Monte Carlo and Markov Chain Techniques for Network Reliability and Sampling[1]

Adam L. Buchsbaum[2]
Milena Mihail[3]

Research Report CS-TR-391-92
September 1992

Abstract

We outline a heuristic to approximate various reliability related parameters of communications networks under link failures. Our scheme is based on Monte Carlo and Markov chain simulation techniques.

We shall present the ideas of these Monte Carlo and Markov chain techniques in terms of a specific reliability measure. However, the general method could be applicable to other reliability measures and, in fact, to other combinatorial problems.

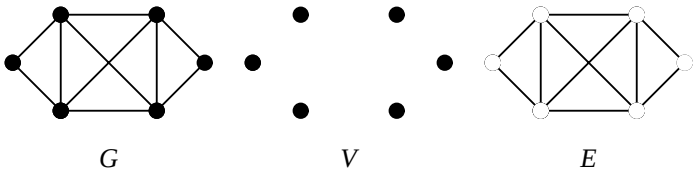
We present initial experimental results which suggest that our approach is efficient in the computational complexity sense (running time polynomial in the size of the input); furthermore, our results suggest practical applicability for medium size networks and single-edge parameters. As an example, we present the results of our experiments on a network that was posed for analysis by Applied Research at Bellcore: We estimated all single-edge parameters on a single DEC-5000 in less than 4 hours.

The software that supported our experiments involves approximately 3000 lines of C-code and is easy to adapt to other applications.

[1]Presented at the DIMACS Workshop on Computational Support for Discrete Mathematics, March, 1992.

[2]Supported by a Fannie and John Hertz Foundation fellowship, National Science Foundation Grant No. CCR-8920505, and the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) under NSF-STC88-09648. Work partially completed while this author was a summer intern at Bellcore.

[3]Bell Communications Research, Morristown, NJ 07960.



(a)

Spanning connected subgraph

Non-spanning disconnected subgraph

(Failing links are denoted by dotted lines.)

(b)

Figure 1: (a) An undirected graph G with vertex set V and edge set E; (b) A spanning connected subgraph and a non-spanning disconnected subgraph of G.

1 Introduction

1.1 Network Reliability

In theoretical computer science *network reliability* refers to the following problem (e.g. see [17, 18, 21] and also Figure 1a.)

Given as input is an undirected graph $G = (V, E)$ (the topology of a network), where $V = \{v_1, \ldots, v_n\}$ is the vertex set (the nodes of the network), and $E = \{e_1, \ldots, e_m\}$ is the edge set (the links between the nodes); hence, $e_k = \{v_i, v_j\}$ denotes an edge between vertices v_i and v_j. Associated with each edge e_k there is a probability p_k that e_k is operational, and, of course, $1 - p_k$ is the probability that e_k fails.

Say that the network is *connected* if and only if the set of edges that do not fail forms a *spanning connected subgraph* of G, i.e. a subgraph containing a path between any two nodes (see Figure 1b).

The aim is to compute

$$\begin{aligned} P &= \text{Probability the network is connected, and/or} \\ Q &= \text{Probability the network is not connected} = 1 - P. \end{aligned}$$

Henceforth we shall assume that $p_k = 1/2$, $1 \leq k \leq m$, since arbitrary probabilities can be simulated efficiently by series parallel graphs with edge failure probabilities 1/2 in well known ways; in fact, in the experiments described in Section 5, we did simulate the failure probabilities corresponding to a realistic network with series parallel subgraphs.

Letting $\mathcal{C}$ be the set of spanning connected subgraphs of G and $\mathcal{D} = 2^E \backslash \mathcal{C}$, we see:

$$P = \frac{|\text{Set of Spanning Connected Subgraphs of } G|}{|\text{Set of Subgraphs of } G|} = \frac{|\mathcal{C}|}{|2^m|}, \tag{1}$$

$$Q = \frac{|\text{Set of non-Spanning or Disconnected Subgraphs of } G|}{|\text{Set of Subgraphs of } G|} = \frac{|\mathcal{D}|}{|2^m|}. \tag{2}$$

Thus, equivalently the aim is to compute $|\mathcal{C}|$ and/or $|\mathcal{D}|$.

Obviously $|\mathcal{C}|$ and $|\mathcal{D}|$, and hence P and Q, can be computed by exhaustive search of all 2^m subgraphs of G and enumeration of those subgraphs that are in $\mathcal{C}$ (for P) or those subgraphs that are in $\mathcal{D}$ (for Q). However, this approach has running time at least 2^m and is therefore computationally inefficient. Furthermore, it is well known that, in general, computing P and/or Q *exactly* cannot

be achieved by efficient computational methods unless very unlikely collapses of complexity classes occur. In particular, computing P and/or Q exactly is NP-hard; more specifically it is $\#P$-complete, which means harder than a whole hierarchy of hard problems, the "easiest" of which is the traveling salesman problem [10, 18, 20, 21].

In view of the above, we outline a *heuristic* to *approximate* P, Q, and various related parameters; the potential of this approach was first observed in [17]. Initial *experimental evidence* (as well as some underlying theory that has appeared elsewhere [9, 17]) suggests that our heuristic is *efficient*. Of course, computational complexity accepts as "efficient" any scheme that runs in time polynomial in the size of the input. This does not necessarily imply practical efficiency. Indeed, our scheme for approximating P and Q does not appear at this point immediately practical, at least not with the limited resources that we used in our experiments. For example, an algorithm with running time n^{10}, where n is the size of the input, is clearly impractical. However, polynomiality, even with prohibitingly high degree polynomials, has more often than not been the first step towards revealing those structural properties that can be further exploited to obtain practical efficiency. It should also be noted that progress in the area of polynomial time approximations for $\#P$-complete functions is very recent [2, 3, 6, 8, 9, 11, 12, 13, 15], etc. For example, the algorithm in [8] to approximate volumes had complexity n^{26}; it was improved in [15] to n^{16}; it is currently at n^{10} [2]; and for special cases it has complexity $n^{2.5}$ [13], which was improved very recently to n^2 [22]. In this sense, any polynomial-time approximation scheme is of interest even purely for its methodology.

On the positive side, from a practical point of view our heuristic for single-edge parameters seems *very practical for medium size networks*. In particular, we were able to analyze the network designed in [14] as a realistic example of a communications network in less than four hours on a single DEC-5000. For the above network we estimated the effect of removing a link upon the reliability of the network. Our experimental results appear intuitively correct.

The software that supported our experiments involves approximately 3000 lines of C-code and is easy to adapt to other applications. A guide to the developed software has been provided separately [5].

1.2 Other Reliability Measures and Parameters

In addition to the specific quantities P and Q that are discussed here, there are many other measures of reliability. For example:

1. *Two-terminal reliability*, in which we are interested in the probability of a path existing between two specific nodes s and t (also known to be $\#P$-complete).

2. *Node-reliability*, in which nodes, rather than edges, fail with certain probabilities (also known to be $\#P$-complete).

3. *Specified root-related problems*, for example how many nodes are, on the average, connected to a given root-node.

4. Problems similar to the above when there are *dependencies* between failures.

We have not yet tested our approach for such quantities, but in principle the method discussed here applies, and we hope to see further work on other measures.

1.3 Single-Edge Parameters

For a specific edge e_i, one might be interested in how crucial e_i is to the performance of the network. For example, we may want to measure how P or Q is affected when edge e_i is deleted from, or, equivalently, when edge e_i is added to a network. Consider the modified network not containing e_i: $G_{\bar{e}_i} = (V, E \setminus \{e_i\})$.

- Let $P_{\bar{e}_i}$ be the reliability of $G_{\bar{e}_i}$, and let $\mathcal{C}_{\bar{e}_i}$ be the set of spanning connected subgraphs of $G_{\bar{e}_i}$ ($\mathcal{D}_{\bar{e}_i}$ is defined analogously.) Then

$$P_{\bar{e}_i} = \frac{|\mathcal{C}_{\bar{e}_i}|}{2^{m-1}} = \frac{|\mathcal{C}_{\bar{e}_i}|}{|\mathcal{C}|}\frac{|\mathcal{C}|}{2^{m-1}} = 2\frac{|\mathcal{C}_{\bar{e}_i}|}{|\mathcal{C}|}P, \tag{3}$$

$$Q_{\bar{e}_i} = 2\frac{|\mathcal{D}_{\bar{e}_i}|}{|\mathcal{D}|}Q, \tag{4}$$

and let

$$r_{\bar{e}_i} = \frac{|\mathcal{C}_{\bar{e}_i}|}{|\mathcal{C}|} \quad \text{and} \quad s_{\bar{e}_i} = \frac{|\mathcal{D}_{\bar{e}_i}|}{|\mathcal{D}|}. \tag{5}$$

- Similarly, let P_{e_i} be the reliability when e_i is contracted (i.e., its endpoints are collapsed to a single point), and let $\mathcal{C}_{e_i}$ be the set of spanning connected subgraphs that contain e_i ($\mathcal{D}_{e_i}$ is defined accordingly.) Then

$$P_{e_i} = \frac{|\mathcal{C}_{e_i}|}{2^{m-1}} = 2\frac{|\mathcal{C}_{e_i}|}{|\mathcal{C}|}\frac{|\mathcal{C}|}{2^m} = 2\frac{|\mathcal{C}_{e_i}|}{|\mathcal{C}|}P, \tag{6}$$

$$Q_{e_i} = 2\frac{|\mathcal{D}_{e_i}|}{|D|}Q, \tag{7}$$

and let

$$r_{e_i} = \frac{|\mathcal{C}_{e_i}|}{|\mathcal{C}|} \quad \text{and} \quad s_{e_i} = \frac{|\mathcal{D}_{e_i}|}{|\mathcal{D}|}. \tag{8}$$

It is obviously important to be able to compute the parameters $r_{\bar{e}_i}$, $s_{\bar{e}_i}$, r_{e_i}, and s_{e_i}, since they determine how modifications to specific edges might affect the reliability parameters. Unfortunately it is known that computing these parameters is $\#P$-complete, but the approximations presented herein perform well experimentally for the above single-edge parameters. Furthermore, the approach seems practical for medium size networks (a few hundred edges) and in particular for the network of [14].

Note that computing the r's and the s's above is also useful for *constructing* reliable networks: Start with a simple network (e.g. a spanning tree). When a new edge is to be added, among various choices pick the one, say e_i, that minimizes $r_{\bar{e}_i}$, (hence $P = (2r_{\bar{e}_i})^{-1}P_{\bar{e}_i}$ is increased), maximizes $s_{\bar{e}_i}$, (hence $Q = (2s_{\bar{e}_i})^{-1}Q_{\bar{e}_i}$ is decreased), and so on. In this sense, our approximation schemes might be practical for construction purposes.

It is also clear, but beyond of the scope of this presentation, that there are many other modifications that apply to the above schemes to accommodate additional requirements. The main purpose of this report is to outline the general Markov chain/Monte Carlo method and present the initial experimental results.

The rest of the report is organized as follows: In Section 2 we give the definitions of various approximations. In particular, we stress that definitions with respect to relative, rather than absolute, error are not only stronger but also more appropriate for network-related parameters. In Section 3 we review the general Monte Carlo framework that formalizes algorithms to approximate P and Q with respect to (i) absolute error in all cases, and (ii) relative error in all cases, provided that certain sampling can be accomplished efficiently. As a simple special case, we get approximations for single-edge parameters. In Section 4 we discuss the Markov chain simulation technique for sampling. This is the key point of the approach, and it is what allows Algorithms 2, 3, and 4 in Section 3, as well as the single-edge parameter estimations, to yield efficient complexities. In Section 5 we give experimental results for Markov chain simulations. Simulations were performed for symmetric and random networks of varying density; the results suggest that the technique is efficient in all cases. In Section 6 we used the Markov chain technique to analyze a realistic communications network. Our experiments ran quickly to completion, and the results appear intuitively correct. Section 7 proposes further work.

2 Approximations

This section uses the following definitions:

- $G = (V, E)$, $|V| = n$, $|E| = m$, (an undirected graph, as in the introduction, where each edge fails with probability 1/2),
- $\epsilon : 0 < \epsilon < 1$ and $\delta : 0 < \delta < 1$, where, as usual, ϵ and δ are written in binary.

2.1 Absolute Error Approximations

- A *polynomial* (ϵ, δ)*-approximation algorithm* for P outputs

$$\tilde{P} : \Pr\left[\left|P - \tilde{P}\right| < \epsilon\right] > 1 - \delta, \tag{9}$$

  and runs in time polynomial in n, ϵ^{-1}, and $\log \delta^{-1}$.

- Similarly, a *polynomial* (ϵ, δ)*-approximation algorithm* for Q outputs

$$\tilde{Q} : \Pr\left[\left|Q - \tilde{Q}\right| < \epsilon\right] > 1 - \delta, \tag{10}$$

  and runs in time polynomial in n, ϵ^{-1}, and $\log \delta^{-1}$.

Note that the above approximations with respect to absolute error correspond to bit accuracy.

2.2 Relative Error Approximations

- A *fully polynomial* (ϵ, δ)*-approximation algorithm* for P outputs

$$\tilde{P} : \Pr\left[\left|P - \tilde{P}\right| < \epsilon P\right] > 1 - \delta, \tag{11}$$

  and runs in time polynomial in n, ϵ^{-1}, and $\log \delta^{-1}$. Equivalently, the algorithm might output an estimate for the number of spanning connected subgraphs

$$\tilde{C} : \Pr\left[\left||\mathcal{C}| - \tilde{C}\right| < \epsilon|\mathcal{C}|\right] > 1 - \delta,$$

  and then $\tilde{P} = \tilde{C}/2^m$ satisfies (1).

- Similarly, a *fully polynomial* (ϵ, δ)*-approximation algorithm* for Q outputs

$$\tilde{Q} : \Pr\left[\left|Q - \tilde{Q}\right| < \epsilon Q\right] > 1 - \delta, \tag{12}$$

  and runs in time polynomial in n, ϵ^{-1}, and $\log \delta^{-1}$. Equivalently, the algorithm might output an estimate for the number of non-spanning or disconnected subgraphs

$$\tilde{D} : \Pr\left[\left||\mathcal{D}| - \tilde{D}\right| < \epsilon|\mathcal{D}|\right] > 1 - \delta,$$

  and then $\tilde{Q} = \tilde{D}/2^m$ satisfies (2).

2.3 Comparison of Absolute versus Relative Error

Note that since $0 < P, Q < 1$, (11)$\Longrightarrow$(9) and (12)$\Longrightarrow$(10); hence approximations with respect to relative error are stronger than approximations with respect to absolute error. In fact, when P or Q are smaller than, say, $O(2^{-m})$, (11) and (12) can yield almost exact results for ϵ being a constant, which keeps the running time of the algorithm polynomial. On the other hand, in order for (9) and (10) to yield almost exact results, we must use $\epsilon = O(2^{-m})$, which produces an exponential blowup in the running time. Of course, when P and Q are larger than, say, a constant, polynomial and fully polynomial approximations are equivalent up to constant factors. So the crucial strength of fully polynomial approximations concerns very small values of P and Q.

To stress further the practical significance of approximating with respect to relative errors by fully polynomial algorithms for small values of P and Q, consider the following case: A network has, as typically desired, very small failure probability, say $Q = O(2^{-m})$. For constant ϵ, a fully polynomial approximation algorithm offers an efficient certificate that the network will operate successfully for the next $\Omega(2^m)$ time units with very high probability; i.e. it will fail only a small constant number of times. In contrast, a polynomial approximation algorithm can offer such an efficient certificate only for the next $O(\text{poly}(m))$ time units. Furthermore, it is not hard to see that approximating P (which is large) with respect to either absolute or relative error can only be used to approximate $Q = 1 - P$ with respect to absolute error and hence offers an efficient certificate of reliable behavior only for the next $O(\text{poly}(m))$ time units. Therefore, the only efficient certificates for the next $\Omega(2^m)$ time units can be obtained by fully polynomial approximations of Q.

In Sections 3 and 4 we propose approximation algorithms with respect to both absolute and relative errors, together with evidence that these algorithms are efficient.

Remark: At this point, the reader might notice that since ϵ and δ are written in binary, running times polynomial in ϵ^{-1} rather than $\log \epsilon^{-1}$ (analogous to $\log \delta^{-1}$) might be non-polynomial in the size of the input. However, it is well known and easy to verify that running times polynomial in $\log \epsilon^{-1}$ would yield exact solutions for P and Q under randomized algorithms, which would imply very unlikely complexity inclusions (namely, that NP, all of the polynomial hierarchy, and $\#P$ all have polynomial time randomized solutions and are thus in BPP; for example, they would imply a randomized polynomial time algorithm for the traveling salesman problem.) In this sense, polynomiality in ϵ^{-1} is the best possible.

We proceed to discuss the specific algorithms.

3 The Monte Carlo Framework

Before presenting the schemes to approximate P and Q for arbitrary values of P and Q and the single-edge parameters r and s, let us review some background from elementary statistics.

Let S be a set of cardinality s, and let A be a subset of S of cardinality a. Let $r = a/s$. Let $X_1, \ldots, X_N$ be independent $\{0,1\}$ random variables that are identically distributed according to r, i.e. $\Pr[X_k = 1] = r$ and $\Pr[X_k = 0] = 1 - r$. Let $\tilde{r} = \sum_k X_k / N$. Then, standard Bernstein-Chernoff bounds suggest (e.g. see [19]) that up to constant factors in the exponents:

$$\Pr[|\tilde{r} - r| < \epsilon] > 1 - e^{-\epsilon^2 N} \quad (13)$$

$$\Pr[|\tilde{r} - r| < \epsilon r] > 1 - e^{-\epsilon^2 r^2 N} \quad (14)$$

In view of the above and of definitions (1) and (2) for P and Q, the first straightforward approach to approximate P and/or Q might be as follows.

3.1 Simple Monte Carlo Algorithm

Let $G = (V, E)$, $|V| = n$, $|E| = m$, $0 < \epsilon, \delta < 1$ be the input as usual, and let $g(n)$ be a polynomial in n. Recall that $\mathcal{C}$ is the set of spanning connected subgraphs of G, and $\mathcal{D}$ is the set of either non-spanning or disconnected subgraphs of G.

begin Algorithm 1.

For absolute error approximations, let $N = \epsilon^{-2} \log \delta^{-1}$.

For relative error approximations, let $N = \epsilon^{-2} \log \delta^{-1} g^2(n)$.

Sampling Phase: Generate N subgraphs of G independently and uniformly at random.

I.e. each edge is present with probability $1/2$.

Let $Y = \{Y_1, \ldots, Y_N\}$ be these subgraphs.

Final Phase: Set $\tilde{P} = |Y \cap \mathcal{C}|/N$, and set $\tilde{Q} = |Y \cap \mathcal{D}|/N$.

end.

Theorem 3.1 **(i)** *Algorithm 1 is a polynomial (ϵ, δ)-approximation algorithm for P and Q.* **(ii)** *If $P > g^{-1}(n)$ then Algorithm 1 is a fully polynomial (ϵ, δ)-approximation algorithm for P.* **(iii)** *If $Q > g^{-1}(n)$ then Algorithm 1 is a fully polynomial (ϵ, δ)-approximation algorithm for Q.* **(iv)** *The running time for the absolute error case is $O(\epsilon^{-2} \log \delta^{-1} g_1(m) g_2(m))$ and for the relative error case is $O(\epsilon^{-2} \log \delta^{-1} g^2(n) g_1(m) g_2(m))$, where $g_1(m) = O(m)$ is the time to generate a random subgraph of G, and $g_2(m) = O(m)$ is the time to test connectivity.*

PROOF: (SKETCH) Follows immediately from (13) and (14). □

So the simple Monte Carlo scheme of Algorithm 1 is always efficient for approximations with respect to absolute errors, and for approximations with respect to relative errors when the values of P and Q are relatively large.

Unfortunately, in the crucial cases where either P or Q is smaller than $g^{-1}(n)$, and in view of the fact that (14) is almost tight, the simple Monte Carlo Algorithm 1 is bound to fail. For these as well as the general cases we consider the following (first observed in [17]).

3.2 Refined Monte Carlo Algorithm for P

- Let $e_1, \ldots, e_m$ be the edges of G.
- Let G_{e_1} be G with e_1 contracted; similarly, let $G_{e_1 \cdots e_k}$ be G with $e_1, \ldots, e_k$ contracted.
- Let $\mathcal{C}_{e_1}$ be the set of spanning connected subgraphs of G that contain e_1 (and notice that $\mathcal{C}_{e_1}$ is isomorphic to the set of spanning connected subgraphs of G_{e_1}), and similarly let $\mathcal{C}_{e_1 \cdots e_k}$ be the set of spanning subgraphs of $G_{e_1 \cdots e_{k-1}}$ that contain e_k (and notice that $\mathcal{C}_{e_1 \cdots e_k}$ is isomorphic to the set of spanning connected subgraphs of $G_{e_1 \cdots e_k}$).
- Let $r_k = |\mathcal{C}_{e_1 \cdots e_k}|/|\mathcal{C}_{e_1 \cdots e_{k-1}}|$. **Remark:** It is crucial to realize that $|\mathcal{C}_{e_1 \cdots e_k}| \geq |\mathcal{C}_{e_1 \cdots e_{k-1}} \setminus \mathcal{C}_{e_1 \cdots e_k}|$; hence $r_k \geq 1/2$, and thus (14) suggests that a few samples of $\mathcal{C}_{e_1 \cdots e_{k-1}}$ can approximate r_k accurately; indeed, this is precisely what Algorithm 2 below does.
- Let $\epsilon' = \epsilon/m$.
- Let $N = 4\epsilon'^{-2} \log \delta^{-1} \log m$.

begin Algorithm 2.

Sampling Phase: For $k = 1, \ldots, m$ do

Generate N spanning connected subgraphs of $G_{e_1 \cdots e_{k-1}}$ independently and uniformly at random among all such subgraphs of $G_{e_1 \cdots e_{k-1}}$.

Let $Z = \{Z_1, \ldots, Z_N\}$ be these subgraphs.

Set $\tilde{r}_k = |Z \cap \mathcal{C}_{e_1 \cdots e_k}|/N$.

Final Phase: Set $\tilde{C} = 1/\prod_{k=1}^{m} \tilde{r}_k$.

Set $\tilde{P} = \tilde{C}/2^m$.

end.

Theorem 3.2 **(i)** $\Pr\left[\left|\tilde{P}-P\right| < \epsilon P\right] > 1-\delta$ *for Algorithm 2 above, and the running time is* $O(m^3\epsilon^{-2}\log\delta^{-1}\log m g_3(m))$, *where* $g_3(m)$ *is the complexity of generating a random spanning connected subgraph of* G. **(ii)** *When* $g_3(m)$ *is a polynomial, Algorithm 2 is a fully polynomial* (ϵ,δ)*-approximation scheme for* P.

PROOF: (SKETCH) $\prod_{k=1}^m \tilde{r}_k \simeq \prod_{k=1}^m r_k = \frac{|\mathcal{C}_{e_1}|}{|\mathcal{C}|}\frac{|\mathcal{C}_{e_1e_2}|}{|\mathcal{C}_{e_1}|}\cdots\frac{|\mathcal{C}_{e_1\cdots e_m}|}{|\mathcal{C}_{e_1\cdots e_{m-1}}|} = \frac{1}{|\mathcal{C}|}$, since $\mathcal{C}_{e_1\cdots e_m}$ is isomorphic to G, and hence $|\mathcal{C}_{e_1\cdots e_m}| = 1$. The rest follows from (14) and the fact that $r_k \geq 1/2$, as noted above. □

So the refined Monte Carlo Algorithm 2 is efficient for approximations with respect to relative errors for all values of P, provided that the complexity $g_3(m)$ of sampling spanning connected subgraphs of G is a polynomial. In Section 4 we shall develop a Markov chain heuristic for this and other sampling problems. However before proceeding to this heuristic let us discuss some more algorithms.

3.3 Refined Monte Carlo Algorithm for Q

In particular, analogous to Algorithm 2 for approximating P, Q can be treated as follows:

- Let $G_{\bar{e}_1\cdots\bar{e}_k}$ be G with $e_1,\ldots,e_k$ deleted.
- Let $\mathcal{D}_{\bar{e}_1\cdots\bar{e}_k}$ be the subgraphs of $G_{\bar{e}_1\cdots\bar{e}_k}$ that are non-spanning or disconnected.
- Let $s_k = |\mathcal{D}_{\bar{e}_1\cdots\bar{e}_k}|/|\mathcal{D}_{\bar{e}_1\cdots\bar{e}_{k-1}}|$, and remark as before that $s_k \geq 1/2$.
- Let ϵ' and N be as for Algorithm 2.

begin Algorithm 3.
  Sampling Phase For $k = 1,\ldots,m$ do
    Generate N non-spanning or disconnected subgraphs of $G_{\bar{e}_1\cdots\bar{e}_{k-1}}$
      independently and uniformly at random among all such
      subgraphs.
    Let $Z' = \{Z'_1,\ldots,Z'_N\}$ be these subgraphs.
    Set $\tilde{s}_k = |Z' \cap \mathcal{D}_{\bar{e}_1\cdots\bar{e}_k}|/N$.
  Final Phase: Set $\tilde{D} = 1/\prod_{k=1}^m \tilde{s}_k$.
    Set $\tilde{Q} = \tilde{D}/2^m$.
end.

Theorem 3.3 **(i)** $\Pr\left[\left|\tilde{Q}-Q\right| < \epsilon Q\right] > 1-\delta$ *for Algorithm 3 above, and the running time is* $O(m^3\epsilon^{-2}\log\delta^{-1}\log m g_4(m))$, *where* $g_4(m)$ *is the complexity of generating a random non-spanning or disconnected subgraph of* G. **(ii)** *When* $g_4(m)$ *is a polynomial, Algorithm 3 is a fully polynomial* (ϵ,δ)*-approximation scheme for* Q.

PROOF: (SKETCH) $\prod_{k=1}^m \tilde{s}_k \simeq \prod_{k=1}^m s_k = \frac{|\mathcal{D}_{\bar{e}_1}|}{|\mathcal{D}|}\frac{|\mathcal{D}_{\bar{e}_1\bar{e}_2}|}{|\mathcal{D}_{\bar{e}_1}|}\cdots\frac{|\mathcal{D}_{\bar{e}_1\cdots\bar{e}_m}|}{|\mathcal{D}_{\bar{e}_1\cdots\bar{e}_{m-1}}|} = \frac{1}{|\mathcal{D}|}$, since $G_{\bar{e}_1\cdots\bar{e}_m}$ is the empty graph, and hence $|\mathcal{D}_{\bar{e}_1\cdots\bar{e}_m}| = 1$. The rest follows from (14) and the fact that $s_k \geq 1/2$, as noted above. □

So again, Algorithm 3 is efficient for all values of Q, provided that the sampling complexity $g_4(m)$ is a polynomial. In Section 4 we shall address this issue using the Markov chain technique.

3.4 Algorithm for Single Edge Parameters

Assuming that there are no bridge edges, i.e. edges whose removal disconnects the network, it can be shown that the following bounds hold for the parameters of (5) and (8):

$$r_{\bar{e}_i} \geq 1/m, \quad s_{\bar{e}_i} \geq 1/2, \quad r_{e_i} \geq 1/2, \quad s_{e_i} \geq 1/m. \tag{15}$$

This assumption is valid, furthermore, because in estimating the single-edge parameters, we can determine a priori that the removal of any bridge edge e_i results in $P_{\bar{e}_i} = 0$. Continuing, on input $G = (V, E)$ as usual, and ϵ, δ:

begin Single Edge Parameter Algorithm.
Set $N = \epsilon^{-2} \log \delta^{-1}$, for absolute errors.
Set $N = \epsilon^{-2} m^2 \log \delta^{-1}$, for relative errors.
Let $Z = \{Z_1, \ldots, Z_N\}$ be as in Algorithm 2.
Let $Z' = \{Z'_1, \ldots, Z'_N\}$ be as in Algorithm 3.
Set $\tilde{r}_{e_i} = |Z \cap \mathcal{C}_{e_i}|/N$, and $\tilde{s}_{e_i} = |Z' \cap \mathcal{D}_{e_i}|/N$.
Set $\tilde{r}_{\bar{e}_i} = |Z \cap \mathcal{C}_{\bar{e}_i}|/N$, and $\tilde{s}_{\bar{e}_i} = |Z' \cap \mathcal{D}_{\bar{e}_i}|/N$.
end.

Theorem 3.4 **(i)** *The above algorithm is a polynomial time approximation for single-edge parameters with respect to absolute errors, and a fully polynomial time approximation with respect to relative errors provided $g_3(m)$ and $g_4(m)$ are polynomials, i.e. the samples Z and Z' can be generated in polynomial time.* **(ii)** *The exact running times are $O(g_i(m)\epsilon^{-2} \log \delta)$ and $O(g_i(m)\epsilon^{-2} m^2 \log \delta)$ respectively, where $i \in \{3, 4\}$ as appropriate.*

Proof: (Sketch) Follows immediately from (13), (14), (15). □

3.5 A Matroid Related Scheme

We give one final example of a reliability algorithm. We treat the case that follows separately, because computing the cardinalities of the levels $\mathcal{C}^l$ below corresponds exactly to counting bases of matroids. Hence Algorithm 4 that follows is the most structured combinatorially, and perhaps therefore the most promising for theoretical analysis (see also [17].)

- Let $\mathcal{C}^l$ be the set of spanning connected subgraphs of G that contain exactly l edges ($n - 1 \leq l \leq m$).
- Let $\mathcal{C}^l_{e_1 \cdots e_k}$ be the set of connected subgraphs of $G_{e_1 \cdots e_k}$ that contain exactly $l - k$ edges (these are of course equivalent to the set of connected subgraphs of G that contain exactly l edges, k of which are $e_1, \ldots, e_k$).
- Let $r_{k,l} = \frac{|\mathcal{C}^l_{e_1 \cdots e_k}|}{|\mathcal{C}^l_{e_1 \cdots e_{k-1}}|}$, and notice that $r_{k,l} \geq 1/m$, since each subgraph in $\mathcal{C}^l_{e_1 \cdots e_{k-1}} \setminus \mathcal{C}^l_{e_1 \cdots e_k}$ can be associated with at least one subgraph in $\mathcal{C}^l_{e_1 \cdots e_k}$ by adding e_k and removing some edge that does not affect connectivity, and each subgraph in $\mathcal{C}^l_{e_1 \cdots e_k}$ can be associated with at most m subgraphs in $\mathcal{C}^l_{e_1 \cdots e_{k-1}} \setminus \mathcal{C}^l_{e_1 \cdots e_k}$.
- Let $\epsilon' = \epsilon/m^2$.
- Let $N = 2\epsilon'^{-2} \log \delta^{-1} m^2$.

begin Algorithm 4.
Sampling Phase: For $l = n - 1, \ldots, m$ do
For $k = 1, \ldots, l$ do

Generate N subgraphs in $\mathcal{C}^l_{e_1 \cdots e_{k-1}}$ independently and
uniformly at random.
Let $Z = \{Z_1, \ldots, Z_N\}$ be these subgraphs.
Set $\tilde{r}_{k,l} = |Z \cap \mathcal{C}^l_{e_1 \cdots e_k}|/N$.
Set $\tilde{C}^l = 1/\prod_{k=1}^{l} \tilde{r}_{k,l}$.
Final Phase: Set $\tilde{C} = \sum_{l=n-1}^{m} \tilde{C}^l$, and $\tilde{P} = \tilde{C}/2^m$.
end.

Theorem 3.5 (i) $\Pr\left[\left|\tilde{P} - P\right| < \epsilon P\right] > 1 - \delta$, *and the running time of level-refined Monte Carlo Algorithm 4 is* $O(m^8 \epsilon^{-2} \log \delta^{-1} g_5(m))$, *where* $g_5(m)$ *is the complexity of generating a random connected subgraph of* G *that contains a fixed number of edges.* **(ii)** *When* $g_5(m)$ *is a polynomial, Algorithm 4 is a fully polynomial* (ϵ, δ)*-approximation scheme for* P.

PROOF: (SKETCH) $\prod_{k=1}^{l} \tilde{r}_{k,l} \simeq \prod_{k=1}^{l} r_{k,l} = \frac{|\mathcal{C}^l_{e_1}|}{|\mathcal{C}^l|} \frac{|\mathcal{C}^l_{e_1 e_2}|}{|\mathcal{C}^l_{e_1}|} \cdots \frac{|\mathcal{C}^l_{e_1 \cdots e_l}|}{|\mathcal{C}^l_{e_1 \cdots e_{l-1}}|} = \frac{1}{|\mathcal{C}^l|}$. The rest follows from (14). □

We proceed to discuss how the Markov chain simulation technique might suggest polynomial sampling complexities for $g_3(m)$, $g_4(m)$, and $g_5(m)$, and hence also for Algorithms 2, 3, and 4.

4 The Markov Chain Simulation Technique for Sampling

The main idea of the Markov chain technique for sampling is as follows. Let S be the set to be sampled. Typically S will be exponentially large, e.g. the set of up to 2^m subgraphs of a given graph. Assuming that some initial element of S can be constructed efficiently, define a Markov chain on state space S such that:

1. Each step of the Markov chain corresponds to a small local perturbation and can be simulated efficiently.
2. The stationary distribution is the uniform over S.
3. Convergence to stationarity is "rapid," in that after poly($\log |S|$) steps, the "distance" of the chain from the uniform stationary distribution is negligible for all practical purposes.

Such a Markov chain is called *rapidly mixing* (see, e.g., [1] for technical details.)

Now the simulation of a rapidly mixing Markov chain can be used as an efficient sampling scheme for S, simply by simulating the chain for poly($\log |S|$) steps and using the final point of the chain as a "random" point of S. This method has been used extensively and with remarkable success in group theory and in statistics [7] and in several combinatorial optimization problems [2, 3, 6, 8, 11, 13, 15]. Here we consider using it in order to produce the samples required for Algorithms 2, 3, and 4, as follows (related schemes were also proposed in [17].)

4.1 Markov chain $\mathcal{MC}4$ for Algorithm 4

- Where G is a graph (G in general will be $G_{e_1 \cdots e_k}$), and for given l, construct some spanning connected subgraph $H(0)$ in $\mathcal{C}^l$.
- Where $H(t)$ is the state at time t, $H(t+1)$ is determined as follows:
  - Toss a fair coin. If "heads" then $H(t+1) = H(t)$.
  - If "tails" then pick e_i and e_j independently and uniformly at random from $\{e_1, \ldots, e_k\}$.
    * If $e_i \notin H(t)$, $e_j \in H(t)$, and $H(t) \cup \{e_i\} \setminus \{e_j\} \in \mathcal{C}^l$, then $H(t+1) = H(t) \cup \{e_i\} \setminus \{e_j\}$; else $H(t+1) = H(t)$.

Roughly, each step of $\mathcal{MC}4$ moves from one spanning connected subgraph to another by adding and removing one edge at a time; intuitively, one might expect that after a "few" such steps the subgraph becomes "random."

4.2 Markov Chain $\mathcal{MC}2$ for Algorithm 2 and Single Edge Parameters

- Where G is a graph (G in general will be $G_{e_1 \cdots e_k}$), construct some spanning connected subgraph $H(0)$ in $\mathcal{C}$.
- Where $H(t)$ is the state on time t, $H(t+1)$ is determined as follows:
  - With probability 1/2, $H(t+1) = H(t)$.
  - With probability 1/4, pick e_i and e_j independently and uniformly at random from $\{e_1, \ldots, e_k\}$.
    - If $e_i \notin H(t)$ and $e_j \in H(t)$, and $H(t) \cup \{e_i\} \setminus \{e_j\} \in \mathcal{C}$, then $H(t+1) = H(t) \cup \{e_i\} \setminus \{e_j\}$, else $H(t+1) = H(t)$.
  - With probability 1/4, pick e_i uniformly at random from $\{e_1, \ldots, e_k\}$.
    - If $e_i \in H(t)$ and $H(t) \setminus \{e_i\} \in \mathcal{C}$, then $H(t+1) = H(t) \setminus \{e_i\}$, else $H(t+1) = H(t)$.
    - If $e_i \notin H(t)$ then $H(t+1) = H(t) \cup \{e_i\}$.

4.3 Markov Chain $\mathcal{MC}3$ for Algorithm 3

- Where G is a graph (G in general will be $G_{\bar{e}_1 \cdots \bar{e}_k}$), construct some non-spanning or disconnected subgraph $H(0)$ in $\mathcal{D}$.
- Where $H(t)$ is the state on time t, $H(t+1)$ is determined as follows:
  - With probability 1/2, $H(t+1) = H(t)$.
  - With probability 1/4, pick e_i and e_j independently and uniformly at random from $\{e_1, \ldots, e_k\}$.
    - If $e_i \notin H(t)$ and $e_j \in H(t)$, and $H(t) \cup \{e_i\} \setminus \{e_j\} \in \mathcal{D}$, then $H(t+1) = H(t) \cup \{e_i\} \setminus \{e_j\}$, else $H(t+1) = H(t)$.
  - With probability 1/4, pick e_i uniformly at random from $\{e_1, \ldots, e_k\}$.
    - If $e_i \notin H(t)$ and $H(t) \cup \{e_i\} \in \mathcal{D}$, then $H(t+1) = H(t) \cup \{e_i\}$, else $H(t+1) = H(t)$.
    - If $e_i \in H(t)$ then $H(t+1) = H(t) \setminus \{e_i\}$.

4.4 Discussion of Markov Chains

It is not hard to see that Markov chains $\mathcal{MC}2$, $\mathcal{MC}3$, and $\mathcal{MC}4$, are symmetric, irreducible, and aperiodic; therefore they converge to the uniform distribution over state spaces $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{C}^l$ respectively.

What is the rate at which the above Markov chains converge to the uniform distribution? In particular, are these chains rapidly mixing, which would imply that they can be used as efficient sampling schemes for $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{C}^l$, thus yielding polynomial sampling and total complexities for $g_3(m)$, $g_4(m)$, and $g_5(m)$ in Theorems 3.1, 3.2, 3.3, 3.4, and 3.5?

Various measures of convergence have been used elsewhere (e.g. see [1, 7]); here we use the natural measure that fits our algorithmic purposes:

- For $\mathcal{MC}2$, where G is the graph in question (recall G is in general $G_{e_1 \cdots e_k}$), let

$$d_e(t) = \frac{|\{\tau \leq t : e \in H(\tau)\}|}{t} \longrightarrow_{t \to \infty} \frac{|\mathcal{C}_e|}{|\mathcal{C}|}. \tag{16}$$

- Where R is an integer "sampling rate," let $R\tau$ denote integer multiples of R. Let

$$d_e^R(t) = \frac{|\{R\tau \leq t : e \in H(R\tau)\}|}{\lfloor t/R \rfloor} \longrightarrow_{t \to \infty} \frac{|\mathcal{C}_e|}{|\mathcal{C}|}. \tag{17}$$

- Note that the above measures converge precisely to r_e defined in (8).

Clearly (17) converges faster than (16), and indeed, when R is the time required for the Markov chain to come arbitrarily close to stationarity, (17) should behave as if the $H(R\tau)$'s were independent and uniformly distributed over $\mathcal{C}$, which of course corresponds to almost accurate simulation of the sampling phase of Algorithm 2 and of r_e.

- Analogously, for $\mathcal{MC}3$, let

$$d_{\bar{e}}(t) = \frac{|\{\tau \leq t : \bar{e} \notin H(\tau)\}|}{t} \longrightarrow_{t \to \infty} \frac{|\mathcal{D}_{\bar{e}}|}{|\mathcal{D}|} \tag{18}$$

$$d_{\bar{e}}^R(t) = \frac{|\{R\tau \leq t : \bar{e} \notin H(R\tau)\}|}{\lfloor t/R \rfloor} \longrightarrow_{t \to \infty} \frac{|\mathcal{D}_{\bar{e}}|}{|\mathcal{D}|}. \tag{19}$$

- Notice that the above measures converge precisely to $s_{\bar{e}}$ as defined in (5).

- And for $\mathcal{MC}4$, let

$$d_{e,l}(t) = \frac{|\{\tau \leq t : e \in H(\tau)\}|}{t} \longrightarrow_{t \to \infty} \frac{|\mathcal{C}_e^l|}{|\mathcal{C}^l|} \tag{20}$$

$$d_{e,l}^R(t) = \frac{|\{R\tau \leq t : e \in H(R\tau)\}|}{\lfloor t/R \rfloor} \longrightarrow_{t \to \infty} \frac{|\mathcal{C}_e^l|}{|\mathcal{C}^l|}. \tag{21}$$

To the best of our knowledge, there is no formal mathematical upper bound for the rate at which any of (16)–(21) converge (except for (20) and (21) when $l = n - 1$, i.e. random walks on spanning trees, which were resolved in [4, 9].) However, there is theoretical evidence (such as categorization of $\mathcal{MC}2$, $\mathcal{MC}3$, and $\mathcal{MC}4$, in general very well structured classes, combined with detailed probability theory analysis) that, in the worst case, we can state the following:

Conjecture 4.1 *For $R = m^5$, $d_e^R(t)$ and $d_{e,l}^R(t)$ converge for all graphs and $d_{\bar{e}}^R(t)$ converges for "most"[4] graphs as if the sampled points $R\tau$ were independently and uniformly distributed on state spaces $\mathcal{C}$, $\mathcal{C}_{e,l}$, and $\mathcal{D}$ respectively.*

The $R = m^5$ bound follows by an analysis that suggests that total variation distances drop at least as fast as $2^m(1 - \Phi^2)^t$, and that the "conductance" Φ is such that $\Phi \geq 1/m^2$ for the Markov chains in question; see e.g. [16, 17] for the technical details. It should be pointed out that this analysis is not necessarily tight, and that the running times could be much faster.

Clearly, if Conjecture 4.1 is true in general, then Algorithms 2, 3, and 4 are fully polynomial (ϵ, δ)-approximation schemes for P and Q. The experiments that we have performed aimed mainly to (i) compare Conjecture 4.1 with the simulation of specific instances, and (ii) test the performance of the Markov chain technique for single-edge parameters.

In all the cases, rather than the $R = m^5$ security rate predicted by theory, rates $R = m^2$ or m^3 appeared sufficient. In some special cases, even m, or $n \log n$ rates also seemed adequate!

[4] For all graphs that appeared in our experiments, $d_{\bar{e}}^R(t)$ converged rapidly and correctly. However, we also know of examples of graphs for which the convergence of $d_{\bar{e}}^R(t)$ is provably slow. At this point, we do not have a precise characterization of classes of graphs for which $d_{\bar{e}}^R(t)$ converges rapidly or slowly.

Graph G	n	m	*Markov Chain*	*Size of State Space (Appr.)*	*Rate* R	*Number of Samples*	*Average Relative Difference between Markov chain Sampling and Exact Sampling or Related Parameter*
$K25$	25	300	$\mathcal{MC}4$ $l=24$	12^{25}	$n=25$	$4 \cdot 10^5$	$< .02$
$K25$	25	300	$\mathcal{MC}4$ $l=24$	12^{25}	$m=300$	$3 \cdot 10^5$	$< .02$
$K25$	25	300	$\mathcal{MC}4$ $l=24$	12^{25}	$m^2 \simeq 9 \cdot 10^4$	$8 \cdot 10^4$	$< .005$
$K25$	25	300	$\mathcal{MC}4$ $l=115$	3^{110}	$n=25$	$2 \cdot 10^5$	$< .01$
$K25$	25	300	$\mathcal{MC}4$ $l=115$	3^{110}	$m=300$	$3 \cdot 10^4$	$< .01$
$K25$	25	300	$\mathcal{MC}4$ $l=115$	3^{110}	$m^2 \simeq 9 \cdot 10^4$	10^4	$< .005$
$K25$	25	300	$\mathcal{MC}4$ $l=206$	1.5^{200}	$n=25$	$5 \cdot 10^4$	$< .005$
$K25$	25	300	$\mathcal{MC}4$ $l=206$	1.5^{200}	$m=300$	$8 \cdot 10^3$	$< .025$
$K25$	25	300	$\mathcal{MC}4$ $l=206$	1.5^{200}	$m^2 \simeq 9 \cdot 10^4$	$3 \cdot 10^3$	$< .01$
$K25$	25	300	$\mathcal{MC}2$	2^{300}	$n=25$	10^6	$< .005$
$K25$	25	300	$\mathcal{MC}2$	2^{300}	$m=300$	10^5	$< .015$
$K25$	25	300	$\mathcal{MC}2$	2^{300}	$m^2 \simeq 9 \cdot 10^4$	$5 \cdot 10^3$	$< .01$
$K25$	25	300	$\mathcal{MC}3$	–	$n=25$	10^6	$< .02$
$K25$	25	300	$\mathcal{MC}3$	–	$m=300$	10^5	$< .02$
$K25$	25	300	$\mathcal{MC}3$	–	$m^2 \simeq 9 \cdot 10^4$	$6 \cdot 10^3$	$< .02$

Table 1: Complete graph on 25 vertices.

5 Experimental Results for Convergence Rates

We tested the convergence rates of $\mathcal{MC}2$, $\mathcal{MC}3$, and $\mathcal{MC}4$ for symmetric networks of different densities and for random networks of different densities.

For symmetric networks we performed simulations for complete graphs (which are of course very dense), for hypercubes (which are of medium density), and for the grid-like torus (which is very sparse). We give three tables here, namely for the complete graph on 25 vertices ($K25$) in Table 1, for the hypercube in 6 dimensions on 64 vertices ($H6$) in Table 2, and for the 15×15 grid-like torus on 225 vertices ($T15$) in Table 3.

During the simulations we used three different sampling rates: n, m, and m^2. As expected, as the sampling rate grew larger, the chain was given a better chance to mix, and fewer samples were needed to achieve small errors.

Most importantly, for example for the complete graph, notice the very large sizes of the state spaces and yet the extremely small number of samples that were used to achieve small differences on the estimates of the Markov chain with respect to that of exact independent sampling (and other related parameters).

Similarly, for random graphs we performed simulations for various densities and sampling rates. In all cases $\mathcal{MC}4$ and exact independent sampling seem statistically indistinguishable after at most 10^6 samples. For $\mathcal{MC}2$ and $\mathcal{MC}3$ exact sampling was in error by no more than 1% after at most 10^6 samples.

Graph G	n	m	*Markov Chain*	*Rate* R	*Number of Samples*	*Average Relative Difference between Markov chain Sampling and Exact Sampling or Related Parameter*
$H6$	64	192	$\mathcal{MC}4$ $l=63$	$n=64$	$3 \cdot 10^5$	$< .02$
$H6$	64	192	$\mathcal{MC}4$ $l=63$	$m=192$	10^5	$< .02$
$H6$	64	192	$\mathcal{MC}4$ $l=63$	$m^2 \simeq 4 \cdot 10^3$	$6 \cdot 10^3$	$< .01$
$H6$	64	192	$\mathcal{MC}4$ $l=105$	$n=64$	$3 \cdot 10^4$	$< .02$
$H6$	64	192	$\mathcal{MC}4$ $l=105$	$m=192$	10^4	$< .02$
$H6$	64	192	$\mathcal{MC}4$ $l=105$	$m^2 \simeq 4 \cdot 10^3$	$3 \cdot 10^3$	$< .01$
$H6$	64	192	$\mathcal{MC}4$ $l=147$	$n=64$	$5 \cdot 10^3$	$< .005$
$H6$	64	192	$\mathcal{MC}4$ $l=147$	$m=192$	$2 \cdot 10^3$	$< .005$
$H6$	64	192	$\mathcal{MC}4$ $l=147$	$m^2 \simeq 4 \cdot 10^3$	$2 \cdot 10^3$	$< .005$
$H6$	64	192	$\mathcal{MC}2$	$n=64$	$3 \cdot 10^5$	$< .01$
$H6$	64	192	$\mathcal{MC}2$	$m=192$	10^5	$< .01$
$H6$	64	192	$\mathcal{MC}2$	$m^2 \simeq 4 \cdot 10^3$	$2 \cdot 10^4$	$< .005$
$H6$	64	192	$\mathcal{MC}3$	$n=64$	10^5	$< .02$
$H6$	64	192	$\mathcal{MC}3$	$m=192$	$5 \cdot 10^4$	$< .02$
$H6$	64	192	$\mathcal{MC}3$	$m^2 \simeq 4 \cdot 10^3$	$8 \cdot 10^3$	$< .02$

Table 2: Hypercube in 6 dimensions.

Graph G	n	m	*Markov Chain*	*Rate* R	*Number of Samples*	*Average Relative Difference between Markov chain Sampling and Exact Sampling or Related Parameter*
$T15$	225	450	$\mathcal{MC}4$ $l=224$	$n=225$	$3 \cdot 10^5$	$< .01$
$T15$	225	450	$\mathcal{MC}4$ $l=224$	$m=5 \cdot 10^4$	10^5	$< .01$
$T15$	225	450	$\mathcal{MC}4$ $l=224$	$m^2 \simeq 10^7$	$2 \cdot 10^3$	$< .005$
$T15$	225	450	$\mathcal{MC}4$ $l=299$	$n=225$	$2 \cdot 10^4$	$< .03$
$T15$	225	450	$\mathcal{MC}4$ $l=299$	$m=5 \cdot 10^4$	10^4	$< .02$
$T15$	225	450	$\mathcal{MC}4$ $l=299$	$m^2 \simeq 10^7$	$5 \cdot 10^3$	$< .005$
$T15$	225	450	$\mathcal{MC}4$ $l=374$	$n=225$	$3 \cdot 10^3$	$< .005$
$T15$	225	450	$\mathcal{MC}4$ $l=374$	$m=5 \cdot 10^4$	10^3	$< .02$
$T15$	225	450	$\mathcal{MC}4$ $l=374$	$m^2 \simeq 10^7$	10^3	$< .03$
$T15$	225	450	$\mathcal{MC}2$	$n=225$	$6 \cdot 10^4$	$< .01$
$T15$	225	450	$\mathcal{MC}2$	$m=5 \cdot 10^4$	$4 \cdot 10^4$	$< .01$
$T15$	225	450	$\mathcal{MC}2$	$m^2 \simeq 10^7$	$5 \cdot 10^3$	$< .025$
$T15$	225	450	$\mathcal{MC}3$	$n=225$	$2 \cdot 10^5$	$< .02$
$T15$	225	450	$\mathcal{MC}3$	$m=5 \cdot 10^4$	$8 \cdot 10^4$	$< .02$
$T15$	225	450	$\mathcal{MC}3$	$m^2 \simeq 10^7$	$4 \cdot 10^3$	$< .02$

Table 3: 15 by 15 grid-like torus.

Indicatively we include four plots in Figure 2: for random graphs on 50 vertices and edge probabilities $p = 1/5$, $1/2$, and $3/4$. Note that in Figure 2c the curve of exact sampling is missing, since for this case there do not exist known means to perform such sampling efficiently; however even in this case the fast convergence of the walk is implied by the plot.

6 Experimental Analysis of a Network

Consider the network in Figure 3a, posed in [14]. The labels on the edges represent lengths, and the assumption is that edges fail with probabilities proportional to their lengths.

For each edge we estimated the ratio

$$s_{\bar{e}_i} = \frac{|\mathcal{D}_{\bar{e}_i}|}{|\mathcal{D}|}$$

using $\mathcal{MC}3$ in order to see how the failure probability of the network would be altered if the edge e_i were removed; recall that (4) and (5) suggest that

$$Q_{\bar{e}_i} = 2 s_{\bar{e}_i} Q.$$

It is of course preferable to remove edges for which $s_{\bar{e}_i}$ is small. Clearly, edges (1,2), (2,3), (4,5), (5,6), and (9,6) should not be removed, because that disconnects the network. For the remaining edges, $s_{\bar{e}_i}$ were estimated as indicated in Figure 3b. We decided when the estimates were accurate enough so that no further sampling was needed by observing the convergence of the estimates until they seemed stable around a specific value. The total experiment ran to completion in less than 4 hours on a DEC-5000.

It is interesting to note how the results favor what is the intuitive expectation: that it is better to remove edges so that the network remains biconnected or is broken into (connected) biconnected components which are as large as possible. For example, edge (6,7) which leaves the network biconnected is given weight .3, while edges (14,21), (21,23), etc. which destroy biconnectivity are given weights approximately 40. Thus it is preferable to remove edge (6,7) rather than, say, (14,21), which is indeed the natural intuition.

7 Further Work

It would be of interest to (i) perform further simulations on real networks, (ii) apply our heuristics to construct reliable networks, (iii) improve upon the efficiencies of the existing schemes, (iv) extend the approach to other reliability measures and other problems, and (v) develop formally the underlying theory of the Markov chain method.

References

[1] D. Aldous. On the Markov chain simulation method for uniform combinatorial distributions and simulated annealing. *Probability in the Engineering and Informational Sciences*, 1(1):33–46, 1987.

[2] D. Applegate and R. Kannan. Samping and integration of near log-concave functions. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 156–63, 1991.

[3] A. Z. Broder. How hard is it to marry at random? (On the approximation of the permanent). In *Proc. 18th ACM Symp. on Theory of Computing*, pages 50–8, 1986.

[4] A. Z. Broder. Generating random spanning trees. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 442–7, 1989.

[5] A. L. Buchsbaum and M. Mihail. Software for simulating and analyzing Markov chains. Technical Report TM-ARH-020-376, Bellcore, 1991.

[6] P. Dagum, M. Luby, M. Mihail, and U. Vazirani. Polytopes, permanents, and graphs with large factors. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 412–21, 1988.

[7] P. Diaconis. Group theory and statistics. Lecture notes from a course taught at Harvard, 1982.

[8] M. Dyer, A. Frieze, and R. Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 375–81, 1989.

[9] T. Feder and M. Mihail. Balanced matroids. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 26–38, 1992.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completness.* W.H. Freeman and Company, New York, 1979.

[11] M. R. Jerrum and A. Sinclair. Conductance and the rapid mixing property for Markov chains: The approximation of the permanent resolved. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 235–44, 1988.

[12] R. M. Karp and M. Luby. Monte-Carlo algorithms for enumeration and reliability problems. In *Proc. 24th IEEE Symp. on Foundations of Computer Science*, pages 56–64, 1983.

[13] A. Karzanov and L. Khachiyan. On the conductance of order Markov chains. Technical report, Rutgers University, 1990.

[14] S. C. Liew and K. W. Lu. A framework for network survivability characterization. In *Proc. IEEE Int'l. Conf. on Communications*, pages 405–10, 1992.

[15] L. Lovász and M. Simonovis. The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 346–50, 1990.

[16] M. Mihail. Conductance and convergence of Markov chains—A combinatorial treatment of expanders—. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 526–31, 1989.

[17] M. Mihail and U. Vazirani. On the magnification of 0-1 polytopes. Technical Report TR-3-89, Aiken Computer Lab., Harvard, 1989. To appear in *JCTB*.

[18] J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph in connected. *SIAM Journal on Computing*, 12(4):777–88, 1983.

[19] J. Spencer. *Ten Lectures on the Probabilistic Method.* CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[20] S. Toda. On the computational power of PP and $\oplus P$. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 514–9, 1989.

[21] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–21, 1979.

[22] U. Vazirani. Personal communication.

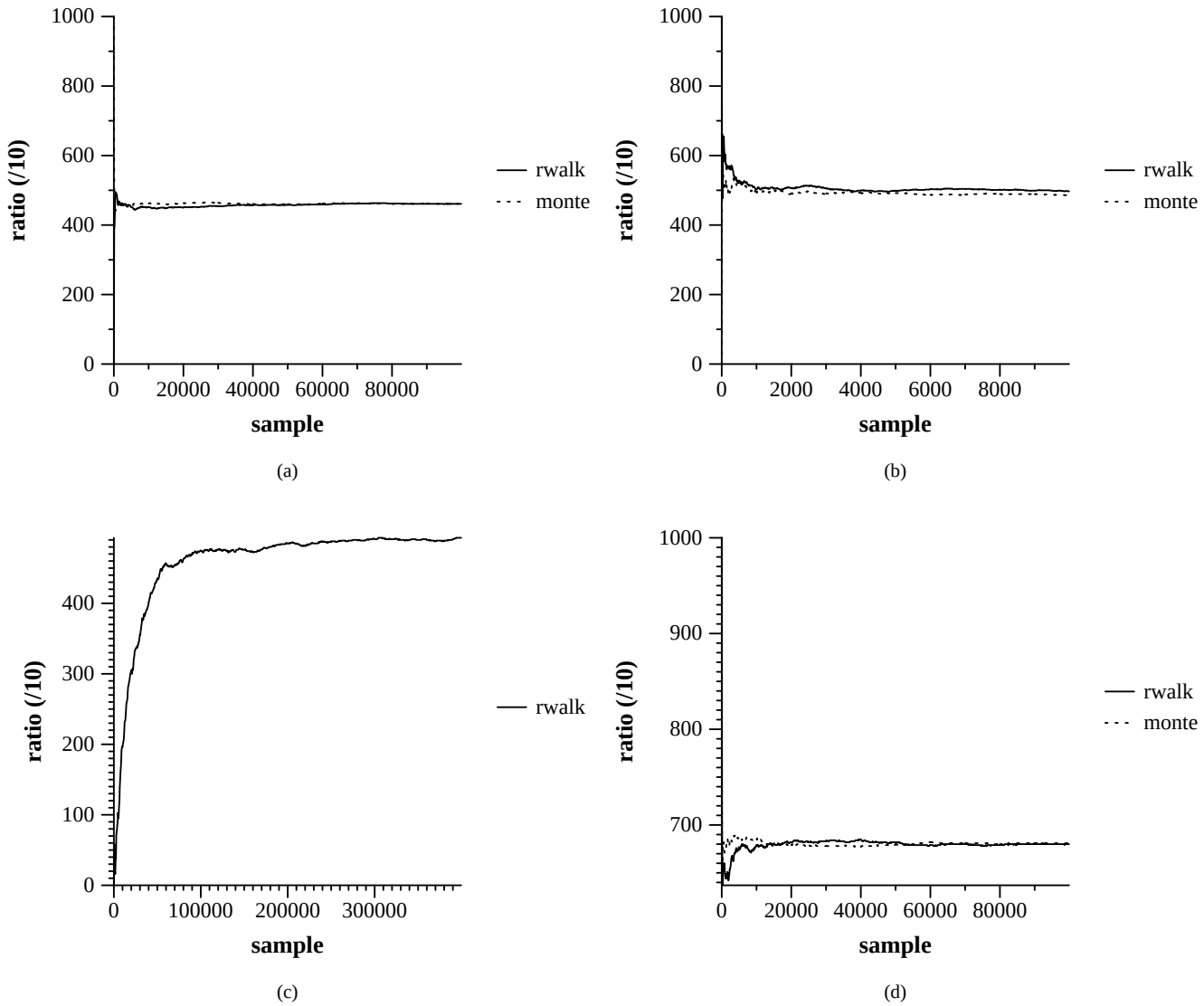


Figure 2: Convergence plots. Solid curves contain data from a random walk simulation of a Markov chain; dotted curves contain data from the naive Monte Carlo algorithm (Algorithm 1). The y-axes represent the single-edge ratios (in tenth-units) measured; the x-axes represent the number of samples. (a) Random graph $n = 50$, $p = 1/5$, size 114, rate m; (b) Random graph $n = 50$, $p = 1/5$, all sizes (connected), rate m^2; (c) Random graph $n = 50$, $p = 1/2$, all sizes (disconnected), rate n; (d) Random graph $n = 50$, $p = 3/4$, size 637, rate m.

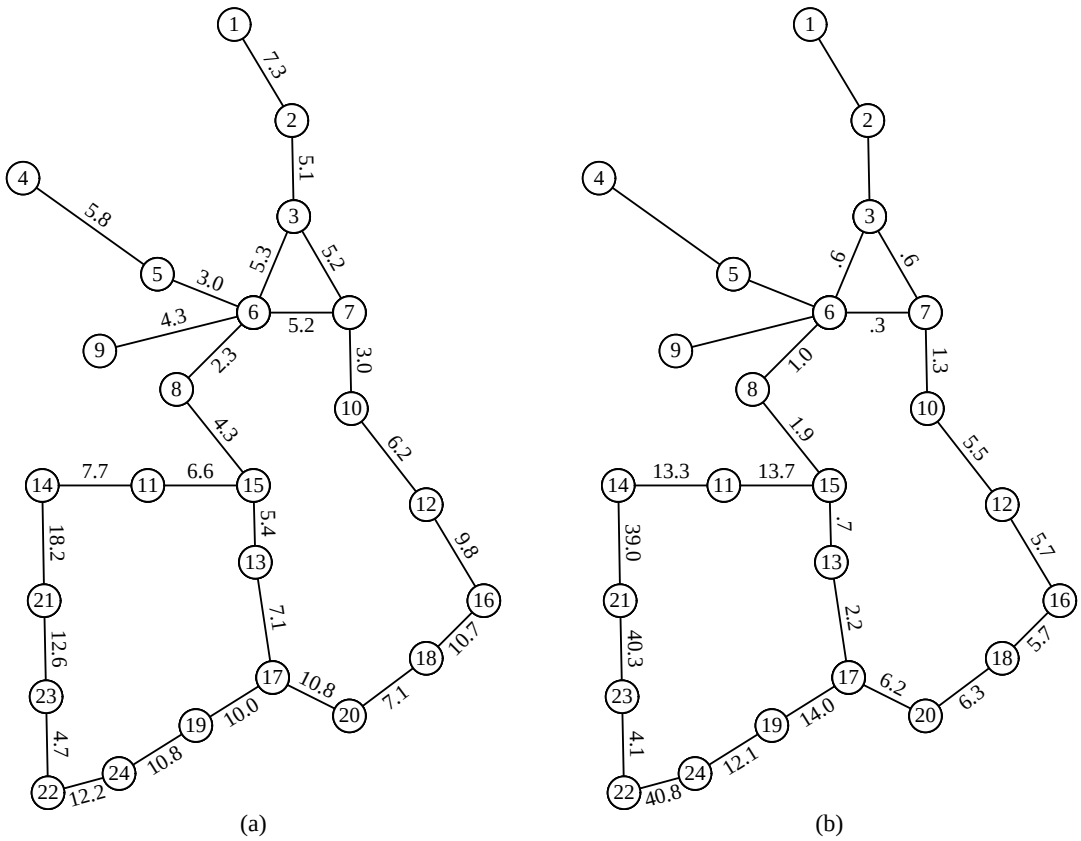


Figure 3: (a) A network; (b) Network with modified failure probabilities.