Unsupervised Learning Algorithms For Graph Data

by Mahesh Mohan

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

August 31, 2017

Dissertation directed by

Claire Monteleoni,
Assistant Professor of Computer Science
The School of Engineering and Applied Science of The George Washington University certifies that Mahesh Mohan has passed the Final Examination for the degree of Doctor of Philosophy as of April 19, 2017. This is the final and approved form of the dissertation.

Unsupervised Learning Algorithms For Graph Data

Mahesh Mohan

Dissertation Research Committee:

Claire Monteleoni, Assistant Professor of Computer Science, Dissertation Director
Rahul Simha, Professor of Computer Science, Committee Member
Poorvi Vora, Professor of Computer Science, Committee Member
Murray Loew, Professor of Electrical and Computer Engineering, Committee Member
Abstract

Unsupervised Learning Algorithms For Graph Data

This thesis presents new techniques for unsupervised learning from graph data. Data from many applications, including social networks, transportation systems, images, and climate variables measured over multiple latitudes and longitudes, can be represented as graphs. Graphs provide a convenient representation of relationships between variables. Unsupervised learning is a critical first step in making sense of unlabeled data. The recent surge in the deployment of sensors and data acquisition systems have led to an explosion in the amount of data available, a significant portion of which is unlabeled.

Many existing approaches for unsupervised machine learning on graphs are computationally expensive and do not scale well to large amounts of data. The work presented here addresses this issue by proposing theoretically justifiable algorithms that provide a good tradeoff between computation time and accuracy. This thesis focuses on two unsupervised learning tasks: spectral clustering and nearest neighbor search using graph kernels.

I present two novel sampling-based approaches to spectral clustering, that scale to a significantly larger number of input points compared to the current state of the art. I also present theoretical results showing that these sampling-based approaches yield a good approximation with respect to a popular spectral clustering objective. In the case of nearest neighbor search, graph kernels are used to propose a hierarchical scheme that allows for efficient queries when the input data has a large number of points in a high dimensional space.
Contents

Abstract iii

List of Figures v

1 Introduction 1

1.1 Spectral Clustering ......................................................... 1

1.1.1 Contributions .......................................................... 3

1.2 Graph Kernels ................................................................. 4

1.2.1 Contributions .............................................................. 5

2 Preliminaries 7

2.1 Graph Clustering Objectives .............................................. 7

2.2 Spectral Clustering Algorithm .......................................... 8

2.3 Related Work ................................................................ 10

2.3.1 The Nyström approximation .......................................... 10

2.3.2 Weighted Kernel K-means ........................................... 13

2.3.3 Graph Sparsification .................................................... 14

3 Nyström-based Spectral Clustering For Sparse Matrices 15

3.1 Introduction ................................................................. 15

3.2 Observations For Sparse Matrices ..................................... 15

3.2.1 Distances Between Sparse Columns .............................. 15

3.2.2 Sample Complexity ...................................................... 17

3.3 Proposed Approach ....................................................... 17

3.4 Analysis ........................................................................ 18

3.4.1 Justification for the initial sampling scheme .................. 18

3.4.2 Why Matrix norms are insufficient for spectral clustering ........................................... 21

3.4.3 Justification for the densification step .................................. 23

3.5 Experimental Results ..................................................... 25

3.5.1 Matrix Norm Errors .................................................... 25
3.5.2 Spectral Clustering ........................................... 28
3.6 Summary .......................................................... 30

4 Effect of Uniform Sampling on Spectral Clustering ........ 32
4.1 Introduction ....................................................... 32
4.2 Preliminaries ..................................................... 33
   4.2.1 Weighted Kernel $k$-means .......................... 33
   4.2.2 Spectral Clustering using Norm-Cuts ............... 34
   4.2.3 Relation between weighted kernel $k$-means and spectral clustering .... 34
4.3 Modified Framework ............................................ 34
4.4 Analysis .......................................................... 35
4.5 Experiments ...................................................... 42
4.6 Summary .......................................................... 44

5 Environment Selection and Hierarchical Place Recognition 45
5.1 Introduction ....................................................... 45
5.2 Place Recognition ................................................ 45
5.3 Related Work ...................................................... 48
5.4 Similarity Criterion ............................................. 51
   5.4.1 Notation ....................................................... 51
   5.4.2 Theoretical Justification ................................. 52
   5.4.3 Constructing the WWC Matrix ......................... 54
   5.4.4 Computational Considerations ......................... 55
5.5 Hierarchical Place Recognition ................................ 55
   5.5.1 Single Environment ....................................... 55
   5.5.2 Hierarchical Approach ................................... 56
5.6 Experimental Results .......................................... 58
   5.6.1 Hierarchical Place Recognition ....................... 58
   5.6.2 Environment Selection ................................... 60
5.7 Summary .......................................................... 65
1.1 Examples of data that can be conveniently represented as graphs: (a) shows
an example of social network [1]. (b) shows the structure of the E. coli protein
fragment modeled as a graph. Nodes represent structure elements and the
edges represent neighborhood along the amino acid chain [52, 6, 47]. . . . . . 1

1.2 Examples where the nodes of a graph need to be clustered. The example on
the left shows a case where the nodes of a social network are grouped into
communities [2]. The figure on the right shows an example where pixels of
an image are grouped together based on similarity in their intensity [53]. We
will describe how an image is represented as a graph in a later chapter. . . . . 2

1.3 Example of dataset where spectral clustering outperforms $k$-means clustering
[34]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

2.1 Justification for the norm cuts objective [46]: Figure (a) shows the input
graph whose nodes need to be clustered. Figure (b) shows the set of edges
removed by different objectives. The edges denoted by the red dotted line will
be chosen if we minimize the min cuts objective, whereas the edges marked
in green will be chosen for removal is we minimize the norm cuts objective.
This shows that the norm cuts objective results in more balanced clusters
[48]. Hence we use the norm cuts objective for the rest of this work. . . . . 8

3.1 Comparison of errors for the modified algorithm (modified), Nyström with
uniform sampling (unif), Subsampled Random Fourier Transform (srft), Gaus-
sian Projections (gaussians) for different datasets. We restrict the rank of
these datasets to 20. The plots show that the modified method performs
better than uniform sampling for all the errors. . . . . . . . . . . . . . . . . . 26

3.2 Comparison of running time: Modified Nyström (modified) v/s Nyström with
Subsampled Random Fourier Transform (srft), Gaussian Projections (gauss)
for different datasets. We restrict the rank of these datasets to 20. The plots
show that the modified method is computationally better than both gauss
and srft. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
3.3 Comparison of columns with norm zero: Modified Nyström (modified) v/s Nyström with Subsampled Random Fourier Transform (srtf), Gaussian Projections (gauss) for different datasets. The plots show that the modified method covers all the columns with significantly fewer samples. In contrast, uniform sampling fails to cover all the columns even after sampling 20% of the columns.

3.4 The first and second rows show the clustering obtained by [53] and the modified approach respectively on several toy problems. The third and fourth rows show the norm cuts ratio and the time taken for each dataset. It can be seen that the modified approach yields a norm cuts ratio similar to [53], [15] while its computation time is comparable to using Nyström with uniform sampling.

3.5 Figures (a) and (c) show the norm cuts ratio, whereas (b), (d) show the time taken by different methods. The modified method outperforms Nyström with uniform sampling (nys) in terms of norm cuts ratio while taking a similar amount of time. The baseline methods shi [53] and kulis [15] use the entire affinity matrix. Thus they offer lower norm cuts ratio, and have a significantly higher computation time.

4.1 The first and second rows show the norm cuts ratio and the time taken for each dataset. It can be seen that wkk-unif and wkk-col, which are variants of the proposed framework yield results similar to kulis and shi, which use the entire affinity matrix. wkk-unif and wkk-col also outperform nys-col and nys-unif in terms of accuracy and computation time.

4.2 Figure (a) shows the norm cuts ratio, whereas (b) shows the time taken by different methods. This shows that methods using weighted kernel $k$-means outperform their Nyström counterparts with respect to computation time, while obtaining a comparable norm cuts ratio.
4.3 Behavior of the various approaches when the graph size increases. This shows that by using only 50% of the points, wkk-unif and wkk-col achieve accuracy comparable to kulis and shi while taking significantly less time. The difference in accuracy reduces as the size of the graph increases. wkk-unif and wkk-col also outperform nys-unif and nys-col both in terms of accuracy and computation time.

5.1 Outline of the ideas proposed in the paper. (a) The figure on the left shows a potential query image being compared against a set of database images grouped into environments shown on the right. Different rows represent images obtained from different datasets. (b) The similarity scores computed using the proposed criterion to compare the query image to the existing environments. The plot of the scores shows that it is sufficiently discriminative to be used to determine the environment the image is most likely to be from. (c) Selecting the best environment helps reduce the number of images to be compared against, thereby improving the search time. We show the potential savings in computation time while querying a database of 85,000 images.

5.2 Proposed algorithm for Hierarchical Place Recognition.

5.3 Figure showing the Confusion matrix for the Environment Selection problem. The environments are {Bicocca25b, NewCollege, Whitmore, Malaga6L, StLucia, Ford2, Nordlands}. The confusion matrix shows that images from StLucia are often classified as Nordlands. From the images in the first row (StLucia) and second row (Nordlands), we see that foliage in both datasets occupy a significant portion of the features obtained, leading to ambiguity.
5.4 PR Curves for Environment Selection for different values of the sparsification threshold $\lambda$. A lower value of $\lambda$ implies fewer elements are discarded, leading to higher accuracy. The interesting thing to note here is that for all the environments, setting $\lambda = m(r)$ gives us the same performance as the full matrix (i.e., when $\lambda = 0$). This implies that words that co-occur less frequently are less informative for environment selection. This is in contrast to place recognition where words that occur rarely are considered more informative.

5.5 PR Curves for Environment Selection using vocabularies of different sizes. This monotonic improvement shows that the discriminative ability of the criterion improves with the size of the vocabulary.

5.6 PR Curves for Environment Selection using different sampling rates. This sampling rate controls the number of images used to construct the WWC matrix. Reducing the number of images used to construct the WWC reduces the discriminative ability of the proposed criterion. However, these curves show that environments can be identified correctly even when we use much fewer images. This highlights a dominant underlying structure to these environments.
Introduction

Figure 1.1: Examples of data that can be conveniently represented as graphs: (a) shows an example of social network [1]. (b) shows the structure of the \textit{E. coli} protein fragment modeled as a graph. Nodes represent structure elements and the edges represent neighborhood along the amino acid chain [52] [6] [17].

Data from many applications can be conveniently represented using graphs. Graphs are especially useful in representing relationships between variables. Fig 1.1 shows two examples. The first example represents data coming from social networks and the second shows how a protein fragment of \textit{E.coli} can be represented as a graph. Other examples include data from transportation systems, image data and climate variables measured over multiple latitudes and longitudes. The recent surge in the deployment of sensors and data acquisition systems have led to an explosion in the amount of data available.

A significant portion of the data made available by such systems is unlabeled. Unsupervised learning is a critical first step in making sense of such data. Many existing approaches for unsupervised machine learning on graphs are computationally expensive and do not scale well to large amounts of data. We describe efficient approximation algorithms that provide a good tradeoff between computation time and accuracy. This thesis will focus on two unsupervised learning tasks: spectral clustering and nearest neighbor search using graph kernels which we describe below.

1.1 Spectral Clustering

The goal of spectral clustering is to partition the nodes of a graph into disjoint groups or clusters. Fig 1.2 shows two common applications where spectral clustering is often used. The first application deals with the extraction of community structure from social networks.
Figure 1.2: Examples where the nodes of a graph need to be clustered. The example on the left shows a case where the nodes of a social network are grouped into communities [2]. The figure on the right shows an example where pixels of an image are grouped together based on similarity in their intensity [53]. We will describe how an image is represented as a graph in a later chapter.

Figure 1.3: Example of dataset where spectral clustering outperforms $k$-means clustering [34].

This is of great importance in various domains like digital marketing and recommendation systems. Another popular application in computer vision is image segmentation. This is often used as a pre-processing step for many high level computer vision tasks like object detection, scene understanding, etc.

In practice, spectral clustering has been shown to outperform $k$-means clustering in many cases where the underlying clusters are not convex, as shown in Fig 1.3. However for large datasets, spectral clustering becomes expensive both in terms of memory requirement and computation time. The goal of this thesis is to develop algorithms that produce a good approximation for spectral clustering while requiring significantly lower computation time.
As we see later in Chapter 2.3 for larger datasets, two major bottlenecks are observed in practice. First, a large matrix called the affinity matrix, that stores the pairwise similarities of graph nodes, needs to be computed and maintained in main memory. This is extremely expensive. Second, spectral clustering involves the computation of the eigenvectors of the affinity matrix, which is computationally prohibitive for large datasets.

1.1.1 Contributions

The most popular approach to speeding up spectral clustering is the Nyström approximation, which is used in practice to obtain a trade-off between the accuracy and computation time. The Nyström approximation is a technique to approximate the eigenvalues and eigenvectors of a symmetric positive semi-definite matrix by using the eigenvectors and eigenvalues of a carefully constructed smaller matrix. The size of this smaller matrix can be increased to obtain higher accuracy, but this incurs a higher computational cost. A benefit of the Nyström approximation is that only a small subset of the entries of the affinity matrix need to be computed. This eliminates the need for explicit computation of the entire affinity matrix. Specifically, in this thesis we address the following questions:

1. Are matrix norm error guarantees useful for spectral clustering?
   A number of techniques have been proposed in the literature for sampling the subset of columns used for the Nyström approximation. However the theoretical analysis of the Nyström approximation in the literature has focused on matrix norm error guarantees. We show that such guarantees are not always helpful in understanding the quality of approximation with respect to spectral clustering.

2. Can sparsity be exploited to speed up the Nyström approximation?
   When the input matrix is sparse, we show that the traditional Nyström method requires a prohibitively large number of samples to obtain a good approximation. We propose a novel sampling approach to select the landmark points used to compute the Nyström approximation. We show that the proposed sampling approach has an error bound that is equal to that of the approach in [7]. To control sample complexity, we propose a densification step based on breadth first traversal to ensure all nodes are cov-
ered. We show that the proposed densification does not change the optimal clustering. Results on real world datasets show that by combining the proposed sampling and densification schemes, we can obtain better accuracy compared to other techniques used for the Nyström method while using significantly lower number of samples.

3. Can we avoid computing the eigenvectors?

A major issue with using the Nyström approximation is that it still requires the explicit computation of the eigenvectors of a sub-matrix. As the matrix grows in size, this becomes expensive. We present a scalable approach to spectral clustering based on the following simple scheme: sample a subset of the input points uniformly at random, compute the clusters for the sampled subset using weighted kernel k-means \cite{15} and use the resulting centers to compute a clustering for rest of the data points. Our main theoretical contribution is the extension of the results in \cite{?} from k-means clustering to spectral clustering. We also show that the number of samples required by the proposed approach is independent of the input size, and only depends on the number of clusters $k$ and the diameter of the metric space. We also demonstrate experimentally that the use of weighted kernel k-means yields significant speedups while maintaining comparable accuracy.

1.2 Graph Kernels

Machine learning in many domains such as bioinformatics, chemoinformatics, drug discovery, web data mining and social networks involves the study of relationships between structured objects. Graphs are often used to model such structures, with nodes representing objects and edges the relations between them. Many applications arise in different domains like bioinformatics, drug discovery, web data mining and social networks that deal with the question “How similar are two given graphs?” For example, the work in \cite{47} predicts the toxicity of a chemical molecule by modeling it as a graph that encodes its structure. A graph kernel is used to compute the similarity between molecules of known and unknown toxicity. Graph kernels are symmetric, positive semi-definite functions which measure the similarity between two graphs or between two nodes of a graph. The fifth chapter of the thesis focuses
on applying graph kernels to the problem of place recognition in Robotics.

In place recognition, images are obtained from a camera mounted on an autonomous car. These images are often represented as a vector of visual “words”. The set of images seen so far represent the map of the environment, where each image represents a place. In place recognition, given a query image, we need to obtain the closest match from the set of previously seen images. As autonomous cars continue to create long-term maps, the number of images being considered may increase until exceeding the available computational resources. In this thesis we consider a scenario where, given multiple independent large maps, possibly from different cities or locations, a robot must effectively and in real time localize itself in one of those known maps. Since the number of images to be handled by such a system is likely to be extremely large, we find that it is beneficial to decompose the set of images into independent groups or environments.

1.2.1 Contributions

Our first contribution is the observation that for different environments, different pairs of visual words co-occurred together. For example, in urban environments, visual words pertaining to cars were seen to co-occur often, whereas in a forest, visual words pertaining to foliage were seen to co-occur. Thus a group of images obtained from the same environment show a similar pattern in the co-occurrence of visual words. This can be used to characterize the environment the images were obtained from (an environment is treated as a collection of images).

We use the observation above to propose a novel place recognition algorithm that divides the collection of images into a hierarchy of environments. By selecting the relevant environment first, we can reduce the number of images to perform place recognition. This reduces the execution time while preserving accuracy.

In order to select the relevant environment, we need to compare co-occurrence frequencies of visual words from different environments. If we represent each visual word as a node in a graph, the co-occurrence matrix can be represented using a graph, where an edge between two nodes represents the co-occurrence frequency of the corresponding visual words. Now the problem of comparing environments reduces to comparing their corresponding co-occurrence
graphs.

We propose a similarity criterion to compare co-occurrence graphs based on a reduction from the direct product graph kernel. This results in a hierarchical data structure that improves the efficiency of nearest neighbor search.
Preliminaries

We begin our discussion by providing the notation and definitions used in our work on spectral clustering. As mentioned in the Introduction, the goal of spectral clustering is to partition the nodes of a graph into disjoint groups or clusters. One way to cluster nodes of a graph is to remove a subset of edges. The set of edges removed depends on the clustering objective being used. We begin by discussing a few graph clustering objectives.

2.1 Graph Clustering Objectives

In spectral clustering, we are given a graph $G = (V, E, A)$, which is made up of a set of vertices $V$ and a set of edges $E$ such that an edge between two vertices represents their similarity. The affinity matrix $A$ is $|V| \times |V|$ whose entries represent the similarity between vertices. Let us denote $\text{links}(S_1, S_2)$ to be the sum of the edge weights between nodes in $S_1$ and $S_2$. In other words, $\text{links}(S_1, S_2) = \sum_{i \in S_1, j \in S_2} A_{ij}$.

Furthermore, let the degree of $A$ be the links of nodes in $S_1$ to all the vertices, i.e. $\text{degree}(A) = \text{links}(A, V)$. The graph partitioning problem seeks to partition the graph into $k$ disjoint partitions or clusters $V_1, ..., V_k$ such that their union is $V$.

A number of different graph partitioning objectives have been proposed and studied. For an overview, we refer the reader to in [15]. Here we outline two popular graph clustering objectives.

1. The minimum cuts objective (or min cuts objective) is expressed as,

$$\text{MinCut}(G) = \min_{V_1, \ldots, V_k} \sum_{c=1}^{k} \text{links}(V_c, V \setminus V_c)$$

2. The normalized cuts objective (or norm cuts objective) is expressed as [18, 15],

$$\text{NCut}(G) = \min_{V_1, \ldots, V_k} \sum_{c=1}^{k} \frac{\text{links}(V_c, V \setminus V_c)}{\text{degree}(V_c)}$$

The choice of graph clustering objective has a strong impact on the kind of clusters obtained. For example, Fig shows a graph whose nodes need to be clustered. Fig 2.1 (b) shows the set of edges removed by different objectives. The edges denoted by the red dotted
Figure 2.1: Justification for the norm cuts objective [46]: Figure (a) shows the input graph whose nodes need to be clustered. Figure (b) shows the set of edges removed by different objectives. The edges denoted by the red dotted line will be chosen if we minimize the min cuts objective, whereas the edges marked in green will be chosen for removal if we minimize the norm cuts objective. This shows that the norm cuts objective results in more balanced clusters [48]. Hence we use the norm cuts objective for the rest of this work.

Algorithm 1 Spectral clustering

| \textbf{Input:} dataset \( S = \{s_1, s_2, ..., s_n\} \in \mathbb{R}^d \), number of clusters \( k \), kernel function \( \kappa : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R} \) |
| \textbf{Output:} \( k \)-clustering of \( S \) |
| \( A \in \mathbb{R}^{n \times n} \) s.t. \( A_{ij} = \delta[i \neq j] \kappa(s_i, s_j) \) |
| \( D \in \mathbb{R}^{n \times n} \) s.t. \( D_{ij} = \delta[i = j] \sum_{j=1}^{n} A_{ij} \) |
| \( L = I - D^{-\frac{1}{2}} AD^{-\frac{1}{2}} \) |
| \( X \in \mathbb{R}^{n \times k} \) s.t. \( \text{SmallestEigenVectors}(L, k) \) |
| \( Y \in \mathbb{R}^{n \times k} \) s.t. \( Y_{im} = X_{im} / \sqrt{\sum_m X_{im}^2} \) |
| \( \mathcal{K} = \text{ClusterRows}(Y) \) (via any \( k \)-clustering algorithm minimizing distortion, i.e. \( k \)-means) |

In this work, we will focus on the normalized cut objective, though the results presented in this work can be generalized to other graph partitioning objectives as well. Next, we describe the popular algorithm in [34] that is used to minimize this objective.

2.2 Spectral Clustering Algorithm

In general, spectral clustering methods can be interpreted as graph partitioning algorithms and the above algorithm (Algorithm 1) can be seen as graph partitioning with a normalized-cut cost function. Algorithm 1 shows the widely used normalized spectral clustering algorithm presented in [34]. Given the set of \( n \) points \( S = \{s_1, s_2, ..., s_n\} \) the algorithm
first builds an $n \times n$ affinity matrix $A$, i.e.:

$$A_{ij} = \kappa(s_i, s_j) \text{ if } i \neq j \text{ and } 0 \text{ otherwise.}$$

Here $A_{ij}$ corresponds to the $i$'th row and $j$'th column of the affinity matrix and $\kappa$ is any kernel function accepting two input data-points and returning a scalar output. A valid kernel function has to satisfy Mercer’s condition, as explained in [10]. Let $D$ be the diagonal matrix where $D_{ii} = \sum_j A_{ij}$. In other words, the $i$-th diagonal entry in $D$ represents the degree of the vertex $i$ in the graph. Once $A$ and $D$ are computed, the normalized graph Laplacian $L$ can be constructed as $L = I - D^{-0.5} A D^{-0.5}$. The first $k$ eigenvectors of $L$ are then normalized and clustered. It was shown in [34] that one can perform spectral $k$-clustering using a perturbed version $\tilde{A}$ of the ideal affinity matrix $A$. Under certain assumptions, the clusterings obtained using $A$ and $\tilde{A}$ will be similar. Based on the analysis in [34] we know that if the four assumptions listed below are satisfied then using either $\tilde{A}$ or $A$ to perform spectral clustering will give similar partitionings of the dataset (and also similar to the true clustering of the dataset):

- **Assumption A1:** $\exists \gamma > 0 \forall i = \{1,2,...,k\} \lambda_i^2 \leq 1 - \gamma$, where $\lambda_i^2$ is the second largest eigenvalue of $L^i$, where $L^i$ is the subblock of $L$ corresponding to cluster $i$.

- **Assumption A2:** $\exists \epsilon_1 > 0 \forall i_1, i_2 = \{1,2,...,k\}, i_1 \neq i_2 \sum_{j \in S_{i_1}} \sum_{l \in S_{i_2}} \frac{\tilde{A}_{jl}}{\tilde{d}_j \tilde{d}_l} \leq \epsilon_1$, where $\tilde{d}_j = \sum_{m \in S_{i_1}} \tilde{A}_{jm}$ and $\tilde{d}_l = \sum_{m \in S_{i_2}} \tilde{A}_{lm}$ and $S_i$ is the set of points belonging to the $i$th cluster.

- **Assumption A3:** $\exists \epsilon_2 > 0 \forall i = \{1,2,...,k\} \sum_{j \in S_i} \frac{\tilde{A}_{jl}}{\tilde{d}_l} \leq \epsilon_2 (\sum_{l,m \in S_i} \tilde{A}_{lm}^2 / \tilde{d}_l \tilde{d}_m)^{-\frac{1}{2}}$.

- **Assumption A4:** $\exists \epsilon_3 > 0 \forall i = \{1,2,...,k\}, j = \{1,2,...,n_i\} \tilde{d}_j \geq (\sum_{l=1}^{n_i} \tilde{d}_l) / (C n_i)$.

Assumption A1 guarantees each cluster to be tight. Assumption A2 and A3 require data points within a cluster to be more connected to each other than they are with data points from any other cluster. Finally, the last assumption requires that the points in any cluster can never be much less’ connected than other points in the same cluster. The similarity of the clusterings obtained using $A$ and $\tilde{A}$ is then assured via Theorem 2.2.0.1. Let $y_{ij}^i$ be the $j$th row of $Y^i$, where $Y^i$ is the subblock of $Y$ corresponding to cluster $i$. Then the following theorem holds.
Theorem 2.2.0.1 (34). Let assumptions A1, A2, A3 and A4 hold. Set \( \epsilon = \sqrt{k(k-1)\epsilon_1 + k\epsilon_2^2} \). If \( \gamma > (2 + \sqrt{2})\epsilon \), then there exist \( k \) orthonormal vectors \( r_1, r_2, \ldots, r_k \) such that \( \mathcal{Y} \) in Algorithm \( \square \) satisfies

\[
\frac{1}{n} \sum_{i=1}^{k} \sum_{j=1}^{n_i} \|y_i^j - r_i\|^2 \leq 4C(4 + 2\sqrt{k})^2 \frac{\epsilon^2}{(\gamma - \sqrt{2}\epsilon)^2}.
\]

2.3 Related Work

The algorithm described above has two major computational bottlenecks, that make it extremely difficult to scale to large amounts of data.

1. Constructing the affinity matrix. For a graph with \( n \) nodes, this step has a complexity of \( O(n^2) \).

2. Computing the eigenvectors of the Normalized Graph Laplacian. For a graph of \( n \) nodes, this has a complexity of \( O(n^3) \).

Typically, the first problem is addressed in practice by using the Nyström approximation, which approximates the eigenvectors of the Normalized Graph Laplacian by sampling a subset of its rows/columns. This eliminates the need to compute the entire affinity matrix. It also replaces the computation of eigenvectors of \( n \times n \) matrix with a \( l \times l \) matrix (where \( l \) is the number of columns sampled).

The computation of eigenvectors can be completely avoided by using the connection to weighted kernel k-means described in [15].

2.3.1 The Nyström approximation

[43] describe iterative methods to compute the low rank approximation and involve the use of the entire matrix. This makes them infeasible for large matrices. An alternate approach, the Nyström approximation has been a standard tool for low rank approximation of symmetric positive semi-definite (SPSD) matrices since its introduction in [49]. In cases where the input matrix has low rank and low coherence, the Nyström approximation is known to return an exact approximation as shown in [15].
Given an input matrix $A$, the Nyström method chooses a subset of $m$ columns $C \in \mathbb{R}^{n \times m}$, and reconstructing the complete kernel matrix by $\hat{A} \approx CW_k^+ C^T$, where $W$ is the principal sub-matrix of $A$ induced by the selected columns and $W_k^+$ is the pseudo-inverse of its rank-$k$ approximation. Various methods have been proposed in the literature to construct the matrices $C$ and $W$. These can be broadly divided into three categories: projection based, sampling based and clustering based.

Projection based methods use a data-independent projection matrix to represent the entries of the matrix as points in lower dimensional space. In other words, the matrices $C$ and $W$ are given by $C = AS$ and $W = S'AS$ respectively, where $S$ is the projection matrix. Examples of projection matrices used include gaussian projections, Subsampled Random Fourier transforms as described in [19].

There has been recent work in improving the efficiency of approximation in the case of spectral clustering by applying clustering-based techniques to columns of Normalized Graph Laplacians. [55] and [40] used the $k$-means algorithm (KS), to select $k$ centroids as landmark points. These landmark points are used to compute the Nyström approximation. However, the results in [7] show that both methods perform poorly for non-convex clusters.

The incremental sampling (IS) algorithm proposed in [56], first randomly samples two points from a dataset, to compute a similarity matrix between the sampled points and the remaining points. The algorithm picks the point with the smallest variance, and then iteratively repeats the process until a desired number of landmarks is reached. However, as shown in [54], in higher dimensions the variance of the Euclidean distance tends to zero. In such cases IS may pick inappropriate landmark points and perform similar to uniform sampling. In order to address this behavior of points in high dimensional spaces, [54] proposed minimum similarity sampling (SS) for clustering in high-dimensional spaces. However, it is outperformed by IS on low dimensional data.

The work of [7] introduced Minimum Sum of Squared Similarities (MSSS), which approximately maximizes the determinant of the reduced similarity matrix that represents the mutual similarities between sampled data points. However all these methods can become computationally expensive as the size of the matrix grows.

Both projection based approaches and clustering based approaches require the computa-
Algorithm 2 Nyström method for matrix approximation [9, 49]

1: Input: matrix $L$, $l$ - number of columns sampled, $r$ - rank approximation ($r \leq l << n$)
2: Output: $\tilde{\Sigma}$ and $\tilde{U}$ such that $\tilde{L} = \tilde{U} \tilde{\Sigma} \tilde{U}^\top$

3: $L \leftarrow$ indices of $l$ columns sampled
4: $C \leftarrow G(:, L)$
5: $W \leftarrow C(L, :)$
6: $W_r \leftarrow$ best $r$-rank approximation to $W$
7: $\tilde{\Sigma} = \frac{n}{l} \Sigma W_r$ and $\tilde{U} = \sqrt{\frac{l}{n}} C U_{W_r} \Sigma_{W_r}^{-1} W_r$, where $W_r = U_{W_r} \Sigma_{W_r} U_{W_r}^\top$.

Performance Guarantees

We now describe the Nyström $r$-rank approximation for any symmetric positive semidefinite (SPSD) matrix $L \in \mathbb{R}^{n \times n}$. After performing sampling, create matrix $C \in \mathbb{R}^{n \times l}$ from the sampled columns. Then, form matrix $W \in \mathbb{R}^{l \times l}$ matrix consisting of the intersection of these $l$ columns with the corresponding $l$ rows of $L$. Let $W = U \Sigma U^\top$, where $U$ is orthogonal and $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_l)$ is a real diagonal matrix with the diagonal sorted in decreasing order. Let $W_r^+$ be the pseudo-inverse of the best rank-$r$ approximation to $W$ ($W_r^+ = \sum_{t=1}^r \sigma_t^{-1} U(t) U(t)^\top$, where $U(t)$ and $U(t)$ are respectively the $t$th column and row of $U$). Then the Nyström approximation $\tilde{L}$ of $L$ can be obtained as follows: $\tilde{L} = CW_r^+ C^\top$. Furthermore if we represent $\tilde{L}$ as $\tilde{L} = \tilde{U} \tilde{\Sigma} \tilde{U}^\top$ then $\tilde{\Sigma} = \frac{n}{l} \Sigma W_r$ and $\tilde{U} = \sqrt{\frac{l}{n}} C U_{W_r} \Sigma_{W_r}^{-1} W_r$, where $W_r = U_{W_r} \Sigma_{W_r} U_{W_r}^\top$. Theorem 2.3.1.1 due to [23] shows the performance bounds for the Nyström method when used with uniform sampling without replacement. In [23] the authors also compare the quality of obtained Nyström approximations, on the experiments with large-scale datasets, when using uniform and non-uniform sampling strategies (they consider both sampling with and without replacement). They consider two most popular
non-uniform sampling techniques: column-norm sampling and diagonal sampling. They show that uniform sampling without replacement is not only more efficient both in time and space but also improves the accuracy of the Nyström method.

**Theorem 2.3.1.1** ([23]). Let $G \in \mathbb{R}^{n \times n}$ be an SPSD matrix. Assume that $l$ columns of $G$ are sampled uniformly at random without replacement, let $\tilde{G}_r$ be the rank-$r$ Nyström approximation to $G$ and let $G_r$ be the best rank-$r$ approximation to $G$. Let $\epsilon > 0$, $l \geq 64r/\epsilon^4$ and $\eta = \sqrt{\log(2/\delta)\xi(\tilde{l},n-l)}$, where $\xi(m,u) = \frac{mu}{m+n-1/2} \frac{1}{(2 \max\{m,u\})^{1/2}}$. Then with probability at least $1 - \delta$,

$$
\|G - \tilde{G}_r\|_F \leq \|G - G_r\|_F + \epsilon \left( \frac{n}{l} \sum_{i \in D(l)} G_{ii} \right) \sqrt{\frac{1}{n} \sum_{i=1}^{n} \frac{n}{\sum_{i=1}^{n} G_{ii}^2} + \eta \max(nG_{ii})}^{1/2},
$$

where $\| \cdot \|_F$ is the Frobenius norm, $\sum_{i \in D(l)} G_{ii}$ is the sum of the largest $l$ diagonal entries of $G$.

### 2.3.2 Weighted Kernel K-means

The major disadvantage of the Nyström-based methods is, if $l$ columns are sampled, they involve the computation of eigenvectors of a sub-matrix of size $l \times l$, which can itself be computationally expensive when the $l$ is large. Also, the subsequent $k$-means step used for rounding the cluster assignment has to be performed for all the input points.

An alternate approach that did not require the computation of eigenvectors was described in [15]. Their work showed the connection between spectral clustering and weighted kernel $k$-means. The algorithm for weighted kernel $k$-means is described in Algorithm 3. Since no sampling is performed, the approach requires the computation of the entire affinity matrix, making it slower than the Nyström approximation. However, since it uses the entire affinity matrix, it is also more accurate. Additionally, it does not involve the computation of eigenvectors, making it much easier to scale to large datasets.
Algorithm 3 Weighted Kernel \(k\)-means

**Input:** Affinity Matrix \(A\), number of clusters \(k\)

**Output:** Matrix \(Y\) (\(Y_{ij} \neq 0\) only if point \(i\) belongs to cluster \(j\))

**Procedure:**

\[ W \leftarrow \text{degree}_\text{matrix}(A) \]
\[ \sigma \leftarrow k/(n - k) \]
\[ K \leftarrow \sigma W^{-1} + W^{-1}AW^{-1} \]
\[ \{m_c\}_{c=1}^k \leftarrow k\text{means}++ \]

**while** not converged **do**

- For each point \(a_i\) and every cluster \(c\), compute

\[
d(a_i, m_c) \leftarrow K_{ii} - \frac{2}{\sum_{a_j \in \pi_c} w_j K_{ij}} + \frac{\sum_{a_j, a_l \in \pi_c} w_j w_l K_{jl}}{\left( \sum_{a_j \in \pi_c} w_j \right)^2}
\]

- For each point \(a_i\),

\[
j \leftarrow \arg\min_c d(a_i, m_c) \]
\[
\pi_j \leftarrow \pi_j \cup a_i
\]

**end while**

### 2.3.3 Graph Sparsification

An alternate approach to speed up spectral clustering is to sparsify the input matrix using techniques similar to [44]. The sparsification step is motivated by the availability of efficient procedures to compute eigenvectors of sparse matrices. These result in approximations that are tighter than the Nyström methods. However, this suffers from two drawbacks. First, the complete affinity matrix still needs to be computed. Second, the eigenvectors of the sparse matrix still need to be computed, which might still be computationally intensive for large matrices. Hence these methods are not considered further in this thesis.
Nyström-based Spectral Clustering For Sparse Matrices

3.1 Introduction

In this chapter, we show that the traditional Nyström method requires a prohibitively large number of samples to obtain a good approximation when the input matrix is sparse. We propose a novel sampling approach to select the landmark points used to compute the Nyström approximation. We show that the modified sampling approach obeys the same error bound as in [7]. To control sample complexity, we use a selective densification step based on breadth-first traversal. We show that densification does not change the optimal clustering. Results on real world datasets show that by combining the modified sampling and densification schemes, we can obtain better accuracy compared to other techniques used for the Nyström method while using significantly fewer samples.

3.2 Observations For Sparse Matrices

In this section we outline two major issues that are faced by sampling based approaches when the matrix is extremely sparse. First we show that distances between columns are not useful for selecting landmark points when the matrix is sparse. Second, we show that the expected number of samples required for uniform sampling can be $O(n)$ when the matrix is sparse. Let the average degree of a node be $d_{avg}$ and the maximum degree of any node be $d_{max}$. For sparse matrices, we assume that $d_{max}^2$ is extremely small compared to the number of vertices $n$.

3.2.1 Distances Between Sparse Columns

Observation 3.2.1.1. Suppose we are given a sparse, symmetric positive semi definite matrix $A$ of size $n \times n$. For any pair of columns of $A$, say $x, y$, with probability $\geq 1 - d_{max}^2/n$,

$$
\|x - y\|^2 = \|x\|^2 + \|y\|^2
$$

Proof. For a column $x$, we call column $y$ favorable if there exists a row $i$ such that $x_i \neq 0$ and $y_i \neq 0$. $\|x - y\|^2 \neq \|x\|^2 + \|y\|^2$ only if $y$ is favorable with respect to $x$. For $y$ to be
favorable with respect to $x$, $x$ and $y$ must have at least one common neighbor. We make two observations: $x$ has at most $d_{\text{max}}$ neighbors and each neighbor can have at most $d_{\text{max}}$ neighbors. Thus, there are at most $d_{\text{max}}^2$ ways to choose a favorable $y$. This implies,

$$\text{Pr}[\text{choosing a favorable } y] \leq \frac{d_{\text{max}}^2}{n}$$

$$\text{Pr}[\|x - y\|^2 \neq \|x\|^2 + \|y\|^2] \leq \frac{d_{\text{max}}^2}{n}$$

Since $\text{Pr}[\|x - y\|^2 = \|x\|^2 + \|y\|^2] = 1 - \text{Pr}[\|x - y\|^2 \neq \|x\|^2 + \|y\|^2]$, the statement in Observation 3.2.1.1 now follows.

Consequence

In [55], the k-means algorithm is used to select the landmark points used to compute the Nyström approximation. The k-means algorithm has three major steps: initialization, cluster assignment and updating the center. In the initialization step, candidate centers are chosen by sampling $k$ columns uniformly at random. Among these centers $\{c_1, c_2, ..., c_k\}$, let $c_{\text{min}}$ be the center with the smallest column norm.

In the cluster assignment step, all the columns are assigned to their closest center. For a column $x$, let its closest center be represented as $c_x$, i.e.

$$c_x = \arg \min_c \|x - c\|^2$$

From Observation 3.2.1.1 this can be rewritten as,

$$c_x = \arg \min_c \|x\|^2 + \|c\|^2$$

This shows, points will be assigned to $c_{\text{min}}$ with high probability. This leads to bad landmarks being chosen, resulting in a poor approximation. Note, other techniques are available to select the initial candidate centers. However, these techniques are computationally intensive. Thus the k-means based landmark selection is not useful when the matrix is sparse.
3.2.2 Sample Complexity

Let $C$ be the sub-matrix created by sampling $l$ columns of the input matrix $A$. A node in the graph is said to be covered if its corresponding row in $C$ has norm greater than zero. All nodes that are not covered are mapped to the origin. This results in a bad clustering and has to be avoided. Consider a matrix $A$ with $n$ rows and columns.

Observation 3.2.2.1. When uniform sampling is used to select the landmark points, to ensure any node $x$ is covered, the number of samples $l$, satisfies,

$$E[l] \geq n/(d_{\text{max}} + 1)$$

Proof. $x$ is covered only if $x$ or one of its neighbors is sampled. Since there are at most $d_{\text{max}}$ neighbors, the probability of choosing $x$ or one of its neighbors is at most $(d_{\text{max}} + 1)/n$. Thus the number of samples needed $l$ is a geometric random variable with the probability of success being $(d_{\text{max}} + 1)/n$. Thus we have $E[l] \geq n/(d_{\text{max}} + 1)$ which is significantly large if $d_{\text{max}} \ll n$.

In this chapter, we address this issue by selectively densifying the sampled matrix $C$ as described in the next section. Selective densification increases both $d_{\text{avg}}$ and $d_{\text{max}}$. This enables sampling based approaches to cover the vertices of the graph with a smaller number of samples. Since the computational complexity of the Nyström approach scales cubically with the number of samples, using fewer samples results in a significant speedup for large datasets.

3.3 Proposed Approach

The proposed approach is outlined in Algorithm 4. It can be divided into three phases. Let $0 \leq \alpha \leq 1$ be user-specified constant. In the first phase, we sample a subset of $\alpha l$ samples as follows: choose a set of $\alpha l$ initial columns uniformly at random. Additional columns are sampled uniformly at random and only the $\alpha l$ columns with the highest column norm are retained. Let us call this subset $S_l$ and the associated sub-matrix $C$ (i.e. $C = A(:,S_l)$).
In this chapter, we use an iterative approach to selecting the columns, but it can be easily parallelized to obtain further improvement in efficiency.

In the second phase, \( C \) is densified as follows: For each column \( c \in S_t \), at step \( i \), the function \texttt{can\_be\_reached()} returns the set of nodes that can be reached from \( c \) in \( i \) steps. This is represented in the Algorithm as \( nn \) and the predecessor for each node in \( nn \) is returned in \( prev \). For each node \( p \in nn \) (let \( q \) be its corresponding node in \( prev \)) and column \( s \), if \( C(p, s) = 0 \), the new densified value is given as, \( C(p, s) = C(q, s) \times A(prev, p) \).

Finally, in the third phase, a pass is made over all the columns to cover the columns that are not covered even after the densification step.

Once the columns have been selected, the eigenvectors are approximated using the approach outlined in [27]. We use the procedure described in [27] due to its ability to handle matrices that are not positive semi-definite. In case the input matrix is positive semi-definite, we can use the simpler procedure described in [9].

3.4 Analysis

3.4.1 Justification for the initial sampling scheme

We briefly outline the MSSS algorithm to facilitate comparison with the proposed approach. The MSSS algorithm proceeds iteratively by selecting the column that has minimum similarity with the centers chosen so far. In other words, at each iteration, the method chooses the column that is farthest from the landmark points chosen so far.

At the end of \( m \) iterations, the proposed approach retains \( t = \alpha l \) columns which have the highest column norms.

**Lemma 3.4.1.1.** Suppose, at iteration \( m \), columns \( x_1, x_2...x_t \) have been chosen so far. Suppose, the similarity between columns is given by \( \text{sim}(x, y) = e^{-0.5|x-y|^2} \). In the sparse case, with probability \( \geq 1 - d_{\text{max}}^2/n \)

\[
\arg\max_y \|y\|^2 = \arg\min_y \sum \text{sim}^2(y, x_i)
\]

In other words, selecting the column with the highest column norm is equivalent to selecting
**Algorithm 4** Proposed Modified Spectral Clustering

**Input**: Matrix $A$, the required number of clusters $k$, number of samples $l$, fraction of points $\alpha$, maximum number of densification steps $max_{\text{hops}}$

**Output**: Cluster assignment $C$

**Procedure**:

1. **Select initial landmark points**
   - $S_0 \leftarrow$ Sample $\alpha l$ columns uniformly at random
   - $q \leftarrow \text{argmin}_{p \in S_0} \text{deg}(p)$
   - $t \leftarrow 0$

2. **while** $t < (1 - \alpha) l$ **do**
   - Sample a row $c_t$ uniformly at random
   - **if** $\|c_t\| > \|q\|$ **then**
     - $S_t \leftarrow S_{t-1} - \{q\} \cup \{c_t\}$
     - $q \leftarrow \text{argmin}_{p \in S_t} \|p\|$
   - **end if**
   - $t \leftarrow t + 1$

3. **end while**

4. **Densify selected columns**
   - $C \leftarrow A(S_t,:)$
   - for each $s \in S_t$ do
     - for $i < max_{\text{hops}}$ do
       - $[nn, prev] \leftarrow \text{can\_be\_reached}(s, i)$
       - $C(nn, s) \leftarrow C(prev, s) * A(prev, nn)$
     - **end for**
   - **end for**

5. **Compute Approximate eigenvectors**
   - $W \leftarrow C(S_t,:)$
   - $D_W \leftarrow \text{degree}(W)$
   - $D \leftarrow \text{degree}(C)$
   - $[U, \Lambda] \leftarrow \text{eig}(W)$
   - $Q \leftarrow D^{-1/2} C D_W^{-1/2} U \Lambda^+$
   - $C \leftarrow \text{discretize}(Q)$
the column with the lowest similarity with all the columns chosen so far.

Proof. Since \( \text{sim}(x, y) = e^{-0.5\|x-y\|^2} \), we can apply Observation 3.2.1.1 to the exponent of the expression and state that, with probability \( \geq 1 - \frac{d_{\text{max}}^2}{n} \),

\[
\text{sim}(x, y) = e^{-0.5\|x\|^2 - \|y\|^2} = e^{-0.5\|x\|^2} e^{-0.5\|y\|^2} 
\]

Thus,

\[
\sum \text{sim}^2(y, x_i) = \sum (e^{-0.5\|y-x_i\|^2})^2 
= \sum (e^{-0.5\|y\|^2} e^{-0.5\|x_i\|^2})^2 
= (e^{-0.5\|y\|^2})^2 \sum (e^{-0.5\|x_i\|^2})^2 
\]

Hence,

\[
\arg \min_y \sum \text{sim}^2(y, x_i) = \arg \min_y e^{-0.5\|y\|^2} 
= \arg \max_y \|y\|^2 
\]

At iteration \( m \), the proposed method retains \( t \) columns that have the maximum column norms over the set of columns sampled till iteration \( m \). Thus we see that at each iteration, with high probability, the proposed sampling step selects the column that has the least similarity to the columns chosen so far. Using the following theorem from [7] we can also say this increases the determinant at each step.

**Theorem 3.4.1.1.** [7] Suppose columns \( X = \{x_1, x_2...x_m\} \) have been chosen so far. Let \( A_X \) be the Nyström approximation obtained by selecting the columns of \( X \). Then for any pair of columns \( p, q \), if,

\[
\sum \text{sim}^2(p, x_i) \leq \sum \text{sim}^2(q, x_i) 
\]

then with high probability,

\[
\det(A_{X \cup p}) \geq \det(A_{X \cup q}) 
\]
This result can be used to prove that the Frobenius norm error of proposed sampling scheme has the same upper bound as MSSS. The upper bound on the Frobenius norm error is given in the following theorem.

**Theorem 3.4.1.2.** Suppose columns $X = \{x_1, x_2, ..., x_l\}$ have been chosen so far. Let $A_k$ be the optimal rank-$k$ approximation and $\hat{A}_k$ be the rank-$k$ Nyström approximation obtained by using the proposed method.

$$
\|A - \hat{A}_k\| \leq \|A - A_k\| + (l + 1) \sum_{i=l+1}^{n} \lambda_i + \gamma \left(1 + \sqrt{\frac{\theta d_S}{A_{\text{max}}}}\right)^{1/2}
$$

Here $d_S = \max_{ij}(A_{ii} + 2A_{jj}A_{ij})$. For more details on the error bound, please refer to Theorem 2 in [7].

If the error bounds are similar, why should the proposed sampling procedure be used? The proposed sampling scheme eliminates the need to compare a candidate column with all columns chosen so far. This improves efficiency especially when a large number of columns need to be sampled. It also allows the selection of columns in parallel making it easier to scale to larger datasets while providing the same level of accuracy.

### 3.4.2 Why Matrix norms are insufficient for spectral clustering

The Nyström approximation has been extensively studied with respect to various matrix error norms, such as the Frobenius norm, trace norm and the spectral norm. In this paper we show that approximation of these norms is not sufficient for producing a good clustering. For simplicity, we focus on the trace norm error. However the examples can be extended to any matrix norm.

**Not Sufficient**

Here we provide an approach to construct two matrices that are extremely similar in terms of their trace norms, but have a significantly different norm cut objective.

In other words, we show that for any value of $\epsilon > 0$, there exists $n$, $k$, $A$ and $B$ such that, $A$ and $B$ represent graphs $G_A, G_B$ with $n$ vertices, and $\|A - B\| \leq (1 + \epsilon)\|A\|_1$ and
$NCut(A, k) = O(k) \ast NCut(B, k)$. Note, that the maximum value of $NCut(A, k)$ is $k$.

Let $A$ and $B$ be the affinity matrices for undirected, unweighted graphs $G_A, G_B$ with $n$ vertices and $k$ and $k - 1$ equal sized components respectively. Further, we assume that each component is a clique. It can be shown that the normalized symmetric laplacian of $A$ (denoted by $L(A)$) has eigenvalues $n/(n-k)$ with multiplicity $n - k$ and 0 with multiplicity $k$. Similarly, $L(B)$ has eigenvalues $n/(n - k + 1)$ with multiplicity $n - k + 1$ and 0 with multiplicity $k - 1$. More generally, we can state that

**Lemma 3.4.2.1.** Let $A, D$ be the affinity and degree matrix of a graph, with $n$ vertices and $k$ equal sized components, Then the eigenvalues of the normalized symmetric laplacian $L(A) = I - D^{-0.5}AD^{-0.5}$ are $n/(n-k)$ with multiplicity $n - k$ and 0 with multiplicity $k$.

**Proof.** We assume that there are no self-loops in graphs $A$ and $B$. The result being used here is similar to the proof of Proposition 1.3 in [8]. Specifically, each component in $A$ is a complete graph with $n/k$ vertices. Thus its eigenvalues are -1 with multiplicity $(n/k) - 1$ and $(n/k) - 1$ with multiplicity 1. Since each component in $A$ is regular, the eigenvalues of $L(A)$ are given as, $\lambda_i(L(A)) = 1 - (1/\text{deg})(\lambda_i(A))$ where $\text{deg} = (n/k) - 1$.

We can use Lemma 3.4.2.1 to compare the trace norm of the difference of the two Laplacians, $L(A)$ and $L(B)$, as follows,

$$
\|L(A)\|_1 - \|L(B)\|_1 = \sum_{i=1}^{n-k} \lambda_i(A) - \lambda_i(B)
$$

$$
= \sum_{i=1}^{n-k} \left( \frac{n}{n-k} - \frac{n}{n-k'} \right) + \frac{n}{n-k'}
$$

$$
= (n-k) \left[ \frac{n}{n-k} - \frac{n}{n-k'} \right] + \frac{n}{n-k'}
$$

$$
= (n-k) \left[ \frac{n}{(n-k)(n-k')} \right] + \frac{n}{n-k'}
$$

$$
= \frac{2n}{n-k+1}
$$

where, $k' = k - 1$. Given a value of $\epsilon$, we can choose appropriate values of $n, k$ such that $\|L(A)\|_1 - L(B)\|_1 \leq (1+\epsilon)\|L(A)\|_1$. Now we examine the norm cuts ratio when we try to partition these graphs into exactly $k$ clusters. It is easy to see that the $NCut(A) = 0$, since
the graph has exactly \( k \) components. However for \( B \), one of the \( k - 1 \) components will have to be split into two equal parts to minimize the norm cuts ratio. This results in a cut that involves \( \frac{n}{k-1} \left( \frac{n}{k-1} + 1 \right) / 2 \) edges. Thus the norm cuts ratio for \( B \) is given as,

\[
NCut(B) = k - \frac{(n(n + 2k - 2)/(k + 1)^2)}{n(n + k - 1)/2(k + 1)^2} \\
= k - \frac{n + 2k - 2}{4(n + k - 1)}
\]

The second term reduces to a constant for a sufficiently large value of \( n \), resulting in a norm cuts ratio of \( O(k) \).

**Not Necessary**

To show that preserving matrix norms is not necessary for an approximation to be good with respect to the norm cuts objective, we consider the case of a block diagonal matrix \( A \) with \( k \) blocks. Let \( L(A) \) be the corresponding normalized Laplacian. We are interested in the eigenvectors of \( L(A) \), which are the same as the eigenvectors of \( L(A)^2, L(A)^3, \) etc. However \( \|L(A)\|_1 - \|L(A)^i\|_1 \) can be arbitrarily high.

### 3.4.3 Justification for the densification step

Now we proceed to provide approximation guarantees for the proposed densification scheme with respect to the norm cuts objective. Specifically, we state that

**Lemma 3.4.3.1.** If the affinity matrix is block diagonal, the clustering induced by the affinity matrix does not change after densification.

**Proof.** We use the connection to the weighted kernel k-means objective shown in [15] which showed that spectral clustering using the norm cuts objective is equivalent to weighted kernel k-means. Specifically, given an affinity matrix \( A \) and its associated degree matrix \( D \), number of clusters \( k \), minimizing the norm cuts objective was shown to be equivalent to weighted kernel k-means problem with kernel matrix \( K = D^{-1}AD^{-1} \) and weight matrix \( W = D \).
Thus,
\[ w_j = d_j \]
is the weighted degree of vertex \( j \) and
\[ K_{ij} = A_{ij} / (d_i \ast d_j) \]

Suppose \( m_c \) is the center of cluster \( \pi_c \). They show that the distance of any point to the center of cluster \( \pi_c \) is given as
\[
\| \phi(a_i) - m_c \|^2 = K_{ii} - 2 \frac{\sum_{a_j \in \pi_c} w_j K_{ij}}{\left( \sum_{a_j \in \pi_c} w_j \right)^2} + \frac{\sum_{a_j, a_l \in \pi_c} w_j w_l K_{jl}}{\left( \sum_{a_j \in \pi_c} w_j \right)^2}
\]

Plugging these values in the expression for \( \| \phi(a_i) - m_c \|^2 \), we get,
\[
\| \phi(a_i) - m_c \|^2 = K_{ii} - 2 \frac{\sum_{a_j \in \pi_c} A_{ij} d_i}{\sum_{a_j \in \pi_c} d_j \sum_{a_j \in \pi_c} d_j} + \frac{\sum_{a_j, a_l \in \pi_c} A_{jl}}{\left( \sum_{a_j \in \pi_c} d_j \right)^2}
\]

Since \( A \) is block diagonal, no entries of column \( i \) are modified except those corresponding to vertices in \( \pi_1 \). Thus we have,
\[
\sum_{a_j \in \pi_c} A_{ij} = d_i
\]
and
\[
\sum_{a_j, a_l \in \pi_c} A_{jl} = \sum_{a_j \in \pi_c} d_j
\]

Thus we get,
\[
\| \phi(a_i) - m_c \|^2 = K_{ii} - \frac{1}{\sum_{a_j \in \pi_c} d_j}
\]

Suppose column \( i \) belonging to cluster \( \pi_c \) is densified. Since \( a_i \) belongs to \( \pi_c \), \( \| \phi(a_i) - m_c \|^2 \) is smaller than the distance to any other center. Let the new affinity matrix after
densification be $A'$. The distance of the point $a_i$ from its center is given by,

$$\|\phi'(a_i) - m_c\|^2 = K'_{ii} - \frac{1}{\sum_{a_j \in \pi_c} d_j'}$$

$$\|\phi'(a_i) - m_c\|^2 - \|\phi(a_i) - m_c\|^2$$

$$= K'_{ii} - \frac{1}{\sum_{a_j \in \pi_c} d_j'} - K_{ii} + \frac{1}{\sum_{a_j \in \pi_c} d_j}$$

$$\leq - \frac{1}{\sum_{a_j \in \pi_c} d_j'} + \frac{1}{\sum_{a_j \in \pi_c} d_j}$$

$$\leq - \frac{1}{\sum_{a_j \in \pi_c} d_j} + \frac{1}{\sum_{a_j \in \pi_c} d_j}$$

$$\leq 0 \quad (3.1)$$

Equation 3.1 follows from the fact that $K_{ii} \geq K'_{ii}$. Equation 3.2 follows from the observation that $\sum_{a_j \in \pi_c} d_j' \geq \sum_{a_j \in \pi_c} d_j$. This holds because, the densification only adds positive entries to $A$.

Thus we see that after densification, the point moves closer to its own center, while its distance to other clusters remains unchanged. This ensures that the clustering remains unchanged.

\[\square\]

### 3.5 Experimental Results

#### 3.5.1 Matrix Norm Errors

For the sake of completeness, we present the errors with respect to various matrix norms. For all our experiments, we used the experimental framework in [19]. The following errors were used: $\|A\|_2 = \|Diag(\Sigma)\|_\infty$ denotes the spectral norm of $A$; $\|A\|_F = \|Diag(\Sigma)\|_2$ denotes the Frobenius norm of $A$; $\|A\|_* = \|Diag(\Sigma)\|_1$ is the trace norm of $A$.

The HEP, GR datasets were obtained from [26]. The datasets are extremely sparse in
Figure 3.1: Comparison of errors for the modified algorithm (modified), Nyström with uniform sampling (unif), Subsampled Random Fourier Transform (srft), Gaussian Projections (gauss) for different datasets. We restrict the rank of these datasets to 20. The plots show that the modified method performs better than uniform sampling for all the errors.

terms of their non-zero entries. In addition, it has been noted that their spectra decays slowly. We restrict the rank of the Laplacian for each dataset to 20. In other words, the low-rank approximation is 'filtered' through a space of rank 20.

Figure 3.1 shows a comparison between the various matrix norm errors, for the modified algorithm (called “modified”) and Nyström using uniform sampling (unif), Subsampled Random Fourier Transforms (srft) and Gaussian projections (gauss). The result shows that the modified method yields a better approximation than srft and gauss for all the errors considered. Even though uniform sampling has lower matrix norm error compared to the
Figure 3.2: Comparison of running time: Modified Nyström (modified) v/s Nyström with Subsampled Random Fourier Transform (sift), Gaussian Projections (gauss) for different datasets. We restrict the rank of these datasets to 20. The plots show that the modified method is computationally better than both gauss and sift.

Figure 3.3: Comparison of columns with norm zero: Modified Nyström (modified) v/s Nyström with Subsampled Random Fourier Transform (sift), Gaussian Projections (gauss) for different datasets. The plots show that the modified method covers all the columns with significantly fewer samples. In contrast, uniform sampling fails to cover all the columns even after sampling 20% of the columns.
other methods, it is important to note that a significantly large number of rows and columns in the resulting approximation had norm zero. This is shown in Figure [3.3]. In contrast, due to the use of the densification and clean up steps, the modified approach significantly reduces the number of uncovered columns.

Figure [3.2] shows a comparison of the computation time for the various methods. This shows that as the number of samples being considered increases, the modified method requires significantly lower time compared to Nyström with srft and gauss making it better suited for large datasets.

### 3.5.2 Spectral Clustering

**Synthetic Data**

We compared the performance of the modified approach to the results of approaches described in [15] and [53]. Both of these methods use the entire affinity matrix and do not perform any sampling. We evaluated the norm cuts ratio and the computation time on six commonly used synthetic datasets [7], described in Table [4.1] and repeated our evaluations 10 times. The modified approach sampled 30% of the points. We measured the clustering quality of each algorithm using the average accuracy across different datasets. The results are shown in Figure [4.1]. The modified approach, the weighted kernel $k$-means approach in [15] and the spectral method in [53] are denoted in Figure [4.1] as “proposed”, “kulis”, “shi” respectively. For all the datasets, the proposed approach results in comparable accuracy while resulting in a significant computational speedup.
Figure 3.4: The first and second rows show the clustering obtained by [53] and the modified approach respectively on several toy problems. The third and fourth rows show the norm cuts ratio and the time taken for each dataset. It can be seen that the modified approach yields a norm cuts ratio similar to [53], [15] while its computation time is comparable to using Nyström with uniform sampling.
Image Segmentation

One of the most popular applications of spectral clustering is image segmentation. In this section, we describe results obtained on an image segmentation benchmark [53]. The affinity matrix and the final discretization were computed using the approach of [53]. Since we had access to the function that was used to generate the affinity matrix, we used a simpler densification step. Namely,

\[ C_{ij} = \begin{cases} \text{similarity}(i, j) & \text{if } j = \arg \max_p \text{similarity}(i, p) \\ 0 & \text{otherwise} \end{cases} \]

The final segmentation was refined using the connected components algorithm in [4]. Experimental results in Table 4.2 show that the modified approach outperforms the traditional Nyström approximation while using significantly fewer samples. Since it essentially relies on computing the eigen values and eigen vectors of a significantly smaller matrix of size \( O(k) \), it also yields a significant speedup compared to using Nyström with uniform sampling.

3.6 Summary

When the input matrix is sparse, we showed that the traditional Nyström method requires a prohibitively large number of samples to obtain a good approximation. To control sample complexity, we propose a selective densification step based on breadth first traversal to ensure all nodes are covered. We show that the densification does not change the optimal clustering when the input matrix is block diagonal. Results on real world datasets show that the modified method outperforms other techniques used for the Nyström approximation.
Figure 3.5: Figures (a) and (c) show the norm cuts ratio, whereas (b), (d) show the time taken by different methods. The modified method outperforms Nyström with uniform sampling (nys) in terms of norm cuts ratio while taking a similar amount of time. The baseline methods shi [53] and kulis [15] use the entire affinity matrix. Thus they offer lower norm cuts ratio, and have a significantly higher computation time.
4.1 Introduction

The connection between spectral clustering and weighted kernel $k$-means was first introduced in [15]. In this chapter, we use this connection to propose a simple framework for spectral clustering that samples a subset $S$ of the input points, computes the clusters for $S$ using weighted kernel $k$-means and uses the resulting centers to compute a clustering for rest of the data points. The proposed framework does not require computation of the entire affinity matrix and does not require the computation of eigenvectors of the selected sub-matrix. Sampling algorithms developed for the Nyström approximation can be leveraged to obtain a tradeoff between accuracy and computational efficiency. Since the proposed method uses weighted kernel $k$-means to compute the clusters, it is easy to parallelize. The contributions of this chapter are:

- We propose a framework for spectral clustering that applies sampling followed by the weighted kernel $k$-means algorithm instead of approximating the affinity matrix or the Laplacian.

- Without making any assumptions about the data, we show that when points are sampled uniformly at random without replacement, the following theorem holds.

**Theorem 4.1.0.1.** Let $0 < \delta < 1$, $\alpha \geq 1$, $0 < \beta < 1$ and $\epsilon > 0$ be approximation parameters. Let $\mathcal{A}$ be an $\alpha$-approximation algorithm for the weighted kernel $k$-means problem. Let $V$ be a set of $n$ points. Suppose we sample a subset $S \subset V$ of size $s$ uniformly at random without replacement such that,

$$s \geq \ln \left( \frac{1}{\delta} \right) \left( 1 + \frac{1}{n} \right) \sqrt{\frac{2\beta^2 \epsilon^2}{\Delta^2 \alpha^2} + \frac{1}{n} \ln \left( \frac{1}{\delta} \right)}$$

where $\Delta = \max_{i,j} w_{ij} \| \phi(v_i) - \phi(v_j) \|^2$. If we run algorithm $\mathcal{A}$ with input $S$, then for the solution $C^*$ obtained, with probability at least $1 - \delta$,

$$\text{NCut}(G, C^*) \leq 4(\alpha + \beta)\text{NCut}(G, C_{\text{opt}}) + \epsilon$$
In other words, by choosing sufficient number of samples $s$, we can obtain centers that provide a good approximation to the optimal cluster centers. As the input size $n$ increases, the term $(1/n)$ vanishes, resulting in a bound for $s$ that is independent of $n$. To the best of our knowledge, the relation between the number of samples and the spectral clustering objective studied in the proposed work has not been explored in the literature.

4.2 Preliminaries

4.2.1 Weighted Kernel $k$-means

Given a set of $n$ points $V = \{v_1, v_2, ..., v_n\}$ with associated weights $\{w_1, w_2, ..., w_n\}$ and a kernel matrix $K$, the weighted kernel $k$-means algorithm seeks to find clusters $V_1, V_2, ..., V_k$ with centers $C = \{C_1, C_2, ..., C_k\}$ that minimize the objective function:

$$\mathcal{W}(V, C) = \sum_{i=1}^{k} \sum_{v_j \in V_i} w_j \|\phi(v_j) - C_i\|^2$$

where $C_i = \frac{\sum_{v_j \in V_i} w_j \phi(v_j)}{\sum_{v_j \in V_i} \|w_j\|}$. Here $\phi(v)$ is the kernel function that maps the vector $v$ to a higher dimensional feature space. It was shown in [15],

$$\|\phi(v_i) - C_j\|^2 = K_{ii} - \frac{2\tau_1(i, j)}{\text{deg}(V_j)} + \frac{\tau_2(V_j)}{(\text{deg}(V_j))^2}$$

where $\text{deg}(V_i) = \sum_{v_j \in V_i} w_j$, $\tau_1(i, j) = \sum_{v_l \in V_j} w_l K_{il}$ and $\tau_2(V_j) = \sum_{v_1, v_m \in V_j} w_l w_m K_{lm}$.

**Definition 4.2.1.1.** Let $A$ be an approximation algorithm for the weighted kernel $k$-means objective. Let $\alpha \geq 1$. $A$ is an $\alpha$-approximation algorithm if the set of centers, $C$, returned by $A$ satisfies,

$$\mathcal{W}(V, C) \leq \alpha \mathcal{W}(V, C_{opt})$$

where $C_{opt}$ is the set of optimal centers.
4.2.2 Spectral Clustering using Norm-Cuts

In spectral clustering, we are given a graph $G = (V, A)$, which is made up of a set of $n$ vertices $V$. The affinity matrix $A$ is $n \times n$ whose entries represent the similarity between vertices. If $V_1, V_2$ are subsets of $V$, let $\text{links}(V_1, V_2) = \sum_{i \in V_1, j \in V_2} A_{ij}$.

Furthermore, let $\text{degree}(V_1) = \text{links}(V_1, V)$. The graph partitioning problem seeks to partition the graph into $k$ disjoint clusters $V_1, ..., V_k$. A number of different graph partitioning objectives have been proposed and studied. In this chapter, we will focus on the normalized cut objective, though the results presented in this chapter can be generalized to other graph partitioning objectives as well. The goal is to minimize the following objective over all possible clusterings \{V_1..V_k\},

$$\text{NCut}(G, \{V_1..V_k\}) = \sum_{i=1}^{k} \frac{\text{links}(V_i, V \setminus V_i)}{\text{degree}(V_i)}$$

4.2.3 Relation between weighted kernel $k$-means and spectral clustering

To convert a spectral clustering problem to a weighted kernel $k$-means problem, it was shown in [15] that we can set $W = D$ and $K = \sigma D^{-1} + D^{-1}AD^{-1}$, where $\sigma = k/(n - k)$. Here the term $\sigma D^{-1}$ is added to ensure $K$ is positive definite and does not change the optimal clustering.

We note that a set of centers $C$ for the weighted kernel $k$-means problem induces a clustering $V_1...V_k$ for the spectral clustering problem. Since both objectives are equivalent, we use $\text{w}(V, C)$, $\text{NCut}(G, \{V_1..V_k\})$ and $\text{NCut}(G, C)$ interchangeably throughout the remainder of this chapter.

4.3 Modified Framework

The modified framework for spectral clustering is described in Algorithm 5. The overall procedure consists of three stages. In the first stage, a subset of the input points are sampled according to some distribution. This subset is indicated as $\text{sub}$. In the second stage, the clusters are computed using weighted kernel $k$-means. However, instead of using the entire
Algorithm 5 Framework for Spectral Clustering

Input: Affinity Matrix \(A\), number of clusters \(k\), number of samples \(s\)
Output: Matrix \(\hat{Y}\) (\(\hat{Y}_{ij} \neq 0\) only if point \(i\) belongs to cluster \(j\))

Procedure:

\(\text{sub} \leftarrow \text{sample}(s)\)
\(A_{\text{sub}} \leftarrow A(\text{sub}, \text{sub})\)
\(Y \leftarrow \text{weighted\_kernel\_kmeans}(A_{\text{sub}}, k)\)
\(\hat{Y} \leftarrow \text{diffuse}(Y, A, \text{sub})\)

affinity matrix, as in [15], we only use the principal sub-matrix induced by indices in \(\text{sub}\).

The final step involves using the cluster centers returned by the weighted kernel \(k\)-means procedure to assign clusters to the remaining data points. The procedure for this step is described in Algorithm 6.

This is scalable to larger datasets because:

- it does not require explicit computation of the eigenvectors
- it only requires the computation of a sub-matrix of size \(n \times k\), which can result in significant savings in memory.
- weighted kernel \(k\)-means is extremely easy to parallelize.

4.4 Analysis

In this section, we present the proof of Theorem 1. We extend the analysis in [14] to the case of sampling without replacement. Specifically, uniform sampling without replacement allows us to use the sharper bound offered by the Serfling inequality stated below. We refer the reader to [?] for a more thorough discussion.

Lemma 4.4.0.1. [?, ?] Let \(X_1, X_s\) be random variables sampled uniformly without replacement. Let \(X = X_1 + \ldots + X_n\) and \(0 \leq X_i \leq \Delta\). Then for all \(\gamma > 0\),

\[
P \left[ \sum_{t=1}^{s} \frac{X_t - E[X_t]}{s} \geq \gamma \right] \leq e^{-\frac{2\gamma^2}{1-(s-1)/n}\Delta^2}
\]

Definition 4.4.0.1. Let \(\beta > 0\) and \(\alpha \geq 1\). A set of \(k\) centers \(C\) is a \(\beta\)-bad, \(\alpha\)-approximation
Algorithm 6 Diffuse

**Input:** Matrix $\hat{Y}$, Affinity Matrix $A$, set of sampled indices $S$

**Output:** Matrix $Y$ ($Y_{ij} \neq 0$ only if point $i$ belongs to cluster $j$)

**Procedure:**

for each $i \in S$
    for $j = 1:k$
        $Y_{ij} \leftarrow \hat{Y}_{ij}$
        if $Y_{ij} \neq 0$
            $\pi_j \leftarrow \pi_j \cup \{i\}$
        end if
    end for
end for

for each $i \notin S$
    for $c = 1:k$
        $d(v_i, m_c) \leftarrow K_{ii} - \frac{2}{\sum_{v_j \in \pi_c} w_j K_{ij}} + \frac{\sum_{v_j, v_l \in \pi_c} w_j w_l K_{jl}}{\left(\sum_{v_j \in \pi_c} w_j\right)^2}$
    end for
    $j \leftarrow \arg\min_{c=1}^{k} d(v_i, m_c)$
    $Y_{ij} \leftarrow 1$
end for

if

$\mathcal{W}(V, C) > (\alpha + \beta)\mathcal{W}(V, C_{opt})$

If not, $C$ is said to be a $\beta$-good, $\alpha$-approximation.

We will need the following lemmas.

**Lemma 4.4.0.2.** Let $S$ be a subset of $V$ of size $s$ such that,

$$s \geq \ln \left(\frac{1}{\delta}\right) \left(1 + \frac{1}{n}\right) / \left(\frac{2\beta^2 m^2}{\Delta^2 \alpha^2} + \frac{\ln(1/\delta)}{n}\right)$$

If an $\alpha$-approximation algorithm for weighted kernel $k$-means $A$ is run on input $S$, then for the set of centers, $C_A$, obtained, with probability $\geq 1 - \delta$,

$$\mathcal{W}(S, C_A) \leq 4(\alpha + \beta) \frac{s}{n} \mathcal{W}(V, C_{opt})$$

where $C_{opt}$ is the set of optimal centers for $V$ and $m = \mathcal{W}(V, C_{opt})/n$. 

36
Proof. For each point \( v_i \in S \), let \( X_i = w_i \| \phi(v_i) - C_i \|_2 \) where \( C_i \) the nearest center to \( \phi(v_i) \) in \( C_{opt} \). Then \( \mathcal{W}(S, C_{opt}) = \sum_{i=1}^{s} X_i \). We also know that \( E[X_i] = m \). Let \( X_i \) be bounded in the interval \([0, \Delta]\).

\[
Pr[\mathcal{W}(S, C_{opt}) > (1 + \frac{\beta}{\alpha}) \frac{s}{n} \mathcal{W}(V, C_{opt})] = \Pr \left[ \sum_{i=1}^{s} X_i > (1 + \frac{\beta}{\alpha})sm \right] 
\leq e^{\frac{2s \Delta^2 m^2}{(1-s/n) \Delta^2}}

The last step is obtained by applying the Serfling inequality. Note, \( e^{\frac{2s \Delta^2 m^2}{(1-s/n) \Delta^2}} \leq \delta \) if

\[
s \geq \ln \left( \frac{1}{\delta} \right) \left( 1 + \frac{1}{n} \right) / \left( \frac{2\beta^2m^2}{\Delta^2 \alpha^2} + \frac{\ln(1/\delta)}{n} \right)
\]

Let \( C' \) be the set of \( k \) centers in \( S \) obtained by replacing each \( C \in C_{opt} \) by its nearest neighbor in \( S \). From the weakened triangle inequality, it can be shown that \( \mathcal{W}(S, C') \leq 4\mathcal{W}(S, C_{opt}) \). This implies with probability at least \( 1 - \delta \), \( S \) contains \( k \) centers with cost at most \( 4(1 + \frac{\beta}{\alpha}) \frac{s}{n} \mathcal{W}(V, C_{opt}) \). In other words,

\[
\mathcal{W}(S, C') \leq 4(1 + \frac{\beta}{\alpha}) \frac{s}{n} \mathcal{W}(V, C_{opt})
\]

If \( C'' \) is the set of optimum centers for \( \mathcal{W}(S, C) \), then

\[
\mathcal{W}(S, C'') \leq \mathcal{W}(S, C')
\]

Since \( A \) is an \( \alpha \)-approximation algorithm,

\[
\mathcal{W}(S, C_A) \leq \alpha \mathcal{W}(S, C'')
\]

This implies,

\[
\mathcal{W}(S, C_A) \leq \alpha \mathcal{W}(S, C')
\]

The lemma now follows. \( \square \)
Lemma 4.4.0.3. Let $S$ be a subset of $V$ of size $s$, sampled uniformly at random such that,

$$s \geq \ln \left( \frac{1}{\delta} \right) \left( 1 + \frac{1}{n} \right) / \left( \frac{\beta^2 m^2}{\Delta^2} + \frac{1}{n} \ln \left( \frac{1}{\delta} \right) \right)$$

Let $\chi$ be the set of $12\beta$-bad $4\alpha$-approximations of $\mathcal{W}(V, C_{opt})$ then,

$$\Pr[C_b \subset S \text{ and } C_b \in \chi \text{ and } \mathcal{W}(S, C_b) \leq 4(\alpha + \beta)m] \leq \delta$$

Proof. Let $s \geq \frac{4\alpha + 5\beta}{\beta}k$. Let $C_b \in \chi$ be a $12\beta$-bad $4\alpha$-approximation of $\mathcal{W}(V, C_{opt})$. If $C_b \subset S$, let $S^*$ be the subset of $S$ with $C_b$ removed, such that $|S^*| = s - k$. Thus,

$$\Pr[C_b \subset S \text{ and } \frac{\mathcal{W}(S, C_b)}{s} \leq 4(\alpha + \beta)m]$$

$$= \Pr \left[ \frac{\mathcal{W}(S, C_b)}{s} \leq 4(\alpha + \beta)m \bigg| C_b \subset S \right] \cdot \Pr[C_b \subset S]$$

$$\leq \Pr \left[ \frac{\mathcal{W}(S^*, C_b)}{s - k} \leq 4 \frac{s - k}{s} (\alpha + \beta)m \right] \cdot \Pr[C_b \subset S]$$

$$\leq \Pr \left[ \frac{\mathcal{W}(S^*, C_b)}{s - k} \leq (4\alpha + 5\beta)m \right] \cdot \Pr[C_b \subset S] \quad (4.1)$$

The second step holds because the mean remains unchanged at each step when the elements are chosen uniformly at random without replacement and $\mathcal{W}(S, C_b) = \mathcal{W}(S^*, C_b)$ when $C_b \subset S$. The third step holds since $s \geq \frac{4\alpha + 5\beta}{\beta}k$.

Let $Y_i$ denote the distance of point $v_i \in S$ from its closest center in $C_b$. Since $C_b \in \chi$, $E[Y_i] \geq (4\alpha + 12\beta)m$. Thus,

$$\Pr \left[ \frac{\mathcal{W}(S^*, C_b)}{s - k} \leq (4\alpha + 5\beta)m \right]$$

38
\[
Pr \left[ \sum_{i=1}^{s-k} Y_i \leq (4 \alpha + 5 \beta) m \right] \\
\leq Pr \left[ \sum_{i=1}^{s-k} Y_i \leq \frac{4 \alpha + 5 \beta}{4 \alpha + 12 \beta} E \left[ \sum_{i=1}^{s-k} Y_i \right] \right] \\
\leq Pr \left[ \sum_{i=1}^{s-k} Y_i \leq \left( 1 - \frac{7 \beta}{4 \alpha + 12 \beta} \right) E \left[ \sum_{i=1}^{s-k} Y_i \right] \right] \\
\leq Pr \left[ \sum_{i=1}^{s-k} Y_i - E[Y_i] \leq \left( -\frac{7 \beta}{4 \alpha + 12 \beta} \right) E \left[ \sum_{i=1}^{s-k} Y_i \right] \right] \\
\leq \exp \left( -2(s - k) \left( \frac{7 \beta E[\sum_{i=1}^{s-k} Y_i]}{4 \alpha + 12 \beta} \right)^2 \right) \\
\leq \exp \left( -2(s - k) \frac{49 \beta^2 m^2}{(1 - (s - 1)/n) \Delta^2} \right) \tag{4.2}
\]

Similar to the previous lemma, we have used the Serfling bound to obtain (2). Thus it follows that,

\[
Pr \left[ \frac{\mathcal{W}(S^*, C_b)}{s - k} \leq (4 \alpha + 5 \beta) m \right] \leq e^{-\frac{s \beta^2 m^2}{(1 - (s - 1)/n) \Delta^2}} \tag{4.3}
\]

We note, \( Pr[C_b \subset S] \leq (s/n)^k \) and \(|\chi| \leq n^k\). Using this and (3) in (1), we have,

\[
Pr[C_b \subset S \text{ and } C_b \in \chi \text{ and } \frac{\mathcal{W}(S, C_b)}{s} \leq 4(\alpha + \beta) m] \\
\leq \sum_{C_b \in \chi} Pr[C_b \subset S \text{ and } \frac{\mathcal{W}(S, C_b)}{s} \leq 4(\alpha + \beta) m] \\
\leq \sum_{C_b \in \chi} Pr \left[ \frac{\mathcal{W}(S^*, C_b)}{s - k} \leq (4 \alpha + 5 \beta) m \right] \cdot (s/n)^k \\
\leq n^k \cdot e^{-\frac{s \beta^2 m^2}{(1 - (s - 1)/n) \Delta^2}} \cdot (s/n)^k \\
\leq s^k \cdot e^{-\frac{s \beta^2 m^2}{(1 - (s - 1)/n) \Delta^2}}
\]

which is upper bounded by \( \delta \) when,

\[
s \geq \ln \left( \frac{1}{\delta} \right) \left( 1 + \frac{1}{n} \right) / \left( \frac{\beta^2 m^2}{\Delta^2} + \frac{1}{n} \ln \left( \frac{1}{\delta} \right) \right)
\]
Next, we use these lemmas to prove Theorem 1.

**Proof.** Let \( \beta \) be some constant. Let \( S \subseteq V \) be a subset of \( s \) points chosen uniformly at random such that,

\[
s \geq \ln \left( \frac{1}{\delta} \right) \left( 1 + \frac{1}{n} \right) / \left( \frac{2\beta^2 m^2}{\Sigma^2 \alpha^2} + \frac{1}{n} \ln \left( \frac{1}{\delta} \right) \right)
\]

Then from Lemma 3, we have, if \( C \subseteq S \) and \( C \in \chi \), with probability at least \( 1 - \delta \),

\[
\mathbb{W}(S, C) > 4(\alpha + \beta)m
\]

On the other hand, if we run algorithm \( A \) for set \( S \), then by Lemma 2, the resulting set of centers satisfies, with probability at least \( 1 - \delta \),

\[
\mathbb{W}(S, C) \leq 4(\alpha + \beta)m
\]

This implies with probability at least \( 1 - 2\delta \) the set \( C \) is a \((12\beta)\)-good \((4\alpha)\)-approximation. In order to remove the dependence on the number of samples on \( m \),

- If \( m \leq \epsilon \), we simply set \( \beta = (1/\sqrt{2})\epsilon/m \). In this case we will obtain the following bound on \( s \).

\[
s \geq \ln \left( \frac{1}{\delta} \right) \left( 1 + \frac{1}{n} \right) / \left( \frac{\epsilon^2}{\Sigma^2 \alpha^2} + \frac{1}{n} \ln \left( \frac{1}{\delta} \right) \right)
\]

- If \( m > \epsilon \), we set \( \beta = \beta/3 \) to obtain the following upper bound for \( s \),

\[
s \geq \ln \left( \frac{1}{\delta} \right) \left( 1 + \frac{1}{n} \right) / \left( \frac{2\beta^2 \epsilon^2}{\Sigma^2 \alpha^2} + \frac{1}{n} \ln \left( \frac{1}{\delta} \right) \right)
\]

The theorem now follows. \( \square \)
<table>
<thead>
<tr>
<th>Name</th>
<th>Instances</th>
<th>Attributes</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation (Agg)</td>
<td>788</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>D31</td>
<td>3100</td>
<td>2</td>
<td>31</td>
</tr>
<tr>
<td>Flame (Fla)</td>
<td>240</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A.K's toy problem (AK)</td>
<td>373</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.1: The synthetic datasets used in our experiments.

Figure 4.1: The first and second rows show the norm cuts ratio and the time taken for each dataset. It can be seen that wkk-unif and wkk-col, which are variants of the proposed framework yield results similar to kulis and shi, which use the entire affinity matrix. wkk-unif and wkk-col also outperform nys-col and nys-unif in terms of accuracy and computation time.
4.5 Experiments

We now present experimental results to show that the use of sampling followed by weighted kernel $k$-means outperforms sampling followed by the Nyström approximation. We evaluated the performance of two variants of the framework presented in this chapter. The first variant uses uniform sampling without replacement to sample a subset in the input points (denoted as wkk-unif). The second variant samples points based on the norm of the corresponding column in the affinity matrix (denoted as wkk-col). These results were compared against approaches described in [15] (denoted as kulis) and [53] (denoted as shi). Both of these methods use the entire affinity matrix and do not perform any sampling. We also compare against the Nyström approximation when the columns are sampled uniformly at random (denoted as nys-unif) and sampled based on their column norm (denoted as nys-col).

Synthetic Data

We evaluated the norm cuts ratio and the computation time on six commonly used synthetic datasets [7], described in Table 4.1, and repeated our evaluations 10 times. We measured the clustering quality of each algorithm using the average accuracy across different datasets.

The results are shown in Figure 4.1. The first row shows the norm cuts ratio obtained by various approaches. It can be seen that norm cuts ratio for wkk-unif is comparable to kulis. This supports the result presented in Theorem 1. Furthermore, the second row in Figure 4.1 shows the time taken by each approach. This highlights the speedup obtained by using wkk-unif and wkk-col over shi and kulis. For all the datasets, the approaches based on combining sampling and the weighted kernel $k$-means algorithm outperform approaches based on the Nyström approximation both in terms of accuracy and computation time.

Image Segmentation

One of the most popular applications of spectral clustering is image segmentation. In this section, we describe results obtained on an image segmentation benchmark [53]. The affinity matrix and the final discretization were computed using the approach of [53].

Experimental results in Figure 4.2 show that wkk-unif and wkk-col obtain the same
Figure 4.2: Figure (a) shows the norm cuts ratio, whereas (b) shows the time taken by different methods. This shows that methods using weighted kernel $k$-means outperform their Nyström counterparts with respect to computation time, while obtaining a comparable norm cuts ratio.

In order to study how the modified approach scales to larger graphs, we vary the size of the input images. The results are shown in Figure 4.3. For shi, the time taken scales cubically with the size of the graph. Hence it is unsuitable for large graphs. In contrast, kulis, wkk-unif and wkk-col scale almost linearly with the size of the graph. Both wkk-unif and wkk-col outperform kulis in terms of execution time. They also outperform the Nyström based approaches both in terms of accuracy and computation time. These results show that uniform sampling followed by the weighted kernel $k$-means algorithm scales well in practice.
Figure 4.3: Behavior of the various approaches when the graph size increases. This shows that by using only 50% of the points, wkk-unif and wkk-col achieve accuracy comparable to kulis and shi while taking significantly less time. The difference in accuracy reduces as the size of the graph increases. wkk-unif and wkk-col also outperform nys-unif and nys-col both in terms of accuracy and computation time.

4.6 Summary

This chapter presented a framework for spectral clustering based on combining sampling and the weighted kernel \( k \)-means algorithm. For the case where the points are sampled uniformly at random without replacement, we show that the number of samples required, is independent of the input size and depends only on the number of clusters and the diameter of the set of points in the kernel space. Experiments show that approaches based on the proposed framework outperform approaches based on the Nyström approximation while achieving accuracy comparable to methods that use the entire affinity matrix.
5.1 Introduction

Many applications arise in different domains like bioinformatics, drug discovery, web data mining and social networks that deal with the question “How similar are two given graphs?” For example, [47] predicts the toxicity of a chemical molecule by modeling it as a graph that encodes its structure. A graph kernel is used to compute the similarity between molecules of known and unknown toxicity.

In this chapter, we focus on the problem of place recognition that arises in robotics. In place recognition, images are obtained from a camera mounted on an autonomous car. These images are often represented as a vector of visual “words”. The set of images seen so far represent the map of the environment, where each image represents a place. In place recognition, given a query image, we need to obtain the closest match from the set of previously seen images. If the images (represented as a vector of visual words) are considered as points in high dimensional space, a query can be formulated as a search for the nearest neighbor of the query image. As autonomous cars continue to create long-term maps, the number of images being considered may increase until exceeding the available computational resources, making a simple linear search computationally expensive. In this chapter we describe a novel hierarchical nearest neighbor data structure that uses graph kernels to speed up the query process.

5.2 Place Recognition

The problem of place recognition has received considerable attention in the past decade. In its most general form, the problem can be formulated as searching for a query image in a database of existing images. Consequently, a large number of techniques for describing and comparing images and for retrieving the best matches have been proposed in the literature [12, 31]. One such popular framework is the Bag Of Words framework introduced in [41]. Here descriptors are extracted from the images and quantized using a vocabulary or code book of visual words. A visual word is a cluster that results from some discretization of the descriptor space. This results in a term frequency (tf) vector based representation of the
given images. Queries can then be answered computing a similarity value using an Inverted Index, a structure that stores the list of images where each visual word appears.

In the context of place recognition using the Bag of Words framework, an increase in the number of images increases the query time, since the underlying system is essentially a linear search. This has motivated recent work that aggregates term frequency (tf) vectors of consecutive frames and constructs a tree based lookup approach; this results in a significant improvement in query time at only a small loss in precision and recall [28]. For example, MacTavish & Barfoot [28] build accumulated bag of words of consecutive images to match groups of images with a single inverted index. This paper follows a similar vein, but it differs from [28] in two important ways. First, we show that the increase in query time for a simple inverted index is directly addressable by maintaining multiple sparse inverted indices. In other words, as the number of images increases, the sparsity of a single large inverted index is reduced. This reduces the efficiency and accuracy of the query retrieval process as more candidates need to be examined. However, separating the set of places into subsets describing environments and building separate smaller inverted indices preserves the sparsity that we seek, thereby speeding up the querying process.

This decomposes the search problem into two parts: searching for the best subset and finding the best match within that subset. Since each subset is assumed to describe an environment, we call this problem environment selection. The second difference between [28] and this paper is the choice of the similarity criterion to select the best subset. At a conceptual level, environments can be treated as a probability distribution over all possible word sequences. MacTavish & Barfoot [28] use the first order statistics of this distribution, i.e. aggregated tf-vectors, for this purpose. In this chapter, we use the second order statistics of this distribution, which are well represented by the frequencies of co-occurrence of word pairs in the environment. This word-word co-occurrence (WWC) matrix representation of the environments motivates a novel similarity criterion that is much more effective for environment selection. This is important because the aggregated tf-vectors cease to be discriminative at larger temporal scales [16] [28], leading to significant loss in accuracy.

An overview of our approach is presented in Figure 5.1. Our proposal is based on considering subsets of images describing environments (represented by bags of binary words
Figure 5.1: Outline of the ideas proposed in the paper. (a) The figure on the left shows a potential query image being compared against a set of database images grouped into environments shown on the right. Different rows represent images obtained from different datasets. (b) The similarity scores computed using the proposed criterion to compare the query image to the existing environments. The plot of the scores shows that it is sufficiently discriminative to be used to determine the environment the image is most likely to be from. (c) Selecting the best environment helps reduce the number of images to be compared against, thereby improving the search time. We show the potential savings in computation time while querying a database of 85,000 images.
Each environment is described in terms of its co-occurrent words. Similarity between environments is established by comparing their co-occurrence matrices. Final image-to-image loop detection is computed by using the environment-specific database. The important contributions of this chapter are:

- We introduce a new problem called *Environment Selection* and propose a similarity criterion to determine similarity between image sets at the environment scale by comparing their corresponding word-word co-occurrence (WWC) matrices. We also provide its derivation from a direct product graph kernel. This is described in Section 5.4.

- A hierarchical approach to place recognition based on the proposed similarity criterion, as described in Section 5.5.

- Experimentally we verify the improvement in the query time on datasets that span a distance of more than 750Km.

- In addition we also provide an experimental analysis of the properties of the proposed similarity criterion.

### 5.3 Related Work

A graph kernel is a kernel, where the objects being compared $x$ and $x'$ are either two graphs or two nodes of a graph. They have been widely used for various tasks like classification, regression etc. In this section, we give a brief description of kernels between two graphs. (For a detailed review the reader is referred to [47] and [39]. It has been shown that the problem of determining the similarity between two graphs is as difficult as finding whether two graphs are isomorphic. Hence approximations have been proposed to compute this.

Graph kernels between graphs can be divided into three major categories. Graph kernels from the first category rely on the observation that similar graphs should produce a similar set of random walks. Thus, random walks are generated from both candidate graphs and the degree of similarity is determined by counting the number of common random walks [22], [18]. These kernels are based on the computation of the direct product graph of two
graphs. If the two graphs being compared contain \(n\) nodes each, the direct product graph takes \(O(n^6)\) time to compute. \[47\] showed that the complexity can be reduced to \(O(n^3)\) by restating the problem in terms of Kronecker products. A major advantage of these kernels is that they allow a trade-off between accuracy and computation time by limiting the length of the random walks being considered.

Methods belonging to the second category are based on comparison of limited size subgraphs. The problem of matching subgraphs is NP Complete for arbitrary sized sub-graphs. However, there exist efficient methods for certain categories of sub-graphs. For example, there exist efficient sampling based schemes for sub-graphs of size \(k \in \{3, 4, 5\}\) as shown in \[39\]. Another approach proposed by \[21\] is to compute the number of pairs of matching cyclical patterns in the two input graphs. Instead of considering pairs of matching cyclical patterns, we can also count identical pairs of rooted subgraphs containing nodes up to a certain distance from the root as shown in \[11\]. For large graphs, these methods become quickly infeasible.

The third category of graph kernels consists of methods that compare sub-trees of the given graphs.

Bags of words are a popular choice when the aim is long-term mapping and large scale place recognition \[28, 33, 35, 17, 13\]. This involves partitioning an image descriptor space given some training data to build a vocabulary whose words are the resulting cluster centers. Feature descriptors obtained from an image can be quantized to their nearest words using this vocabulary. Images can then be represented by a vector indicating the proportion of each word in the image. This together with an inverted index (or inverted file) allows the creation of a place database for efficient image retrieval.

FAB-MAP 2 \[13\] is one of the most notable works that makes use of words and an inverted index to find loops in a 1000 Km trajectory. The appearance information of the vocabulary is enriched with the spatial relationships of the words that co-occur in the training data. This is encoded in a Chow Liu tree to approximate the co-occurrence joint distribution and to reduce the memory requirements. They use a probabilistic formulation to compute the probability of a new frame being a revisited place or a new one. The matches that reach a fixed threshold are considered loop closures. In our work, we build co-occurrence graphs.
of words present in full environments to compare them online with the words that co-occur when querying the database.

The initial motivation for using co-occurrences to determine similarity came from [30] which leverages covisibility information to improve the accuracy of place recognition. The covisibility graph, being topometric in nature results in landmark-centric notion of places. While we do not harness the power of the covisibility graph in this work, we note that it inherently relies on the same contextual information that we seek. This is because, foreground elements (like cars and people) are less likely to be covisible with the same background features, resulting in weak edges in the covisibility graph.

The problem of establishing similarity between environments is closely related to scene categorization described in the computer vision literature. For example, in [16], the authors show that there are distinct stylistic elements that describe a city and can be used to distinguish it from other cities. This is achieved through the use of discriminative learning and iterative clustering of image patches obtained from various images of the environment. In [25] images are classified according to the depicted scene by augmenting visual words with spatial information. However, these works do not address the final problem of detecting a loop in a map. For mapping, the authors of [33] present an approach to incrementally discover topics that can describe and group together images collected in the same place. Then, to address scalability, the most distinctive areas are retained by clustering the topic space.

Scalability is one of the main issues to address in long-term mapping. A large collection of images affect not only to the computational requirements but also the accuracy of the system due to the confusion that additional data causes. An efficient management of the available memory can achieve real-time performance in large maps [24]. However, this requires disregarding observation of places, leading to miss matches in future operations. When considering all the processed places, hierarchical approaches tackle scalability by dividing the image collection into tractable groups, so that only the most promising ones are inspected. For example, the work in [29] approaches the problem in two steps: a global localization yields a subset of candidate images by matching hue histograms, and a final single-image match is obtained by comparing SIFT features. However, they access the
images sequentially, which is less efficient than using a data structure as an inverted index [41].

Similarly, the authors of [28] aim to achieve logarithmic complexity localization by grouping images together. They analyze the optimal group size to compute accumulative bag-of-words vectors and obtain matches between groups. Nevertheless, they do not compute an image-to-image match required to close a loop in a SLAM map.

5.4 Similarity Criterion

We begin by assuming that two visual words co-occur if they are visible in the same image and the distance (in pixels) between their corresponding key points in the image is less than a threshold. We define a word-word co-occurrence (WWC) matrix as a symmetric matrix \( W \) where entry \( W(i,j) \) contains the co-occurrence frequency of words \( w_i \) and \( w_j \).

Formally, suppose we are given two environments, \( E_1 \) and \( E_2 \), with WWC matrices \( W_{E_1} \) and \( W_{E_2} \) respectively and a vocabulary of cardinality \( |V| \). We define an environment as a sequence of images collected by an agent around some physical area.

In order to account for the various word-word inter-dependencies, which is a characteristic attribute of any environment, we consider the second order statistics of the distribution. In order to account for co-occurrence of a pair of words, we propose the following expression to compute the similarity:

\[
S(E_1, E_2) = \sum_{i,j=1}^{|V|} \min[W_{E_1}(i,j), W_{E_2}(i,j)].
\]  

(5.1)

Before we present the justification for this criterion, we briefly outline graph kernels that are used to compare two graphs.

5.4.1 Notation

In this work, we use the direct product graph kernel. In this kernel, random walks are generated from each candidate graph being compared and the degree of similarity is determined by counting the number of common random walks [22], [18].
Before we describe the direct product graph kernels, we specify the notation being used. Specifically, a graph $G$ consists of an ordered set of vertices $\zeta$ and a set of undirected edges $\Upsilon$. This is represented as a matrix where $\Upsilon_{ij} = 1$ if there is an edge between vertices $v_i$ and $v_j$, and 0 otherwise. Each edge $(v_i, v_j)$ is also associated with a weight $w_{ij}$. The adjacency matrix of the graph is represented as $A$ (such that $A_{ij} = w_{ij}$). $D$ is the diagonal matrix of node degrees, i.e. $D_{ii} = \sum_j A_{ij}$. The direct product graph $G \times = \{\zeta \times, \Upsilon \times\}$ of two graphs $G_1$ and $G_2$ is given by $\zeta_x = \zeta_1 \times \zeta_2$ and $\Upsilon_x = \Upsilon_1 \otimes \Upsilon_2$, where $\otimes$ denotes the Kronecker product of two matrices. Let $p_1$ and $p_2$ be the starting probabilities of the random walks on $G_1$ and $G_2$, respectively. The starting probabilities of the random walks on the product graph is given by $p_x = p_1 \otimes p_2$. The stopping probabilities $q_1, q_2$ and $q_x$ are defined similarly. The graph kernel can now be described as

$$K(G_1, G_2) = \sum_{i=1}^{\infty} \lambda_i q_x \Gamma_x^i p_x,$$

where $\lambda_i$ is a decay factor. Each entry in $\Gamma_x$ corresponds to a pair of edges ($(e^1 = v^1_i, v^1_j)$ from graph $G_1$ and $(e^2 = v^2_i, v^2_j)$ from graph $G_2$) such that,

$$\Gamma_x(i, j) = f(v^1_i, v^2_i, v^1_j, v^2_j, e^1, e^2).$$

Note that no assumptions have been made about the sizes of $G_1$ and $G_2$. The basic idea is that each walk in the direct product graph is equivalent to a joint walk in the two graphs $G_1$ and $G_2$. Thus the similarity of the two graphs can be determined using the weight of the walks in the direct product graph.

5.4.2 Theoretical Justification

In this section, we describe how the similarity criterion described earlier can be derived from the direct product graph kernel. In the context of place recognition using the bag-of-words approach, we assume that the environment is a stochastic process that generates a document based on an unknown underlying probability distribution over all the words in
the vocabulary, i.e.

\[ P(w_1, w_2, ..., w_{|V|}|E) \]

where \(|V|\) is the number of words in the vocabulary.

Now for two environments to be similar, we impose the restriction that their corresponding joint distributions should be similar.

However, we note that computing the similarity between two joint probability distributions is prohibitively expensive in practice. We also note that the results in [16] indicate that comparing first order marginals \(P(w_i|E_1)\) and \(P(w_i|E_2)\) is not sufficiently accurate. Hence in this work, we propose the use of second order marginal distributions, i.e \(P(w_i, w_j|E_1)\) and \(P(w_i, w_j|E_2)\) for comparing the similarity between two environments.

Comparing the second order marginal distributions is equivalent to comparing their corresponding graphical structures. To visualize this, consider a weighted undirected graph where the words \(w_i\) are the nodes and the weight of the edge between \(w_i\) and \(w_j\) represents \(P(w_i, w_j|E_1)\). We can construct a similar graph for environment \(E_2\). In practice, we do not have access to the underlying probability distributions. However we can determine the empirical distribution using the observed co-occurrence frequencies. The WWC matrices \(W_{E_1}\) and \(W_{E_2}\) are specific instances of these probability distributions. Now the problem of determining the similarity between the two second order marginal distributions \(P(w_i, w_j|E_1)\) and \(P(w_i, w_j|E_2)\), is approximately equivalent to determining the similarity between the two corresponding graphs denoted by the WWC matrices. To compare the two graphs, we use the graph kernel described in the previous section.

Specifically, we set the function \(f\) as follows:

\[
f(v^1_i, v^2_i, v^1_j, v^2_j, e_i, e_j) = \begin{cases} 
\min(e_i, e_j) & \text{if } v^1_i = v^2_i \land v^1_j = v^2_j \\
0 & \text{otherwise.} \end{cases}
\tag{5.4}
\]

Also, we set both \(p_x\) and \(q_x\) to be the uniform distribution, i.e. \(p_x(i) = 1/\|\zeta_x\|\) (and similarly for \(q_x\)). This reduces the expression for \(K(G_1, G_2)\) to the sum of the entries of the matrix \(\Gamma_x\). This choice is motivated by the observation that the different images in the environment are likely to have a different number of words and without prior information,
Algorithm 7 Construct WWC Matrix

Input: I : dataset containing images $I_1, I_2, \ldots, I_n$
Output: W : The word-word co-occurrence matrix for I

for $j = 1 : n$ do
    words = ExtractWords($I_k$)
    for each pair $\langle w_1, w_2 \rangle$ in words do
        if co_occur($w_1, w_2$) then
            if $w_1 < w_2$ then
                $W(w_1, w_2) = W(w_1, w_2) + 1$;
            end if
        end if
    end for
end for

all document lengths are equally likely.

The final approximation is that we restrict ourselves to only random walks containing one edge (or two vertices). In other words, we set $\lambda_1 = 1$ and $\lambda_i = 0$ for all $i > 1$. This reduces the graph kernel in the previous section to the proposed similarity criterion. In order to show that this is a valid kernel, we note that the min() function is a valid kernel (from the Histogram Intersection Kernel). The sum of kernel functions is still a kernel. Since the proposed criterion is just the sum of a set of min() functions, the validity follows.

5.4.3 Constructing the WWC Matrix

The algorithm is shown in Algorithm 7. Coarse words are extracted from each image in the dataset. The explanation of the coarseness is given in Section 5.5.2. Co-occurrence counts are incremented for a pair of words $\langle w_1, w_2 \rangle$ if the distance between the corresponding features lie within a certain distance of each other in the image. If a pair of words occurs multiple times in the same image, the corresponding co-occurrence counts are incremented multiple times.

The threshold used in the definition of co-occurrence used here has two major purposes:

- It enforces a weak geometric consistency check, and
- ensures greater sparsity in the WWC matrix.
5.4.4 Computational Considerations

From an implementation standpoint, we note that the matrix $\Gamma_x$ is extremely large, of size $O(|V|^2)$. However, since the tf-vector for each image is generally sparse, the WWC matrix is reasonably sparse. Consequently, $\Gamma_x$ is also relatively sparse. This means that the proposed criterion can be computed fairly efficiently in practice (since it only depends on the number of non-zero terms in the WWC matrices). Also, the matrix is maintained in an upper triangular form for efficiency. However, as the size of the vocabulary increases, the WWC matrices are likely to contain more word pairs. In such cases, we note that the pairs of features that have a low frequency of co-occurrence are less descriptive of the environment and hence can be disregarded from the similarity computation. Specifically, this is done by setting entries in the WWC matrix to zero, if the corresponding entries are below a threshold $\lambda$ (we call this threshold the sparsification threshold in the rest of this work).

5.5 Hierarchical Place Recognition

Our loop detection approach follows the loop detection algorithm for single environments DBoW2 presented in [17]. For the reader’s convenience, we start by giving a brief description of that technique.

5.5.1 Single Environment

DBoW2 uses a fixed-size visual vocabulary tree of binary words to represent images. This is the result of a hierarchical $k$-means clustering of the descriptor space, so that clusters at deeper levels of the tree correspond with finer discretization levels. The leaves of the tree are the vocabulary words and are given a term frequency - inverse document frequency (tf-idf) weight that depends on their discriminative power (according to the training data). Images are indexed by an inverted index that stores for each word in the vocabulary the images that contain it, allowing fast retrieval of those images with words in common. Finally, a direct index stores the features of the images at any level of the vocabulary tree, i.e. at any descriptor discretization level.
Given a query image $q$, a normalized similarity score $\eta_{qi}$ based on the Bhattacharyya coefficient is computed for each $i$-th candidate match. Those that exceed a threshold $\alpha$ are grouped together when they were taken at close positions, yielding a cumulative group score $H_q$. The best-ranked image of the group that maximizes $H_q$ is selected as a loop candidate. If this candidate is consistent with $k$ previous loop candidates, it is selected as a loop detection. This test is called temporal consistency and provides robustness to the results. In [17] a final geometrical verification based on epipolar geometry was done before accepting a match. However, we deactivated that functionality in our work because it veils our analysis.

### 5.5.2 Hierarchical Approach

Our loop detection approach in this work is depicted by Figure 5.2. We use a single static vocabulary tree of branching factor 10 and 6 depth levels (resulting in $10^6$ words) trained with 99M ORB features obtained from 108K independent and generic images of the SUN397 dataset [50]. This ensures a very descriptive configuration and fast access time. As an example, we show in Table 5.1 the execution time of performing image matching with different vocabulary sizes in a sequence of 71K images. The larger the vocabulary, the more sparse
Table 5.1: Median of execution times in Nordlandsbanen with 71534 images (ms)

<table>
<thead>
<tr>
<th>Vocabulary size</th>
<th>Insertion</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>1.3</td>
<td>453.0</td>
</tr>
<tr>
<td>$10^5$</td>
<td>1.8</td>
<td>100.1</td>
</tr>
<tr>
<td>$10^6$</td>
<td>2.5</td>
<td>22.0</td>
</tr>
</tbody>
</table>

its inverted index, requiring fewer comparison when computing candidate matches. On the other hand, a very high number of words can make them too specific, so that similar image features lie in different words, preventing them from matching and decreasing accuracy. The vocabulary sizes we selected do not exhibit this problem, as we show in Section 5.6.

We assume that we are provided with a set of environments obtained from other robots’ maps. In an offline stage, we compute the ORB features and the the bag of words of all the images of each environment to create its WWC matrix. Since the size of this matrix is quadratic in the number of words, we consider two different discretization levels for the words: a fine level of $10^6$ words to represent single images, and a coarser level of $10^5$ words for environments. Selecting a coarse level can be done effortlessly and without demanding another vocabulary because it only requires to pick the parent node of the fine word in the vocabulary tree. Although the coarseness level trades off speed and accuracy, $10^5$ words provide a good balance as we show in Section 5.6.

To provide a fast environment score computation, we encode all the WWC matrices in a high-level inverted index. At the same time, each environment stores the bag-of-words of its images in an environment-level inverted index. Note that the high-level inverted index indexes entries by pairs of words and the environment specific inverted index by single words. All of them compose the hierarchy of inverted indices of our approach.

During the online stage, given a query image, two bag-of-words vectors are computed at fine and coarse levels. The coarse vector is used to build a WWC matrix $I$ of that single image. The high-level inverted index is inspected then to compute the score $S(I, E_i)$ (equation 5.1) for any environment $E_i$ with pairs of co-occurrent words in common with $I$. The environment $E^* = \arg \max_{E_i} S(I, E_i)$ is selected as the most similar one. Finally, the environment-level inverted index related to $E^*$ is accessed to obtain an image-to-image loop detection. At this stage, the temporal consistency check explained above is applied to
enhance the results.

## 5.6 Experimental Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
<th>Length (Km)</th>
<th>Average speed (m/s)</th>
<th>Image size (px × px)</th>
</tr>
</thead>
<tbody>
<tr>
<td>New College [22]</td>
<td>Outdoors, dynamic</td>
<td>2.3</td>
<td>1.5</td>
<td>512 × 384</td>
</tr>
<tr>
<td>Bicocca 2009-02-25b [25]</td>
<td>Indoors, static</td>
<td>0.8</td>
<td>0.5</td>
<td>640 × 480</td>
</tr>
<tr>
<td>Ford Campus 2 [37]</td>
<td>Urban, slightly dynamic</td>
<td>4</td>
<td>6.9</td>
<td>407 × 621</td>
</tr>
<tr>
<td>Malaga 2009 Parking 6L [5]</td>
<td>Outdoors, slightly dynamic</td>
<td>1.2</td>
<td>2.8</td>
<td>1024 × 768</td>
</tr>
<tr>
<td>Whitmore (own)</td>
<td>Urban, slightly dynamic</td>
<td>-</td>
<td>-</td>
<td>640 × 480</td>
</tr>
<tr>
<td>St Lucia 19-08-00 08:45 [20]</td>
<td>Urban, dynamic, JPG artifacts</td>
<td>17.6</td>
<td>12.1</td>
<td>640 × 480</td>
</tr>
</tbody>
</table>

### 5.6.1 Hierarchical Place Recognition

To show the versatility of the place recognition, we used 7 datasets collected by other authors, shown by Table 5.6. These are long image sequences taken by mounting a camera on some kind of vehicle (robot, car, train) taken with different cameras and under different conditions, producing a total of 650K images of different sizes.

In each experiment, we create two subsets of images for each dataset. One is added to the image database of the tested loop detector and the other is used to perform queries to obtain loop matches. Each subset is created by reading the images of each dataset at a certain frequency $f$, applying a time offset to obtain disjoint subsets. In order to ensure a fair comparison between the methods, no geometric verification was used in any of the experiments to verify the matches obtained.

We performed two sets of experiments. The first experiment sampled the datasets at approximately 2Hz to obtain a total of 85,508 images, while the second experiment used a sampling frequency of 5Hz to obtain 214,483 images. The first experiment was used to verify the accuracy of the system, since fewer training images were used, whereas the second experiment was used to verify the computational scalability of the proposed approach. We compared our hierarchical approach with a detector baseline [17] that adds all the images into a single common database. Table 5.2 shows the execution time obtained on a MacBook Pro 2.7 GHz Intel Core i7 with 16 GB DDR3 memory. We used a visual vocabulary of $10^6$ words for performing place recognition and a vocabulary of $10^5$ words for building the WWC matrices. The sparsification threshold was set to $m(r)$ (the mean of the non-zero elements...
Figure 5.3: Figure showing the Confusion matrix for the Environment Selection problem. The environments are \{Bicocca25b, NewCollege, Whitmore, Malaga6L, StLucia, Ford2, Nordlands\}. The confusion matrix shows that images from StLucia are often classified as Nordlands. From the images in the first row (StLucia) and second row (Nordlands), we see that foliage in both datasets occupy a significant portion of the features obtained, leading to ambiguity.

In order to accommodate the results for all the datasets, we summarize the results as follows: for each dataset, we compute the maximum precision obtained and report the corresponding value of recall. These results are shown in Table 5.3 for the case $f = 2\,\text{Hz}$. The results clearly show that the Hierarchical Inverted Index achieves slightly lower but comparable accuracy to the single Inverted Index in most cases, and requiring a much lower execution time, as shown in Table 5.2. In addition, there are two interesting observation we make:

- The first observation is that for the Whitmore dataset, the Hierarchical Index actually outperforms the single Inverted Index. This is attributed to the fact that by selecting the correct environment, we can eliminate a significant number of ambiguous matches.

- The second observation is that both the Single Inverted Index approach and the Hierarchical Inverted Index approach perform poorly on the StLucia dataset. Further examination of the confusion matrix, shown in Figure 5.3 reveals that some images from the StLucia and Nordlands datasets look alike. Images from both the datasets are also shown in Figure 5.3. Although this is a challenge for the similarity criterion, the results of our Hierarchical Inverted Index are still similar to those of the Single Inverted Index.
Table 5.2: Comparison of mean execution time for different methods (ms)

<table>
<thead>
<tr>
<th>Images</th>
<th>Index Type</th>
<th>Environment Selection</th>
<th>Query</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>85,508</td>
<td>Hierarchical</td>
<td>7.09</td>
<td>10.7</td>
<td>17.79</td>
</tr>
<tr>
<td></td>
<td>Single</td>
<td>-</td>
<td>24.51</td>
<td>24.51</td>
</tr>
<tr>
<td>214,483</td>
<td>Hierarchical</td>
<td>6.89</td>
<td>37.73</td>
<td>44.6</td>
</tr>
<tr>
<td></td>
<td>Single</td>
<td>-</td>
<td>66.37</td>
<td>66.37</td>
</tr>
</tbody>
</table>

Table 5.3: Comparison of recall (R) for maximum values of precision (MP) for different methods

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Single</th>
<th>Hierarchical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MP</td>
<td>R</td>
</tr>
<tr>
<td>Bicocca25b</td>
<td>100</td>
<td>94.49</td>
</tr>
<tr>
<td>NewCollege</td>
<td>99.99</td>
<td>90.75</td>
</tr>
<tr>
<td>Malaga6L</td>
<td>100</td>
<td>94.48</td>
</tr>
<tr>
<td>Whitmore</td>
<td>99.49</td>
<td>57.6</td>
</tr>
<tr>
<td>StLucia</td>
<td>98.75</td>
<td>25.55</td>
</tr>
<tr>
<td>Ford2</td>
<td>100</td>
<td>95.77</td>
</tr>
</tbody>
</table>

5.6.2 Environment Selection

In this subsection we discuss a few properties of the similarity criterion. There are three major factors that affect the discriminative ability (the ability to differentiate the true environment from other environments) of the proposed similarity criterion, namely the quantizing vocabulary $V$, the sparsification threshold $\lambda$ and the number of images used to construct the WWC matrix. We proceed to examine each in turn. For all the experiments in this section, we obtain the test and training data in the same way as described in the previous subsection. The training data belonging to a specific environment is used to compute the corresponding WWC matrix. Each of the test images is treated as a separate query. We then use the similarity criterion on the test images to determine the environment they belong to.
Figure 5.4: PR Curves for Environment Selection for different values of the sparsification threshold $\lambda$. A lower value of $\lambda$ implies fewer elements are discarded, leading to higher accuracy. The interesting thing to note here is that for all the environments, setting $\lambda = m(r)$ gives us the same performance as the full matrix (i.e., when $\lambda = 0$). This implies that words that co-occur less frequently are less informative for environment selection. This is in contrast to place recognition where words that occur rarely are considered more informative.
Sparsification Threshold

Throughout our experiments, we have used $\lambda(r)$ to the mean of the sum of the non-zero entries in the row $r$. In other words,

$$\lambda(r) = \text{mean}(W(r,j))$$

$$\approx \sum_j W(r,j)/|W(r,:)|$$

$$= 1/|W(r,:)|$$  \hspace{1cm} (5.5)

Here $|W(r,:)|$ indicates the number of non-zero elements in the row $r$. Since the WWC is row normalized, sum of the elements in each row is 1. This gives us the final expression $\lambda(r) = 1/|W(r,:)|$. Since this is not the true mean, we represent it by $m(r)$. We experimentally justify this choice by comparing it with 4 other thresholds. Specifically we present PR curves for $\lambda(r) = \{0, m(r)/2, (m(r) + \max)/2, \max\}$ in Figure 5.4.

The results show that choosing $\lambda(r) = 1/|W(r,:)|$ gives us the best performance in many datasets. This can be explained by observing that at the environment level, pairs that co-occur very rarely, hardly contribute to the similarity score. This is in contrast to standard place recognition, where the infrequently occurring words contain most of the discriminative information. Another observation is that if a pair of words co-occur too frequently, they are no longer discriminative. This explains the reduced performance for $\lambda > \text{mean}$.

Vocabulary Size

The discriminative ability of the BoW approach improves with the size of the vocabulary. Since the WWC matrix relies heavily on the vocabulary, the discriminative ability also improves with vocabulary size. This is shown for various environments in Figure 5.5. However, we note that higher vocabulary sizes imply larger WWC matrices and consequently takes longer to compute the similarity. This offers a nice tradeoff between memory consumed and accuracy.
Figure 5.5: PR Curves for Environment Selection using vocabularies of different sizes. This monotonic improvement shows that the discriminative ability of the criterion improves with the size of the vocabulary.
Figure 5.6: PR Curves for Environment Selection using different sampling rates. This sampling rate controls the number of images used to construct the WWC matrix. Reducing the number of images used to construct the WWC reduces the discriminative ability of the proposed criterion. However, these curves show that environments can be identified correctly even when we use much fewer images. This highlights a dominant underlying structure to these environments.
Effect of Number of images

The number of images used to construct the WWC directly affects the discriminative ability of the WWC matrix. In order to verify the impact on the PR curve, we used a subset of the total images in the dataset. The subset was obtained by choosing images at fixed intervals. In separate runs of the experiment, the interval size was adjusted to obtain different number of images in the subset. The results are shown in Figure 5.6. These indicate that the higher the number of images used to construct the WWC matrix, the better the results. However, we note that even with as few as 1000 images per dataset, we obtain reasonable accuracy for the environment selection problem. This underlines the existence of a distinct dominant latent structure to these environments and thus justifies our use of the criterion.

5.7 Summary

In this work we have derived a graph kernel to compare graphs of co-occurrent image words. With this, we have formulated a similarity criterion to measure the similitude in appearance and geometrical spaces between environments for place recognition for SLAM. We have also proposed a novel hierarchical place recognition algorithm to detect loop closures in imagery obtained from very large and heterogeneous trajectories, adding up to more than 750Km in length. Leveraging the presented environment similarity criterion, we have showed that by using two nested levels of inverted indices, we are able to discriminate between environments to reduce the search space of image candidates. This leads to a decrease of the required execution time for the full place recognition without affecting the accuracy.

Aiming at a scenario where many robots or users with mobile phones can create, share and need to reuse maps, we consider that this work is a first step to a long-term algorithm for place creation and maintenance. To accomplish this goal, in a future work, we will address issues such as automatic online creation of new environments or environment fusion after successful recognition of places in different environments.
Bibliography

[1] https://moz.com/blog/authorship-google-plus-link-building


