IMPROVED BOUNDS AND ALGORITHMS FOR GRAPH CUTS AND NETWORK RELIABILITY[‡]

DAVID G. HARRIS[1] AND ARAVIND SRINIVASAN[2]

Abstract. Karger (*SIAM Journal on Computing*, 1999) developed the first fully-polynomial approximation scheme to estimate the probability that a graph G becomes disconnected, given that its edges are removed independently with probability p. This algorithm runs in $O(n^{5+o(1)}\epsilon^{-3})$ time to obtain an estimate within relative error ϵ.

We improve this runtime in two key ways, one algorithmic and one graph-theoretic. From an algorithmic point of view, there is a certain key sub-problem encountered by Karger, for which a generic estimation procedure is employed. We show that this sub-problem has a special structure for which a much more efficient algorithm can be used. From a graph-theoretic point of view, we show better bounds on the number of edge cuts which are likely to fail. Karger's analysis depends on bounds for various graph parameters; we show that these bounds cannot be simultaneously tight. We describe a new graph parameter, which simultaneously influences all the bounds used by Karger, and use it to obtain much tighter estimates of the behavior of the cuts of G. These techniques allow us to improve the runtime to $O(n^{3+o(1)}\epsilon^{-2})$; our results also rigorously prove certain experimental observations of Karger & Tai (*Proc. ACM-SIAM Symposium on Discrete Algorithms*, 1997). Our proofs, while rigorous, are motivated by certain non-rigorous differential-equation approximations which, however, track the worst-case trajectories of the relevant parameters.

A key driver of Karger's approach (and other cut-related results) is his earlier bound on the number of small cuts: we also show how to improve this when the min-cut size is "small" and odd, augmenting, in part, a result of Bixby (*Bulletin of the AMS*, 1974).

1. Introduction.

Let G be a connected undirected multi-graph with vertex set V; as usual, we let $|V| = n$. Unless stated otherwise, the graphs we deal with will be *multi-graphs with no self-loops*, presented in *adjacency-matrix format*. We define $R(p)$, the reliability polynomial of G, to be the probability that the graph remains connected when edges are removed independently with probability p. One can verify that this is a polynomial in p. This polynomial has various physical applications, for example determining the reliability of a computer network or power grid. Although there is no currently known algorithm for estimating $R(p)$, the complementary probability $U(p) = 1 - R(p)$, which we call the *unreliability* of the graph, can be estimated in polynomial time. In a breakthrough paper ([7], see also [8]), Karger developed the first fully-polynomial randomized approximation scheme (FPRAS) to estimate $U(p)$ up to relative error ϵ in time polynomial in n and $1/\epsilon$; even a constant-factor approximation was not known prior to his work.

We let $\exp(x)$ denote e^x; all logarithms are to **base e** unless indicated otherwise. With an appropriate choice of parameters, Karger's algorithm can run in time $\exp(3\log n + \sqrt{\log n}\sqrt{4\log n + \sqrt{2}\log(1/\epsilon) + o(\log n)})$. In particular, if $\epsilon = \Theta(1)$, this is $O(n^{5+o(1)})$. (We assume that $\epsilon \leq O(1)$, say $\epsilon \leq 1/2$, throughout.) Karger's algorithm is based on an algorithm for finding graphs cuts which have close to minimal weight. This algorithm is the *Contraction Algorithm*, first introduced by [6]. This algorithm has emerged as an important building block for many graph algorithms.

[1]Department of Applied Mathematics, University of Maryland, College Park, MD 20742. Research supported in part by NSF Award CNS-1010789. Email: davidgharris29@hotmail.com.

[2]Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported in part by NSF Award CNS-1010789. Email: srin@cs.umd.edu.

[‡]A preliminary version of this work appeared in the *Proc. ACM-SIAM Symposium on Discrete Algorithms* (SODA), 2014.

Just as importantly, it can be viewed as a stochastic process which provides structural results about the graph cuts, for example, a bound on the number of small cuts.

In this paper, we provide a more detailed analysis of the Contraction Algorithm and its consequences. This enables us to show a variety of improved bounds and algorithms for graph problems. The focus of this paper is on improving the algorithm for estimating $U(p)$.

The following definition will be useful:

DEFINITION 1.1. *The minimum cut-size in G, also known as the edge-connectivity of G, will be denoted by c. Given $\alpha \geq 1$, an "α-cut" in G is cut with at most αc edges.*

Our results.

The main focus of our analysis is a much faster algorithm for estimate $U(p)$:

THEOREM 1.2. *There is an algorithm to estimate $U(p)$ in time $n^{3+o(1)}\epsilon^{-2}$.*

Our analysis of the Contraction Algorithm will also allow purely structural bounds on the cut structure of certain graphs:

THEOREM 1.3. *Suppose c is an odd constant and $\alpha \geq 1$. Then there are at most $O(n^{\alpha \frac{2c}{c+1}})$ α-cuts. This exponent is optimal, in the sense that one cannot show a bound of $O(n^{\alpha x})$ for any $x < \frac{2c}{c+1}$. (The constant term in the asymptotic notation may depend on c).*

In [10], an algorithm called the *Recursive Contraction Algorithm* was developed for finding all α-cuts. We obtain a small improvement to this algorithm for the general problem of finding α-cuts. Unlike the previous improvements, this only reduces the logarithmic terms:

THEOREM 1.4. *There is an algorithm that accepts as input a graph G and a real number $\alpha \geq 1$. This algorithm succintly enumerates, with high probability, all α-cuts of G in time $O(n^{2\alpha} \log n)$.* By contrast, the algorithm of [10] requires time $O(n^{2\alpha} \log^2 n)$.

1.1. Overview of Karger's algorithm, and our improvements. Karger's algorithm for estimating $U(p)$ essentially consists of two separate algorithms, depending on the size of $U(p)$. When $U(p) > n^{-2-\delta}$, where δ is some constant, then Monte Carlo sampling provides an accurate estimate. As the samples are unbiased with relative variance $1/U(p)$, then after $n^{2+\delta}\epsilon^{-2}$ samples we estimate $U(p)$ accurately. Naively, it might appear to require n^2 time to compute a sample (the cost of reading in the adjacency matrix), but Karger describes a method of sparsifying the graph to reduce this to $n^{1+o(1)}$. We will not modify Karger's algorithm for Monte Carlo sampling.

When $U(p)$ is small, then Monte Carlo sampling no longer can produce an accurate estimate. In the regime $U(p) < n^{-2-\delta}$, Karger develops an alternative algorithm. In this case, Karger shows that the event of graph failure is dominated by the event that a *small* cut of G has failed. This is done by analyzing the number of cuts of small weight. In particular, there is some α^* such that $U(p)$ is closely approximated by the probability that an α^*-cut has failed. Karger provides an upper bound on the critical value α^*; for example, when $\epsilon = \Theta(1)$, we have $\alpha^* \leq 1 + 2/\delta = O(1)$.

This significantly simplifies the problem, because instead of having to consider the exponentially large collection of all cuts of G, we can analyze only the polynomial-size collection of α^*-cuts. Using an algorithm developed in [10], the *Recursive Contraction Algorithm*, Karger's algorithm can catalogue all such α^*-cuts in time $n^{2\alpha^*+o(1)}$. We refer to this as the *cut-enumeration* phase of Karger's algorithm. Note that there is a tradeoff between the Monte Carlo phase and cut-enumeration phase, depending on the size of δ.

Having catalogued all such small cuts, we need to piece them together to estimate the probability that one of them has failed. For this, Karger uses as a subroutine a statistical procedure developed by Karp, Luby, Madras [12]. This procedure can examine any collection of partially overlapping clauses, to provide an unbiased estimate with relative error $O(1)$ that one such clause is satisfied. By running ϵ^{-2} iterations of this, we achieve the desired error bounds for $U(p)$. The running time is linear in the size of the collection of clauses. We refer to this as *statistical sampling* phase of Karger's algorithm.

We will improve both the cut-enumeration phase and statistical sampling phase of Karger's algorithm. To improve the cut-enumeration phase, we will show a tighter bound on α^*. In [11], a version of Karger's algorithm was programmed and tested on real graphs. This found experimentally that the average number of cuts needed to accurately estimate the graph failure probability was about n^3, not the worst case n^5 as predicted by [7]. In this paper, we show this fact rigorously. We will show that the number of small graph cuts in reliable graphs is far smaller than in general graphs. This is the most technical part of our paper; we briefly describe our methodology for proving this.

The basic idea is to analyze the dynamics of the Contraction Algorithm, and to show that any small cut C has a large probability of being selected by this algorithm. The Contraction Algorithm is a complicated stochastic process, and the graph G can change in difficult and unpredictable ways through its evolution. We approximate the Contraction Algorithm by a continuous, deterministic dynamical system. This gives rise to a system of differential equations, which can be solved in closed form. Even though our approximation was basically heuristic, it turns out that the *solution* to the differential system is correct, and can be proved accurate rigorously. It turns out that the heuristic approximations represent the worst-case behavior for the Contraction Algorithm. Furthermore, we prove this by induction, which depends on knowing the answer in advance — which we would not have been able to do without our heuristic approximation!

This analysis is quite difficult technically, because it requires analyzing the three-way interconnections between the number of α-cuts of G, the effectiveness of the Contraction Algorithm, and the value of $U(p)$. The critical parameter that ties these three together is the *expected number of failed graph cuts.*

Our improvement to the statistical sampling phase is more straightforward. The algorithm of [12] is able to handle very general collections of clauses, which can overlap in complicated ways. However, this is unnecessary for our purposes. We will show that the collection of cuts produced by the cut-enumeration phase has a simple distribution, both statistically and computationally. This allows us to use a faster and simpler statistical procedure to piece together the α^*-cuts. As a result, the running time of each sample reduces to about $O(n^2)$; in particular, we do not need to read the entire collection of α^*-cuts for each sample.

In total, we reduce the running time of Karger's algorithm to about $n^{3+o(1)}\epsilon^{-2}$. Obtaining faster run-times with Karger's recipe appears to be quite difficult; for if c is the size of the min-cut, then in the regime where $p^c \geq n^{-2+\Omega(1)}$, it is not clear how to stop only at "small" cuts, and one appears to need Monte-Carlo sampling — which requires a runtime that matches ours.

We note that another approach to estimating $U(p)$ has been discussed in [15]. This algorithm finds a pair of minimal cuts which are mostly disjoint, and hence there is a negligible probability that both cuts fail simultaneously. By continuing this process, it is found in [15] experimentally that a branching process can efficiently enumerate the probability that some small cut fails. While this algorithm is promising, the analysis of [15] is fully experimental, and it appears that in the worst case this approach might require super-polynomial time.

Obtaining an FPRAS – or even a constant-factor approximation – for $R(p)$ remains a very intriguing open problem.

1.2. Outline. In Sections 2 and 3, we begin by reviewing some key results and definitions concerning graph cuts. Most of these results are recapitulations of [7].

In Section 4, we introduce the Contraction Algorithm, an algorithm described by [10]. This algorithm provides an efficient randomized procedure for finding small-weight cuts in G. It also can be viewed as a probabilistic process which can used to estimate the number of such cuts, thus providing purely structural bounds on G. We will be interested in the following question, which is more general than that considered by [10]: suppose we are given a fixed target cut C of G. Under what circumstances does the Contraction Algorithm select C? What can we say about the dynamics of the Contraction Algorithm *in those cases in which C will ultimately be selected*? We will prove a series of technical Lemmata which describe these dynamics. Roughly speaking, these show that the Contraction Algorithm has a *uniform behavior* regardless of which target cut C (if any) we are interested in.

In Section 5 we apply this machinery to analyze the Contraction Algorithm in the case in which the (unweighted) graph G has c an odd constant. These results are not necessary for our analysis of Karger's algorithm. We include them here for two reasons. First, the number of small cuts is basically known in the case of even c — the worst case behavior comes from a cycle graph, in which there are n vertices in a ring with $c/2$ edges between successive vertices. Further bounds in this case would require adding extra conditions, say on the *minimality* of these cuts; this appears to be quite difficult. The case of odd c has been mostly overlooked in the literature.

Second, the case of small odd c provides an easier warm-up exercise for the analysis in Section 6. In Section 5, we keep track of the number of minimum cuts remaining in G during the evolution of the Contraction Algorithm; in Section 6 we must keep track of the number of cuts of all sizes. Section 6 analyzes how the dynamics of the Contraction Algorithm are affected by the magnitude of $U(p)$. The critical parameter is $\bar{Z}$, the *expected number of failed cuts* in G. We show how $\bar{Z}$ affects the number of edges which are available in any round of the Contraction Algorithm, and we show how $\bar{Z}$ itself changes during the execution.

The proof method used in Sections 5, 6 is worth noting. We give a function f, and prove by induction that the probability of a cut surviving the Contraction Algorithm is at least f. This function f is very complicated and could not have been guessed from first principles. Instead, we first simplify our stochastic process using some unwarranted independence and monotonicity assumptions, and then translate the stochastic time-steps into a differential equation which can be solved in closed-form. This heuristic analysis gives us the mysterious function f, which we then rigorously prove correct. In particular, these "unwarranted independence and monotonicity assumptions" actually correspond to the worst-case behavior of our stochastic process!

In Section 7, we apply the analysis of Section 6 to show that α^*, which is the bound of the size of the cuts necessary to approximate $U(p)$, is smaller than the bound given by Karger. We note that while Karger gave a simple bound which could be computed explicitly, our bound depends on the graph parameter $\bar{Z}$ which we cannot determine easily. However, using this formula, one can determine an upper bound on the total number of iterations of the Contraction Algorithm that are needed to find the α^*-cuts; this bound is irrespective of $\bar{Z}$.

In Section 8, we analyze the Recursive Contraction Algorithm. This is an algorithm which allows us to run the Contraction Algorithm "in bulk". If we are interested in finding many graph cuts, we must run the Contraction Algorithms multiple times. However, most of the work for processing the graph can be amortized across these multiple runs. In particular, we can essentially reduce the running time for the Contraction Algorithm from n^2 (dominated by reading the original graph) to n^a where a is a small constant.

In Section 9, we describe a new statistical sampling algorithm to analyze the resulting collection of α^*-cuts. We show that this algorithm has a faster running time, and is just as accurate, as the algorithm of [12] used in [7, 8]. In Section 10, we describe how to combine all the pieces and obtain a full algorithm. This includes deciding when to use Monte Carlo sampling and when to enumerate the small cuts, a detail omitted in [7]. Finally, Section 11 concludes.

2. Preliminaries. For a multi-graph G, we define a *cut* of G to be a partition of the vertices into two classes $V = A \sqcup A'$, with $A, A' \neq \emptyset$, such that the removal of all edges crossing from A to A' disconnects the graph. We must distinguish this from an *edge-cut*, which is a subset E' of the edges of G such that removal of E' disconnects G. Observe that every cut of G induces an edge-cut of G. The *weight* or *size* of an edge-cut is the number of edges it contains. We say an edge-cut is *minimum* if it has the smallest size of all edge-cuts, and we denote by c the size of the smallest edge-cut. We say an edge-cut is *minimal* if no proper subset is an edge-cut.

We note that any cut corresponds to a unique edge-cut. Although not every edge-cut corresponds to a cut, we can observe that any minimal edge-cut corresponds to a unique cut. We say that a cut is *minimal* iff the corresponding edge-cut is minimal. Note that the graph may contain up to 2^{n-1} cuts and up to 2^m edge-cuts. For the most part, we will only consider minimal cuts in this paper. In this case, the distinction between a cut and an edge-cut disappears; every minimal cut corresponds to a unique minimal edge-cut, and vice-versa. We will often abuse notation so that a cut C may refer to either the vertex-partition *or* the edge-cut it induces. Unless stated otherwise, whenever we refer to a cut C, we view C as a set of *edges*.

As mentioned before, for any $\alpha \geq 1$ we define an α-cut to be a cut whose corresponding edge-cut has at most αc edges.

In this paper, we seek to estimate the probability that graph becomes disconnected when edges are removed independently with probability p. When edges are removed in this way, we say that an edge-cut *fails* iff all the corresponding edges are removed. In this case, we may concern ourselves solely with cuts. The reason for this is that graph G becomes disconnected iff some minimal edge-cut of G fails (which in turn happens iff some cut of G fails). As there are far fewer cuts than edge-cuts, we will mostly be concerned with cuts.

This method can be generalized to allow each edge to have its own independent failure probability p_e. As described in [7], given a graph G with non-uniform edge failure probabilities, one can transform this to a multi-graph G' with uniform edge failure probabilities by replacing each edge of G with a bundle of edges, and setting p appropriately. This may cause a large increase in the number of edges in the graph and a large increase in the size of the minimum cut c. Because this transformation can greatly increase the number of edges m, we will describe the running time of our algorithms as a function of n alone.

We will assume that the graph G is presented as an adjacency matrix with n^2 words. Each word records the number of edges that are present between the indicated vertices. As described in [6], it is possible to transform an arbitrary graph, in which the number of edges may be unbounded, into one with cells of size $O(\log(1/\epsilon)+\log n)$ with similar reliability; hence the arithmetic operations will take polylogarithmic time in any computational model. This transformation may have a running time which is super-polynomial in n, although it is close to linear time as a function of the input data size. We will ignore these issues, and simply assume that we can perform arithmetic operations to precision ϵ and random number generation of an entire word in a single time step. In most cases, the arithmetic operations can be done in lower precision, at least for the inner-most loops, but we will not consider these issues.

As our algorithm closely parallels Karger's, we use the notation of [7] wherever possible. Recall that c is the size of the smallest edge-cut. As in [7], we let $p^c = n^{-2-\delta}$, with $\delta > 0$ being the case of primary interest (the complementary case can be handled by simple Monte-Carlo sampling). We note that δ can be determined in time $n^{2+o(1)}$, simply by finding the size of the minimum cut. Hence we can assume δ is known. Often we will derive estimates assuming $\delta \geq \delta_0$ for some constant $\delta_0 > 0$. This assumption allows us to simplify many of the asymptotic notations, whose constant terms may depend on δ_0. In fact, our algorithm is best balanced when $\delta \to 0$. It is not hard to see that all of our asymptotic bounds when $\delta \geq \delta_0$ can be relaxed to allow $\delta \to 0$ sufficiently slowly, for example as $\delta = 1/\log\log\log\log n$.

We will be very interested in α-cuts, where normally α should be thought of as essentially constant. (In fact, the case where $\alpha = O(1)$ would be basically sufficient to analyze all of our algorithmic improvements.) We will show how to bound the number of such small cuts, and we will show that overlap among these cuts is also controlled. The following is a well-known theorem of [5], and indeed, the rest of this section is basically a recapitulation of Karger's work from [5].

Theorem 2.1. *([5]) The number of α-cuts is at most $n^{2\alpha}$.*

The following combinatorial principle will be used in a variety of contexts.

Lemma 2.2. *Let $F : \mathbf{R} \to \mathbf{R}$ be any increasing function with the property that, for any $\alpha \geq 1$, the number of α-cuts is at most $F(\alpha)$. Let $g : \mathbf{R} \to [0,\infty)$ be any continuous decreasing real-valued function. Then we have*

$$\sum_{\substack{\text{Cuts } C \text{ of weight} \\ |C| \geq \alpha c}} g(|C|/c) \leq F(\alpha) g(\alpha) + \int_{x=\alpha}^{\infty} g(x)\, dF(x)$$

Proof. This is simply integration by parts. □

Combining Lemma 2.2 and Theorem 2.1, gives a simple proof of the following result from [7]:

Corollary 2.3. *Let $0 < q < n^{-2}$, $\alpha \geq 1$. Then we have*

$$\sum_{\substack{\text{Cuts } C \text{ of weight} \\ |C| \geq \alpha c}} q^{|C|/c} \leq (n^2 q)^{\alpha} \frac{1}{1 + \frac{\log n^2}{\log q}}$$

In particular, if $q \leq n^{-2-\Omega(1)}$, then we have

$$\sum_{\substack{\text{Cuts } C \text{ of weight} \\ |C| \geq \alpha c}} q^{|C|/c} = O((n^2 q)^{\alpha})$$

Proof. Apply Lemma 2.2, using $F(x) = n^{2x}$ and $g(x) = q^x$. We have

$$\sum_{\substack{\text{Cuts } C \text{ of weight} \\ |C| \geq \alpha c}} q^{|C|/c} \leq q^\alpha n^{2\alpha} + \int_{x=\alpha}^{\infty} q^x \times 2n^{2x} \log n dx = (n^2 q)^\alpha \frac{1}{1 + \frac{\log n^2}{\log q}}$$

□

As an example of how this principle may be used we have the following result. This will later be strengthened in Proposition 7.1.

COROLLARY 2.4. *([7]) Suppose $\delta \geq \delta_0$ for some constant $\delta_0 > 0$. Then the probability that a cut of weight $\geq \alpha c$ fails is at most $O(n^{-\alpha\delta})$.*

Proof. By the union-bound, the probability that a cut of weight $\geq \alpha c$ fails is at most

$$\mathbf{P}(\text{Cut of weight} > \alpha c \text{ fails}) \leq \sum_{\substack{\text{Cuts } C \text{ of weight} \\ |C| \geq \alpha c}} (p^c)^{|C|/c}$$

Now apply Corollary 2.3. □

This leads to one of the key theorems of Karger's original work:

THEOREM 2.5. *([7]) Suppose $\delta \geq \delta_0$ for some constant $\delta_0 > 0$. Then $U(p)$ can be approximated, up to relative error $O(\epsilon)$, by the probability that a cut of weight $\leq \alpha^* c$ fails, where*

$$\alpha^* = 1 + 2/\delta - \frac{\log \epsilon}{\delta \log n}$$

Proof. The *absolute error* committed by ignoring cuts of weight $\alpha^* c$ is the probability that such a cut fails; by Corollary 2.4 it is at most $O(n^{-\alpha^* \delta})$.

The minimum cut of G fails with probability $p^c = n^{-2-\delta}$, so we have $U(p) \geq p^c = n^{-2-\delta}$.

Hence the *relative error* committed by ignoring cuts of weight $\geq \alpha^* c$ is at most

$$\text{Rel err} = O(\frac{n^{-\alpha^* \delta}}{n^{-2-\delta}}) = O(\epsilon)$$

□

We do not want to get ahead of ourselves, but one of the main goals of this paper will be to improve on the estimate of Theorem 2.5. In proving Theorem 2.5, we used two quite different bounds. These bounds are tight separately, but for very different kinds of graphs. In estimating the absolute error, we bound the number of α-cuts that may appear in G by $O(n^{2\alpha})$. This is tight for cycle graphs, as discussed in Lemma 3.1, in which there are a very large number of cuts. On the other hand, when we estimate $U(p) \geq p^c$, we are assuming that the graph has just a single small cut. As we will show, no graph can have both bounds be simultaneously tight.

3. The total number of failed cuts.

We define the random variable Z to be the number of cuts of G which fail when edges are removed independently with probability p. The expectation of Z is

$$\bar{Z} = \sum_{\text{cuts } C} p^{|C|}.$$

This is an overestimate of the graph failure probability which ignores the overlap between the cuts. The quantity $\bar{Z}$ will play a crucial role in our estimates. We begin by recalling some results of [7] which bound the random variable Z:

LEMMA 3.1. *Let $p^c = n^{-2-\delta}$ for $\delta \geq 0$. Suppose we remove edges from G with probability p, and let H denote the resulting graph. Then*

$$\mathbf{P}(H \textit{ has } \geq r \textit{ connected components}) \leq e^{-1}(e/r)^r n^{-r\delta/2} \leq n^{-r\delta/2}$$

Proof. As shown in [13], [14], [7], the probability that H has $\geq r$ connected components under edge-failure probability p, is at most the probability that the graph C has $\geq r$ connected components under edge failure $q = p^{c/2}$, where C is the cycle graph which contains n vertices and an edge between successive vertices in the cycle. If r edge bundles fail in C, then the resulting number of distinct components is $\max(1, r)$. Hence the probability that C results in r distinct components is at most the probability a binomial random variable, with n trials and success probability q, is at least r.

We can use the Chernoff bound to estimate the probability of such an extreme deviation. In this case, the mean of the binomial is $\mu = nq = n^{-\delta/2}$ and the relative deviation is $D \geq rn^{\delta/2} - 1$. By the Chernoff bound,

$$\begin{aligned}\mathbf{P}(\geq r \text{ connected components}) &\leq (\frac{e^D}{(1+D)^{1+D}})^\mu \\ &\leq e^{-1}(e/r)^r n^{-r\delta/2} \leq n^{-r\delta/2}\end{aligned}$$

□

LEMMA 3.2. *Suppose $\delta > 0$. Suppose we remove edges from G with probability p, then*

$$\mathbf{P}(Z \geq t) \leq e^{-1}\Big(\frac{e}{1+\log_2 t}\Big)^{1+\log_2 t} n^{-(1+\log_2 t)\delta/2} \leq n^{-\delta \log_2 t/2}$$

Proof. Let H be the graph obtained after edges fail with probability p. Let r be the number of connected components of H, and let G' be the graph obtained from G by contracting all components of H. So G' has r vertices. Note that any cut of G' is a cut of G as well. Hence the minimal cut of G' has size $\geq c$, so G' has at least $rc/2$ edges.

Suppose $Z \geq t$, so there are t distinct cuts in G. Hence $t \leq 2^{r-1}$, that is, the graph H has $r \geq 1 + \log_2 t$ connected components. By Lemma 3.1, the probability of this event is at most $e^{-1}(\frac{e}{1+\log_2 t})^{1+\log_2 t} n^{-\delta(1+\log_2 t)/2}$. □

The quantity $\bar{Z}$ is much more tractable than $U(p)$, but they are nearly equivalent asymptotically:

PROPOSITION 3.3. *For $\delta \geq \delta_0 > 0$, we have $U(p) = \Theta(\bar{Z})$* .

Proof.

By the union bound, $U(p) \leq \bar{Z}$.

Now note $U(p) = \mathbf{P}(Z \geq 1) = \frac{\mathbf{E}[Z]}{\mathbf{E}[Z \mid Z \geq 1]}$. So it suffices to show that $\mathbf{E}[Z \mid Z \geq 1] = O(1)$.

By Lemma 3.2, we have

$$\begin{aligned}\mathbf{E}[Z \mid Z \geq 1] &= \sum_x \mathbf{P}(Z \geq x \mid Z \geq 1) \\ &\leq \sum_x \min(1, \mathbf{P}(Z \geq x) n^{2+\delta}) \\ &\leq \sum_x \min(1, n^{-\log_2 x\delta/2} n^{2+\delta}) \\ &\leq \sum_i 2^i \min(1, n^{-i\delta/2} n^{2+\delta})\end{aligned}$$

For i and n sufficiently large, the term $2^i n^{-i\delta/2} n^{2+\delta}$ is less than $e^{-\Omega(i)}$, so the summands decays exponentially and the sum converges to a constant. □

We note that $\bar{Z}$ is itself bounded:

COROLLARY 3.4. *For $\delta \geq \delta_0 > 0$, we have $n^{-2-\delta} \leq \bar{Z} \leq n^{-2-\delta} n^{2+o(1)}$.*

Proof. The bound $\bar{Z} = n^{-2-\delta} n^{2+o(1)}$ follows immediately from Lemma 2.3.

The bound $\bar{Z} \geq n^{-2-\delta}$ follows since the probability that the minimum cut fails is $n^{-2-\delta}$. □

4. The Contraction Algorithm. The key to our analysis is an algorithm of [10] for finding cuts in a graph. This algorithm is called the Contraction Algorithm, and can be summarized as follows. Suppose we wish to find α-cuts of G.

```
1. Repeat the following while $G$ has at least $\lceil 2\alpha + 1 \rceil$ vertices:
    2. Select an edge $e$ of $G$ uniformly at random.
    3. Let $G'$ denote the graph obtained by contracting the edge
       $e \in G$. This reduces the number of vertices of $G$ by one.
       Eliminate any resulting self-loops from $G'$
    4. Update $G \leftarrow G'$.
5. Once the graph $G$ has reduced to $\lceil 2\alpha \rceil$ vertices, select a cut
   uniformly at random among all cuts of $G$.
```

We are only partially interested in the Contraction Algorithm as an algorithm *per se*, that is, we do not intend to execute this procedure. Rather, we will use this algorithm as a tool to analyze the number of cuts that are present in the graph. Note that this algorithm produces a single randomly chosen cut; the size of this cut is variable.

We will show that, for an arbitrary cut C, the contraction algorithm selects C with a certain probability. In this type of analysis, we regard C as *fixed.* We refer to C as the *target cut.* During the evolution of the Contraction Algorithm, we start with the original graph G and obtain a variety of subgraphs by contracting edges. For each r decreasing from n down to $\lceil 2\alpha \rceil$, we obtain a subgraph G_r with r vertices and M_r edges (where $G_n = G$). In order for the Contraction Algorithm to find C, this cut must remain in all the subgraphs G_r. In this case we say the Contraction Algorithm *succeeded.* We emphasize that the Contraction Algorithm is run without any target in mind, and hence we cannot really say that it succeeds or fails; this is only relative to a notional target.

The contraction process for C. The Contraction Algorithm can be thought of as a branching process, in which at each stage we select one edge in the current graph G_r and obtain a new subgraph G_{r-1}. In order to find the probability of retaining the target cut C, we define the *contraction process for C* as follows. In each iteration, we uniformly select an edge of G_r *other than C itself.* This can be thought of as a kind of sequential importance sampling in which we are trying to estimate the number of paths which preserve C: at each stage, we count how many choices are available which preserve C, but we never choose to visit any path which removes C (because exploring such a path can provide no information).

The contraction process can be applied to any edge-set L, not only a cut. In this case, it is possible for the edges in L to be contracted, even though they are not themselves selected, if another edge with the same end-points is selected. (This never happens if L is a cut). The Contraction Algorithm can be thought of as the contraction process for the empty edge set.

We define the random variable $M_r^{G,C}$ to be the number of edges in G_r starting with graph G, when we run the contraction process for C. Also define

$$S_r^{G,C} = \frac{c}{M_{r+1}^{G,C}} + \cdots + \frac{c}{M_n^{G,C}};$$

recall here that c is the weight of the min-cut.

When the cut C or graph G is understood, we sometimes omit them and write M_r and S_r. To simplify some notations, when r is a real number, we define $S_r = S_{\lceil r \rceil}$.

Showing an upper bound on $S_r^{G,C}$ is key to proving that the target α-cut C is selected with high probability:

LEMMA 4.1. *Let $\alpha \geq 1$.*

The probability that the Contraction Algorithm selects a given α-cut C is at least

$$\mathbf{P}(\textit{select } C) \geq e^{-O(\alpha)} \mathbf{E}[-\exp(\alpha S_{2\alpha}^{G,C})]$$

Note that by Jensen's inequality, this implies that

$$\mathbf{P}(\textit{select } C) \geq e^{-O(\alpha)} \exp(-\alpha \mathbf{E}[S_{2\alpha}^{G,C}])$$

Proof. In order to select C, the following events must happen. First, for each iteration $r = n, \dots, \lceil 2\alpha + 1 \rceil$, we must select an edge of G_r other than those contained in C. The probability of this event is at least $1 - \alpha c / M_r$. Finally, we must select the cut C from the graph $G_{\lceil 2\alpha \rceil}$. This graph has $2^{\lceil 2\alpha \rceil - 1}$ cuts, so we select C with probability at least $2^{-2\alpha}$. The probability of retaining C up to $G_{2\alpha}$ is the expected value of the product, over $r = \lceil 2\alpha + 1 \rceil, \dots, n$, of $(1 - \alpha c / M_r)$. Note that this expectation is taken only over the paths through the branching process which retain C. Hence, this expectation can also be thought of as the unconditioned expectation *for the contraction process for C*; this equivalence is useful to keep in mind.

Note that each subgraph G_r has minimum cut at least c, and in particular, $M_r \geq rc/2$ with certainty.

Putting these together, the total probability of retaining G is at least

$$\begin{aligned}
\mathbf{P}(\text{select } C) &\geq \mathbf{E}[2^{-2\alpha} \prod_{i=\lceil 2\alpha+1 \rceil}^{n} (1 - \frac{\alpha c}{M_r})] \\
&\geq 2^{-2\alpha} \mathbf{E}[\prod_{i=\lceil 2\alpha+1 \rceil}^{n} \exp(-\alpha c / M_r) \frac{1 - \frac{\alpha c}{rc/2}}{\exp(-\frac{\alpha c}{rc/2})}] \\
&\quad \text{(since } x \mapsto e^x (1-x) \text{ decreases in } [0,1] \text{ and since } M_r \geq rc/2) \\
&= 2^{-2\alpha} \mathbf{E}[\exp(-\alpha S_{2\alpha}) \prod_{i=\lceil 2\alpha+1 \rceil}^{n} (1 - \frac{2\alpha}{r}) \exp(\frac{2\alpha}{r})]
\end{aligned}$$

$$\geq 2^{-2\alpha}\binom{n}{2\alpha}^{-1}(\frac{n}{2\alpha+1})^{2\alpha}\mathbf{E}[\exp(-\alpha S_{2\alpha})]$$
$$\geq e^{-O(\alpha)}\mathbf{E}[\exp(-\alpha S_{2\alpha})] \qquad \text{(by Stirling's formula)}$$

□

Note that

$$S_r \leq \sum_{i=r+1}^{n} \frac{c}{ic/2} \leq 2\log(n/r) \tag{4.1}$$

with certainty. Combining this with Lemma 4.1, we get a simple and well-known lower bound on the probability of retaining C that does not depend on any other properties of G:

Corollary 4.2. *The probability that the Contraction Algorithm selects a given α-cut C is at least*

$$\mathbf{P}(\textit{select } C) \geq e^{-O(\alpha)}(n/(2\alpha))^{-2\alpha}.$$

During the contraction process for C, the cut C complicates the dynamics. In particular, cuts C' that overlap heavily with C are preserved with high probability. The following series of lemmas show how we can "factor out C". To do so, we will analyze not G itself, but graphs related to G by edge contraction. We define the graph G/L to be the result of contracting in G the edges in the set L; $G/\{e\}$ will often be referred to just as G/e. We show that the Contraction Process for C can be approximated by the Contraction Algorithm on G/L, where $L \subseteq C$ is a set of well-chosen edges from the cut C.

Lemma 4.3. *Let C be a cut of a connected graph G and let L a subset of the edges of G. Suppose $r > |L|$. Then the random variable $M_r^{G,C}$ stochastically dominates $|L| + M_{r-|L|}^{G/L,C/L}$. Note that C/L is not necessarily a cut of G/L.*

In particular we have

$$\mathbf{E}[S_r^{G,C}] \leq \mathbf{E}[S_{r-|L|}^{G/L,C/L}]$$

Proof. As described in [7], there is an alternative description of the contraction process for G, C: for each edge $e \in G - C$, we select a random permutation of the edges. We then process these edges in order, either contracting the edge or ignoring it if it was already contracted.

Now consider a coupling experiment: given a permutation ρ of the edges $e \in G$, we run respectively the contraction processes on the graphs G/L and G, using the common permutation ρ to order the relevant edges in both cases.

When ρ is fixed in this way, the relevant terms $M_r^{G,C}$ and $M_{r-|L|}^{G/L,C/L}$ are fully deterministic. We claim that for a fixed ρ, we have

$$M_r^{G,C} \geq M_{r-|L|}^{G/L,C/L} + |L|$$

and this will show our claim.

Now fix ρ, and list the edges of $G - C$ in order of ρ as $e_1, e_2, \ldots, e_{m-|C|}$. Observe that the contraction process for both G, C and $G/L, C/L$ successively selects these edges in this order, and contracts it if still present in the graph (either G or G/L

respectively). We say that the process is at stage k if it has processed edges $e_1, \dots, e_k$ in order.

Suppose that when this process is at stage k, then the graph G has reduced to r vertices and contains s edges of L. Then the contraction process for G/L at this stage must have $r' \geq r - s$ vertices, as it has at most s additional edges contracted away. Also note that every edge in G/L at stage k also remains in graph G, and in addition G has s edges which are not present for G/L. Hence we have

$$M_r^{G,C} \geq M_{r'}^{G/L,C/L} + s \geq M_{r-s}^{G/L,C/L} + s$$

Next, note that every time the number of vertices is decreased by one in the process, so too must the number of edges be reduced by at least one. Hence we have

$$M_{r-s}^{G/L,C/L} + s \geq M_{r-s-1}^{G/L,C/L} + s + 1 \geq \dots M_{r-|L|}^{G/L,C/} + |L|.$$

and our claim is proved. $\square$

To illustrate how we can use Lemma 4.3, suppose we are interested in the contraction process for C, where C contains very few edges. In this case, if we apply this lemma with $L = C$, then we would have

$$\mathbf{E}[S_r^{G,C}] \leq \mathbf{E}[S_{r-|C|}^{G/C}] \approx \mathbf{E}[S_r^{G/C}]$$

that is, the contraction process for C has approximately the same behavior as the contraction algorithm on the graph G/C. In fact, we will use precisely this strategy in Section 5. For a general graph, however, the number of edges in the cut C may be far larger than n. In this case, Lemma 4.3 cannot be directly applied with $L = C$. Instead of contracting away the entire cut C, we will contract only a small, well-chosen subset of its edges. This will largely, but not entirely, remove the influence of C from the residual graph G/L. In analyzing the contraction process for C, we will need to keep track of not only the cut C, but all the other cuts of G as well. The follow lemmas illustrate how such cuts C' can be tracked throughout the contraction process.

Lemma 4.4. *Let $\theta \in (0,1)$; θ could be a function of n or other parameters. Given G and the target α-cut C, there is a subset of the edges $L \subseteq C$ with the following properties:*

1. *$|L| \leq O(\frac{\alpha \log n}{\theta})$.*
2. *For any cut C' disjoint to L, we have $|C' - C| \geq (1-\theta)|C'|$. (Recall that C, C' here should be interpreted as sets of edges.)*

*We refer to the second condition as the θ-*cut-independence property (with respect to C).

Proof. Choose L to be a random subset of C of size exactly $\frac{3\alpha \log n}{\theta}$. (If this number is larger than the number of edges in C, we simply set $L = C$, in which case the lemma is trivially true.)

Let C' be any cut of G of weight $|C'| = \alpha' c$, and satisfying $|C' - C| < (1-\theta)C'$. Thus C' overlaps in at least $\theta \alpha' c$ edges with C. The probability that no such edges are chosen in L is

$$\begin{aligned}\mathbf{P}(C' \text{ disjoint to } L) &= \frac{\binom{\alpha c - \theta \alpha' c}{(3/\theta)\alpha \log n}}{\binom{\alpha c}{(3/\theta)\alpha \log n}} \\ &\leq (\frac{\alpha c - \theta \alpha' c}{\alpha c})^{(3/\theta)\alpha \log n} \\ &\leq n^{-3\alpha'}\end{aligned}$$

By the union bound, the probability that there is *some* such cut C' is at most

$$\mathbf{P}(\text{Some cut } C' \text{ has } |C' - C| \leq (1-\theta)|C'|) \leq \sum_{\text{Cuts } C'} (n^{-3})^{|C'|/c}$$

By Lemma 2.3, this is $O(n^{-1})$, where the constant term is independent of θ. In particular, for n sufficiently large, such an L exists. □

Finally, we show that cuts C' that do not overlap too much with C are destroyed during the contraction process with high probability:

PROPOSITION 4.5. *Let C' be any cut of G. Then the probability of retaining C' through a single iteration of the contraction process for $L \subseteq C$ is at most*

$$\mathbf{P}(\textit{retain } C') \leq 1 - \frac{|C' - C|}{|E|}$$

In particular, if G has θ-cut-independence with respect to C, then

$$\mathbf{P}(\textit{retain } C') \leq 1 - \frac{|C'|(1-\theta)}{|E|}$$

Proof. In a single iteration of the contraction process, we select an edge of $G - L$ uniformly at random. There are $|E| - |L|$ such edges, hence we select an edge of $C' - L$ with probability $\frac{|C'-L|}{|E|-|L|}$. So the probability of retaining C' is

$$\mathbf{P}(\text{retain } C') \leq 1 - \frac{|C' - L|}{|E| - |L|} \leq 1 - \frac{|C' - C|}{|E|}$$

as desired. □

5. Bounds for small, odd c. Our ultimate goal for this algorithm is show that the reliability of a graph influences the number of cuts it can have. As a warm-up exercise, we will show that graphs with connectivity c, where c is a constant odd number, have noticeably fewer α-cuts than the worst case (where c is even and the graph is a cycle with each edge having multiplicity c). This result is not needed for our main algorithm, and can be skipped.

For the purposes of this section only, we regard c as constant. So the constant terms hidden in the asymptotic notations may depend on c. Using techniques developed in Section 6, it is possible to develop estimates in which c is allowed to grow unboundedly. However, when c is large these estimates are not very useful.

We use the following fact about the minimum cuts of G, when c is odd. This is shown in [1], [2]:

PROPOSITION 5.1. *Suppose G has minimum cut c, for c odd. Then G has at most $2n$ mincuts, which are represented by the edges of a spanning tree of G.*

Although our goal is the Contraction Process for a cut C, it will suffice to analyze the unconditioned Contraction Algorithm. The basic strategy of this proof is use induction on the number of vertices to show a bound on S_i. We will need to track the behavior of S_i not only for the original graph G, but for subgraphs H which arise during the evolution of the Contraction Algorithm. Specifically, in order to prove the induction, we will need to generalize the induction hypothesis to track the number of minimum-weight cuts in these subgraphs.

The induction proof by itself is not very intuitive, because it requires guessing a bound on $\mathbf{E}[S_i]$ and then proving that this bound is correct. The proof by itself gives very little help in deriving this bound. So we will give an intuitive and non-rigorous derivation of the proof. We will then prove rigorously that the bound we obtain is in fact correct.

Suppose we start with a graph G with r vertices and k minimum-weight cuts; note that $k \leq 2r$ by Proposition 5.1. *This fact that $k \leq 2r$ is the only part of the proof where we use the assumption that c is odd.* At each stage of the contraction process, we have

$$\mathbf{E}[S_{i-1}] = \mathbf{E}[S_i] + \mathbf{E}[c/M_i]$$

where M_i is the number of edges in the subgraph G_i and $S_i = \frac{c}{M_{i+1}^G} + \cdots + \frac{c}{M_n^G}$. As shown in Proposition 4.5, at every step, the expected number of minimum cuts in the graphs G_i decreases by e^{-c/M_i}, hence the expected number of minimum cuts in the graph G_i should be about $\mathbf{E}[k_i] \approx \mathbf{E}[ke^{-S_i}] \approx ke^{-\mathbf{E}[S_i]}$.

Now the neighborhood of each vertex of G_i defines a cut. As G_i has k_i minimum cuts, this implies that at most k_i vertices may have the minimum degree c, while the others must have degree at least $c+1$, so that

$$M_i \geq k_i c/2 + (i - k_i)(c+1)/2$$

and hence

$$\begin{aligned} c/M_i &\leq \frac{c}{k_i c/2 + (i-k_i)(c+1)/2} \\ &= \frac{c}{i \cdot [(k_i/i)c/2 + (1-k_i/i)(c+1)/2]} \\ &\leq \frac{k_i}{i}\frac{2}{i} + (1 - \frac{k_i}{i})\frac{2c}{(c+1)i}, \end{aligned}$$

where in the last line, we used Jensen's inequality for $\mathbf{E}[1/Z]$ with Z being a two-point distribution. Thus we get the approximation

$$\begin{aligned} \mathbf{E}[c/M_i] &\leq \frac{\mathbf{E}[k_i]}{i}\frac{2}{i} + (1 - \frac{\mathbf{E}[k_i]}{i})\frac{2c}{(c+1)i} \\ &\approx \frac{2(ke^{-\mathbf{E}[S_i]} + ci)}{(1+c)i^2} \end{aligned}$$

This gives us a recurrence relation in $\mathbf{E}[S_i]$:

$$\mathbf{E}[S_{i-1}] = \mathbf{E}[S_i] + \frac{2(ke^{-\mathbf{E}[S_i]} + ci)}{(1+c)i^2}$$

We relax this recurrence relation to a differential equation

$$\begin{aligned} \frac{d\mathbf{E}[S_i]}{di} &= -\frac{2(ke^{-\mathbf{E}[S_i]} + ci)}{(1+c)i^2} \\ S_r &= 0 \end{aligned}$$

which can be solved in closed form to obtain

$$\mathbf{E}[S_i] = \log\Big[\frac{(i/r)^{\frac{2}{c+1}} r(2k + (c-1)r) - 2ik}{(c-1)i^2}\Big]$$

This derivation makes a number of unwarranted independence and monotonicity assumptions on the behavior of the random variables, which do not hold in general. The most problematic of these is extending Proposition 4.5 to multiple steps of the Contraction Algorithm, assuming independence at each step. However, as we will see, this argument does accurately capture the *worst-case* behavior for all the random variables. That is, even though the random variables are not independent, any dependency would only give us better bounds.

We contrast this approach with that of [16], which gives a survey of methods to use differential equations to solve stochastic systems. As in [16], we make some heuristic assumptions to convert the stochastic system to a dynamical system. Most notably, we assume that a random variable X acts like a deterministic variable which is equal to $\mathbf{E}[X]$. We then solve this using differential equations. This gives us a good starting hypothesis for the behavior of the stochastic system, which we then seek to prove rigorously.

However, in [16], the usual strategy at this point is to use *concentration inequalities* to prove that the random variables are in fact close to their means. In that case, the heuristic assumptions we made earlier were approximately correct, so it is not a surprise that the behavior of the stochastic and dynamical systems agree. In our case, we are not able to show concentration inequalities for the relevant random variables. The reason is that a single step of the Contraction Algorithm, namely contracting an edge, can change the graph in a far-reaching way; this makes it hard to show the type of "local effect" required for a concentration inequality.

By contrast, we will prove that any violation of these heuristic assumptions will only help us; we do not necessarily show that the heuristic assumptions are true or approximately true. This is very similar to how one can apply Jensen's inequality to interchange the expectation of a function with the function of its expectation. As in Jensen's inequality, our proof relies very strongly on the concavity and monotonicity properties of the function we are estimating.

In the following theorem, we carry out this approach and prove that our heuristic formula is in fact a correct bound:

Theorem 5.2. *Let $c > 1$. Define the function*

$$f(i,r,k) = \log\Big[\frac{(i/r)^{\frac{2}{c+1}} r(2k + (c-1)r) - 2ik}{(c-1)i^2}\Big]$$

Suppose H is a graph with r vertices, k cuts of weight c, and no cuts of weight $< c$. (If $k > 0$, c is the minimum cut). Then for i sufficiently large and $r \geq i$ we have $E[S_i^H] \leq f(i,r,k)$.

Proof. For simplicity, we will defer some technical analysis of the function f to after the main proof. First, note that by Proposition 5.3, the function f is well-defined for appropriate parameters.

We prove the Theorem by induction on r. When $r = i$, we have $S_i = f(i,r,k) = 0$.

Now suppose $r \geq i+1$, and H has m edges and $k \leq 2r$ cuts of weight c. In the first step of the Contraction Algorithm, we select an edge of H to contract, arriving at a new graph H'. So we have $\mathbf{E}[S_i^H] = c/m + \mathbf{E}_{H'}\mathbf{E}[S_i^{H'}]$. We have broken the expectation into two components. First, we randomly select the next subgraph H'; then, we continue the contraction algorithm on that subgraph.

The graph H' has $r-1$ vertices and has $K' \leq 2(r-1)$ weight-c cuts. Here K' is a random variable. By Proposition 4.5, we have $\mathbf{E}[K'] \leq ke^{-c/m}$.

By the inductive hypothesis, we have

$$\mathbf{E}[S_i^{H'}] \leq f(i, r-1, K')$$

By Proposition 5.4, this is a concave-down increasing function of K', hence by Jensen's inequality we have

$$\mathbf{E}[S_i^H] \leq c/m + f(i, r-1, ke^{-c/m})$$

First suppose $k \leq r$. Now the neighborhood of each vertex of H defines a cut, and for $r \geq 3$ these are all distinct. Hence at most k vertices may have the minimum degree c, while the others must have degree at least $c+1$. That implies that

$$m \geq rc/2 + (r-k)/2$$

By Proposition 5.5, the expression $c/m + f(i, r-1, k^{-c/m})$ is decreasing in m. Then we have the bound

$$\begin{aligned}\mathbf{E}[S_i^H] &\leq \frac{2c}{-k+r+cr} + f(i, r-1, ke^{-\frac{2c}{cr-k+r}}) \\ &\leq f(i,r,k) \qquad \text{by Proposition 5.6.}\end{aligned}$$

Suppose $k \geq r$. Then we have $m \geq rc/2$, so we have the bound

$$\begin{aligned}\mathbf{E}[S_i^H] &\leq \frac{2}{r} + f(i, r-1, ke^{-\frac{2}{r}}) \\ &\leq f(i,r,k) \qquad \text{by Proposition 5.7.}\end{aligned}$$

This completes the induction. □

To complete this proof, we must verify that f has certain concavity and monotonicty properties.

Proposition 5.3. *For $1 \leq i \leq r$, the function f is well-defined.*

Proof. We need to show that the argument of the logarithm is strictly greater than 0. We have

$$\frac{r\left(\frac{i}{r}\right)^{\frac{2}{c+1}}((c-1)r+2k) - 2ik}{(c-1)i^2} \geq \frac{r^2\left(\frac{i}{r}\right)^{\frac{2}{c+1}}}{i^2} > 0$$

□

Proposition 5.4. *For $1 \leq i \leq r$ and $c \geq 1$ the function $f(i,r,k)$ is an increasing, concave-down function of k.*

Proof. The function f applies the logarithm function to an argument which is a linear function of k. The slope of that linear function is

$$\frac{2r((i/r)^{\frac{2}{c+1}} - (i/r))}{(c-1)i^2} > 0$$

for $c > 1$ and $0 \leq i \leq r$. □

Proposition 5.5. *Suppose $c \geq 3$ and $i, k, r \geq 0$. Then the expression*

$$c/m + f(i, r, ke^{-c/m})$$

is a decreasing function of m.

Proof. Let $t = c/m$. We wish to show that $t + f(i, r, ke^{-t})$ is increasing in t. Differentiating with respect to t, it suffices to show that

$$\frac{2k((i/r) - (i/r)^{\frac{2}{c+1}})}{(i/r)^{\frac{2}{c+1}}((c-1)re^t + 2k) - 2(i/r)k} + 1 \geq 0$$

By differentiating with respect to i (viewing i as a continuous parameter), we see that the left-hand side of the above expression is increasing in i. Hence it suffices to show that this holds when $i \to 0$. In this case, as $i \to 0$, the terms $(i/r)^{2/(c+1)}$ are negligible compared to the linear terms in (i/r), so the above approaches a limit as $i \to 0$ of

$$\frac{(c-1)re^t}{(c-1)re^t + 2k}$$

which is obviously positive. □

Proposition 5.6. *For $c \geq 3, r \geq i+1$ and $k \leq r$ we have*

$$\frac{2c}{-k + r + cr} + f(i, r-1, ke^{-\frac{2c}{cr-k+r}}) \leq f(i, r, k)$$

Proof. It suffices to show that

$$\exp\Big(\frac{2c}{-k + r + cr} + f(i, r-1, ke^{-\frac{2c}{cr-k+r}})\Big) - \exp(f(i, r, k)) \leq 0$$

This "simplifies" to showing that

$$\frac{(c-1)(r-1)^2\left(\frac{i}{r-1}\right)^{\frac{2}{c+1}} e^{\frac{2c}{cr-k+r}} + 2k(r-1)\left(\frac{i}{r-1}\right)^{\frac{2}{c+1}} - r\left(\frac{i}{r}\right)^{\frac{2}{c+1}}((c-1)r + 2k)}{(c-1)i^2}$$

is non-positive. Removing terms of known sign, it suffices to show that

$$(c-1)(r-1)^2\left(\frac{1}{r-1}\right)^{\frac{2}{c+1}} e^{\frac{2c}{cr-k+r}} + 2k(r-1)\left(\frac{1}{r-1}\right)^{\frac{2}{c+1}} - r\left(\frac{1}{r}\right)^{\frac{2}{c+1}}((c-1)r + 2k) \tag{5.1}$$

is non-positive. We prove this next.

The second derivative of (5.1) with respect to k is given by

$$(c-1)\left(\frac{1}{r-1}\right)^{\frac{2}{c+1}-2}\left(\frac{4c^2 e^{\frac{2c}{cr-k+r}}}{(cr-k+r)^4} + \frac{4ce^{\frac{2c}{cr-k+r}}}{(cr-k+r)^3}\right) \geq 0$$

This implies that (5.1) is either increasing in k, or initially decreasing before increasing. In either case, it attains its maximum value at either $k = 0$ or $k = r$. So it suffices to show that (5.1) is non-positive in these two cases.

When $k = 0$, our goal simplifies to

$$e^{\frac{2c}{cr+r}}\left(\frac{1}{r-1}\right)^{-\frac{2c}{c+1}} \leq \left(\frac{1}{r}\right)^{-\frac{2c}{c+1}}$$
$$\Leftrightarrow e^{1/r}(r-1) \leq r$$

which holds for all $r \geq 0$ since $e^{-1/r} \geq 1 - 1/r$.

Similarly, when $k = r$, our goal becomes

$$(c-1)e^{2/r}(r-1)^{\frac{2c}{c+1}} + 2r(r-1)\left(\frac{1}{r-1}\right)^{\frac{2}{c+1}} - (c+1)r^{\frac{2c}{c+1}} \leq 0. \tag{5.2}$$

To prove this, we make the transformation $y = \frac{c-1}{c+1}$; note that $1/2 \leq y \leq 1$. Thus we need to establish

$$\begin{aligned}(r-1)^y \cdot \Big[r(1-y) + ye^{2/r}(r-1)\Big] &\leq r^{y+1}, \text{ i.e.,} \\ 1 - y + ye^{2/r}(1 - 1/r) &\leq \left(\frac{r}{r-1}\right)^y.\end{aligned}$$

Now, elementary calculus shows that $e^{2/r}(1-1/r) \leq 1 + 1/r$ for $r \geq 1$. (Set $z = 1/r$ and observe that the derivative of $e^{2z}(1-z)$, which is $e^{2z}(1-2z)$, is at most 1 for positive z since $e^{-2z} \geq 1 - 2z$.) Thus, it suffices to prove that

$$1 + y/r \leq \left(\frac{r}{r-1}\right)^y$$

for $r > 1$ and $1/2 \leq y \leq 1$. However, note that since $e^{-1/r} \geq 1 - 1/r$, we have that $\frac{r}{r-1} \geq e^{1/r}$. Thus it is enough to show that $1 + y/r \leq e^{y/r}$, which is indeed the case. □

Proposition 5.7. *For $c \geq 3, r \geq i+1$ we have*

$$\frac{2}{r} + f(i, r-1, ke^{-\frac{2}{r}}) \leq f(i, r, k)$$

Proof. It suffices to show that

$$\exp\Big(\frac{2}{r} + f(i, r-1, ke^{-\frac{2}{r}}\Big) - \exp(f(i, r, k)) \leq 0$$

This simplifies to

$$\frac{(r-1)\left(\frac{i}{r-1}\right)^{\frac{2}{c+1}}\big((c-1)e^{2/r}(r-1) + 2k\big) - r\left(\frac{i}{r}\right)^{\frac{2}{c+1}}((c-1)r + 2k)}{(c-1)i^2} \leq 0$$

Hence it suffices to show that

$$\left(\frac{1}{r-1}\right)^{\frac{2}{c+1}-1}\Big((c-1)e^{2/r}(r-1) + 2k\Big) + \left(\frac{1}{r}\right)^{\frac{2}{c+1}-1}(-cr - 2k + r) \leq 0$$

The LHS of this can be easily seen to be decreasing in k, so it suffices to show that this holds at $k = r$. In this case we need to show that

$$\left(\frac{1}{r-1}\right)^{\frac{2}{c+1}-1}\Big((c-1)e^{2/r}(r-1) + 2r\Big) - (c+1)\left(\frac{1}{r}\right)^{-\frac{2c}{c+1}} \leq 0$$

which is straightforward. □

A similar argument deals with the case $c = 1$. We omit the proof, as it is essentially identical to Theorem 5.2:

PROPOSITION 5.8. *Suppose H is a graph with r vertices and cut-value 1, and with $k \leq r$ cuts of weight 1 (i.e., bridges). Then for i sufficiently large and $r \geq i$ we have*

$$\mathbf{E}[S_i^H] \leq \log\Big[\frac{r - k\log(i/r)}{i}\Big]$$

Theorem 5.2 gives us a very precise estimate of $\mathbf{E}[S_i]$. Usually, a cruder and simpler estimate suffices:

LEMMA 5.9. *Suppose G has minimum cut c, for c odd and constant. Then we have*

$$\mathbf{E}[S_i] = \frac{2c}{c+1}\log(n/i) + O(1)$$

where the constant term does not depend on c.

Proof. The graph G has n vertices and $k \leq 2n$ cuts of weight c.

First suppose $c \geq 3$. Then by Theorem 5.2 we have

$$\begin{aligned}\mathbf{E}[S_i^G] &\leq f(i,n,2n)\\ &= \log\left(\frac{(c+3)(i/n)^{\frac{2}{c+1}} - 4(i/n)}{(c-1)(i/n)^2}\right)\\ &\leq \log\Big[\frac{(c+3)(i/n)^{2/(c+1)}}{(c-1)(i/n)^2}\Big] + \frac{4(i/n)^{1-2/(c+1)}}{c+1}\\ &= \frac{2c}{c+1}\log(n/i) + (1+\log(3/2))\end{aligned}$$

A similar proof, using Proposition 5.8, applies when $c = 1$. □

We can now estimate the probability of selecting the α-cut C:

THEOREM 5.10. *Suppose c is an odd constant and C is an α-cut. Then C is selected by the Contraction Algorithm with probability*

$$\mathbf{P}(\textit{Contraction Algorithm selects cut } C) \geq e^{-O_c(\alpha)}(n/\alpha)^{-\frac{2\alpha c}{c+1}} = \Omega_c(n^{-\alpha\frac{2c}{c+1}}).$$

Proof. First, suppose $n < \alpha c + 2\alpha$. In this case, the simple analysis of the Contraction Algorithm of Corollary 4.2 (which ignores the fact that c is odd) shows that the probability of selecting C is at least $\exp(-O(\alpha))(n/\alpha)^{-2\alpha}$. For $n < \alpha c$ this is $\exp(-O(\alpha))$.

Next, suppose $n > \alpha c + 2\alpha$. We now have

$$\begin{aligned}\mathbf{E}[S_{2\alpha}^{G,C}] &\leq \mathbf{E}[S_{2\alpha+\alpha c}^{G,C}] + 2\log(\frac{\alpha c}{2\alpha})\\ &\leq \mathbf{E}[S_{2\alpha}^{G/C,\emptyset}] + O(1) \qquad \text{by Lemma 4.3}\\ &\leq \frac{2c}{c+1}\log(n/\alpha) + O(1)\end{aligned}$$

Hence the probability of selecting C is at least $e^{-O(\alpha)}(n/\alpha)^{-\alpha\frac{2c}{c+1}}$.

When α grows large, the super-exponential term $\alpha^{-\alpha\frac{2c}{c+1}}$ dominates the simply exponential term $e^{-O(\alpha)}$. So the terms involving α alone are bounded, and

$$\mathbf{P}(\text{Contraction Algorithm selects cut } C) = \Omega(n^{-\alpha\frac{2c}{c+1}}).$$

□

The following example shows that we cannot achieve any probability of the form $n^{-\alpha x}$ where x is a constant with $x < \frac{2c}{c+1}$:

PROPOSITION 5.11. *Let c be odd, and let $\alpha = k\frac{c+1}{2c}$ with k an integer. Then there is a graph G of minimum cut c in which the number of α-cuts is*

$$(\frac{n}{\alpha})^{\frac{2c}{c+1}\alpha}\exp(\Omega(\alpha))$$

Proof. Consider a cycle graph consisting of bundles of $(c+1)/2$ edges between each adjacent pair of vertices, except for one edge bundle of $(c-1)/2$ edges. This graph has minimum cut c. Suppose we select any k edge-bundles which use the $(c+1)/2$ edges to fail. This gives a cut of weight $k(c+1)/2$, which is $k\frac{c+1}{2c}$ times the minimum cut as indicated. There are $\binom{n-1}{k}$ such choices, so the total number of such α-cuts is at least

$$\binom{n-1}{k} = \binom{n-1}{\frac{2\alpha c}{1+c}} = (\frac{n}{2\alpha})^{\frac{2c}{c+1}\alpha}\exp(\Omega(\alpha)).$$

□

6. The effects of graph reliability on the Contraction Algorithm. We now show how the graph reliability affects the behavior of the Contraction Algorithm. We begin with a warm-up exercise, which will illustrate some of the ideas in the proof. We then discuss how this simple proof falls short and how to improve it, at the cost of greater complexity.

Suppose we are given a target cut C. We show how to draw a connection between $\bar{Z}$ and the behavior of the Contraction Algorithm. If $\bar{Z}$ is large (say as large as n^2p^c), then the graph G could be essentially like the cycle graph, and the Contraction Algorithm is tight. So we will suppose $\bar{Z}$ is much smaller than this.

PROPOSITION 6.1. *Suppose $p^c = n^{-2-\delta}$ and $\bar{Z} = n^{-2-\delta+\beta}$, where $\delta \geq \delta_0 > 0$.*

(Here, δ, β should be thought of as "parameters" of the graph; δ measures the probability that the smallest cut of G fails, while $\beta \in [0, 2+o(1)]$ counts the number of small cuts. The case $\beta = 0$ means there is only a single small cut and the other cuts are very large; the case $\beta = 2$ corresponds to the cycle graph).

Define

$$h(\beta,\delta,x) = \begin{cases} 2(2+\delta)(\log(3-\beta+\delta) - \log(2-\beta+\delta+x)) & \text{if } x \geq \beta \\ 2(2+\delta)(\log(3-\beta+\delta) - \log(2+\delta)) + 2(\beta - x) & \text{if } x \leq \beta \leq 1 \\ 2(1-x) & \text{if } \beta \geq 1 \end{cases}$$

Then for $x \in [0,1]$ we have

$$S_i \leq h(\beta, \delta, \frac{\log i}{\log n})\log n$$

Proof. Consider iteration i during the Contraction Algorithm, arriving at a subgraph with i vertices. The neighborhoods of each vertex v determine a cut with weight d_v, where d_v is the degree of v.

As the neighborhood of each vertex of G_i is a distinct cut, we have

$$\sum_v p^{d_v} \leq \bar{Z} e^{-d/c(1-\theta)\mathbf{E}[S_i]}.$$

Now note that $ip^{\frac{\sum_v d_v}{i}} \leq \sum_v p^{d_v}$ by concavity. Hence we have

$$ip^d \leq \bar{Z}$$

which in turn implies that

$$M_i \geq \frac{i \log(\bar{Z}/i)}{2 \log p}$$

We also have the bound $M_i \geq ic/2$, which is a better bound for $i \leq n^\beta$. We thus have

$$\begin{aligned}
S_i &\leq \sum_{r=i+1}^{n} \min(\frac{c}{rc/2}, \frac{2c \log p}{r \log(\bar{Z}/r)}) \\
&\leq \int_{r=i}^{n} \min(\frac{c}{rc/2}, \frac{2c \log p}{r \log(\bar{Z}/r)}) dr \\
&\leq h(\beta, \delta, x) \log n \qquad \text{by simple calculus}
\end{aligned}$$

□

This simple proof demonstrates the effect of the graph reliability on the behavior of the Contraction Algorithm. In fact, this Proposition 6.1 is already sufficient to obtain a substantially improved run-time compared to Karger's algorithm. We will also need this Proposition 6.1 for technical reasons later in this paper.

However, Proposition 6.1 is not tight, because it assumes that the value of $\bar{Z}$ does not change during the execution of the Contraction Algorithm. But in fact, most of the cuts in the graph are being contracted away, and these disappear from $\bar{Z}$. Hence in reality, $\bar{Z}$ should be decreasing rapidly.

To describe this behavior at least intuitively, suppose we are given a target cut C and the graph G with r vertices and which has θ-cut-independence with respect to C. In order to estimate $\mathbf{E}[S_i]$, we need to estimate $\mathbf{E}[c/M_i]$ for the graph G_i obtained during the evolution of the Contraction Process for C.

Suppose that all of the cuts of G have size $d \geq c$. In this case, in any individual step of the Contraction Process, a cut C' survives with probability $e^{-(1-\theta)d/M_i}$. Multiplying the probabilities for each iteration, we expect that each cut survives with probability about $e^{-(1-\theta)d/cS_i}$, and so we should have

$$\mathbf{E}[\bar{Z}^{G_i}] \approx \bar{Z} e^{-d/c\mathbf{E}[S_i]}$$

As the neighborhood of each vertex of G_i is a distinct cut, we have

$$ip^d \leq \bar{Z} e^{-d/c(1-\theta)\mathbf{E}[S_i]}$$

which in turn implies that

$$M_i \geq -\frac{c \log\left(i/\bar{Z}\right)}{2(c \log p + \mathbf{E}[S_i](1-\theta))}$$

As in the case of Theorem 5.2, we will use an induction argument to bound $\mathbf{E}[S_i]$. This induction is unintuitive, so we begin with a heuristic and non-rigorous derivation.

Suppose we are given a target cut C and the graph G with r vertices and which has θ-cut-independence with respect to C. In order to estimate $\mathbf{E}[S_i]$, we need to estimate $\mathbf{E}[c/M_i]$ for the graph G_i obtained during the evolution of the Contraction Process for C.

Suppose that all of the cuts of G have size $d \geq c$. In this case, in any individual step of the Contraction Process, a cut C' survives with probability $e^{-(1-\theta)d/M_i}$. Multiplying the probabilities for each iteration, we expect that each cut survives with probability about $e^{-(1-\theta)d/cS_i}$, and so we should have

$$\mathbf{E}[\bar{Z}^{G_i}] \approx \bar{Z} e^{-d/c\mathbf{E}[S_i]}$$

We hope that Jensen-type inequality holds, so that

$$\mathbf{E}[c/M_i] \approx -\frac{2(c\log p + \mathbf{E}[S_i](1-\theta))}{i\log\left(i/\bar{Z}\right)}$$

So we have a recurrence relation in S_r, namely

$$\mathbf{E}[S_{i-1}] = \mathbf{E}[S_i] - \frac{2\mathbf{E}[S_i](1-\theta) + 2c\log p}{i\log(i/\bar{Z})}$$

We relax this to a differential equation

$$\frac{d\mathbf{E}[S_i]}{di} = \frac{2\mathbf{E}[S_i](1-\theta) + 2c\log p}{i\log(i/\bar{Z})}$$
$$S_r = 0$$

which can be solved in closed form to obtain

$$\mathbf{E}[S_i] = \frac{-c\log p\Big(1 - \Big(\frac{\log(\bar{Z}/i)}{\log(\bar{Z}/r)}\Big)^{2-2\theta}\Big)}{1-\theta}$$

This derivation is completely non-rigorous, as it makes a number of unwarranted independence and monotonicity assumptions. However, as we will see, all of these assumptions turn out to be the worst-case behavior for S_r. Hence the above bound is essentially correct.

In order to carry out the induction proof, we will need to consider the expected number of failed cuts with the original edge-failure probability p as well as other probabilities $p' > p$. To simplify the notation, we will slightly reparametrize $\bar{Z}$. We define the graph parameter

$$A_\gamma^G = \sum_{\text{cuts } C' \text{ of } G} e^{\gamma|C'|/c}$$

Note that $A_{c\log p} = \bar{Z}$, and note that $A_\gamma^G > 0$ for all γ.

THEOREM 6.2. *Define the function*

$$f(i,r,a,\gamma) = \begin{cases} \frac{-\gamma\Big(1-\Big(\frac{\log(a/i)}{\log(a/r)}\Big)^{2-2\theta}\Big)}{1-\theta} & \textit{if } a \leq 1 \\ 2\log(r/i) & \textit{if } a > 1 \end{cases}$$

Fix a graph G with n vertices and minimum cut c. Let C be a fixed target cut of G, and suppose G has θ-cut-independence with respect to C, for some parameter $\theta \in (0,1)$. Let $L \subseteq C$ be an arbitrary subset of the edges of C. Let H be a subgraph obtained from G by edge contraction. Let γ be a real number in the range $0 \leq \gamma \leq -2\log r$ and let $a > 0$.

Then for i sufficiently large, and $i \leq r$ we have

$$\mathbf{E}[S_i^{H,L}] \leq f(i, r, A_\gamma^H, \gamma)$$

Proof. We prove this by induction on r. We defer the proof of some technical properties, for sake of clarity.

When $r = i$, we have $S_i^{H,L} = f(i, r, a, \gamma) = 0$.

Now suppose $r \geq i + 1$, and H has m edges with $A_\gamma^H = a$. We may assume that $a \leq 1$, as otherwise this follows immediately from (4.1).

So $\mathbf{E}[S_i^{H,L}] = c/m + \mathbf{E}_{H'}\mathbf{E}[S_i^{H',L'}]$. We have broken the expectation into two components: first, we select an edge e of $H - E$ to contract, leading to the graph $H' = H/e$ and to $L' = L/e$; second, we continue the Contraction Process on the subgraph H'. Note that the subgraph H' must also have θ-cut-independence with respect to C.

Define $\gamma' = \gamma + (1-\theta)c/m$. The graph H' has $r - 1$ vertices. By Proposition 4.5, each cut C' of H survives to H' with probability at most $e^{-|C'|(1-\theta)/m}$. Hence

$$\mathbf{E}_{H'}[A_{\gamma'}^{H'}] \leq a$$

Note that $m \geq rc/2$, so we have $\gamma' \leq -2\log r + 2/r \leq -2\log(r-1)$. So the induction hypothesis applies to the graph H', and we obtain

$$\mathbf{E}[S_i^{H',L'}] \leq f(i, r-1, A_{\gamma'}^{H'}, \gamma')$$

By Proposition 6.3, Jensen's inequality applies, and hence we have

$$\mathbf{E}[S_i^{H,L}] \leq c/m + f(i, r-1, a, \gamma + (1-\theta)c/m)$$

We will now bound the number of edges m of the graph H. Suppose the vertices of H have degrees $d_1, \dots, d_r$. As the neighborhood of each vertex defines a distinct cut, we must have

$$\sum_{j=1}^{r} \exp(\gamma d_j / c) \leq A_\gamma^H = a$$

and the total number of edges is $m = \sum d_j/2$.

By concavity, the smallest value of m is obtained when all the degrees are equal to $d = \frac{c\log(a/r)}{\gamma}$. This gives us

$$m \geq \frac{rc\log(a/r)}{2\gamma}$$

As shown in Proposition 6.4, the quantity $c/m + f(i, r-1, a, \gamma + (1-\theta)c/m)$ is decreasing in m. Hence we must have

$$\begin{aligned}\mathbf{E}[S_i^{H,L}] &\leq \frac{2\gamma}{r\log(a/r)} + f(i, r-1, a, \gamma + (1-\theta)\frac{2\gamma}{r\log(a/r)}) \\ &\leq f(i, r, a, \gamma) \qquad \text{by Proposition 6.5}\end{aligned}$$

This completes the induction. □

We now show that f has the required concavity and monotonicity properties:

PROPOSITION 6.3. *For i sufficiently large and $\gamma < 0$, the function $f(i,r,a,\gamma)$ is concave-down in a. For $a \in (0,1]$, it is non-decreasing in a.*

Proof. Let us first examine the behavior of f in the interval $a \in [0,1)$. The derivative of f with respect to a is

$$f_a = \frac{2\left(\log\left(\frac{a}{r}\right) - \log\left(\frac{a}{i}\right)\right)\gamma\left(\frac{\log\left(\frac{a}{i}\right)}{\log\left(\frac{a}{r}\right)}\right)^{1-2\theta}}{a\log^2\left(\frac{a}{r}\right)}$$

By the assumption $a \leq 1$ and $i \leq r$, the term $\log(a/i)/\log(a/r)$ is positive and the term $\log(a/r) - \log(a/i)$ is negative. Hence $f_a \geq 0$ and so the function f is non-decreasing for $a \leq 1$.

The second derivative f_{aa} is given by

$$\frac{2\left(\log\left(\frac{a}{i}\right) - \log\left(\frac{a}{r}\right)\right)\gamma\left(\frac{\log\left(\frac{a}{i}\right)}{\log\left(\frac{a}{r}\right)}\right)^{-2\theta}\left(\log\left(\frac{a}{i}\right)\left(\log\left(\frac{a}{r}\right) - 2\theta + 3\right) + (2\theta - 1)\log\left(\frac{a}{r}\right)\right)}{a^2\log^4\left(\frac{a}{r}\right)}$$

To show $f_{aa} \leq 0$, it suffices to show that

$$\log(a/i)(\log(a/r) - 2\theta + 3) + 2(\theta - 1)\log(a/r) \geq 0$$

For $i \geq 21$, the terms $\log(a/i), \log(a/r) - 2\theta + 3, \log(a/r)$ are all negative, and thus $f_{aa} \leq 0$.

To finish showing that f is concave-down, we must show that it is decreasing at its point of discontinuity at $a = 1$. That is, we need to show that

$$\frac{-\gamma(1 - (\frac{\log i}{\log r})^{2-2\theta})}{1-\theta} \geq 2\log(r/i)$$

The left-hand side is a decreasing function of γ. As $\gamma \leq -2\log r$, it suffices to show that

$$\frac{\log r(1 - (\frac{\log i}{\log r})^{2-2\theta})}{1-\theta} - \log(r/i) \geq 0$$

The left-hand side is decreasing in i and is equal to zero when $i = r$. Hence this inequality holds. □

PROPOSITION 6.4. *For $a \leq 1$ and $r \geq i+1$, the expression*

$$c/m + f(i, r-1, a, \gamma + (1-\theta)c/m)$$

is non-increasing in m.

Proof. For $a \leq 1$, the derivative of this quantity with respect to m is given by

$$\frac{-c\log^2\left(\frac{a}{i}\right)\left(\frac{\log\left(\frac{a}{i}\right)}{\log\left(\frac{a}{r-1}\right)}\right)^{-2\theta}}{m^2\log^2\left(\frac{a}{r-1}\right)}$$

By our assumption that $a \leq 1$ and $r \geq i+1$, this is clearly ≤ 0. □

PROPOSITION 6.5. *For $a \leq 1, \gamma \leq 0$, and r sufficiently large, we have*

$$\frac{2\gamma}{r\log(a/r)} + f(i, r-1, a, \gamma + (1-\theta)\frac{2\gamma}{r\log(a/r)}) \leq f(i, r, a, \gamma)$$

Proof. The RHS minus the LHS is given by

$$\frac{-\gamma \log^2\left(\frac{a}{i}\right)\left(\left(\frac{\log\left(\frac{a}{i}\right)}{\log\left(\frac{a}{r}\right)}\right)^{-2\theta} - \frac{\log\left(\frac{a}{r}\right)\left(r\log\left(\frac{a}{r}\right)-2\theta+2\right)\left(\frac{\log\left(\frac{a}{i}\right)}{\log\left(\frac{a}{r-1}\right)}\right)^{-2\theta}}{r\log^2\left(\frac{a}{r-1}\right)}\right)}{(1-\theta)\log^2\left(\frac{a}{r}\right)}$$

Removing terms with obvious signs, it suffices to show that

$$\log(a/r)(r\log(a/r) - 2\theta + 2)\left(\frac{\log(a/(r-1))}{\log(a/r)}\right)^{2\theta} - r\log^2(a/(r-1)) \geq 0 \qquad (6.1)$$

We claim that the LHS of (6.1) is a decreasing function θ for r sufficiently large. For, differentiating with respect to θ, we wish to show

$$2\log\left(\frac{a}{r}\right)\left(\frac{\log\left(\frac{a}{r-1}\right)}{\log\left(\frac{a}{r}\right)}\right)^{2\theta}\left(\log\left(\frac{\log\left(\frac{a}{r-1}\right)}{\log\left(\frac{a}{r}\right)}\right)\left(r\log\left(\frac{a}{r}\right) - 2\theta + 2\right) - 1\right) \leq 0$$

$$\Leftrightarrow \log\left(\frac{\log\left(\frac{a}{r-1}\right)}{\log\left(\frac{a}{r}\right)}\right)\left(r\log\left(\frac{a}{r}\right) - 2\theta + 2\right) - 1 \geq 0$$

$$\Leftarrow \log\left(\frac{\log\left(\frac{a}{r-1}\right)}{\log\left(\frac{a}{r}\right)}\right)\left(r\log\left(\frac{a}{r}\right) + 2\right) - 1 \geq 0$$

which is a simple exercise in calculus.

Furthermore, when $\theta = 1$, then the LHS of (6.1) simplifies to 0 and the inequality holds. □

We next remove the condition that the graph G must have θ-cut-independence.

THEOREM 6.6. *Suppose $p^c = n^{-2-\delta}$ and $\bar{Z} = n^{-2-\delta+\beta}$, where $\delta \geq \delta_0 > 0$. Define*

$$\bar{h}(\beta, \delta, x) = \frac{(\delta+2)(1-x)(5-2\beta+2\delta+x)}{(3-\beta+\delta)^2}$$
$$v_{\beta,\delta} = \bar{h}(\beta, \delta, 0) = \frac{(\delta+2)(5-2\beta+2\delta)}{(3-\beta+\delta)^2}$$

Then

$$\mathbf{E}[S_{2\alpha}] \leq v_{\beta,\delta}\log n + O(\log\log n) - \Omega(\log\alpha)$$

Also, if we have $\alpha \leq O(1)$ and $x \in [0,1]$, then Then

$$\mathbf{E}[S_i] \leq \bar{h}(\beta, \delta, \frac{\log i}{\log n})\log n + O(\log\log n)$$

Proof. We will only prove the bound on $\mathbf{E}[S_{2\alpha}]$; the other bound is essentially the same.

Note that by Corollary 3.4, we must have $0 \leq \beta \leq 2 + o(1)$.

First, suppose $\alpha \geq \sqrt{n}$. Then, (4.1) gives

$$S_{2\alpha} \leq 2\log(\frac{n}{2\alpha}) + O(1) \leq 1.02\log n - 0.01\log\alpha + O(1).$$

A simple analysis of $v_{\beta,\delta}$ shows that it attains a minimum value of 1.11 at $\beta = \delta = 0$. Hence $S_{2\alpha} \leq v_{\beta,\delta}\log n + O(1) - \Omega(\log\alpha)$ with certainty in this case.

Next, suppose $\alpha < \sqrt{n}$. Given a target cut C, set $\theta = 1/\log n$. Let $L \subseteq C$ be as given by Lemma 4.4, and set $H = G/L$ Note that $|L| \leq O(\frac{\alpha \log n}{\theta}) = O(\alpha\log^2 n)$.

Suppose that H has $r \leq n$ vertices. We must have $\bar{Z}^H \leq \bar{Z} < 1$ and $p^c \leq r^{-2-\delta}$. Hence

$$\begin{aligned}
\mathbf{E}[S_{2\alpha}^{G,C}] &\leq \mathbf{E}[S^{G,C}_{2\alpha+O(\alpha\log^2 n)}] + 2\log(\frac{2\alpha + O(\alpha\log^2 n)}{2\alpha}) \\
&\leq \mathbf{E}[S_{2\alpha}^{G/L,C/L}] + O(\log\log n) \quad \text{(by Lemma 4.3)} \\
&\leq f(2\alpha, n, \bar{Z}, c\log p) + O(\log\log n) \\
&\leq f(1, n, \bar{Z}, c\log p) + O(\log\log n) - \Omega(\log\alpha) \qquad \text{(as } \alpha \leq \sqrt{n}) \\
&\leq v_{\beta,\delta}\log n + O(\log\log n) - \Omega(\log\alpha)
\end{aligned}$$

□

COROLLARY 6.7. *Suppose $p^c = n^{-2-\delta}$ and $\bar{Z} = n^{-2-\delta+\beta}$, where $\delta \geq \delta_0 > 0$. The contraction algorithm selects any given α-cut with probability $\Omega(n^{-\alpha v_{\beta,\delta}}\log^{-O(\alpha)} n)$. In particular, the number of such cuts is at most $O(n^{\alpha v_{\beta,\delta}}\log^{O(\alpha)} n)$.*

Proof. By Lemma 4.1, the probability of selecting any given α-cut C is at least

$$\mathbf{P}(\text{select } C) \geq \exp(-O(\alpha) - \alpha\mathbf{E}[S_{2\alpha}]).$$

By Lemma 6.6 this is at least $\exp(-O(\alpha))\exp(\Omega(\alpha\log\alpha))n^{v_{\beta,\delta}\alpha}\log^{O(\alpha)} n$. Note that all the terms in α alone are bounded, so this is $O(n^{v_{\beta,\delta}\alpha}\log^{O(\alpha)} n)$ as desired. □

7. Bounds on α^*.

Recall that the approach of [7] is to show that most unreliability is due to small cut failure. We can use our bound on the number of cuts to estimate this more precisely than Theorem 2.5.

To simplify the notation in the remainder of this paper, we will write v instead of $v_{\beta,\delta}$. Typically, the value of δ, β should be instead understood from context.

PROPOSITION 7.1. *Suppose $\delta \geq \delta_0 > 0$. Define $U_\alpha(p)$ to be the probability that some cut of weight $\leq \alpha c$ fails.*

For all α we have

$$U(p) - U_\alpha(p) = n^{\alpha v_{\beta,\delta} - \alpha(2+\delta)}\log^{O(\alpha)} n$$

Proof. This follows from Lemma 2.2 and Theorem 6.6. □

As in Karger [7], we estimate $U(p)$ by examining only the smallest cuts. We define α^* to be the minimal value of α such that

$$U(p)(1-\epsilon) \leq U_{\alpha^*}(p) \leq U(p)$$

The parameter α^* plays a crucial role in the analysis. We can estimate $U(p)$ to the desired relative error by estimating $U_{\alpha^*}(p)$, which we can do by enumerating all the α^*-cuts. This is a relatively small collection, which we can explicitly collect and list. In Theorem 2.5, we showed how to bound α^*.

For the rest of the paper, we define

$$\rho = -\log \epsilon / \log n. \tag{7.1}$$

Although in general we allow ρ to increase arbitrarily, we find that these bounds can become confusing because of the interplay between the asymptotic growth of n and ϵ. One can get most of the intuition behind these results by restricting attention to the case $\rho = O(1)$.

Using our improved bounds on the Contraction Algorithm, we can tighten Theorem 2.5, the key theorem of [7]:

LEMMA 7.2. *Suppose $\delta \geq \delta_0 > 0$. Then we have*

$$\alpha^* \leq \frac{(3-\beta+\delta)^2(2-\beta+\delta+\rho)}{(2-\beta+\delta)^2(2+\delta)} + O((\log\log n)/\log n).$$

Proof. As shown in Proposition 7.1, the absolute error introduced by ignoring cuts of weight $\geq \alpha c$ is $U(p) - U_\alpha(p) = n^{\alpha(v-2-\delta)} \log^{O(\alpha)} n$.

The relative error is then at most

$$\begin{aligned} \text{Rel Err} &= O(n^{\alpha(v-2-\delta)} \log^{O(\alpha)} n / U(p)) \\ &= O(n^{\alpha(v-2-\delta)} \log^{O(\alpha)} n / \bar{Z}) \qquad \text{by Proposition 3.3} \\ &= n^{\alpha(v-2-\delta)} n^{2+\delta-\beta} \log^{O(\alpha)} n \end{aligned}$$

For

$$\alpha = \frac{-\log\epsilon + (2-\beta+\delta)\log n + O(\log\log n)}{(2+\delta-v)\log n},$$

the relative error is at most ϵ. Hence,

$$\alpha^* \leq \frac{(3-\beta+\delta)^2(2-\beta+\delta+\rho)}{(2-\beta+\delta)^2(2+\delta)} + O((\log\log n)/\log n).$$

□

It is tempting to simply modify Karger's original algorithm, using this improved estimate for α^* in place of Karger's estimate. We cannot do this directly, as this estimate depends on the parameter β which we cannot simply compute. It is possible to eliminate this parameter, obtaining a usable and tighter bound than Karger's:

COROLLARY 7.3. *Suppose $U(p) < n^{-K}$, for some constant $K > 2$. Then we have*

$$\alpha^* \leq \frac{(K+1)^2(K+\rho)}{K^3} + o(1)$$

Proof. Note that we have $U(p) \geq p^c = n^{-2-\delta}$, and $U(p) = \Theta(\bar{Z}) = \Theta(n^{\beta-2-\delta})$. Hence $\delta + 2 \geq K$ and $\delta + 2 - \beta \geq K + t$, where $|t| = O(1/\log n)$.

Suppose we have $\delta = K - 2 + \beta$ exactly. Then

$$\alpha^* \leq \frac{(K+1)^2(K+\rho)}{K^2(\beta+K)} + o(1) \leq \frac{(K+1)^2(K+\rho)}{K^3} + o(1)$$

The bound on α^* is a differentiable function of β. When we perturb this by adding t to β, this adds a further error of $o(1)$. □

Using that estimate, one could set $K = 2.73$ and, using Karger's analysis otherwise, immediately obtain a total running time of $n^{3.73}\epsilon^{-O(1)}$. This is good, but we can do better.

To explain our approach intuitively, consider the following straw-man algorithm. Suppose we run $n^{\alpha^* v_{\beta,\delta}+o(1)}\epsilon^{-o(1)}$ independent executions of the Contraction Algorithm, selecting some $\alpha^*_{\max}$-cut each time, for

$$\alpha^*_{\max} = \frac{(K+1)^2(K+\rho)}{K^3} + o(1) = O(1+\rho)$$

This will give us a large collection A of cuts, of various sizes. Although we will not know an exact bound for α^*, which would depend on knowing β, it is not hard to see that, with high probability, A will contain all the α^*-cuts with high probability. We can use the collection A to estimate $U(p)$.

Using this approach, it will suffice to provide an upper bound on the product $\alpha^* v_{\beta,\delta}$ irrespective of β, without bounding either term individually. This will let us determine how many samples of the Contraction Algorithm are needed.

Proposition 7.4. *Suppose $U(p) < n^{-K}$ for some constant $K > 2$. Then*

$$\alpha^* v_{\beta,\delta} \leq 2 + \frac{1}{K} + \frac{2K+1}{K^2}\rho + O(\log\log n/\log n)$$

Proof. Note that we have $U(p) \geq p^c = n^{-2-\delta}$, and $U(p) = \Theta(\bar{Z}) = \Theta(n^{\beta-2-\delta})$. Hence $\delta + 2 \geq K$ and $\delta + 2 - \beta \geq K + t$, where $|t| = O(1/\log n)$.

Suppose we have $\delta = K - 2 + \beta$ exactly. Then

$$\begin{aligned}\alpha^* v &= \frac{K + \rho + O(\log\log n/\log n)}{K + \beta - v_{\beta,K-2+\beta}} v_{\beta,K-2+\beta} \\ &= (2 + 1/K) + \frac{2K+1}{K^2}\rho + O(\log\log n/\log n)\end{aligned}$$

The bound on $\alpha^* v$ is a differentiable function of β. When we perturb this by adding t to β, this adds a further error of $O(1/\log n)$. □

If we use this strawman algorithm directly, then we must run completely separate instances of the Contraction Algorithm for each sample. Each execution of the Contraction Algorithm takes time $O(n^2)$ (to process the graph), and so the total work would be $\approx n^{\alpha^* v+2}$.

In [10], an efficient algorithm called the *Recursive Contraction Algorithm* was introduced for running multiple samples of the Contraction Algorithm simultaneously. This amortizes the work of processing the graph across the multiple iterations, effectively reducing the time for a single execution of the Contraction Algorithm from $O(n^2)$ to $O(1)$.

In the next section, we will discuss how to combine the Recursive Contraction Algorithm with our improved analysis of the Contraction Algorithm. Unlike in [10], we will not be able to show that a single application of the Recursive Contraction Algorithm is as powerful as $O(n^2)$ independent applications of the Contraction Algorithm. We will still show it is powerful enough to substantially reduce the cost of multiple applications of the Contraction Algorithm. The next section will examine the Recursive Contraction Algorithm in detail.

8. The Recursive Contraction Algorithm.

The basic method of finding the α^*-cuts is to run the Contraction Algorithm for many iterations and collect all the resulting minimal cuts that are found. Each iteration requires $O(n^2)$ work (to process the entire graph). As described in [10], this data-processing can be amortized across the multiple iterations. The resulting algorithm, called the Recursive Contraction Algorithm (RCA), can enumerate all the minimal α-cuts in time $O(n^{2\alpha} \log^2 n)$.

Note that as each cut may be of size $\Omega(n)$, merely outputting all the cuts could take time $\Omega(n^{2\alpha+1})$. The reason that the Recursive Contraction Algorithm is able to run faster than this is that we only need to perform a few simple operations for each cut. In practice, the RCA is quite flexible, and will easily accommodate the relatively simple operations we will need to perform such as hashing, counting, and sampling in time $n^{o(1)}\epsilon^{-o(1)}$. See Appendix A for more details.

We will run the RCA for a large, fixed number of iterations. This will produce a large collection of cuts, of various sizes. We will then show that the resulting collection, regardless of β, will contain all the α^*-cuts with high probability. It may also contain some cuts of size larger than $\alpha^* c$. Note that we will still not necessarily know the exact value of α^* or β.

We define the RCA with parameter $\alpha_{\max}$ as follows.

```
1. If the graph $G$ has fewer than $2\alpha_{\max}$ vertices, output a random cut of
      $G$.
2. Otherwise, run the following step twice:
3. Contract randomly edges of $G$ until the resulting graph $G'$ has
      $\lceil n/\sqrt{2} \rceil$ edges. Run the Recursive Contraction Algorithm with
      parameters $\alpha_{\max}$ on $G'$.
```

As shown in [10], the RCA executes in time $O(n^2 \log n)$. The key is show that it succeeds with good probability.

We can view the Recursive Contraction Algorithm as a binary tree, of height $2 \log_2 \frac{n}{2\alpha}$. Each node of height i corresponds to a graph G' with n_i vertices. We say a node if the branching tree *succeeds* iff the corresponding graph G' contains C. Note that if a leaf node succeeds, then the Recursive Contraction Algorithm will find the cut C with probability $\exp(-\alpha_{\max})$. So it suffices to calculate the probability that some leaf node succeeds. In the ideal case (if the leaves corresponded to independent executions of the Contraction Algorithm), then then probability of a successful leaf node is nearly equal to the expected number of leaf nodes. In the analysis of [10], which used weaker bounds for the Contraction Algorithm, this was indeed the case.

The following Lemma 8.1 is technically difficult in this paper, and we want to explain the intuition behind it first. We have some lower bounds on the probability of retaining the cut C through several rounds of the Contraction Algorithm. The best of these bounds depend on the behavior of certain random variables, most notably the size of $\bar{Z}$, through this process. As the subgraphs produced during the RCA are related to each other, these random variables can be dependent in potentially complicated ways. However, we also have *unconditional* bounds on the variable S_i (i.e. not a bound on its expectation, but a bound on its maximum possible value). This means that there is always a certain low-level probability that the RCA succeeds from any intermediate state.

In addition, there is a global probability that a given branch of the RCA succeeds, due to the expected change of the random variables. These could come in large clusters, which leads to a lower success probability than if each branch succeeded independently. The presence of this extra "state" is what makes our analysis much more difficult than a standard percolation argument such as the original paper of [10] introducing the RCA.

However, this clustering behavior is not necessarily harmful. Suppose we know that half-way through the RCA (in the language of the proof of Lemma 8.1, up to the point at which subgraphs contain n^x nodes), then the successful stories came in clusters of size G. Suppose we also knew that any successful node goes onto a successful leaf node with probability q. Suppose further that $Gq \leq 1/2$. In this case, if the *expected* number of nodes at this half level is T, then the the probability of a succesful leaf node is simply Tq. In other words, we can still use our information about the expected number of surviving nodes to determine the probability of a successful leaf node.

There are two types of clustering which can arise during the RCA. The first type comes from situations in which the random variable S_{n^x} is spread away from its mean, or its mean is smaller than our upper bounds on it would imply. This type of clustering is more than paid for by an increased mean number of successes, so it is actually beneficial for us.

The second type of clustering is harder to deal with. It is possible that the random variables S_{n^x} is equal to its mean, but that S_i is very large for $i > n^x$. In this case, there would be a "choke-point", at which there is a very small probability that the branching process reaches a level with n^y vertices, but when it does so there is an increased number of survivors from that point onward. This is the hardest type of clustering to control.

There are several other complications to overcome in Lemma 8.1. We ne need to show that the random variables S behave deterministically, in the sense that the worst case is when they are equal to their means $\mathbf{E}[S]$. The proof is based on inclusion-exclusion; paradoxically, inclusion-exclusion estimates can get worse when the expected number of successes becomes too large. So must also ensure that the inclusion-exclusion estimates stay "in range."

We make several assumptions on the size of α in this proof; when α is out of these bounds, then much simpler algorithms can be used.

LEMMA 8.1. *Suppose $1.49 < \alpha \leq \alpha_{max} = O(1)$, and we have a target α-cut C.*
Suppose x, y are real numbers in with $0 \leq x \leq y \leq 1$ and satisfying

$$h(\beta, \delta, y) = \bar{h}(\beta, \delta, x)$$
$$x = y/\alpha + \log(2\phi)/\log n$$

where ϕ is a constant (which will be specified in the proof).

Then the RCA finds the cut C with probability at least

$$\mathbf{P}(\textit{RCA succeeds}) \geq n^{2-\alpha h(\beta,\delta,y)-2y-o(1)}$$

Proof. We incur a probability hit of $2^{1-\alpha_{\max}} = O(1)$ that we select the cut C once we have a subgraph with 2α vertices; henceforth we will just keep track of the probability of retaining C up to this point.

When n is large, then the graph changes little between the branching steps of the RCA (i.e. between n_i and n_{i+1}). To keep the discussion simple, we will ignore any quantization issues due to the fact that i is restricted to be integral, and due to the fact that n_{i+1} is rounded from $n_i\sqrt{2}$. Henceforth we will simply say that $n_i = 2^{-i/2}$ exactly and that i is not restricted to be integral. In this vein, for instance, there are n^{2-2l} nodes whose subgraph contains n^l vertices, for $l \in [0, 1]$.

We will divide the probability that the RCA succeeds into two parts. First, we count the number of surviving node with n^x vertices, for some parameter $x \in [0,1]$ to be specified. (That is, the RCA survives to level $i = 2(1-x)\log_2 n$). We refer to each such node as a *middle node*. Next, each middle node survives with probability $\geq Q_x$, independently of each other. Using arguments from [10], one can see easily that for $\alpha \geq 1 + \Omega(1)$ we have

$$Q_x = \Omega(n^{x(2-2\alpha)})$$

To count the number of surviving middle nodes, we use the Inclusion-Exclusion principle: this probability as at least $\mu - \Delta$, where μ is the expected number of surviving middle nodes and Δ is the expected number of pairs of them. One common problem with inclusion-exclusion is that it can sometimes can give worse estimates (or even useless estimates) when the number of survivors becomes too large. For this reason we can attenuate the number of survivors, by independently applying some additional probabilistic check to each pontential survivor. There are two ways we do this. First, we will assume that each middle node goes onto a successful leaf node with probability *exactly* Q_x (not just lower bounded by Q_x).

The second attenuation factor is more subtle. By Lemma 4.1, the probability that a given middle node survives is $e^{-O(\alpha)}\mathbf{E}[e^{-\alpha S_x}]$. When S becomes to small, then this probability becomes too large, causing Inclusion-Exclusion to go "out of range." Thus, if $S_{n^x} \leq s_0$, where s_0 will be specified later, we will choose to either discard the middle node, or to retain it with probability $e^{-\alpha(S_{n^x} - s_0)}$. In this way, the probability that a given middle node survives is

$$\mathbf{P}(\text{Middle node survives}) \geq \Omega(\mathbf{E}[e^{-\alpha \tilde{S}}])$$

where we define the random variable $\tilde{S} = \max(S, s_0)$. Recall where we have $\alpha \leq O(1)$, so the constant term in this notation may depend on α. (We will not remark on this henceforth in the proof.)

We are now ready to use inclusion-exclusion. The mean is simply

$$\mu \geq n^{2-2x} Q_x \cdot \Omega(\mathbf{E}[\exp(-\alpha \tilde{S})])$$

Let us consider a pair of leaf nodes with common ancestor at level $i = 2(1 - z)\log_2 n$. These leaf nodes correspond to two executions of the Contraction Algorithm, with two sequences of graphs $G_n, \dots, G_1$ and $G'_n, \dots, G'_1$. For $j > n_i$, we have $G_j = G'_j$ and the event that the cut survives is the same for both graphs; for $j' \leq n_i$ it is independent given the number of edges in the two graphs. If the pair both survive, then they correspond to valid executions through the contraction process. (The number of edges in graphs G'_j and G_j are themselves very much dependent.)

For $z > y$ we estimate the probability that both middle nodes survive as

$$\begin{aligned}
\mathbf{P}(\text{both leaves survive}) = \mathbf{P}&(\text{retain both nodes}) \\
&\mathbf{E}\Big[\prod_{j=n_i+1}^{n}(1 - \frac{\alpha c}{M_j}) \prod_{j=n^x}^{n_i}(1 - \frac{\alpha c}{M_j}) \prod_{j=n^x}^{n_i}(1 - \frac{\alpha c}{M'_j})\Big] \\
&\leq \mathbf{E}\exp(-\alpha(\tilde{S} + \tilde{S}' - S'_{n_i}))]Q_x^2 \\
&\leq \exp(\alpha h(\beta, \delta, y)\log n)\mathbf{E}[\exp(-\alpha(\tilde{S} + \tilde{S}'))]Q_x^2 \\
&\qquad \text{by Proposition 6.1}
\end{aligned}$$

$$\leq n^{h(\beta,\delta,z)}\mathbf{E}[\exp(-2\alpha\tilde{S})]Q_x^2 \qquad \text{by Cauchy-Schwarz}$$

For $z \leq y$ we use a simpler estimate, in which we ignore the second element of the pair altogether:

$$\begin{aligned}
\mathbf{P}(\text{both leaves survive}) &= \mathbf{P}(\text{retain both nodes}) \\
&\quad \mathbf{E}\Big[\prod_{j=n_i+1}^{n}(1-\frac{\alpha c}{M_j})\prod_{j=n^x}^{n_i}(1-\frac{\alpha c}{M_j})\prod_{j=n^x}^{n_i}(1-\frac{\alpha c}{M'_j})\Big] \\
&\leq \mathbf{P}(\text{retain both nodes})\mathbf{E}\Big[\prod_{j=n_i+1}^{n}(1-\frac{\alpha c}{M_j})\prod_{j=n^x}^{n_i}(1-\frac{\alpha c}{M_j})\Big] \\
&\leq \mathbf{E}[\exp(-\alpha\tilde{S})]Q_x^2
\end{aligned}$$

There are $n^{4-4x}/2$ pairs of middle nodes, of which $\frac{n^{4-4x}}{4n^{2-2z}}$ share a common ancestor with n^z vertices. Thus:

$$\begin{aligned}
\Delta &\leq Q_x^2(\sum_{i=0}^{2(1-y)\log_2 n}\frac{n^{4-4x}}{4n^{2-2z}}n^{h(\beta,\delta,z)}\mathbf{E}[\exp(-2\alpha\tilde{S})] \\
&\quad + \sum_{i=2(1-y)\log_2 n}^{2(1-x)\log_2 n}\frac{n^{4-4x}}{4n^{2-2z}}\mathbf{E}[\exp(-\alpha\tilde{S})]) \\
&\leq Q_x^2(\int_{z=y}^{1}\frac{n^{4-4x}}{4n^{2-2z}}n^{\alpha h(\beta,\delta,z)}\mathbf{E}[\exp(-2\alpha\tilde{S})]\cdot(\log_2 n)^{-1}dz \\
&\quad + \int_{z=x}^{y}\frac{n^{4-4x}}{4n^{2-2z}}\mathbf{E}[\exp(-\alpha\tilde{S})]\cdot(\log_2 n)^{-1})dz
\end{aligned}$$

Simple analysis shows that for n sufficiently large, and $\alpha > 1.49$, we have $y < 0.7$. This further implies that the derivative of the term $\alpha h(\beta,\delta,z)+2z$ with respect to z is $-\Omega(1)$, which implies the integrand $\frac{n^{4-4x}}{4n^{2-2z}}n^{\alpha h(\beta,\delta,z)}$ is decreasing faster than exponentially in z. Hence this integral can be estimated:

$$\Delta \leq Q_x^2(n^{\alpha h(\beta,\delta,y)}n^{4-4x-2+2y}\mathbf{E}[\exp(-2\alpha\tilde{S})]+n^{4-4x+2-2y}\mathbf{E}[\exp(-\alpha\tilde{S})])$$

For the remainder of this proof, we write $h = h(\beta,\delta,y) = \bar{h}(\beta,\delta,x)$. By the Inclusion-Exclusion principle, the probability the RCA succeeds is at least

$$\Omega(\mathbf{E}[\exp(-\alpha\tilde{S})Q_x n^{2-2x} - \phi Q_x^2 n^{2-4x+2y}(n^{\alpha h}\exp(-2\alpha\tilde{S})+\exp(-\alpha\tilde{S}))] \qquad (8.1)$$

where $\phi > 0$ is some constant.

At this point, we would hope that S is deterministic, rather than a random variable. By Jensen's inequality, this would be the worst case if (8.1) if were increasing concave-down in S throughout its full domain. Although this is not the completely the case, it is close enough that a modified form Jensen's inequality will apply.

Define $M = n^{2-2x}Q_x$ and define the function

$$f(s) = M\exp(-\alpha s) - M^2\phi n^{2y-2}(n^{\alpha h}\exp(-2\alpha s)+\exp(-\alpha s))$$

So we have that $\mathbf{P}(\text{RCA succeeds}) \geq \Omega(\mathbf{E}[f(\tilde{S})])$.

Now note that by our assumption on x we have $Mn^{2y}\phi = n^2/2$. Now set

$$s_0 = \alpha^{-1} \log 2 + h \log n$$
$$s_1 = \alpha^{-1} \log 4 + h \log n$$

For $s \geq s_0$ the function f is positive and decreasing. It is concave-down for $s_0 \leq s \leq s_1$, and concave-up for $s \geq s_1$. Because of this concavity, for a given value of $\mathbf{E}[S]$, the expectation $\mathbf{E}[f(\tilde{S})]$ is minimized with the following distribution: there is an atom at $S = 0$ and an atom at $S = s$ for some $s \geq \max(s_1, \mathbf{E}[S])$. This yields

$$\mathbf{P}(\text{RCA succeeds}) \geq (1 - \frac{\mathbf{E}[S]}{s}) f(s_0) + \frac{\mathbf{E}[S]}{s} f(s) \tag{8.2}$$

The derivative of this expression with respect to s is the product of three terms:

- $e^{-2\alpha s}\mathbf{E}[S] n^{2-\alpha h-2y} 2^{-4} \phi^{-1} s^{-2}$; and
- $e^{\alpha s} - 2n^{\alpha h}$; and
- $e^{\alpha s} - 2n^{\alpha h}(1 + 2\alpha s)$

The first two terms are positive for all $s \geq s_1$. The third term has a unique root, at $s_2 = h \log n + \Theta(\log \log n)$.

Now, suppose first that $\mathbf{E}[S] \geq s_2$. Then (8.2) is minimized at $s = \mathbf{E}[S]$ and so

$$\begin{aligned}
\mathbf{P}(\text{RCA succeeds}) &\geq f(\mathbf{E}[S]) \\
&\geq f(h \log n + O(\log \log n)) \\
&\qquad \text{as } f(s) \text{ is decreasing for } s \geq f(\mathbf{E}[S]) \\
&\qquad \text{ and by Proposition 6.6 we have } \mathbf{E}[S] \leq h \log n + O(\log \log n) \\
&\geq n^{2-\alpha h-2y-o(1)}
\end{aligned}$$

Similarly, if $\mathbf{E}[S] \leq s_2$ we have

$$\mathbf{P}(\text{RCA succeeds}) \geq (1 - \frac{\mathbf{E}[S]}{s_2}) f(s_0) + \frac{\mathbf{E}[S]}{s_2} f(s_2)$$

and now note that both $s_0, s_2 = h \log n + \Theta(\log \log n)$, and verify that $f(h \log n + \Theta(\log \log n)) = n^{2-\alpha h-2y-o(1)}$. □

We now put all of our analysis of the RCA together, also removing some of the side conditions on the magnitude of α:

THEOREM 8.2. *Suppose that $U(p) \leq n^{-K}$ for some constant $K \geq 2$. Then there is an algorithm to accumulate all α^*-cuts with high probability in time $n^{3+o(1)}\epsilon^{-1.99}$.*

(Note: better exponents than $3, 1.99$ can be shown for n, ϵ respectively; but this requires more careful calculations.)

Proof. First, suppose that ρ is larger than some sufficiently large constant. In this case, we do not use the RCA; rather, we use the naive strategy of independent repetitions of the Contraction Algorithm. By Proposition 7.4, this requires $n^{\alpha^* v+o(1)} \leq n^{2+1/K+o(1)}\epsilon^{-(2K+1)/K^2}$ trials, with a total running time of $n^{\alpha^* v+o(1)} \leq n^{4+1/K+o(1)}\epsilon^{-(2K+1)/K^2}$. For $K \geq 2$, this is a time of $\leq n^{4.5}\epsilon^{-1.25}$; for ρ sufficiently large this is indeed less than $n^3\epsilon^{-1.99}$.

Next, suppose that $\alpha^* \leq 3/2$. In this case, we use Karger's analysis of the RCA; this requires time $n^{2\alpha+o(1)} = n^{3+o(1)}$.

Next, suppose $\beta \geq 1$. In this case, we use Karger's analysis of the RCA; the total time required is $n^{2\alpha^*+o(1)}$. As $U(p) \leq n^{-K}$ we have $\delta \geq \delta_0$ for some $\delta_0 > 0$, and $\delta \geq \beta - O(1/\log n)$. Subject to these conditions, it is easy to see that

$$2\alpha^* \leq 3 + 3/2\rho + o(1)$$

Finally, we come to the heart of the proof: the case when $\rho = O(1), \alpha^* \geq 3/2, \beta \leq 1$. As $U(p) \leq n^{-K}$ we have $\delta \geq \delta_0$ for some $\delta_0 > 0$. We can use the RCA and the analysis of Lemma 8.1, as follows.

The success probability of the RCA, according to Lemma 8.1, will be $n^{2-\alpha^* h(\beta,\delta,y)-2y-o(1)}$. As each application of the RCA requires time $n^{2+o(1)}$, it will suffice to show that there are $0 \leq x \leq y \leq 1$ such that

$$-2 + \alpha^* h(\beta, \delta, y) + 2y \leq 1 + 1.99\rho + o(1) \tag{8.3}$$

for all $\delta \geq 0, \delta \geq \beta - O(1/\log n), x = y/\alpha + \phi \log^{-1} n, h(\beta, \delta, y) = \bar{h}(\beta, \delta, x), \alpha = \alpha^* \leq \frac{(3-\beta+\delta)^2(2-\beta+\delta+\rho)}{(2-\beta+\delta)^2(2+\delta)} \leq \alpha^* + o(1), \rho \geq -1/\log n$.

The proof of this fact is not complicated conceptually — we first show that the derivative of (8.3) with respect to ρ is at most $3/2$, and we then show that when $\rho = 0$ that the LHS of (8.3) is maximized at $\beta = \delta = 0$, and that the equation is satisfied there. However, there is some quite involved and tricky numerical analysis necessary to show these facts. For sake of clarity, these are deferred to Appendix B. □

8.1. Further applications of the Recursive Contraction Algorithm. The Recursive Contraction Algorithm provides a powerful approach to enumerate all the approximately-minimum cuts in a given graph G. Our reliability-estimation algorithm uses it in a very specific way based on bounds for the number of cuts of G. However, we can slightly improve the analysis of the Recursive Contraction Algorithm in other situations. Given a particular value of α and a given graph G (with no information about the cut structure or reliability of G), how to enumerate the α-cuts of G. The Recursive Contraction Algorithm was first introduced in [10] to solve this problem. However, the original description of the RCA was not parameterized in the best way, leading to slightly sub-optimal running time. We can improve this analysis as follows.

THEOREM 8.3. *There is an algorithm to enumerate, with high probability, all α-cuts of G in time $O(n^{2\alpha} \log n)$.*

In contrast, the algorithm of [10] requires time $O(n^{2\alpha} \log^2 n)$.

Proof. First, suppose $\alpha < 3/2$. Then [9] describes a data structure to represent all the α-cuts in time $O(n^2)$ and to enumerate all the α-cuts in time $O(n^2 \log n)$.

Finally, suppose $3/2 \leq \alpha \leq \sqrt{n}$. In this case, let us set $2 < t < 3$ to be some arbitrary constant and we execute the following variant of the Recursive Contraction Algorithm:

```
1. If the graph $G$ has fewer than $2\alpha$ vertices, output a random cut of
      $G$.
2. Otherwise, run the following step twice:
3. Contract randomly edges of $G$ until the resulting graph $G'$ has
      $\lceil n2^{-1/t} \rceil$ edges.  Run this modified Recursive Contraction
      Algorithm on $G'$.
```

A simple application of the Master Theorem for recurrences along with arguments from [10] shows that this algorithm executes in time $O(n^t)$. Also using arguments from [10], we can show that for any α-cut C, and any execution of this algorithm, the cut is selected with probability $n^{t-2\alpha} \exp(\Omega(\alpha \log \alpha))$.

Thus we can view the process of finding all α-cuts as a Coupon Collector Problem. As there are at most $n^{2\alpha}$ cuts, if we run $(2\alpha \log n) \times n^{2\alpha - t} \exp(-\alpha \log \alpha) = n^{2\alpha - t} \log n$ independent executions of the RCA, then we find all cuts with high probability. This gives a total run time of $O(n^{2\alpha} \log n)$.
□

9. Estimating failure probability from the α^*-cuts. In the previous section, we showed how one can find a set A of cuts which contains the α^*-cuts with high probability, where α^* is bounded by $\alpha^* \leq \alpha^*_{\max} = O(1 + \rho)$. We can then use this sample to estimate $U(p)$ itself.

Define $U_A(p)$ to be the probability that some cut from A fails when edges are removed independently with probability p. If A contains all the α^*-cuts, then we have $(1 - \epsilon)U(p) \leq U_A(p) \leq U(p)$. So it will suffice to estimate $U_A(p)$.

Karger's analysis uses an elegant algorithm developed by Karp, Luby, Madras [12] for this problem. However, as a starting point for our algorithm, it is useful to consider the simpler generic statistic to estimate $U_A(p)$;

```
1. Select a cut $C_0$ from the collection $A$.  The probability of
   selecting cut $C_0 = C$ is $\frac{p^C}{\sum_{C' \in A} p^{C'}}$.
2. Let $L$ be a random subset of the edges in $G - C_0$, in which each
   such edge is chosen independently with probability $p$.
3. Count $J$, the number of cuts in $A$ which contain $L \cup C_0$.
4. Estimate $\hat{U}(p) = \frac{\sum_{C' \in A} p^{|C'|}}{J}$
```

The main cost of this algorithm is the step in which we must count the cuts in A containing $L \cup C_0$. This requires, at the minimum, reading all $|A|$ cuts, for a total work factor of $|A|$ per iteration. This algorithm treats the cuts as if they were clauses in a DNF formula. As such, it ignores the graph-theoretic structure of these cuts. By keeping track of the graph structure we can count J much more quickly than by testing each cut individually.

The basic intuition is that, after we remove $C_0 \cup L$ from the original graph G, we define the graph G' by contracting all the connnected components of $G - C_0 - L$. Then we can determine J by counting the cuts of the small graph G'. The number of such cuts, and the work needed to find them, will be small compared to n.

Consider the following algorithm, which we refer to as the *estimation algorithm for A*:

```
0. Precompute data structures corresponding to the set $A$.
1. Select a cut $C_0$ from $A$ with probability $\propto p^{|C|}$.
2. Let $L$ be a random subset of the edges in $G - C_0$, in which each
   such edge is chosen independently with probability $p$.  Let $H =
   G - C_0 - L$.
3. Enumerate the connected component structure of $H$.  Let $G'$ be
   the graph obtained from $G$ by contracting all components of $H$.
4. Enumerate all cuts of $G'$.
5. For each such cut $C'$ of $G'$, test if $C' \in A$.  Let $J$ denote the
   number of such cuts.
6. Estimate $\hat{U}(p) = \frac{\sum_{C' \in A} p^{|C'|}}{J}$
```

We will first bound the running time of the estimation algorithm.

PROPOSITION 9.1. *Suppose $\delta \geq \delta_0 > 0$. Then the expected running time of the estimation algorithm for A is $O(n^2)$.*

Proof. Step (3) requires decomposing the graph G into its connected components. This can be done via depth-first search in time $O(n^2)$.

Let us now examine step (4), which is the only step that can potentially take $\omega(n^2)$ time. We enumerate the cuts C' of G' and test them against the set A. Testing whether a given cut $C' \in A$, using a binary search, will cost $O(n^2)$ (see Appendix A). Let R denote the number of connected components of the graph $G - C_0 - L$. Then step (4) can take time $2^R n^2$, as the graph G' has R vertices and hence at most 2^R cuts.

We will show a bound on $\mathbf{E}[2^R]$. We first consider conditioning on selecting a fixed α-cut C. Note that, conditioning on a fixed selection of $C_0 = C$ is equivalent to conditioning on the event that C has failed. Hence we have

$$\begin{aligned}
\mathbf{E}[2^R \mid C_0 = C] &\leq 1 + \sum_{r \geq 1} \mathbf{P}(H \text{ has} \geq r \text{ connected components} \mid C \text{ fails})(2^r - 2^{r-1}) \\
&\leq 1 + \sum_{r \geq 1} 2^{r-1} \min(1, \mathbf{P}(H \text{ has} \geq r \text{ connected components})(p^c)^{-\alpha}) \\
&\leq 1 + \sum_{r \geq 1} 2^{r-1} \min(1, (e/r)^r n^{-r\delta/2} n^{(2+\delta)\alpha}) \qquad \text{by Lemma 3.1}
\end{aligned}$$

The cut C is selected with probability $\frac{p^C}{\sum_{C' \in A} p^{C'}}$. As A contains the minimum cut, the denominator is at least $n^{-2-\delta}$. Hence the total contribution of all cuts of weight $\geq x$ is at most

$$\begin{aligned}
\mathbf{E}[2^R \times \big[|C| \geq xc\big]] &\leq \sum_{\alpha = x}^{\infty} n^{2\alpha} \frac{p^{\alpha c}}{n^{-2-\delta}} (1 + \sum_{r \geq 1} 2^{r-1} \min(1, (e/r)^r n^{-r\delta/2} n^{(2+\delta)\alpha})) \\
&\leq \sum_{\alpha = x}^{\infty} n^{2\alpha} p^{\alpha c} n^{2+\delta} O(\sum_{r \geq 1} n^{-r\delta/2} n^{(2+\delta)\alpha}) \\
&\leq \sum_{\alpha = x}^{\infty} n^{2\alpha} p^{\alpha c} n^{2+\delta} O(\alpha n^{2+\delta/2})
\end{aligned}$$

For some $x = O(1)$ the summand is decreasing exponentially, and thus by Lemma 2.2 contributes a total of $O(1)$.

Next, suppose C is an α-cut for some $\alpha = O(1)$, then

$$\begin{aligned}
\mathbf{E}[2^R \mid C_0 = C] &\leq 1 + \sum_{r \geq 1} 2^{r-1} \min(1, (e/r)^r n^{-r\delta/2} n^{(2+\delta)\alpha}) \\
&\leq 1 + \sum_{r \leq 2\alpha \frac{2+\delta}{\delta}} 2^{r-1} + \sum_{r \geq 2\alpha \frac{2+\delta}{\delta}} n^{-r\delta/2} n^{(2+\delta)\alpha} \\
&\leq 1 + O(1) + \frac{2}{\delta \log n} = O(1).
\end{aligned}$$

In either case, the contribution to the expectation $\mathbf{E}[2^R]$ is at most $O(1)$. □

The final piece of the puzzle is to prove that the estimation algorithm has good accuracy for estimating $U_A(p)$. The key difference between the estimation algorithm and the algorithm of [12] is that, in the latter, the number of samples may become as large as $\Omega(|A|)$ where our estimation algorithm uses a single sample. The algorithm of [12] automatically adjusts so that either few samples are needed, *or* samples can be generated quickly; thus the total amount of work is guaranteed to remain low. But we will never need to use many samples:

PROPOSITION 9.2. *Suppose $\delta \geq \delta_0 > 0$. Then the relative variance of the estimation algorithm is $O(1)$.*

Proof. We claim that there is some $t = O(1)$ and some constant $\phi < 1$ such that we have $\mathbf{P}(J \geq t) \leq \phi$. This will suffice to show that $\mathbf{E}[J^{-1}] = \Omega(1)$.

We first show that there is a probability $\Omega(1)$ of selecting an α-cut, for some $\alpha = O(1)$. For, the total probability of selecting a cut of weight greater than α is

$$\begin{aligned}
\mathbf{P}(|C_0| \geq \alpha c) &= \frac{\sum_{C \in A, |C| \geq \alpha c} p^c}{\sum_{C \in A} p^c} \\
&\leq \frac{\sum_{C \in A, |C| \geq \alpha c} p^{|C|}}{n^{-2-\delta}} \\
&= o(1) \qquad \text{for } \alpha \text{ sufficiently large constant, by Lemma 2.2}
\end{aligned}$$

Conditioned on $C_0 = C$, the random variable J is distributed as the number of failed cuts in G, conditioned on C failing. Suppose that C is an α-cut for $\alpha = O(1)$. Then we have

$$\begin{aligned}
\mathbf{P}(J \geq t \mid C_0 = C) &\leq P[Z \geq t \mid C_0 = C] \\
&\leq n^{\alpha(2+\delta)} n^{-\delta \log_2 t} \qquad \text{by Lemma 3.2} \\
&\leq o(1) \qquad \text{for } t \text{ sufficiently large constant}
\end{aligned}$$

Hence $\mathbf{E}[J^{-1}] \geq \phi/t \geq \Omega(1)$. As $J \geq 1$, this immediately shows that $\mathbf{V}[J^{-1}]/\mathbf{E}[J^{-1}]^2 = O(1)$. □

Putting all this together, we have the following:

PROPOSITION 9.3. *Suppose $\delta \geq \delta_0 > 0$, and let A be a set consisting of cuts including a minimum-weight cut. Then there is an algorithm for estimating $U_A(p)$ to within relative error ϵ with high probability, and running in time $O(n^2 \epsilon^{-2})$.*

Proof. A single iteration of the algorithm has relative error $O(1)$ and costs $O(n^2)$ time. We repeat $\lambda\epsilon^{-2}$ indepedent trials of this algorithm for a suifficiently-large absolute constant λ and extract the sample mean. This reduces the relative variance of the resulting unbiased statistic to $O(1)$. By Chebyshev's inequality this implies that, with high probability, it estimates $U_A(p)$ to relative error ϵ. □

10. Putting it all together.

We may now put all the pieces of the algorithm together. Our basic plan is to use the cut-enumeration if $U(p) < n^{-K}$, and use Monte-Carlo sampling when $U(p) \geq n^{-K}$, where K is some chosen parameter close to 2.

How do we determine which of these two methods to apply? At first, it would appear that this decision requires knowing $U(p)$, which is what we are trying to determine in the first place. A simple way to make this decision is to run a preliminary Monte-Carlo sampling for ϕn^{-K} trials, where $\phi > 1$ is constant. If during any of these samples we observe the graph become disconnected, we will use Monte-Carlo sampling to estimate $U(p)$; otherwise we use cut-enumeration.

This achieves at least a gross discrimination between the two regimes:

PROPOSITION 10.1. *Suppose $U(p) \geq n^{-K}$. Then the probability of observing the graph become disconnected, is bounded from below by a constant which approaches* 1 *for ϕ sufficiently large.*

Suppose $U(p) \leq \phi' n^{-K}$, for ϕ' a constant. Then the probability of observing the graph become disconnected, is bounded from above by a constant which approaches 0 *for ϕ' sufficiently small.*

Proof. As $n \to \infty$, the number of times the graph becomes disconnected approaches a Poisson random variable. In the first case, when $U(p) > n^{-K}$, the expected number of successes is at least ϕ, and so the probability of at least one success approaches $1 - e^{-\phi}$. The proof for the second case is similar. □

Now, when $U(p) \geq n^{-K}$, we will with high constant probability choose Monte-Carlo sampling. This is good because the cut-enumeration algorithm may not be well-behaved when $U(p) \geq n^{-K}$; most of our theorems completely break down in the regime $U(p) \geq n^{-2}$. When $U(p) \leq \phi' n^{-K}$, we will use cut-enumeration with high constant probability. We will show that is the appropriate algorithm in that case. When $\phi' n^{-K} < U(p) < n^{-K}$, we may use either Monte Carlo sampling or cut-enumeration. In this regime, *either* of these two algorithms gives good performance. The Monte-Carlo sampling will have relative variance $O(n^{-K})$. We have already shown that when $U(p) < n^{-K}$ that cut-enumeration behaves correctly.

We now obtain our main result:

Proposition 10.2. *Let $\gamma > 0$ be any constant. Then we can estimate $U(p)$ in time $O(n^{3+\gamma}\epsilon^{-2})$.*

Proof. Let $K = 2 + \gamma/2$.

Suppose that $U(p) < n^{-K}$ and we elect to use the cut-enumeration procedure. As shown in Theorem 8.2 the work to find a collection of cuts A which contains all the α^*-cuts is $O(n^{3+o(1)}\epsilon^{-1.99})$. Next, using the estimation algorithm for A, one can use this collection A to estimate $U_A(p)$ in time $O(n^2\epsilon^{-2})$. By definition of α^*, we have $U_A(p)$ within relative error $O(\epsilon)$ of $U(p)$. Hence this gives us an accurate estimate of $U(p)$ as desired.

Suppose instead that $U(p) \geq \phi' n^{-K}$, where ϕ' is a constant chosen in Proposition 10.1, and welect to use Monte-Carlo sampling. As shown in [7] there is an algorithm based on Monte-Carlo sampling in time $O(n^{K+1}\epsilon^{-2}\log^{O(1)} n)$.

The total work is $O(n^{3+\gamma}\epsilon^{-2})$ either way. By Proposition 10.1, with arbitrarily high constant probability, one of these two cases holds. This implies that we find a good estimate with probability $> 3/4$, which is the goal of our FPRAS.
□

Note that our estimates have been developed under the assumption that δ is uniformly bounded away from 0. This means that we can attain a running time $n^{3+\gamma}$ for any constant $\gamma > 0$. This does not necessarily mean that a single algorithm can attain a running time of $n^{3+o(1)}$, because the hidden constant may blow up as $\gamma \to 0$, possibly faster than any computable function. Such "speed-up" phenomena are possible, but pathological. They do not occur in our case, and we get:

Theorem 10.3. *There is an algorithm to estimate $U(p)$ in time $n^{3+o(1)}\epsilon^{-2}$.*

11. Concluding Remarks.

Two natural open questions are to see if a run-time bound such as $O(n^2/\epsilon^2)$ is possible, and to identify other possible applications of differential-equation approximations such as ours.

It remains an outstanding open question to make progress on the approximability of $R(G, p) = R(p)$, the probability of G remaining *connected*: as pointed out by Leslie Goldberg to us, even very weak approximations here can be turned into PTAS-type approximations, and a proof of this from [4] is as follows. For example, suppose for some $\epsilon > 0$ that there is a polynomial-time algorithm that given as input any connected graph H with n_H vertices, produces a value $\hat{R}(H,p)$ satisfying

$$(1-\epsilon)^{\sqrt{n_H}} R(H,p) \leq \hat{R}(H,p) \leq (1+\epsilon)^{\sqrt{n_H}} R(H,p); \tag{11.1}$$

note that this is a very weak approximation algorithm. However, it can be turned into essentially an $(1 \pm \epsilon)$–approximation algorithm as follows. Take the connected input graph G for which we want to well-approximate $R(G,p)$. Choose an arbitrary vertex v of G, and construct a graph H by taking n_G copies of G, but by also graph-theoretically "identifying" (i.e., fusing together) the copies of vertex v. Then $R(H,p) = R(G,p)^(n_G)$ and $n_H = n_G^2 - n_G + 1 \sim n_G^2$; in particular, $\sqrt{n_H} = n_G - 1/2 + o(1)$. Thus, by (11.1),

$$\begin{aligned} (1-\epsilon)^{\sqrt{n_H}} R(G,p)^{n_G} \quad &\leq \hat{R}(H,p) \leq \quad (1+\epsilon)^{\sqrt{n_H}} R(G,p)^{n_G}, \quad \text{i.e.,} \\ (1-\epsilon)^{1-o(1)} R(G,p) &\leq (\hat{R}(H,p))^{1/n_G} \leq (1+\epsilon)^{1-o(1)} R(G,p); \end{aligned}$$

thus, $(\hat{R}(H,p))^{1/n_G}$ is an excellent approximation to $R(G,p)$. Thus, the approximability of $R(G,p)$ is either excellent, or very poor; determining the truth here is a very intriguing open question.

Acknowledgments. We thank the SODA 2014 referees for their helpful comments, and Leslie Goldberg for a valuable discussion on $R(p)$. The discussion with Leslie Goldberg that is included in Section 11, as well as part of Aravind Srinivasan's writing, transpired while Srinivasan attended the ICERM Workshop on *Stochastic Graph Models* in March 2014. We thank the workshop organizers for their kind invitation.

REFERENCES

[1] Bixby, R.: The minimum number of edges and vertices in a graph with edge connectivity n and m n-bonds. Bulletin of the American Mathematical Society 80, 700-704 (1974).

[2] Chandran, L., Shankar, L.: On the number of minimum cuts in a graph. SIAM Journal of Discrete Mathematics 18, 177-194 (2004).

[3] Dinitz, E., Karzanov, A., Lomonosov, M. On the structure of a family of minimum weighted cuts in a graph. Studies in Discrete Optimization, 290-306 (1976).

[4] Goldberg, L. A., Jerrum, M.: Approximating the Partition Function of the Ferromagnetic Potts Model. Journal of the ACM 59, 25:1–25:31 (2012).

[5] Karger, D.: Random Sampling in Graph Optimization Problems. Ph.D. Thesis, Department of Computer Science, Stanford University, 1994.

[6] Karger, D.: Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, 21-30 (1993).

[7] Karger, D.: A Randomized Fully Polynomial Time Approximation Scheme for the All Terminal Network Reliability Problem. SIAM Journal on Computing 29-2, 492-514 (1999)

[8] Karger, D.: A Randomized Fully Polynomial Time Approximation Scheme for the All Terminal Network Reliability Problem. SIAM Review 43-3, 499-522 (2001)

[9] Karger, D.: Minimum Cuts in Near-Linear Time. Journal of the ACM 47 1, 46-76 (2000).

[10] Karger, D., Stein, C.: A new approach to the minimum cut problem. Journal of the ACM 43, 601-640 (1995).

[11] Karger, D., Tai, P.: Implementing a Fully Polynomial Time Approximation Scheme for All Terminal Network Reliability. Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, 334-343 (1997).

[12] Karp, R. M., Luby, M., Madras, N. Monte-Carlo approximation algorithms for enumeration problems. Journal of Algorithms 10, 429-448 (1989).

[13] Lomonosov, M.: Bernoulli scheme with closure. Problems of Information Transmission 10, 73-81. (1974)

[14] Lomonosov, M., Polesskii, V. P.: Lower bound of network reliability. Problems of Information Transmission 7, 118-123. (1971)

[15] Sharafat, A., Ma'rouzi, O.: All-Terminal network reliability using recursive truncation algorithm. IEEE Transactions on Reliability 58, 338 - 347. (2009)

[16] Wormald, N.: The differential equation method for random graph processes and greedy algorithm. Annals of Applied Probability 5-4, pp. 875 - 1240 (1995).

Appendix A. Data structures for the Recursive Contraction Algorithm. Our algorithm depends on running the Recursive Contraction Algorithm (RCA) for a large number of iterations. At each leaf node of the RCA, a single cut will be produced. In fact, this cut is not generated explicitly, that is, we are not listing out all the vertices in that cut.

Our goal is to enumerate a large collection A of α^*_{max}-cuts, for $\alpha^*_{\text{max}} = O(1+\rho)$. With high probability, this includes all α^*-cuts. Each such cut will likely be generated multiple times, and we will need to remove any duplicates. We will then need to extract certain key information about each such cut for use in the enumeration algorithm.

The most important task we will need to perform is to check, given an arbitrary cut C, whether $C \in A$. During the RCA, this will allow us to remove duplicates from A. During the enumeration algorithm, this is necessary to compute the desired statistics.

A method based on hashing is discussed in [10]. We will need to generalize this method slightly and get slightly better parameters, so we reiterate the method here.

We use the following data structure to accomplish this. To each vertex v, we associate a random b-bit number $w(v)$. We then define the *identifier* for a cut C as the sum modulo 2^b, over all vertices v which are in the same shore as vertex 1, of $w(v)$.

The following proposition is straightforward:

Proposition A.1. *There is some choice of* $b = O(\min(n, \log n + \log \epsilon^{-1}))$ *such that, with high probability, every* α^*_{max}*-cut has a distinct identifier.*

Proof. Any two cuts C, C' have a probability of 2^{-b} of having the same identifier. Now take a union bound over the $(\min(2^n, n^{2\alpha^*_{\text{max}}}))^2$ pairs of cuts. □

During the execution of the RCA, one can maintain the identifier of the current cut fairly readily: whenever we merge two vertices v, v', we add their corresponding identifiers. Furthermore, using a heap data structure, one can maintain the sorted identifiers. Each time a cut is produced, one looks up its identifier in the heap. If it is absent, one adds the cut; otherwise, we skip the current cut as it is likely a duplicate. The total run-time for these lookups is $O((\log n + \log \epsilon^{-1})^2) = n^{o(1)} \epsilon^{-o(1)}$.

During the enumeration algorithm, one must look up cuts as well, however in this analysis our goal is to achieve a run-time of $O(n^2)$. Our bound accomplishes this as well, regardless of ϵ.

Finally, we must select a certain number of cuts for use in the enumeration algorithm. Each cut is selected with probability proportional to $p^{|C|}$. During the RCA, one can maintain the weight of each cut dynamically, so it requires $n^{o(1)}$ time to output $p^{|C|}$. From this it is straightforward to determine $\sum_{C \in A} p^{|C|}$ and to select any number of desired cuts. Note that for those cuts actually *selected* by the enumeration algorithm, we are allowed to spend the time to output the entire cut.

Appendix B. Numerical analysis for Theorem 8.2.

Throughout this section we suppose $\alpha^* \geq 3/2, \rho = O(1), \beta \leq 1$. (Other cases are covered already in Theorem 8.2.) These assumptions will not be stated again. This section depends heavily on the properties of the functions $h, \bar{h}$, as well as our upper bound for α^*; we repeat them here for clarity:

$$h(\beta,\delta,x) = \begin{cases} 2(2+\delta)(\log(3-\beta+\delta) - \log(2-\beta+\delta+x)) & \text{if } x \geq \beta \\ 2(2+\delta)(\log(3-\beta+\delta) - \log(2+\delta)) + 2(\beta - x) & \text{if } x \leq \beta \leq 1 \\ 2(1-x) & \text{if } \beta \geq 1 \end{cases}$$

$$\bar{h}(\beta,\delta,x) = \frac{(\delta+2)(1-x)(5-2\beta+2\delta+x)}{(3-\beta+\delta)^2}$$

$$\alpha^\sharp(\beta,\delta,\rho) = \frac{(3-\beta+\delta)^2(2-\beta+\delta+\rho)}{(2-\beta+\delta)^2(2+\delta)}.$$

Note that $\alpha^* \leq \alpha^\sharp + o(1)$; we distinguish between the quantities $\alpha^\sharp$ and α^*; the former is a smooth function of β,δ,ρ while the latter could have a much more complicated and discontinuous behavior.

The success probability of the RCA, according to Lemma 8.1, will be $n^{2-\alpha^* h(\beta,\delta,y)-2y-o(1)}$. As each application of the RCA requires time $n^{2+o(1)}$, it will suffice to show that there are $0 \leq x \leq y \leq 1$ such that

$$-2 + \alpha h(\beta,\delta,y) + 2y \leq 1 + 1.99\rho + o(1) \qquad \text{(B.1)}$$

for all $\delta \geq 0, \delta \geq \beta - O(1/\log n), x = y/\alpha + \phi \log^{-1} n, h(\beta,\delta,y) = \bar{h}(\beta,\delta,x), \alpha \leq \alpha^\sharp + o(1), \rho \geq -1/\log n$.

We first claim that we can remove many of the small perturburations from these conditions:

Proposition B.1. *Suppose there are* $0 \leq x \leq y \leq 1$ *such that*

$$-2 + \alpha^\sharp(\beta,\delta,\rho)h(\beta,\delta,y) + 2y \leq 1 + 1.99\rho + o(1) \qquad \text{(B.2)}$$

for all $\delta \geq 0, \beta \geq \delta, x = y/\alpha^\sharp(\beta,\delta,\rho), h(\beta,\delta,y) = \bar{h}(\beta,\delta,x), \rho \geq 0$.

Then (B.1) can also be satisfied.

Proof. Note that we can take $\alpha = \alpha^\sharp + o(1)$, as every α^*-cut is necessarily also a $(\alpha^\sharp + o(1)) - cut$.

Let $x = y/\alpha + d$, where $d = o(1)$.

We simply need to show that perturbing the arguments by $o(1)$ only changes the objective function

$$-2 + \alpha^*(\beta,\delta,\rho)h(\beta,\delta,y) + 2y$$

by a factor of $o(1)$ as well.

All of the parameters behave in essentially the same way, so we will show how the objective function changes when d changes from $o(1)$ to 0 — this is the simplest case.

We can use implicit differentiation. When for example d changes, then $\partial x/\partial d$ can be computed as

$$-\frac{\partial[h(\beta,\delta,y) - \bar{h}(\beta,\delta,y/\alpha+d)]/\partial d}{\partial[h(\beta,\delta,y) - \bar{h}(\beta,\delta,y/\alpha+d)]/\partial y}$$

The numerator is easily seen to be $O(1)$. The denominator can be seen to be $\Omega(1)$, for d sufficiently small. Hence when d changes by $o(1)$, then x changes by $o(1)$ as well. It is now routine to verify that the objective function changes by $o(1)$. $\square$

Next, we claim that x, y are uniquely determined to satisfy Proposition B.1:

Proposition B.2. *For any given value of* β, δ, *there is a unique value* $0 \leq x^* \leq y* \leq 1$ *such that* $x^* = y^*/\alpha^\sharp$ *and such that*

$$h(\beta, \delta, y^*) = \bar{h}(\beta, \delta, x^*)$$

Proof. Consider the expression

$$h(\beta, \delta, y) - \bar{h}(\beta, \delta, y/\alpha) \tag{B.3}$$

When $y = 0$, this is positive; when $y = 1$, this is negative.

Next, we claim that the derivative with respect to y is negative. There are two cases. When $y \geq \beta$, this has derivative

$$\partial/\partial y = 2(\delta + 2)\left(\frac{\alpha(-\beta + \delta + 2) + y}{\alpha^2(-\beta + \delta + 3)^2} - \frac{1}{-\beta + \delta + y + 2}\right)$$

which can be seen to be ≤ 0 throughout; a similar calculation reveals this for $y \leq \beta$ □

We can regard x^*, y^* now as functions of β, δ, ρ; they are easily seen to be differentiable. In this vein now, we define the objective function

$$g(\beta, \delta, \rho) = \alpha^\sharp(\beta, \delta, \rho)h(\beta, \delta, y^*) + 2y^* = \alpha^\sharp(\beta, \delta, \rho)(\bar{h}(\beta, \delta, x^*) + 2x^*)$$

and we want to show that $g(\beta, \delta, \rho) \leq 3 + 1.99\rho + o(1)$.

First, we examine how this scales with ρ.

Proposition B.3. *We have the bound:*

$$\frac{\partial g(\beta, \delta, \rho)}{\partial \rho} \leq 3/2$$

Proof. There are two cases, depending on whether $y \leq \beta$. We begin by assuming $y \leq \beta$ and we use implicit differentiation to determine $\partial x^*/\partial \rho$:

$$\begin{aligned}\partial x^*/\partial \rho &= -\frac{\partial[h(\beta, \delta, x^*\alpha^\sharp) - \bar{h}(\beta, \delta, x^*)]/\partial \rho}{\partial[h(\beta, \delta, x^*\alpha^\sharp) - \bar{h}(\beta, \delta, x^*)]/\partial x^*} \\ &= \frac{x^*(3 - \beta + \delta)^4}{(2 - \beta + \delta)^2(\delta + 2)^2(2 - \beta + \delta + x^*) - (3 - \beta + \delta)^4(2 - \beta + \delta + \rho)}\end{aligned}$$

and thus we have

$$\frac{\partial g(\beta, \delta, \rho)}{\partial \rho} = \frac{\partial \alpha^\sharp(\beta, \delta, \rho)}{\partial \rho}(\bar{h}(\beta, \delta, x^*) + 2x^*) + \alpha^\sharp(\beta, \delta, \rho)\frac{\partial[\bar{h}(\beta, \delta, x^*) + 2x^*)}{\partial x^*}\frac{\partial x^*}{\partial \rho}$$

which gives us

$$\begin{aligned}\frac{\partial g(\beta, \delta, \rho)}{\partial \rho} &= \frac{(3 - \beta + \delta)^2}{(\delta + 2)(2 - \beta + \delta)^2} \\ &\quad \times \Big(\frac{2x^*(3 - \beta + \delta)^2(2 - \beta + \delta + \rho)\,(5 - \beta(4 - \beta + \delta) - (\delta + 2)x^* + 2\delta)}{(2 - \beta + \delta)^2(\delta + 2)^2(2 - \beta + \delta + x^*) - (3 - \beta + \delta)^4(2 - \beta + \delta + \rho)} \\ &\quad + \frac{(\delta + 2)(1 - x^*)(-2\beta + 2\delta + x^* + 5)}{(3 - \beta + \delta)^2} + 2x^*\Big)\end{aligned} \tag{B.4}$$

Now simple calculations show that if we take derivative of (B.4) with respect to ρ, *but ignore the dependency of x^* on ρ*, then the resulting quantity is ≥ 0. (That is, when we take the derivative with respect to ρ, we treat x^* formally as a constant term even though it does indeed depend on ρ.) Thus, we can upper bound (B.4) by taking the limit as $\rho \to \infty$ and treating x^* as a constant in this limiting process:

$$\begin{aligned}\frac{\partial g(\beta,\delta,\rho)}{\partial \rho} &\leq \frac{5 - 2\beta + 2\delta + (x^*)^2}{(2-\beta+\delta)^2} \\ &\leq 3/2\end{aligned}$$

A similar process works for the case $y \geq \beta$: we differentiate with respect to ρ, obtaining an expression which is formally a function of x^*, ρ. We then take the limit as $\rho \to \infty$, ignoring the dependency of x^* on ρ. The calculations are somewhat messy and we obtain

$$\frac{\partial g(\beta,\delta,\rho)}{\partial \rho} \leq \phi(x^*, \beta, \delta)$$

where ϕ is a (quite complicated) rational function. Using a symbolic algebra package, one can then verify that we have $\phi(x,\beta,\delta) \leq 1.99$ for all $x \in [0,1], \beta \in [0,1], \delta \geq \beta$. ◻

In light of Proposition B.3, we have essentially of reduced the problem to analyzing the case when $\rho = 0$.

PROPOSITION B.4. *For all $\beta \in [0,1], \delta \geq \beta$ we have*

$$g(\beta,\delta,0) \leq 3$$

Proof. We claim that for any $y \in [0,1]$, if we have

$$h(\beta,\delta,y) \geq \bar{h}(\beta,\delta,y/\alpha^*) \tag{B.5}$$

then this implies that $y^* \geq y$ and furthermore that if $y^* \geq y$ we have

$$g(\beta,\delta,\rho) \leq \alpha^* h(\beta,\delta,y) + 2y \tag{B.6}$$

These facts be seen by differentiating (B.5), (B.6) with respect to y.

Now, suppose $\delta \geq 2.5$. We have $y^* \geq 0$ trivially, hence

$$\begin{aligned}g(\beta,\delta,0) &\leq \alpha^* h(\beta,\delta,0) \\ &\leq 1.49383 \times 2 \leq 3\end{aligned}$$

as desired.

Next, suppose $\delta \leq 2.5$. Substitute $y = 0.313$ into (B.5); we claim that this is satisfied for all β, δ. This is again a simple but tedious exercise. Then we have

$$\begin{aligned}g(\beta,\delta,0) &\leq \alpha^* h(\beta,\delta,0.313) + 0.626 \\ &\leq 2.967 \qquad \text{maximized at } \beta = \delta = 0\end{aligned}$$

◻

Putting all these cases together, we have our desired result:

PROPOSITION B.5. *Suppose $\alpha^* \geq 3/2, \rho = O(1), \beta \leq 1$. there are $0 \leq x \leq y \leq 1$ such that*

$$\alpha^* h(\beta,\delta,y) + 2y \leq 3 + 1.99\rho + o(1)$$

for all $\delta \geq 0, \delta \geq \beta - O(1/\log n), x = y/\alpha^* + \phi \log^{-1} n, h(\beta, \delta, y) = \bar{h}(\beta, \delta, x), \alpha \leq \alpha^\sharp + o(1), \rho \geq -1/\log n$.

Proof. By Propositions B.1, B.2 it suffices to show that

$$g(\beta, \delta, \rho) \leq 3 + 1.99\rho + o(1)$$

where $\delta \geq \beta$, and x^*, y^* are defined by $x^* = y^*/\alpha^\sharp, h(\beta, \delta, y^*) = \bar{h}(\beta, \delta, x^*)$.

Now by Proposition B.3 we have

$$g(\beta, \delta, \rho) \leq g(\beta, \delta, 0) + 1.99\rho$$

and by Proposition B.4 we have $g(\beta, \delta, 0) \leq 3 + o(1)$. □