, we set $q^c = 1/\sqrt{r}$ for some larger constant r, then generate kr distinct subproblems instead of 2. This changes our intuitive runtime recurrence to $T(n) = kr(T(n/\sqrt{r}) + O(n^2))$ which evaluates to $T(n) = (kr)^{\log_{\sqrt{r}} n} = n^{\log_{\sqrt{r}} kr} = n^{2+2\log_r k}$ and can be formalized the same way. Thus, if we set r to be some slowly growing function of n, we get the $n^{2+o(1)}$ time bound claimed for our algorithm.

We've now filled in all the details of our recursive algorithm. It remains to prove the key result that the relative variance is $O(q^{-c})$.

III. Revisiting the Contraction Algorithm

We're going to find several uses of the *Contraction Algorithm* [11] as a tool for analyzing graph structure. Recall that this algorithm repeatedly chooses a random non-loop edge of G and contracts it until some stopping point is reached. We say a cut *survives the contractions* if no edge of that cut is contracted; in this case the cut corresponds to one of the cuts of the contracted graph.

A key application of the Contraction Algorithm has been to bound the *partition function* $z_G(p) = \sum p^{c_i}$ over all cut values c_i in G. This is the expected number of failed cuts in $G(p)$ and thus upper bound $u_G(p)$. We're going to generalize this original analysis in two ways, and a natural way to do this is to present the original analysis and then, later, explain how we change it.

As we contract G from its original size n, let m_r denote the number of non-loop edges present in the contracted graph at size r. Note that m_r is a random variable, but we will lower bound it. If a cut under consideration has αc edges then the probability that an edge of the cut is selected in the next contraction is $\alpha c/m_r$; it follows that the probability the cut survives *all* the contractions is $\prod(1 - \alpha c/m_r)$. It is convenient to approximate this product by a sum as follows.

Lemma III.1 (Product Approximation)**.** *Consider a product $P = \prod_{r \geq 2\gamma}(1 - x_r)$ such that $x_r \leq \gamma/r$ and let $S = \sum_{r \geq 2\gamma} x_r$. Then*

$$2^{-\gamma}\exp(-S) \leq P \leq \exp(-S)$$

This Lemma was proven with calculus by Harris and Srinivasan [4, Lemma 5.4], but we give a different proof.

Proof: The upper bound follows immediately from the bound $1 - x \leq \exp(-x)$. For the lower bound we note that $e^x \geq 1 + x$ which implies that $(1 - x)e^x \geq 1 - x^2$. Introducing an ultimately irrelevant upper index

n, it follows that

$$\begin{aligned}
&P \cdot \exp(S) \\
&\geq \prod_{r=2\gamma}^{n} (1 - x_r^2) \\
&\geq \prod (1 - (\gamma/r)^2) \\
&= \prod (1 - \gamma/r)(1 + \gamma/r) \\
&= \prod_{r=2\gamma}^{n} \left(\frac{r-\gamma}{r}\right)\left(\frac{r+\gamma}{r}\right) \\
&= \prod_{r=\gamma}^{n-\gamma} \left(\frac{r}{r+\gamma}\right) \prod_{r=2\gamma}^{n} \left(\frac{r+\gamma}{r}\right) && \text{re-indexing} \\
&= \prod_{r=\gamma}^{2\gamma-1} \left(\frac{r}{r+\gamma}\right) \prod_{r=n-\gamma+1}^{n} \left(\frac{r+\gamma}{r}\right) && \text{canceling} \\
&\geq \prod_{r=\gamma}^{2\gamma-1} \left(\frac{r}{r+\gamma}\right)
\end{aligned}$$

as the second term exceeds 1

$$\geq \left(\frac{\gamma}{2\gamma}\right)^{\gamma}$$

as $\gamma \leq r < 2\gamma$ in the product

$$= 2^{-\gamma}$$

■

Lemma III.2 (Contraction)**.** *Suppose that in a contraction process that edge count m_r at size r is guaranteed to satisfy the lower bound $m_r \geq rc/2$. Then the probability that a particular set of αc edges survives contraction to size 4α is at least*

$$2^{-2\alpha} \exp\left(-\frac{c}{2} \sum_{r=2\alpha}^{n} \frac{1}{m_r}\right)$$

Proof: The probability the cut survives to size r is simply the product of the probabilities that it survives each contraction, namely $\prod(1 - \alpha c/m_r)$. Write $x_r = \alpha c/m_r$; from $m_r \geq nc/2$ it follows that $x_r \leq 2\alpha/r$. Thus, we can take $\gamma = 2\alpha$ in Lemma III.1 and conclude the claimed result. ■

We can use this approximation to bound the basic Contraction Algorithm:

Corollary III.3 (Cut Survival)**.** *A particular cut C of value αc survives contraction of G to size 4α with probability at least $(\alpha/n)^{2\alpha}$.*

Proof: When G is contracted to r vertices, having min-cut c means it will have minimum degree c and thus at least $rc/2$ edges. The probability p_r that we choose an edge of C at this step is thus at most $\alpha c/(rc/2) = 2\alpha/r$. Setting $\gamma = 2\alpha$ in the above lemma we satisfy its conditions, meaning that

$$\begin{aligned}
\prod_{r=4\alpha}^{n} (1 - p_r) &\geq 2^{-4\alpha} \exp(-\sum_{r=4\alpha}^{n} 2\alpha/r) \\
&= 16^{-\alpha} \exp(2\alpha(H_n - H_{4\alpha})) \\
&\geq 16^{-\alpha} \exp(-2\alpha \ln(n/4\alpha)) \\
&\geq (\alpha/n)^{2\alpha}
\end{aligned}$$

■

This in turn gives us a bound on the partition function $z_G(p) = \sum p^{\alpha_i c}$:

Corollary III.4 (Partition Function)**.** *If $n^2 p^c \leq \alpha$ then $\sum_{\alpha_i \geq \alpha} p^{\alpha_i c} \leq (np^{c/2}/2\alpha)^{2\alpha}$. In particular, if $n^2 p^c \leq 1$ then $z_G(p) \leq n^2 p^c$.*

Proof: Taking the cut values of G as $\alpha_i c$, consider the following algorithm for selecting *one cut* from G. First, choose a target size $r \geq 2$ with probability 2^{1-r}. Then, contract G randomly to size r. Finally, choose a random cut uniformly from the resulting graph.

The probability that we choose a particular cut of value αc is at least the probability that we choose $r = \lceil 2\alpha \rceil$, which is $2^{1-\lceil 2\alpha \rceil} \geq 2^{-2\alpha}$, times the probability the cut survives contraction to size 2α, which is at least $(\alpha/n)^{2\alpha}$ by the preceding corollary, times the probability that we choose this cut from the resulting graph. Since the resulting graph has size 2α, there are $2^{2\alpha-1}$ cuts, so this last probability is at least $2^{1-2\alpha}$. Multiplying, we select the cut with probability

$$2^{-2\alpha} \cdot (\alpha/n)^{2\alpha} \cdot 2^{1-2\alpha} = 2(2\alpha/n)^{2\alpha}$$

Finally, observe that since our algorithm chooses exactly one cut, these cut selection probability must sum to 1. It follows that

$$\sum 2(2\alpha/n)^{2\alpha} \leq 1$$

x

This lets us bound the partition function:

$$\begin{aligned}
\sum_{\alpha_i \geq \alpha} p^{\alpha_i C} &= \sum (2\alpha_i/n)^{2\alpha_i} \cdot p^{\alpha_i c} \cdot (n/2\alpha_i)^{2\alpha_i} \\
&= \sum (2\alpha_i/n)^{2\alpha_i} \cdot (np^{c/2}/2\alpha_i)^{2\alpha_i}
\end{aligned}$$

But by assumption, $np^{c/2} \leq \alpha$. It follows that the second term is always less than 1 and is maximized at minimum $\alpha_i = \alpha$, meaning our sum is

$$\begin{aligned} &\leq \sum (2\alpha_i/n)^{2\alpha_i} \cdot (n^2 p^c/2\alpha)^{2\alpha} \\ &\leq (n^2 p^c/2\alpha)^{\alpha} \sum (2\alpha_i/n)^{2\alpha_i} \\ &\leq (n^2 p^c/2\alpha)^{2\alpha} \end{aligned}$$

where the last step follows from the previous lemma. For $z_G(p)$ we note that all $\alpha_i \geq \alpha$. ■

Remark. In the analysis above, we implicitly took the "size" of G to be the number n of vertices. But essentially all our above lemma actually relied on was that the size decrements with each contraction and that $m_r \geq rc/2$. We'll leverage this flexibility to generalize these results by redefining both "size" and n below.

IV. A Better Unreliability Lower Bound

We turn towards our first key analytic result, that $u_H(p/q)$ has relative variance $O(q^c - 1)$. This requires upper bounding $E[u_H(p/q)^2]/u_G(p)^2$. Because $u_G(p)$ is a messy function, past work used more tractable upper and lower bounds. A natural upper bound is the *partition function* we used in Section III:

Definition IV.1. The *partition function* $z_G(p) = \sum p^{c_i}$ where c_i indexes over the values of all cuts in G.

That is, $z_G(p)$ is the *expected number of failed cuts in* $G(p)$, and thus serves as an upper bound (the union bound) on the probability $u_G(p)$ that at least one cut fails. For a lower bound, past work simply observed that $u_G(p)$ is at least p^c, the probability that a min-cut fails. And in Section III we rederived the result that $u_G(p) \leq z_G(p) \leq n^2 p^c$ when $p^c \leq n^{-2}$ (and of course $u_G(p) \leq 1 \leq n^2 p^c$ if $p^c \geq n^2$). Unfortunately, there is a significant gap between these bounds which prevents any analysis using them from being tight.

In this section, we will prove that $z_G(p)$ also serves as an asymptotic *lower bound* for $u_G(p)$ when $p^c \ll n^{-2}$:

Theorem IV.2. *There is a constant ζ such that if $p^c \leq \zeta n^{-2}$ then $z_G(p) \geq u_G(p) \geq (1/4) z_G(p)$.*

Karger [7] showed this bound for $p^c = O(n^{-4})$; Harris showed it for $p^c = n^{2+\epsilon}$ for any constant ϵ. Here we improve it to the entire range $p^c = O(n^{-2})$ where naive Monte Carlo sampling is difficult. Note that this result is tight, as on the n vertex cycle with $c/2$ parallel edges between neighbors setting $p^c = 4/n^2$ gives $z_G(p) > 4 > 4u_G(p)$.

We devote this section to proving the theorem. For a set (of cut edges) C, let p^C be shorthand for $p^{|C|}$. Since $u_G(p)$ is the probability of the union of individual cut failure events, the inclusion-exclusion bound shows

$$u_G(p) \geq \sum p^{C_i} - \sum p^{C_i \cup C_j},$$

where we sum over all cuts C_i and C_j. Our approach is to prove that under the hypothesis of the theorem,

$$\sum p^{C_i \cup C_j} \leq (3/4) \sum p^{C_i}.$$

The theorem follows immediately.

A. Conditional Cut Failures

Observe that the combined failure of C_i and C_j will break G into more than 3 or 4 pieces, so we essentially need to bound the expected number of *3- and 4-way cuts* that arise from edge failures. To do so, we expand

$$\begin{aligned} \sum_{i,j} p^{C_i \cup C_j} &= \sum_{i,j} p^{C_i} p^{C_j - C_i} \\ &= \sum_i p^{C_i} \sum_j p^{C_j - C_i} \end{aligned}$$

In other words, we separately consider the edges of $C_i \cup C_j$ that are in C_i, and those that are in C_j but not in C_i. Essentially, this analysis is conditioning on cut C_i failing (probability p^{C_i}) then bounding the expected number of other cuts that fail under this condition (probability $p^{C_j - C_i}$). We do so by generalizing the Contraction Algorithm analysis of Section III.

Lemma IV.3. *Given some α-minimum cut A, index the cuts and let $\beta_i c$ be the number of edges of cut i* not *in A. Then if $n^2 p^c \leq \beta$,*

$$\sum_{\beta_i \geq \beta} p^{\beta_i c} \leq 4^{\alpha} (np^{c/2}/\beta)^{2\beta}$$

Proof: We generalize the Contraction Algorithm analysis. Consider a particular cut B with βc edges not in A. Consider running the Contraction Algorithm on the graph G with the edges of A *removed* and halting at $4\beta + 2\alpha$ vertices instead of 4α. We now (re)define the *size* of G when it has r vertices to be be $r - 2\alpha$. When $r + 2\alpha$ vertices remain, G has at least $(r+2\alpha)c/2$ edges; thus $G - A$ has at least $(r+2\alpha)c/2 - \alpha c = rc/2$ edges. It follows that $G - A$ satisfies $m_r \geq rc/2$ for this definition of size, meeting the condition of Contraction Lemma III.2 and Cut Survival Corollary III.3.

This means $G - A$ also satisfies the analysis of Partition Function Corollary III.4 with two small changes. The first is syntactic: the target cut size α used there is now the quantity β we use here. The second is numeric: since the contractions stop at size r, which now means $r + 2\alpha$ vertices, the final selection step of choosing a random cut chooses from $2^{r+2\alpha}$ instead of 2^r cuts; thus the sum of Partition Function Corollary III.4 is increased by a $2^{2\alpha}$ factor. This yields the result of this lemma. ■

B. Cut Triples

Next, we identify some useful structure in the pairs of cuts we need to analyze. We show how each pair is associated in a "three-way duality" relationship with a third cut that will be useful in our analysis. We need to define these triples because one of the three cuts in it is obviously the smallest—a fact we need in the next section.

Consider any two cuts C_i and C_j. Label each vertex with two bits according to which side of each cut it is on. There will be at most 4 distinct combined labels, and there must be at least 3 since otherwise C_i and C_j are the same cut. Thus, C_i and C_j together naturally define a 3 or 4 way cut.

For intuition, consider first the case where C_i and C_j together define a 3-way cut T; for example just consider a triangle on three vertices. Each of C_i and C_j is created by merging two of the 3 components of T. But note that there is a *third* way to merge 2 components of T to produce a complementary cut C_k that we will call the *xor cut* of C_i and C_j. Note further that each cut edge crosses between exactly 2 of the 3 components of T. This determines 3 distinct groups of edges that we will call *semicuts* because each 2-way cut is the union of exactly two of the edge groups. Conversely, each semicut is the intersection of exactly two of the three 2-way cuts.

Although the triangle picture doesn't quite carry over, everything we've just described applies unchanged if C_i and C_j together form a 4-way cut. Label each vertex by two bits x_i and x_j describing which side of each of the two cuts it is on. We now define a third *xor-cut* C_k for C_i and C_j by giving each vertex label $x_k = x_i \oplus x_j$ where $\oplus$ denotes xor. Note that the x_k must take both values since otherwise C_i and C_j would be the same cut. Thus, the x_k define a third *xor cut* of C_i and C_j. Note that C_i is the xor-cut for C_j and C_k and so forth since $(x_i \oplus x_j) \oplus x_j = x_i$. The three cuts exhibit a three-way duality.

Note further that any edge must cross an *even number* of these three cuts—i.e. 0 or 2. To see this, observe that an edge crosses cut i if its endpoints' labels x_i and y_i satisfy $x_i \oplus y_i = 1$ and similarly for cuts j and k. Thus, if we xor the three cut indicator variables we get $(x_i \oplus y_i) \oplus (x_j \oplus y_j) \oplus (x_k \oplus y_k)$. Since the final term is $(x_i \oplus x_j) \oplus (y_i \oplus y_j)$, the entire xor evaluates to 0 which means either 0 or 2 of the terms are 1.

We can therefore group the edges crossing (any of) the three cuts into three groups according to which pair of cuts they cross. Each cut is the union of exactly two of these groups, so we will call each group a *semicut*. Conversely, each semicut is the intersection of two of the cuts.

In summary, for any pair of cuts C_i and C_j, regardless of whether their union defines a 3 or a 4 way cut, there is a third xor-cut C_k such each cut edge is in *exactly one* semicut (intersection of two of the three cuts). This intersection is exactly the edges *not* in the third cut. And finally, each cut is the union of two of the three semicuts.

C. The Lower Bound

We combine our two ideas to upper bound $\sum p^{C_i \cup C_j}$. We have shown that for each $C_i \cup C_j$ in the pairs term, there is a xor cut C_k and two other pairs $C_i \cup C_k$ and $C_j \cup C_k$ in the sum that produce the same union. Group the pairs into these trios of related pairs.

Of course, $p^{C_i \cup C_j}$ is the same for all three pairs in a trio. The trio is characterized by its three semicuts as discussed above. Without loss of generality, let C_i be the *smallest* of the three cuts in the triple, meaning it is the union of the two smallest semicuts. It follows that the third semicut contains at least 1/3 of the edges of the union while C_i contains at most 2/3; thus the semicut is at least half the size of C_i. That is, if we let α_i be the size of C_i and let $\beta_{ij} = \beta_{ik}$ be the number of edges in the semicut $C_j - C_i = C_k - C_i$, then $p^{C_i \cup C_j} = p^{\alpha_i + \beta_{ij}}$ and the total contribution of the three cuts in this triple to the sum is $3p^{\alpha_i + \beta_{ij}}$. To capture this we *assign* cut C_j to C_i with the additional factor 3 to represent the entire triple and ignore the other two pairs. Let A_i denote the set of cuts assigned to C_i from some triple. Note that each C_j may be assigned to many cuts i since it is in many triples.

To compute the entire sum, we sum over each C_i and within we sum over all cuts assigned to C_i. This is valid because every triple is assigned to some C_i. Thus,

$$\begin{aligned}\sum p^{C_i \cup C_j} &= \sum_i \sum_{j \in A_i} 3p^{C_i \cup C_j} \\ &= \sum_i 3p^{C_i} \sum_{j \in A_i} p^{C_j - C_i} \\ &= \sum_i 3p^{\alpha_i c} \sum_{j \in A_i} p^{\beta_{ij} c}\end{aligned}$$

where $\alpha_i c$ denotes the number of edges in cut C_i and $\beta_{ij} c$ denotes the number of edges in $C_j - C_i$. By our choice of cut assignments, $\beta_{ij} \geq \alpha_i/2$.

We now invoke Lemma IV.3. Since all $\beta_{ij} \geq \alpha_i/2 \geq 1/2$, we conclude that

$$\sum_{j \in A_i} p^{\beta_{ij} c} \leq \left(\frac{p^{c/2} n}{\alpha_i/2} \right)^{\alpha_i} \cdot 4^{\alpha_i} \leq (8p^{c/2} n)^{\alpha_i}$$

Since by assumption $np^{c/2} < 1/8$. It follows that

$$3\sum_i p^{\alpha_i c}\sum_j p^{\beta_{ij} c} \le 3\sum_i p^{\alpha_i c}\cdot(8np^{c/2})^{\alpha_i}$$
$$\le (8np^{c/2})\sum p^{\alpha_i c}$$

which is at most $3z_G(p)/4$ if $np^{c/2} \le 3/32$.

We summarize this result as follows:

Lemma IV.4. *When $np^{c/2} \le 1/64$,*

$$\sum p^{C_i \cup C_j} \le \frac{3}{4}\sum p^{C_i}$$

Corollary IV.5. *When $np^{c/2} \le 1/64$,*

$$u_G(p) \ge \frac{1}{4} z_G(p)$$

V. The Relative Variance

We can now bound the relative variance. We focus on bounding $E[X^2]/E[X]^2$, from which we then subtract one. The numerator is $E[u_H(p/q)^2] \le E[z_H(p/q)^2]$. Note that $z_H(p/q)$ is a sum of $(p/q)^{c_i}$ over all cuts that survive. Thus, $E[z_H(p/q)^2]$ expands as a sum over pairs of cuts; cut pair C_i, C_j contributes $(p/q)^{C_i+C_j}$ (denoting a sum of sizes, not a union) if both cuts survive, which happens with probability $q^{C_i \cup C_j}$ (this denoting a union). Thus,

$$E[z_H(p/q)^2] = \sum_{ij} q^{C_i\cup C_j}(p/q)^{C_i+C_j}$$
$$= \sum p^{C_i+C_j}/q^{C_i\cap C_j}$$

The denominator $u_G(p)^2$ can be lower bounded using the two terms of the inclusion-exclusion expansion discussed earlier. That is,

$$u_G(p)^2 \ge ((1-O(np^{c/2}))z_g(p))^2$$
$$= (1-O(np^{c/2}))\sum p^{C_i+C_j}$$

In other words, up to the $1-O()$ factor in the denominator, both numerator and denominator are a sum of terms over all pairs i and j, where the difference is that each term in the numerator has an extra factor of $1/q^{C_i\cap C_j}$.

Thus, to bound the relative variance, we will group corresponding terms of both numerator and denominator into three groups and show that each group has ratio $O(q^c)$; combining will yield the desired result. We get the three groups breaking the terms into three groups: the terms where $i = j$, the terms where $i \ne j$ and $|C_i \cap C_j| < c$, and the terms where $i \ne j$ and $|C_i \cap C_j| \ge c$.

For the first group, terms where $i = j$, the terms in the numerator have the form

$$X = \sum p^{2C_i}/q^{C_i}$$
$$= \sum (p/q)^c p^{C_i} \qquad \text{since } C_i \ge c$$
$$\le (p/q)^c \sum p^{C_i}$$
$$= (p/q)^c z_G(p)$$

Conversely, the denominator is $\Omega(z_G(p)^2)$. Since $z_G(p) \ge p^c$, the ratio of these two quantities is $O(q^{-c})$ as required. For the second group where $C_i \cap C_j < c$, we have

$$Y = \sum_{C_i\cap C_j<c} q^{C_i\cup C_j}(p/q)^{C_i+C_j}$$
$$= \sum_{C_i\cap C_j<c} p^{C_i+C_j}/q^{C_i\cap C_j}$$
$$\le \sum_{C_i\cap C_j<c} p^{C_i+C_j}/q^c$$
$$\le q^{-c}\sum_{C_i\cap C_j<c} p^{C_i+C_j}.$$

Each term in this sum has a corresponding $\Omega(p^{C_i+C_j})$ term in the denominator, so again the ratio is $O(q^{-c})$. Finally, for the third group,

$$Z = \sum_{C_i\cap C_j\ge c} p^{C_i+C_j}/q^{C_i\cap C_j}$$
$$= \sum p^{C_i\cup C_j}(p/q)^{C_i\cap C_j}$$
$$\le (p/q)^c \sum_{C_i\cap C_j\ge c} p^{C_i\cup C_j} \qquad \text{since } C_i\cap C_j \ge c$$
$$\le (p/q)^c z_G(p) \qquad \text{(proven above)}$$
$$\le q^{-c}(z_G(p))^2 \qquad \text{since } p^c \le z_G(p)$$

Thus, the third term also has ratio $O(q^{-c})$.

We've shown that all three terms in the numerator have ratio $O(q^{-c})$, which means that their sum does as well. This concludes our proof that the relative variance is $O(q^{-c}) - 1$.

As this bound on the relative variance was the only detail outstanding, we can now declare the analysis of our $n^{2+o(1)}$-time estimator to be complete.

Remark. The first and third terms in our expansion actually contribute ratios of $O(q^{-c}(p/z_G(p)))$. We used the pessimistic bound $z_G(p) \ge p^c$, bit if $z_G(p)$ is significantly larger then these two terms become negligible and the variance converges to $q^{-c}(1+O(n^2p^c)) \to q^c$ for small p. In particular, if G has more than $O(\log n)$ minimum cuts, we no longer need to perform the constant factor "correction" of Section II-F.

VI. Accelerating the Unreliable Case

Our analysis of our recursive estimator works when $p^c \le n^{-2}$. We suspect that in fact the relative variance is bounded by q^{-c} for *all* values of p, meaning the recursive estimator can always be used. But without a proof of this fact we need a different approach for large p.

We consider the 2-step estimator of our previous work [5], which generates $H \sim G(q)$ for $q = n^{-2/c}$ and estimates $u_G(p)$ as $u_H(p/q)$. While that paper foregrounded a relative variance of $\tilde{O}(n^2)$ for this estimator, it actually proved something stronger. As discussed there in Section II.C, the 2-step estimator is fast for any q such that $G(q)$ has $O(\log n)$ vertices with high probability, and its relative variance is

$$O(\frac{p_c \log^2 n}{q^c u_G(p)}).$$

That work observed that $G(n^{-2/c})$ has $O(\log n)$ vertices with high probability, and that $u_G(p) \ge p^c$, and the $\tilde{O}(n^2)$ variance followed. But that paper also observed that in certain graphs—such as two cliques connected by c edges—$G(q)$ has $O(\log n)$ edges even for a much larger value of q, which yields a faster runtime. In this section, we define an *effective size* that captures this phenomenon. We then show that if $u_G(p)$ is particularly small (the case which seems to make the above bound large) then the graph has small effective size and can use a larger q (which keeps the above bound small).

A. Effective Size

Definition VI.1. Consider a graph G which can have m_r vertices when contracting to size r. Define the *effective size* of G to be

$$\exp\left(\max \frac{c}{2} \sum_{r=2}^{n} \frac{1}{m_r}\right).$$

where the maximum is over all possible sequences of m_r. In particular, we can take m_r to be the minimum r-way cut in G, as this always lower bounds the number of edges.

Note that if G has min-cut c then each component of any r-way cut must have degree c, meaning $m_r \ge rc/2$. Thus, the effective size of G is $\exp(\sum 1/r) \le 2n$.

Lemma VI.2. *If G has effective size N and $p^c \le N^{-2}$ then $z_G(N^{-2/c}) \le 1$.*

Proof: This is simply a strengthened restatement of Partition Function Corollary III.4. Note that Contraction Lemma III.2 gives its bound in terms of the effective size we have just defined, which means that Cut Survival Corollary III.3 and thus Partition Function Corollary III.4 both inherit it. ∎

Lemma VI.3. *In a graph of effective size N, the sample $G(N^{-2/c})$ has $O(\log n)$ vertices with high probability.*

Corollary VI.4. *In a graph of effective size N, taking $q = N^{-2/c}$, sampling $H \sim G(q)$ and estimating $u_G(p) = u_H(p/q)$ yields an unbiased estimator of relative variance $\tilde{O}(N^2(p^c/z_G(p))) \le N^2$ for any $p \le q$.*

Proof: $z_G(N^{-2/c})$ is the expected number of cuts in $G(N^{-2/c})$ and is at most 1 by the previous lemma. It follows by the Markov inequality that $G(N^{-2/c})$ has at most $n^{O(1)}$ cuts with high probability. But a graph with r vertices has 2^{r-1} cuts, which proves the bound.

The corollary was proven in our previous work [5, Section II.C]. ∎

Remark. This last proof is quite similar to the standard proof of the Chernoff bound, applying the Markov inequality to an exponential function of the random variable we wish to bound.

B. The Effective Size of Reliable Graphs

So let us write $z = z_G(q)$ and suppose $z \le n^{-\lambda}$. Consider an r-way cut of G with m_r edges and let d_i denote the degrees of the components so that $\sum d_i = 2m_r$. It follows that $z \ge \sum q^{d_i}$. Convexity shows that for fixed $\sum d_i$, the quantity $\sum q^{d_i}$ is minimized when all d_i are equal. Inverting this result, if we fix $\sum q^{d_i}$ then $\sum d_i = 2m_r$ is maximized when all d_i are equal $2m_r/r$. In other words, we have shown that $z \ge \sum q^{d_i} \ge rq^{2m_r/r}$. Inverting, we find that

$$\begin{aligned} rq^{2m_r/r} &\le z \\ (2m_r/r)\log q &\le \log(z/r) \\ m_r &\ge \frac{r}{2}\log_q(z/r) \qquad \text{since } q \le 1 \end{aligned}$$

We already have a bound of $m_r \ge rc/2$, but this new bound is stronger one whenever

$$\begin{aligned} \frac{r}{2}\log_q(z/r) &\ge rc/2 \\ \log_q(z/r) &\ge c \\ z/r &\le q^c \\ r &\ge zq^{-c} \end{aligned}$$

In particular, this bound is never stronger than the naive one if $zq^{-c} \ge n$ since then no relevant r (in the range $1, \ldots, n$) satisfies it. On the other hand if $zq^{-c} \le n$ then

it applies for r near n and thus

$$\begin{aligned}\sum \frac{c}{2m_r} &\le \sum_{r=2}^{zq^{-c}} \frac{1}{r} + \sum_{r=zq^{-c}}^{n} \frac{c}{r \log_q z/r} \\ &= H_{zq^{-c}} + c(\ln 1/q) \int_{zq^{-c}}^{n} \frac{1}{r \ln r/z}\, dr \\ &= \ln zq^{-c} + c(\ln 1/q) \int_{q^{-c}}^{n/z} \frac{1}{s \ln s}\, ds \\ &= \ln zq^{-c} + (c \ln 1/q) \ln\ln s|_{q^{-c}}^{n/z} \\ &= \ln zq^{-c} + (\ln q^{-c}) \ln \frac{\ln n/z}{\ln q^{-c}}\end{aligned}$$

In particular, if we take $q = n^{-2/c}$ and write $z_G(q) = n^{-\lambda}$, we find that the effective size

$$N \le n^{2-\lambda+2\ln\frac{1+\lambda}{2}}.$$

which is 1 for $\lambda = 1$ and shrinks to $n^{.81}$ as $\lambda \to 2$. This is the largest possible λ since we know $z_G(p) \ge p^c$. A pair of cliques connected by c edges would have $\lambda \approx 2$.

Note also that the term zq^{-c} can intuitively be seen as estimating the number of minimum cuts in G, since if there k minimum cuts they contribute kq^c to the value of z. This indicates that the effective says grows with the number of small cuts.

C. Application

We've now shown that for $q = n^{2/c}$ a graph with $z_G(q)$ close to n^{-2} has effective size significantly less than n. We leverage this as follows. Recall that when $p^c \le n^{-2}$ we apply the RCA while above we used NMC. Recall also that at the transition $q = n^{-2/c}$, we showed that $u_G(q) = \Theta(z_G(q))$. We'll choose an appropriate threshold $\lambda < 2$. If $z_G(q) \ge n^{-\lambda}$ then a fortiori for any $p \ge q$ we have $u_G(p) \ge u_G(q) = \Omega(n^{-\lambda})$, which means that NMC estimation (on a sparsified G) takes time $\tilde{O}(n^{1+\lambda})$ whenever we use it.

If on the other hand $z_G(q) \le n^{-\lambda}$, then our analysis above shows that G has effective size $O(n^\gamma)$ for $\gamma = 2 - \lambda + 2\ln((1+\lambda)/2) < 1$. This means that by Corollary VI.4 we can run the two-step estimator with parameter $q = n^{-2\gamma/c}$ instead of $n^{-2/c}$, which improves its relative variance to $\tilde{O}(n^{2\gamma})$ and its runtime to $\tilde{O}(n^{1+2\gamma})$.

In this case we only need to use NMC if $p^c \ge n^{-2\gamma}$, which means it too will take time $\tilde{O}(n^{1+2\gamma})$ on the sparsified G.

To recap, we switch algorithms based on the value $z_G(n^{-2/c})$. If $z_G(n^{-2/c}) \ge n^{-\lambda}$, then we apply our original algorithm, using NMC for $p \ge n^{-2/c}$ and the RCA for smaller p. The dominant time bound is NMC estimation, but this takes time $\tilde{O}(n/u_G(p)) = \tilde{O}(n/u_G(q)) = \tilde{O}(n/z_G(q)) = \tilde{O}(n^{1+\lambda})$. On the other hand, if $z_G(q) \le n^{-\lambda}$, then we can use our 2-step estimator algorithm whenever $p^c \le n^{-2\gamma}$ (and not bother with the RCA). In this case we only run NMC when $p^c \ge n^{-2\gamma}$, so the bottleneck NMC takes at worst $\tilde{O}(n^{1+2\gamma})$ time.

We balance these two runtimes by setting $\lambda = 2\gamma$. Recalling that $\gamma = 2 - \lambda + 2\ln((1+\lambda)/2)$, we solve

$$\lambda = 2(2 - \lambda + 2\ln((1+\lambda)/2))$$

Which gives $\lambda \approx 1.78$ and a runtime of $\tilde{O}(n^{2.78})$.

Obviously this is only a small improvement over $O(n^3)$. It's importance is not the amount of improvement, but rather the demonstration that it is possible to improve at all, moving beyond the n^3 barrier that appeared to be limiting improvement in the previous approximation algorithms.

Remark. We must be a bit careful if we want our algorithm above to remain an unbiased estimator. Normally, we might imagine running NMC for n^λ steps and then returning a value (if we have determined it is large) or switching to the 2-step estimator if we have determined we need to. But this introduces a bias in the estimator, since it will *only* return larger values from the NMC step, while an unconditioned NMC algorithm has a chance to return a small value. So instead, we do a little bit of extra work. *First*, we use NMC to decide which case we are in. Based on what we find, we *either* run NMC *again* to get an unbiased estimator, or else run the 2-step algorithm if we need it.

Of course, there a small chance that our initial NMC test returns an incorrect result. But we get the correct outcome with high probability, and our estimator has polynomial relative variance even if we are wrong, so the small chance of error does not significantly affect the relative variance of our combined estimator.

VII. Conclusion

We've given a new-old algorithm for network reliability by showing that the Recursive Contraction Algorithm can compute a low-variance unbiased estimator. The key to our work are new analytic techniques that prove that the relative variance of a natural estimator $u_H(p/q)$ is $O(q^{-c} - 1)$. Along the way, we showed that $u_G(p) = \Theta(z_G(p))$, which may be useful for a variety of other network reliability applications.

This approach works so long as $p^c \le n^{-2}$, which is the threshold at which at least some graphs (in particular

the cycle) begin to become unreliable. For larger p, we developed a concept of *effective size* and use it to improve our use of naive Monte Carlo sampling; this allows to provide a small improvement in the runtime of the naive method which we use for larger p. Taken together, our results break the cubic barrier for network reliability and yield an $\tilde{O}(n^{2.78})$ runtime.

We've only shown a limited application of the effective size concept; we can do more with it. For example, it is possible to apply the relative variance ideas to the recursive estimator, improving the range of p over which it can work and further improving our running time. Details will be given in the full paper.

Our work began with a conjecture, that the relative variance of $u_H(p/q)$ is at worst $q^{-c} - 1$. No better bound is possible as this is the answer for two cliques connected by c-edges. But it seems plausible that this is the worst-case graph. For having a different graph with more small cuts should only *smooth* the behavior of the sampling experiment and reduce the relative variance. For example, the relative variance of the cycle with its many min-cuts is constant.

If our conjecture is true, then the Recursive Contraction Algorithm serves as an estimator for *all* values of p with a running time of $\tilde{O}(n^2)$.

REFERENCES

[1] L. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, pp. 410–421, 1979.

[2] J. S. Provan and M. O. Ball, "The complexity of counting cuts and of computing the probability that a network remains connected," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 777–788, 1983.

[3] D. R. Karger, "A randomized fully polynomial approximation scheme for the all terminal network reliability problem," *SIAM Journal on Computing*, vol. 29, no. 2, pp. 492–514, 1999, a preliminary version appeared in Proceedings of the 27^{th} ACM Symposium on Theory of Computing. A corrected version was published in SIAM Review 43(3).

[4] D. G. Harris and A. Srinivasan, "Improved bounds and algorithms for graph cuts and network reliability," in *Proceedings of the 25^{th} Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM-SIAM. ACM Press, Jan. 2014.

[5] D. R. Karger, "A fast and simple unbiased estimator for network (un)reliability," in *Proceedings of the 48th annual IEEE Symposium on Foundations of Computer Science*, Oct. 2016.

[6] D. R. Karger and C. Stein, "A new approach to the minimum cut problem," *Journal of the ACM*, vol. 43, no. 4, pp. 601–640, Jul. 1996, preliminary portions appeared in SODA 1992 and STOC 1993.

[7] D. R. Karger, "A randomized fully polynomial approximation scheme for the all terminal network reliability problem," *SIAM Review*, vol. 43, no. 3, pp. 499–522, 2001, a preliminary version appeared in Proceedings of the 27^{th} ACM Symposium on Theory of Computing. This corrects a version published in SICOMP.

[8] R. M. Karp, M. Luby, and N. Madras, "Monte Carlo approximation algorithms for enumeration problems," *Journal of Algorithms*, vol. 10, no. 3, pp. 429–448, Sep. 1989.

[9] D. R. Karger, "A fast and simple unbiased estimator for network (un)reliability," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, Oct 2016, pp. 635–644.

[10] D. R. Karger and C. Stein, "An $\tilde{O}(n^2)$ algorithm for minimum cuts," in *Proceedings of the 25^{th} ACM Symposium on Theory of Computing*, A. Aggarwal, Ed., ACM. ACM Press, May 1993, pp. 757–765, journal version appears in Journal of the ACM 43(4).

[11] D. R. Karger, "Global min-cuts in $\mathcal{RNC}$ and other ramifications of a simple mincut algorithm," in *Proceedings of the 4^{th} Annual ACM-SIAM Symposium on Discrete Algorithms*. ACM-SIAM, Jan. 1993, pp. 21–30, this work was merged with later work into Journal of the ACM43(4).