The Complexity of Network Design for s-t Effective Resistance

by

Pak Hay Chan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2018

© Pak Hay Chan 2018

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We consider a new problem of designing a network with small s-t effective resistance. In this problem, we are given an undirected graph $G = (V, E)$ where each edge e has a cost c_e and a resistance r_e, two designated vertices $s, t \in V$, and a cost budget k. Our goal is to choose a subgraph to minimize the s-t effective resistance, subject to the constraint that the total cost in the subgraph is at most k. This problem has applications in electrical network design and is an interpolation between the shortest path problem and the minimum cost flow problem.

We present algorithmic and hardness results for this problem. On the hardness side, we show that the problem is NP-hard by reducing the 3-dimensional matching problem to our problem. On the algorithmic side, we use dynamic programming to obtain a fully polynomial time approximation scheme when the input graph is a series-parallel graph. Finally, we propose a greedy algorithm for general graphs in which we add a path at each iteration and we conjecture that the algorithm is a 3.95-approximation algorithm for the problem.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Lap Chi Lau for the continuous support of my Master's study, and for his patience, motivation and immense knowledge. His guidance and encouragement helped my research and writing of this thesis tremendously.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Eric Blais and Prof. Chaitanya Swamy, for their encouragement and insightful comments.

I must also thank my research group mates, Vedat Levi Alev, Tsz Chiu Kwok, Akshay Ramachandran and Hong Zhou for the stimulating discussions, for their patience to clear up my confusion about mathematics, and for all the fun time we had in the last two years.

Last but not least, I would like to thank my parents, for their unconditional love and trust. My girlfriend Yin Ki, for her support and prayers that took me through many times of doubt and worry. Most importantly, God, for all his grace and blessings in my life. I dedicate this thesis to them.

Table of Contents

List of Figures viii

1 Overview 1

2 Background 5

2.1 Graphs and Matrices 5

2.1.1 Graphs 5

2.1.2 Matrices 6

2.2 Electrical Networks 9

2.2.1 Electrical Flow and Voltage 9

2.2.2 Effective Resistance 13

2.2.3 Energy and Thomson's Principle 15

2.3 Applications of Effective Resistance 18

2.3.1 Analyzing Random Walks 19

2.3.2 Sampling Random Spanning Trees 23

2.3.3 Spectral Sparsification 26

2.4 Network Design 30
2.4.1 Edge Connectivity and Iterative Rounding 30
2.4.2 Edge Connectivity on Directed Graphs 32
2.4.3 Element Connectivity 33
2.4.4 Vertex Connectivity 33
2.4.5 Bounded Pairwise Distance 33
2.4.6 Edge Connectivity with Degree Bounds 34
2.5 Spectral Requirements 34
2.5.1 Mixing Time 35
2.5.2 Algebraic Connectivity 35
2.5.3 Total Effective Resistance 36
2.5.4 Experimental Design 37

3 Network Design for Minimizing s-t Effective Resistance 41

3.1 Introduction 41
3.2 NP-completeness for Unit Cost Unit Resistance 43
3.3 Dynamic Programming Algorithms for Series-Parallel Graphs 47
3.3.1 Series-Parallel Graphs 48
3.3.2 Polynomial Time Algorithm for the Unit Cost Case 49
3.3.3 Fully Polynomial Time Approximation Scheme 51
3.4 Greedy Approach 56
3.4.1 Observations and Intuition 56

3.4.2 Greedy Algorithm 58
3.4.3 Analyzing the Greedy Algorithm 60
3.4.4 Discussion 67
3.5 Conclusions 72

References **74**

List of Figures

1 An illustration of why effective resistance is a natural connectivity measure 3

2 An illustration of constructing the graph G from a 3DM instance. 44

3 The subgraph H when the 3DM instance has q disjoint triples. 46

4 The subgraph H when U is non-empty. 46

5 An example of a SP-tree. 49

6 A counterexample showing that s-t effective conductance is not submodular. 57

7 A tight example of Conjecture 3.4.4. 67

8 An example of getting a 2-approximation using greedy algorithm. 68

9 A counterexample to Conjecture 3.4.4 when the resistances have arbitrary values. 69

10 An example that greedy algorithm will get $\Omega(k)$-approximation if the resistances have arbitrary values. 70

11 A counterexample showing that total effective resistance is not submodular. 71

Chapter 1

Overview

Networks appear in various forms in all areas of computer science (e.g. computer networks, social networks, biological networks, etc.) The task of designing "good" networks is a fundamental problem in different research areas from operations research to electrical engineering.

In computer science and operations research, network design problems are generally about finding a minimum cost subgraph that satisfies certain "connectivity" requirements. The most well-studied problem is the survivable network design problem [38, 2, 39, 41, 34], where the requirement is to have a specified number $r_{u,v}$ of edge-disjoint paths between every pair of vertices u, v. Other combinatorial requirements are also well studied, including vertex connectivity [46, 28, 12, 20, 48, 16] and shortest path distances [23, 22].

Besides the combinatorial connectivity requirements, designing networks with good spectral properties are also studied. Some examples include spectral expansion [45, 6], total effective resistance [37, 59], and mixing time [10]. These requirements are closely related to quantities in random walks, but much less is known about these problems in general.

In this thesis, we study a basic question in designing networks with a spectral requirement

– the effective resistance between two vertices.

First, we motivate the problem by observing that the s-t effective resistance is an interpolation between the s-t shortest path distance and the s-t edge connectivity.

Let $f \in \mathbb{R}^{|E|}$ be a unit s-t flow in G and define the ℓ_p-energy of f as $\mathcal{E}_p(f) := (\sum_e |f_e|^p)^{1/p}$. Let $\mathcal{E}_p(s,t) := \min_f\{\mathcal{E}_p(f) \mid f \text{ is a unit } s\text{-}t \text{ flow}\}$ be the minimum ℓ_p-energy of a unit s-t flow that the graph G can support. Thomson's principle (see Chapter 2.2.3) states that $\text{Reff}_G(s,t) = \mathcal{E}_2^2(s,t)$, so that a graph of small s-t effective resistance can support a unit s-t flow with small ℓ_2-energy. Note that the shortest path distance between s and t is equal to $\mathcal{E}_1(s,t)$ (i.e. the ℓ_1-energy of a flow is just the average path length and is minimized by a shortest s-t path), and so a graph with small $\mathcal{E}_1(s,t)$ has a short path between s and t. Note also that the edge-connectivity between s and t is equal to the reciprocal of $\mathcal{E}_\infty(s,t)$ (i.e. if there are k edge-disjoint s-t paths, set the flow value on each path to be $1/k$), and so a graph with small $\mathcal{E}_\infty(s,t)$ has many edge-disjoint s-t paths. As ℓ_2 is between ℓ_1 and ℓ_∞, the objective function $\text{Reff}_G(s,t) = \mathcal{E}_2^2(s,t)$ takes both the s-t shortest path distance and the s-t edge-connectivity into consideration.

The example in Figure 1 also demonstrates a simple property why ℓ_2-energy may be even more desirable over ℓ_1 and ℓ_∞ as a connectivity measure. Conceptually, adding edges to a graph G would make s and t more "connected". For ℓ_1 and ℓ_∞, however, adding extra edges to the graph may not yield a better energy if the edges added does not improve the shortest path and the edge connectivity respectively. In contrast, the ℓ_2-energy would typically improve after adding extra edges, and so ℓ_2-energy provides a smoother qualitative measure that better captures our intuition on how well s and t are connected in a network.

In addition, Thomson's principle states that the electrical flow between s and t is the unique flow that minimizes the ℓ_2-energy. So, designing a network with small s-t effective resistance has natural applications in designing electrical networks that minimize energy dissipation [26, 37, 40].

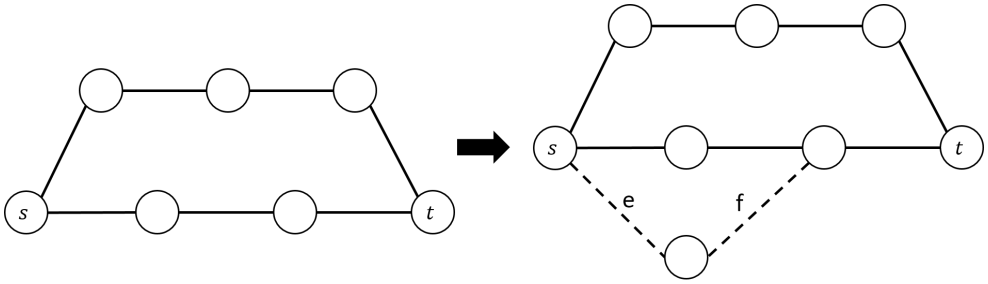


Figure 1: An illustration of why effective resistance is a natural connectivity measure

On the left, the s-t shortest path distance is 3, the s-t edge connectivity is 2 and the s-t effective resistance is $1/(1/3 + 1/4) = 12/7$. On the right, after adding edges e and f, both the s-t shortest path distance and the edge connectivity remain unchanged, but the s-t effective resistance decreases to $1/(1/4 + 1/2) = 4/3 < 12/7$.

Furthermore, the s-t effective resistance has a probabilistic interpretation as the expected commute time between s and t in a random walk [14] (see Chapter 2.3 for more applications and interpretations). Given these reasons, the s-t effective resistance can be seen as a nice and natural alternative connectivity measure between s and t in network design.

In this thesis, we study the problem of keeping the total cost of the wires to be at most k in the input electrical network to minimize the s-t effective resistance. So the electrical flow between s and t can still be sent with small amount of energy while we switch off many wires in the electrical network.

The main contributions in this thesis are as follow:

- We show that the problem is NP-hard even when all the wires (resistors) have the same resistance and the same cost. This contrasts with the shortest path distance or the edge connectivity which are solvable in polynomial time (see Chapter 3.2).
- We consider the special case when the input graph is a series-parallel graph, since

our problem is related to electrical network design. In Chapter 3.3, we use dynamic programming to design a fully polynomial time approximation scheme for the problem when the ratio between the maximum and minimum resistance is bounded, and an exact algorithm when every edge has the same cost.

- For general graphs, we propose a greedy algorithm that we add a path at each iteration. We suggest a framework to analyze the algorithm and conjecture that the algorithm has a constant approximation ratio (see Chapter 3.4).

Background material and related work of our problem are presented in Chapter 2. Formal definitions of the problem and our main results are presented in Chapter 3.

Chapter 2

Background

The aim of this chapter is to introduce the necessary background (Section 2.1-2.2), applications (Section 2.3) and related work (Section 2.4-2.5) of our problem.

2.1 Graphs and Matrices

2.1.1 Graphs

An **undirected graph** is a pair $G = (V, E)$ consists of a set V of vertices (nodes or points) and a set E of edges. Each edge $e \in E$ is a pair of vertices (u, v), indicating that there is an edge joining the two endpoints u and v. When $(u, v) \in E$, we write $u \sim v$ and say that u is a **neighbor** of v (also that v is a neighbor of u). A **weighted graph** is a graph in which there is a weight $w(e) \in \mathbb{R}$ on each edge $e \in E$. A **path** of a graph G is a sequence of distinct vertices $\{v_0, v_1, \ldots v_k\}$ such that $(v_i, v_{i+1}) \in E$ for all $0 \leq i < k$. A **cycle** of a graph G is a path where the start vertex and the end vertex are the same. A graph G is **connected** if for every pair of vertices u, v, there is a path starting at u and ending at v.

We denote the number of vertices by $n := |V|$ and the number of edges by $m := |E|$. For a subset of edges F, we write $w(F) := \sum_{e \in F} w(e)$ as the total weight of edges in F. For a subset of vertices $S \subseteq V$, we write $\delta(S)$ as the set of edges with one endpoint in S and the other endpoint in $V \backslash S$. For a vertex v, we write $\delta(v) = \delta(\{v\})$ as the set of edges incident on a vertex v, and $\text{wdeg}(v) := w(\delta(v))$ as the weighted degree of v. Throughout this thesis, we assume that G is an undirected connected graphs with positive edge weights.

2.1.2 Matrices

The **Laplacian matrix** L of an undirected graph is defined as $L := D - A$, where $D \in \mathbb{R}^{V \times V}$ is the diagonal weighted degree matrix with $D_{u,u} = \text{wdeg}(u)$ and $A \in \mathbb{R}^{V \times V}$ is the adjacency matrix of the graph where $A_{u,v} = w(uv)$ for all $u, v \in V$. By the fundamental theorem of symmetric matrices, we can write $L = \sum_{i=1}^{n} \lambda v_i v_i^T$ where $\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{R}$ are the eigenvalues of L and $v_1, v_2, \ldots, v_n$ are the corresponding eigenvectors that are orthonormal.

We now show a few facts in linear algebra and a few important properties of the Laplacian matrix.

Definition 2.1.1 (Positive Semidefinite Matrix)**.** *A real symmetric matrix M is positive semidefinite (PSD) if all eigenvalues of M are non-negative, denoted by $M \succeq 0$.*

Lemma 2.1.2. *M is positive semidefinite if and only if for all $x \in \mathbb{R}^n$, $x^T M x \geq 0$.*

Proof. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of M with the corresponding orthonormal eigenvectors $v_1, v_2, \ldots, v_n$. Note that $x^T M x = x^T (\sum_{i=1}^{n} \lambda_i v_i v_i^T) x = \sum_{i=1}^{n} \lambda_i (x^T v_i)^2$, hence

$$\forall x \in \mathbb{R}^n,\ x^T M x \geq 0 \iff \forall x \in \mathbb{R}^n,\ \sum_{i=1}^{n} \lambda_i (x^T v_i)^2 \geq 0 \iff \forall i\ \lambda_i \geq 0.$$

□

Fact 2.1.3. *The Laplacian matrix L of a graph G is positive semidefinite.*

Proof. Let L_e be the Laplacian of a graph with only one edge e. Note that $L = \sum_{e \in E} L_e$. Then for any $x \in \mathbb{R}^n$,

$$x^T L x = x^T \left(\sum_{e \in E} L_e \right) x = \sum_{e \in E} x^T L_e x = \sum_{(i,j) \in E} w(ij)(x_i - x_j)^2 \geq 0.$$

□

Lemma 2.1.4. *If G is connected, then L has rank $n - 1$.*

Proof. We show that the nullspace of L is of dimension 1. For every $x \in \text{nullspace}(L)$, i.e. $Lx = \vec{0}$, we have

$$0 = x^T L x = \sum_{(i,j) \in E} w(ij)(x_i - x_j)^2.$$

But since G is connected and $w(ij) > 0$, we must have $x_i = x_j$ for all i, j. Hence, the vector x must be a multiple of $\vec{1}$. □

Fact 2.1.5. *$\lambda_1 = 0$ and the corresponding eigenvector is $v_1 = \vec{1}/\sqrt{n}$.*

The **pseudo-inverse** of the Laplacian matrix L of a connected graph is defined as

$$L^\dagger := \sum_{i=2}^{n} \frac{1}{\lambda_i} v_i v_i^T,$$

which maps every vector x orthogonal to v_1 to the unique vector b such that $Lx = b$ and $b \perp \vec{1}$.

When the graph is connected, we can characterize the set of solutions to the Laplacian system $Lx = b$.

Lemma 2.1.6. *Let L be the Laplacian matrix of a connected graph G. The Laplacian system $Lx = b$ has solutions if and only if b is perpendicular to $\vec{1}$. If $b \perp \vec{1}$, then the set of all solutions for $Lx = b$ is $\{L^\dagger b + c\vec{1} | c \in \mathbb{R}\}$.*

Proof. Since G is connected, from Fact 2.1.5, we know that nullspace$(L) = \vec{1}$. So, for every $x \in \mathbb{R}^n$, Lx is always perpendicular to $\vec{1}$. Therefore, for $Lx = b$ to have a solution, it is necessary that b is perpendicular to $\vec{1}$.

On the other hand, it is sufficient for $b \perp \vec{1}$ so that $Lx = b$ has a solution. To see this, if $b \perp \vec{1}$, then we can write $b = \sum_{i=2}^{n} a_i v_i$ for some $a_2, \ldots a_n \in \mathbb{R}^n$. Now we can let $x = \sum_{i=2}^{n} \frac{a_i}{\lambda_i} v_i$ such that $Lx = L\left(\sum_{i=2}^{n} \frac{a_i}{\lambda_i} v_i\right) = \sum_{i=2}^{n} \frac{a_i}{\lambda_i} L v_i = \sum_{i=2}^{n} \frac{a_i}{\lambda_i} (\lambda_i v_i) = b$. So x is a solution to the system. Note that $L^\dagger b = \left(\sum_{i=2}^{n} \frac{1}{\lambda_i} v_i v_i^T\right) \left(\sum_{i=2}^{n} a_i v_i\right) = \sum_{i=2}^{n} \frac{a_i}{\lambda_i} v_i = x$, hence $L^\dagger b$ is a solution to the system.

To characterize the set of all solutions, note that $L^\dagger$ maps any vector b in the range of L to the unique vector x such that $Lx = b$ and $x \perp$ kernel(L). Since the graph is connected, kernel$(L) = \{c\vec{1} | c \in \mathbb{R}\}$. So the set of all solutions satisfying $Lx = b$ is $\{L^\dagger b + c\vec{1} | c \in \mathbb{R}\}$.

□

When G is connected, since we know the nullspace of L, we can manipulate the pseudo-inverse by manipulating the inverse of a related matrix.

Fact 2.1.7. *Let $J = \vec{1}\vec{1}^T$ be the all one matrix. Then for any $b \perp \vec{1}$, $L^\dagger b = (L + J)^{-1} b$.*

Proof. Since $b \perp \vec{1}$, then we can write $b = \sum_{i=2}^{n} a_i v_i$ for some $a_2, \ldots a_n \in \mathbb{R}^n$. Then we have $(L + J)^{-1} b = \left(\frac{1}{n}\vec{1}\vec{1}^T + \sum_{i=2}^{n} \frac{1}{\lambda_i} v_i v_i^T\right) b = \frac{1}{n}\vec{1}\vec{1}^T b + L^\dagger b = L^\dagger b$, where the last equality follows from the fact that $\vec{1}^T b = 0$. □

Sherman and Morrison gave the formula to compute the inverse of a matrix A when updated by a rank one matrix uv^T.

Theorem 2.1.8 (Sherman-Morrison formula)**.** *Suppose $A \in \mathbb{R}^{n \times n}$ is an invertible square matrix and $u, v \in \mathbb{R}^n$ are column vectors. Then $A + uv^T$ is invertible if and only if $1 + v^T A^{-1} u \neq 0$. If $A + uv^T$ is invertible, then its inverse is given by*

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}.$$

Combined with Fact 2.1.7, we can use the Sherman-Morrison formula to compute a solution to a Laplacian system when the Laplacian matrix is updated by a rank-1 matrix.

2.2 Electrical Networks

Given a weighted graph, if all the edge weights are positive, we can interpret the graph as an **electrical network**, where each edge e is regarded as a conductor with conductance $w(e)$, or resistance $r(e) = 1/w(e)$. In the following subsections, we will introduce some important concepts of electrical networks.

2.2.1 Electrical Flow and Voltage

Before we define the electrical flow, we first define the standard s-t flows. Given an undirected graph $G = (V, E)$, let $k \geq 0$ be the net flow out of source s and into sink t. A k-unit **s-t flow** is a vector $f \in \mathbb{R}^{2|E|}$ defined on oriented edges which satisfies the following:

- (Antisymmetry) $f(uv) = -f(vu)$ for all $(u, v) \in E$.
- (Flow conservation) For every vertex other than s and t, the incoming flow is equal to the outgoing flow, i.e.

$$\sum_{u:v\sim u} f(vu) = \begin{cases} k & \text{if } v = s \\ -k & \text{if } v = t \\ 0 & \text{if } v \in V\backslash\{s, t\}. \end{cases}$$

For each edge $e = (u, v)$, $f(uv)$ is positive if the flow is going from u to v and negative otherwise. We define $\boldsymbol{\nu}(f) := k$ to be the flow value of f. When $k = 1$, we call f a **unit s-t flow**.

An s-t **electrical flow** is an s-t flow that satisfies **Ohm's law**, such that there exists a potential vector $p \in \mathbb{R}^V$ so that for every edge $(u, v) \in E$,

$$(p(u) - p(v)) = r(uv) \cdot f(uv).$$

Ohm's law states that the amount of electrical flow passing through a resistor between two points is directly proportional to the electrical potential difference between the two points. In the context of electrical network, we sometimes called the electrical flow **current** and the potential **voltage**.

Matrix Formulation: Next, we write the relation between the potential and current in matrix form. For $u, v \in V$, let $b_{uv} = \chi_u - \chi_v$, where $\chi_v \in \mathbb{R}^n$ is the unit vector with 1 in the v-th entry and 0 in other entries.

For each vertex v, by flow conservation and Ohm's law, the potential vector $p \in \mathbb{R}^V$ of a unit s-t electrical flow satisfies

$$\sum_{u:v\sim u} f(vu) = \sum_{u:v\sim u} (p(v) - p(u)) \cdot w(vu) = \begin{cases} 1 & \text{if } v = s \\ -1 & \text{if } v = t \\ 0 & \text{otherwise.} \end{cases}$$

Writing the above equations in matrix form, we can see that the potential vector p is a solution to the following linear system:

$$L \cdot p = b_{st}, \tag{2.2.1}$$

where L is the Laplacian matrix of the graph defined in Section 2.1. One solution to the system is $p = L^{\dagger} b_{st}$. Since G is connected and L has rank $n - 1$, any solution has the form $p + c \cdot \vec{1}$ for some $c \in R$ (Lemma 2.1.6).

The vector b_{st} in the linear system is also called the **external current** vector which specifies the net external current to each vertex. The external current vector b should be parallel to $\vec{1}$ to guarantee flow conservation.

By Ohm's law, the flow on each edge only depends on the potential difference between its two endpoints, hence the electrical flow is unique given the external current vector.

Electrical flow is also equivalent to a flow that satisfies the **Kirchhoff's cycle law**. Later we will use this property to prove Thomson's Principle (Theorem 2.2.6).

Proposition 2.2.1 (Kirchhoff's cycle law)**.** *Let f be an electrical flow. Then, for any cycle $\{v_0, v_1, \ldots, v_k = v_0\}$, we have*

$$\sum_{i=0}^{k-1} r(v_i v_{i+1}) f(v_i v_{i+1}) = 0.$$

Kirchhoff's cycle law states that for every cycle in the network, the sum of the potential differences of each edge in the cycle is zero.

Proof. By Ohm's law,

$$\sum_{i=0}^{k-1} r(v_i v_{i+1}) f(v_i v_{i+1}) = \sum_{i=0}^{k-1} r(v_i v_{i+1}) \frac{1}{r(v_i v_{i+1})} (p(v_i) - p(v_{i+1})) = \sum_{i=0}^{k-1} (p(v_i) - p(v_{i+1})) = 0.$$

□

Conversely, we can show that if an s-t flow satisfies Kirchhoff's cycle law then it is an electrical flow.

Proposition 2.2.2. *Let f^* be an electrical flow. If f is an s-t flow that satisfies Kirchhoff's cycle law and $\boldsymbol{\nu}(f) = \boldsymbol{\nu}(f^*)$, then $f = f^*$.*

Proof. Let $d = f^* - f$ be the difference between the two s-t flows. By flow conservation and the cycle law of both f^* and f, d is also an s-t flow that satisfies the cycle law. If d is not a zero flow, then we will show that we can find a cycle that violates cycle law. Suppose $d \neq \mathbf{0}$ and $d(uv) > 0$ for some $u, v \in V$. Then, by flow conservation, v must lead

to some other vertex v' with $d(vv') > 0$. We can iterate the process repeatedly to obtain a sequence of vertices where the flow value on the edge is strictly positive. Since the network is finite, this sequence must revisit a vertex and form a cycle. This cycle would violate the cycle law. □

Next, we show that given an s-t flow, the potential at s and t serve as a boundary of the potential of all other vertices.

Fact 2.2.3. *Let f be a non-zero s-t electrical flow and p be a corresponding potential vector. Then, for every vertex x, we have $p(t) < p(x) < p(s)$.*

Proof. By flow conservation and Ohm's law, for any vertex $x \in V \backslash \{s, t\}$, we have

$$0 = \sum_{u:u\sim x} f(ux) = \sum_{u:u\sim x} (p(u) - p(x))w(ux).$$

Rearranging, we have

$$p(x) = \sum_{u:u\sim x} \frac{w(ux)}{\text{wdeg}(x)} p(u),$$

which shows that the potential at x is a convex combination of all of its neighbor. Now we will show that $p(x) < p(s)$ for any vertex x. Let $A := \{a : p(a) = \max_a p(a)\}$. We need to show that $A = \{s\}$. Suppose to the contrary that there exists a vertex $x \neq s, t$ in A. If x has a neighbor y where $p(y) < p(x)$, using the fact that the above convex combination is true for $v \neq s, t$, we have

$$p(x) = \sum_{z:z\sim x} \frac{w(zx)}{\text{wdeg}(x)} p(z) = \frac{w(yx)}{\text{wdeg}(x)} p(y) + \sum_{z:z\sim x, z\neq y} \frac{w(zx)}{\text{wdeg}(x)} p(z) < \max_a p(a),$$

a contradiction. So it follows that $p(y) \in A$ for every neighbor y of x. Since the graph is connected, there is always a path from x to s and t and hence $s, t \in A$. Since p corresponds to a non-zero s-t electrical flow, we must have $p(s) > p(t)$ and thus a contradiction. Similarly, we can show that $p(t) < p(x)$ for any vertex x to complete the proof. □

2.2.2 Effective Resistance

Given an electrical flow f and the corresponding potential vector p, the effective resistance between s and t is defined as

$$\text{Reff}(s,t) = \frac{p(s) - p(t)}{\nu(f)},$$

which is the ratio between the potential difference between s and t and the flow value. Informally, the s-t effective resistance indicates how "hard" it is to send electrical flow between s and t. It can be interpreted as the resistance of the whole graph G as a big resistor.

There are many applications and interpretations of effective resistance. See Section 2.3.

We can express the effective resistance in terms of the Laplacian matrix. Recall in Equation 2.2.1 that the potential vector $p \in \mathbb{R}^V$ of a unit s-t electrical flow is a solution to the linear system

$$L \cdot p = b_{st}.$$

Note that $p = L^\dagger b_{st}$ is a solution. Since L has rank $n-1$ (G is connected), the set of solutions has the form of $p + c \cdot \vec{1}$ for $c \in \mathbb{R}$ (Lemma 2.1.6). Hence, by the definition of effective resistance, we can write

$$\text{Reff}(s,t) = p(s) - p(t) = b_{st}^T L^\dagger b_{st}. \tag{2.2.2}$$

The s-t effective conductance is the inverse of the s-t effective resistance, i.e.

$$\text{Ceff}(s,t) = \frac{1}{\text{Reff}(s,t)}. \tag{2.2.3}$$

Effective resistance provides an alternative way to measure the distance of two vertices in a graph. It can be used as a distance metric to identify cluster in a graph [5]. To show that effective resistance is a distance metric, it is sufficient to check it satisfies the triangle inequality.

Theorem 2.2.4 (Resistance Metric)**.** *Let a, b and c be vertices in a graph. Then*

$$\text{Reff}(a, c) \leq \text{Reff}(a, b) + \text{Reff}(b, c).$$

Proof. The main idea is to add the unit current flow from a to b to the unit current flow from b to c that gives the unit current flow from a to c. Then we consider the corresponding voltage vectors. Let $\overrightarrow{v_{ab}}, \overrightarrow{v_{ac}}, \overrightarrow{v_{bc}}$ be the voltage vectors when one unit of current is sent from a to b, a to c, b to c respectively, i.e.

$$\overrightarrow{v_{ab}} = L_G^\dagger(\chi_a - \chi_b), \quad \overrightarrow{v_{ac}} = L_G^\dagger(\chi_a - \chi_c), \quad \overrightarrow{v_{bc}} = L_G^\dagger(\chi_b - \chi_c).$$

Summing up the three equations, we have $\overrightarrow{v_{ab}} + \overrightarrow{v_{bc}} = \overrightarrow{v_{ac}}$. By the definition of effective resistance, we have

$$\text{Reff}(a, c) = (\chi_a - \chi_c)^T \overrightarrow{v_{ac}} = (\chi_a - \chi_c)^T \overrightarrow{v_{ab}} + (\chi_a - \chi_c)^T \overrightarrow{v_{bc}}.$$

Note that

$$(\chi_a - \chi_c)^T \overrightarrow{v_{ab}} = \overrightarrow{v_{ab}}(a) - \overrightarrow{v_{ab}}(c) \leq \overrightarrow{v_{ab}}(a) - \overrightarrow{v_{ab}}(b) = \text{Reff}(a, b),$$

since from Fact 2.2.3, we have $\overrightarrow{v_{ab}}(b) \leq \overrightarrow{v_{ab}}(c) \leq \overrightarrow{v_{ab}}(a)$ for all vertex $c \in V$. Similarly, we have $(\chi_a - \chi_c)^T \overrightarrow{v_{bc}} \leq \text{Reff}(b, c)$, and the theorem follows. □

To measure of how well "connected" the electrical network is in terms of the average resistance distance, sometimes we consider the total effective resistance of the network, i.e. the sum of the effective resistance between all pairs of vertices. The following theorem shows that the total effective resistance is proportional to the trace of the pseudo-inverse of the Laplacian matrix.

Theorem 2.2.5 (Total Effective Resistance)**.** *Let* $\text{Reff}_{total} = \sum_{i=1}^{n} \sum_{j=1}^{n} \text{Reff}(i, j)$ *be the sum of the effective resistances between all pairs of vertices. Then*

$$\text{Reff}_{total} = 2n \cdot \text{tr}(L^\dagger).$$

Proof.

$$\text{Reff}_{\text{total}} = \sum_{i,j} \text{Reff}(i,j) = \sum_{i,j} b_{ij}^T L^\dagger b_{ij} = \sum_{i,j} \left(L^\dagger_{(ii)} + L^\dagger_{(jj)} - L^\dagger_{(ij)} - L^\dagger_{(ji)} \right)$$
$$= 2n \sum_i L^\dagger_{(ii)} - 2 \sum_{i,j} L^\dagger_{(ij)} = 2n \cdot \text{tr}(L^\dagger),$$

where the last equality follows from $L^\dagger \vec{1} = 0$ and thus each row of $L^\dagger$ sums to zero. □

2.2.3 Energy and Thomson's Principle

We define the **energy** of an s-t flow f by

$$\mathcal{E}(f) := \sum_{e \in E} r(e) f(e)^2.$$

Theorem 2.2.6 (Thomson's Principle [42])**.**

$$\text{Reff}(s,t) = \inf\{\mathcal{E}(f) : f \text{ is a unit } s\text{-}t \text{ flow}\}.$$

The unique minimizer in the infimum is the unit s-t electrical flow.

Proof. The set of unit s-t flows is a closed and bounded subset of $\mathbb{R}^{|E|}$, hence by compactness there exists a flow f with $\nu(f) = 1$ minimizing $\mathcal{E}(f)$. By Proposition 2.2.2, to show that the unit electrical flow is the unique minimizer, it is enough to check that any unit flow f of minimal energy satisfies the cycle law.

Let $\{v_0, v_1, \ldots, v_k\}$ be any cycle and let $\{e_1, \ldots e_k\}$ be the corresponding oriented edges. Let θ be a zero s-t flow with $\theta(e_i) = 1$ for all $1 \leq i \leq k$ and zero on all other edges. By the minimality of the energy, for any $\varepsilon \in \mathbb{R}$, we have

$$0 \leq \mathcal{E}(f + \varepsilon\theta) - \mathcal{E}(f) = \sum_{i=1}^{k} \left[(f(e_i) + \varepsilon)^2 - f(e_i)^2 \right] r(e_i)$$
$$= 2\varepsilon \sum_{i=1}^{k} r(e_i) f(e_i) + \varepsilon^2 r(e_i).$$

Dividing both sides by $\varepsilon > 0$ we have

$$0 \leq 2\sum_{i=1}^{k} r(e_i)f(e_i) + \varepsilon r(e_i),$$

and letting $\varepsilon \to 0$ shows that $0 \leq \sum_{i=1}^{k} r(e_i)f(e_i)$. Similarly dividing $\varepsilon < 0$ and letting $\varepsilon \to 0$ shows that $0 \geq \sum_{i=1}^{k} r(e_i)f(e_i)$. Therefore, we must have $\sum_{i=1}^{k} r(e_i)f(e_i) = 0$, showing that f satisfies the cycle law and hence f is the unique electrical flow minimizing the energy, by Proposition 2.2.2.

Finally, we show $\mathcal{E}(f) = \text{Reff}(s,t)$ when f is the unit s-t electrical flow. Let p be the corresponding potential vector of f such that $Lp = b_{st}$. Then we have

$$\mathcal{E}(f) = \sum_{uv \in E} r(uv)f^2(uv) = \sum_{uv \in E} w(uv)((p(u) - p(v))^2 = p^T \left(\sum_{uv \in E} w(uv) b_{uv} b_{uv}^T \right) p = p^T L p.$$

On the other hand, we have

$$\text{Reff}(s,t) = b_{st}^T L^\dagger b_{st} = (Lp)^T L^\dagger (Lp) = p^T L L^\dagger L p = p^T L p,$$

which completes the proof. □

From Thomson's Principle, we can show that the effective resistance is monotonic, i.e. increasing the resistance of an edge would not decrease the effective resistance between any two vertices.

Theorem 2.2.7 (Rayleigh's Monotonicity Law)**.** *If $\{r(e)\}$ and $\{r'(e)\}$ are sets of resistances on the edges of the same connected graph $G = (V, E)$ with $r(e) \leq r'(e)$ for all $e \in E$, then for all $x, y \in V$*

$$\text{Reff}_r(x,y) \leq \text{Reff}_{r'}(x,y).$$

Proof. Let f^* be the unit x-y electrical flow on the resistances $\{r'(e)\}$. By Thomson's Principle, we have

$$\text{Reff}_r(x,y) = \inf_f \sum_e r(e)f(e)^2 \leq \sum_e r(e)f^*(e)^2 \leq \sum_e r'(e)f^*(e)^2 = \text{Reff}_{r'}(x,y).$$

□

In addition, we can show that effective resistance is a convex function in terms of conductances using Thomson's principle.

Theorem 2.2.8 (Convexity of Effective Resistance)**.** *Let* $\mathrm{Reff}_x(s,t)$ *be the effective resistance when* $x \in \mathbb{R}_{>0}^{|E|}$ *is the conductance on the edges. Then,* $\mathrm{Reff}_x(s,t)$ *is convex with respect to* x*.*

Proof. It suffices to show that $\mathrm{Reff}_{(x+y)/2}(s,t) \leq \frac{1}{2}(\mathrm{Reff}_x(s,t) + \mathrm{Reff}_y(s,t))$ for any $x, y \in \mathbb{R}_{>0}^{|E|}$. Let $f_x(e)$ and $f_y(e)$ be the unit electrical flow on e with respect to the conductance vector x and y. First, we show that for any edge e,

$$\frac{f_x(e)^2}{x_e} + \frac{f_y(e)^2}{y_e} \geq \frac{(|f_x(e)| + |f_y(e)|)^2}{x_e + y_e}.$$

It is equivalent to

$$\begin{aligned} & f_x(e)^2\left(1 + \frac{y_e}{x_e}\right) + f_y(e)^2\left(1 + \frac{x_e}{y_e}\right) \geq f_x(e)^2 + f_y(e)^2 + 2|f_x(e)f_y(e)| \\ \iff \quad & \frac{|f_x(e)|y_e}{|f_y(e)|x_e} + \frac{|f_y(e)|x_e}{|f_x(e)|y_e} \geq 2, \end{aligned}$$

which is true since $a + 1/a \geq 2$ for $a > 0$.

Therefore, we have

$$\begin{aligned} \frac{\mathrm{Reff}_x(s,t) + \mathrm{Reff}_y(s,t)}{2} &= \sum_{e\in E} \frac{f_x(e)^2}{2x_e} + \sum_{e\in E} \frac{f_y(e)^2}{2y_e} \\ &\geq \sum_{e\in E} \frac{(|f_x(e)| + |f_y(e)|)^2}{2x_e + 2y_e} \geq \sum_{e\in E} \frac{(f_x(e) + f_y(e))^2}{2x_e + 2y_e} \\ &= \sum_{e\in E} \left(\frac{f_x(e) + f_y(e)}{2}\right)^2 \cdot \frac{2}{x_e + y_e} \\ &\geq \mathrm{Reff}_{(x+y)/2}(s,t), \end{aligned}$$

where the first equality and the last inequality follows from Thomson's Principle. □

2.3 Applications of Effective Resistance

Effective resistance has proven to have many interpretations and applications in different contexts. It is connected to the study of random walks on undirected graphs [24, 67], where it is directly related to the commute time and cover time. Effective resistance also has combinatorial interpretations on graphs. Kirchhoff [43] discovered that the marginal probability of an edge in a random spanning tree of a graph is proportional to the effective resistance between the two endpoints of the edge. Subsequent to their work, there has been substantial research studying the use of effective resistance in molecular graphs [9, 52]. Another direct application is in the study of electrical network design, where usually the primary objective is to optimize power dissipation [37].

Recently, effective resistance has found various applications in designing fast algorithms for graph problems. Recall that to find the electrical flow or a potentials vector of an electrical network, we need to solve a Laplacian system in the form $Lx = b$. In a breakthrough work in 2004, Spielman and Teng gave the first nearly-linear time Laplacian solver [65]. Followed by the effort of many researchers, it is now known that Laplacian systems can be solved in $\tilde{O}(m\sqrt{\log n})$ time [21], where m is the number of nonzero entries and n is the dimension of the Laplacian matrix. With this fundamental tool of computing electrical flows quickly, effective resistance has found surprising applications in designing fast algorithms for graph problems, including constructing spectral sparsifiers [64], computing maximum flow [19], generating random spanning trees [53, 63] and graph clustering [5].

In the following subsections, we will present some classical results of using effective resistance in analyzing random walks (section 2.3.1), in studying random spanning trees (section 2.3.2), and in constructing spectral sparsifiers (section 2.3.3). The aim of these subsections are to motivate the study of effective resistance. Readers who are familiar with these topics could skip these subsections.

2.3.1 Analyzing Random Walks

In this subsection, we will show how effective resistance is related to the commute time between a pair of vertices and the cover time of a random walk.

A Markov chain is a sequence of random variables $(X_0, X_1, X_2, \dots)$ with state space Ω and a stochastic transition matrix $P \in \mathbb{R}_{\geq 0}^{\Omega \times \Omega}$ satisfying the Markov property, namely that the probability of moving to the next state depends only on the present state and not on the previous states, i.e.

$$\Pr(X_{n+1} = x \mid X_0 = x_0, X_1 = x_1, \dots, X_n = x_n) = \Pr(X_{n+1} = x \mid X_n = x_n) = P(x_n, x),$$

if both conditional probabilities are well defined ($\Pr(X_0 = x_0, X_1 = x_1, \dots, X_n = x_n) > 0$).

Given an undirected weighted graph $G = (V, E, \{w_e\})$, we can define a simple **random walk** on G to be the Markov chain with state space V and transition matrix

$$P(x, y) = \begin{cases} \dfrac{w(xy)}{\operatorname{wdeg}(x)} & \text{if } y \sim x \\ 0 & \text{otherwise.} \end{cases}$$

We demonstrate how effective resistance is related to the commute time and cover time of a random walk. The **hitting time** from a to b, denoted by h_{ab}, is the expected number of steps to reach b if the random walk starts at a. The **commute time** is the expected number of steps to reach b and back to a if the random walk starts at a, defined as $c_{ab} := h_{ab} + h_{ba}$. The **cover time** is the expected number of steps to reach every state at least once, regardless of the starting state.

The following theorem gives an explicit relation between commute time and effective resistance.

Theorem 2.3.1. $c_{st} = \left(\sum_{v \in V} \operatorname{wdeg}(v)\right) \cdot \operatorname{Reff}(s, t)$.

Proof. Let $v \in V\backslash\{t\}$. From the definition of h_{vt}, we have

$$h_{vt} = \sum_{x:vx\in E} \frac{w(ux)}{\text{wdeg}(v)}(1 + h_{xt})$$

and $h_{tt} = 0$. After rearranging the terms, we have

$$\text{wdeg}(v) = \sum_{x:vx\in E} w(vx)(h_{vt} - h_{xt}) \text{ for all } v \in V\backslash\{t\},$$

which is equivalent to a Laplacian system of linear equations.

Let $W = \sum_{v\in V} \text{wdeg}(v)$. Let ϕ_{vt} be the voltage at v with $\phi_{tt} = 0$ when $\text{wdeg}(v)$ units of currents are injected from $v \in V\backslash\{t\}$ and $W - \text{wdeg}(t)$ units of current are removed from t.

We claim that the values ϕ_{vt} and h_{vt} satisfy the same Laplacian system. Let b_t be the vector of the external currents with $b_t(v) = \text{wdeg}(v)$ for $v \in V\backslash\{t\}$ and $b_t(t) = -W + \text{wdeg}(t)$. By Ohm's law, for $v \in V$, $b_t(v) = \sum_{x:vx\in E} w(vx)(\phi_{vt} - \phi_{xt})$.

Let $\vec{\phi_t}$ be a vector in $\mathbb{R}^V$. Then the values of ϕ_{vt} is the solution to the Laplacian system $L_G\vec{\phi_t} = b_t$ with $\vec{\phi_t}(t) = 0$. Recall that the set of the solution to the system is $\{L_G^\dagger b_t + c\vec{1} | c \in R\}$. Hence there is a unique solution with $\vec{\phi_t}(t) = 0$ and therefore we must have $h_{vt} = \phi_{vt}$. Let $\vec{h_t}$ be the hitting time vector with $\vec{h_t}(v) = h_{vt}$.

Similarly, let b_s be the vector of external currents with $b_s(v) = \text{wdeg}(v)$ for $v \in V\backslash\{s\}$ and $b_s(s) = -W + \text{wdeg}(s)$. Let $\vec{h_s}$ be the hitting time vector with $\vec{h_s}(v) = h_{vs}$ and $\vec{h_s}(s) = h_{ss} = 0$. Then $\vec{h_s}$ is the unique solution to $L_G\vec{h_s} = b_s$ with $\vec{h_s}(s) = 0$.

Note that $L_G(\vec{h_t} - \vec{h_s}) = b_t - b_s = W(\chi_s - \chi_t)$, so $(\vec{h_t} - \vec{h_s})/W = L_G^\dagger(\chi_s - \chi_t)$ is the voltage vector when one unit of current is sent from s to t.

Finally, by the definition of s-t effective resistance, we have

$$\begin{aligned}\text{Reff}(s,t) = (\chi_s - \chi_t)^T L_G^\dagger (\chi_s - \chi_t) &= (\chi_s - \chi_t)^T (\overrightarrow{h_t} - \overrightarrow{h_s})/W \\ &= \frac{1}{W}(\overrightarrow{h_t}(s) + \overrightarrow{h_s}(t)) \\ &= \frac{1}{W}(h_{st} + h_{ts}) \\ &= \frac{1}{W} c_{st}\end{aligned}$$

and the claim follows. □

Theorem 2.3.2. *For an unweighted graph G, let $R(G) = \max_{u,v} \text{Reff}(u,v)$ be the effective resistance diameter and let t_{cover} be the cover time. Then*

$$mR(G) \leq t_{cover} \leq 2e^3 mR(G) \log n + n.$$

Proof. First, observe that for any edge uv, $c_{uv} \leq 2m$ since the effective resistance between u and v in an unweighted graph is at most 1. Using this observation, we can show that the cover time of an unweighted graph is at most $2m(n-1)$.

Let T be a spanning tree of G. Consider a walk that goes through T where each edge in T is traversed once in each direction. Then this is a walk that visits every vertex at least once. So the cover time of G is bounded by the expected length of this walk, which is at most

$$\sum_{uv \in T} (h_{uv} + h_{vu}) = \sum_{uv \in T} c_{uv} \leq 2m(n-1).$$

We show the lower bound on the cover time. Let $R(G) = \text{Reff}(u,v)$ for some u, v. Then $2m\text{Reff}(u,v) = c_{uv} = h_{uv} + h_{vu}$. So the cover time is at least $\max\{h_{uv}, h_{vu}\} \geq c_{uv}/2 = m\text{Reff}(u,v)$.

For the upper bound, since the maximum hitting time is at most $2mR(G)$, regardless the starting vertex. Therefore, the probability that a vertex is not covered after $2e^3 m \cdot$

$R(G)$ steps is at most $1/e^3$ by the Markov's inequality. So if the random walk runs for $2e^3 mR(G)\log n$ steps, then a vertex is not covered with probability at most $1/n^3$. By a union bound over all vertices, the probability of some vertex is not covered after $2e^3 mR(G)\log n$ steps is at most $1/n^2$. When this happens, we can use the bound that the cover times is at most $2m(n-1) \leq n^3$. Then the cover time is at most $2e^3 mR(G)\log n + n^3/n^2 = 2e^3 mR(G)\log n + n$.

□

Furthermore, the effective resistance of an edge (s,t) is directly related to the probability that a random walk starts at s reaches t for the first time by the edge (s,t). The following theorem would be useful in analyzing the algorithm of generating random spanning tree in Section 2.3.2.

Theorem 2.3.3. *Let (s,t) be an edge and $Q(s,t)$ be the probability that a random walk started at s reaches t for the first time by the edge (s,t). Then*

$$Q(s,t) = w(st) \cdot \mathrm{Reff}(s,t).$$

Proof (Sketch). Let $h(a)$ be the probability that a random walk that starts at a would visit t before s. Observe that $Q(s,t)$ can be expressed by the following recurrence:

$$Q(s,t) = \frac{w(st)}{\deg(s)} + \left(1 - \sum_{x \in V \setminus \{s\}} P(s,x)h(x)\right) Q(s,t).$$

Similar to the proof of Theorem 2.3.1, by writing the recurrence of vector h, we could see that h is a potential vector when we send $1/\mathrm{Reff}(s,t)$ unit of electrical flow from s to t. The theorem would follow by substituting the values of $h(x)$. □

2.3.2 Sampling Random Spanning Trees

In this subsection, we will show the classic results [11, 4] of relating random spanning trees to effective resistances. For simplicity, we assume that G is unweighted and d-regular in the following. The results can be generalized to weighted and non-regular graphs.

Theorem 2.3.4. *For any unweighted d-regular graph $G = (V, E)$, any edge $(s, t) \in E$ and T be a spanning tree uniformly sampled from the set of spanning trees,*

$$\Pr[(s, t) \in T] = \mathrm{Reff}(s, t).$$

This theorem immediately gives a new interpretation of effective resistance as a probability distribution by observing the following lemma.

Lemma 2.3.5. $\sum_{ij \in E} \mathrm{Reff}(i, j) = n - 1$.

Proof. Note that any spanning tree has $n - 1$ edges, so we have $n - 1 = \sum_{ij \in E} \Pr[(s, t) \in T] = \sum_{ij \in E} \mathrm{Reff}(s, t)$. □

Proof of Theorem 2.3.4. The proof is by Algorithm 1 that generates a random spanning tree. We will use Theorem 2.3.3 to show that an edge (s, t) is in the spanning tree generated by the algorithm with probability $\mathrm{Reff}(s, t)$. And then we will prove that the algorithm generates a uniform random spanning tree. The theorem follows by combining the two steps.

Note that Algorithm 1 only adds an edge that connects an uncovered vertex to a covered vertex, so $|T| = n - 1$, the edges in T are connected and hence T is a spanning tree. An edge (s, t) is in T if and only if the walk reaches t for the first time by the edge (s, t). We assume without loss of generality that the algorithm covers s before t and so it is equivalent that the walk start at s. Then, by Theorem 2.3.3, the probability that an edge $(s, t) \in T$ is $\mathrm{Reff}(s, t)$. So it remains to show the algorithm generates a uniform random spanning tree.

Algorithm 1 Generating Random Spanning Tree using Random Walk

```
Let u be a random vertex in G as the start vertex of the random walk
while the walk has not covered all vertices do
    Go to a random neighbor v of u
    if it is the first time we reach v then
        Add (u, v) to T
    end if
end while
return T
```

In addition, the expected runtime of Algorithm 1 is $O(n^3)$ by the upper bound of cover time in Theorem 2.3.2.

Let $Y = X_0, X_1, \ldots, X_t, \ldots$ be the sample path of the random walk in G and let $Y_t = X_0, X_1, \ldots X_t$ be the first t steps of the walk. First we define two directed forests for this walk. We define the **forward forest** $\mathrm{FF}(Y_t)$ as follows: for any vertex v in Y_t except the start vertex, we add the reversed edge we used when we first enter v, i.e. if $t_v = \min\{l : X_l = v\}$, then we add (X_{t_v}, X_{t_v-1}) to $\mathrm{FF}(Y_t)$. Since the vertices must be visited in some order, $\mathrm{FF}(Y_t)$ does not have any cycles and hence it is a forest. Note that when the walk Y_t covers all vertices, $\mathrm{FF}(Y_t)$ is a directed spanning tree rooted at the starting vertex u and it would not change afterwards.

Similarly in the **backward forest** $\mathrm{BF}(Y_t)$, for any vertex v in Y_t except the last vertex, we add the forward edge we used in our last visit to v, i.e. if $t_v = \max\{l : X_l = v\}$, then we add (X_{t_v}, X_{t_v+1}) to $\mathrm{BF}(Y_t)$. Let $\mathrm{rev}(Y_t)$ be the reverse walk of Y_t, we can see that

$$\mathrm{BF}(\mathrm{rev}(Y_t)) = \mathrm{FF}(Y_t).$$

The key idea of the proof is that, when Y covers the whole graph at time t^*, $\mathrm{FF}(Y_{t^*})$ and $\mathrm{BF}(Y_{t^*})$ will both be rooted spanning tree of G. We know that for $t \geq t^*$, $\mathrm{FF}(Y_t)$ remains

unchanged while $\mathrm{BF}(Y_t)$ is changing. We will show that starting from $t = t^*$, $\mathrm{BF}(Y_t)$ is a Markov chain on all rooted spanning trees of G with uniform distribution. This means as $t \to t^*$, $\mathrm{BF}(Y_t)$ will be a uniform rooted spanning tree of G.

Since the graph is d-regular, the probability of traversing from any state A to any state B is equal to the probability of traversing from B to A. Also note that X_0 is chosen uniformly at random, so Y_t and $\mathrm{rev}(Y_t)$ are identically distributed. Therefore $\mathrm{BF}(\mathrm{rev}(Y_t))$ is also identically distributed as $\mathrm{BF}(Y_t)$ for $t \geq t^*$. Recall $\mathrm{BF}(\mathrm{rev}(Y_t)) = \mathrm{FF}(Y_t)$ from the above, so $\mathrm{FF}(Y_t)$ is a uniform rooted spanning tree of G. By dropping the direction of the edges we could obtain a uniform spanning tree.

It remains to show that $\mathrm{BF}(Y_t)$ converges to uniform distribution on all spanning trees of G, we need to show (1) there is a path between any two trees, (2) for any rooted tree T we go to d rooted trees each with probability $1/d$, (3) there are d rooted trees come to T each with probability $1/d$. We outline the proofs of the three statements as follows. To prove (1), we consider two rooted trees T_1, T_2 with roots r_1, r_2 respectively. Observe that any walk started from r_1 that leads to r_2 will transit T_1 to a tree $T_1^{'}$ with root r_2. Now consider the depth first search walk starting from r_2 that visits all the leaves of T_2 and returns back to r_2. This walk will transit $T_1^{'}$ to T_2. To prove (2), we consider a tree T with root r and show T will go to d rooted trees each with probability $1/d$. Since the graph is d-regular, there are d neighbors of r. Then the set of directed trees reachable from T in one step are as follows: For each neighbor v of r, we construct a tree rooted at v by adding the directed arc (r, v) to T and removing the unique arc that is leaving v. The proof of (3) is similar to (2), we would obtain the set of trees that transit to T in one step as follows: For each neighbor v of r, we can construct a tree rooted at v by adding arc (r, v) to T and removing the arc point to r in the unique path from v to r.

□

2.3.3 Spectral Sparsification

In this subsection, we will show how to construct a spectral approximator by sampling the edges with respect to their effective resistance.

Before we define spectral approximator, we first introduce and motivate the concept of a cut approximator. Given a weighted undirected graph $G = (V, E, \{w_e\})$, for a subset of vertices $S \subseteq V$, let $\delta_G(S)$ be the set of edges with one endpoint in S and another endpoint in $V - S$, and let $w(\delta_G(S))$ be the total weight of the edges in $\delta_G(S)$. A cut approximator of a graph G is a graph that approximates every cut in G.

Definition 2.3.6 (Cut Approximator)**.** *We say a graph H is a $(1 \pm \varepsilon)$-cut approximator of G if*

$$(1-\varepsilon)w(\delta_G(S)) \leq w(\delta_H(S)) \leq (1+\varepsilon)w(\delta(S)) \text{ for all } S \subseteq V \ .$$

Note that H has the same set of vertices and the edges may have different weights. A cut approximator with a small number of edges would be useful in designing fast approximation algorithms for graph connectivity problems such as the minimum cut and the maximum flow problem [8]. The exact algorithms for these problems usually have the running time depends on the number of edges m which can be up to $O(n^2)$. By constructing a sparse cut approximator, we could replace m by $n \log n$ in the running time essentially. Benczúr and Karger [8] showed how to construct a cut sparsifier with $O(n \log n/\varepsilon^2)$ edges efficiently by non-uniform sampling of the edges.

Spectral approximator is a generalization of cut approximator.

Definition 2.3.7 (Spectral Approximator)**.** *A graph H is a $(1\pm\varepsilon)$-spectral approximator of G if $(1-\varepsilon)L_G \preceq L_H \preceq (1+\varepsilon)L_G$, where L_G is the Laplacian matrix of G. Or equivalently, $(1-\varepsilon)x^T L_G x \leq x^T L_H x \leq (1+\varepsilon)x^T L_G x$ for all $x \in \mathbb{R}^n$, where n is the number of vertices of G.*

Fact 2.3.8. *If H is a $(1 \pm \varepsilon)$-spectral approximator of G, then H is a $(1 \pm \varepsilon)$-cut approximator of G.*

Proof. Let $S \subseteq V$ and $\chi_S \in \mathbb{R}^n$ be the characteristic vector such that $\chi_S(i) = 1$ if $i \in S$ and zero otherwise. Since H is a $(1 \pm \varepsilon)$-spectral approximator of G, we have $(1-\varepsilon)\chi_S^T L_G \chi_S \leq \chi_S^T L_G \chi_S \leq (1+\varepsilon)\chi_S^T L_G \chi_S$. Note that $x^T L_G x = \sum_{ij \in E} w(ij)(x_i - x_j)^2$, and thus $\chi_S^T L_G \chi_S = \sum_{ij \in E} w(ij)(\chi_S(i) - \chi_S(j)) = \sum_{ij \in \delta_G(S)} w(ij) = w(\delta_G(S))$. Therefore, the spectral approximation implies that $(1-\varepsilon)w(\delta_G(S)) \leq w(\delta_H(S)) \leq (1+\varepsilon)w(\delta(S))$ for all $S \subseteq V$. □

The following seminal result by Spielman and Srivastava [64] shows the existence of a sparse spectral approximator for any graph.

Theorem 2.3.9. *For any graph G and $\varepsilon > 0$, there is a $(1 \pm \varepsilon)$-spectral approximator H with $O(\frac{n \log n}{\varepsilon^2})$ edges.*

We assume without loss of generality that G is unweighted. We write $L_G = \sum_{ij \in E} L_{ij}$ where $L_{ij} = (x_i - x_j)(x_i - x_j)^T$ is the Laplacian matrix of the edge (i, j). So we can view L_G as a sum of m matrices.

The main idea of constructing the spectral approximator is to pick a subset of edges with probability proportional to the effective resistance and then reweight them.

We have the following algorithm:

The resulting graph H would have at most k edges. We want to show if $k = O(n \log n/\varepsilon^2)$, then H would be a $(1 \pm \varepsilon)$-spectral approximator of G.

First, we show that the algorithm will set the weights to have the correct expected value, i.e. $\mathbb{E}[L_H] = L_G$. Then we would like to show if k is large enough, then H will "concentrate" around its expectation.

Algorithm 2 Spectral Approximator

1: Let $p \propto \text{Reff}(i,j)$ for $(i,j) \in E$ be a probability distribution over the edges.
2: Let $H = (V, E, \{w_e\})$ and initialize all the new edge weights w_e to zero.
3: **loop** $k = O(n \log n/\varepsilon^2)$ times
4: Pick a random edge e from the probability distribution p.
5: Set $w_e \leftarrow w_e + \dfrac{1}{kp_e}$.
6: **end loop**
7: **return** H

Lemma 2.3.10. $\mathbb{E}[L_H] = L_G$.

Proof. Let e_i be the i-th edge we picked in the algorithm and $Z_i = \frac{1}{kp_{e_i}} L_{e_i}$ be the Laplacian matrix of the edge. Note that

$$\mathbb{E}[Z_i] = \sum_{e \in E} \frac{1}{kp_e} L_e \cdot \Pr(e \text{ is picked}) = \sum_{e \in E} \frac{1}{kp_e} L_e \cdot p_e = \sum_{e \in E} \frac{L_e}{k} = \frac{1}{k} L_G.$$

Then

$$\mathbb{E}[L_H] = \mathbb{E}[\sum_{i=1}^{k} Z_i] = \sum_{i=1}^{k} \mathbb{E}[Z_i] = \sum_{i=1}^{k} \frac{L_G}{k} = L_G.$$

□

To prove concentration, we use the following result by Ahlswede and Winter [3].

Theorem 2.3.11 (Matrix Concentration)**.** *Let Z be a random $n \times n$ real symmetric PSD matrix. Suppose $Z \preceq R \cdot \mathbb{E}[Z]$ for some $R \geq 1$. Let $Z_1, Z_2 \ldots Z_k$ be independent copies of Z. Then for any $\varepsilon \in (0,1)$, we have*

$$\Pr\left[(1-\varepsilon) \preceq \frac{1}{k} \sum_{i=1}^{k} Z_i \preceq (1+\varepsilon)\mathbb{E}[Z]\right] \geq 1 - 2n \exp\left(\frac{-\varepsilon^2 k}{4R}\right).$$

In our setting, we have $\mathbb{E}[Z] = \frac{1}{k} L_G$ and $\sum_{i=1}^{k} Z_i = H$. It remains to upper bound k such that $Z_i = \frac{1}{kp_e} L_e \preceq \frac{R}{k} L_G$ for a small R, i.e., $L_{ij} \preceq \text{Reff}(i,j) \cdot R \cdot L_G$ for some small R. First, we show how $\text{Reff}(i,j)$ relates L_{ij} and L_G.

Lemma 2.3.12. $L_{ij} \preceq \text{Reff}(i,j) \cdot L_G$

Proof. We would like to show that $x^T L_{ij} x \leq \text{Reff}(i,j) x^T L_G x$ for all $x \in \mathbb{R}^n$. Note that $\text{nullspace}(L_G) \subseteq \text{nullspace}(L_{ij})$, so for $x \in \text{nullspace}(L_{ij})$ the inequality holds trivially. For $x \perp \text{nullspace}(L_{ij})$, it follows that $x \perp \text{nullspace}(L_G)$. Thus such an x can be written as $L_G^{\dagger/2} y$ for some y, where $L_G^{\dagger}$ is the pseudo inverse of L_G.

Therefore,

$$\frac{x^T L_{ij} x}{x^T L_G x} = \frac{y^T L_G^{\dagger/2} L_{ij} L_G^{\dagger/2} y}{y^T y} \leq \lambda_{\max}(L_G^{\dagger/2} L_{ij} L_G^{\dagger/2}).$$

To bound $\lambda_{\max}(L_G^{\dagger/2} L_{ij} L_G^{\dagger/2})$, notice that since both $L_G^{\dagger/2}$ and L_{ij} are positive semidefinite, we have $L_G^{\dagger/2} L_{ij} L_G^{\dagger/2} \succeq 0$, and thus $\lambda_{\max}(L_G^{\dagger/2} L_{ij} L_G^{\dagger/2}) \leq \text{tr}((L_G^{\dagger/2} L_{ij} L_G^{\dagger/2}))$ as the trace of a matrix equals the sum of eigenvalues and all eigenvalues are non-negative. Note that by the cyclic property of trace,

$$\text{tr}(L_G^{\dagger/2} L_{ij} L_G^{\dagger/2}) = \text{tr}(L_e L_G^{\dagger}) = \text{tr}((\chi_i - \chi_j)(\chi_i - \chi_j)^T L_G^{\dagger}) = (\chi_i - \chi_j)^T L_G^{\dagger} (\chi_i - \chi_j) = \text{Reff}(i,j).$$

□

Recall from Lemma 2.3.5 that $\sum_{ij \in E} \text{Reff}(i,j) = n - 1$. Combining with Lemma 2.3.12, we can set $p_e = \frac{\text{Reff}(e)}{\sum_{e \in E} \text{Reff}(e)} = \frac{\text{Reff}(e)}{n-1}$ and we have $L_e \preceq p_e (n-1) L_G$ and thus $R = n - 1$.

Using the above matrix concentration result, the failure probability that L_H is not an ε-approximator is at most $2n \exp(-\varepsilon^2 k / 4R) = 2n \exp(-\varepsilon^2 k / 4(n-1))$. Finally, by setting $k = O(n \log n / \varepsilon^2)$, the failure probability is an inverse polynomial in n.

Remark: Direct implementation of the algorithm would takes $\tilde{O}(m^2)$ time by using a near-linear time Laplacian solver. To speed up the computation, we can use a general

technique called dimension reduction to get a good approximation of $\text{Reff}(i, j)$ quickly, which would improve the running time to $\tilde{O}(m)$ [64].

2.4 Network Design

In this section, we will discuss some previous work of network design problems under different types of combinatorial connectivity requirements.

2.4.1 Edge Connectivity and Iterative Rounding

In this subsection, we present the classical results for the survivable network design problem. Given a weighted undirected graph and a connectivity requirement r_{uv} for each pair of vertices u, v, the goal is to find a minimum cost subgraph such that there are at least r_{uv} edge-disjoint paths between u and v for all u, v. This problem is very well-studied and captures many interesting special cases [38, 2, 39, 34]. The best approximation algorithm for this problem is due to Jain [41], who introduced the technique of iterative rounding to design a 2-approximation algorithm.

In the following, we will present the 2-approximation algorithm given by Jain [41] for this problem and highlight the key arguments of the proof.

Overview of the algorithm:
The algorithm first solves the linear relaxation of the survivable network design problem, and then iteratively turn the fractional solution into an integral solution. The extreme point solutions of the linear program are characterized by a laminar family of tight constraints. Using this fact, Jain showed that there is at least one edge with fractional value at least $1/2$ in any extreme point solution. By repeatedly picking such edges, the total cost of the integral solution is at most twice the cost of the LP solution.

Linear Programming Relaxation:

The problem can be formulated as a linear program by representing the connectivity requirements as a weakly supermodular function. A function $f : 2^V \to \mathbb{Z}$ is called weakly supermodular if at least one of the following two conditions holds for any two subsets $S, T \subseteq V$.

$$f(S) + f(T) \leq f(S \cup T) + f(S \cap T)$$
$$f(S) + f(T) \leq f(S - T) + f(T - S)$$

It can be checked that the function f defined by $f(S) := \max_{u \in S, v \notin S}\{r_{uv}\}$ for each subset $S \subseteq V$ is a weakly supermodular function. Thus we can write the following linear programming relaxation for the survivable network design problem, denoted by $\mathrm{LP}_{\mathrm{SNDP}}$, with the function f being weakly supermodular.

$$\begin{array}{lll} \text{minimize} & \sum_{e \in E} c_e x_e & \\ \text{subject to} & x(\delta(S)) \geq f(S) & \forall S \subseteq V \\ & 0 \leq x_e \leq 1 & \forall e \in E \end{array}$$

Although it is not know whether there is a polynomial separation oracle for a general weakly supermodular function f, this linear program can be solved in polynomial time by using a maximum flow algorithm as a separation oracle.

Iterative Algorithm:

The following is Jain's iterative rounding algorithm. We write $d_F(S)$ as the number of edges in F with exactly one endpoint in S.

Algorithm 3 Iterative algorithm for survivable network design problem

Input: $G = (V, E, \{c_e\}), \{r_{uv}\}$
Initialization: $F \leftarrow \emptyset, f(S) \leftarrow \max_{u \in S, v \notin S}\{r_{uv}\}$
while $f \not\equiv 0$ **do**
    Find an optimal extreme point solution x to LP_{SNDP} with cut requirement f
    Remove every edge e with $x_e = 0$ from the graph.
    For every edge e with $x_e \geq 1/2$, add e to F and delete e.
    For every $S \subseteq V$, update $f(S) \leftarrow \max\{f(S) - d_F(S), 0\}$. (Update residual problem)
end while
Output: $H = (V, F)$

Jain proved the following important theorem that leads to the 2-approximation guarantee.

Theorem 2.4.1. *If f is a weakly supermodular and x is an extreme point solution to LP_{SNDP}, then there exists an edge e with $x_e \geq 1/2$.*

The proof is based on a token counting argument and the characterization of the extreme point solution to LP_{SNDP}. The iterative algorithm will terminate assuming Theorem 2.4.1. By an inductive argument, the returned solution is a 2-approximation solution.

2.4.2 Edge Connectivity on Directed Graphs

Melkonian and Tardos [55] generalized Jain's framework to directed graphs and proved that iterated rounding achieves an approximation ratio of 4 and conjecture the true approximation ratio is 2. Gabow [33] improved the ratio by showing the rounding scheme gives a factor 3 approximation.

2.4.3 Element Connectivity

In the setting of element-connectivity, the set of vertices is partitioned into terminals and nonterminals. The edges and nonterminals of the graph are called elements. The terminal vertices can be regarded as reliable nodes in the network. Using the technique of iterative rounding, Fleischer, Jain and Williamson [29] and Cheriyan, Vempala and Vetta [17] showed that the 2-approximation result of edge connectivity requirement extends to the problem when we require r_{ij} element-disjoint paths for all pairs of terminals i, j.

2.4.4 Vertex Connectivity

A natural variant is to require that there are $r_{u,v}$ internally vertex disjoint paths for every pair of vertices u, v. It turns out this problem is much harder to approximate. Kortsarz, Krauthgamer and Lee [46] showed that unless NP $\subseteq$ DTIME($n^{\text{poly}\log n}$), there is no polynomial time $2^{\log^{1-\varepsilon} n}$-approximation algorithm for this problem. Chuzhoy and Khanna [20] showed that when the maximum connectivity requirement k is small, there is a $O(k^3 \log n)$-approximation algorithm.

Global vertex connectivity is also well studied. Fakcharoenphol and Laekhanukit [28] showed that there is a $O(\log^2 k)$-approximation algorithm to obtain a minimum cost k-vertex-connected subgraph. Cheriyan and Végh [16] gives a 6-approximation algorithm if we further assume the number of vertices n is at least $k^3(k-1)+k$ using the iterative rounding approach.

2.4.5 Bounded Pairwise Distance

Another natural problem is to have a path length requirement between every pair of vertices. Given a directed or undirected graph G with edge cost $\{c_e\}$ and edge length $\{l_e\}$,

we want to find a minimum cost subgraph such that there is a path of length d_{uv} between every pair of vertices u, v. Dodis and Khanna [23] gave both hardness and algorithmic results on this problem. When every edge has the same cost and the same length, they showed that it is $\Omega(\log n)$-hard to approximate and gave an $O(\log \log d)$-approximation algorithm where $d = \max_{u,v} d(u, v)$. When either the edges costs vary or the edges lengths vary, they showed the problem is $\Omega(2^{\log^{1-\varepsilon} n})$-hard to approximate.

2.4.6 Edge Connectivity with Degree Bounds

It is also interesting to study when there are multiple types of combinatorial constraints. When we incorporates degree upper bounds b_v on each vertex v to the edge connectivity constraints, Lau, Naor, Salavatipour, and Singh [49] showed that there is a $(2, 2b_v + 3)$-approximation algorithm, where the cost of the returned solution is at most twice the cost of an optimal solution that satisfies the degree bounds, and the degree of each vertex v is at most $2b_v + 3$. Lau and Zhou [50] further improved the results to a $(2, \min\{b_v + 3r_{\max}, 2b_v + 2\})$-approximation.

We have seen that there are positive results over different types of combinatorial constraints. The ultimate goal of our work is to incorporate effective resistance constraints into network design problems.

2.5 Spectral Requirements

In this section, we will present some previous work on network design problems with spectral requirements.

2.5.1 Mixing Time

Boyd, Diaconis, and Xiao [10] considered the problem of assigning probabilities to the edges in a Markov chain to minimize its mixing time. The mixing time of a Markov chain is the number of steps required for the walk to converge to the stationary distribution. It is known that the mixing time is governed by the second largest eigenvalue λ_2 of the transition matrix P. They showed that this problem could be formulated as a convex optimization problem, which can be expressed as the following semidefinite program (SDP):

$$\begin{array}{ll} \text{minimize} & s \\ \text{subject to} & -sI \preceq P - (1/n)\vec{1}\vec{1}^T \preceq sI, \\ & P \geq 0, \quad P\vec{1} = \vec{1}, \quad P = P^T, \\ & P_{ij} = 0 \quad \forall (i,j) \notin E, \end{array}$$

where $\left\| P - (1/n)\vec{1}\vec{1}^T \right\|_2$ is the spectral expression of λ_2. The above program can be transformed to a standard semidefinite problem, which is solvable in polynomial time. Other than formulating the problem as an SDP, they also describe a subgradient method to solve the program when the graph is large.

2.5.2 Algebraic Connectivity

Ghosh and Boyd [36] studied the problem of adding edges to a graph to maximize the second smallest eigenvalue of the graph Laplacian, which governs the algebraic connectivity of a graph. Given a graph $G = (V, E_{\text{base}} \cup E_{\text{cand}})$, the problem is to add k edges from E_{cand} to the existing edges E_{base} to maximize the algebraic connectivity. Let $x \in [0,1]^{|E_{\text{cand}}|}$ be the relaxed solution and $L(x) = L_{\text{base}} + \sum_{e \in E_{\text{cand}}} x_e b_e b_e^T$ be the Laplacian matrix of the resulting graph corresponds to x. They formulated the relaxation of the problem as follows:

$$\begin{aligned} \text{minimize} \quad & \lambda_2(L(x)) \\ \text{subject to} \quad & \vec{1}^T x = k \\ & x \in [0,1]^{|E_{\text{cand}}|} \end{aligned}$$

From the characterization that $\lambda_2(L(x)) = \inf\{y^T L(x) y \mid \|y\| = 1, \vec{1}^T y = 0\}$, the above program can be furthered formulated as a semidefinite program. The solution to the SDP serves as an upper bound of the optimal value. They also gave a heuristic algorithm to turn the fractional solution of the SDP to an integral solution but with no performance guarantee provided.

2.5.3 Total Effective Resistance

Ghosh, Boyd and Saberi [37] first studied connectivity requirements related to effective resistances. They studied the problem of allocating conductance to the edges of a given graph under a fixed total conductance budget to minimize the total effective resistance, i.e., the sum of the resistances between all pairs of vertices. Using the fact that the total effective resistance is the trace of the pseudo-inverse of the Laplacian matrix (see Theorem 2.2.5), they formulated the problem as the following semidefinite program:

$$\begin{aligned} \text{minimize} \quad & \operatorname{tr}(Y) \\ \text{subject to} \quad & 1^T x = 1, \qquad x \geq 0 \\ & \begin{bmatrix} L_x & I \\ I & Y \end{bmatrix} \succeq 0 \end{aligned}$$

where $x \in \mathbb{R}^{|E|}$ are the variables of the edge weights, $L_x = \sum_{e \in E} x_e b_e b_e^T$ is the Laplacian matrix corresponding to the conductance assignment x and $Y \in \mathbb{R}^{n \times n}$ is the slack symmetric matrix. Note that the last constraint in the program is equivalent to $Y \succeq L_x^\dagger$. Since the objective function is convex, they further showed that this SDP could be solved

numerically using a custom interior-point algorithm. These earlier works only proposed convex programming relaxations and heuristic algorithms, but did not show the hardness of the problem or give any algorithm with a provable approximation guarantee.

2.5.4 Experimental Design

Approximation guarantees are only obtained in two recent papers ([59], [6]). Both papers studied the more general problem of experimental design, which is a classical problem in statistics. We will first introduce the problem of experimental design, and then explain the connections to network design problems.

In the simplest setting, we want to estimate a hidden vector $w \in \mathbb{R}^d$ via linear measurements of the form $y_i = v_i^T w + \eta_i$, where $v_i \in \mathbb{R}^d$ are possible experiments and η_i is a small independent and identically distributed unbiased Gaussian error in the measurement of the experiment. Given a subset S of experiments, we can estimate $\hat{w}$ of w by least squares approximation. The error vector $w - \hat{w}$ has a Gaussian distribution with mean 0 and covariance matrix $\Sigma := \left(\sum_{i \in S} v_i v_i^T\right)$. In the experimental design problem, our goal is to pick a set of vectors S with $|S| = k$ out of the m vectors such that the measurement error is minimized. There are different objective functions f, or known as optimal criterion, to quantify the "size" of the covariance matrix. Popular choices of f include:

- A(verage) $f_A(\Sigma) = \text{tr}(\Sigma^{-1})/d$;
- D(eterminant) $f_D(\Sigma) = \det(\Sigma^{-1})$;
- T(race) $f_T(\Sigma) = d/\text{tr}(\Sigma^{-1})$;
- E(igen) $f_E(\Sigma) = \|\Sigma^{-1}\|_2$.

We will show that spectral network design problems are special cases of experimental design. To see this, we can interpret the set of possible experiments as the edges of the

graph, where each edge e corresponds to a vector b_e. Then, the covariance matrix becomes the Laplacian matrix $L = \sum_{e \in S} b_e b_e^T$. Different optimal criterions of the covariance matrix would correspond to different objective functions on the Laplacian matrix.

- By Theorem 2.2.5, the trace of the pseudo-inverse Laplacian is the total effective resistance of the graph, hence the A-optimal criterion corresponds to the total effective resistance.
- The E-optimal criterion corresponds to the spectral gap because $\left\|L^\dagger\right\|_2 = \lambda_{\max}(L^\dagger) = 1/\lambda_2(L)$.
- In our problem, the s-t effective resistance can be expressed as a function of the Laplacian $f_{\text{Reff}}(L) = b_{st}^T L^\dagger b_{st}$, but it does not correspond to the standard criterion in experimental design.

The two recent papers obtained different approximations in different optimal criterion. Nikolov, Singh, and Tantipongpipat [59] obtained an $(1+\varepsilon)$-approximation for the A-optimality when $k = \Omega(d/\varepsilon)$. This result corresponds to the case when every edge has the same cost, and there is a $(1+\varepsilon)$-approximation algorithm for minimizing the total effective resistance when the budget is at least $\Omega(n/\varepsilon)$ where n is the number of vertices in the graph. The main idea of their algorithm is by proportional volume sampling, which involves picking a set of columns S of size k with probability proportional to $\mu(S)$ times $\det(\Sigma)$ for some measure μ.

Allen-Zhu, Li, Singh, and Wang [6] obtained approximation algorithms for a wider class of optimal criterion (including A, D, E-optimality). They showed that if the optimal criterion f is convex, monotone and reciprocal multiplicative ($f(t\Sigma) = t^{-1} f(\Sigma)$ for all $t > 0$), then they can achieve $(1+\varepsilon)$-approximation with $\Omega(d/\varepsilon^2)$ experiments.

The main idea of their work is as follows. By the convexity of the objective function, they showed that the convex relaxation of the problem could be solved in polynomial time. Then

the key point is to round the fraction solution π to an integral solution $\hat{s}$. By applying a linear transformation on the vectors v_i, they reduced the problem to finding a set of k vectors S to maximize the minimum eigenvalue of $\sum_{v \in S} vv^T$. Their main technical result is a one-sided lower bound on the minimum eigenvalue:

Theorem 2.5.1. *Suppose $\varepsilon \in (0, 1/3]$ and $k \geq n/\varepsilon^2$. Let $\pi \in \mathcal{C}_k = \{\pi \in [0,1]^m : \sum_{i=1}^m \pi_i \leq k\}$ that satisfies $\sum_{i=1}^m \pi_i v_i v_i^T = I_{n \times n}$. Then we can find $\hat{s} \in \mathcal{S}_k = \{s \in \{0,1\}^m : \sum_{i=1}^m s_i \leq k\}$ such that*

$$\lambda_{\min}\left(\sum_{i=1}^m \hat{s}_i v_i v_i^T\right) \geq (1 - 3\varepsilon) \cdot I.$$

The key step of proving the theorem is to further reduce the problem to bounding the regret of a particular Follow-The-Regularized-Leader algorithm which admits closed-form solutions. The close form solution gives rise to a "swapping" algorithm to lower bounding the minimum eigenvalue. The algorithm starts with an arbitrary initial solution set $S_0 \subseteq [m]$ of cardinality k. In each iteration $t \geq 0$, it selects a pair of indices $i_t \in S_t$ and $j_t \notin S_t$ and makes a "swap" by updating $S_{t+1} = S_t \cup \{j_t\} \backslash \{i_t\}$. Indices i_t, j_t are chosen to minimize a potential function $\Phi(S_t) := \sum_i \frac{1}{\lambda_i - l}$, where l is intended to be an lower bound of the minimum eigenvalues, and $\lambda_1, \ldots \lambda_d$ are the eigenvalues of $\sum_{v \in S_t} vv^T$. To avoid blowing up the potential function, the lower bound l would increase slowly over iterations. Hence, it gives a holistic measure of our current solution of how close it is to the identity matrix.

The one-sided lower bound results can be used to upper bound effective resistances. Theorem 2.5.1 implies that we can find a subgraph H with $L_H \succeq (1 - \varepsilon) L_\pi$, where L_π is the Laplacian matrix of the optimal fraction subgraph. Then, recalling Equation 2.2.2 that the effective resistance of two vertices s, t can be written as $b_{st} L^\dagger b_{st}$, and the fact that a lower bound of the Laplacian matrix is an upper bound for the pseudo-inverse, we have $\text{Reff}_H(s,t) = b_{st}^T L_H^\dagger b_{st} \leq (1-\varepsilon)^{-1} b_{st}^T L_\pi^\dagger b_{st} = (1 + 2\varepsilon)\text{Reff}_\pi(s,t)$.

Remark: In our problem, our objective function is the s-t effective resistance, and it satisfies convexity, monotonicity and reciprocal multiplicativity (see the proof of convexity in Theorem 2.2.8). Therefore, there is a $(1+\varepsilon)$-approximation when the budget is at least $\Omega(n/\varepsilon^2)$. The results from both papers require the budget to be much larger than the number of vertices. For our problem, the interesting regime is when k is much smaller than n, where the techniques in [6, 59] do not apply.

Chapter 3

Network Design for Minimizing s-t Effective Resistance

The results in Chapter 3.2 and Chapter 3.3 are based on the joint work with Lap Chi Lau, Aaron Schild, Sam Chiu-wai Wong and Hong Zhou [13].

3.1 Introduction

The following is the formal formulation of our problem.

Definition 3.1.1 (The s-t effective resistance network design problem)**.** *The input is an undirected graph $G = (V, E)$ where each edge e has a non-negative cost c_e and a non-negative resistance r_e, two specified vertices $s, t \in V$, and a cost budget k. The goal is to find a subgraph H of G that minimizes $\mathrm{Reff}_H(s, t)$ subject to the constraint that the total edge cost of H is at most k, where $\mathrm{Reff}_H(s, t)$ denotes the effective resistance between s and t in the subgraph H with resistances r_e on the edges.*

In Chapter 1, we introduce s-t effective resistance as a connectivity measure and observe that it is the interpolation between s-t shortest path distance and s-t edge connectivity. In the following sections, we try to answer the following questions:

- How hard is the s-t effective resistance network design problem?
- In what special types of input can we solve the problem exactly or with arbitrarily small error?
- Is there any fast heuristic algorithm for general input to obtain a "good" approximate solution?

The following are our main contributions in this thesis.

Unlike shortest path distance or edge connectivity, we show that the problem is NP-hard (see Chapter 3.2), even when all the edges (resistors) have the same resistance and the same cost.

Theorem 3.1.2. *The s-t effective resistance network design problem is NP-hard, even when every edge has the same cost and the same resistance ($c_e = r_e = 1$ for every edge e).*

Since our problem is related to electrical network design, it is natural to consider the special case when the input graph is a series-parallel graph. In Chapter 3.3, we use dynamic programming to design a fully polynomial time approximation scheme for the problem when the ratio between the maximum and minimum resistance is bounded, and an exact algorithm when every edge has the same cost.

Theorem 3.1.3. *There is a dynamic programming based $(1+\varepsilon)$-approximation algorithm for the s-t effective resistance network design problem when the input graph is a series-parallel graph. The running time of the algorithm is $O(|E|^7 U^2/\varepsilon^2)$ where $U =$*

$\max_e r_e / \min_e r_e$ *is the ratio between the maximum and minimum resistance. If we assume further that* $c_e = 1$ *for all edges* e*, there is an exact algorithm for the problem with running time* $O(|E|k^2)$.

For general graphs, we propose a greedy algorithm that we add a path at each iteration. We suggest a framework to analyze our algorithm and conjecture that the algorithm would achieve constant approximation (see Chapter 3.4).

Conjecture 3.1.4. *There is a greedy based* 3.95*-approximation algorithm for the* s*-*t *effective resistance network design problem when every edge has the same cost and the same resistance. The running time of the algorithm is* $O(m^2\sqrt{\log n})$.

3.2 NP-completeness for Unit Cost Unit Resistance

In this section, we prove that the problem is NP-hard even if every edge has the same cost and the same resistance.

We will prove Theorem 3.1.2 in this subsection. The following is the decision version of the problem.

Problem 3.2.1 (s-t effective resistance network design with unit-cost unit-resistance)**.**

> **Input:** *An undirected graph* $G = (V, E)$ *where each edge* $e \in E$ *has resistance one, two vertices* $s, t \in V$*, and two parameters* k *and* R.
>
> **Question:** *Does there exist a subgraph* H *of* G *with at most* k *edges and* $\text{Reff}_H(s, t) \leq R$*?*

We will show that this problem is NP-complete by a reduction from the 3-Dimensional Matching (3DM) problem.

Problem 3.2.2 (3-Dimensional Matching)**.**

Input: *Three disjoint sets of elements* $X = \{x_1, \ldots, x_q\}, Y = \{y_1, \ldots, y_q\}$ *and* $Z = \{z_1, \ldots, z_q\}$*, a set of triples* $\mathcal{T} \subseteq X \times Y \times Z$ *where each triple contains exactly one element in* X, Y *and* Z*.*

Question: *Does there exist a subset of* q *pairwise disjoint triples in* $\mathcal{T}$*?*

Reduction: Given an instance of 3DM with $\{(X, Y, Z), \mathcal{T}\}$, we let $\tau = |\mathcal{T}|$ and denote the triples by $\mathcal{T} = \{T_1, \ldots, T_\tau\}$.

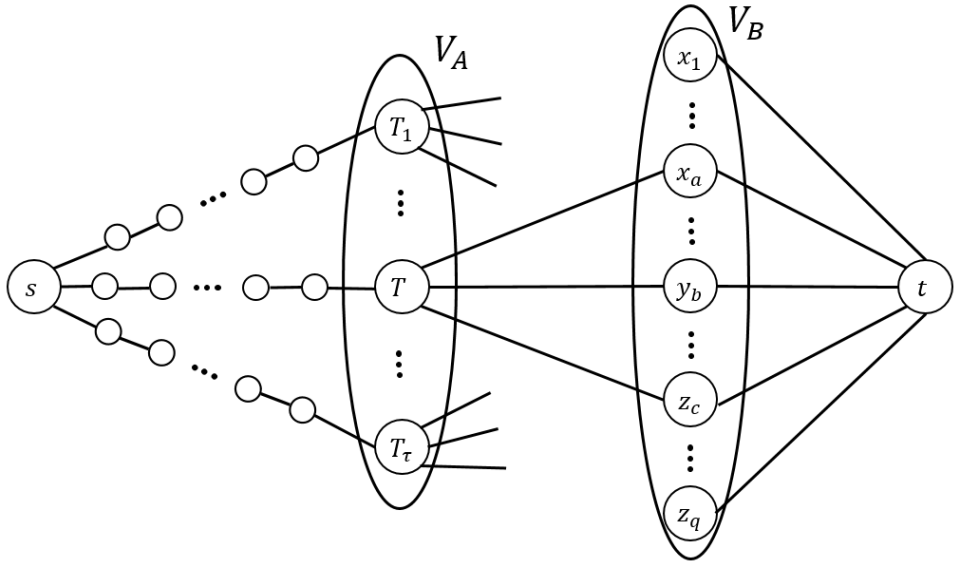


Figure 2: An illustration of constructing the graph G from a 3DM instance.

We construct a graph $G = (V, E)$ as follows.

Vertex Set: The vertex set V of the graph G is the disjoint union of five sets $\{s\}, \{t\}, V_A$, V_B, and D. Each vertex in V_A corresponds to a triple in $\mathcal{T}$, that is $V_A = \{T_1, \ldots, T_\tau\}$. Each vertex in V_B corresponds to an element in $X \cup Y \cup Z$,

that is $V_B = \{x_1, \ldots, x_q, y_1, \ldots, y_q, z_1, \ldots, z_q\}$. Let $l = 3\tau + 3q$. The set D consists of $\tau \cdot l$ "dummy" vertices $\{d_{i,j} \mid 1 \leq i \leq \tau, 1 \leq j \leq l\}$. So, there are totally $\tau + 3q + 2 + \tau(3\tau + 3q)$ vertices in G, which is a polynomial in the input size of the 3DM instance.

Edge Set: The edge set E of the graph G is the disjoint union of three edge sets F_1, F_2 and P. There are 3τ edges in F_1, where there are three edges (T, x_a), (T, y_b) and (T, z_c) for each triple $T = (x_a, y_b, z_c) \in \mathcal{T}$. There are $3q$ edges in F_2, where there is an edge from each vertex in V_B to t. There are $\tau(l+1)$ edges in P, where there is a path $P_i := (s, d_{i,1}, d_{i,2}, \ldots, d_{i,l}, T_i)$ for each triple $T_i \in \mathcal{T}$ for $1 \leq i \leq \tau$. So, there are totally $3\tau + 3q + \tau(3\tau + 3q + 1)$ edges in E, which is a polynomial in the input size of the 3DM instance.

The proof of the following claim completes the proof of Theorem 3.1.2.

Lemma 3.2.3. *Let $k = q(l+1) + 3\tau + 3q$ and $R = (3(l+1)+2)/3q$. The 3DM instance has q disjoint triples if and only if the graph G has a subgraph H with at most k edges and* $\mathrm{Reff}_H(s,t) \leq R$.

Proof. One direction is easy. If there are q disjoint triples in the 3DM instance, say $\{T_1, \ldots, T_q\}$, then H will consist of the q paths $P_1, \ldots, P_q$, the $3q$ edges in F_1 incident on $T_1, \ldots, T_q$, and all the $3q$ edges in F_2. There are $(l+1)q + 3q + 3q \leq k$ edges in H, and $\mathrm{Reff}_H(s,t) = (l+1)/q + 1/3q + 1/3q = (3(l+1)+2)/3q = R$, as in the graph in Figure 3.

The other direction is more interesting. If there do not exist q disjoint triples in the 3DM instance, then we need to argue that $\mathrm{Reff}_H(s,t) > R$ for any H with at most k edges. First, note that $k < (q+1)(l+1)$, and so the budget is not enough for us to buy more than q paths. As it is useless to buy only a proper subset of a path, we can thus assume that H consists of q paths and all the edges in F_1 and all the edges in F_2, a total of exactly $q(l+1) + 3\tau + 3q = k$ edges. For any such H, we will argue that $\mathrm{Reff}_H(s,t) > R$.

Without loss of generality, we assume that H consists of $P_1, \ldots, P_q$ and all edges in F_1 and F_2. As $T_1, \ldots, T_q$ are not disjoint, there are some vertices in V_B that are not neighbors of $T_1 \cup \ldots \cup T_q$, call those vertices U.

We consider the following modifications of H to obtain H', and use $\text{Reff}_{H'}(s,t)$ to lower bound $\text{Reff}_H(s,t)$. For every pair of vertices in V_B, we add an edge of zero resistance. For each edge incident on $T_{q+1}, \ldots, T_\tau$, we decrease its resistance to zero. By the monotonicity principle, the modifications will not increase the s-t effective resistance, as we either add edges with zero resistance or decreasing the resistance of existing edges. The modifications are equivalent to contracting the vertices with zero resistance edges in between, and so H' is equivalent to the graph in Figure 3. Therefore, we have $\text{Reff}_H(s,t) \geq \text{Reff}_{H'}(s,t) \geq R$.

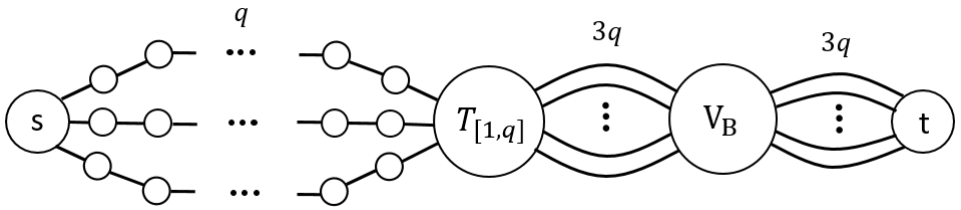


Figure 3: The subgraph H when the 3DM instance has q disjoint triples.

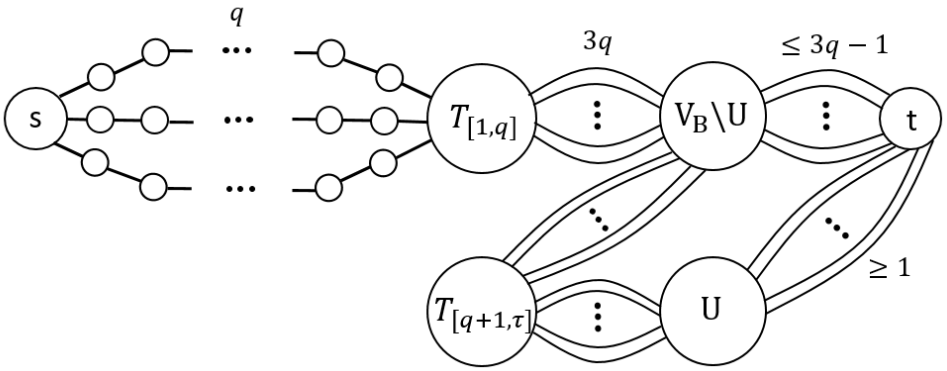


Figure 4: The subgraph H when U is non-empty.

We will prove that one of the inequalities in $\text{Reff}_H(s,t) \geq \text{Reff}_{H'}(s,t) \geq R$ must be strict when $U \neq \emptyset$ (Figure 4). To argue the strict inequality, we look at the unit s-t electrical flow f in H and consider two cases.

- If there exists some vertex $u \in U$ with no incoming electrical flow, then we can delete such a vertex without changing $\text{Reff}_H(s,t)$. But then in the modified graph H', the number of parallel edges to t is strictly smaller than $3q$, and therefore $\text{Reff}_{H'}(s,t) > R$.
- If there exists some vertex $u \in U$ with some incoming electrical flow, then $f(T_j u) > 0$ for some $j \geq q+1$. Since we have decrease the resistance of such an edge $T_j u$ to zero, the energy of f in H' is strictly smaller than the energy of f in H. By Thomson's principle, we have $\text{Reff}_{H'}(s,t) \leq \mathcal{E}_{H'}(f) < \mathcal{E}_H(f) = \text{Reff}_H(s,t)$.

Since the 3DM instance has no q disjoint triples, it follows that $U \neq \emptyset$ and thus one of the above two cases must apply. In either case, we have $\text{Reff}_H(s,t) > R$ and this completes the proof of the other direction. □

3.3 Dynamic Programming Algorithms for Series-Parallel Graphs

In this section, we will present the dynamic programming algorithms for solving the s-t effective resistance network design problem on series-parallel graphs. We first review the definitions of series-parallel graphs in Subsection 3.3.1. Then, we present the exact algorithm in Theorem 3.1.3 when every edge has the same cost in Subsection 3.3.2, and the fully polynomial time approximation scheme in Theorem 3.1.3 in Subsection 3.3.3.

3.3.1 Series-Parallel Graphs

Definition 3.3.1 (two-terminal series-parallel graph)**.** *A two-terminal series-parallel graph (SP graph) is a graph with two distinguished vertices (the source vertex s and the target vertex t) that can be constructed recursively as follows:*

- *Base case: A single edge (s, t)*
- *Compose step: If G_1 and G_2 are two series parallel graphs with source s_i and target t_i ($i = 1, 2$), then we can combine them in two ways:*
  - *Series-composition: We identify t_1 with s_2 as the same vertex, the source of the new graph is s_1 and the target is t_2.*
  - *Parallel-composition: We identify s_1 with s_2 as the same vertex and t_1 with t_2 as the same vertex, the new source is $s_1 = s_2$ and the new target is $t_1 = t_2$.*

Given the sequence of steps of constructing a series-parallel graph G, we can define a tree T (SP-tree) as follows.

Definition 3.3.2 (SP-tree)**.**

- *Leaf node: If G is a single edge, then T is a single node containing the edge.*
- *Recurse step: G is either a series-composition (S) or a parallel-composition (P) of G_1 and G_2, then T is a S-node (P-node) containing G, and its children are roots of the SP-trees of G_1 and G_2.*

For a tree node v in a SP-tree T, let G_v be the subgraph that v represents, s_v, t_v be the two terminals of G_v, and v_l, v_r to be its left and right child if v is an internal node. Note that the SP-tree is a fully binary tree with $2m - 1$ nodes.

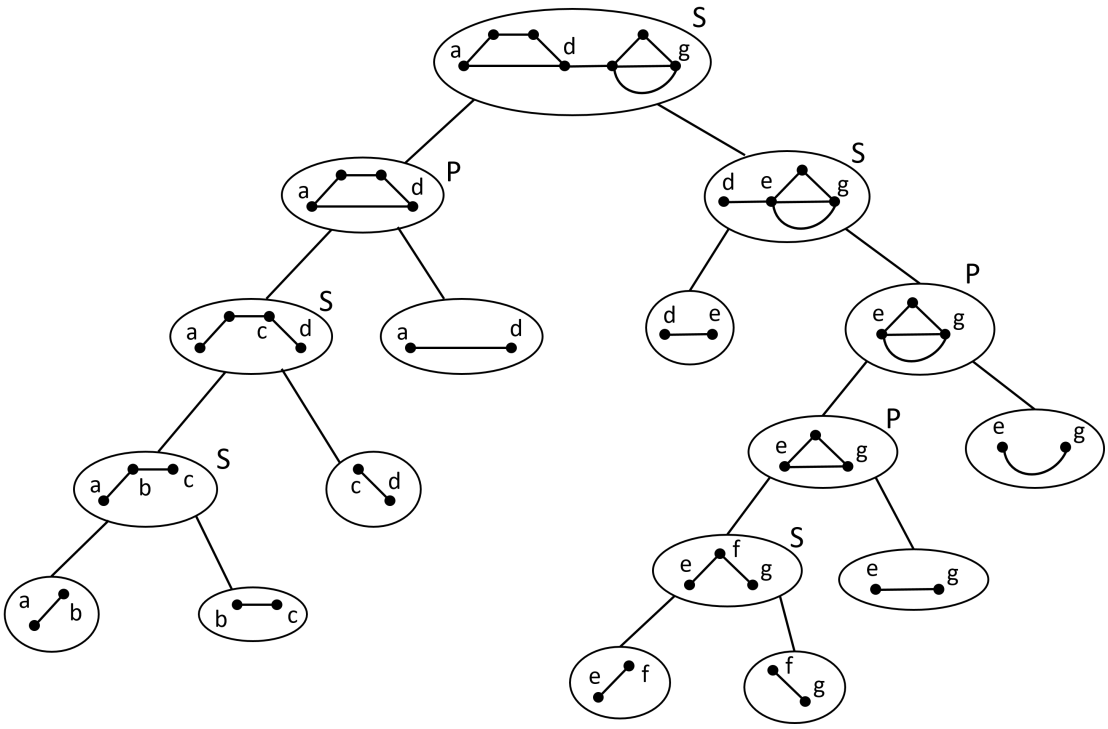


Figure 5: An example of a SP-tree.

Given a two-terminal SP graph, the corresponding SP-tree can be computed in $O(n+m)$ time. The linear time SP-graph recognition algorithm in [68] will give us the construction sequence of G, and we can build the SP-tree in a bottom-up manner.

3.3.2 Polynomial Time Algorithm for the Unit Cost Case

The following fact shows that the s-t effective resistance can be computed easily from the SP-tree.

Fact 3.3.3 (Resistance of series-parallel network)**.** *Let G be a two-terminal SP graph and each edge e has a non-negative resistance r_e. Let T be the corresponding SP-tree. For every tree node v, we can compute the source-target effective resistance as follows.*

Leaf node: $\mathrm{Reff}_{G_v}(s_v, t_v) = r_e$ *if* v *is a leaf node with a single edge* e.

S-node: $\mathrm{Reff}_{G_v}(s_v, t_v) = \mathrm{Reff}_{G_{v_l}}(s_{v_l}, t_{v_l}) + \mathrm{Reff}_{G_{v_r}}(s_{v_r}, t_{v_r})$.

P-node: $\mathrm{Reff}_{G_v}(s_v, t_v) = \dfrac{\mathrm{Reff}_{G_{v_l}}(s_{v_l}, t_{v_l}) \cdot \mathrm{Reff}_{G_{v_r}}(s_{v_r}, t_{v_r})}{\mathrm{Reff}_{G_{v_l}}(s_{v_l}, t_{v_l}) + \mathrm{Reff}_{G_{v_r}}(s_{v_r}, t_{v_r})}$.

We can design the dynamic programming algorithm by defining the subproblems using the SP-tree T.

For every tree node v and $b = 0, 1 \ldots k$, we define the subproblem

$$R(v, b) := \min_{H \subseteq G_v} \left\{ \mathrm{Reff}_H(s_v, t_v) \mid \sum_{e \in H} c_e \leq b \right\}.$$

Since we assume that every edge e has cost $c_e = 1$, there are at most $2mk$ subproblems, as the SP-tree has at most $2m$ nodes and there are at most k possibilities for the cost of a subgraph.

It follows from the definition that $R(v_{\text{root}}, k)$ would be the optimal s-t effective resistance for our problem. To compute $R(v, b)$, with Fact 3.3.3, we can use the following recurrence which exhausts all possible distributions of the budget among the two children:

$$R(v, b) = \begin{cases} \infty & \text{if } v \text{ is a leaf node and } b < c_e \\ r_e & \text{if } v \text{ is a leaf node and } b \geq c_e \\ \min\limits_{b'=0 \ldots b} R(v_l, b') + R(v_r, b - b') & \text{if } v \text{ is a S-node} \\ \min\limits_{b'=0 \ldots b} \dfrac{R(v_l, b') \cdot R(v_r, b - b')}{R(v_l, b') + R(v_r, b - b')} & \text{if } v \text{ is a P-node.} \end{cases}$$

As there are $O(mk)$ subproblems and each subproblem can be computed in $O(k)$ time, the time complexity of this dynamic programming algorithm is $O(mk^2)$.

3.3.3 Fully Polynomial Time Approximation Scheme

In this subsection, we use dynamic programming to design a fully polynomial time approximation scheme to prove Theorem 3.1.3. In the previous subsection, we assume that every edge has the same cost to obtain an exact algorithm, by having a bounded number of subproblems in dynamic programming. When the cost could be arbitrary, the number of subproblems can no longer be bounded by a polynomial. Since the cost constraint must be satisfied, we do not change the cost of the edges, but instead we discretize the resistance of the edges and optimize over the cost, and show that it gives an arbitrarily good approximation when the discretization is fine enough.

Rescaling:

First, by rescaling, we assume that $\min_e r_e = 1$ and $\max_e r_e = U$ in G. Let $m = |E|$ and $L = \varepsilon/m^2$ where $\varepsilon > 0$ is the parameter in the approximation guarantee. We further rescale the resistance by setting $r_e \leftarrow r_e/L$. This rescaling ensures that for any subgraph of G in which s-t is connected, the s-t effective resistance is upper bounded by Um/L (when all the edges are in series) and is lower bounded by $1/(mL)$ (when all the edges are in parallel).

Subproblems and Recurrence:

Let T be the SP-tree of G and let v_{root} be the root of T. We define two similar sets of subproblems. For every tree node v and a value $R \in [1/(mL), Um/L]$, we define the subproblem

$$C(v, R) := \min_{H \subseteq G_v} \left\{ \sum_{e \in H} c_e \mid \text{Reff}_H(s_v, t_v) \leq R \right\}.$$

Similar to the reasoning in the previous subsection, the subproblems satisfy the following recurrence relation:

$$C(v,R) = \begin{cases} c_e & \text{if } v \text{ is a leaf node with a single edge } e \text{ and } R \geq r_e \\ \infty & \text{if } v \text{ is a leaf node with a single edge } e \text{ and } R < r_e \\ \min\limits_{R_1,R_2 \in [1/(mL),Um/L]} \{C(v_l,R_1) + C(v_r,R_2) \mid R_1 + R_2 \leq R\} & \text{if } v \text{ is a S-node} \\ \min\limits_{R_1,R_2 \in [1/(mL),Um/L]} \left\{C(v_l,R_1) + C(v_r,R_2) \mid \dfrac{R_1 R_2}{R_1+R_2} \leq R\right\} & \text{if } v \text{ is a P-node.} \end{cases}$$

Discretized subproblems:

We cannot use dynamic programming to solve the above recurrence relation efficiently as there are unbounded number of subproblems. Instead, we use dynamic programming to compute the solution of all the "discretized" subproblems using the same recurrence relation. For every integer R from $\lceil 1/(mL) \rceil$ to $\lceil Um/L \rceil$, we define

$$\overline{C}(v,R) := \begin{cases} c_e & \text{if } v \text{ is a leaf node with a single edge } e \text{ and } R \geq \lceil r_e \rceil \\ \infty & \text{if } v \text{ is a leaf node with a single edge } e \text{ and } R < \lceil r_e \rceil \\ \min\limits_{R_1,R_2 \in \{\lceil 1/(mL) \rceil \ldots \lceil Um/L \rceil\}} \{\overline{C}(v_l,R_1) + \overline{C}(v_r,R_2) \mid R_1 + R_2 \leq R\} & \text{if } v \text{ is a S-node} \\ \min\limits_{R_1,R_2 \in \{\lceil 1/(mL) \rceil \ldots \lceil Um/L \rceil\}} \left\{\overline{C}(v_l,R_1) + \overline{C}(v_r,R_2) \mid \left\lceil \dfrac{R_1 R_2}{R_1+R_2} \right\rceil \leq R\right\} & \text{if } v \text{ is a P-node.} \end{cases}$$

We can think of $\overline{C}(v,R)$ as the minimum cost required to select a subset of edges such that the effective resistance between s_v and t_v is at most R, when the effective resistance is rounded up to an integer during each step of the computation in the recurrence relation.

Algorithm and Complexity:

After computing all $\overline{C}(v,R)$, the algorithm will return

$$\min\{R \mid \overline{C}(v_{\text{root}},R) \leq k\}$$

as the approximate minimum s-t effective resistance. Given a tree node v, by trying all possible integral values of R_1 and R_2, we can compute the values of $\overline{C}(v,R)$ for each

possible R in $O((Um/L)^2)$ time. Therefore, the total running time of computing all $\overline{C}(v,R)$ is $O(m)\cdot O((Um/L)^2) = O(m^7U^2/\varepsilon^2)$. To output the optimal edge set, we can store the optimal values of R_1, R_2 for each pair of (v,R) to reconstruct the edge set.

Correctness and Approximation Guarantee:
Since we have not changed the edge cost, the solution returned by the algorithm will have total cost at most k. It remains to show that the s-t effective resistance is at most $(1+\varepsilon)$ times the optimal s-t effective resistance. For every tree node v and every $b \in [0,k]$, we define

$$R(v,b) := \min\{R \mid C(v,R) \le b, R \in [1/(mL), Um/L]\}$$
$$\overline{R}(v,b) := \min\{R \mid \overline{C}(v,R) \le b, R \in \{\lceil 1/(mL)\rceil, \dots \lceil Um/L\rceil\}\}.$$

First we show $\overline{R}(v,b)$ has similar recurrence as $R(v,b)$.

Claim 3.3.4. *For every non-leaf node v of T and $b \in [0,k]$, we have*

$$\overline{R}(v,b) := \begin{cases} \min\limits_{b_1,b_2 \mid b_1+b_2=b} \{\overline{R}(v_l,b_1) + \overline{R}(v_r,b_2)\} & \text{if } v \text{ is a S-node} \\ \min\limits_{b_1,b_2 \mid b_1+b_2=b} \left\{ \left\lceil \dfrac{1}{1/\overline{R}(v_l,b_1) + 1/\overline{R}(v_r,b_2)} \right\rceil \right\} & \text{if } v \text{ is a P-node.} \end{cases}$$

Proof. If v is a S-node, then from the definition of $\overline{R}(v,b)$, we have

$$\begin{aligned} \overline{R}(v,b) &= \min\{R \mid \overline{C}(v,R) \le b, R \in \{\lceil 1/(mL)\rceil, \dots \lceil Um/L\rceil\}\} \\ &= \min\{R_1 + R_2 \mid \overline{C}(v_l,R_1) + \overline{C}(v_r,R_2) \le b, R_1, R_2 \in \{\lceil 1/(mL)\rceil, \dots \lceil Um/L\rceil\}\}. \end{aligned}$$

Let $b_1 := \overline{C}(v_l,R_1)$ and $b_2 := \overline{C}(v_r,R_2)$, then the pair of R_1, R_2 that achieves the minimum would satisfy $\overline{R}(v_l,b_1) = R_1$ and $\overline{R}(v_r,b_2) = R_2$. Thus we have

$$\overline{R}(v,b) = \min\{\overline{R}(v_l,b_1) + \overline{R}(v_r,b_2) \mid b_1 + b_2 = b\}.$$

Similarly, we can show the recurrence of $\overline{R}(v,b)$ when v is a P-node. □

It follows from the definitions that the optimal s-t effective resistance is $R(v_{\text{root}}, k)$, and the output of our algorithm will be $\overline{R}(v_{\text{root}}, k)$. The following lemma establishes the approximation guarantee.

Lemma 3.3.5. *For every tree node v and for every $b \in [0, k]$, it holds that*

$$\overline{R}(v, b) \leq \left(1 + \frac{\varepsilon |E(G_v)|}{m}\right) R(v, b).$$

Proof. We prove the lemma by induction on the tree node of the SP-tree.

Base Case: Suppose v is a leaf node of T and G_v is a graph of a single edge e.

- For $b < c_e$, we have $\overline{R}(v, b) = R(v, b) = \infty$.
- For $b \geq c_e$, we have $R(v, b) = r_e$ and

  $$\overline{R}(v, b) = \lceil r_e \rceil \leq r_e + 1 = r_e + \left(\frac{\varepsilon}{m}\right)\left(\frac{1}{mL}\right) \leq r_e + \frac{\varepsilon}{m} r_e = \left(1 + \frac{\varepsilon |E(G_v)|}{m}\right) R(v, b),$$

  where the second inequality uses that every resistance is at least $1/(mL)$, and the last equality uses that $|E(G_v)| = 1$ and $r_e = R(v, b)$.

S-node: Suppose v is a S-node. For every $b \in [0, k]$, we have

$$\begin{aligned}
\overline{R}(v, b) &= \min_{b_1, b_2 | b_1 + b_2 = b} \{\overline{R}(v_l, b_1) + \overline{R}(v_r, b_2)\} \\
&\leq \min_{b_1, b_2 | b_1 + b_2 = b} \left\{ \left(1 + \frac{\varepsilon |E(G_{v_l})|}{m}\right) R(v_l, b_1) + \left(1 + \frac{\varepsilon |E(G_{v_r})|}{m}\right) R(v_r, b_2) \right\} \\
&\leq \min_{b_1, b_2 | b_1 + b_2 = b} \left\{ \left(1 + \frac{\varepsilon |E(G_v)|}{m}\right) (R(v_l, b_1) + R(v_r, b_2)) \right\} \\
&= \left(1 + \frac{\varepsilon |E(G_v)|}{m}\right) \min_{b_1, b_2 | b_1 + b_2 = b} \{(R(v_l, b_1) + R(v_r, b_2))\} \\
&= \left(1 + \frac{\varepsilon |E(G_v)|}{m}\right) R(v, b),
\end{aligned}$$

where the first equality follows from Claim 3.3.4, the first inequality follows from the induction hypothesis, and the second inequality follows from the fact that $\max(|E(G_{v_l})|, |E(G_{v_r})|)$ is at most $|E(G_v)| - 1$.

P-node: Suppose v is a P-node. For every $b \in [0, B]$, we have

$$\begin{aligned}
\overline{R}(v,b) &= \min_{b_1,b_2|b_1+b_2=b} \left\{ \left\lceil \frac{1}{1/\overline{R}(v_l,b_1) + 1/\overline{R}(v_r,b_2)} \right\rceil \right\} \\
&\leq \min_{b_1,b_2|b_1+b_2=b} \left\{ \left\lceil \left(1 + \frac{\varepsilon(|E(G_v)|-1)}{m}\right) \frac{1}{1/R(v_l,b_1) + 1/R(v_r,b_2)} \right\rceil \right\} \\
&= \left\lceil \left(1 + \frac{\varepsilon(|E(G_v)|-1)}{m}\right) R(v,b) \right\rceil \\
&\leq \left(1 + \frac{\varepsilon(|E(G_v)|-1)}{m}\right) R(v,b) + 1 \\
&= \left(1 + \frac{\varepsilon(|E(G_v)|-1)}{m}\right) R(v,b) + \frac{\varepsilon}{m}\frac{1}{mL} \\
&\leq \left(1 + \frac{\varepsilon(|E(G_v)|-1)}{m}\right) R(v,b) + \frac{\varepsilon}{m} R(v,b) \\
&= \left(1 + \frac{\varepsilon|E(G_v)|}{m}\right) R(v,b),
\end{aligned}$$

where the first equality follows from Claim 3.3.4, the first inequality follows from the induction hypothesis and the fact that $\max(|E(G_{v_l})|, |E(G_{v_r})|) \leq |E(G_v)| - 1$, and the last inequality holds as the minimum resistance of any subgraph is at least $1/(mL)$.

Therefore, the lemma follows by an induction on the SP-tree. □

By substituting $v = v_{\text{root}}$ and $b = k$, we have

$$\overline{R}(v_{\text{root}}, k) \leq \left(1 + \frac{m\varepsilon}{m}\right) R(v_{\text{root}}, k) = (1+\varepsilon) R(v_{\text{root}}, k),$$

and this completes the proof of Theorem 3.1.3.

3.4 Greedy Approach

In [13], we obtain a constant factor approximation algorithm for our problem when every edge has the same cost and the same resistance. Our technique is based on rounding the fractional solution to the convex relaxation problem.

In this section, we design a greedy combinatorial algorithm to our problem when every edge has the same cost and the same resistance. We provide some intuition and suggest a framework to analyze our algorithm. We conjecture that our algorithm is a 3.95-approximation algorithm to the problem.

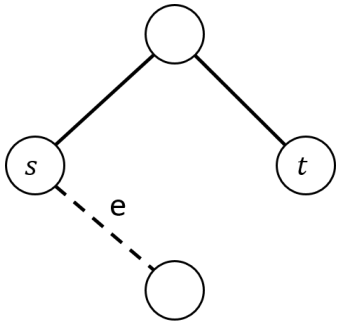
3.4.1 Observations and Intuition

For the greedy algorithm, it is more convenient to consider the s-t effective conductance (Chapter 2.2.2). If we treat the s-t effective conductance as a set functions on the set of edges, then the problem becomes finding the best subset $F \subseteq E$ where $|F| \leq k$ that maximizes the s-t effective conductance. In the following, we denote the s-t effective resistance on the edge set F by $\text{Reff}(F)$ and the effective conductance by $\text{Ceff}(F) := 1/\text{Reff}(F)$.

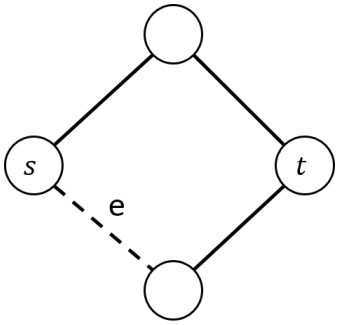
When the set function is submodular, monotone and non-negative, there is a famous result by Nemhauser, Wolsey and Fisher in 1978 [58] stating that if we pick an element x which maximizes the marginal benefit $f(S \cup \{x\}) - f(S)$ each time, we will obtain a set S that achieves a $(1 - 1/e)$-approximation of the optimum.

Definition 3.4.1. *A set function $f : 2^X \to \mathbb{R}$ is called **submodular** if for all subsets $S \subset T \subset X$ and all $x \in X \backslash T$,*

$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T).$$



(a) The edge set F_1 with a single edge e



(b) The edge set F_2 with a single edge e

Figure 6: A counterexample showing that s-t effective conductance is not submodular.

In other words, the marginal return of adding x is always diminishing as the set grows.

Although the s-t effective conductance is non-negative and monotonic, it is not a submodular function.

Fact 3.4.2. *The s-t effective conductance is not submodular.*

Proof. In Figure 6, we have $F_1 \subset F_2$. Note that

$$\text{Ceff}(F_1 \cup \{e\}) - \text{Ceff}(F_1) = \frac{1}{\text{Reff}(F_1 \cup \{e\})} - \frac{1}{\text{Reff}(F_1)} = \frac{1}{2} - \frac{1}{2} = 0$$

$$\text{Ceff}(F_2 \cup \{e\}) - \text{Ceff}(F_2) = \frac{1}{\text{Reff}(F_2 \cup \{e\})} - \frac{1}{\text{Reff}(F_2)} = \frac{1}{1} - \frac{1}{2} = \frac{1}{2}.$$

So $\text{Ceff}(F_1 \cup \{e\}) - \text{Ceff}(F_1) < \text{Ceff}(F_2 \cup \{e\}) - \text{Ceff}(F_2)$ and hence Ceff is not submodular.

□

From the example in the above proof, we could see that adding a single edge might not help to increase the s-t effective conductance if the edge does not connect two vertices that are previously connected. This suggests that a good greedy algorithm should consider adding edges that introduce new connection of the graph at each iteration. If we can show the marginal increase of the effective conductance of adding a path is diminishing, then we might be able to follow the idea in [58] to design a good approximation algorithm.

3.4.2 Greedy Algorithm

The main idea of our greedy strategy is to keeping adding a path that maximizes the ratio between the marginal increase of conductance and its length.

Initially, the set of used edges U is empty, and our algorithm will partition the unused edges into paths and return them as a list P in some greedy order. Finally, to output the actual selection of edges, we keep selecting the paths in P in order until we run out of budget.

The key part of the algorithm is how we partition the edges into paths. We keep track of a set of vertices V^*, which contains s, t and the vertices connected by the used edges. At each iteration, for any pair of vertices in V^*, we compute a shortest path that uses the minimum number of edges between the two vertices, where the path can only use the unused edges and only pass through vertices that are not connected before. Then, we pick a path that maximizes the ratio between the marginal increase in s-t effective conductance of the resulting graph and its length. We mark this path as used and repeat the iteration until all edges are used.

The following is the pseudo-code for the path partitioning algorithm.

Algorithm 4 Greedy Path Partitioning Algorithm

Input $\{G = (V, E), s, t, \{r_e\}, \{c_e\}, U\}$ ▷ U is the set of selected edges
$V^* \leftarrow \{s, t\} \cup \{\text{vertices appear in } U\}$ ▷ set of connected vertices
$E^* \leftarrow U$ ▷ set of used edges
$P \leftarrow$ empty list ▷ list of paths in greedy order
while $E^* \neq E$ **do**
    $A \leftarrow \emptyset$ ▷ set of possible paths in this iteration
    for $u, v \in V^*$ **do**
        $p_{uv} \leftarrow$ any (u, v) shortest path on the graph $G' = (V \backslash V^* \cup \{u, v\}, E \backslash E^*)$
        $A \leftarrow A \cup p_{uv}$
    end for
    $p^* = \arg\max_{p \in A} \frac{\text{Ceff}(E^* \cup \{p\}) - \text{Ceff}(E^*)}{c(p)}$ ▷ $c(p)$ is the length of path p
    update the set of connected vertices: $V^* \leftarrow V^* \cup V(p)$
    update the set of used edges: $E^* \leftarrow E^* \cup E(p)$
    Add p^* to P
end while
Output P

Let $P = \{p_1, p_2, \ldots, p_\tau\}$ be the list of paths returned by the greedy algorithm on input $\{G = (V, E), s, t, \{r_e\}, \{c_e\}, U = \emptyset\}$ where we start from an empty graph and τ is the total number of paths in P. Let $P_i := \cup_{j=1}^{i} p_j$ be the partial solution after the i-iteration of the algorithm. For any subset of edges F, let $c(F) := \sum_{e \in F} c_e$ be the total edge cost of F. Let i^* be the largest i such that $c(P_i) \leq k$, so our algorithm will return the set of edges P_{i^*}.

Given any subset of edges $E^* \subseteq E$, we can compute the s-t effective resistance (conductance) on E^* in $\tilde{O}(m\sqrt{\log n})$ time using Laplacian solvers [21]. Since we can compute the shortest path between two points in $O(n + m)$ time and there are at most m iterations, the time complexity of the algorithm is $\tilde{O}(m^2\sqrt{\log n})$.

3.4.3 Analyzing the Greedy Algorithm

We have the following conjecture on the performance guarantee on our algorithm.

Conjecture 3.4.3. *Algorithm 4 is a greedy-based* 3.95*-approximation algorithm for the s-t effective resistance network design problem when* $c_e = 1$ *and* $r_e = 1$ *for every edge* e*.*

Similar to the proof of the greedy algorithm on minimizing supermodular function, we need to assume the objective function $\text{Ceff}(F)$ has a certain "submodular" property. We conjecture the following about the diminishing return property of the greedy algorithm.

Conjecture 3.4.4 (Approximate Diminishing Return Property)**.** *Let* P_i *be a partial greedy solution for some* $i \in [0 \dots \tau]$ *and* F *be any subset of the edges such that* $F \cap P_i = \emptyset$*, let* $\{f_1, f_2, \dots\}$ *where* $F = \bigcup_i f_i$ *be the list of paths output by the greedy algorithm on input* $\{G = (V, F \cup P_i), s, t, \{r_e\}, \{c_e\}, U = P_i\}$*. (We pre-select the edges in* P_i*.) Then we have*

$$\frac{\text{Ceff}(P_i \cup \{f_1\}) - \text{Ceff}(P_i)}{c(f_1)} \geq \frac{1}{2} \cdot \frac{\text{Ceff}(P_i \cup F) - \text{Ceff}(P_i)}{c(F)}.$$

Let the **effective conductance increase ratio** of a set of edges X to a set of existing edges Y be the effective conductance increased after adding X to Y divided by the cost of adding X, i.e.

$$\nabla_Y(X) := \frac{\text{Ceff}(Y \cup X) - \text{Ceff}(Y)}{c(X)}.$$

This conjecture states that if we start from a partial greedy solution P_i and the next path we are going to add is f_1, then the effective conductance increase ratio of f_1 is always greater than one half the effective conductance increase ratio any set of edges that contains f_1. This suggests that after adding f_1, adding subsequent paths would not improve the effective conductance increase ratio by more than a factor of 2, and hence we say the algorithm has the **approximate diminishing return property**. Indeed, the constant $1/2$ in the conjecture is the best possible ratio we can get, by the example in Figure 7 in

Subsection 3.4.4. Now, we first prove the approximation guarantee assuming Conjecture 3.4.4. More discussion on Conjecture 3.4.4 will be in Subsection 3.4.4.

Let d^* be the shortest distance between s and t. Without loss of generality, we can assume the budget k is not less than d^*. We suggest a framework to analyze our algorithm as follows:

1. We show that when $k \leq c_1 \cdot d^*$ for some constant $c_1 \geq 1$, returning any shortest s-t path at the first iteration of the algorithm is already a c_1-approximation algorithm for our problem. See Lemma 3.4.6.

2. Otherwise, if our algorithm used a constant fraction of the budget, i.e. $c(P_{i^*}) \geq c_2 \cdot k$ for some constant $0 < c_2 \leq 1$, we show that the algorithm would achieve $(1 - e^{-c_2/2})^{-1}$-approximation assuming the diminishing return property of the algorithm. See Lemma 3.4.7.

3. In the final case where $c(P_{i^*}) < c_2 \cdot k$, we argue that the remaining edges of the graph is not that "useful" and we show that the algorithm would achieve $\left(1 + \frac{2}{(1-c_2)^2 c_1}\right)$-approximation assuming the diminishing return property. See Lemma 3.4.8.

Combining the three lemmas above, we can conclude that Algorithm 4 is a 3.95-approximation algorithm.

Theorem 3.4.5. *Assuming Conjecture 3.4.4, Algorithm 4 is a greedy-based* 3.95*-approximation algorithm for the s-t effective resistance network design problem when $c_e = 1$ and $r_e = 1$ for every edge e.*

Proof. From Lemma 3.4.6, Lemma 3.4.7 and Lemma 3.4.8, it remains to choose $c_1 \geq 1$ and $0 < c_2 \leq 1$ to minimize

$$\max\left\{c_1, (1 - e^{-c_2/2})^{-1}, \left(1 + \frac{2}{(1-c_2)^2 c_1}\right)\right\}.$$

Notice that the first term increases with c_1, the second term decreases with c_2 and the third term decreases with c_1 but increases with c_2, so the minimum is attained when the three terms are equal. By solving this system of equations, we can get $c_1 \approx 3.95$ and $c_2 \approx 0.585$. Substituting the values, we can see that all three terms are at most 3.95, which completes the proof. □

In the following, we will state and prove the three lemmas.

Lemma 3.4.6. *When the budget k is at most $c_1 \cdot d^*$ for some $c_1 \geq 1$, returning any shortest s-t path would be a c_1-approximation of the s-t effective resistance.*

Proof. First, observe that $i^* \geq 1$ since $k \geq d^*$, thus P_{i^*} would contain a s-t shortest path and $\text{Reff}(P_{i^*}) \leq d^*$. So it is enough to show that $\text{Reff}(F) \geq (d^*)^2/k$ for any subset of edges F with $c(F) \leq k$.

For any $F \subseteq E$ with $c(F) \leq k$, let $G_F = (V, F)$ be the corresponding graph of F. Now we construct another graph G'_F on vertex set $\{v_0, \dots v_{d^*}\}$ by identifying some vertices on G_F:

- For all $j \in 1, \dots d^*$, we identify those vertices with distance j to s as vertex v_j.
- For all vertices with distance to s greater than d^*, we identify them as vertex v_{d^*}.

Since identifying a pair of vertices is equivalent to adding an edge with zero resistance between them, so by Rayleigh's monotonicity principle (Theorem 2.2.7), we have $\text{Reff}_{G'_F}(s,t) \leq \text{Reff}_{G_F}(s,t) = \text{Reff}(F)$. Note that G'_F connects s and t with a sequence of parallel edges. Let E_i be the set of edges connecting v_j and v_{j+1}. Using the Cauchy-Schwarz inequality and the fact that $\text{Reff}_{G'_F}(s,t) = \sum_{j=1}^{d^*} \frac{1}{\sum_{e \in E_j} 1/r_e} = \sum_{j=1}^{d^*} 1/|E_j|$ from the series rule and the parallel rule (Fact 3.3.3), we have

$$d^* = \sum_{j=1}^{d^*} \sqrt{|E_j|} \cdot \frac{1}{\sqrt{|E_j|}} \leq \sqrt{\sum_{j=1}^{d^*} |E_j|} \cdot \sqrt{\sum_{j=1}^{d^*} \frac{1}{|E_j|}} \leq \sqrt{k} \cdot \sqrt{\text{Reff}_{G'_F}(s,t)},$$

which completes the proof. □

Lemma 3.4.7. *Assuming Conjecture 3.4.4, if the output of our algorithm P_{i^*} consumes at least c_2 fraction of the budget, then it obtains a $\left(1-e^{-c_2/2}\right)^{-1}$ approximation of the problem.*

Proof. The proof follows the main idea in [58]. For any $i \in [0 \dots \tau]$ and for any $F \subseteq E$ with $c(F) \leq k$, if we can show that

$$\text{Ceff}(P_i) \geq \left(1 - e^{-\frac{c(P_i)}{2k}}\right) \text{Ceff}(F)$$

assuming Conjecture 3.4.4, then the lemma will follow after substituting $c(P_{i^*}) \geq c_2 \cdot k$ and the fact that effective resistance is reciprocal to effective conductance.

For any $i \in [1, \dots \tau]$ and any $F \subseteq E$ with $c(F) \leq k$ and $F \cap P = \emptyset$, let $\{f_1, f_2, \dots\}$ be the list of paths output by the greedy algorithm on input $\{G = (V, F \cup P_i), s, t, \{r_e\}, \{c_e\}, U = P_i\}$ where we pre-selected edges in the partial solution P_i. Then we have

$$\begin{aligned}
\text{Ceff}(F) &\leq \text{Ceff}(P_i \cup F) \\
&= \text{Ceff}(P_i) + (\text{Ceff}(P_i \cup F) - \text{Ceff}(P_i)) \\
&\leq \text{Ceff}(P_i) + 2c(F) \cdot \frac{\text{Ceff}(P_i \cup f_1) - \text{Ceff}(P_i)}{c(f_1)} \\
&\leq \text{Ceff}(P_i) + 2k \cdot \frac{\text{Ceff}(P_{i+1}) - \text{Ceff}(P_i)}{c(p_{i+1})},
\end{aligned}$$

where the first inequality follows from Rayleigh's monotonicity principle (Theorem 2.2.7), the second inequality follows from Conjecture 3.4.4, and the last inequality follows from the optimality of p_{i+1} in our greedy algorithm.

Multiply both sides by $c(p_{i+1})/k$, we have

$$\frac{c(p_{i+1})}{2k}\text{Ceff}(F) \leq \text{Ceff}(P_{i+1}) - \left(1 - \frac{c(p_{i+1})}{2k}\right) \text{Ceff}(P_i).$$

Adding $(1 - c(p_{i+1})/2k) \cdot \text{Ceff}(F)$ on both sides and rearrange, we have

$$\text{Ceff}(F) - \text{Ceff}(P_{i+1}) \leq \left(1 - \frac{c(p_{i+1})}{2k}\right)(\text{Ceff}(F) - \text{Ceff}(P_i)).$$

Let $a_i := \text{Ceff}(F) - \text{Ceff}(P_i)$. We can rewrite the above inequality as

$$a_{i+1} \leq \left(1 - \frac{c(p_{i+1})}{2k}\right) a_i.$$

By induction, we can relate a_{i+1} and a_0. Note that $a_0 = \text{Ceff}(F) - \text{Ceff}(P_0) = \text{Ceff}(F)$. Then by using the inequality $1 - x \leq e^{-x}$, we can conclude that

$$a_{i+1} \leq \prod_{j=0}^{i} \left(1 - \frac{c(p_{j+1})}{2k}\right) \text{Ceff}(F) \leq \Pi_{j=0}^{i} \left(e^{-\frac{c(p_{j+1})}{2k}}\right) \cdot \text{Ceff}(F) = e^{-\frac{c(P_{i+1})}{2k}} \cdot \text{Ceff}(F).$$

Substitute $a_i := \text{Ceff}(F) - \text{Ceff}(P_i)$ and rearrange, we complete the proof by having

$$\text{Ceff}(P_i) \geq \left(1 - e^{-\frac{c(P_i)}{2k}}\right) \text{Ceff}(F).$$

□

Lemma 3.4.8. *Assuming Conjecture 3.4.4, if the output of our algorithm P_{i^*} consumes less than c_2 fraction of the budget and $k \geq c_1 \cdot d^*$, then returning P_{i^*} would be a $\left(1 + \frac{2}{(1-c_2)^2 c_1}\right)$-approximation of the minimum s-t effective resistance.*

Proof. The intuition behind is that since $c(P_{i^*}) < c_2 \cdot k$ and $c(P_{i^*+1}) > k$, then if we let $l := c(p_{i^*+1})$, we know that $l \geq (1 - c_2)k$ which is a constant fraction of the budget. Hence p_{i^*+1} is a relatively long path compare to the shortest path as we have $k \geq c_1 \cdot d^*$. By the optimality of p_{i^*+1}, this suggests that the subsequent paths would only give small contribution to the s-t effective conductance and it is affordable to ignore them.

First, we would like to show that $\text{Ceff}(P_{i^*} \cup \{p_{i^*+1}\})$ is upper bounded by $\text{Ceff}(P_{i^*}) + 1/l$, the effective conductance after adding a s-t path of length l to the set of edges P_{i^*}, by lower bounding $\text{Reff}(P_{i^*} \cup \{p_{i^*+1}\})$.

Claim 3.4.9. $\text{Ceff}(P_{i^*} \cup p_{i^*+1}) \leq \text{Ceff}(P_{i^*}) + \frac{1}{l}$.

Proof. Let L be the Laplacian matrix of the edge set P_{i^*} with the rows and columns only correspond to the vertices in P_{i^*}. Recall the definition of effective resistance in Section 2.2.2, we can write

$$\text{Reff}(P_{i^*}) = b_{st}^T L^\dagger b_{st}.$$

Let p_{i^*+1} be a path connecting u and v. Now adding a path from u to v of length l is equivalent to modify the Laplacian matrix from L to $L+(1/l)\cdot b_{uv}b_{uv}^T$, since a path from u to v of length l can be regarded as a resistor connecting u, v with resistance l (or conductance $1/l$).

To express the s-t effective resistance after adding p_{i^*+1}, we will need the Sherman-Morrison formula (Theorem 2.1.8). By the Sherman-Morrison formula and Fact 2.1.7, we have

$$\text{Reff}(P_{i^*} \cup \{p_{i^*+1}\}) = b_{st}^T \left(L + \frac{1}{l} b_{uv} b_{uv}^T \right)^\dagger b_{st} = b_{st}^T L^\dagger b_{st} - \frac{(b_{st}^T L^\dagger b_{uv})^2}{l + b_{uv}^T L^\dagger b_{uv}}.$$

Note that $L^\dagger b_{uv}$ is a potential vector when one unit of current is sent from u to v, so we can interpret the term $b_{uv}^T L^\dagger b_{uv}$ as the potential difference between u and v. By Fact 2.2.3 that the potential difference is maximum between u, v when an electrical flow is sent from

u to v, we have

$$\begin{aligned}
\mathrm{Reff}(P_{i^*} \cup \{p_{i^*+1}\}) &= b_{st}^T L^\dagger b_{st} - \frac{(b_{st}^T L^\dagger b_{uv})^2}{l + b_{uv}^T L^\dagger b_{uv}} \\
&\geq b_{st}^T L^\dagger b_{st} - \frac{(b_{st}^T L^\dagger b_{uv})^2}{l + b_{st}^T L^\dagger b_{uv}} \\
&\geq b_{st}^T L^\dagger b_{st} - \frac{(b_{st}^T L^\dagger b_{st})^2}{l + b_{st}^T L^\dagger b_{st}} \\
&= \mathrm{Reff}(P_{i^*}) - \frac{\mathrm{Reff}(P_{i^*})^2}{l + \mathrm{Reff}(P_{i^*})} \\
&= \frac{1}{\mathrm{Ceff}(P_{i^*})}\left(1 - \frac{1}{l \cdot \mathrm{Ceff}(P_{i^*}) + 1}\right) \\
&= \frac{l}{l \cdot \mathrm{Ceff}(P_{i^*}) + 1} = \frac{1}{\mathrm{Ceff}(P_{i^*}) + 1/l},
\end{aligned}$$

where the second inequality also follows from Fact 2.2.3 and the fact that $\frac{x^2}{l+x}$ is increasing on x for $l > 0$. □

Now we are ready to upper bound the difference between $\mathrm{Ceff}(F)$ and $\mathrm{Ceff}(P_{i^*})$, for any $F \subseteq E$ with $c(F) \leq k$. We have

$$\begin{aligned}
\mathrm{Ceff}(F) - \mathrm{Ceff}(P_{i^*}) &= k \cdot \frac{\mathrm{Ceff}(F) - \mathrm{Ceff}(P_{i^*})}{k} \\
&\leq k \cdot \frac{\mathrm{Ceff}(P_{i^*} \cup F) - \mathrm{Ceff}(P_{i^*})}{k} \\
&\leq \frac{2k}{l}(\mathrm{Ceff}(P_{i^*} \cup \{p_{i^*+1}\}) - \mathrm{Ceff}(P_{i^*})) \\
&\leq \frac{2k}{l^2} && \text{(by Claim 3.4.9)} \\
&\leq \frac{2}{(1-c_2)^2 c_1 d^*} && \text{(by } l \geq (1-c_2)k \text{ and } k \geq c_1 d^*\text{)} \\
&\leq \frac{2\mathrm{Ceff}(P_{i^*})}{(1-c_2)^2 c_1},
\end{aligned}$$

where the first inequality is by Rayleigh's monotonicity (Theorem 2.2.7), the second inequality follows from Conjecture 3.4.4 and the optimality of p_{i^*+1} and the last inequality follows from the fact that $\mathrm{Reff}(P_{i^*}) \leq d^*$ since P_{i^*} contains the s-t shortest path.

After rearranging, we have $\text{Ceff}(F) \leq \text{Ceff}(P_{i^*}) \left(1 + \frac{2}{(1-c_2)^2 c_1}\right)$. It is equivalent to $\text{Reff}(P_{i^*}) \leq \text{Reff}(F) \left(1 + \frac{2}{(1-c_2)^2 c_1}\right)$, which completes the proof. □

3.4.4 Discussion

In this subsection, we discuss some examples and intuition of the approximate diminishing return property (Conjecture 3.4.4). We will also show why the greedy approach might not be able to generalize to the case where the resistances of the edges are arbitrary. In the end of this subsection, we will discuss the attempt in [66] of adding a set of edges to the graph to minimize total effective resistance by a greedy algorithm.

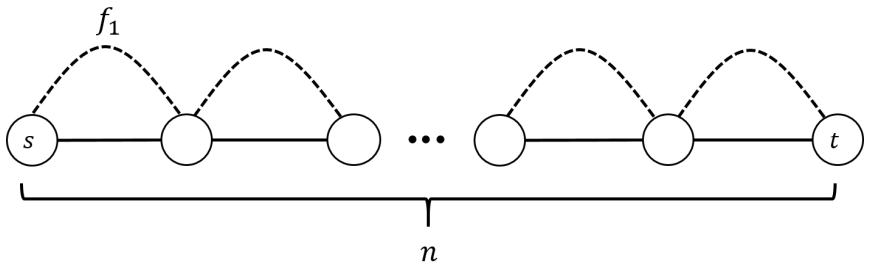


Figure 7: A tight example of Conjecture 3.4.4.

Tight example of Conjecture 3.4.4: In Figure 7, let P_1 be the s-t path consisting of solid edges and F be the set of dashed edges. Given the partial solution P_1, f_1 will be the next path that the greedy algorithm selects from the remaining edge set F. Then we have

$$\begin{aligned} \text{Ceff}(P_1) &= \frac{1}{n}, \\ \text{Ceff}(P_1 \cup \{f_1\}) &= \frac{1}{n - 1/2} = \frac{2}{2n-1}, \\ \text{Ceff}(P_1 \cup F) &= \frac{2}{n}. \end{aligned}$$

So we can compare the conductance increase ratio of adding f_1 and adding F respectively,

$$\nabla_{P_1}(f_1) = \frac{\text{Ceff}(P_1 \cup \{f_1\}) - \text{Ceff}(P_1)}{c(f_1)} = \frac{2}{2n-1} - \frac{1}{n} = \frac{1}{n(2n-1)},$$
$$\nabla_{P_1}(F) = \frac{\text{Ceff}(P_1 \cup F) - \text{Ceff}(P_1)}{c(F)} = \frac{2/n - 1/n}{n} = \frac{1}{n^2}.$$

Therefore, we would see that $\frac{\nabla_{P_1}(f_1)}{\nabla_{P_1}(F)} = \frac{n}{2n-1}$, which tends to $1/2$ when n tends to infinity. Although Conjecture 3.4.4 is tight in the above example, the greedy algorithm actually performs well - it will return the optimal set of edges for every possible value of the budget. Next, we show the worst example to our algorithm that we found so far.

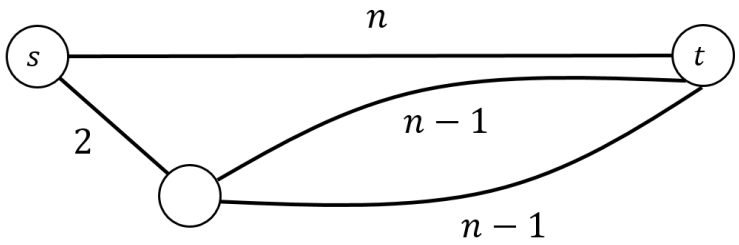


Figure 8: An example of getting a 2-approximation using greedy algorithm.

In Figure 8, the number x on an edge means this edge consists of x unit-cost-unit-resistance edges. When the budget $k = 2n$, the optimal s-t effective resistance is $2 + \frac{n-1}{2}$ by selecting all edges except the s-t shortest path of length n, but our greedy algorithm would only select the s-t shortest path of effective resistance n. Hence the ratio is $\frac{n}{2+(n-1)/2} = 2 - \frac{6}{n+3} \approx 2$ when n is large. In this example, the main reason for the large gap is that our greedy algorithm would waste almost half of the budget. Note that in the unit resistance setting, the "bad" examples we found so far to Conjecture 3.4.4 and the algorithm are both simple instances. We believe that there might not be examples that exceed the conjectural ratio.

Examples when the resistances have arbitrary values: Next, we demonstrate a few examples to show the greedy algorithm cannot generalize to the case where the edges have the same cost but have arbitrary resistance.

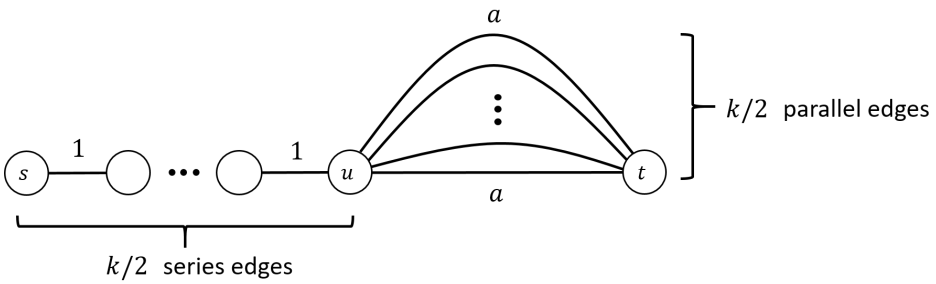


Figure 9: A counterexample to Conjecture 3.4.4 when the resistances have arbitrary values.

In Figure 9, the number next to an edge is the resistance of that edge. The partial solution P_0 is the empty set and F is the whole set of edges. We will show that Conjecture 3.4.4 does not hold in this example. The next path f_1 returned by the greedy algorithm consists of $k/2$ edges of resistances 1 and one edge of resistance a. Then we have

$$\begin{aligned}
\text{Ceff}(P_0) &= 0, \\
\text{Ceff}(P_0 \cup \{f_1\}) &= \frac{1}{k/2 + a} = \frac{2}{k + 2a}, \\
\text{Ceff}(P_0 \cup F) &= \frac{1}{k/2 + 2a/k} = \frac{2k}{k^2 + 4a}.
\end{aligned}$$

So we can compare the conductance increase ratio of adding f_1 and adding F respectively,

$$\begin{aligned}
\nabla_{P_0}(f_1) &= \frac{\text{Ceff}(P_0 \cup \{f_1\}) - \text{Ceff}(P_1)}{c(f_1)} = \frac{2}{(2a + k)(k/2 + 1)} = \frac{4}{(2a + k)(k + 2)}, \\
\nabla_{P_0}(F) &= \frac{\text{Ceff}(P_0 \cup F) - \text{Ceff}(P_1)}{c(F)} = \frac{2k}{(k^2 + 4a)k} = \frac{2}{k^2 + 4a}.
\end{aligned}$$

Therefore, if we pick $a \gg k^2$, we would see that

$$\frac{\nabla_{P_0}(f_1)}{\nabla_{P_0}(F)} = \frac{4(k^2 + 4a)}{2(2a + k)(k + 2)} \approx \frac{4 \cdot 2}{2(k + 2)} = O(1/k) < O(1).$$

Using the above example, we can construct an example that most path-based greedy algorithms would have bad performance.

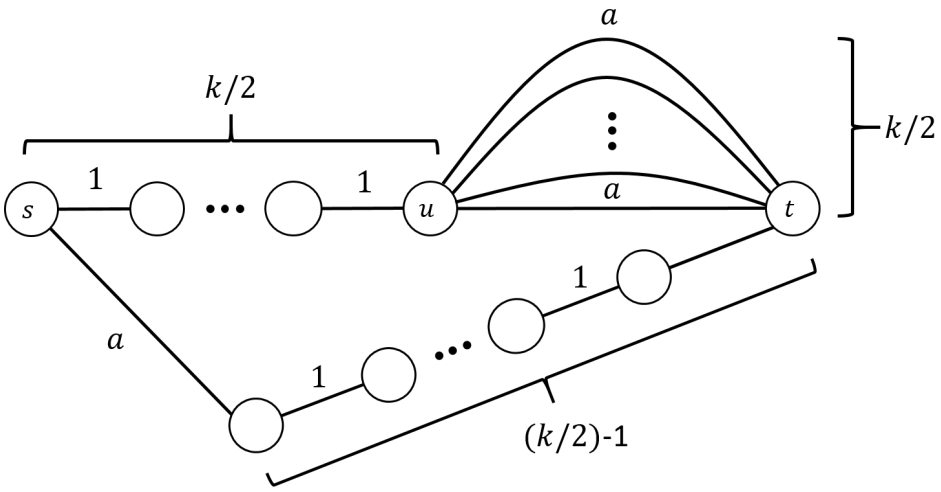


Figure 10: An example that greedy algorithm will get $\Omega(k)$-approximation if the resistances have arbitrary values.

In Figure 10, we could see that the bottom s-t path consists of $k/2$ edges with total resistance $a + \frac{k}{2} - 1$ while the top s-t path consists of $k/2 + 1$ edges with total resistance $a + \frac{k}{2}$. So the bottom s-t path has a lower resistance and lower cost, which is likely to be selected by most of the path-based greedy algorithm. Unfortunately, as long as the algorithm picks the bottom path first, it would be far from the optimal solution. With total budget k, the optimal solution is to pick all the edges in the upper half of the graph and thus the s-t effective resistance is $\frac{k}{2} + \frac{2a}{k}$. On the other hand, a path-based greedy algorithm would only select the bottom s-t path with s-t effective resistance $a + \frac{k}{2} - 1$. Therefore, assuming $a \gg k^2$, the ratio between the optimal solution and the greedy solution would be

$$\frac{a - 1 + k/2}{k/2 + 2a/k} = \frac{2ak - 2k + k^2}{k^2 + 4a} \approx \frac{k}{2} = \Omega(k).$$

From this example, we could see that for any greedy algorithm that is better than $O(k)$-approximation, it cannot just consider the "local" information of the graph (e.g. the resistance and the cost of a path), it has to take some "global" information of the graph

into account (e.g. the upper s-t path has many cheap-cost improvement). This suggests that when the resistance are arbitrary, the problem might be much harder and we might need a different technique.

Greedy approach in minimizing total effective resistance: There is an attempt in [66] to minimize the total effective resistance using the approach of submodular function maximization. Given an unweighted connected graph and a set of candidate edges, the goal is to add a subset of candidate edges to the graph to minimize the total effective resistance. They try to show the negative of total effective resistance is a submodular function, and thus there is a $(1 - 1/e)$-approximation greedy algorithm. Recall Theorem 2.2.5 that the total effective resistance is proportional to the trace of the Laplacian matrix. They define $f_e : 2^{E\backslash\{e\}} \mapsto \mathbb{R}$ be the marginal return function of edge e over all subset of edges of $E\backslash\{e\}$, i.e. $f_e(F) := \mathrm{tr}(L_F^\dagger) - \mathrm{tr}(L_{F\cup\{e\}}^\dagger)$ They try to show that $f_e(E_1) \leq f_e(E_2)$ for all $E_2 \subseteq E_1 \subseteq E\backslash\{e\}$ by considering the derivatives of f_e.

Unfortunately, their proof is incorrect and they found a counterexample to show the total effective resistance is not submodular.

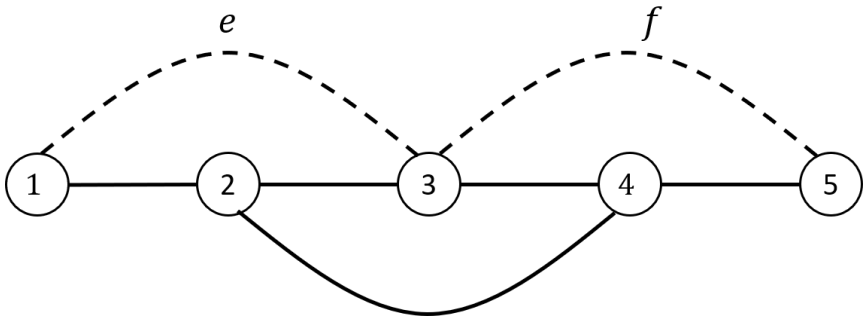


Figure 11: A counterexample showing that total effective resistance is not submodular.

In Figure 11, all edges have the same resistance. Let the set of solids edges be F. We can

check that

$$\begin{aligned} \mathrm{Reff}_{\mathrm{total}}(F) &\approx 26.67, & \mathrm{Reff}_{\mathrm{total}}(F \cup \{e\}) &\approx 20.50 \\ \mathrm{Reff}_{\mathrm{total}}(F \cup \{f\}) &\approx 20.50, & \mathrm{Reff}_{\mathrm{total}}(F \cup \{e, f\}) &\approx 13.91. \end{aligned}$$

Hence $\mathrm{Reff}_{\mathrm{total}}(F \cup \{f\}) - \mathrm{Reff}_{\mathrm{total}}(F \cup \{e, f\}) \approx 6.59 > 6.17 = \mathrm{Reff}_{\mathrm{total}}(F) - \mathrm{Reff}_{\mathrm{total}}(F \cup \{e\})$, which contradicts to the definition of submodularity.

For our greedy approach to the problem, although the effective conductance is not submodular, we generalize the constraint of submodularity to the approximate diminishing return property. In fact, as long as the factor in Conjecture 3.4.4 is a constant, we can still obtain a constant approximation algorithm.

It is plausible that an analog of Conjecture 3.4.4 in total effective resistance would be true when all edges have unit resistance, and hence the greedy algorithm may be a constant factor approximation algorithm for minimizing the total effective resistance using a similar analysis in this subsection.

3.5 Conclusions

In this thesis, we introduce a new problem of s-t effective resistance network design problem and study it from both the complexity and the algorithmic point of view. From the complexity point of view, we show that the problem is NP-hard even in the simplest setting where every edge has the same resistance and cost. On the other hand, we design a dynamic programming based algorithm for series-parallel graphs and a fast greedy algorithm for general graphs that we conjecture to have a constant approximation ratio.

Finally, other than the conjecture on the performance of our greedy algorithm, we note the following open problems:

1. How hard is it to approximate the solution when the edges no longer have unit resistance? We conjecture the approximate hardness ratio should depend on the ratio $r_{\max} := \frac{\max_e r_e}{\min_e r_e}$.

2. As in the survivable network design problem, we can generalize the problem by having an effective resistance requirement for each pair of vertices. Given the resistance upper bounds r_{uv} for all pair of vertices, the problem is to find a minimum cost subgraph such that the effective resistance between u and v is at most r_{uv} for every u, v. The main open question is that whether this problem admits a constant factor approximation algorithm.

The ultimate goal is to incorporate effective resistance as constraints for network design problems. This will give the network designers a much better control on the resulting networks. We believe these problems are interesting and solving them would require deeper understanding on the spectral properties and lead to the development of interesting and useful techniques.

References

[1] Alexander A Ageev and Maxim I Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

[2] Ajit Agrawal, Philip Klein, and R Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[3] Rudolf Ahlswede and Andreas Winter. Strong converse for identification via quantum channels. *IEEE Transactions on Information Theory*, 48(3):569–579, 2002.

[4] David J Aldous. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM Journal on Discrete Mathematics*, 3(4):450–465, 1990.

[5] Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. Graph Clustering using Effective Resistance. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 41:1–41:16, 2018.

[6] Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang. Near-optimal discrete optimization for experimental design: A regret minimization approach. *arXiv preprint arXiv:1711.05174*, 2017.

[7] Nima Anari and Shayan Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric tsp. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 20–39. IEEE, 2015.

[8] András A Benczúr and David R Karger. Approximating st minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 47–55. ACM, 1996.

[9] Danail Bonchev, Alexandru T Balaban, Xiaoyu Liu, and Douglas J Klein. Molecular cyclicity and centricity of polycyclic graphs. i. cyclicity based on resistance distances or reciprocal distances. *International journal of quantum chemistry*, 50(1):1–20, 1994.

[10] Stephen Boyd, Persi Diaconis, and Lin Xiao. Fastest mixing markov chain on a graph. *SIAM review*, 46(4):667–689, 2004.

[11] Andrei Broder. Generating random spanning trees. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 442–447. IEEE, 1989.

[12] Tanmoy Chakraborty, Julia Chuzhoy, and Sanjeev Khanna. Network design for vertex connectivity. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 167–176. ACM, 2008.

[13] Pak Hay Chan, Lap Chi Lau, Aaron Schild, Sam Chiu wai Wong, and Hong Zhou. Network design for s-t effective resistance, 2018. Manuscript.

[14] Ashok K Chandra, Prabhakar Raghavan, Walter L Ruzzo, Roman Smolensky, and Prasoon Tiwari. The electrical resistance of a graph captures its commute and cover times. *Computational Complexity*, 6(4):312–340, 1996.

[15] Shiri Chechik and Christian Wulff-Nilsen. Near-optimal light spanners. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 883–892. Society for Industrial and Applied Mathematics, 2016.

[16] Joseph Cheriyan and László A Végh. Approximating minimum-cost k-node connected subgraphs via independence-free graphs. *SIAM Journal on Computing*, 43(4):1342–1362, 2014.

[17] Joseph Cheriyan, Santosh Vempala, and Adrian Vetta. Network design via iterative rounding of setpair relaxations. *Combinatorica*, 26(3):255–275, 2006.

[18] Markus Chimani and Joachim Spoerhase. Network design problems with bounded distances via shallow-light steiner trees. *arXiv preprint arXiv:1409.6551*, 2014.

[19] Paul Christiano, Jonathan A Kelner, Aleksander Madry, Daniel A Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 273–282. ACM, 2011.

[20] Julia Chuzhoy and Sanjeev Khanna. An $o(k^3 \log n)$-approximation algorithm for vertex-connectivity survivable network design. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 437–441. IEEE, 2009.

[21] Michael B Cohen, Rasmus Kyng, Gary L Miller, Jakub W Pachocki, Richard Peng, Anup B Rao, and Shen Chen Xu. Solving sdd linear systems in nearly $m\sqrt{\log n}$ time. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 343–352. ACM, 2014.

[22] Michael Dinitz and Zeyu Zhang. Approximating low-stretch spanners. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 821–840. Society for Industrial and Applied Mathematics, 2016.

[23] Yevgeniy Dodis and Sanjeev Khanna. Design networks with bounded pairwise distance. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 750–759. ACM, 1999.

[24] Peter G Doyle and J Laurie Snell. *Random walks and electric networks*. Mathematical Association of America,, 1984.

[25] David Durfee, Rasmus Kyng, John Peebles, Anup B Rao, and Sushant Sachdeva. Sampling random spanning trees faster than matrix multiplication. *arXiv preprint arXiv:1611.07451*, 2016.

[26] Jeremy Elson, Richard M Karp, Christos H Papadimitriou, and Scott Shenker. Global synchronization in sensornets. In *Latin American Symposium on Theoretical Informatics*, pages 609–624. Springer, 2004.

[27] Alina Ene and Ali Vakilian. Improved approximation algorithms for degree-bounded network design problems with node connectivity requirements. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 754–763. ACM, 2014.

[28] Jittat Fakcharoenphol and Bundit Laekhanukit. An $o(\log^2 k)$-approximation algorithm for the k-vertex connected spanning subgraph problem. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 153–158. ACM, 2008.

[29] Lisa Fleischer, Kamal Jain, and David P Williamson. An iterative rounding 2-approximation algorithm for the element connectivity problem. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 339–347. IEEE, 2001.

[30] Ronald M Foster. The average impedance of an electrical network. *Contributions to Applied Mechanics (Reissner Anniversary Volume)*, pages 333–340, 1949.

[31] András Frank. Connectivity and network flows. *Handbook of combinatorics*, 1:111–177, 1995.

[32] Takuro Fukunaga, Zeev Nutov, and R Ravi. Iterative rounding approximation algorithms for degree-bounded node-connectivity network design. *SIAM Journal on Computing*, 44(5):1202–1229, 2015.

[33] Harold N Gabow. On the l_∞-norm of extreme points for crossing supermodular directed network lps. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 392–406. Springer, 2005.

[34] Harold N Gabow, Michel X Goemans, Éva Tardos, and David P Williamson. Approximating the smallest k-edge connected spanning subgraph by lp-rounding. *Networks*, 53(4):345–357, 2009.

[35] Shayan O Gharan. University of Washington Computer Science, CSE 599, Lecture Notes: Recent Developments in Approximation Algorithms, 2015. URL: https://homes.cs.washington.edu/~shayan/courses/cse599/. Last visited on 2018/03/30.

[36] Arpita Ghosh and Stephen Boyd. Growing well-connected graphs. In *Decision and Control, 2006 45th IEEE Conference on*, pages 6605–6611. IEEE, 2006.

[37] Arpita Ghosh, Stephen Boyd, and Amin Saberi. Minimizing effective resistance of a graph. *SIAM review*, 50(1):37–66, 2008.

[38] Michel X Goemans, Andrew V Goldberg, Serge A Plotkin, David B Shmoys, Eva Tardos, and David P Williamson. Improved approximation algorithms for network design problems. In *SODA*, volume 94, pages 223–232, 1994.

[39] Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[40] Rabih A Jabr, Ravindra Singh, and Bikash C Pal. Minimum loss network reconfiguration using mixed-integer convex programming. *IEEE Transactions on Power systems*, 27(2):1106–1115, 2012.

[41] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[42] William Thomson Baron Kelvin and Peter Guthrie Tait. *Treatise on natural philosophy*, volume 1. Clarendon Press, 1867.

[43] Gustav Kirchhoff. 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, 1847.

[44] Douglas J Klein and Milan Randić. Resistance distance. *Journal of mathematical chemistry*, 12(1):81–95, 1993.

[45] Alexandra Kolla, Yury Makarychev, Amin Saberi, and Shang-Hua Teng. Subgraph sparsification and nearly optimal ultrasparsifiers. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 57–66. ACM, 2010.

[46] Guy Kortsarz, Robert Krauthgamer, and James R Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM Journal on Computing*, 33(3):704–720, 2004.

[47] Guy Kortsarz and Zeev Nutov. Approximating minimum cost connectivity problems. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2010.

[48] Bundit Laekhanukit. Parameters of two-prover-one-round game and the hardness of connectivity problems. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1626–1643. SIAM, 2014.

[49] Lap Chi Lau, Joseph Naor, Mohammad R Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. *SIAM Journal on Computing*, 39(3):1062–1087, 2009.

[50] Lap Chi Lau and Hong Zhou. A unified algorithm for degree bounded survivable network design. *Mathematical Programming*, 154(1-2):515–532, 2015.

[51] Yin Tat Lee, Aaron Sidford, and Santosh S Vempala. Efficient convex optimization with membership oracles. *arXiv preprint arXiv:1706.07357*, 2017.

[52] Istvan Lukovits, Sonja Nikolić, and Nenad Trinajstić. Resistance distance in regular graphs. *International Journal of Quantum Chemistry*, 71(3):217–225, 1999.

[53] Aleksander Madry, Damian Straszak, and Jakub Tarnawski. Fast generation of random spanning trees and the effective resistance metric. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 2019–2036. Society for Industrial and Applied Mathematics, 2015.

[54] Peter Matthews. Covering problems for brownian motion on spheres. *The Annals of Probability*, pages 189–199, 1988.

[55] Vardges Melkonian and Éva Tardos. Algorithms for a network design problem with crossing supermodular demands. *Networks: An International Journal*, 43(4):256–265, 2004.

[56] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Cost-distance: Two metric network design. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 624–630. IEEE, 2000.

[57] Adam Meyerson and Brian Tagiku. Minimizing average shortest path distances via shortcut edge addition. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 272–285. Springer, 2009.

[58] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[59] Aleksandar Nikolov, Mohit Singh, and Uthaipon Tao Tantipongpipat. Proportional volume sampling and approximation algorithms for a-optimal design. *arXiv preprint arXiv:1802.08318*, 2018.

[60] Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook (version: November 15, 2012), 2012.

[61] Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 755–764. ACM, 2010.

[62] Prasad Raghavendra, David Steurer, and Madhur Tulsiani. Reductions between expansion problems. In *Computational Complexity (CCC), 2012 IEEE 27th Annual Conference on*, pages 64–73. IEEE, 2012.

[63] Aaron Schild. An almost-linear time algorithm for uniform random spanning tree generation. *arXiv preprint arXiv:1711.06455*, 2017.

[64] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[65] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2004.

[66] Tyler Summers, Iman Shames, John Lygeros, and Florian Dörfler. Topology design for optimal network coherence. In *Control Conference (ECC), 2015 European*, pages 575–580. IEEE, 2015.

[67] Prasad Tetali. Random walks and the effective resistance of networks. *Journal of Theoretical Probability*, 4(1):101–109, 1991.

[68] Jacobo Valdes, Robert E Tarjan, and Eugene L Lawler. The recognition of series parallel digraphs. In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 1–12. ACM, 1979.