Thesis Proposal
Combinatorial Building Blocks in Spectral Graph Theory and Their Applications

Shen Chen Xu

December 1, 2016

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Gary L. Miller, Chair
Bernhard Haeupler
Daniel D. K. Sleator
Noel J. Walkington
Ioannis Koutis, University of Puerto Rico

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.
Abstract

Recent progresses on a number of combinatorial and numerical problems benefited from combining ideas and techniques from both fields to design faster and more powerful algorithms. A prime example is the field of spectral graph theory, which involves the interplay between combinatorial graph algorithms with numerical linear algebra, and this led to the first nearly linear time solvers for symmetric and diagonally dominant (SDD) linear systems.

In this thesis proposal we focus on a number of combinatorial building blocks of spectral graph theory as well as their applications in solving SDD linear systems. We give new (and often parallel) algorithms for low diameter decompositions, low stretch tree embeddings, graph spanners, and combinatorial sparsifiers. We propose to improve our techniques developed in solving the above problems to data sensitive metrics, in particular the nearest neighbor metric or equivalently the edge squared metric, when vertices are samples from some probability distribution. We also propose to investigate spectral methods for partitioning probability distributions in Euclidean domains from random samples.
## Contents

1 Introduction .................................................. 1
   1.1 Preliminaries ........................................... 1

2 Low Diameter Graph Decomposition ....................... 3
   2.1 Low Stretch Tree Embedding ............................ 4

3 Graph Spanners and Sparsifiers ............................ 6
   3.1 Combinatorial Graph Sparsifiers ........................ 7

4 Parallel Shortest Paths and Hopsets ...................... 9

5 From General to Special Metrics .......................... 11

6 Spectral Methods for Partitioning Distributions .......... 12
   6.1 Spectral Graph Partitioning ............................ 12
   6.2 Spectral Clustering .................................... 13
   6.3 Proposed Approach .................................... 14

Bibliography .................................................... 16
Chapter 1

Introduction

Over the past few years we saw dramatic developments in fast algorithms in areas such as symmetric and diagonally dominant (SDD) linear system solvers, numerical linear algebra, combinatorial optimization, and linear programming. An important idea underpinning these recent progresses is to combine techniques developed in combinatorial and discrete algorithm design with numerical approaches. A prime example is the field of spectral graph theory, which involves the interplay between graph algorithms and linear algebra. For instance, the series of works on nearly linear time graph Laplacian solver [ST14, KMP14, KMP11, CKM+14] uses discrete graph algorithms to find good preconditioners, while [KOSZ13], another nearly linear time solver, is entirely combinatorial. These linear system solvers then lead to breakthroughs in long standing graph optimization problems such as finding maximum flows and shortest paths [CKM11, LRS13, Mad13, KLOS14, CMSV16]. The ideas and techniques developed in the graph setting such as sampling and effective resistances/leverage scores also partly inspired a number of results in numerical linear algebra [CKM+11, LRS13, Mad13, KLOS14, CMSV16], and linear programming [LS14].

This thesis will focus on some of the combinatorial building blocks used in algorithmic spectral graph theory, as well as their applications. In Chapter 2 we describe a parallel graph decomposition routine and its application in low stretch tree embeddings. In Chapter 3 we apply this result to obtain improved parallel graphs spanners and combinatorial constructions of spectral graph sparsifiers. In Chapter 4 we describe another application of this decomposition routine in approximating parallel shortest paths. In Chapter ?? we propose to study the techniques developed above in the class of data sensitive metrics, in the hope of obtaining better bounds. Finally in Chapter 6 we propose to investigate spectral methods for partitioning probability distributions from samples.

1.1 Preliminaries

In this section we quickly introduce some of the notations and concepts we will need in the rest of this proposal.

Let $G = (V, E, w)$ be a graph with vertex set $V$, edge set $E$, and edge weight function $w : E \to \mathbb{R}^+$. We usually let $V = [n] = \{1, 2, \ldots, n\}$ and $|E| = m$, and assume $w(u, v) = 0$ if vertices $u$ and $v$ are not connected by an edge. We use $\text{dist}(\cdot, \cdot)$ to denote the shortest path
distance metric on $G$ with respect to $w$, possibly decorated with superscripts or subscripts to describe further restrictions, which we will make explicit when the occasion arises. Given an directed graph $G = (V, E, w)$, the entries of adjacency $A_G$ is defined by

$$(A_G)_{u,v} = w(u, v),$$

and the diagonal degree matrix $D_G$ is defined by

$$(D_G)_{u,u} = \sum_{v \neq u} w(u, v).$$

The graph Laplacian $L_G$ associated with $G$ is then given by

$$L_G = D_G - A_G,$$

and is symmetric and positive semidefinite.

We will also be interested in the design of parallel algorithms and we use two quantities to measure the complexity of a parallel algorithm: depth and work. Depth ($D$) is the maximum length of the sequential dependencies of the algorithm, i.e. a sequence of computations in which latter ones depends on the result of earlier ones. Work ($W$) is defined the total number of operations performed by the algorithm. In practice, the number of processors available ($P$) is often limited, therefore if $W/P > D$, the actual running time of the algorithm no longer depends on the depth, but is rather bottlenecked on the work (the union of operations performed by all the processors). Therefore in this thesis, we focus on designing work efficient parallel algorithms. In other words, we try to make the work term match the sequential run time as closely as possible, since these algorithms can achieve parallel speedup with only a modest number of processors.
Chapter 2

Low Diameter Graph Decomposition

Low diameter decomposition refers to the common task of breaking up a graph into clusters with small diameter, such that few edges have endpoints in different clusters. Notice that these are two conflicting objectives: At one extreme we can achieve minimal diameter by putting each vertex into different clusters and cutting every edges; At the other extreme we can leave the graph as a single cluster with diameter up to \( n \), without cutting any edges.

Low diameter decomposition is a fundamental algorithmic tool in spectral graph theory, forming the basis of algorithms for low stretch spanning trees [AKPW95, EEST08] and low stretch tree embeddings [Bar98, FRT04, CMP+14]. It also have applications in distributed computing [Awe85, EN16], graph spanners [PS89, MPVX15], and various other graph optimization problems [CKR05, FHRT03, MPVX15]. In this thesis we will study the following definition of low diameter graph decomposition, which will become an important building block for the next few sections.

**Definition 2.0.1.** Given a possibly weighted graph \( G = (V,E,w) \), a \((\beta,d)\)-decomposition of \( G \) is a partition of the \( V(G) \) into clusters \( \{C_1, C_2, \ldots , C_k\} \) such that

1. The (strong) diameter of each \( C_i \) is at most \( d \).
2. For each edge \( e \in E \), the endpoints of \( e \) are in different clusters with probability at most \( \beta w(e) \).

Bartal [Bar96, Bar98] gave a sequential algorithm for constructing an optimal \((\beta, O(\frac{\log n}{\beta}))\)-decompositions for any \( \beta \). Blelloch et al. [BGK+14] gave a parallel construction in their parallel graph Laplacians solver, although with suboptimal parameters. In [MPX13] we presented the following simple parallel (and distributed) algorithm based on exponential delays in the graph search, outlined in Algorithm 1.

**Algorithm 1 Exponential-Delay-Clustering**

**Input:** Weighted graph \( G = (V,E,w) \) and parameter \( \beta \).

1: draw independent random variables \( X_v \sim \text{Exp}(\beta) \) for each \( v \in V \)

2: assign each \( v \in V \) to the cluster \( C_{u^*} \) centered around \( u^* = \arg\min_u \text{dist}(u,v) - X_u \)

3: return \( \{C_u\} \)

**Theorem 2.0.2.** For any graph \( G \) and parameter \( \beta \), **Exponential-Delay-Clustering** produces a \((\beta, O(\frac{\log n}{\beta}))\)-decomposition of \( G \) with high probability.

It turns out that our algorithm admits a generalization of the “edge guarantee” to arbitrary
subgraphs, this will play an important role in our parallel and distributed construction of spanners.

**Theorem 2.0.3.** Let $H$ be a subgraph of $G$ and let $\delta$ be the diameter of $H$. Then the probability that $S$ intersects $k$ or more clusters in the output of \textsc{Exponential-Delay-Clustering}(G, $\beta$) is at most $(1 - \exp(-\beta \delta))^{k-1}$.

If we let $k = 2$ and $H$ be an edge of $G$, by observing that $1 - \exp(-\beta \delta) \leq \beta \delta$ we recover the original edge cutting probability guarantee. The proofs of the above theorems are based on the memoryless properties of the exponential distribution. The formulation of Algorithm 1, while mathematically clean, is not very intuitive. We give an alternative formulation which hopefully sheds more light on its parallel and distributed nature in Algorithm 2.

**Algorithm 2 \textsc{Exponential-Delay-Clustering-Impl}**

**Input:** Weighted graph $G = (V, E, w)$ and parameter $\beta$.

1. draw independent random variables $X_v \sim \text{Exp}(\beta)$ for each $v \in V$
2. add a super source $s$ to $G$ and edges from $s$ to each vertex $v$ with weight $X_{\text{max}} - X_v$
3. compute the shortest path tree $T$ from $s$
4. return the subtrees at the root of $T$

Line 3 from Algorithm 2 can be implemented using both existing parallel and distributed search routines. The quantity $X_{\text{max}}$ can be replaced by $O\left(\frac{\log n}{\beta}\right)$ in the distributed setting and the algorithm still succeeds with high probability. Using standard complexity bounds for parallel and distributed graph searches, we get the following theorem:

**Theorem 2.0.4.** Given any graph $G$ with $n$ vertices and $m$ edges and parameter $\beta$, \textsc{Exponential-Delay-Clustering-Impl} runs in:

1. $O(m + n \log n)$ sequential time.
2. $O\left(\frac{\log n}{\beta}\right)$ parallel depth and $O(m)$ work.
3. $O\left(\frac{\log n}{\beta}\right)$ distributed rounds with unit message length.

## 2.1 Low Stretch Tree Embedding

Low stretch spanning trees and tree embeddings have many applications in spectral graph theory, in particular as a crucial component of the fast Laplacian solvers. Let $G = (V, E, w)$ be a weighted graph with $n$ vertices and $m$ edges, and let $T = (G_T, E_T, w_T)$ be a tree such that $V \subseteq V_T$. For each edge $e = uv$ in $G$, the stretch of $e$ with respect to $T$ is

$$\text{str}_T(e) \overset{\text{def}}{=} \frac{\text{dist}_T(u, v)}{w(e)},$$

where $\text{dist}_T(\cdot, \cdot)$ is the shortest path distance in $T$ with respect to edge weights $w_T$.

Alon et al. [AKPW95] introduced the notion of the low stretch spanning tree and gave an algorithm for constructing spanner trees with average stretch $\exp(O(\sqrt{\log n \log \log n}))$. This was subsequently improved by Elkin et al. [EEST08] to $O(\log^2 n \log \log n)$, then by Abraham et al. [ABNo8] to $O(\log n \log \log n (\log \log \log n)^3)$. More recently Abraham and Neiman [AN12] showed how to construct spanning trees with $O(\log n \log \log n)$ average stretch, approaching
the optimal and conjectured\(^1\) bound of $O(\log n)$. Their algorithm runs in $O(m \log n \log \log n)$ time and is used in the $O(m \log n \log \log n)$ time solver \(^2\) by Koutis et al. [KMP\(11\)].

Generally speaking, trees with lower average stretch will lead to faster solvers for the graph Laplacians under the framework pioneered by Spielman and Teng [ST\(14\)] and subsequently improved by [KMP\(14\), KMP\(11\), CKM\(+\)\(14\)]. In order to obtain the $\tilde{O}(m \sqrt{\log n})$ time Laplacian solver in [CKM\(+\)\(14\)], we introduce two relaxation to the low stretch spanning tree objective. First we relax the requirement for the tree to be a spanning tree, and only ask the tree to be embeddable into the original graph. In other words, we require a mapping from $V_T$ to $V$ and $E_T$ to paths in $G$ such that under this map $T$ is a subgraph of $G$. This is closely related to the problem approximating arbitrary metrics with tree metrics. For this problem Bartal [Bar\(98\)] first gave an algorithm with $O(\log n \log \log n)$ expected stretch and Fakcharoenphol et al. [FRT\(04\)] subsequently improved this bound to the optimal $O(\log n)$ expected stretch. We further introduce the notion of $\ell_p$-stretch:

$$
\text{str}_T^p (e) \overset{\text{def}}{=} (\text{str}_T(e))^p.
$$

Compared to $p = 1$, this relaxation allows us to discount the cost of highly stretched edges, giving an average $\ell_p$ stretch of $O((\frac{1}{1-p})^2 \log^p n)$, but is still suitable for the iterative methods used in the solver.

The other bottleneck to a faster solver is the runtime of tree finding algorithms. To this end we combine the bottom-up algorithm of Alon et al. [AKPW\(95\)] (a linear time algorithm with suboptimal stretch guarantee) with the top down approach of Bartal [Bar\(98\)] (an expensive algorithm with good stretch guarantee), both of which have low diameter decomposition as a core subroutine. This gives us to the following result on finding good tree embeddings for Laplacian solvers.

**Theorem 2.1.1.** Let $G = (V, E, w)$ a weighted graph with $n$ vertices and $m$ edges. For any $0 < p < 1$, there is an algorithm for constructing a tree $T$ embeddable into $G$ such that for any edge $e \in E$ its expected $\ell_p$-stretch in $T$ is $O((\frac{1}{1-p})^2 \log^p n)$. The runtime of this algorithm is $O(\frac{1}{1-p} m \log \log n)$ in the RAM model.

\(^1\)By Alon et al. [AKPW\(95\)].

\(^2\)All the solver runtimes in this section omit a $O(\log(1/\epsilon))$ factor.
Chapter 3

Graph Spanners and Sparsifiers

A (multiplicative) spanner is a sparse subgraph that approximates all-pair distances of the original graph up to some multiplicative factor.

**Definition 3.0.1.** Given a possibly weighted graph $G = (V, E, w)$, a $k$-spanner $H$ is a subgraph of $G$ such that for any $u, v \in V$, $\text{dist}_G(u, v) \leq k \cdot \text{dist}_H(u, v)$.

Given a stretch factor $k$, the goal is then to find spanners with fewest possible number of edges. Peleg and Schäffer [PS89] introduced this graph theoretic concept and gave a linear time sequential algorithm for $(2k - 1)$-spanners in unweighted graphs with $O(n^{1+1/k})$ edges, for any parameter $k$. This result was later extended by Althöfer et al. [ADD+93] to weighted graphs. This trade-off between the stretch factor and the spanner size is essentially optimal, up to a conjecture on graph girth $[?]$. ¹

Spanners have numerous applications such as in distributed computing [Awe85, PU89], approximate shortest path distances [ABCP98, Coh98, TZ05], and graph sparsification [KX16], which we will discuss in more detail in Section 3.1.

In [MPVX15] we introduced the following parallel and distributed algorithm for finding $O(k)$-spanners in unweighted graphs.

**Algorithm 3 Unweighted-Spanner**

**Input:** Unweighted graph $G = (V, E)$ and parameter $k$.

1: $S \leftarrow \text{EXPO-NENTIAL-DELAY-CLUSTERING}(G, \log n/k)$
2: for each pair of adjacent cluster do
3: add an (arbitrary) edge between them to $S$
4: return $S$

**Theorem 3.0.2.** Given an unweighted graph $G$ and parameter $k > 0$, **Unweighted-Spanner** produces a $O(k)$-spanner of $G$ with high probability, the expected size of the produced spanner is $n^{1+1/k}$.

We can extend this result to the weighted case, by first partitioning the edge set into disjoint subsets according to weights, then running the unweighted algorithm on each subset with appropriate graph contraction. However we lose the distributed property of the unweighted algorithm.

¹ Namely, there exists graphs with $\Omega(n^{1+1/k})$ edges and girth greater than $2k$
A spectral sparsifier of graph $G$ is a sparse graph $H$ sharing the same vertex set and such that
\[\forall x \in \mathbb{R}^n \quad (1 - \epsilon)x^T L_G x \leq x^T L_H x \leq (1 + \epsilon)x^T L_G x,\]
for some error parameter $\epsilon$. It follows that a spectral sparsifier shares most of the spectral properties of the original graph despite having fewer edges, allowing it to substitute the original graph in many applications [?] in order to save computation.

Spectral sparsification of graphs was introduced by Spielman and Teng as a component in the first nearly-linear time SDD linear system solver [ST13, ST11, ST14], and keeps to play a crucial role in the subsequent solvers [CKM+14]. Spielman and Teng’s sparsification algorithm is combinatorial in nature, it relies on intricate graph partitioning followed by uniform sampling in some of the partitions. Unfortunately this produces a sparsifier of size $O(n \log^c n / \epsilon^2)$ for a fairly large constant $c$. Spielman and Srivastava [SS11] later introduced an elegant construction of spectral sparsifiers with only $O(n \log n / \epsilon^2)$ edges based on sampling edges with effective resistances. However, in order to estimate these effective resistances, their algorithm requires $O(\log n)$ linear system solves in the graph Laplacian.

Recent efforts have been made in trying to design combinatorial algorithms for graph sparsification [KP12]. In [Kou14] Koutis showed how to compute estimates of effective resistances by repeatedly finding spanners with $O(\log n)$ stretch, and gave a nearly-linear work parallel combinatorial algorithm for constructing sparsifiers of size $O(s \log^2 \rho \log^2 n \rho / \epsilon^2 + m / \rho)$, where $s$ is the size of the spanners used and $\rho$ is a parameter.
Using the spanner construction from the previous section, and tightening the “resistance in parallel” argument from [Kou14] we obtain the first combinatorial algorithm for graph sparsification down to $O(n \log^2 n / \epsilon^2)$ edges.

**Theorem 3.1.1.** There exists an algorithm, when given a graph $G$ and error parameter $\epsilon$, outputs a graph $\tilde{G}$ that:

1. With probability $1 - 1/n^2$, $(1 - \epsilon)x^T L_G x \leq x^T L_H x \leq (1 + \epsilon)x^T L_G x$ for all $x \in \mathbb{R}^n$.
2. The expected number of edges in $\tilde{G}$ is $O^*(n \log^2 n \text{ poly}(\log \log n) / \epsilon^2)$.

Compared to the Spielman-Srivastava sparsifier which also runs in nearly linear time, the size of our combinatorial sparsifier is bigger by a factor of $O(\log n)$, this leads to the following natural open question:

**Question 3.1.2.** Can we improve the size of the combinatorial sparsifiers to $O(n \log n / \epsilon^2)$ (ignoring poly$(\log \log n)$ factors)?
Chapter 4

Parallel Shortest Paths and Hopsets

In this section we discuss another application of our low diameter decomposition algorithm: approximating shortest paths in parallel. In the sequential setting, Dijkstra’s algorithm can be implemented in $O(m + n \log n)$ time using Fibonacci heap, and the ultimate goal for parallel shortest path algorithms would be to achieve low depth (polylogarithmic or even simply sub-linear) with about the same amount of work (i.e. nearly linear). However such an algorithm remained elusive for decades, and this is referred to as the transitive closure bottleneck in the literature.

A natural direction is then to consider approximations. Hopsets were formalized by Cohen [Coh00] as a crucial component for parallel approximate shortest paths algorithms, and was implicitly used in a number of earlier works [KS97, UY91]. The goal is to add a set of extra edges to the graph so that we can get good approximation on distances by only considering paths with few edges.

**Definition 4.0.3.** Given a graph $G = (V, E, w)$, a $(\epsilon, h, m')$-hopset is a set of edges $E'$ such that:

1. $|E'| \leq m'$.
2. Each edge $uv \in E'$ corresponds to a $uv$-path $p$ in $G$ such that $w(uv) = w(p)$.
3. For any vertices $u$ and $v$, with probability $1/2$ we have:

$$dist_{E \cup E'}^h(u, v) \leq (1 + \epsilon)dist_E(u, v).$$

Here $dist_{E \cup E'}^h(\cdot, \cdot)$ denotes the shortest path distance using only $h$ edges from $E$ and $E'$.

Given an $(\epsilon, h, m')$-hopset, Klein and Subramanian [KS97] showed how to approximate shortest paths in $O(h/\epsilon)$ depth and $O((m + m')/\epsilon)$, here $m$ is the number of edges including the additional hopset. Thus in the rest of this section, we focus on the problem of finding hopsets with $h = o(n)$ and $m' = O(m)$.

Building on Cohen’s work [Coh00] and using the low diameter graph decomposition from the previous section, we showed the following results in [MPVX15].

**Theorem 4.0.4.** Let $G$ be an unweighted graph with $n$ vertices and $m$ edges. For any constant error factor $\epsilon, \delta > 1$ and $\gamma_1 < \gamma_2 < 1$, we can construct a $(\epsilon \log n, h, O(n))$-hopset in expected $O(n^{\gamma_2} \log^2 n \log^* n)$ depth and $O(m \log^{1+\delta} n / \epsilon^\delta)$ work, where $h = n^{1+1/\delta + \gamma_1(1-1/\delta) - \gamma_2}$.

Figure 4.1 contains a comparison of our hopset construction and previous works (ignoring the dependencies on $\epsilon$). Our main contribution here is to achieve nearly linear work with
<table>
<thead>
<tr>
<th>Hop count</th>
<th>Size</th>
<th>Work</th>
<th>Depth</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n^{1/2})$</td>
<td>$O(n)$</td>
<td>$O(mn^{1/3})$</td>
<td>$O(n^{0.5} \log n)$</td>
<td>[KS97, SS99]</td>
</tr>
<tr>
<td>$O(\text{poly log } n)$</td>
<td>$O(n^{1+a}) \text{ poly log } n$</td>
<td>$\tilde{O}(mn^a)$</td>
<td>$O(\text{polylog } n)$</td>
<td>[Cohoo]</td>
</tr>
<tr>
<td>$(\log n)^{O((\log \log n)^2)}$</td>
<td>$O\left(n^{O\left(\frac{1}{\log \log n}\right)}\right)$</td>
<td>$\tilde{O}\left(mn^{O\left(\frac{1}{\log \log n}\right)}\right)$</td>
<td>$(\log n)^{O((\log \log n)^2)}$</td>
<td>[Cohoo]</td>
</tr>
<tr>
<td>$O(n^{\frac{4+\alpha}{3+\alpha}})$</td>
<td>$O(n)$</td>
<td>$O\left(m \log^{3+a} n\right)$</td>
<td>$O\left(n^{\frac{4+\alpha}{3+\alpha}}\right)$</td>
<td>new</td>
</tr>
</tbody>
</table>

Figure 4.1: Performances of Hopset Constructions, omitting $\epsilon$ dependency.

Sublinear depth, as oftentimes work is the bottleneck to empirical performances due to limited number of processors.

For an concrete example, we can set $\epsilon' = \epsilon / \log n$, $\delta = 1.1$, $\gamma_2 = 0.96$ and $\gamma_1$ to some small constant, then applying Klein and Subramanian’s parallel search routine [KS97] to get the following bound.

**Corollary 4.0.5.** Let $G$ be an unweighted graph with $n$ vertices and $m$ edges. For any constant error factor $\epsilon' > 0$, there exists an algorithm for finding $(1+\epsilon')$-approximation to any $s-t$ shortest path in $G$ that runs in $O(n^{0.96} \log^2 \log^* n)$ depth and $O(m \log^{3.2} n)$ work.

Without going into too much detail, we note that Theorem 4.0.4 can be extended to graphs with edge weights.

**Theorem 4.0.6.** Let $G$ be a weighted graph with $n$ vertices and $m$ edges. For any constant error factor $\epsilon$, $\delta > 1$ and $\gamma_1 < \gamma_2 < 1$, we can construct a $(\epsilon \log n, h, O(n))$-hopset in expected $O((n/\epsilon)\gamma_2 \log^2 n \log^* n)$ depth and $O(m \log^{1+\delta} n / \epsilon^\delta)$ work, where $h = n^{1+1/\delta-\gamma_1(1-1/\delta)-\gamma_2 / \epsilon^{1-\gamma_2}}$. 

10
Chapter 5

From General to Special Metrics

In this section we propose to study our techniques the from previous sections in special classes of metrics in the hope of obtaining better bounds. In Euclidean space, improved results on low diameter decomposition and spanners are known: combining the partitioning algorithm of Charikar et al. [CCG+98] with the Johnson-Lindenstrauss lemma gives an algorithm for \((\beta, O(\sqrt{\log n}/\beta))\)-decompositions of \(n\) points in Euclidean space. This decomposition is can in turn be applied to obtain improved Euclidean spanners [HPIS13]. A crucial fact enabling these improvements is that in \(d\) dimensions, the length of the projection of a vector onto a random direction is only \(1/\sqrt{d}\) of its original length. By decoupling the starting points of our exponential delayed search from the points that we are trying to partition, we believe we can obtain a similar statement, thus similar improvement of decompositions and spanners in Euclidean space.

Of particular interest to us is the class of data sensitive metrics. Given a set of samples from an underlying probability density, one can define a metric on the samples or on the entire space as a function of the density or the samples themselves. One example is the edge squared metric defined in [CFM+15], where one constructs a complete graph on the set of samples with edge weights being the square of the Euclidean distances, and taking the shortest path metric on the graph. This metric favors paths through dense areas (areas with high probability density) because \(f(x) = x^2\) is a convex function, and has nice properties for data analysis and clustering.

There are a variety of problems that we are interested in this class of metrics such as spanners, hopsets and distance oracles. To our knowledge, graph problems in data sensitive metrics is relatively unexplored, and this can be one open-ended direction of further research in this thesis.
Chapter 6

Spectral Methods for Partitioning Distributions

In this section we propose another direction involving random samples from an unknown underlying probability distribution. Clustering is a fundamental task in machine learning and data analysis. Spectral clustering, one of the most popular clustering algorithms, has attracted a lot of attention because it can handle many non convex clustering problems and is easy to implement (many algebraic packages readily available). With the recent development of fast graph Laplacian solvers [ST14, KMP14, KMP11, CKM+14], it also achieves nearly linear run time in theory. Here we would like to have a better understanding of the theory behind spectral clustering by bridging the gap between Cheeger’s inequality on normalized graph cuts and its continuous counterpart.

Spectral methods, with which one looks to the spectrum of a graph or a manifold for insights on its structure, has long history. Cheeger [Che70] first related the isoperimetric number of a manifold to the second smallest eigenvalue of the Laplace-Beltrami operator. Shortly after, Fiedler [Fie73, Fie75] noticed the relation between the second smallest eigenvector of the graph Laplacian and bi-partition of the graph. Building on these Alon and Milman [Alo86, AM84, AM85] proved the discrete analogue of Cheeger’s result on graphs. More recently, the multi-way partition problem has been studied on graphs [LGT14] and manifolds [Fun13], relating the quality of $k$-way partitions with the $k$ lowest eigenpairs of the spectrum.

In this thesis, we focus on the setting where the points to be clustered come from a probability distribution in $\mathbb{R}^d$. In addition to clustering these data points, we are also interested in partitioning the probability distribution and its domain.

6.1 Spectral Graph Partitioning

Given a graph $G = (V, E)$, an edge weight function $w : E \to \mathbb{R}^+$ and a vertex mass function $m : V \to \mathbb{R}^+$, we define the isoperimetric ratio of a set $S \subset V$ to be

$$\phi(S) \overset{\text{def}}{=} \frac{\text{Cut}(S, \bar{S})}{\min\{\text{Mass}(S), \text{Mass}(\bar{S})\}},$$
where \( \text{Cut}(S, \bar{S}) = \sum_{u \in S, v \in \bar{S}} w(u, v) \) and \( \text{Mass}(S) = \sum_{v \in S} m(v) \). The isoperimetric number of the graph \( G \) is then defined to be

\[
\Phi(G) \overset{\text{def}}{=} \min_{S \subseteq V} \phi(S). \tag{6.1}
\]

Computing \( \Phi(G) \) and the minimizing set \( S \) is equivalent up to constant factors to many other widely used cut objectives such as sparsest cut and normalized cut, and we will refer to (6.1) as the isoperimetric cut.

Spectral graph partitioning solves a convex relaxation of this problem by solving for the second smallest eigenpair of the graph Laplacian \( L_G x = \lambda_2 x \), and returning the best threshold cut using the eigenvector \( x \) (i.e. vertices whose value in \( x \) is above certain threshold are returned). Cheeger’s inequality on graphs states that

\[
\frac{\lambda_2}{2} \leq \Phi(G) \leq \sqrt{2 \lambda_2 \max_{v \in V} \frac{d(v)}{m(v)}}, \tag{6.2}
\]

where \( d(v) \) is the degree of \( v \). Furthermore the cut \( S_{\text{spectral}} \) produced by the spectral graph partitioning algorithm satisfies

\[
\phi(S_{\text{spectral}}) \leq \sqrt{2 \lambda_2 \max_{v \in V} \frac{d(v)}{m(v)}},
\]

thus we have the following approximation guarantee

\[
\Phi(G) \leq \phi(S_{\text{spectral}}) \leq \sqrt{\Phi(G) \max_{v \in V} \frac{d(v)}{m(v)}}.
\]

### 6.2 Spectral Clustering

In machine learning, often we are only given points \( X = \{x_1, x_2, \ldots, x_n\} \) in \( \mathbb{R}^d \) instead of a graph. To apply spectral clustering one needs to first construct an affinity graph with \( X \) as the vertex set. There exists several popular ways for constructing the affinity graph (e.g. the \( r \)-neighborhood graph, the \( k \)-nearest neighbor graph, and the complete graph), as well as ways for choosing the edge weights (e.g. unit weights and Gaussian weights) [Lux07]. Assuming the data points \( X = \{x_1, x_2, \ldots, x_n\} \) are i.i.d. samples drawn from an underlying probability distribution, Maier et al. [MLHo8] showed that different graph constructions in fact lead to different graph partitions in the limit (thus different outcomes of the data clustering problem). Therefore, it makes sense to directly define and study a partitioning objective of the underlying data space, at least under this probabilistic assumption.

One can define the continuous analogue of the graph partitioning objective from (6.1). Let \( M \) be a bounded domain in \( \mathbb{R}^d \), and \( A \subset M \) an open subset, define

\[
h(A) \overset{\text{def}}{=} \frac{\text{Vol}_{d-1}(\partial A)}{\min\{\text{Vol}_d(A), \text{Vol}_d(\bar{A})\}}.
\]
Then the Cheeger constant of $M$ is defined as

$$H(M) \overset{\text{def}}{=} \inf_{A \subset M} h(A).$$

This is the approach taken by Hein et al. [HAL05]. They showed that as the number of samples $n$ goes to infinity, for an $r$-neighborhood graph $G$ with suitably chosen $r$ and unit edge weights, the graph Cheeger constant $\Phi(G)$ can be related to $H(M)$. However, consistent estimation of $H(M)$ using an affinity graph remains open. Trillos et al. [TSB+14] also considered $r$-neighborhood graphs with Gaussian edge weights. They established the consistency of a few NP-hard graph partition problems and showed the optimal graph cuts converge to the optimal continuous cuts.

Belkin and Niyogi studied the limiting Laplacian of the complete graph on $X$ with Gaussian edge weights. They showed convergence of the graph Laplacian to the Laplace-Beltrami operator of $M$ [BN08], as well as convergence of eigenvalues and eigenfunctions [BN06]. In a closely related work, Trillos et al. [TS15] also considered the complete graph with Gaussian similarity measure and showed convergence of the graph spectrum to that of the limiting Laplace operator. Together with Cheeger’s inequality on manifolds [Che70], these results yield an upper bound on the quality of the partition produced by (the continuous analogue of) spectral clustering in terms of the objective in (6.3):

$$H(M) \leq H(A_{\text{spectral}}) \leq \sqrt{2\lambda_2}$$

where $\lambda$ is the second smallest eigenvalue of $M$. Unfortunately, unlike in the graph case (6.2), a lower bound of $H(M)$ in terms of $\lambda_2$ does not exist in general. Buser [Bus82] gave a lower bound in terms of the Ricci curvature of $M$, but this result does not apply to manifolds with a boundary.

This thesis proposes to build upon this line of work based on Cheeger and Poincaré-type inequalities on graph and manifolds. One common aspect of these previous works is that they are all asymptotic analyses where the number of samples tends to infinity. In this thesis, we would like to understand the performance guarantee of these spectral methods as a function of the number of samples.

### 6.3 Proposed Approach

First suppose we are given an oracle for the underlying density function over $M$. We will devise numerical schemes to approximate the second smallest eigenvalue and the corresponding eigenvector of $M$.

**Question 6.3.1.** Given the underlying density, how do we construct graphs and pick edges so that $\lambda_2(G) \approx \lambda(M)$?

Computing eigenvalues and eigenfunctions is a well-studied problem in numerical analysis, and many classical results have been developed, for example the finite difference method and the finite element method. In order to leverage the recent fast graph Laplacian solvers, we propose to use the control volume method instead [Nic92, MTTW99]. On a high level, the control volume method constructs a Delaunay triangulation graph, and assigns edge weights and vertex masses using properties from the dual Voronoi diagram and the underlying density.
This will allow us to obtain error formulas for the eigenvalue problem and the partitioning problem in terms of the number of samples.

Of course the underlying density is unknown to us, thus whenever our numerical scheme calls the density oracle, we need to replace it with some density estimator. We will then try to analyze the expected behavior of our scheme under random samples, as well as to prove concentration results. This leads us to the study of the following abstract question: given a graph with random edge weights, how does the spectrum compare the expected graph?

**Question 6.3.2.** Let $G = (V, E, w)$ be a graph and $\tilde{G} = (V, E, \tilde{w})$ be the same graph with different and random edge weights $\tilde{w}$. Suppose that for each $e \in E$, $E[\tilde{w}(e)] = \Theta(w(e))$, how does $\lambda_2(\tilde{G})$ compare to $\lambda_2(G)$? What property does $\tilde{w}$ need to have in order to guarantee $E[\lambda_2(\tilde{G})] = O(\lambda_2(G))$? What can be said about the corresponding eigenvector?

One possible way obtain concentration is to apply or extend tail bounds for sums of random matrices [Tro12]. If we limit ourselves to kernel density estimators, our randomly weighted Delaunay triangulation graph can be decomposed into a sum of i.i.d. random graphs, each coming from a single sample.

Recall that Buser’s inequality [Bus82] gave a lower bound of $H(M)$ in terms of $\lambda_2(M)$ and the Ricci curvature of $M$. However this inequality does not apply to manifolds with boundaries, for example hypercubes which are very natural domains for clustering problems. Therefore we are interested in other conditions that will guarantee a similar bound on $H(M)$.

**Question 6.3.3.** What condition (other than Buser’s inequality) does the underlying density needs to satisfy in order to guarantee a non-trivial lower bound on the Cheeger constant in terms of $\lambda_2$?

Lastly, one draw back to this approach is that in high dimension the Delaunay triangulation graph we use can become dense. One more ambitious question to tackle is the following.

**Question 6.3.4.** We know that all graph have nearly linear sized spectral sparsifiers. Can we sparsify the Delaunay triangulation without first constructing it explicitly?
Bibliography

[BGK+14] Guy E. Blelloch, Anupam Gupta, Ioannis Koutis, Gary L. Miller, Richard Peng, and


[Coh98] Edith Cohen. Fast algorithms for constructing $t$-spanners and paths with stretch $t$. 17


