Improved bounds and algorithms for graph cuts and network reliability

David G. Harris* Aravind Srinivasan†

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 \( \epsilon \).

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 \( n^{3+o(1)} \epsilon^{-2} \), which is essentially best-possible for the meta-approach proposed by Karger; our results also rigorously prove certain experimental observations of Karger & Tai (Proc. ACM-SIAM Symposium on Discrete Algorithms, 1997). 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 (Bull. 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. 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 (6), see also [7]), Karger developed the first fully-polynomial randomized approximation scheme (FPRAS) to estimate \( U(p) \) up to relative error \( \epsilon \) in time polynomial in \( n \) and \( 1/\epsilon \); even a constant-factor approximation was not known prior to his work. With an appropriate choice of parameters, Karger’s algorithm can run in time \( \exp(3 \log n + \sqrt{\log n} \sqrt{\log n - \sqrt{2 \log \epsilon + o(\log n)}}) \). In particular, if \( \epsilon = O(1) \), this is \( O(n^5) \). (We assume that \( \epsilon \leq O(1) \) throughout; logarithms will be to the base \( e \) unless specified otherwise.)

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 \cite{9}. This algorithm is an important building block for many graph algorithms. 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 “\( \alpha \)-cut” in \( G \) is cut with at most \( \alpha c \) edges.

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

**Theorem 1.1.** 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 1.2.** Suppose \( c \) is an odd constant and \( \alpha \geq 1 \). Then there are at most \( O(n^{c+\frac{2\alpha}{c}}) \) \( \alpha \)-cuts. This exponent is optimal, in the sense that one cannot show a bound
of $O(n^{\alpha x})$ for any $x < \frac{3}{1+1/x}$. (The constant term in the
asymptotic notation may depend on $c$).

In [9], an algorithm called the Recursive Contraction Algorithm was developed for finding all $\alpha$-cuts. We can apply this algorithm for finding $\alpha$-cuts in the case of small odd $c$ (not merely showing that they exist):

**Theorem 1.3.** There is an algorithm that accepts as input a graph $G$ and a real number $\alpha \geq 1 + 1/c$. This algorithm succinctly enumerates, with high probability, all $\alpha$-cuts of $G$ in time $O(n^{2\alpha} \log n)$.

Note that merely reading the graph might require $n^2$ time, so this running time cannot scale to $\alpha < 1 + 1/c$.

Finally, we mention a small improvement to the Recursive Contraction Algorithm, in the general case. 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 succinctly enumerates, with high probability, all $\alpha$-cuts of $G$ in time $O(n^{2\alpha} \log n)$.

By contrast, the algorithm of [9] 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 $\delta$ 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 $\alpha^*$ such that $U(p)$ is closely approximated by the probability that an $\alpha^*$-cut has failed. Karger provides an upper bound on the critical value $\alpha^*$; 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 $\alpha^*$-cuts. Using an algorithm developed in [9], the Recursive Contraction Algorithm, Karger’s algorithm can catalogue all such $\alpha^*$-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 $\delta$.

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 [11]. 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 $\epsilon^{-2}$ 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 $\alpha^*$. In [10], 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 [6]. 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 $\alpha$-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 [11] is able to handle very general collections of clauses, which can overlap in complicated ways. However, this is overkill 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 collection of cuts produced by the cut-enumeration phase, reducing to about $n^2$; in particular, we do not need to read the entire collection of $\alpha$-cuts for each sample.

In total, we reduce the running time of Karger’s algorithm to about $n^{3+o(1)}\epsilon^{-2}$. Such a run-time appears to be near-optimal for any approach that uses Karger’s recipe; 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 [14]. 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 [14] experimentally that a branching process can efficiently enumerate the probability that some small cut fails. While this algorithm is promising, the analysis of [14] 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.

2 Preliminaries

We let $\exp(x)$ denote $e^x$; all logarithms are to base $e$ unless indicated otherwise.

For a multi-graph $G$, we define a cut of $G$ to be a partition of the vertices into two classes $V = A \cup A'$, 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 $2m$ 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.

For any $\alpha \geq 1$ we define an $\alpha$-cut to be a cut whose corresponding edge-cut has $\leq \alpha c$ edges.

In this paper, we seek to estimate the probability that graph becomes disconnected when edges are removed with probability $p$. When edges are disconnected 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 [6], 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 [5], 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 arithmetical operations to precision $\epsilon$ 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 [6] wherever possible. Recall that \( c \) is the size of the smallest edge-cut. As in [6], we let \( p^\varepsilon = n^{-2-\delta} \), with \( \delta > 0 \) being the case of primary interest (the complementary case can be handled by simple Monte-Carlo sampling). We note that \( \delta \) can be determined in time \( n^{2+o(1)} \), simply by finding the size of the minimum cut. Hence we can assume \( \delta \) 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 \( \delta_0 \). 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 \( \delta = (\log \log \log \log n)^{-1} \).

We will be very interested in \( \alpha \)-cuts, where normally \( \alpha \) 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 [4], and indeed, the rest of this section is basically a recapitulation of Karger’s work from [4].

**Theorem 2.1.** ([4]) The number of \( \alpha \)-cuts is at most \( n^{2\alpha} \).

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

**Lemma 2.1.** Let \( F : \mathbb{R} \to \mathbb{R} \) be any increasing function with the property that, for any \( \alpha \geq 1 \), the number of \( \alpha \)-cuts is at most \( F(\alpha) \). Let \( g : \mathbb{R} \to [0,\infty) \) be any continuous decreasing real-valued function. Then we have

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

**Proof.** Let \( F'(x) \) denote the number of \( x \)-cuts, for any \( x \geq 1 \); so \( F'(x) \leq F(x) \) for all \( x \geq 1 \). The function \( F' \) is an increasing which takes on integer values. Hence we can enumerate points of discontinuity \( \alpha = x_0 < x_1 < x_2 < \ldots \), such that \( F'(x_{i+1}) - F'(x_i) > 0 \) and \( F' \) is constant between those values. Then

\[
\sum_{\text{Cuts } C \text{ of weight } |C| \geq \alpha c} g(|C|/c) = F'(\alpha)g(\alpha) + \int_{x=\alpha}^{\infty} g(x)dF'(x)
\]

As \( g \) is decreasing, by standard measure-theoretic arguments this expression is decreasing also in \( F' \). Hence this is \( \leq F(\alpha)g(\alpha) + \int_{x=\alpha}^{\infty} g(x)dF(x) \) as desired.

Combining Lemma 2.1 and Theorem 2.1, gives a simple proof of the following result from [6]:

**Corollary 2.1.** Let \( 0 < q < n^{-2} \), \( \alpha \geq 1 \). Then we have

\[
\sum_{\text{Cuts } C \text{ of weight } |C| \geq \alpha c} q^{\alpha c} \leq \frac{n^2q^{\alpha}}{1 + \log \frac{n^2}{\log q}}
\]

In particular, if \( qn^2 \) is uniformly bounded below 1, then we have

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

**Proof.** Apply Lemma 2.1, using \( F(x) = n^{2x} \) and \( g(x) = q^x \). We have

\[
\sum_{\text{Cuts } C \text{ of weight } |C| \geq \alpha c} q^{\alpha c} \leq n^{2\alpha} \int_{x=\alpha}^{\infty} q^x \cdot 2n^{2x} \log ndx
\]

\[
= \frac{n^{2\alpha}}{1 + \log \frac{n^2}{\log q}}
\]

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 2.2.** ([6]) Suppose \( \delta \geq \delta_0 \) for some constant \( \delta_0 > 0 \). Then the probability that a cut of weight \( \geq \alpha \) fails is at most \( O(n^{-\alpha\delta}) \).

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

\[
P(\text{Cut of weight } \geq \alpha \text{ fails}) \leq \sum_{\text{Cuts } C \text{ of weight } |C| \geq \alpha} (p^c)^{|C|/c} \]

Now apply Corollary 2.1.

This leads to one of the key theorems of Karger’s original algorithm:

**Theorem 2.2.** ([6]) 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 2.2 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

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

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 2.2. In proving Theorem 2.2, we used two quite different bounds. These bounds are tight respectively, but for very different kinds of graphs. In estimating the absolute error, we bound the number of $\alpha$-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

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

This is an overestimate of the graph failure probability which ignores the overlap between the cuts. The quantity $Z$ will play a crucial role in our estimates. We begin by recalling some results of [6] 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

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

*Proof.* As shown in [12], [13], [6], 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 $q = p^c = n^{-2-\delta}$, 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 $D \geq r n^{\delta/2} - 1$. By the Chernoff bound,

$$P(\geq r \text{ connected components}) \leq \left(\frac{e^D}{(1 + D)^{1+D}}\right)^\mu \leq e^{-1}(e/r)^r n^{-r\delta/2} \leq n^{-r\delta/2}$$

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

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

*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 $e^{-1} \left(\frac{e}{1 + \log_2 t}\right)^{1+\log_2 t} n^{-(1+\log_2 t)\delta/2}$.

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

**Proposition 3.1.** For $\delta \geq \delta_0 > 0$, we have $U(p) = \Theta(Z)$.

*Proof.* By the union bound, $U(p) \leq Z$.

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

By Lemma 3.2, we have

$$E[Z | Z \geq 1] = \sum_x P(Z \geq x | Z \geq 1)$$

$$\leq \sum_x \min(1, P(Z \geq x)n^{2+\delta})$$

$$\leq \sum_x \min(1, n^{-\log_2 x^{\delta/2} n^{2+\delta}})$$

$$\leq 2^{\delta} \min(1, n^{-\delta^2/2 \delta n^{2+\delta}})$$

For $\delta$ and $n$ sufficiently large, the term $2^{\delta} n^{-\delta^2/2 \delta n^{2+\delta}}$ is less than $e^{-\Omega(1)}$, so the summands sum to a constant.
We note that $\tilde{Z}$ is itself bounded:

**Corollary 3.1.** For $\delta \geq \delta_0 > 0$, we have $n^{-2-\delta} \leq \tilde{Z} \leq n^{-2-\delta_0}2^{2+o(1)}$.

**Proof.** The bound $\tilde{Z} = n^{-2-\delta_0}n^{2+o(1)}$ follows immediately from Lemma 2.1.

The bound $\tilde{Z} \geq n^{-2-\delta}$ 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 [9] for finding cuts in a graph. This algorithm is called the Contraction Algorithm, and can be summarized as follows. Suppose we wish to find a target cut $C$.

1. Repeat the following while $G$ has $\geq 2\alpha$ 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 $2\alpha$ 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 on a computer. Rather, we will use this algorithm as a tool to analyze the contraction algorithm, which preserve $C$.

### 4.1 The Contraction Algorithm

The contraction algorithm selects a given $\alpha$-cut $C$ 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 $r = 2\alpha, \ldots, n$ we obtain a subgraph $G_r$ with $r$ vertices and $M_r$ edges (and $G_n = G$). 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 those contained in $C$. 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 endpoints is selected. (This never happens if $L$ is a cut). The Contraction Algorithm can be thought of as the contraction process with the null cut (i.e. the partition $V \sqcup \emptyset$).

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 $S_r^{G,C} = \frac{c}{M_r^{G,C}} + \cdots + \frac{c}{M_r^{G,C}}$; 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 $S_r = S_{[r]}$.

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

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

The probability that the Contraction Algorithm selects a given $\alpha$-cut $C$ is at least

$$P(\text{select } C) \geq e^{-O(\alpha)}E[-\exp(\alpha S_r^{G,C})]$$

Note that by Jensen’s inequality, this implies that

$$P(\text{select } C) \geq e^{-O(\alpha)} \exp(-\alpha E[S_r^{G,C}])$$

**Proof.** In order to select $C$, the following events must happen. First, for each iteration $r = n, \ldots, [2\alpha]$, 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 $G_{[2\alpha]}$. This graph has $2^{2\alpha-1}$ 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 $i = [2\alpha], \ldots, n$, of $(1 - \alpha c/M_i)$. 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 unconditional 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

$$P\text{(select } C\text{)} \geq \mathbb{E}[2^{-2\alpha} \prod_{i=[2\alpha+1]}^{n} (1 - \frac{\alpha c}{M_r})]$$

$$\geq 2^{1-2\alpha} \mathbb{E}[\prod_{i=[2\alpha+1]}^{n} \exp(-\alpha c/M_r) \frac{1 - \frac{\alpha c}{rc/2}}{\exp(-\alpha c/rc/2)},$$

(since $x \mapsto e^{x}(1-x)$ decreases in $[0, 1]$)

$$= 2^{-2\alpha} \mathbb{E}[\exp(-\alpha S_{2\alpha})] \prod_{i=[2\alpha+1]}^{n} (1 - \frac{2\alpha}{r}) \exp\left(\frac{2\alpha}{r}\right)]$$

$$\geq 2^{-2\alpha} \left(\frac{n}{2\alpha}\right)^{-1} \left(\frac{n}{2\alpha+1}\right)^{2\alpha} \mathbb{E}[\exp(-\alpha S_{2\alpha})]$$

$$\geq e^{-O(\alpha)} \mathbb{E}[\exp(-\alpha S_{2\alpha})]$$

Note that

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

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 4.1.** The probability that the Contraction Algorithm selects a given $\alpha$-cut $C$ is at least

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

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 4.2.** Let $C$ be a cut of a connected graph $G$ and let $L \subseteq C$ be a subset of its edges with $|L| = k$. Suppose $r > k$. Then the random variable $M^{G,C}_r$ stochastically dominates $k + M^{G/L,C/L}_{r-k}$. Note that $C/L$ is not necessarily a cut of $G/L$.

In particular we have

$$\mathbb{E}[S_{r,C}^{G,C}] \leq \mathbb{E}[S_{r-k,C}^{G/L,C/L}]$$

**Proof.** We use the notation $X \sim Y$ to mean that the random variables $X$ and $Y$ follow the same law. We prove that $M^{G,C}_r$ stochastically dominates $M^{G/L,C/L}_{r-k}$ by induction on $|G|$.

When $G$ contains exactly $k + 1$ vertices, then $r = k + 1$ is the only valid choice. Here $M^{G,C}_r \geq k$ as $G$ is connected, and $M^{G/L,C/L}_r = 0$.

Now fix a graph $G$ and $r > k$. For any $l \geq 0$ we have

$$P(M^{G,C}_r \geq l) = P(M^{G/e,C}_r \geq l),$$

where $e$ is chosen uniformly from $G - C$

$$\geq P(M^{G/(L\cup\{e\}),C/L}_r + |L/e| \geq l),$$

by inductive hypothesis

$$\geq P(M^{G/(L\cup\{e\}),C/L}_r + k \geq l)$$

where the last line follows since every round of the contraction algorithm removes at least one edge. So it will suffice to show that, as $e$ varies over $G - C$, we have $M^{G/(L\cup\{e\}),C/L}_r \sim M^{G/L,C/L}_r$.

This edge $e$ may be an edge of $G/L$, or the edge $e$ may have already been contracted away in $G/L$. We denote the first event $E_0$.

Suppose we select an edge $e \in G - C$ uniformly at random, and condition on event $E_0$. This is equivalent to simply selecting an edge $e' \in G/L$ uniformly at random, as every edge $e \in G$ corresponds to at most one edge $e' \in G/L$ and every edge $e' \in G/L - C/L$ corresponds to exactly one edge of $G - C$. So, conditional on $E_0$, as $e$ varies the random variable $M^{G/(L\cup\{e\}),C/L}_r$ follows the same law as $M^{G/L,C/L}_r$.

Next, suppose event $E_0$ does not occur. So edge $e$ was contracted away by $L$ and we have $G/(L\cup\{e\}) = G/L$. As this is true for any such $e$, it follows that as $e$ varies we have $M^{G/(L\cup\{e\}),C/L}_r \sim M^{G/L,C/L}_r$.

Putting these two cases together, we have that $M^{G,C}_r$ stochastically dominates $k + M^{G/L,C/L}_r$.

To illustrate how we can use Lemma 4.2, suppose we are interested in the contract process for $C$, where $C$ contains very few edges. In this case, if we apply this lemma with $L = C$, then we would have

$$\mathbb{E}[S_{r,C}^{G,C}] \leq \mathbb{E}[S_{r-k,C}^{G/L,C/L}] \approx \mathbb{E}[S_{r,C}^{G,C}]$$

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 4.2 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 $G/E$. 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 4.3.** Let $\theta \in (0,1)$; $\theta$ could be a function of $n$ or other parameters. Given a graph $G$ and a target $\alpha$-cut $C$, there is a subset of the edges $L \subseteq C$ with the following properties:

1. $|L| \leq O(\frac{\log n}{\theta})$.
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 \(3\)-cut-independence property (with respect to $C$).

**Proof.** Choose $L$ to be a random subset of $C$ of size exactly $\frac{\alpha n \log n}{2}$. (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 $\alpha' c$. If $|C' - C| \geq |C'| (1 - \theta)$, then $C'$ already satisfies property (2). If $|C' - C| < (1 - \theta)|C'|$, then $C'$ overlaps in at least $\theta \alpha' c$ edges with $C$. Hence the probability that no such edges are chosen in $L$ is

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

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

$$\sum \text{Cuts } C' (n^{-3}|C'|/c).$$

By Lemma 2.1, this is $O(n^{-1})$, where the constant term is independent of $\theta$. 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 4.1.** Suppose $G$ has $m$ edges, and 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

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

In particular, if $G$ has $\theta$-cut-independence with respect to $C$, then

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

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

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

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 fewer $\alpha$-cuts than a typical graph. 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 $E[S_i]$ and then proving that this bound is correct. However, 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 out with a graph $G$ with $r$ vertices and $k \leq 2r$ minimum-weight cuts. At each stage of the contraction process, we have

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

where $M_i$ is the number of edges in the subgraph $G_i$ and $S_i = \frac{c}{M_i+1} + \cdots + \frac{c}{M_i}$. As shown in Proposition 4.1, 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 $E[k_i] \approx E[ke^{-S_i}] \approx ke^{-E[S]}$.

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

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

and hence

$$E[c/M_i] \leq \frac{E[k_i]}{i} + (1 - \frac{E[k_i]}{i}) \frac{2c}{(c + 1)i}$$

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

(Note that for this estimate we are not applying, nor will we need, Jensen’s inequality).

This gives us a recurrence relation in $E[S_i]$:

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

We relax this recurrence relation to a differential equation

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

$$S_r = 0$$

which can be solved in closed form to obtain

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

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 4.1 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.

In the following Theorem, we prove that this heuristic formula is in fact a correct bound:

**Theorem 5.1.** Let $c > 1$. Define the function

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

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 omit some technical analysis of the function $f$.

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 $E[S_i^{H'}] = c/m + E[H']E[S_i^{H'}]$. 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 4.1, we have $E[K'] \leq ke^{-c/m}$.

By the inductive hypothesis, we have

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

One can show this is a concave-down increasing function of $K'$, hence by Jensen’s inequality we have

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

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

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

One can show that the expression $c/m + f(i, r - 1, ke^{-c/m})$ is decreasing in $m$. Then we have the bound

$$E[S_i^{H'}] \leq f(i, r, k)$$
Suppose \( k \geq r \). Then we have \( m \geq r\alpha/2 \), so we have the bound
\[
\mathbb{E}[S_i^H] \leq \frac{2}{r} + f(i, r-1, ke^{-\frac{r}{2}}) \\
\leq f(i, r, k)
\]
This completes the induction.

Theorem 5.1 gives us a very precise estimate of \( \mathbb{E}[S_i] \). Usually, a cruder and simpler estimate suffices:

**Lemma 5.1.** Suppose \( G \) has minimum cut \( c \), for \( c \) odd and constant. Then we have
\[
\mathbb{E}[S_i] = \frac{2c}{c+1} \log(n/i) + O(1)
\]
where the constant term does not depend on \( c \).

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

We omit the proof in the case \( c = 1 \). For \( c \geq 3 \), by Theorem 5.1 we have
\[
\mathbb{E}[S_i^C] \leq f(i, n, 2n) \\
= \log \left( \frac{(c+3)(i/n)^{2/c} - 4(i/n)}{(c-1)(i/n)^2} \right) \\
\leq \log \left( \frac{(c+3)(i/n)^{2/(c+1)}}{(c-1)(i/n)^2} \right) + \frac{4(i/n)1^{-2/(c+1)}}{c+1} \\
= \frac{2c}{c+1} \log(n/i) + (1 + \log(3/2))
\]
We can now estimate the probability of selecting the \( \alpha \)-cut \( C \):

**Theorem 5.2.** Suppose \( c \) is an odd constant and \( C \) is an \( \alpha \)-cut. Then \( C \) is selected by the Contraction Algorithm with probability
\[
\mathbb{P}(\text{Contraction Algorithm selects cut } C) \geq e^{O(\alpha)}(n/\alpha)^{-\frac{2\alpha}{c+1}} = \Omega(n^{-\frac{2\alpha}{c+1}}).
\]

**Proof.** First, suppose \( n < \alpha c + 2\alpha \). In this case, the simple analysis of the Contraction Algorithm of Corollary 4.1 (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
\[
\mathbb{E}[S_{2\alpha}^{G,C}] \leq \mathbb{E}[S_{2\alpha+\alpha c}^{G,C}] + 2\log(\frac{\alpha c}{2\alpha}) \\
\leq \mathbb{E}[S_{2\alpha}^{G,C}] + O(1) \quad \text{by Lemma 4.2} \\
\leq \frac{2c}{c+1} \log(n/\alpha) + O(1)
\]
Hence the probability of selecting \( C \) is at least \( e^{-O(\alpha)}(n/\alpha)^{-\frac{2\alpha}{c+1}} \).

When \( \alpha \) grows large, the super-exponential term \( \alpha^{-\frac{2\alpha}{c+1}} \) dominates the simply exponential term \( e^{-O(\alpha)} \). So the terms involving \( \alpha \) alone are bounded, and
\[
\mathbb{P}(\text{Contraction Algorithm selects cut } C) = \Omega(n^{-\frac{2\alpha}{c+1}}).
\]

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

**Proposition 5.2.** 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 \( \alpha \)-cuts is
\[
\left( \frac{n}{\alpha} \right)^{\frac{c}{2c} \alpha} \exp(\Omega(\alpha))
\]

**Proof.** Consider a cycle graph consisting on bundles of \( (c+1)/2 \) edges between each adjacent pair of vertices, except for one edge bundle \( (c-1)/2 \) 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 \( \alpha \)-cuts is at least
\[
\text{Number of cuts} \geq \binom{n-1}{k} \leq \left( \frac{n}{\alpha} \right)^{\frac{c}{2c} \alpha} \exp(\Omega(\alpha))
\]

6 The effects of graph reliability on the Contraction Algorithm

We now show how the graph reliability affects the behavior of the Contraction Algorithm. As in the case of Theorem 5.1, we will use an induction argument to bound \( \mathbb{E}[S_i] \). 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 \( \theta \)-cut-independence with respect to \( C \). In order to estimate \( \mathbb{E}[S_i] \), we need to estimate \( \mathbb{E}[c/M_t] \) for the graph \( G_t \) 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 \( e^{-\theta d/M_t} \). Multiplying the probabilities for each iteration, we expect that each cut survives with probability about \( e^{-\theta d/M_t} \), and so we should have
\[
\mathbb{E}[Z_{G_t}] \approx Z e^{-\theta d/M_t} \mathbb{E}[S_i]
\]
As the neighborhood of each vertex of \( G_t \) is a distinct cut, we have
\[
i d \leq Ze^{-\theta(1-\theta)}e^{-d} \mathbb{E}[S_i]
\]
Theorem 6.1. Define the function
\[ M_i = -\frac{c \log (i/Z)}{2(c \log p + E[S_i](1 - \theta))} \]
We hope that Jensen-type inequality holds, so that
\[ E[c/M_i] \approx -\frac{2(c \log p + E[S_i](1 - \theta))}{i \log (i/Z)} \]
So we have a recurrence relation in \( S_r \), namely
\[ E[S_{r-1}] = E[S_r] - \frac{2E[S_r](1 - \theta) + 2c \log p}{i \log (i/Z)} \]
We relax this to a differential equation
\[ \frac{dE[S_i]}{dt} = \frac{2E[S_i](1 - \theta) + 2c \log p}{i \log (i/Z)} \]
which can be solved in closed form to obtain
\[ E[S_i] = \frac{-c \log p \left(1 - \frac{\log(Z/i)}{\log(Z/r)}\right)^{2-2\theta}}{1 - \theta} \]

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 \( Z \). We define the graph parameter
\[ A^G_{\gamma} = \sum_{c' \in G} e^{\gamma |C'|/c} \]
Note that \( A_{c \log p} = Z \), and note that \( A^G_{\gamma} > 0 \) for all \( \gamma \).

**Theorem 6.1.** Define the function
\[ f(i, r, a, \gamma) = \begin{cases} \frac{-\gamma(1 - \frac{\log(i/a)}{\log(Z/r)})^{2-2\theta}}{2 \log(r/i)} & \text{if } a \leq 1 \\ \frac{-\gamma(1 - \frac{\log(i/a)}{\log(Z/r)})^{2-2\theta}}{2 \log(r/i)} & \text{if } a > 1 \end{cases} \]
Fix a graph \( G \) with \( n \) vertices and minimum cut \( c \). Let \( C \) be a fixed target cut of \( G \), and suppose \( G \) has \( \theta \)-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 \( \gamma \) 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
\[ E[S^H_{r, L}] \leq f(i, r, A^H_{\gamma}, \gamma) \]

**Proof.** We prove this by induction on \( r \). We state without proof some technical properties, for sake of clarity.

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

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

So \( E[S^H_{i, L}] = c/m + E_H E[S^H_{i, L}] \). 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 \( L' = L/c \); second, we continue the Contraction Process on the subgraph \( H' \). Note that the subgraph \( H' \) must also have \( \theta \)-cut-independence with respect to \( C \).

Define \( \gamma' = \gamma + (1 - \theta)c/m \). The graph \( H' \) has \( r - 1 \) vertices. By Proposition 4.1, each cut \( C' \) of \( H \) survives to \( H' \) with probability at most \( e^{-\gamma' |C'|(1-\theta)/m} \).

Thus we have
\[ E[H'[A^H_{\gamma'}]] \leq a \]
Note that \( m \geq rc/2 \), so we have \( \gamma' \geq -2 \log r + 2/r \geq -2 \log(r - 1) \). So the induction hypothesis applies to the graph \( H' \), and we obtain
\[ E[S^H_{i, L}'] \leq f(i, r - 1, A^H_{\gamma'}, \gamma') \]
By Proposition 6.1, Jensen’s inequality applies, and hence we have
\[ E[S^H_{i, L}'] \leq c/m + f(i, r - 1, a, \gamma, (1 - \theta)c/m) \]
We will now bound the number of edges \( m \) of the graph \( H \). Suppose the vertices of \( H \) have degrees \( d_1, \ldots, d_r \). As the neighborhood of each vertex defines a distinct cut, we must have
\[ \sum_{j=1}^{r} \exp(\gamma d_j/c) \leq A^H_{\gamma} = a \]
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
\[ m \geq \frac{r \log(a/r)}{2\gamma} \]
As shown in Proposition 6.2, the quantity \( c/m + f(i, r - 1, a, \gamma + (1 - \theta)c/m) \) is decreasing in \( m \). Hence we must have
\[ E[S^H_{i, L}] \leq \frac{2\gamma}{r \log(a/r)} + f(i, r - 1, a, \gamma + (1 - \theta)\frac{2\gamma}{r \log(a/r)}) \]
\[ \leq f(i, r, a, \gamma) \quad \text{by Proposition 6.3} \]
This completes the induction.
We now describe (without proof) the required concavity and monotonicity properties of $f$.

**Proposition 6.1.** 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 increasing in $a$.

**Proposition 6.2.** For $a \leq 1$ and $r \geq i + 1$, the expression
\[c/m + f(i, r - 1, a, \gamma) + (1 - \theta)c/m\]
is decreasing in $m$.

**Proposition 6.3.** For $a \leq 1$, $\gamma \leq 0$, and $r$ sufficiently large, we have
\[
\frac{2\gamma}{r \log(a/r)} + f(i, r - 1, a, \gamma) + \frac{(1 - \theta)2\gamma}{r \log(a/r)} \\
\leq f(i, r, a, \gamma)
\]

Theorem 6.1 requires the graph $G$ to have $\theta$-cut-independence. We can condition this:

**Theorem 6.2.** Suppose $p^c = n^{-2-\delta}$ and $\bar{Z} = n^{-2-\delta+\beta}$, where $\delta \geq \delta_0 > 0$.

Define
\[v_{\beta, \delta} = \frac{(\delta + 2)(5 - 2\beta + 2\delta)}{(3 - \beta + \delta)^2}.
\]

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

**Proof.** Note that by Corollary 3.1, we must have $0 \leq \beta \leq 2 + o(1)$.

First, suppose $\alpha \geq \sqrt{n}$. Then, (4.1) gives
\[S_{2\alpha} \leq 2 \log \left(\frac{n}{2\alpha}\right) + O(1) \leq 1.02 \log n - 0.01 \log \alpha + O(1).
\]

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 4.3, and set $H = G/L$ Note that $|L| \leq O\left(\frac{\alpha \log n}{2}\right) = O(\alpha \log^2 n)$.

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
\[E[S_{2\alpha}^{G,C}] \leq E[S_{2\alpha}^{G,C}] + 2 \log \left(\frac{O(\alpha \log^2 n)}{2\alpha}\right)
\]
\[
\leq E[S_{2\alpha}^{G,C}] + O(\log \log n) \quad \text{(by Lemma 4.2)}
\]
\[
\leq f(2\alpha, n, \bar{Z}, c \log p) + O(\log \log n)
\]
\[
\leq f(1, n, \bar{Z}, c \log p) + O(\log \log n) - \Omega(\log \alpha)
\]
(as $\alpha \leq \sqrt{n}$)
\[
\leq v_{\beta, \delta} \log n + O(\log \log n) - \Omega(\log \alpha)
\]

**Corollary 6.1.** 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 $\alpha$-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 $\alpha$-cut $C$ is at least
\[P(\text{select } C) \geq \exp(-O(\alpha) - \alpha E[S_{2\alpha}]).
\]

By Lemma 6.2 this is at least $\exp(-O(\alpha)) \exp(\Omega(\alpha \log \alpha)) n^{\alpha v_{\beta, \delta}} \log^{O(\alpha)} n$. Note that all the terms in $\alpha$ alone are bounded, so this is $O(n^{\alpha v_{\beta, \delta}} \log^{O(\alpha)} n)$ as desired.

**7 Bounds on $\alpha^*$**
Recall that the approach of [6] 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 2.2.

To simplify the notation in the remainder of this paper, we will write $v$ instead of $v_{\beta, \delta}$. Typically, the value of $\delta, \beta$ 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 $\leq \alpha$ fails.

For all $\alpha$ we have
\[U(p) - U_\alpha(p) = n^{\alpha v_{\beta, \delta} - \alpha(2+\delta)} \log^{O(\alpha)} n
\]

**Proof.** This follows from Lemma 2.1 and Theorem 2.2.

As in Karger [6], we estimate $U(p)$ by examining only the smallest cuts. We define $\alpha^*$ to be the minimal value of $\alpha$ such that
\[U(p)(1 - \epsilon) \leq U_{\alpha^*}(p) \leq U(p)
\]

The parameter $\alpha^*$ 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 $\alpha^*$-cuts. This is a relatively small collection, which we can explicitly collect and list. In Theorem 2.2, we showed how to bound $\alpha^*$.

For the rest of the paper, we define
\[\rho = -\log \epsilon / \log n.
\]

Although in general we allow $\rho$ to increase arbitrarily, we find that these bounds can become confusing because of the interplay between the asymptotic growth of $n$ and $\epsilon$. 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 2.2, the key theorem of [6]:

\[\]
Lemma 7.1. Suppose $\delta \geq \delta_0 > 0$. Then we have
\[ \alpha^* \leq \frac{(3 - \beta + \delta)(2 - \beta + \delta + \rho)}{(2 - \beta + \delta)^2(2 + \delta)} + O((\log \log n) / \log n). \]

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

The relative error is then at most
\[ \text{Rel Err} = O(n^{\alpha(v-2-\delta)} \log^{O(\alpha)} n / U(p)) \]
by Proposition 3.1
\[ = O(n^{\alpha(v-2-\delta)} \log^{O(\alpha)} n / Z) \]
where $Z$ is a differentiable function of $\alpha$. Hence
\[ \alpha^* \leq \frac{(3 - \beta + \delta)(2 - \beta + \delta + \rho)}{(2 - \beta + \delta)^2(2 + \delta)} + O((\log \log n) / \log n). \]

For $\alpha = -\log \epsilon + (2 - \beta + \delta) \log n + O(\log \log n)$, this is reduced to $\alpha^* \leq \frac{(3 - \beta + \delta)(2 - \beta + \delta + \rho)}{(2 - \beta + \delta)^2(2 + \delta)} + O((\log \log n) / \log n)$.

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

Corollary 7.1. 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 $U(p) = \Theta(Z) = \Theta(n^{\delta-2-\delta})$. Hence $\delta + 2 \geq K$ and $\delta + 2 - \beta \geq K + \Omega(\log^{-1} n)$. Let $t = \delta + 2 - \beta - K$, so $t = O(\log^{-1} n)$.

Suppose we have $\delta = K - 2 + \beta$ exactly. Then
\[ \alpha^* \leq \frac{(K+1)^2(K+\rho)}{K^2(K+\beta)} + o(1) \leq \frac{(K+1)^2(K+\rho)}{K^3} + o(1) \]

The bound on $\alpha^*$ is a differentiable function of $\beta$. When we perturb this by adding $t$ to $\beta$, 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 $n^{\alpha^* v_{\beta,\delta}}$ independent executions of the Contraction Algorithm, selecting some $\alpha^*_{\text{max}}$-cut each time, for
\[ \alpha^*_{\text{max}} = \frac{(K+1)^2(K+\rho)}{K^3} + O(1 + \rho) \]
This will give us a large collection $A$ of cuts, of various sizes. Although we will not know an exact bound for $\alpha^*$, which would depend on knowing $\beta$, it is not hard to see that, with high probability, $A$ will contain all the $\alpha^*$-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 $\beta$, without bounding either term individually. This will let us determine how many samples of the Contraction Algorithm are needed.

Proposition 7.2. 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 $U(p) = \Theta(Z) = \Theta(n^{\delta-2-\delta})$. Hence $\delta + 2 \geq K$ and $\delta + 2 - \beta \geq K + \Omega(\log^{-1} n)$. Let $t = \delta + 2 - \beta - K$, so $t = O(\log^{-1} n)$.

Suppose we have $\delta = K - 2 + \beta$ exactly. Then
\[ \alpha^* v = \frac{K + \rho + O(\log \log n / \log n)}{K + \beta - v_{\beta,K-2+\beta} - v_{\beta,K-2+\beta}} \]
\[ = (2 + 1/K) + \frac{2K + 1}{K^2} - \rho + O(\log \log n / \log n) \]

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

If we use this straw-man 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 $O(n^2)$. In [9], 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 samples, effectively reducing the time for a single execution of the Contraction Algorithm from $O(n^2)$ to $O(1)$. However, this method also introduces dependencies between the multiple samples of the Contraction Algorithm. So we will need to do more work to show that these dependencies remain well-behaved. The next section will examine the Recursive Contraction Algorithm in detail.

8 The Recursive Contraction Algorithm

The basic method of finding the $\alpha^*$-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 [9], this data-processing can be amortized across the multiple iterations. The resulting algorithm, called the Recursive Contraction Algorithm (RCA), can enumerate all the minimal $\alpha$-cuts in time $O(n^{20\log^2 n})$.

Note that as each cut may be of size $\Omega(n)$, merely printing out all the cuts could take time $\Omega(n^{20\alpha+1})$. 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 $n^{\Theta(1)}e^{-\Theta(1)}$.

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 $\beta$, will contain all the $\alpha^*\beta$-cuts with high probability. It may also contain some cuts of size larger than $\alpha^*\beta$. Note that we will still not necessarily know the exact value of $\alpha^*\beta$.

We define the RCA with parameters $t, \alpha_{max}$ as follows.

1. If the graph $G$ has fewer than $2\alpha_{max}$ vertices, output a random cut of $G$.
2. Otherwise, execute the following step twice:
3. Contract randomly edges of $G$ until the resulting graph $G'$ has $\lfloor 2^{-\frac{t}{\alpha}}n \rfloor$ edges. Run the Recursive Contraction Algorithm with parameters $t, \alpha_{max}$ on $G'$.

Following arguments in [9], the Master Theorem for recurrences yields:

**Proposition 8.1.** If $t$ is a constant greater than 2, then the RCA with parameters $t, \alpha_{max}$ runs in time $O(n^t)$. The RCA with parameters $2, \alpha_{max}$ runs in time $O(n^{\log n})$.

Note: when we are proving bounds on running time which are tight up to constant terms, we must be extremely careful in specifying the model of computation. This running time bound holds only when we are allowed to manipulate words of size $O(\log n)$ with appropriate operations, including random sampling, in time $O(1)$. For bounds of the form $n^{a+\Theta(1)}$, we can be much more sloppy in our computational model.

The key is show that the RCA succeeds with high probability:

**Lemma 8.1.** Suppose $t \geq 2$, and $\alpha > 0$ are given constants, and $t/2 \leq \alpha_0 \leq \alpha$ where $\alpha_0$ is constant.

Suppose we are given an $\alpha$-cut $C$ of $G$. Then the RCA finds the cut $C$ with probability at least

$$P(\text{RCA succeeds}) \geq e^{-O(\alpha_{max})}\min \left( (\log(n/\alpha))^{-\alpha}, n^t \exp(-\alpha E[S_{2\alpha}]) \right)$$

**Proof.** We incur a probability hit of $2^{1-\alpha_{max}}$ as we are stopping the Contraction Algorithm prematurely, at $\alpha_{max}$ instead of $\alpha$. Henceforth we assume for simplicity $\alpha_{max} = \alpha$. To simplify the notation we also write $S$ for $S_{2\alpha}$.

We can view the Recursive Contraction Algorithm as a binary tree, of height $h = t \log_2 \frac{n}{2\alpha}$. Each node of height $i$ corresponds to a graph $G'$ with $n_i = n2^{-i/t}$ vertices. We say a node if the branching tree succeeds if 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.

We will use the Inclusion-Exclusion principle to estimate the probability of a leaf node surviving. This estimates the probability as at least $\mu - \Delta$, where $\mu$ is the expected number of successful leaf nodes and $\Delta$ is the expected number of pairs of successful leaf nodes.

For purposes of analysis only, we will attenuate the probability of a survival of a leaf node. Obviously this can only decrease the probability of selecting the cut. By Lemma 4.1, the probability that a given leaf survives is $e^{-O(\alpha)E[e^{-\alpha S}]}$. Each leaf node corresponds to an execution through the Contraction Algorithm. Each successful node corresponds to an execution through the contraction process for $C$, with an associated random variable $S$. For a leaf node with $S \leq s^*$, where $s^*$ will be specified later, we will choose to either discard the resulting cut, or to retain it with probability $e^{-\alpha(s^*-s)}$.

In this way, the probability that a given is leaf node survives

$$P(\text{Leaf survives}) \geq e^{-O(\alpha)E[e^{-\alpha S}]}(P(S \geq s^*) + P(S \leq s^*)E[e^{-\alpha(s^*-s)} | S \leq s^*]) \geq e^{-O(\alpha)E[e^{-\alpha \tilde{S}}]}$$

where we define the random variable $\tilde{S} = \min(S,s^*)$. Hence we have

$$\mu \geq 2^h \exp(-\phi \alpha)E[\exp(-\alpha \tilde{S})]$$

for some constant $\phi > 0$.

Let us consider a pair of leaf nodes with common ancestor at level $i$. These leaf nodes correspond to two executions of the Contraction Algorithm, with two sequences of graphs $G_n, \ldots, G_1$ and $G'_n, \ldots, G'_1$. For
For $s \geq s^*$ the function $f$ is positive, decreasing. It is concave-down for $s^* \leq s \leq s_1$, where 
\[
    s_1 = \frac{\log\left(\frac{2^{-4\alpha + 2\alpha h + 2}}{4\alpha/t - 2}\right)}{\alpha} + \phi.
\]
We want to bound $E[f(S)]$ in terms of $E[S]$. By the concavity, for a given value of $E[S]$, the expectation $E[f(S)]$ is minimized with the following distribution: For some $s \geq s_1$, there is an atom at $S = 0$ and an atom at $S = s$ with probability $E[S]/s$. This yields
\[
    (8.3) \quad P(\text{RCA succeeds}) \geq (1 - \frac{E[S]}{s})f(s^*) + \frac{E[S]}{s}f(s)
\]

Define $y = \phi + h\log\frac{2}{\alpha} + \log h$. One can show that the derivative of (8.3) with respect to $s$ is positive for $s \geq y$ and $n$ sufficiently large.

Now there are two cases. First, suppose $E[S] \geq y$. So the RHS of (8.3) is maximized at $s = E[S]$, yielding
\[
    P(\text{RCA succeeds}) \geq f(E[S]) = \Omega(2^h \exp(-\phi \alpha) \exp(-\alpha S)) = n^t \exp(-O(\alpha)) \exp(-\alpha E[S])
\]

Next, suppose $E[S] \leq y$. In this case, we have
\[
    P(\text{RCA succeeds}) \geq (1 - \frac{E[S]}{s})f(s^*) + \frac{E[S]}{s}f(s)
\]
for some $s^* \leq s \leq y$
\[
    \geq f(y) = e^{-O(\alpha)} h^{-\alpha} = e^{-O(\alpha)} (\log(n/\alpha))^{-\alpha}
\]

Hence we have
\[
    P(\text{RCA succeeds}) = e^{-O(\alpha)} \min((\log(n/\alpha))^{-\alpha}, n^t \exp(-\alpha E[S]))
\]

We would like to explain what this result means intuitively. The best case for us would be if each of the leaf nodes in the branching tree had an independent chance of succeeding. The worst case would be if the solutions came in large clusters. We have shown that this clustering behavior cannot occur when the expected number of successful leaf nodes is small. In that case, then the total probability of success is within a constant factor of the number of expected successes. When the expected number of successes is large, our analysis does not rule out the possibility of clustering. In this case, the probability of success is $(\log n)^{-\alpha}$, but we cannot guarantee that it approaches to one.

In light of this, we can use multiple independent iterations of the Recursive Contraction Algorithm to find our given cut with high probability.
Proof. Suppose \( \alpha \geq (x-2)/2 \). So \( \alpha \) is bounded uniformly away from 1. In this case, the probability of finding the target cut \( C \) in an given iteration is

\[
P(\text{find } C) \geq e^{-O(\alpha_{\max})} \min((\log n)^{-\alpha}, n^2 \exp(-\alpha E[S_{2\alpha}]))
\]

\[
\geq e^{-O(\alpha^2)} \min((\log n)^{-\alpha}, n^2 \exp(-\alpha(v_3,\alpha \log n - \Omega(\alpha^2) + O(\alpha \log n)))
\]

by Theorem 6.2.

\[
\geq e^{-\alpha(1)} n^{-\alpha(1)} \min((\log n)^{-\alpha}, n^2 - v_3, \alpha - \alpha(1))
\]

\[
\geq e^{-\alpha(1)} n^{-2-v_3, \alpha - \alpha(1)} \text{ for } n \text{ sufficiently large}
\]

Repeating this for the \( e^{-\alpha(1)} n^{-2+\alpha(1)} \) iterations increases this probability to \( \Omega(1) \).

Suppose \( \alpha < (x-2)/2 \). Then we cannot use our estimates from Lemma 8.1. However, in this case the simple analysis of the Recursive Contraction Algorithm of [9] shows that a single application of the RCA with parameters 2, \( \alpha_{\max} \) finds a given cut \( C \) with probability \( \Omega(n^{2-2\alpha-\alpha(1)} \exp(-\alpha_{\max})) \) and costs \( O(n^2 \log n) \) work. By repeating this for \( n^{x-2+\alpha(1)} \) iterations, the probability of finding the given cut is increased to \( 1 - (1 - n^{2-2\alpha-\alpha(1)} x^{-2+\alpha(1)}) \geq 1 - e^{x-2\alpha} \geq \Omega(1) \).

Either way, we have shown that after \( n^{x-2+\alpha(1)} e^{-\alpha(1)} \) iterations we have any given cut with probability \( \Omega(1) \). We can view the process of finding all cuts as a coupon collector problem. The total number of cuts is \( O(n^{2\alpha}) \), so by repeating a further \( O(2n \log n) = O(\log \epsilon) \) iterations, we collect all such cuts with high probability.

**Corollary 8.2.** Suppose \( K \) is a constant greater than 2. Then the cost of enumerating all \( \alpha^* \) cuts is \( O(n^{5/2} \epsilon^{-5/4}) \). This running time will be negligible as a proportion of the overall algorithm.

**Proof.** By Proposition 7.2 we have \( \alpha^* v \leq 2 + \frac{1}{2\alpha^*} + \frac{1}{2\alpha^*} + O(\log \log n / \log n < (5/2 - \Omega(1)) + (5/4 - \Omega(1)) + O((\log \log n) / \log n). \)

Hence, by Corollary 8.1, the cost of enumerating all \( \alpha^* \)-cuts is \( O(n^{5/2} \epsilon^{-5/4}) \).

**Further applications of the Recursive Contraction Algorithm**

The Recursive Contraction Algorithm is a powerful algorithm to enumerate all the approximately-minimum cuts in a given graph \( G \). Our reliability-estimation algorithm uses in a very specific way based on bounds for the number of cuts of \( G \). However, the theorems we have proved about the behavior of the Contraction Algorithm and the Recursive Contraction Algorithm can be extended to more general situations.

We begin by considering the most simple problem of all. Given a particular value of \( \alpha \) and a given graph \( G \) (with no information about the cut structure or reliability of \( G \)), how to enumerate the \( \alpha \)-cuts of \( G \). The Recursive Contraction Algorithm was first introduced in [9] 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 8.1.** There is an algorithm to enumerate, with high probability, all \( \alpha \)-cuts of \( G \) in time \( O(n^{2\alpha} \log n) \).

In contrast, the algorithm of [9] requires time \( O(n^{2\alpha} \log^2 n) \).

**Proof.** First, suppose \( \alpha < 3/2 \). Then [8] describes a data structure to represent all the \( \alpha \)-cuts in time \( O(n^2) \) and to enumerate all the \( \alpha \)-cuts in time \( O(n^2) \).

Suppose \( \alpha \geq \sqrt{n} \). In this case, by repeating \( O(2n \log n) \) iterations of the ordinary Contraction Algorithm, one enumerates all \( \alpha \)-cuts with high probability in time \( O(n(\alpha (x)^{2-2\alpha} \alpha \log n) = O(n^{2\alpha} \log n) \).

Finally, suppose \( 3/2 \leq \alpha \leq \sqrt{n} \). Let us set \( 2 < t < 3 \) to be constant and apply the bound of Lemma 8.1, so that the probability of finding a given \( \alpha \)-cut is

\[
P(\text{RCA succeeds}) = e^{-O(\alpha)} \min((\log n)^{-\alpha}, n^2 \log n / \alpha)
\]

As \( \alpha \leq \sqrt{n} \), the term \( (n/\alpha) \) is increasing polynomially in \( n \). Hence, for any fixed \( t \) and any \( \alpha \geq 3/2 \), the term \( (n/\alpha)^{t-2\alpha} \) decreases faster than \( \log \alpha \). So, for \( n \) sufficiently large, we have

\[
P(\text{RCA succeeds}) = e^{-O(\alpha)} (n/\alpha)^{t-2\alpha}
\]

We can view the process of finding all \( \alpha \)-cuts as a coupon collector problem. As there are at most \( n^{2\alpha} \) cuts, if we run \( \frac{O(n^{2\alpha}) \log n}{\alpha} \) independent executions of the RCA we find all \( \alpha \)-cuts with high probability. This gives a total run time of \( O(n^{2\alpha} \log n) \).

**Remark.** In using this algorithm, one must be extremely careful as to what operations are performed.
on the resulting cut. Even simple operations, such as counting the number of distinct cuts, may require data-structures which take time $\log n$ or higher.

A slight modification of this argument uses the bounds we have developed for graphs with small odd $c$:

**THEOREM 8.2.** Suppose $c$ is a constant and odd. For $\alpha \geq 1 + 1/c$ there is an algorithm to enumerate, with high probability, all $\alpha$-cuts of $G$ in time $O(n^{\frac{\alpha}{c+1}} \log n)$.

9 Estimating failure probability from the $\alpha^*$-cuts

In the previous section, we showed how one can find a set $A$ of cuts which contains the $\alpha^*$-cuts with high probability, where $\alpha^*$ 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 $\alpha^*$-cuts, then we have $(1 - O(\epsilon)) U(p) \leq U_A(p) \leq U(p)$. So it will suffice to estimate $U_A(p)$.

Karger’s analysis uses a sophisticated algorithm developed by Karp, Luby, Madras [11] 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 $\sum_{C' \in A} p^{\left|C'\right|}$.
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 $\hat{U}(p) = \sum_{C' \in A} p^{\left|C'\right|}$

   The main cost of this algorithm is the step in which we must count the cuts in $A$ containing $E \cup C_0$. 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-theoretical 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 $C_0 \cup E$ from the original graph $G$, we define the graph $G'$ by contracting all the connected components of $G - C_0 - E$. 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 $\alpha p^{\left|C_0\right|}$.
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 $\hat{U}(p) = \sum_{C' \in A} p^{\left|C'\right|}$

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 longer time than $O(n^2)$. 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 $O(n^2)$. Let $R$ denote the number of connected components of the graph $G - C_0 - L$. Then step (4) can take time $2Rn^2$, as the graph $G'$ has $R$ vertices and hence at most $2R$ cuts.

We will show a bound on $E[2R]$. We first consider conditioning on selecting a fixed $\alpha$-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

\[
E[2R \mid C_0 = C] \leq 1 + \sum_{r \geq 1} \Pr(H \text{ has } r \text{ components } \mid C \text{ fails})(2^r - 2^{r-1})
\]
\[
\leq 1 + \sum_{r \geq 1} 2^{r-1} \min(1, p(\geq r \text{ components})^2) + \alpha)
\]

\[
\leq 1 + \sum_{r \geq 1} 2^{r-1} \min(1, (e/r)^x n^{-\delta/2} n^{(2+\alpha)})
\]

by Lemma 3.1.

The cut \( C \) is selected with probability \( \frac{p^C}{\sum_{A \in A} p^A} \). 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

\[
E[2^R \times \mid C \mid \geq xc] 
\leq \sum_{\alpha=x}^{\infty} n^{2\alpha} \frac{p^{\alpha c}}{n^{2-\delta}}
\]

\[
(1 + \sum_{r \geq 1} 2^{r-1} \min(1, (e/r)^x n^{-\delta/2} n^{(2+\delta)\alpha}))
\]

\[
\leq \sum_{\alpha=x}^{\infty} n^{2\alpha} \frac{p^{\alpha c} n^{2+\delta} O(\sum n^{-\delta/2} n^{(2+\delta)\alpha})}
\]

\[
\leq \sum_{\alpha=x}^{\infty} n^{2\alpha} \frac{p^{\alpha c} n^{2+\delta} O(\alpha n^{2+\delta/2})}
\]

For some \( x = O(1) \) the summand is decreasing exponentially so by Lemma 2.1 contributes a total of \( O(1) \).

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

\[
E[2^R \mid C_0 = C]
\]

\[
\leq 1 + \sum_{r \geq 1} 2^{r-1} \min(1, (e/r)^x n^{-\delta/2} n^{(2+\delta)\alpha})
\]

\[
\leq 1 + \sum_{r \leq 2n^{2+\delta/2}} 2^{r-1} + \sum_{r \geq 2n^{2+\delta}} n^{-\delta/2} n^{(2+\delta)\alpha}
\]

\[
\leq 1 + O(1) + \frac{2}{\delta \log n}
\]

= \( O(1) \)

In either case, the contribution to the expectation \( E[2^R] \) 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 between the estimation algorithm and the algorithm of [11] 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 [11] uses a dynamic self-adjustment scheme so that these samples can be generated quickly, and thus the total amount of work is guaranteed to remain low. In our case, we will never need that 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 \( \Pr(J \geq t) \leq \phi \). This will suffice to show that \( E[J] = \Omega(1) \).

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

\[
\Pr(|C_0| \geq \alpha c) = \frac{\sum_{C \in A, |C| \geq \alpha c} p^C}{\sum_{C \in A} p^C}
\]

\[
\leq \frac{\sum_{C \in A, |C| \geq \alpha c} p^C}{n^{-2-\delta}}
\]

\[
= o(1)
\]

for \( \alpha \) sufficiently large, by Lemma 2.1

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 \( \alpha \)-cut for \( \alpha = O(1) \). Then we have

\[
\Pr(J \geq t \mid C_0 = C) \leq \Pr[Z \geq t \mid C_0 = C]
\]

\[
\leq n^{\delta(2+\delta)} n^{-\delta \log \epsilon} t \]

by Lemma 3.2

\[
= o(1)
\]

for \( t \) sufficiently large constant

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

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 to \( U_A(p) \) to relative error \( \epsilon \) in time \( n^2 \epsilon^{-2} \).

**Proof.** A single iteration of the algorithm has relative error \( O(1) \) and costs \( n^2 \). We repeat \( \epsilon^{-2} \) independent trials of this algorithm, 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 \( \epsilon \).

**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.

One detail that has been glossed over in [6], is how to 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.
Suppose Proposition 10.1. Between the two regimes: we observe the graph become disconnected, we will use Monte-Carlo sampling. This is good expected number of successes is at least \(\phi\). As shown in [6] there is an algorithm based on Monte-Carlo sampling in time \(O(n^{3+\gamma}n^{-2})\), where \(\gamma > 0\) is constant.

Proof. As \(n \to \infty\), the number of times the graph becomes disconnected approaches to a Poisson random variable. In the first case, when \(U(p) > n^{-K}\), the expected number of successes is at least \(\phi\), and so the probability of at least one success approaches to 1. 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 \(O(n^{3+\gamma}n^{-2})\).

**Proof.** Let \(K = 2 + \gamma/2\).

Suppose that \(U(p) < n^{-K}\) and we elect to use the cut-enumeration procedure. As shown in Corollary 8.2 the work to find a collection of cuts \(A\) which contains all the \(\alpha^*\)-cuts is \(O(n^{3+\gamma}n^{-2})\), which is negligible. Next, using the estimation algorithm for \(A\), one can use this collection \(A\) to estimate \(U_A(p)\) in time \(O(n^{3+\gamma}n^{-2})\). By definition of \(A\), 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 that \(U(p) \geq \phi' n^{-K}\), where \(\phi'\) is a constant chosen in Proposition 10.1, and elect to use Monte-Carlo sampling. As shown in [6] there is an algorithm based on Monte-Carlo sampling in time \(O(n^{3+\gamma}n^{-2})\).

The total work is \(O(n^{3+\gamma}n^{-2})\) 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 algorithm.

Note that our estimates have been developed under the assumption that \(\delta\) 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 \(n^{3+\gamma}\) 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. This does not occur in our case, and we obtain:

**Theorem 10.1.** There is an algorithm to estimate \(U(p)\) in time \(O(n^{3+\gamma}n^{-2})\).

**Acknowledgments.** We thank the referees for their very helpful comments.

**References**