Sequential Importance Sampling Algorithms for Estimating the All-terminal Reliability Polynomial of Sparse Graphs

David G. Harris*[1] and Francis Sullivan[2]

1 Department of Applied Mathematics, University of Maryland, College Park, MD 20742, USA
davidgharris29@hotmail.com

2 IDA/Center for Computing Sciences, Bowie, MD 20715, USA
fran@super.org

Abstract

The all-terminal reliability polynomial of a graph counts its connected subgraphs of various sizes. Algorithms based on sequential importance sampling (SIS) have been proposed to estimate a graph's reliability polynomial. We show upper bounds on the relative error of three sequential importance sampling algorithms. We use these to create a hybrid algorithm, which selects the best SIS algorithm for a particular graph G and particular coefficient of the polynomial.

This hybrid algorithm is particularly effective when G has low degree. For graphs of average degree ≤ 11, it is the fastest known algorithm; for graphs of average degree ≤ 45 it is the fastest known polynomial-space algorithm. For example, when a graph has average degree 3, this algorithm estimates to error ϵ in time $O(1.26^n \epsilon^{-2})$.

Although the algorithm may take exponential time, in practice it can have good performance even on medium-scale graphs. We provide experimental results that show quite practical performance on graphs with hundreds of vertices and thousands of edges. By contrast, alternative algorithms are either not rigorous or are completely impractical for such large graphs.

1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph theory, G.3 Probability and statistics

Keywords and phrases All-terminal reliability, sequential importance sampling

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.323

1 Introduction

Let G be a connected undirected multi-graph with vertex set V and edge set E. We define $\mathrm{Rel}(p)$, the all-terminal reliability polynomial of G, to be the probability that the graph remains connected when edges are removed independently with probability p. This function is a polynomial which can be factored as

$$\mathrm{Rel}(p) = \sum_{i=0}^{m} N_i (1-p)^i p^{m-i}$$

where N_i is the number of connected subgraphs of G with i edges. This polynomial has various physical applications, for example determining the reliability of a computer network or power grid.

* Research supported in part by NSF Awards CNS-1010789 and CCF-1422569.

18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) / 19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Editors: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim; pp. 323–340
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Exactly computing the reliability polynomial is known to be #P-complete [20]. The best-known algorithm at this time is due to [3]; it runs in 2^n time and 2^n space, or 3^n time and polynomial space. While this is a great theoretical achievement, and works well on small graphs ($n \approx 40$), it is completely impractical for larger graphs.

In this paper, we give an algorithm targeting sparse graphs (graphs with average degree $\alpha = O(1)$), running in polynomial space and exponential time, to estimate the reliability polynomial coefficients N_t to some relative error ϵ. When α is a small constant, the algorithm may be significantly faster than [3]. When $\alpha \leq 11$, this algorithm runs faster than the exponential-space variant of [3], plus our algorithm is polynomial-space; in fact, our algorithm is the fastest known method in this regime. When $\alpha \leq 45$, it runs faster than the polynomial-space variant of [3]. Furthermore, our algorithm can be easily parallelized — absolutely no communication or memory is required between different processing elements.

Furthermore, in practice this algorithm seems to scale much better than the theoretical guarantees would imply. We have conducted extensive experiments on relatively large graphs, up to hundreds of vertices. The algorithm can achieve a 10% error rate on all coefficients after a minutes' run-time. This put real-world networks within reach. By contrast, the existing algorithms such as [3] simply cannot be run on such large graphs — these algorithms will use more space or time than exists on any computer.

1.1 Background

A variety of approaches have been proposed to estimate the reliability polynomial, or fragments of it, for a graph. Some algorithms seek to compute $\mathrm{Rel}(p_0)$ for a fixed probability p_0, such as [10] or [16]. The problem of estimating $\mathrm{Rel}(p_0)$ is related to the problem of estimating the reliability polynomial coefficients N_i, but they are not equivalent especially in terms of evaluating the relative error. For example, Karger's algorithm [16] gives a fully-polynomial relative approximation scheme (fpras) for the problem of estimating $1 - \mathrm{Rel}(p_0)$.

Other algorithms, for example [19], [20] can estimate N_t in polynomial time, but only for a narrow range of value of t, such as $t = O(1)$ or $t = m - n - O(1)$. [5] gives an algorithm to transform a given dense graph G into a relatively sparse graph G', with only $O(n \log n)$ edges, which has approximately the same reliability. There is no known polynomial-time algorithm to estimate arbitrary coefficients.

Many heuristic algorithms have been studied to approximate the all-terminal reliability. For example, [4] discusses an algorithm based on Monte-Carlo-Markov-Chain sampling of connected subgraphs. For the special case of directed acyclic graphs, [18] describes an algorithm based on Monte-Carlo sampling for the closely related problem of estimating $s - t$ connectivity. Algorithms based on sequential importance sampling are described in [2],[14],[8]. While these approximations can be effective in practice, typically they do not have rigorous complexity bounds. This can be especially problematic for estimation algorithms: if the algorithm is run for too few iterations, then it may produce a wrong estimate despite outwardly appearing to run successfully.

1.2 The Tutte polynomial

The reliability polynomial is a special case of the Tutte polynomial, a very powerful graph invariant. While the reliability polynomial counts only the connected subgraphs of G, the Tutte polynomial also encodes information about the decomposition of G into its connected components.

In [3], Bjorklund et al. gave an algorithm running in 2^n time and space, or time 3^n and polynomial space, to exactly compute all the Tutte polynomial coefficients. The Tutte polynomial can be specialized to compute the reliability polynomial, and in fact the algorithm of [3] provides the fastest algorithm, in general, for computing the reliability polynomial. Under appropriate complexity-theoretic assumptions, the Tutte polynomial cannot be computed in time $\exp(o(n))$ [9], or even approximated in polynomial-time [12]. (It is possible that the reliability polynomial may be a tractable special-case, though).

For classes of sparse graphs, alternative algorithms may be still faster. For example, the algorithm of Bjorklund et al. itself can be specialized for graphs of maximum degree Δ; on such a graph, the algorithm has running time $\exp((2^{\Delta+1}-1)^{1/(\Delta+1)}n + o(1))$. Other algorithms have been proposed to regular graphs (such as [7]) or bounded-degree graphs (such as [11]), although these appear to be dominated by the algorithm of [3].

The algorithm of Bjorklund et al. represents a breakthrough from the theoretical point of view, in reality is impractical graphs of even moderate size. For example, experiments in [3] describe a computation time of four days to solve a graph with 22 vertices.

1.3 Our contribution

In this paper, we will propose a new algorithm to estimate the reliability polynomial coefficients N_t for graphs whose *average degree* α is smal. The running time of this algorithm will have the form $(\chi_\alpha + o_\alpha(1))^n/\epsilon^2$, where ϵ is the desired relative error and χ_α is a parameter depending on α. The space required by the algorithm is polynomial in all parameters. This algorithm has a number of advantages over the existing ones, both from a theoretical and a practical point of view.

First, the theory: for small α, the algorithm may be significantly faster than Bjorklund et al. or any other known algorithm. As we will see, this algorithm will be faster than any known algorithm for $\alpha \leq 11$. At this point, the algorithm of Bjorklund et al. becomes faster, although the latter requires exponential space. Our algorithm will be faster than any known *polynomial-space* algorithm for $\alpha \leq 45$.[1] Note that we are considering the most general class of sparse graphs, while other algorithms of [3], [7], [11] were specialized to more restrictive classes of sparse graphs (most prominently, bounded-degree).

Second, in practice this algorithm seems to scale much better than the theoretical guarantees would imply. We have conducted extensive experiments on medium-scale graphs, up to hundreds of vertices. The algorithm can achieve a 10% error rate on all coefficients after a minutes' run-time. This puts real-world networks within reach, whereas the alternative exponential-time algorithms would never finish their computations on such large graphs.

There is a synergy between the theoretical analysis and the practical performance. Certain key parameters must be tuned accurately for our algorithm to guarantee good performance. These parameters are difficult to set empirically, and if they are set incorrectly, the algorithm can appear to run perfectly well but nevertheless return inaccurate results. By setting these parameters in accordance with the worst-case analysis, we tend to achieve results that are accurate in practice. This is despite the fact that the worst-case analysis, for such large graphs, is not directly relevant to practical computations.

[1] This is likely a conservative estimate of the worst-case behavior of our algorithm; an accurate estimate would require solving some open problems in extremal graph theory.

2 General approach

The algorithm we propose is based on sequential-importance sampling (SIS). Three SIS approaches were proposed in [2],[14],[8]. Experimental evidence showed that these algorithms could give quite effective estimates, at least in certain parameter ranges.

The SIS algorithms can be divided into two types. The first, which we refer to as top-down, start by removing edges from the initial graph G and counting the connectivity of the resulting subgraph. The second, which we refer to as bottom-up, start from a spanning tree of G and add edges. On any particular graph, the top-down algorithms tend to have better relative variance when the t is large; the bottom-up algorithms tend to have better relative variance when t is small. (Recall N_t is the coefficient of interest.)

There is a natural strategy to combine the strengths of these algorithms: for any given graph G and any desired coefficient N_t, run all three algorithms in parallel and select the one which gives the best estimate. Unfortunately, there is not any generic method to determine which statistic has lowest variance, and in fact exponentially many samples may be required to calculate a sample variance accurately. Thus, exponential time might be incurred in simply accumulating enough statistical information in order to select the appropriate algorithm.

In this paper, we compute upper bounds on the precision of the three algorithms. These upper bounds play two roles. First, they allow us to find upper bounds on the running time required to achieve any desired level ϵ of relative accuracy. The main probabilistic tool we use in this paper is to estimate the relative variance of the estimate

$$\mathrm{rv}(F) = \mathbf{E}[F^2]/\mathbf{E}[F]^2.$$

As a consequence of the Chebyshev inequality, one achieves the desired precision with high probability after extracting $\Theta(\mathrm{rv}(F)/\epsilon^2)$ independent samples. Hence, if we can bound the relative variance of *any* of the SIS algorithms as $(\chi + o(1))^n$, this implies that the running time of the resulting hybrid algorithm will also be $(\chi + o(1))^n/\epsilon^2$. We will see that for a fixed value of the average degree α, we can achieve good bounds on χ_α.

The second, less-obvious function of these upper bounds, is that they allow us to create a hybrid algorithm, which computes the upper bounds and publishes the statistic with smallest upper-bound on precision. That is, for any fixed graph G, these bounds give us a sensible strategy on which coefficients to estimate with the top-down algorithm and which coefficients to estimate with a bottom-up algorithm.

In practice, these SIS algorithms can be far more accurate than the worst-case analysis would predict. We will examine some examples of this in Section 5. Thus, we can evaluate this hybrid algorithm empirically, and we see that it can give useful estimates for graphs of moderate size (hundreds of vertices), while the exact exponential-time algorithms are impractical for tens of vertices.

Once we estimate N_t, we can automatically estimate $\mathrm{Rel}(p)$ for any fixed value of p. For, if we estimate $\widehat{N_t}$ up to relative error ϵ for all t, we may then estimate

$$\widehat{\mathrm{Rel}(p)} = \sum_i \widehat{N_i} p^i (1-p)^{m-i} \tag{1}$$

and we note that all the summands in (1) are positive.

Despite the advantages of our algorithm, we note that it is ultimately much more specialized than that of Bjorklund et al., for three reasons. First, we only estimate the coefficients instead of computing them exactly, and we do so probabilistically instead of deterministically. Second, this algorithm applies only to the reliability polynomial, which is a special case of the Tutte polynomial. Third, the complexity of our algorithm is exponential in the number of edges, and hence for dense graphs it is slower.

2.1 Notation

We are given an connected undirected graph G, which may have multiple edges or self-loops. We will use m, n to refer to the number of edges and vertices in the graph G. Our algorithm seeks to estimate N_t, and t is always the index of the coefficient we are interested in. The average degree α is given by $\alpha = 2m/n$. Note that as G is connected, we must have $\alpha \geq 2 - 2/n$.

In this paper, we will only be concerned with the case that α is a small constant. Hence, we will assume throughout that $\alpha \leq Cn$, where C is some large fixed constant. Together with the restriction that $\alpha \geq 1$, this implies that any asymptotics in m can be reduced to equivalent asymptotics in n. We will make this assumption $m = \Theta(n)$ throughout this paper.

We let $K = m - n + 1$ and let $k = t - n + 1$. Note that spanning trees of G have $n - 1$ edges, and so k counts the distance of t from its maximal value K. This is particularly useful for algorithms which add edges to spanning trees. We let $\kappa(G)$ denote the number of (labelled) spanning trees of G. Note that using the Kirchoff formula, $\kappa(G)$ can be computed in polynomial time.

We will frequently use the entropy function in estimating binomial coefficients. To simplify the notation in these estimates we define $l(x) = x \ln x$. By Stirling's formula, for any $c \in [0, 1]$ we have

$$\exp(n(-l(1-c) - l(c)) - o(1)) \leq \binom{n}{cn} \leq \exp(n(-l(1-c) - l(c)) + o(1)) \tag{2}$$

We will always use the notation $\beta = t/n$. Note that $\beta \in [1 - 1/n, \alpha/2]$.

2.2 The Kruskal-Katona Theorem

We will frequently need to lower-bound N_t in terms of $\kappa(G)$. A key technical tool to do so comes from the Krusal-Katona Theorem [17]. This is a basic combinatorial principle which gives bounds on the number of objects in a family of sets which is upward-closed. Graph connectivity has this property, as if a graph H is connected and $H' \supseteq H$ then H' must be connected as well. We will use a simplified version of this pricniple, which is slightly less accurate than the full Kruskal-Katona bound but it adequate for our purposes.[2]

▶ **Theorem 1** ([17]). *Let m' be the unique integer such that*

$$\binom{m'}{K} \leq \kappa(G) < \binom{m'+1}{K}.$$

Then for all $t \geq n - 1$ we have

$$N_t \geq \binom{m'}{m-t}.$$

[2] We contrast our use of the Kruskal-Katona Theorem to that of [1]. [1] uses this bound, or variants of it, to estimate N_t itself. We use this bound to estimate the error committed by our SIS algorithms, but these algorithms do not themselves refer to the Kruskal-Katona bounds in any way.

We can explain the intuition behind this result. If we completely ignore graph connectivity, then there are exactly $\binom{m}{m-t}$ subgraphs with t edges. Thus, the Kruskal-Katona Theorem states that the number of connected subgraphs and the number of subgraphs have a similar behavior as a function of t.

We will use throughout the notation

$$\gamma' = m'/n$$

Note that $K \leq m' \leq m$, so that $\gamma \in [\alpha/2 - 1 - 1/n, \alpha/2]$.

3 The basic algorithms

We begin by describing two very simple statistics to estimate N_t. These statistics are accurate for different ranges of the parameter t. We then discuss how to select the best statistic for a given value of t.

3.1 Top-down algorithm bounds

Let us first consider a top-down estimate, which starts with the original graph and removes edges until it reaches a disconnected graph. Two such algorithms are discussed in [2], [15], which use a variety of heuristics to select which edge to remove. These heuristics are somewhat difficult to analyze, although they are very useful in practice, so for this paper we will consider a simplified algorithm. The following algorithm, which we refer to as TOPDOWN, has strictly larger relative variance compared to [2], [15]:

1. Select uniformly at random a subgraph $H \subseteq G$ with t edges.
2. Check if H is connected.
3. If H is connected estimate $\widehat{N_t} = \binom{m}{t}$; otherwise estimate $\widehat{N_t} = 0$.

Note that this algorithm is not really a "top-down" algorithm any more, since we could produce H by adding edges as well as removing edges.

The statistic produced by TOPDOWN is clearly an unbiased estimator for N_t.

▶ **Proposition 2.** *Define*

$$f_T = l(\alpha/2) - l(\beta) - l(\gamma) + l(\beta + \gamma - \alpha/2)$$

Then TOPDOWN has relative variance $\exp(n(f_T + o(1)))$.

Proof. TOPDOWN is a Bernoulli random variable with probability $N_t / \binom{m}{t}$, so it has relative variance $\frac{\binom{m}{t}}{N_t}$. By Theorem 1, this is at most $\frac{\binom{m}{t}}{\binom{m'}{m-t}}$. Recalling that $t = \beta n, m = alphan/2, m' = \gamma n$, e apply Stirling's formula (2). (We note here that $m = \Theta(n)$, so all asymptotic terms can be reduced to $o(n)$). This gives the estimate:

$$\mathrm{rv}_T \leq \exp(n(f_T + o(1)))$$

◀

We note one key difference between this algorithm, based on estimating the reliability coefficients, and similar algorithms such as [21] which seek to compute the coefficients exactly. In this case, we are doing a Monte-Carlo estimation of the number of connected subgraphs, so our algorithm is more efficient when the subgraphs are numerous. Enumerative algorithms, by contrast, must explore each subgraph individually, and so they are more efficient when the subgraphs are few.

3.2 Bottom-up, no edge weighting

The bottom-up algorithm is based on starting with a random spanning tree and adding edges to it. This type of algorithm is discussed in [8], [14]. In Section 4, we will consider more sophisticated variants which use add edges with a non-uniform probability. However, as a warm-up exercise, we consider the following simple algorithm which we refer to as BOTTOMUP:

1. Choose a spanning tree T uniformly at random from G. (This can be done efficiently as described in e.g. [22])
2. Choose $k = t - n + 1$ edges uniformly at random from $G - T$, and add these edges to obtain a connected subgraph $H \subseteq G$.
3. Estimate $\widehat{N_t} = \frac{\kappa(G)\binom{K}{k}}{\kappa(H)}$

Recall that $\kappa(G)$ can be computed in polynomial time, so this is a polynomial-time algorithm.

▶ **Proposition 3.** *Define*

$$f_B = -l(\beta - 1) - l(1 - \alpha/2 + \gamma) + l(\beta - \alpha/2 + \gamma)$$

Then algorithm BOTTOMUP is an unbiased estimator with relative variance at most $\exp(n(f_B + o(1)))$.

Proof. Let H be a connected subgraph of G with t edges. Then BOTTOMUP selects H with probability $p_H = \frac{\kappa(H)}{\kappa(G)\binom{K}{k}}$. Integrating over all such H, we see that the expected value of $\widehat{N_t}$ is given by

$$\mathbf{E}[\widehat{N_t}] = \sum_H p_H \frac{\kappa(G)\binom{K}{k}}{\kappa(H)} = \sum_H 1 = N_t$$

as desired.

Next, we can estimate

$$\mathbf{E}[\widehat{N_t}^2] = \sum_H p_H \Big(\frac{\kappa(G)\binom{K}{k}}{\kappa(H)}\Big)^2 = N_t \kappa(G)\binom{K}{k}\mathbf{E}_H[1/\kappa(H)] \leq N_t \kappa(G)\binom{K}{k}$$

Hence, the relative variance is given by

$$\text{rv} \leq \frac{N_t \kappa(G)\binom{K}{k}}{N_t^2} \leq \frac{m\binom{m'}{K}\binom{K}{k}}{\binom{m'}{m-t}}$$

Applying Stirling's formula (2), and recalling $m = \Theta(n)$, shows that this is at most $\exp(n(f_B + o(1)))$, as desired. ◀

3.3 Hybrid algorithm

Now define the hybrid algorithm which, for any graph G and any desired coefficient t, computes f_T and f_B. If $f_T < f_B$, then this algorithm outputs TOPDOWN; otherwise it outputs BOTTOMUP. By Propositions 2, 3, this hybrid algorithm satisfies

$$\text{rv} \leq \exp(n \min(f_T, f_B) + o(n))$$

Let us examine how to bound the quantity $\min(f_T, f_B)$ as a function of β, γ. For any fixed γ, f_T is an decreasing function of β and f_B is a increasing function of β, reaching extreme values $f_T = 0$ at $\beta = \alpha/2$ and $f_B = 0$ at $\beta = 1$. Hence the maximum value of $\min(f_T, f_B)$ occurs at the point where $f_T = f_B$.

We seek to maximize f_B subject to $f_T = f_B$. In general, there is no closed form solution. However, for any given value of α we can numerically optimize this as follows. Given any value of γ, there is a unique value β such that $f_T = f_B$, which can be found to arbitrary accuracy via bisection. This essentially reduces the search to a single variable γ. One can verify that the resulting function f_T is unimodal in γ, and hence the maximum value of γ can be found again by a golden-section search strategy. This allows us to find a value β^*, γ^* maximizing $\min(f_T, f_B)$ to an arbitrary accuracy. The resulting value χ_α is the worst-case for the running time, as a function of α. That is, when α is constant and $n \to \infty$, we have $\mathrm{rv} = \exp(n\chi_\alpha + o(n))$ for graphs with average degree $\leq \alpha$.

The following table shows bounds on χ_α for various values of α. Note that unlike algorithms which restrict to regular or bounded-degree graphs, there is no restriction on the integrality of α.

α	χ_α	α	χ_α
3	1.32	15	2.34
4	1.51	20	2.55
6	1.76	25	2.72
8	1.93	30	2.87
10	2.08	35	2.99

For $\alpha \leq 8$, this algorithm is strictly faster than [3]; for $\alpha \leq 35$ this algorithm is faster than the polynomial-space variant of Bjorklund et al. We will improve this still further in the next section.

4 Bottom-up, edge weighting

We consider a variant of the bottom-up algorithm in which a weighting function is used to select the edges to add to the spanning tree, as described in [14]. Again without concerning ourselves with polynomial efficiency, we summarize this algorithm as

1. Select a spanning tree T uniformly at random
2. For $i = 1, \dots, k$, repeat the following:
   3. Randomly select an edge e_i to add to T. We use the probability distribution P_i (which is conditioned on $e_1, \dots, e_{i-1}, T$) to select the edge e_i, and this probability distribution P_i is given by
$$P_i(e') \propto \kappa(T \cup e_1 \dots \cup e_{i-1} \cup e')^{-\rho}$$
4. Estimate
$$\widehat{N_t} = \frac{\kappa(G) P_1(e_1) P_2(e_2) \dots P_k(e_k)}{k! \kappa(H)}$$

This is a generalization of the algorithm of [14], in that we allow a weighting factor $\rho \in [0, 1]$. (In the algorithm of [14], $\rho = 1$). Note that for $\rho = 0$, there is a uniform distribution on the new edge e_i, and so this reduces to the unweighted bottom-up algorithm of Section 3.2.

Experimental evidence in [14] suggested that this algorithm had better variance for estimating N_t for small values of t. In this section, we examine this algorithm rigorously and show an upper bound which is better than that of Section 3.2.

For any connected subgraph $H \subseteq G$, we define the notations for this section:

$$\lambda_H(e) = \kappa(H \cup e)/\kappa(H)$$
$$S_H = \sum_{e \in G-H} \lambda_H(e)^{-\rho}$$

and we define

$$H_i = T \cup e_1 \cup \dots \cup e_{i-1}$$

where T is the spanning tree selected at step (1) of this algorithm. Thus, we have $P_i(e) = \lambda_{H_i}(e)/S_{H_i}$ for $i = 1, \dots, k$.

We first show that this is an unbiased estimator. Any $T, e_1, \dots, e_k$ is selected with probability $p = \frac{1}{\kappa(G)} P_1(e_1) P_2(e_2) \dots P_k(e_k)$. Integrating over $T, e_1, \dots, e_k$, we compute the expected value:

$$\begin{aligned}\mathbf{E}[\widehat{N_t}] &= \sum_{T, e_1, \dots, e_k} p \times \frac{1}{\kappa(H) k! p} \\ &= \sum_H \sum_{\substack{T \subseteq H \\ e_1, \dots, e_k \in H - T}} \frac{1}{k! \kappa(H)} \\ &= \sum_H \frac{1}{k! \kappa(H)} \kappa(H) k! = N_t\end{aligned}$$

so the statistic is unbiased. Next, the mean square is given by

$$\begin{aligned}\mathbf{E}[\widehat{N_t}^2] &= \sum_H \frac{\kappa(G)}{k!^2 \kappa(H)^2} \sum_{\substack{T \subseteq H \\ e_1, \dots, e_k \in H - T}} S_{H_1} S_{H_2} \dots S_{H_k} \lambda_{H_1}(e_1)^\rho \dots \lambda_{H_k}(e_k)^\rho \\ &= \sum_H \frac{\kappa(G)}{k!^2 \kappa(H)^2} \sum_{\substack{T \subseteq H \\ e_1, \dots, e_k \in H - T}} S_{H_1} S_{H_2} \dots S_{H_k} \Big(\frac{\kappa(H_2)}{\kappa(H_1)}\Big)^\rho \Big(\frac{\kappa(H_3)}{\kappa(H_2)}\Big)^\rho \cdots \Big(\frac{\kappa(H)}{\kappa(H_k)}\Big)^\rho \\ &= \sum_H \frac{\kappa(G)}{k!^2 \kappa(H)^2} \sum_{\substack{T \subseteq H \\ e_1, \dots, e_k \in H - T}} S_{H_1} S_{H_2} \dots S_{H_k} \kappa(H)^\rho\end{aligned}$$

To interpret this quantity, consider the following random process. We select a connected subgraph H with t edges, uniformly at random among all such subgraphs. Next, we select a random spanning tree T of H and a random permutation of the remaining edges $e_1, \dots, e_k \in H - T$. We then output the random variable R given by

$$R = \frac{S_{H_1} \dots S_{H_k}}{k! \kappa(H)^{1-\rho}}.$$

We see now that $\mathbf{E}[\widehat{N_t}^2] \leq N_t \kappa(G) \mathbf{E}[R]$, and hence the relative variance of the weighted bottom-up estimate is given by

$$\mathrm{rv} \leq \frac{N_t \kappa(G) \mathbf{E}[R]}{N_t^2} \leq \frac{\kappa(G) \mathbf{E}[R]}{\binom{m'}{m-t}}$$

Thus, it suffices to estimate $\mathbf{E}[R]$. We will in fact show an upper bound on R. We begin with a simple estimate. For any graph H, we have $\lambda_H(e) \geq 1$. Thus $S_{H_i} \leq K - i$. Noting that $\rho \leq 1$ and that $\kappa(H) \geq 1$, we have $R \leq \binom{K}{k}$. This simple estimate leads to the same bound as in Section 3.2.

Lemma 4 improves on this estimate as follows:

▶ **Lemma 4.** *Let $\rho \in [0,1]$.*
Define the function

$$A_\rho(y) = \int_0^y (1 - (\frac{x}{1+x})^\rho) dx.$$

Then, for any connected subgraph $H \subseteq G$ with βn edges, we have $R \leq \exp(n(f_{B2} + o(1)))$, for

$$f_{B2} = \max_{\phi \in [0,\alpha/2-\beta]} -(1-\rho)\Big(l(\gamma) - l(\alpha/2-1) - l(1-\alpha/2+\gamma) - l(1+\phi) + l(\phi)\Big) \\ - l(\beta-1) + l\big(\alpha/2 - 1 - A_\rho(\phi)\big) - l\big(\alpha/2 - \beta - A_\rho(\phi)\big)$$

Proof. See Appendix B. ◀

This then implies that the relative variance of the bottom-up algorithm is bounded by $\text{rv} = \exp((f_{B2} + o(1))n)$. Note that to compute f_{B2} itself requires a numerical maximization over $\phi \in [0, \alpha/2 - \beta]$.

As before, for any given α we seek β^*, γ^* so that the resulting upper bound $\min(f_T, f_{B2})$ is maximized. This expression is too complicated for us to solve in closed form, or even to prove that all relevant functions have the appropriate smoothness to allow a rigorous numerical analysis. However, for any fixed $\rho \in [0,1]$ we can approximately solve this numerically. Using off-the-shelf numerical libraries, we optimize f_{B2} subject to $f_T = f_{B2}$. We can furthermore set ρ to *minimize* the resulting f_{B2}.

For any average degree α, we select an optimal parameter ρ^*. The following table shows various values of α as well as the corresponding ρ^* and χ_α:

α	ρ^*	χ_α	α	ρ^*	χ_α
3	0.71	1.26	12	0.84	2.01
4	0.74	1.41	15	0.85	2.15
6	0.79	1.62	20	0.87	2.34
8	0.81	1.78	30	0.89	2.64
10	0.83	1.90	40	0.90	2.87
11	0.83	1.96	45	0.91	2.97

For $\alpha \leq 11$, this algorithm is strictly faster than [3]; for $\alpha \leq 45$ this algorithm is faster than the polynomial-space variant of [3].

5 Practical Performance

One key advantage of this algorithm is that it can be used on real-world graphs up to hundreds of vertices. In this case, the worst-case analysis would indicate exponentially low accuracy. However, in practice the accuracy may be much better than this.

For our first test case, we generated Erdős-Renyi graphs of average degree 10 and ran the algorithm as specified in Appendix A. (Qualitatively similar results are seen for other edge densities). We tabulate the estimated relative error as well as the running time of a single iteration. For the most part, implementing this algorithm requires only slight modifications to the codes of [14],[15]; see these for more details. Figure 1 lists the estimated relative error of this algorithm.

The relative variance is clearly growing exponentially with n, but the rate of growth (about 1.05^n) is much slower than the bound of 1.89^n as indicated in the worst-case analysis. Hence for graphs of moderate size $n \approx 200$ this algorithm remains quite practical.

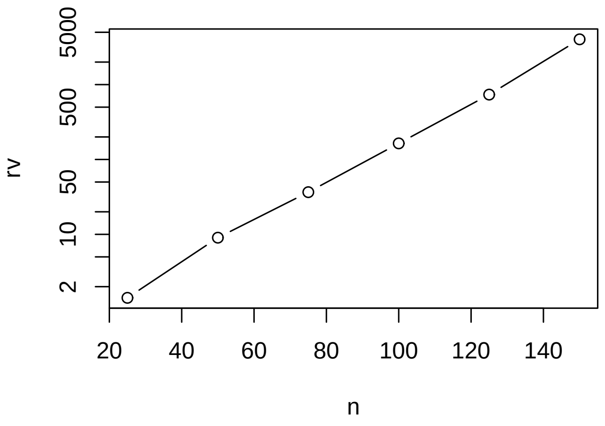

Figure 1 Relative error for Erdős-Renyi graphs. Note logarithmic vertical scale.

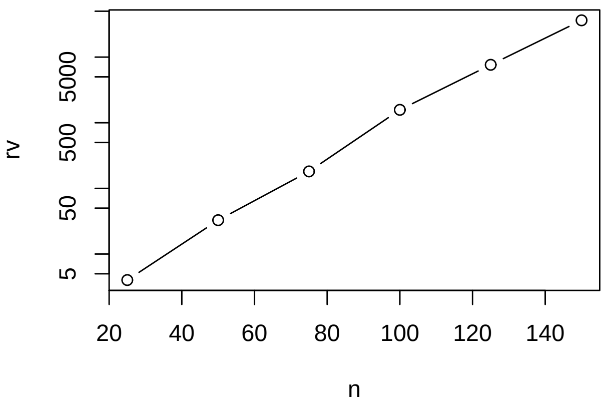

Figure 2 Relative error for Barabasi-Albert graphs. Note logarithmic vertical scale.

A similar result was seen for Barabasi-Albert random graphs, again of average degree 10, as shown in Figure 2.

The relative variance is significantly higher in this case, but the rate of increase remains slowly exponential, about 1.075^n.

The running times of these algorithms are relatively modest, and growing at a rate of about $n^{1.5}$, as indicated in Figure 3.

Recall that to achieve a relative error ϵ, we must repeat this algorithm for a number of samples $T = \text{rv} \times \epsilon^{-2}$. In order to achieve a relative error of say 10% on the Erdős-Renyi graph with $n = 150$, we would need to run for approximately 200000 iterations; this would entail a running time of about 300 seconds. (And furthermore this work could be completely parallelized). Hence this algorithm provides a quite practical method for estimating graph reliability on medium-scale graphs.

By way of comparison, [13] implemented an optimized version of an algorithm to exactly compute the Tutte polynomial. This program requires days of computations for graphs with only ~ 20 vertices and ~ 100 edges.

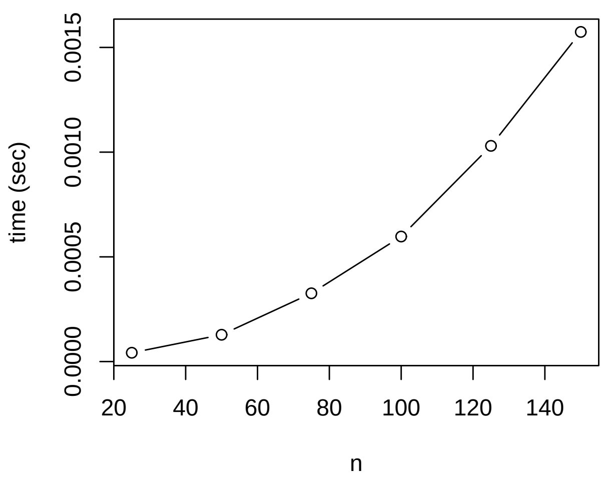

Figure 3 Running time for a single sample on an Erdős-Renyi graph.

5.1 Possible further improvements

The parameters (specifically, the choice of β) suggested by our worst-case analysis are not optimal for these sample graphs. By choosing parameters better, we could reduce the error by orders of magnitude. There seem to be three main reasons these bounds are not tight in practice. First, the top-down algorithm of [15] is much more accurate than the simple top-down algorithm we analyze in this paper.

Second, our estimate for the accuracy of the bottom-up algorithm is too conservative. The bottom-up error should be discounted by a factor of $\mathbf{E}[1/\kappa(H)]$, where H is the subgraph chosen by the bottom-up algorithm. In the worst case, the expected value of this term might be very large if some subgraphs H have many spanning trees and some have few. In practice, all the subgraphs H tend to have about the same number of spanning trees, and so $\mathbf{E}[1/\kappa(H)]$ is small.

Third, our method of setting the parameter ρ depends on estimating $\kappa(H \cup e_1 \cdots \cup e_i)$ where H is a subgraph of G and e_i are edges in $G - H$. It is currently an open problem in graph theory to determine tight bounds in this case. We are forced to use an upper bound for $\kappa(H \cup e_1 \cdots \cup e_i)$ which is much larger than necessary. This causes us to set ρ to an excessively large value.

6 Conclusion

We have shown exponential bounds on the relative variance of three SIS algorithms for estimating the graph reliability polynomial. These bounds are simple computable functions of G. By choosing the algorithm which minimizes the upper bound, we define a hybrid algorithm with worst-case relative variance $O(\chi_\alpha^n)$. Hence with $O(\chi_\alpha^n/\epsilon^2)$ work, one can, with high probability, estimate the graph reliability polynomial to relative error ϵ. Although this is exponential, we believe it is the fastest known algorithm for estimating the graph reliability polynomial when the average degree α is small.

Note that this bound on relative variance depended on bounding the number of spanning trees of certain sparse graphs. As this is an open problem in graph theory, the bounds we use are far from tight. It is likely that the true behavior of this algorithm is much better than the indicated values of χ_α.

In practice, these SIS algorithms tend to exhibit exponential relative variance on real-world graphs [14], [15]; however, the errors increase much slower than the worst-case analysis predicts. Hence, on many medium-scale graphs ($n \approx 200$) these algorithms can give a quite practical approach to estimate the graph reliability. In these cases exact, exponential-time algorithms such as [3] are absolutely infeasible.

Acknowledgments. Thanks to Isabel Beichl and Aravind Srinivasan for helpful comments and revisions.

References

1 Michael O. Ball and J. Scott Provan. Bounds on the reliablity polynomial for shellable independence systems. *SIAM Journal on Algebraic and Discrete Methods*, 3:166–181, 1982.

2 Isabel Beichl, Brian Cloteaux, and Francis Sullivan. An approximation algorithm for the coefficients of the reliability polynomial. *Congressus Numerantium*, 197:143–151, 2010.

3 Andreas Bjorklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Computing the Tutte polynomial in vertex-exponential time. *Foundations of Computer Science (FOCS)*, pages 677–686, 2008.

4 Adams L. Buchsbaum and Milena Mihail. Monte Carlo and Markov chain techniques for network reliability and sampling. *Networks*, 25:117–130, 1995.

5 Shiri Chechik, Yuval Emek, Boaz Patt-Shamir, and David Peleg. Sparse reliable graph backbones. *Automata, Languages, and Processing*, pages 261–272, 2010.

6 Andrew Chen. On graphs with large numbers of spanning trees. *PhD dissertation for Michigan State University Department of Computer Science*, 2005.

7 Fan RK Chung and Ronald L. Graham. On the cover polynomial of a digraph. *Journal of Combinatorial Theory, Series B*, 65:273–290, 1995.

8 Charles J. Colbourn, Bradley M. Debroni, and Wendy J. Myrold. Estimating the coefficients of the reliability polynomial. *Congress Numerantium*, 62:217–223, 1988.

9 Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlén. Exponential time complexity of the permanent and Tutte polynomial. *ACM Transactions on Algorithms*, 10-4, 2014.

10 George S. Fishman. A Monte Carlo sampling plan for estimating network reliability. *Operations Research*, 34:581–594, 1986.

11 Heidi Gebauer and Yoshio Okamoto. Fast exponential-time algorithms for the forest counting in graph classes. *Theory of Computing Australasian Symposium*, 65:63–69, 2007.

12 Leslie Ann Goldberg and Mark Jerrum. Inapproximability of the Tutte polynomial. *Information and Computation*, 206-7:908–929, 2008.

13 Gary Haggard, David J. Pearce, and Gordon Royle. Computing Tutte polynomials. *ACM Transactions on Mathematical Software*, 37-3 Article # 24, 2010.

14 David G. Harris, Francis Sullivan, and Isabel Beichl. Linear algebra and sequential importance sampling for network reliability. *Winter Simulation Conference*, 2011.

15 David G. Harris, Francis Sullivan, and Isabel Beichl. Fast sequential importance sampling to estimate the graph reliability polynomial. *Algorithmica*, 68-4:916–939, 2014.

16 David Karger. A randomized fully polynomial time approximation scheme for the all terminal network reliability problem. *SIAM Journal on Computing*, 29:11–17, 1996.

17 Joseph B. Kruskal. The number of simplices in a complex. *Mathematical Optimization Techniques*, 1963.

18 Marco Laumanns and Rico Zenklusen. High-confidence estimation of small s-t reliabilities in directed acylic networks. *Networks*, 57-4:367–388, 2011.

19 Wendy Myrvold. Counting k-component forests of a graph. *Networks*, 22:647–652, 1992.

20 J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12:777–788, 1983.
21 Kyoko Sekine, Hiroshi Imai, and Seiichiro Tani. Computing the Tutte polynomial of a graph of moderate size. *Lecture Notes in Computer Science*, 1004:224–233, 1995.
22 David Bruce Wilson. Generating random spanning trees more quickly than the cover time. *ACM Symposium on Theory of Computing (STOC)*, pages 296–303, 1996.

A Full algorithm

For completeness, we include a pseudo-code of the entire hybrid algorithm. Suppose we are given a graph G, and wish to estimate N_t up to relative error ϵ.

```
1. Compute $\alpha = m/n$ and $\beta = t/n$.
2. Using the Kirchoff formula, compute the number of spanning trees $\kappa(G)$. Find $m'$ such that $\binom{m'}{m-n} = \kappa(G)$ and let $\gamma = m'/n$.
3. Compute the bound on the top-down algorithm $f_T$.
4. Find the critical $h^*$ for $r(h, \alpha, \beta, \gamma)$. Use this to obtain the bottom-up bound $f_{B2}$.
5. If $f_T < f_{B2}$, draw the following statistic $F_1$ for $T = \exp(n(f_T + c))/\epsilon^2$ iterations:
    6. Select a subgraph $H \subseteq G$ uniformly at random among subgraphs with $t$ edges.
    7. Check if $H$ is connected
    8. If $H$ is connected set $F_1 = \binom{m}{t}$ else set $F_1 = 0$.
9. Else if $f_{B2} \leq f_T$, draw the following statistic $F_2$ for $T = \exp(n(f_{B2} + c))/\epsilon^2$ iterations:
    10. Select a spanning tree $H$ uniformly at random
    11. Successively select edges $e_1, \dots, e_k$ to add to $H$. At stage $i$, select edge $e_i$ with probability $P_i$ given by
            $$P_i(e') \propto 1/\kappa(H \cup e_1 \dots \cup e_{i-1} \cup e')^{\rho^*}$$
    12. Estimate
            $$F_2 = \frac{\kappa(G) P_1(e_1) P_2(e_2) \dots P_k(e_k)}{k! \kappa(H)}$$
13. Average the $T$ samples of the appropriate statistic and output this sample mean.
```

This algorithm estimates N_t within relative error ϵ with probability at least $3/4$ for n sufficiently large; furthermore, the worst-case running time of this algorithm is $(\chi_\alpha + o(1))^n/\epsilon^2$.

In practice, we use the algorithm of [2],[15] for the top-down estimation instead of the indicated steps (5) — (8). It is almost as fast as the simple Monte Carlo top-down estimation, and it is as least as accurate (in the worst-case) while being much more accurate in practice.

B Proof of Lemma 4

The heart of Lemma 4 is to show an upper bound on the quantity $S_H = \sum_{e \in G-H} \lambda_H(e)^{-\rho}$. We begin with two elementary propositions concerning the number of spanning trees in various subgraphs.

▶ **Proposition 5.** *For any graph H and edges $e_1, \dots, e_i \notin H$, we have*

$$\kappa(H \cup e_1 \dots \cup e_i) \leq \kappa(H) \binom{n-1+i}{n-1}$$

Proof. Any spanning tree T of $H \cup e_1 \dots \cup e_i$ may be formed as follows: choose a spanning tree T' of H, add the edges $e_1, \dots, e_i$, and extract a spanning tree T of $T' \cup e_1 \cup \dots \cup e_i$. ◀

We note that Proposition 5 can be improved exponentially. However, the formulas are quite complicated and the improvement is slight, so we elect to take this simple estimate. As discussed in [6], the state of research is very poor for estimating $\kappa(H \cup e_1 \cdots \cup e_i)$, even when H is itself a spanning tree. Preliminary results for small graphs seem to indicate that the true upper bound is much smaller than $\kappa(H)\binom{n-1+i}{n-1}$. If so, this would lead to much better bounds on the behavior of our algorithm.

▶ **Proposition 6.** *For any graph H and edges $e_1, \dots, e_i \notin H$, we have*

$$\kappa(H \cup e_1 \cdots \cup e_i) \leq \lambda_H(e_1) \cdots \lambda_H(e_i)$$

Proof. It suffices to show that $\lambda_{H \cup e'}(e) \leq \lambda_H(e)$ for any $H, e, e' \in G - H$. We recall the Kirchoff matrix-tree theorem used to count the number of spanning trees of a graph. Let A_G be the adjacency matrix of G, and let D be a diagonal matrix whose ith entry is the degree of vertex v_i. The Kirchoff formula states that $\kappa(G) = \det(D - A_G)_{11}$, the minor of $D - A_G$ obtained by removing the first row and column.

When we update H by adding edge e', we must update λ_H to the new $\lambda_{H \cup e'}$. For any edge $e = \langle i, j \rangle$ we define δ_e to be the column vector is $+1$ in coordinate i, is -1 in coordinate j, and is zero elsewhere. Observe that when edge e' is added to G, the matrix L changes by $\delta_{e'} \delta_{e'}^T$:

$$L_{H \cup e'} = L_H + \delta_{e'} \delta_{e'}^T.$$

Now let us examine how to update λ:

$$\begin{aligned}
\lambda_{H \cup e'}(e) &= \kappa(H \cup e \cup e')/\kappa(H \cup e') \\
&= \det(L_{H \cup e \cup e'})/\det(L_{H \cup e'}) \\
&= \det(L_H + \delta_e \delta_{e'}^T + \delta_e \delta_e^T)/\det(L_H + \delta_{e'} \delta_e^T) \\
&= \det\Big(I + (L_H + \delta_{e'} \delta_{e'}^T)^{-1} \delta_e \delta_e^T\Big) \\
&= 1 + \delta_e^T (L_H + \delta_{e'} \delta_{e'}^T)^{-1} \delta_e \\
&= 1 + \delta_e^T \big(L_H^{-1} - \frac{uu^T}{1 - \delta_e^T u}\big) \delta_e \qquad \text{where } u = L_H^{-1} \delta_{e'} \\
&= 1 + \delta_e^T L_H^{-1} \delta_e - \frac{(\delta_e^T u)^2}{1 - \delta_e^T u} \\
&= \lambda_H(e) - \frac{(\delta_e^T u)^2}{\lambda_H(e')} \leq \lambda_H(e)
\end{aligned}$$

◀

▶ **Proposition 7.** *Suppose $H \subseteq G$ is a subgraph with t edges. Suppose $s \in \mathbf{Z}_+$ satisfies $\kappa(H)\binom{n-1+s}{n-1} < \kappa(G)$. Then $s \leq m - t$.*

Proof. Suppose that $s > m - t$, then we would have:

$$\begin{aligned}
\frac{\kappa(G)}{\kappa(H)\binom{n-1+s}{n-1}} &< \frac{\kappa(G)}{\kappa(H)\binom{n-1+m-t}{n-1}} \\
&\leq \frac{\kappa(H)\binom{n-1+(m-t)}{n-1}}{\kappa(H)\binom{n-1+m-t}{n-1}} \qquad \text{by Proposition 5} \\
&\leq 1
\end{aligned}$$

contradicting our hypothesis on s. ◀

We can combine Propositions 5 and 6 to bound S_H for subgraphs $H \subseteq G$:

▶ **Lemma 8.** *Let $H \subseteq G$ be a connected subgraph with $t = n - 1 + k$ edges. Suppose $s \in \mathbf{Z}_+$ satisfies $\kappa(H)\binom{n-1+s}{n-1} < \kappa(G)$. Define*

$$A_\rho(y) = \int_0^y (1 - (\frac{x}{1+x})^\rho)dx$$

Then we have

$$\sum_{e \in G-H} \lambda_H(e)^{-\rho} \leq K + 1 - k - nA_\rho(s/n)$$

Proof. Observe that by Proposition 7, we have $s \leq m - t$.

Let $e_1, \dots, e_{m-t}$ enumerate the edges of $G - H$ sorted by decreasing order of λ_H, so that $\lambda_H(e_1) \geq \lambda_H(e_2) \geq \dots \geq \lambda_H(e_{m-t})$. To simplify the notation, write $\lambda_i = \lambda_H(e_i)$. By Propositions 5, 6, for any $i \leq m - t$, we have that

$$\begin{aligned}\kappa(G) &= \kappa(H \cup e_1 \dots \cup e_i \cup e_{i+1} \cup \dots \cup e_{m-t}) \\ &\leq \kappa(H \cup e_{i+1} \dots \cup e_{m-t})\binom{n-1+i}{n-1} \\ &\leq \kappa(H)\lambda_{i+1} \dots \lambda_{m-t}\binom{n-1+i}{n-1}\end{aligned}$$

Define $\lambda_{m-t+1} = 1$. Then $\vec{\lambda}$ satisfies the following system of constraints for $i = 1, \dots, m-t$:

$$\lambda_i \dots \lambda_{m-t} \geq \frac{\kappa(G)}{\kappa(H)\binom{n-2+i}{n-1}} \qquad \text{(Constraint } C_i\text{)}$$
$$\lambda_i \geq \lambda_{i+1}$$

Hence, it suffices to maximize on $S' = \sum_{i=1}^{m-t} \lambda_i^{-\rho}$ subject to these constraints. By compactness, such a maximum exists.

We first claim that in any such maximum, $\vec{\lambda}$ must satisfy $\lambda_i > \lambda_{i+1}$ for $i = 1, \dots, s$ strictly. Suppose that we have a block of equalities of the form $\lambda_i = \dots = \lambda_k$, where $i \leq s$ is minimal and k is maximal. We assume for simplicity that $i > 1$ (the case in which $i = 1$ is essentially identical.) Let $\eta = \lambda_i = \dots = \lambda_k$.

There are two ways in which k could be maximal. First, it might be that $k = m - t + 1$. In this case, we have $\lambda_i = \dots = \lambda_{m-t} = 1$. But then constraint C_i states that

$$1 \geq \frac{\kappa(G)}{\kappa(H)\binom{n-2+i}{n-1}}$$

which implies that

$$\kappa(G) \leq \frac{\kappa(H)}{\binom{n-2+i}{n-1}} \leq \frac{\kappa(H)}{\binom{n-2+s}{n-1}}$$

which contradicts the definition of s.

The other case is that we have $\lambda_k > \lambda_{k+1}$ for $k \leq m - t$. We claim that in this case, it must be that constraints $C_{i+1}, \dots, C_k$ are slack. For, suppose that constraint C_j is tight for some j in the range $i + 1, \dots, k$. Collecting all the terms other than $\lambda_{j-1}, \lambda_j, \lambda_{j+1}$ into a single constant c gives us the constraints:

$$\binom{n-2+(j-1)}{n-1}\eta^2 \geq c \qquad (C_{j-1})$$

$$\binom{n-2+j}{n-1}\eta = c \qquad (C_j)$$
$$\binom{n-2+j+1}{n-1} \geq c \qquad (C_{j+1})$$

(Note that constraint (C_j) is an equality.)

From constraint C_{j-1}, we can eliminate η to obtain that

$$c \geq \frac{\binom{n-2+j}{n-1}^2}{\binom{n-2+(j-1)}{n-1}}$$

and, substituting this into constraint C_{j+1} we obtain:

$$\frac{\binom{n-2+(j+1)}{n-1}\binom{n-2+(j-1)}{n-1}}{\binom{n-2+j}{n-1}^2} \geq 1$$

which reduces to

$$\frac{(j+n-1)(j-1)}{j(j+n-2)} \geq 1$$

which is a contradiction.

We have shown that if $\lambda_i = \cdots = \lambda_k$ then constraints $C_{i+1}, \ldots, C_k$ must be slack. Now divide λ_k by δ and multiply λ_i by δ for some $\delta > 1$. For δ sufficiently small, this does not change the sorted order of $\lambda_1, \ldots, \lambda_{m-t}$. Furthermore, this only affects the constraints $C_{i+1}, \ldots, C_k$, which are slack, and thus for δ sufficiently small all constraints remain satisfied. As $\lambda_i = \lambda_k$ this modification increases S', which is a contradiction.

We have thus shown that $\lambda_i > \lambda_{i+1}$ for $i = 1, \ldots, s$.

We next claim that all of the constraints $C_1, \ldots, C_s$ are tight. For, if C_i was slack for some $i \geq 2$, then we could multiply λ_{i-1} by δ and divide λ_i by δ for sufficiently small $\delta > 1$. As $\lambda_{i-2} > \lambda_{i-1} > \lambda_i > \lambda_{i+1}$, for δ sufficiently small this does not affect the sorted order of $\vec{\lambda}$, preserves all constraints, and increases S'. (For $i = 1$, simply divide λ_i by δ.)

We have now shown that when S' is maximized then all constraints $C_1, \ldots, C_s$ must be tight. Dividing constraint C_i by C_{i+1} yields $\lambda_i = \frac{n+i-1}{i}$ for $i = 1, \ldots, s-1$. We also must have $\lambda_i \geq 1$ for $i = s, \ldots, m-t$, so we have

$$\begin{aligned}
S_H \leq S' &\leq (m-t-s+1) + \sum_{j=1}^{s-1} (\frac{n+j-1}{n})^{-\rho} \\
&\leq (K-k-s+1) + \int_{j=0}^{s} (\frac{j}{n+j})^{\rho} dj \\
&= (K-k-s+1) + n\int_{x=0}^{s/n} (\frac{x}{1+x})^{\rho} dx \qquad \text{setting } x = j/n \\
&= K+1-k-nA_\rho(s/n).
\end{aligned}$$

◀

▶ **Corollary 9.** *Suppose S satisfies $\kappa(H)\binom{n-1+s}{n-1} < \kappa(G)$. Let $H \subseteq G$ be a subgraph of G and T a spanning tree of H and $e_1, \ldots, e_k$ enumerate the edges of $H - T$ (in any order). Then for $i = 1, \ldots, k$ we have*

$$S_{T \cup e_1 \cdots \cup e_{i-1}} \leq K + 1 - i - nA_\rho(s/n)$$

Proof. By Lemma 8 we have $S_{H_i} \leq K + 1 - i - nA_\rho(s_i/n)$ where s_i is maximal such that $\kappa(H_i)\binom{n-1+s_i}{n-1} < \kappa(G)$. Observe that $\kappa(H_i) \leq \kappa(H)$ and so $s_i \geq s$. Thus $S_{H_i} \leq K + 1 - i - nA_\rho(s/n)$ as claimed. ◀

We now pass to the limit, bounding the asymptotic growth of R.

▶ **Lemma 10.** *Let $\rho \in [0,1]$.*
Define the function

$$A_\rho(y) = \int_0^y (1 - (\frac{x}{1+x})^\rho) dx.$$

Then, for any connected subgraph $H \subseteq G$ with βn edges, we have $R \leq \exp(n(r + o(1)))$, for

$$\begin{aligned} r = \max_{\phi \in [0, \alpha/2 - \beta]} & -(1-\rho)\Big(l(\gamma) - l(\alpha/2 - 1) - l(1 - \alpha/2 + \gamma) - l(1+\phi) + l(\phi)\Big) \\ & - l(\beta - 1) + l\big(\alpha/2 - 1 - A_\rho(\phi)\big) - l\big(\alpha/2 - \beta - A_\rho(\phi)\big) \end{aligned}$$

Proof. Let $h = \ln(\kappa(H))/n$. Let s be maximal such that $\kappa(H)\binom{n-1+s}{n-1} < \kappa(G)$, and let $\phi = s/n$. By Corollary 9, we have $S_{H_i} \leq K - i - nA_\rho(\phi)$ for $i = 1, \ldots, k$. Also, by Proposition 7, we have $s \leq m - t$ and so $\phi \leq \alpha/2 - \beta$.

So

$$\ln R \leq \ln(K - nA(\phi)) + \cdots + \ln(K - k - nA(\phi)) - (\rho - 1)\ln \kappa(H) - \ln(k!)$$

We have that s is maximal such that $\binom{n-1+s}{n-1} < \kappa(G)/\kappa(H)$. Hence $\binom{n-1+s}{n-1}$ is within a factor of n of $\kappa(G)/\kappa(H)$, so that

$$\ln \kappa(G)/\kappa(H) - \ln n \leq \ln \binom{n-1+s}{n-1}$$

We apply Stirling's formula (2), and divide by n to get that

$$l(\phi + 1) - l(\phi) \geq \frac{\ln \kappa(G)/\kappa(H)}{n} - o(1)$$

Now note that, by definition of m' and γ, we have that

$$\kappa(G) \geq \binom{m'}{K} = \exp(n(l(\gamma) - l(\alpha/2 - 1) - l(1 - \alpha/2 + \gamma)) - o(1))$$

Thus it follows that

$$l(\phi + 1) - l(\phi) \geq l(\gamma) - l(\alpha/2 - 1) - l(1 - \alpha/2 + \gamma) - h - o(1)$$

Now we have:

$$\begin{aligned} \ln R &\leq \ln(K + 1 - nA(\phi)) + \cdots + \ln(K + 1 - k - nA(\phi)) - (1-\rho)\ln \kappa(H) - \ln(k!) \\ &\leq \ln \binom{K + 1 - nA(\phi + o(1))}{k} - (1-\rho)h + o(n) \\ &\leq \ln \binom{K - nA(\phi) + o(n)}{k} \\ &\qquad - (1-\rho)\Big(l(\phi+1) - l(\phi) - l(\gamma) + l(\alpha/2 - 1) + l(1 - \alpha/2 - \gamma)\Big) + o(n) \\ &\leq r + o(n) \qquad \text{as } \phi \in [0, \alpha/2 - \beta] \end{aligned}$$

◀