

CHENNAI MATHEMATICAL INSTITUTE

MASTERS THESIS

---

# Random Spanning Trees

---

*Author:*

Bhishmaraj S

*Supervisor:*

Dr. Samir DATTA

*A thesis submitted in fulfillment of the requirements  
for the degree of Master of Science*

*in the*

Computer Science at CMI

June 9, 2020



## Declaration of Authorship

I, Bhishmaraj S, declare that this thesis titled, “Random Spanning Trees” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself

Signed:

---

Date:

---



CHENNAI MATHEMATICAL INSTITUTE

# *Abstract*

Faculty Name  
Computer Science at CMI

Master of Science

**Random Spanning Trees**

by Bhishmaraj S

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too....  
I see



## *Acknowledgements*

The acknowledgments and the people to thank go here, don't forget to include your project advisor. . .





# Contents

<b>Declaration of Authorship</b>	<b>iii</b>
<b>Abstract</b>	<b>v</b>
<b>Acknowledgements</b>	<b>vii</b>
<b>Contents</b>	<b>ix</b>
<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Abbreviations</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.0.1 Algorithms for sampling random spanning trees . . . . .	1
Matrix Based . . . . .	1
Random Walk Based . . . . .	2
Approximation Algorithms . . . . .	2
1.0.2 Motivation . . . . .	2
1.0.3 Structure of the thesis . . . . .	2
<b>2 Background</b>	<b>3</b>
2.1 Preliminary Linear Algebra . . . . .	3
2.2 Laplacian of a Graph . . . . .	3
2.2.1 Kirchoff Matrix Tree Theorem . . . . .	4
2.3 Electric Networks . . . . .	4
Incidence Matrix . . . . .	4
Unweighted Case . . . . .	4
<b>3 Matrix Approach</b>	<b>5</b>
3.1 Harvey, Xu . . . . .	5
Naive chain rule algorithm . . . . .	5
3.1.1 Summary of the paper . . . . .	5
3.1.2 Harvey, Xu Algorithm . . . . .	6
Notation . . . . .	6
An Example . . . . .	8
3.1.3 Structure of the paper . . . . .	9
3.1.4 Facts used . . . . .	9
3.1.5 Technical details of the results . . . . .	11
Deletion . . . . .	11
Contraction . . . . .	13
3.1.6 Summary . . . . .	15

<b>4 Conclusion</b>	<b>17</b>
<b>Bibliography</b>	<b>19</b>

# List of Figures

2.1	An example of a graph and it's corresponding electric network . . . . .	4
3.1	Harvey, Xu algorithm . . . . .	7
3.2	Example . . . . .	8
3.3	Dependencies between the results . . . . .	9



# List of Tables



# List of Abbreviations

**LAH** List Abbreviations Here  
**WSF** What (it) Stands For





# 1 Introduction

Spanning trees have been a central object of study in graph theory for a long time. Kirchhoff [20] established a linear algebraic relationship between spanning trees of graphs and determinants while studying electric networks. We are interested in the following problem.

**Problem 1** (Uniform random spanning tree). *Given an undirected connected graph  $G = (V, E)$ , sample a spanning tree  $T$  with probability  $\frac{1}{|\mathcal{T}|}$  where  $\mathcal{T}$  denotes the set of all spanning trees of  $G$ .*

Sampling spanning trees happens to be a primitive used in various problems such as

- Constructing expanders ([14], [11])
- Approximation algorithms for the travelling salesman problem([13], [2])
- Graph Sparsification ([12], [9])
- Analysis of network reliability ([5],[22], [7])
- Sequence shuffling problem in Bioinformatics ([18])

Bar-Ilan and Zernik [3] proposed a distributed algorithm for this problem. Recently Russo, Teixeira, and Francisco [24] implemented some of the random walk algorithms and have compared their efficiencies.

## 1.0.1 Algorithms for sampling random spanning trees

There has been a lot of progress in this problem for the past 40 years. These algorithms can be broadly classified into 3 categories.

### Matrix Based

These algorithms are based on Kirchhoff Matrix Tree theorem and involve computing determinants of the laplacian matrix of the graph to sample spanning trees. These notions would be made clear in later part of the thesis.

- Random Spanning Tree, [15]
- Unranking and ranking spanning trees of a graph, [6]
- Generating random combinatorial objects, [21]
- Two Algorithms for Unranking Arborescences, [8]
- Generating random spanning trees via fast matrix multiplication, [17]

## Random Walk Based

These algorithms simulate variants of random walk on the input graph and constructs a spanning tree from this simulation.

- Generating random spanning trees, [4]
- The random walk construction of uniform spanning trees and uniform labelled trees, [1]
- Generating random spanning trees more quickly than the cover time, [28]
- How to Get a Perfectly Random Sample from a Generic Markov Chain and Generate a Random Spanning Tree of a Directed Graph, [23]

## Approximation Algorithms

These are more recent algorithms which sample a uniform spanning tree with a high probability. The main theme here is to employ tools from approximation algorithms.

- Faster generation of random spanning trees, [19]
- Sampling Random Spanning Trees Faster than Matrix Multiplication, [10]
- An Almost-Linear Time Algorithm for Uniform Random Spanning Tree Generation, [25]

### 1.0.2 Motivation

The main motivation with which we started exploring this problem was to extend sampling random spanning trees to a dynamic setting. We wanted to sample a spanning tree faster than computing it from scratch.

### 1.0.3 Structure of the thesis

First we define the basic ideas which would be used later. Then we give a brief summary of the algorithms of some of the papers listed above. Then we explain the Harvey and Xu [17] algorithm in detail.

## 2 Background

To make the exposition self contained I have chosen the results which are relevant to understanding the Harvey, Xu algorithm. Most of these results are taken from Vishnoi [26]

### 2.1 Preliminary Linear Algebra

**Fact 1.** *If  $A$  is a  $n \times n$  real symmetric matrix, then all it's eigenvalues are real.*

**Fact 2** (Eigenvectors of different eigenvalues are orthogonal). *Let  $\lambda_i$  and  $\lambda_j$  be two eigenvalues of a symmetric matrix  $A$  and  $\mathbf{u}_i, \mathbf{u}_j$  be it's corresponding eigenvectors. If  $\lambda_i \neq \lambda_j$  then  $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = 0$*

**Fact 3** (Min-Max Characterizations of Eigenvalues). *If  $A$  is a  $n \times n$  real symmetric matrix, then the largest eigenvalue of  $A$  is*

$$\lambda_n(A) = \max_{v \in \mathbb{R}^n \setminus \{0\}} \frac{v^T A v}{v^T v}$$

$$\lambda_1(A) = \min_{v \in \mathbb{R}^n \setminus \{0\}} \frac{v^T A v}{v^T v}$$

**Fact 4** (Courant-Weyl-Fisher min-max principle).

$$\lambda_k(A) = \min_{\substack{v \in \mathbb{R}^n \setminus \{0\} \\ v^T u_i = 0 \quad \forall i \in \{1, \dots, k-1\}}} \frac{v^T A v}{v^T v}$$

$$\lambda_k(A) = \max_{\substack{v \in \mathbb{R}^n \setminus \{0\} \\ v^T u_i = 0 \quad \forall i \in \{1, \dots, k-1\}}} \frac{v^T A v}{v^T v}$$

**Fact 5** (Positive Semi Definite). *A matrix  $A$  is said to be positive semi definite (PSD) if  $\lambda_1(A) \geq 0$ .  $A$  is said to be positive definite if  $\lambda_1(A) > 0$*

### 2.2 Laplacian of a Graph

**Definition 1** (Laplacian). *For an undirected unweighted graph  $G = (V, E)$  the Laplacian  $L_G$  is a  $|V| \times |V|$  matrix defined as*

$$L_G(i, j) = \begin{cases} -1 & \text{if } (i, j) \in E \\ \deg(i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

*It can also be seen that*

$$L_G = D - A$$

where  $D$  is a diagonal matrix with diagonal entries as degree of the corresponding vertex.

**Definition 2** (Weighted Laplacian). For an undirected weighted graph  $G = (V, E)$  and a weight function  $w : E \rightarrow \mathbb{R}_{\geq 0}$  the Laplacian  $L_G$  is a  $|V| \times |V|$  matrix defined as

$$L_G(i, j) = \begin{cases} -w(i, j) & \text{if } (i, j) \in E \\ \sum_{(i, v) \in E} w(i, v) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

**Fact 6.** The Laplacian of a graph is PSD

**Fact 7.** Let  $L$  be the Laplacian of a graph  $G = (V, E)$ , then

$$\lambda_2(L) > 0 \iff G \text{ is connected}$$

### 2.2.1 Kirchhoff Matrix Tree Theorem

## 2.3 Electric Networks

### Incidence Matrix

<sup>1</sup>

### Unweighted Case

Given an unweighted graph  $G$  we associate a electrical network by replacing each edge with a resistor with resistance  $1 \Omega$ .

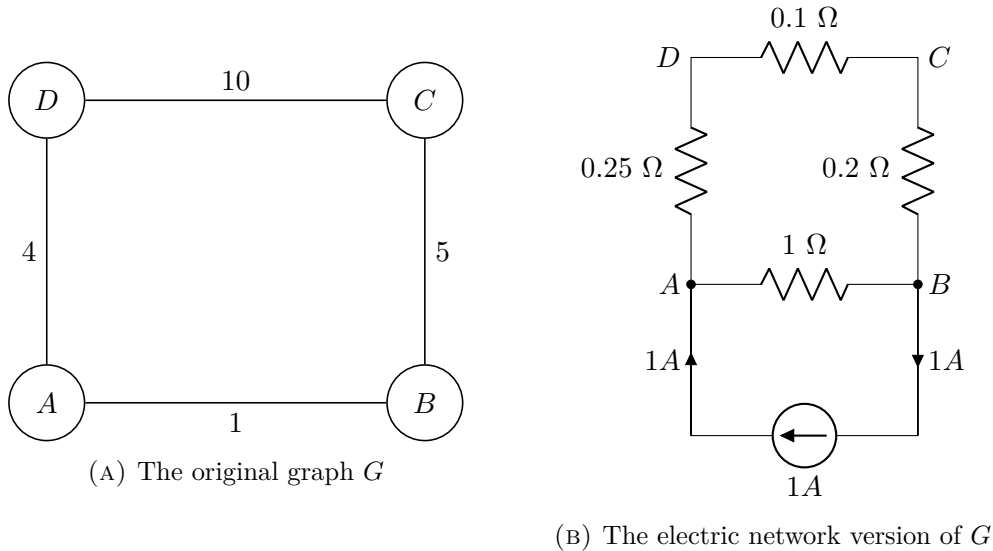


FIGURE 2.1: An example of a graph and it's corresponding electric network

<sup>1</sup>I have used a transposed version compared to Vishnoi [26] so that it's consistent with the notation used later

## 3 Matrix Approach

### 3.1 Harvey, Xu

Harvey and Xu [17] proposed a  $\mathcal{O}(N^\omega)$  (where  $\omega$  is the exponent of fast matrix multiplication) algorithm for sampling a uniform spanning tree which is much simpler compared to the one proposed earlier by CMN which also has the same running time. The initial starting point for the algorithm is using the relationship between effective resistance and the probability that an edge belongs to a spanning tree. And also the fact that sampling an edge corresponds to contracting it and discarding the edge corresponds to deleting it. The following naive chain rule algorithm works on the same principle.

#### Naive chain rule algorithm

**Input:**  $G = (V, E)$  and  $L_G^+$   
**Output:** Set of edges corresponding to a random spanning tree

```

1 for  $e = (u, v) \in E$  do
2    $R_e^{\text{eff}} = (\chi_u - \chi_v)^T L_G^+ (\chi_u - \chi_v)$ ;
3   if  $(X \sim \text{Bernoulli}(R_e^{\text{eff}})) = 1$  then
4     Add edge  $e$  to the spanning tree;
5      $G = G/e$ ;
6   else
7      $G = G \setminus e$ ;
8   end
9   Update  $L_G^+$ ;
10 end
```

#### Algorithm 1: Sampling uniform spanning tree using chain rule

Computing  $L_G^+$  takes  $\mathcal{O}(N^3)$  hence the overall running time is  $\mathcal{O}(MN^3)$ . The main bottleneck of this algorithm is to modify the graph each time after making a decision to sample an edge.

#### 3.1.1 Summary of the paper

On a high level the main ideas used in the paper are

1. Use a divide and conquer algorithm to break the graph into smaller parts and sample on each parts separately and update the pseudoinverse of the laplacian lazily only on the subgraph when needed
2. The important insight here is that the sampling probability of an edge depends only on 4 entries of the pseudoinverse of the laplacian. Hence we don't need to update all the entries of the matrix when the graph is modified

3. A well known method to compute inverse of a matrix with updates is to use the Sherman-Morrison-Woodbury formula. But in this case the formula has to be modified to work for the case where only a submatrix is modified.
4. Since while contracting an edge the number of vertices decreases it would get cumbersome to modify the dimension of the matrix everytime. So they overcome this issue by considering the formula on the limit case. When the graph is considered as a electric network then increasing the weight of an edge corresponds to shorting that link, hence in the limit case we get the same result as contracting the edge
5. One of the main improvements over the previous algorithms of the same complexity (Colbourn, Myrvold, and Neufeld [8]) is that the intricacies of LU decomposition is avoided since the current algorithm uses only matrix inversion.
6. All the formulas are derived for the submatrix case hence the complexity of a sub problem depends only on the size of the subproblem.

### 3.1.2 Harvey, Xu Algorithm

Given a graph  $G = (V_G, E_G)$ . The algorithm proceeds by splitting the vertex set into 2 equal halves  $V_G = S \uplus R$ . And now they define subsets of edges  $E[S] = (S \times S) \cap E_G$  and  $E[R, S] = (S \times R) \cap E_G$ .

Suppose  $S = S_1 \uplus S_2$  and  $R = R_1 \uplus R_2$ . Then we can see that

$$E[S] = E[S_1] \cup E[S_2] \cup E[S_1, S_2]$$

$$E[R, S] = \bigcup_{i,j \in \{1,2\}} E[R_i, S_j]$$

This formula is used to recurse on the subproblems for each subset. In this manner each edge will be visited exactly once by the algorithm

#### Notation

- $N$  - This is an auxillary matrix which starts with  $L_G^+$  and gets updated lazily as the algorithm progresses.
- **function**  $SampleSpanningTree(G = (V, E))$  - This is the main function from which the execution of the algorithm starts. It takes input a graph and outputs a set of edges corresponding to a random spanning tree.
- **function**  $SampleEdgesWithin(S)$  - This function takes input a set of vertices  $S$  and returns sets  $F$  and  $D$  which are set of edges contracted and deleted respectively in the subgraph induced by  $S$  on  $G$ .
- **function**  $SampleEdgesCrossing(R, S)$  - This function works similar to the above one but samples edges crossing the sets  $R$  and  $S$ . Also the base case of the entire algorithm (case where  $|R| = |S| = 1$ ) of making the decision to sample an edge is handled here.
- **procedure**  $Update(S, F, D)$  - This function updates the sub matrix  $N_{S,S}$  based on the formulas derived in **Corollary 1** and **Theorem 3**

**Algorithm 2.** A Recursive Algorithm

---

```

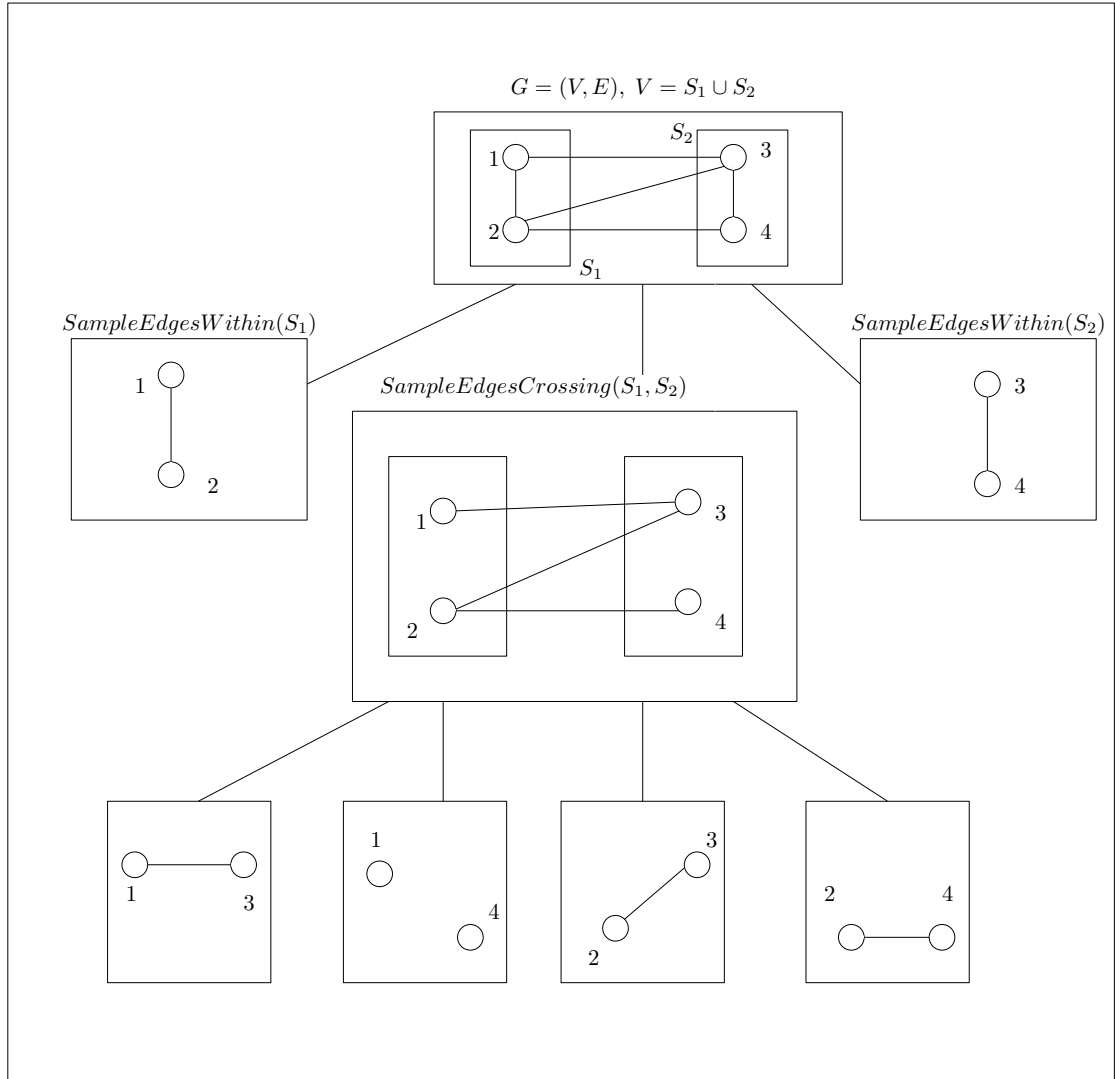
1: function SAMPLESPANNINGTREE( $G = (V_G, E_G)$ )
2:    $N \leftarrow L_G^+$ 
3:   SAMPLEEDGESWITHIN( $V_G$ )
4:   return the uniform spanning tree  $T$ 
5: function SAMPLEEDGESWITHIN( $S$ )
6:   if  $|S| = 1$  then return
7:   Divide  $S$  in half:  $S = S_1 \cup S_2$ 
8:   for  $i \in \{1, 2\}$  do
9:     SAMPLEEDGESWITHIN( $S_i$ )
10:    Restore  $N_{S_i, S_i}$  to its value before entering the recursion
11:     $F \leftarrow$  the set of edges contracted in SAMPLEEDGESWITHIN( $S_i$ )
12:     $D \leftarrow$  the set of edges deleted in SAMPLEEDGESWITHIN( $S_i$ )
13:    UPDATE( $S, F, D$ )
14:   SAMPLEEDGESCROSSING( $S_1, S_2$ )
15: function SAMPLEEDGESCROSSING( $R, S$ )
16:   if  $|R| = 1$  then
17:     Let  $r \in R$  and  $s \in S$ ,  $R^{\text{eff}} \leftarrow (\mathcal{X}_r - \mathcal{X}_s)^T N (\mathcal{X}_r - \mathcal{X}_s)$ 
18:     Flip a biased coin that turns head with probability  $R^{\text{eff}}$ 
19:     if head then
20:       Add  $e_{r,s}$  to the uniform spanning tree  $T$  and the set of contracted edges
21:     else
22:       Add  $e_{r,s}$  to the set of deleted edges
23:   else
24:     Divide  $R$  and  $S$  each in half:  $R = R_1 \cup R_2$  and  $S = S_1 \cup S_2$ 
25:     for  $i \in \{1, 2\}$  and  $j \in \{1, 2\}$  do
26:       SAMPLEEDGESCROSSING( $R_i, S_j$ )
27:       Restore  $N_{R_i \cup S_j, R_i \cup S_j}$  to its value before entering the recursion
28:        $F \leftarrow$  the set of edges contracted in SAMPLEEDGESCROSSING( $R_i, S_j$ )
29:        $D \leftarrow$  the set of edges deleted in SAMPLEEDGESCROSSING( $R_i, S_j$ )
30:       UPDATE( $R \cup S, F, D$ )
31: procedure UPDATE( $S, F, D$ )
32:   Let  $V$  be the incidence matrix for  $F$ 
33:   Let  $L_D$  be the Laplacian matrix for  $D$ 
34:    $N_{S,S} \leftarrow N_{S,S} - N_{S,S} V_{S,*} (V_{S,*}^T N_{S,S} V_{S,*})^{-1} V_{S,*}^T N_{S,S}$ 
35:    $N_{S,S} \leftarrow N_{S,S} - N_{S,S} (L_{D_{S,S}} N_{S,S} - I)^{-1} L_{D_{S,S}} N_{S,S}$ 

```

---

FIGURE 3.1: Harvey, Xu algorithm <sup>a</sup><sup>a</sup>This image is exactly reproduced from the paper.

## An Example

FIGURE 3.2: An example<sup>a</sup>

<sup>a</sup>The function calls to *SampleEdgesWithin*( $S_1$ ) and *SampleEdgesWithin*( $S_2$ ) in turn would call their respective *SampleEdgesCrossing* functions. But it would be the base case and would look similar to the one portrayed here and hence omitted for the sake of brevity.



### 3.1.3 Structure of the paper

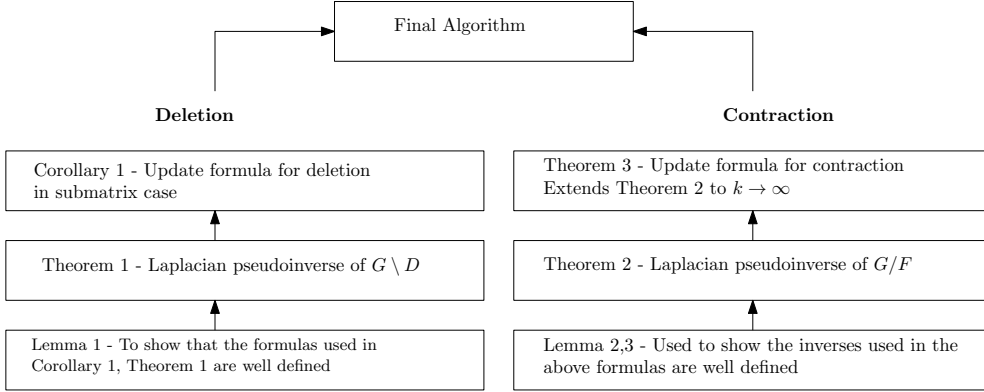


FIGURE 3.3: Dependencies between the results

### 3.1.4 Facts used

**Fact 1** (Woodbury matrix identity). *Let  $M \in \mathbb{M}_{n \times n}$ ,  $U \in \mathbb{M}_{n \times k}$ ,  $V \in \mathbb{M}_{n \times k}$ . Suppose  $M$  is non-singular then  $M + UV^T$  is non-singular  $\iff I + V^T M^{-1} U$  is non-singular. If  $M + UV^T$  is non-singular, then*

$$(M + UV^T)^{-1} = M^{-1} - \left( M^{-1} \cdot U \cdot (I + V^T M^{-1} U)^{-1} \cdot V^T \cdot M^{-1} \right)$$

**Fact 2.** *For any  $L \in \mathbb{M}_{n \times n}$  with  $\ker(L) = \text{span}(\mathbf{1})$ , we have  $LL^+ = I - \frac{\mathbf{1}\mathbf{1}^T}{n}$  and  $P := I - \frac{\mathbf{1}\mathbf{1}^T}{n}$  is called the **projection matrix**.*

The following set of facts are about the properties of matrix operations (addition, multiplication, etc.) on sub-matrices. The first 4 are easy to see, so I haven't derived them. For the last one I have written a derivation using Shur's complement from Wikipedia

**Fact 3** (Sub-matrices). *For all the results below,  $S$  denotes a index set and  $S^c$  denotes it's complement.*

1. For any  $A, B \in \mathbb{M}_{n \times n}$ ,  $(A + B)_{S,S} = A_{S,S} + B_{S,S}$
2. If  $C = D \cdot E \cdot F$  then  $C_{S,S} = D_{S,*} \cdot E \cdot F_{*,S}$
3. For  $A \in \mathbb{M}_{m \times n}$ ,  $B \in \mathbb{M}_{n \times l}$ , If  $A_{S^c, S^c} = 0$  or  $B_{S^c, S^c} = 0$  then  $(AB)_{S,S} = A_{S,S} \cdot B_{S,S}$
4. For any matrix  $C$  where  $C = D \cdot E \cdot F$ . If  $D_{*, S^c} = 0$  and  $F_{S^c, *} = 0$ , then  $C = D_{*, S} \cdot E_{S,S} \cdot F_{S,*}$
5.  $D = \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}$ , and  $E = \begin{bmatrix} A & B \\ X & Y \end{bmatrix}$  where  $M, A \in \mathbb{M}_{n \times n}$  and If  $(MA - I)$  is invertible Then,

$$(DE - I)^{-1} = \begin{bmatrix} (MA - I)^{-1} & (MA - I)^{-1} \cdot M \cdot B \\ 0 & -I \end{bmatrix}$$

*Proof.*  $(DE - I)^{-1}$  can be computed using Shur's Complement([27]) .

Suppose  $N = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}$  and Shur's complement of block  $S$  and  $P$  is

$$N/S := P - QS^{-1}R \quad N/P := S - RP^{-1}Q$$

Then

$$N^{-1} = \begin{bmatrix} P^{-1} + (P^{-1}Q(N/P)^{-1}RP^{-1}) & -(P^{-1}Q(N/P)^{-1}) \\ -((N/P)^{-1}RP^{-1}) & (N/P)^{-1} \end{bmatrix}$$

In our case  $N = \begin{bmatrix} MA - I & MB \\ 0 & -I \end{bmatrix}$  and  $(N/P) = -I$ . From this it follows that

$$N^{-1} = (DE - I)^{-1} = \begin{bmatrix} (MA - I)^{-1} & (MA - I)^{-1} \cdot M \cdot B \\ 0 & -I \end{bmatrix}$$

■

**Fact 4.** Let  $A, B \in \mathbb{M}_{n \times n}$  with  $B$  being symmetric PSD. Suppose  $x$  is an eigenvector of  $AB$  corresponding to eigenvalue  $\lambda$ . Then  $\sqrt{B}x$  is an eigenvector of  $\sqrt{B}A\sqrt{B}$  corresponding to eigenvalue  $\lambda$

**Fact 5** (Laplacian and graph connectivity (Fiedler value)). Let  $G$  be a graph with  $n$  vertices. Suppose  $(\lambda_1, \lambda_2 \dots \lambda_n)$  be the eigenvalues corresponding to the eigenvectors  $(v_1, v_2 \dots v_n)$  of the Laplacian of  $G$  denoted as  $L_G$ .  $L_G$  is symmetric PSD with  $\lambda_1 = 0$  and  $v_1 = \mathbf{1}$ . The following properties relate the eigenvalues of  $L_G$  with the connectivity of  $G$  :

1.  $\lambda_2 > 0 \iff G$  is connected
2.  $G$  is disconnected  $\iff \exists z$  with  $z^T \mathbf{1} = 0$  and  $z^T L_G z = 0$

The above is true for  $L_G^+$  also

**Definition 1** ( $\chi_u$ ).  $\chi_u$  is a vector of size  $|V|$

$$\chi_u(i) = \begin{cases} 1, & \text{if } i = u \\ 0, & \text{otherwise} \end{cases}$$

**Definition 2** (Uniform random spanning tree). Let  $\hat{T}$  be the random variable denoting a uniformly random spanning tree, then  $\mathbb{P}(\hat{T} = T) = \frac{1}{|\mathcal{T}|}$ , where  $\mathcal{T}$  is the set of all spanning trees of  $G$ .

**Fact 6.** Given a graph  $G = (V, E)$  with laplacian  $L_G$ , the effective resistance of an edge  $e = \{u, v\} \in E$  is

$$R_e^{\text{eff}} = (\chi_u - \chi_v)^T L_G^+ (\chi_u - \chi_v)$$

Then for any  $e \in E$  we have

$$\mathbb{P}(e \in \hat{T}) = R_e^{\text{eff}}$$

### 3.1.5 Technical details of the results

#### Deletion

The first step in obtaining a updation formula for deletion is to make sure  $(I - L_D L_G^+)$  is invertible. As the inverse of this term would be used in the expansion of  $(L_G - L_D)^+$

For the first direction of Lemma 1, the main result used is **Fact 5** ( $G$  is disconnected  $\iff \exists z$  with  $z^T \mathbf{1} = 0$  and  $z^T L_G^+ z = 0$ ). So if we can show this for a suitable  $z$  we are done. Now using the hypothesis that  $(I - L_D L_G^+)$  is singular and **Fact 4** they derive the following  $y^T \cdot \sqrt{L_G^+} \cdot L_{G \setminus D} \cdot \sqrt{L_G^+} \cdot y = 0$ . As we can see the remaining part is to show  $(z = \sqrt{L_G^+} y) \perp \mathbf{1}$

**Lemma 1** (Formulas in **Theorem 1** are well defined). *Let  $G = (V, E)$  be a connected graph and  $D \subseteq E$  then*

$$(I - L_D L_G^+) \text{ is invertible} \iff G \setminus D \text{ is connected}$$

*Proof.* First let's show that If  $(I - L_D L_G^+)$  is singular then  $G \setminus D$  is disconnected

- Since  $(I - L_D L_G^+)$  is singular  $\exists x \neq 0$  s.t.  $(I - L_D L_G^+)x = 0$

$$\implies L_D L_G^+ x = x \quad (3.1)$$

$$\implies 1 \in \text{eigenvalues}(L_D L_G^+) \quad (3.2)$$

$$\implies 1 \in \text{eigenvalues}((L_G - L_{G \setminus D}) L_G^+) \quad (3.3)$$

- Let  $x \perp \mathbf{1}$  be an eigenvector of  $(L_G - L_{G \setminus D}) L_G^+$  with eigenvalue 1.

- By **Fact 4**,  $y = \frac{\sqrt{L_G^+} x}{\|\sqrt{L_G^+} x\|}$  is an eigenvector of  $\sqrt{L_G^+} (L_G - L_{G \setminus D}) \sqrt{L_G^+}$

$$= y^T \cdot \sqrt{L_G^+} (L_G - L_{G \setminus D}) \sqrt{L_G^+} \cdot y = 1 \quad (3.4)$$

$$= y^T \sqrt{L_G^+} L_G \sqrt{L_G^+} y = y^T L_G^+ L_G y = y^T P y \quad (3.5)$$

$$= y^T \left( I - \frac{\mathbf{1}^T \mathbf{1}}{n} \right) y = y^T y - \left( \frac{y^T \mathbf{1}^T \mathbf{1} y}{n} \right) = y^T y = 1 \quad (3.6)$$

- $\therefore y^T \sqrt{L_G^+} L_{G \setminus D} \sqrt{L_G^+} y = 0$  now if we consider  $z = \sqrt{L_G^+} y$  and show that  $z^T \mathbf{1} = 0$  then we can use **Fact 5** to complete the proof

$$y^T \sqrt{L_G^+} \mathbf{1} = x^T \sqrt{L_G^+} \sqrt{L_G^+} \mathbf{1} = 0 L_G^+ \quad (3.7)$$

$$G \setminus D \text{ is disconnected} \quad (3.8)$$

Now to prove the converse, If  $G \setminus D$  is disconnected then  $I - L_D L_G^+$  is singular

- If  $G \setminus D$  is disconnected then  $\exists y \perp \mathbf{1}, \|y\| = 1$  we have

$$1. y^T \cdot \sqrt{L_G^+} \cdot L_{G \setminus D} \cdot \sqrt{L_G^+} \cdot y = 0$$

$$2. y^T \cdot \sqrt{L_G^+} \cdot L_G \cdot \sqrt{L_G^+} \cdot y = y^T y = 1$$

- From (1) and (2) we get  $y^T \sqrt{L_G^+} (L_G - L_{G \setminus D}) \sqrt{L_G^+} y = 1$

$$\implies y^T \cdot \sqrt{L_G^+} \cdot L_D \cdot \sqrt{L_G^+} \cdot y = 1 \quad (3.9)$$

$$\implies 1 \in \text{eigenvalues}(L_D L_G^+) \quad (3.10)$$

$$\implies (I - L_D L_G^+) \text{ is singular} \quad (3.11)$$

■

In **Theorem 1** they just show that the formula for the updated pseudoinverse is indeed true. This is shown using the following identity  $LL^+ = P$

**Theorem 1** (Update identity for Deletion). *Let  $G = (V, E)$  be a connected graph and  $D \subseteq E$ . If  $G \setminus D$  is connected then*

$$(L_G - L_D)^+ = L_G^+ - \left( L_G^+ \cdot (L_D L_G^+ - I)^{-1} \cdot L_D \cdot L_G^+ \right)$$

*Proof.* If R.H.S is indeed true then it should satisfy the property of  $LL^+ = P$

$$\begin{aligned} & (L_G - L_D) \cdot (L_G - L_D)^+ \\ & (L_G - L_D) \cdot \left( L_G^+ - \left( L_G^+ \cdot (L_D L_G^+ - I)^{-1} \cdot L_D \cdot L_G^+ \right) \right) \\ & [P - L_D L_G^+] - \left[ (L_G L_G^+ - L_D L_G^+) \cdot (L_D L_G^+ - I)^{-1} \cdot L_D \cdot L_G^+ \right] \\ & [P - L_D L_G^+] + \left[ \left( (L_D L_G^+ - I) + \frac{\mathbf{1}\mathbf{1}^T}{n} \right) \cdot (L_D L_G^+ - I)^{-1} \cdot L_D \cdot L_G^+ \right] \\ & [P - L_D L_G^+] + \left[ (L_D L_G^+) + \left( \frac{\mathbf{1}\mathbf{1}^T}{n} \cdot (L_D L_G^+ - I)^{-1} \cdot L_D \cdot L_G^+ \right) \right] \end{aligned}$$

We can see that  $-\mathbf{1}^T (L_D L_G^+ - I) = -\mathbf{1}^T L_D L_G^+ + (I\mathbf{1})^T = 0 + \mathbf{1}^T = \mathbf{1}^T$ . Hence  $\mathbf{1}^T (L_D L_G^+ - I)^{-1} = -\mathbf{1}^T$ . And also  $\mathbf{1}^T L_D = 0$ . Hence,

$$P - L_D L_G^+ + L_D L_G^+ + \mathbf{1}^T L_D L_G^+ = P$$

■

**Definition 3** (Submatrix). *A submatrix of a matrix  $A$  containing rows  $S$  and columns  $T$  is denoted as  $A_{S,T}$*

**Corollary 1** modifies the update formula in **Theorem 1** to work for submatrices and hence reduce the complexity to  $\mathcal{O}(|S|^\omega)$ . They do this by first applying the facts related to submatrices **Fact 3.3, 3.5**.

**Corollary 1** (Improved **Theorem 1** for submatrix). *Let  $G = (V, E)$  be a connected graph and  $D \subseteq G$ . For  $S \subseteq V$  define  $E[S]$  as  $(S \times S) \cap E$ . Suppose  $E_D \subseteq E[S]$  and  $G \setminus D$  is connected then*

$$(L_G - L_D)_{S,S}^+ = (L_G^+)_{S,S} - \left( (L_G^+)_{S,S} \cdot ((L_D)_{S,S} (L_G^+)_{S,S} - I)^{-1} \cdot (L_D)_{S,S} \cdot (L_G^+)_{S,S} \right)$$

*Proof.* From **Theorem 1** we know that  $(L_G - L_D)^+ = L_G^+ - \left( L_G^+ \cdot (L_D L_G^+ - I)^{-1} \cdot L_D \cdot L_G^+ \right)$ . If we apply **Fact 3.1** to  $(L_G - L_D)^+$  we get

$$(L_G^+)_{S,S} - \left( L_G^+ \cdot (L_D L_G^+ - I)^{-1} \cdot L_D \cdot L_G^+ \right)_{S,S}$$

Applying **Fact 3.3** we get

$$(L_G^+)_{S,S} - \left( (L_G^+)_{S,S} \cdot (L_D L_G^+ - I)_{S,S}^{-1} \cdot (L_D)_{S,S} \cdot (L_G^+)_{S,S} \right)$$

Now applying **Fact 3.5** to  $(L_D L_G^+ - I)_{S,S}^{-1}$

**Fact 3.5** states that If  $D = \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}$ , and  $E = \begin{bmatrix} A & B \\ X & Y \end{bmatrix}$  where  $M, A \in \mathbb{M}_{n \times n}$  and If  $(MA - I)$  is invertible Then,

$$(DE - I)^{-1} = \begin{bmatrix} (MA - I)^{-1} & (MA - I)^{-1} \cdot M \cdot B \\ 0 & -I \end{bmatrix}$$

Here we have  $L_D = \begin{bmatrix} (L_D)_{S,S} & 0 \\ 0 & 0 \end{bmatrix}$ , and  $L_G = \begin{bmatrix} (L_G)_{S,S} & (L_G)_{S,S^c} \\ (L_G)_{S^c,S} & (L_G)_{S^c,S^c} \end{bmatrix}$

$$\therefore (L_D L_G - I)^{-1} = \begin{bmatrix} ((L_D)_{S,S} (L_G)_{S,S} - I)^{-1} & (L_D)_{S,S} (L_G)_{S,S^c} \\ 0 & -I \end{bmatrix}$$

Hence we get the required result

$$(L_G - L_D)_{S,S}^+ = (L_G^+)_{S,S} - \left( (L_G^+)_{S,S} \cdot ((L_D)_{S,S} (L_G^+)_{S,S} - I)^{-1} \cdot (L_D)_{S,S} \cdot (L_G^+)_{S,S} \right)$$

■

### Contraction

The main approach proposed to tackle contraction updates is to increase the weight of the edges that are to be contracted to a large value  $k$ .

**Definition 4** (Incidence Matrix). Let  $G = (V, E)$ , given an edge  $e = u, v \in E$  the incidence vector of  $e$  is defined as  $v_e = (\chi_u - \chi_v)$ . Given a set of edges  $D = \{e_1, e_2 \dots e_m\} \subseteq E$ , the incidence matrix of  $D$  is defined as  $B_D = [v_{e_1} | v_{e_2} \dots | v_{e_m}]$

**Note** - I have used a different notation for the incidence matrix compared to the original paper I found it to be a bit confusing. And  $B$  is the common notation for incidence matrices in other resources.

**Definition 5** ( $G + ke$ ).  $G + ke$  is the weighted graph obtained by increasing  $e$ 's weight by  $k$

**Lemma 2** (Formulas in **Theorem 2** are well defined). Let  $G = (V, E)$  be a connected graph. Given  $F \subseteq E$  with  $|F| = r$  and let  $B_F$  be the incidence matrix of  $F$ .

$$B_F^T L_G^+ B_F \text{ is invertible} \iff F \text{ is a forest}$$

*Proof.* First they show

$$F \text{ is a forest} \implies B_F^T L_G^+ B_F \text{ is invertible}$$

So the main idea of this proof is to show that  $B_F^T L_G^+ B_F$  is positive definite. This is enough because positive definite matrices are non singular (if not then then they will have 0 as an eigenvalue). Now using the following claim

**Claim 1.** The incidence matrix of a acyclic graph has full column rank

Hence for any  $x \in \mathbb{R}^r, x \neq 0$ , let  $y = B_F x$  and  $y \neq 0$ . Also  $y^T \mathbf{1} = x^T B_F^T \mathbf{1} = 0$ . Hence  $y \perp \ker(L_G^+)$ . Now since  $G$  is connected we have  $\lambda_2(L_G) > 0$ . Now since  $y$  corresponds to all vectors perpendicular to  $\ker(L_G^+)$  we can say that  $y^T L_G^+ y > 0$  now expanding  $y = B_F x$  we get  $x^T (B_F^T L_G^+ B_F) x > 0$ . Hence  $B_F^T L_G^+ B_F$  is positive definite and hence invertible.

Now for the converse (TODO) ■

**Lemma 3** (Formulas in **Theorem 2** are well defined). *Let  $G = (V, E)$  be a connected graph. Given  $F \subseteq E$  and let  $B_F$  be the incidence matrix of  $F$ . For any  $k > 0$ ,*

*If  $F$  is a forest then  $\left(\frac{I}{k} + B_F^T L_G^+ B_F\right)$  is invertible for any  $k > 0$*

*Proof.* Suppose  $A, B$  are positive definite matrices then  $A + B$  is also positive definite. Since  $A, B$  are positive definite we have  $x^T A x > 0, x^T B x > 0$  for any  $x$ . Combining these two identity we get  $x^T (A + B) x > 0$ . Hence  $A + B$  is also positive definite.

By **Lemma 2**  $B_F^T L_G^+ B_F$  is positive definite. And  $I/k$  is also positive definite because all the eigenvalues are  $1/k$  and we have  $k > 0$ . Since positive definite matrices are non-singular,  $\left(\frac{I}{k} + B_F^T L_G^+ B_F\right)$  is invertible ■

**Theorem 2** uses **Lemma 2** to show that the contraction update formula for finite  $k$  is well defined.

**Theorem 2** (Contraction update formula for finite  $k$ ). *Let  $G = (V, E)$  be a connected graph. Given  $F \subseteq E$  and let  $B_F$  be the incidence matrix of  $F$ . For any  $k > 0$ ,*

$$(L_G + k L_F)^+ = L_G^+ - \left( L_G^+ \cdot B_F \cdot \left( \frac{I}{k} + B_F^T L_G^+ B_F \right)^{-1} \cdot B_F^T \cdot L_G^+ \right)$$

*Proof.* They use the same strategy used in **Theorem 1**. Also note that  $B_F B_F^T = L_F$

$$\begin{aligned} & \left[ L_G + k B_F B_F^T \right] \cdot \left[ L_G^+ - \left( L_G^+ B_F \left( \frac{I}{k} + B_F^T L_G^+ B_F \right)^{-1} B_F^T L_G^+ \right) \right] \\ &= P + k B_F B_F^T L_G^+ - \left( (L_G L_G^+ B_F + k B_F B_F^T L_G^+ B_F) \left( \frac{I}{k} + B_F^T L_G^+ B_F \right)^{-1} B_F^T L_G^+ \right) \end{aligned}$$

Here  $L_G L_G^+ B_F = (I - \frac{\mathbf{1}\mathbf{1}^T}{n}) B_F = B_F$ . Because each column sum of  $B_F$  is 0.

$$\begin{aligned} &= P + k B_F B_F^T L_G^+ - \left( k B_F \left( \frac{I}{k} + B_F^T L_G^+ B_F \right) \left( \frac{I}{k} + B_F^T L_G^+ B_F \right)^{-1} B_F^T L_G^+ \right) \\ &= P + k B_F B_F^T L_G^+ - k B_F B_F^T L_G^+ = P \end{aligned}$$
■

This corollary is a direct consequence of the facts pertaining to submatrices (**Fact 3**)

**Corollary 2** (Improves **Theorem 2** for sub-matrices). *Let  $G = (V, E)$  be a connected graph. Given  $F \subseteq E$  and let  $B_F$  be the incidence matrix of  $F$ . Suppose  $F \subseteq E[S]$ , where  $S \subseteq V$ . For any  $k > 0$ ,*

$$(L_G + k L_F)_{S,S}^+ =$$

$$(L_G^+)_{S,S} - \left( (L_G^+)_{S,S} (B_F)_{S,*} \left( \frac{I}{k} + (B_F^T)_{S,*} (L_G^+)_{S,S} (B_F)_{S,*} \right)^{-1} (B_F^T)_{S,*} (L_G^+)_{S,S} \right)$$

In **Theorem 3** they extend the contraction formula from **Theorem 2** to the case  $k \rightarrow \infty$ . The main idea of the proof is to define  $M_k = L_{G+kF}^+$ <sup>1</sup> and to show that  $\lim_{k \rightarrow \infty} \|M_k - M\| = 0$

**Theorem 3** (Extends **Theorem 2** to  $k \rightarrow \infty$  case). *For a forest  $F_1 \subseteq E$ , let  $G(k) = G + k F_1$  as defined in **Definition 3**. Let  $F_2 \subseteq E$  be disjoint from  $F_1$  such that  $F_1 \cup F_2$  is a forest. Let  $B_{F_2}$  be the incidence matrix of  $F_2$ . For  $k > 0$  define  $N = \lim_{k \rightarrow \infty} L_{G(k)}^+$*

$$\lim_{k \rightarrow \infty} L_{G(k)+kF_2}^+ = N - \left( N \cdot B_{F_2} \cdot (B_{F_2}^T N B_{F_2}) \cdot B_{F_2}^T \cdot N \right)$$

$$\text{Also } \ker \left( \lim_{k \rightarrow \infty} L_{G(k)+kF_2}^+ \right) = \text{span} (B_{F_1 \cup F_2} \cup \mathbf{1})$$

*Proof.* From **Theorem 2** we have  $M_k = L_G^+ - \left( L_G^+ \cdot B_F \cdot \left( \frac{I}{k} + B_F^T L_G^+ B_F \right)^{-1} \cdot B_F^T \cdot L_G^+ \right)$  and they define  $A = B_F^T L_G^+ B_F$

$$\|M_k - M\| = \left\| L_G^+ B_F \left( \left( \frac{I}{k} + A \right)^{-1} - A^{-1} \right) B_F^T L_G^+ \right\|$$

Using the property of matrix norm  $\|X Y\| \leq \|X\| \|Y\|$ , they get

$$\|M_k - M\| \leq \|L_G^+\|^2 \|B_F\| \|B_F^T\| \left\| \left( \frac{I}{k} + A \right)^{-1} - A^{-1} \right\| \quad (1)$$

Now they expand the last term in the above inequality using the Sherman-Morrison-Woodbury identity and apply the matrix norm inequality and with some straightforward manipulations they show that

$$\lim_{k \rightarrow \infty} \left\| \left( \frac{I}{k} + A \right)^{-1} - A^{-1} \right\| = 0$$

Now applying this to (1) they prove that  $\lim_{k \rightarrow \infty} \|M_k - M\| = 0$  ■

### 3.1.6 Summary

The ideas discussed here such as lazy updates were used in Harvey [16] to get a randomized algorithm for non-bipartite matching in  $\mathcal{O}(N^\omega)$ .

---

<sup>1</sup>I have changed the notation to  $M$  as the proof used in the paper used  $N$  for defining 2 different formulas





## 4 Conclusion



# Bibliography

- [1] David J Aldous. 1990. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM Journal on Discrete Mathematics*, 3, 4, 450–465.
- [2] Arash Asadpour, Michel X. Goemans, Aleksander Mądry, Shayan Oveis Gharan, and Amin Saberi. 2017. An  $o(\log n / \log \log n)$ -approximation algorithm for the asymmetric traveling salesman problem. *Operations Research*, 65, 4, 1043–1061. DOI: [10.1287/opre.2017.1603](https://doi.org/10.1287/opre.2017.1603). eprint: <https://doi.org/10.1287/opre.2017.1603>. <https://doi.org/10.1287/opre.2017.1603>.
- [3] Judit Bar-Ilan and Dror Zernik. 1989. Random leaders and random spanning trees. In *Distributed Algorithms*. Jean-Claude Bermond and Michel Raynal, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12. ISBN: 978-3-540-46750-2.
- [4] A. Broder. 1989. Generating random spanning trees. In *30th Annual Symposium on Foundations of Computer Science*, 442–447.
- [5] Charles J. Colbourn. 1987. *The Combinatorics of Network Reliability*. Oxford University Press, Inc., USA. ISBN: 0195049209.
- [6] Charles J Colbourn, Robert P.J Day, and Louis D Nel. 1989. Unranking and ranking spanning trees of a graph. *Journal of Algorithms*, 10, 2, 271 –286. ISSN: 0196-6774. DOI: [https://doi.org/10.1016/0196-6774\(89\)90016-3](https://doi.org/10.1016/0196-6774(89)90016-3). <http://www.sciencedirect.com/science/article/pii/0196677489900163>.
- [7] Charles J Colbourn, Bradley M Debroni, and Wendy J Myrvold. 1988. Estimating the coefficients of the reliability polynomial. *Congressus Numerantium*, 62, 217–223.
- [8] Charles J. Colbourn, Wendy J. Myrvold, and Eugene Neufeld. 1996. Two algorithms for unranking arborescences. *Journal of Algorithms*, 20, 2, 268 –281. ISSN: 0196-6774. DOI: <https://doi.org/10.1006/jagm.1996.0014>. <http://www.sciencedirect.com/science/article/pii/S0196677496900140>.
- [9] Shlomi Dolev and Daniel Khankin. 2016. Random spanning trees for expanders, sparsifiers, and virtual network security. *arXiv preprint arXiv:1612.02569*.
- [10] David Durfee, Rasmus Kyng, John Peebles, Anup B. Rao, and Sushant Sachdeva. 2017. Sampling random spanning trees faster than matrix multiplication. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*. Association for Computing Machinery, Montreal, Canada, 730–742. ISBN: 9781450345286. DOI: [10.1145/3055399.3055499](https://doi.org/10.1145/3055399.3055499). <https://doi.org/10.1145/3055399.3055499>.
- [11] Alan Frieze, Navin Goyal, Luis Rademacher, and Santosh Vempala. 2014. Expanders via random spanning trees. *SIAM Journal on Computing*, 43, 2, 497–513. DOI: [10.1137/120890971](https://doi.org/10.1137/120890971). eprint: <https://doi.org/10.1137/120890971>. <https://doi.org/10.1137/120890971>.

- [12] Wai Shing Fung and Nicholas J. A. Harvey. 2010. Graph sparsification by edge-connectivity and random spanning trees. *CoRR*, abs/1005.0265. arXiv: [1005.0265](https://arxiv.org/abs/1005.0265). <http://arxiv.org/abs/1005.0265>.
- [13] S. O. Gharan, A. Saberi, and M. Singh. 2011. A randomized rounding approach to the traveling salesman problem. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, 550–559.
- [14] Navin Goyal, Luis Rademacher, and Santosh Vempala. 2009. Expanders via random spanning trees. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA '09). Society for Industrial and Applied Mathematics, New York, New York, 576–585.
- [15] A Guénoche. 1983. Random spanning tree. *Journal of Algorithms*, 4, 3, 214–220. ISSN: 0196-6774. DOI: [https://doi.org/10.1016/0196-6774\(83\)90022-6](https://doi.org/10.1016/0196-6774(83)90022-6). <http://www.sciencedirect.com/science/article/pii/0196677483900226>.
- [16] Nicholas J. A. Harvey. 2009. Algebraic algorithms for matching and matroid problems. *SIAM J. Comput.*, 39, 2, (July 2009), 679–702. ISSN: 0097-5397. DOI: [10.1137/070684008](https://doi.org/10.1137/070684008). <https://doi.org/10.1137/070684008>.
- [17] Nicholas JA Harvey and Keyulu Xu. 2016. Generating random spanning trees via fast matrix multiplication. In *LATIN 2016: Theoretical Informatics*. Springer, 522–535.
- [18] D. Kandel, Y. Matias, R. Unger, and P. Winkler. 1996. Shuffling biological sequences. *Discrete Applied Mathematics*, 71, 1, 171–185. ISSN: 0166-218X. DOI: [https://doi.org/10.1016/S0166-218X\(97\)81456-4](https://doi.org/10.1016/S0166-218X(97)81456-4). <http://www.sciencedirect.com/science/article/pii/S0166218X97814564>.
- [19] Jonathan A. Kelner and Aleksander Madry. 2009. Faster generation of random spanning trees. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science* (FOCS '09). IEEE Computer Society, USA, 13–21. ISBN: 9780769538501.
- [20] G. Kirchhoff. 1847. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148, 12, 497–508. DOI: [10.1002/andp.18471481202](https://doi.org/10.1002/andp.18471481202). eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.18471481202>. <https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.18471481202>.
- [21] V.G Kulkarni. 1990. Generating random combinatorial objects. *Journal of Algorithms*, 11, 2, 185–207. ISSN: 0196-6774. DOI: [https://doi.org/10.1016/0196-6774\(90\)90002-V](https://doi.org/10.1016/0196-6774(90)90002-V). <http://www.sciencedirect.com/science/article/pii/019667749090002V>.
- [22] Louis D. Nel and Charles J. Colbourn. 1990. Combining monte carlo estimates and bounds for network reliability. *Networks*, 20, 3, 277–298. DOI: [10.1002/net.3230200303](https://doi.org/10.1002/net.3230200303). eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230200303>. <https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230200303>.
- [23] James Gary Propp and David Bruce Wilson. 1998. How to get a perfectly random sample from a generic markov chain and generate a random spanning tree of a directed graph. *J. Algorithms*, 27, 170–217.
- [24] Luís M. S. Russo, Andreia Sofia Teixeira, and Alexandre P. Francisco. 2018. Linking and cutting spanning trees. *Algorithms*, 11, 4. ISSN: 1999-4893. DOI: [10.3390/a11040053](https://doi.org/10.3390/a11040053). <https://www.mdpi.com/1999-4893/11/4/53>.

- [25] Aaron Schild. 2018. An almost-linear time algorithm for uniform random spanning tree generation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing* (STOC 2018). Association for Computing Machinery, Los Angeles, CA, USA, 214–227. ISBN: 9781450355599. DOI: [10.1145/3188745.3188852](https://doi.org/10.1145/3188745.3188852). <https://doi.org/10.1145/3188745.3188852>.
- [26] Nisheeth K. Vishnoi. 2013.  $Lx = b$ . *Foundations and Trends® in Theoretical Computer Science*, 8, 1–2, 1–141. ISSN: 1551-305X. DOI: [10.1561/04000000054](https://doi.org/10.1561/04000000054). <http://dx.doi.org/10.1561/04000000054>.
- [27] Wikipedia contributors. 2020. Schur complement — Wikipedia, the free encyclopedia. [Online; accessed 28-May-2020]. (2020). [https://en.wikipedia.org/w/index.php?title=Schur\\_complement&oldid=947233275](https://en.wikipedia.org/w/index.php?title=Schur_complement&oldid=947233275).
- [28] David Bruce Wilson. 1996. Generating random spanning trees more quickly than the cover time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (STOC '96). Association for Computing Machinery, Philadelphia, Pennsylvania, USA, 296–303. ISBN: 0897917855. DOI: [10.1145/237814.237880](https://doi.org/10.1145/237814.237880). <https://doi.org/10.1145/237814.237880>.