CLUSTERING BY GRAPH DENSITY VARIATION ANALYSIS (GDVA) WITH DENSITY-BASED CLUSTER VALIDITY INDICES (DVI)

by

THANG VIET LE

A dissertation submitted to the
Graduate School-New Brunswick
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
For the degree of
Doctor of Philosophy
Graduate Program in Computer Science
Written under the direction of
Casimir Alexander Kulikowski
And approved by

________________________
________________________
________________________
________________________

New Brunswick, New Jersey

October, 2011
Cluster analysis is an important problem in data mining and machine learning. In reality, clustering has been a useful exploratory technique for many different applications such as image segmentation, document grouping, gene expression data analysis, paralog/ortholog detection, etc. Typically, clustering algorithms seek to partition a data set into subsets such that data objects in each subset are very similar, while data objects from different subsets are not so similar. Graph-based clustering methods partition proximity graphs, where nodes represent objects and edge weights indicate pairwise similarities, into subgraphs such that nodes of each subgraph are strongly connected, while nodes of different subgraphs are weakly connected.

This thesis presents a computationally efficient clustering method based on graph density variation analysis, where the density of an object with respect to a set of objects is defined as its average similarity to the objects of the set. The method partitions a proximity graph using the assumption that each cluster has a core region composed of high-density nodes which are strongly interconnected by edges that have a high and comparable weight. Cores of clusters are identified by extracting nodes from the core regions, and then expanded by assigning non-core nodes to the most similar cluster. Novel density-based cluster validity
indices are incorporated to measure the quality of clustering solutions so that parameter values can be determined automatically. For a direct and objective evaluation and comparison with other graph clustering techniques, random graphs are generated as input data and the results of clustering are matched against a known ground truth. The method is robust to noise and has a low complexity with two main parameters which can be used to adjust the granularity of results. Experiments show that the method is less dependent on parameters if clusters in the data are well-separated or have a dense cluster core. Good results on heterogeneous data such as multi-dimensional data, images, text and gene expression data demonstrate the effectiveness and versatility of the method.
Acknowledgements

I would like to express my deep gratitude to my advisors, Prof. Casimir Kulikowski and Prof. Ilya Muchnik, for their valuable guidance, support, encouragement, and patience during all the years I work at Rutgers. This dissertation would not have been possible without their intelligence and efforts.

I am grateful to Prof. Ahmed Elgammal, Prof. Vladimir Pavlovic and Prof. Alexander Statnikov for spending their valuable time as members of my qualifying and defense committees. Their comments on my work have been very helpful.

I thank Joseph Yun, Akshay Vashist and Nikita Lytkin, former graduate students of the Department of Computer Science. We have had many useful discussions and their results and techniques on cluster analysis have brought interesting ideas.

I appreciate very much all the technical as well as non-technical assistance from LCSR and DCS staff, especially Carol DiFrancesco who is always ready to help with complex paperwork.

I also would like to thank Vietnam Education Foundation for granting me the VEF fellowship. Their financial support and other assistance have helped me in fulfilling my graduate study at Rutgers.

Last but not least, I thank my parents, family members, and all friends for their love and strong support. They have in many different ways made my research work here possible.
# Table of Contents

Abstract .............................................................................................................................................. ii

Acknowledgements ............................................................................................................................. iv

Table of Contents ................................................................................................................................. v

List of Tables .......................................................................................................................................... viii

List of Figures ......................................................................................................................................... ix

1. Introduction ........................................................................................................................................ 1
   1.1. Motivation ....................................................................................................................................... 1
   1.2. Graph clustering method based on density analysis ........................................................................ 4
   1.3. Organization of the thesis ................................................................................................................. 7

2. Graph-based approaches for cluster analysis ...................................................................................... 8
   2.1. Cluster analysis and graph-based clustering ..................................................................................... 8
   2.2. Measuring similarity for feature-based data .................................................................................... 11
   2.3. Clustering algorithms on weighted graphs ...................................................................................... 14
      2.3.1. Agglomerative hierarchical clustering ....................................................................................... 14
      2.3.2. Clustering by minimizing graph cuts .......................................................................................... 17
      2.3.3. Markov cluster algorithm ....................................................................................................... 21
      2.3.4. Affinity propagation ................................................................................................................ 23
      2.3.5. Clustering by quasi-concave set function optimization ............................................................... 24
   2.4. Towards the clustering method based on analyzing graph density variation ................................ 28
   2.5. Evaluating clustering results ......................................................................................................... 31
      2.5.1. Rand index and adjusted Rand index ......................................................................................... 32
2.5.2. Mutual information and normalized mutual information ............................................... 33
2.5.3. Evaluation of image segmentation results ................................................................. 34

3. Graph density variation analysis (GDVA) .............................................................................. 37
3.1. Graph density estimation ................................................................................................. 37
3.2. The sequence of density variation ................................................................................... 39
3.3. Clustering by graph density variation analysis ............................................................... 46
3.4. Illustrative example ........................................................................................................ 52
3.5. Parameter settings .......................................................................................................... 53
3.6. Exploring cluster structure in clustering results ......................................................... 55
3.7. Algorithm complexity ...................................................................................................... 56

4. Density-based cluster validity indices (DVIs) ....................................................................... 58
4.1. Several existing cluster validity indices ........................................................................... 58
4.1.1. Dunn index .................................................................................................................. 58
4.1.2. Davies-Bouldin index ............................................................................................... 59
4.1.3. C index ....................................................................................................................... 60
4.1.4. Silhouette index ........................................................................................................... 60
4.2. Density-based cluster validity indices .............................................................................. 61
4.3. Using DVI for automatic determination of GDVA parameters ....................................... 63
4.4. Evaluating cluster validity indices .................................................................................. 63

5. Clustering experiments and evaluations ............................................................................... 68
5.1. Evaluating clustering methods using random weighted graphs ........................................ 68
5.1.1. Random weighted graphs .......................................................................................... 69
5.1.2. Comparisons of GDVA, NCut, MCL, and AP .......................................................... 70
5.2. Clustering experiments with heterogeneous data ................................................................. 72

5.2.1. Visual examples with data points on the plane .............................................................. 73

5.2.2. UCI data sets ................................................................................................................. 77

5.2.3. Gene expression data ..................................................................................................... 78

5.2.4. Document data set ........................................................................................................ 81

5.3. Experimental results on image segmentation ...................................................................... 82

5.3.1. Graph-based representation for images .......................................................................... 83

5.3.2. Image segmentation by GDVA .................................................................................... 85

5.3.3. Comparisons ................................................................................................................ 89

6. Parameter analysis .............................................................................................................. 93

6.1. The parameter space of GDVA ....................................................................................... 93

6.2. Choosing appropriate parameter settings for a data set ............................................... 99

7. Conclusions and future work ............................................................................................ 101

References ............................................................................................................................ 108
List of Tables

Table 4.1. Spearman correlations between NMI, ARI, DVI₁, DVI₂, DVI₃, and Silhouette ......................... 66
Table 4.2. Kendal correlations between NMI, ARI, DVI₁, DVI₂, DVI₃, and Silhouette ............................ 67
Table 5.1. Parameters of GDVA, NCut, MCL, and AP for different inter-cluster parameter values .......... 72
Table 5.2. Execution times of GDVA, NCut, MCL, and AP on five graphs with different sizes. ............... 72
Table 5.3. Accuracy measured by NMI for K-means and GDVA results on several UCI data sets ........... 78
Table 5.4. Execution times of the NCut and GDVA on image's proximity graphs of different sizes. ....... 92
List of Figures

Figure 1.1. An example of data clustering. .......................................................... 2
Figure 1.2. Proximity graph clustering example. .................................................. 3
Figure 1.3. Clusters with high-density regions as cluster cores and lower-density regions as non-core data. 4
Figure 2.1. The maximizer identification procedure. .............................................. 25
Figure 2.2. The maximizer of a graph in layered clustering. ................................... 26
Figure 2.3. Limitations of clustering based on the maximizer of a graph. .................... 28
Figure 3.1. Density of node $v$ with respect to subgraph $H$ takes into account the edges linking $v$ with $H$. 39
Figure 3.2. The sequence-construction procedure. ............................................... 40
Figure 3.3. The density of nodes in $S$ is higher than that of nodes in $V \setminus S$. ............ 42
Figure 3.4. Graph $G = (V, W)$ consists of two dense regions $S_1$ and $S_2$ connected weakly to each other. 44
Figure 3.5. Demonstration of clustering a graph with two clusters. ......................... 47
Figure 3.6. Four steps of the clustering algorithm by graph density variation analysis (GDVA). 48
Figure 3.7. The core-node-extraction procedure. ................................................ 48
Figure 3.8. The cluster-core-identification procedure. ........................................ 50
Figure 3.9. The cluster-core-expansion procedure. ............................................. 51
Figure 3.10. A weighted graph with two clusters. .............................................. 52
Figure 3.11. Sequence of minimum density variation produced by Step 1 of GDVA. .......... 53
Figure 3.12. Core nodes and cluster cores extracted by Steps 2 and 3 of GDVA. ............ 53
Figure 3.13. Full clusters expanded from cluster cores. ....................................... 53
Figure 3.14. Execution times of the steps of the algorithm on random graphs with different sparseness. 57
Figure 4.1. Node with an intra-cluster density higher than its inter-cluster density. ........ 62
Figure 4.2. An example of the correlation between ARI (x-axis) and NMI (y-axis). ........... 65
Figure 4.3. Example of the correlations of DVI$_1$, DVI$_2$, DVI$_3$, and Silhouette (x-axis) with NMI (y-axis). 66
Figure 4.4. Example of the correlations of DVI$_1$, DVI$_2$, DVI$_3$, and Silhouette (x-axis) with ARI (y-axis). 66
Figure 5.1. Weight matrix of a randomly generated graph of 100 nodes with four pre-specified clusters. . 69
Figure 5.2. Comparison of clustering accuracy of GDVA, NCut, MCL, and AP using NMI………………. 70
Figure 5.3. Comparison of clustering accuracy of GDVA, NCut, MCL, and AP using ARI……………….. 71
Figure 5.4. Example of clustering 80 points generated from three different distributions………………….. 74
Figure 5.5. Example of clustering 80 points generated from a normal distribution……………………… 76
Figure 5.6. Example of clustering 80 points generated from a uniform distribution……………………… 76
Figure 5.7. Example of clustering a set of points with core clusters………………………………………. 77
Figure 5.8. Dendrogram based on the core graph…………………………………………………………… 80
Figure 5.9. Image segmentation procedure…………………………………………………………………… 85
Figure 5.10. Example of grayscale image segmentation……………………………………………………… 86
Figure 5.11. Segmentation results on noise-added images…………………………………………………… 87
Figure 5.12. Segmentation results by using δ = 0.99, 0.9, and 0.8…………………………………………… 87
Figure 5.13. Example of grayscale image segmentation with different values of δ………………………… 88
Figure 5.14. Image segmentation examples on natural color images from the Berkeley image data set…… 89
Figure 5.15. Segmentations by NCut and GDVA on a simple grayscale image…………………………… 91
Figure 5.16. Effects of parameters on segmentation results of GDVA and NCut…………………………… 91
Figure 5.17. Comparison of segmentation results of GDVA and NCut……………………………………… 91
Figure 6.1. Projection of the Iris data set onto the first two principal components………………………… 95
Figure 6.2. Accuracy of clustering results over the parameter space when clustering the Iris data set…… 95
Figure 6.3. Accuracy of clustering results over the parameter space for the document grouping example.. 96
Figure 6.4. Accuracy for clustering random graphs with four clusters of sizes 10, 20, 30, and 40………… 97
Figure 6.5. Accuracy for clustering random graphs with four clusters of size 25…………………………… 98
Figure 6.6. Comparison of clustering random graphs with and without cluster cores…………………… 99
Chapter 1. Introduction

1.1. Motivation

Cluster analysis is a central problem for machine learning and data mining, since in many cases there are either insufficient labeled data points for learning or no labels at all, however extracting some structure by which to group or categorize data is desirable. In cluster analysis, a set of unlabeled data objects is partitioned into clusters, such that objects in each cluster are alike while objects from different clusters are different according to some similarity criterion or measure [2, 41]. Data clustering has proven a very useful exploratory technique to discover structure in data for a wide range of applications such as image/video segmentation in computer vision, document grouping in text mining, gene/protein family detection in bioinformatics, market segmentation in market research, community discovery in social network analysis, etc [11]. In computer vision, an image can be partitioned into segments such that each segment consists of similar pixels with respect to some characteristics (e.g., location, intensity, or color). The segmentation of an image can be used to locate objects and boundaries in the image. Another example of clustering application is document grouping in text mining. As a corpus of documents usually contains many different topics and sub-topics, clustering these documents can reveal hidden topics and categories in the corpus and classify the documents in these categories. In bioinformatics, an active research field, an application of clustering is to detect groups in collections of genes or proteins. Specifically, for ortholog/paralog detection in sequence analysis, clustering is used to identify similar protein sequences that have a common ancestor in the evolution process. Also, we may discover co-regulated and functionally related groups of genes or reveal new sub-cell types based on gene expression data. Clustering methods have also been applied in market research and social network analysis to discover market targets and communities of people who share common interests.
One of common data representations is to express each data object as a point in a multi-dimensional space. Figure 1.1 shows an example of a data set represented in two-dimensional space, where there are two well-separated clusters, one of which may be considered to be partitioned into two sub-clusters for a more fine-grained clustering. Many cluster analysis methods have been designed to find groups of close points in this data representation. In another approach, one can assume that the input data set has been preprocessed, so the data is represented by a proximity graph, where nodes correspond to data objects and edge weights are non-negative numbers representing the degrees of similarity between pairs of data objects. In the example of proximity graph shown in Figure 1.2, if short edges represent a high degree of similarity between two nodes and vice versa, then the graph exposes three well-separated clusters. Data sets that are represented in a feature space, such as a set of documents or pixels of a digital image, can be transformed to a proximity graph by using a similarity measurement method [63]. The objective of most other graph clustering methods is to divide a proximity graph into subgraphs such that the nodes within each subgraph are strongly connected, while the nodes belonging to different subgraphs are weakly connected [9, 63]. Such subgraphs found by a graph-based clustering algorithm correspond to clusters of objects in the input data set.

In comparison with vector-space methods, graph-based approaches have a wider applicability because it is always possible to construct proximity graphs from multi-dimensional space data while the other direction is generally not possible. Thus, graph-based methods can be applied to vector-space data as well as the data that is only representable in graph representation such as social or interaction networks. In terms of time complexity, vector-space approaches depend on the number of dimensions, so they can be less efficient
than graph-based approaches when we have data in a very high dimensional space. In addition, proximity graphs can be built with the number of edges much less than the maximum number of possible edges, so that graph-based methods can take advantage of the sparsity of the graphs to perform faster and yield better results.

Figure 1.2. Proximity graph clustering example.

In the literature, there have been many vector-based and graph-based clustering methods which are generally different because they make use of different assumptions on clusters of data. Thus, to get a good result for a data set, we need to use the method whose assumptions match best with the data. In fact, every method has its own advantages and limitations, which should be taken into consideration when choosing the method for each particular data set. Methods that are based on an objective function are usually NP-hard optimization problems, which are computationally expensive. Naturally, one of the goals is to create more efficient clustering techniques. By assuming that each cluster has a core region composed of high-density nodes which are strongly interconnected by edges that have a high and comparable weight, we have designed a computationally efficient clustering method based on graph density variation analysis (GDVA). Here the density at an object with respect to a set of objects is defined as its average similarity with the objects of the set. Novel density-based cluster validity indices (DVI) are incorporated to measure the quality of clustering solutions so that parameter values can be selected automatically to reduce the number of parameters that need to be specified by users. In addition to a fast time complexity, the GDVA method has a number of advantages, for example, the number of clusters need not be provided in advance, parameter settings can be used to control the granularity of clustering results, and structure of clusters can
be uncovered by extra information about node's density in the output. It is robust to noise and less dependent on parameter settings when clusters in the data are well-separated or have a dense cluster core. The effectiveness and versatility of GDVA are demonstrated through good results on heterogeneous data such as random graphs, multi-dimensional data, images, text and gene expression data. Empirical studies show promising comparative results to other graph clustering methods such as normalized cut [66], Markov cluster algorithm [19], affinity propagation [27], and layered clustering [53].

1.2. Graph clustering method based on density analysis

Density is a familiar notion in data analysis to measure the concentration of data objects in a particular region. An advantage of using density is that this measurement is generally robust to noise and outliers because it is weighted in collectively by all data objects in a region. We translate this notion to the density for a node of proximity graphs as its average pairwise similarity to other nodes. Therefore, a high-density region in a graph is a subgraph, of which all the nodes have a high similarity to each other. In other words, the subgraph is composed of nodes that are strongly interconnected by edges that have a high and comparable weight. By assuming that each cluster has a core which is a high-density region, we developed a computationally efficient clustering method using graph-based density variation analysis and density-based cluster validation. Figure 1.3 illustrates this assumption by showing the density in a data set with two clusters, each of which has a high-density core region. Besides the core, clusters also contain non-core regions where the density is lower, but still the data in non-core regions is more similar to other data of the same cluster than to the data of other clusters. The interpretation of the assumption in regard to the vector-based data representation is simple: Assume the data of each cluster has a unimodal distribution, the cluster core is located at the mode of the distribution.

Figure 1.3. Clusters with high-density regions as cluster cores and lower-density regions as non-core data.
The main idea of the clustering algorithm by graph density variation analysis (GDVA) is to identify and extract the core regions from the data. The separation of core regions should be much better than the separation of original clusters, so partitioning the core regions into clusters is much easier. Once every cluster core has been identified, the non-core data is assigned to the cluster that is most similar. In order to extract the core regions, we analyze the change of the density of the nodes while ordering the nodes into a sequence from low to high density. Basically, nodes are potentially in core regions if the density of their neighbors drops significantly when they are removed and also their density drops significantly when their neighbors are removed from the graph. Ideally, after extracting core regions, each cluster core is a connected component disconnected from other cluster cores. Otherwise, we can find cluster cores by removing all weak connections or using another method. Without non-core data, the remaining data should be well-separated. In the last step, the non-core data is scanned from high to low density and assigned to the most similar cluster. So an advantage of this approach is that the number of clusters need not be specified as a parameter. Experiments show that GDVA is more robust to noise and less sensitive to parameter settings when clusters are well-separated and/or have a dense core region. In terms of time complexity, GDVA runs in $O(n \log n)$ time on sparse graphs and $O(m)$ time on dense graphs, where $n$ and $m$ denote the numbers of nodes and edges, respectively. Therefore, it takes advantage of the circumstances under which graphs are sparse by simple assumptions about the weigh structure, which leads to a scalable and efficient computation.

We incorporate into GDVA a density-based cluster validity index (DVI) which can evaluate the quality of clustering results to select the best parameter values automatically. Therefore, the number of parameters needed to be specified is reduced. There are three variants of DVI, which all take into account both inter and intra-cluster connectivity by using the densities at each data object with respect to other data objects inside and outside its cluster. Compared to other validity indices in the literature, DVIs run very fast with a low complexity and also perform well under our evaluations based on correlations with known ground truths using randomly generated graphs.

In comparison with several other graph clustering techniques, the most noticeable advantage of GDVA is that it runs significantly faster than other methods such as normalized cut [66], Markov cluster algorithm
[19], and affinity propagation [27]. As the performance of any clustering methods depends on whether their assumptions on data hold, comparing different methods is strongly dependent on which data as well as what pre-processing are used in the experiments. To evaluate the methods on a wider range of data structure, we generate random graphs as input data with known ground truth and measure the similarity of clustering results to the ground truth. Thus, no data transformation is needed and we can vary the cluster separation degree to obtain more evaluation results. In addition, the application of GDVA is demonstrated by several clustering experiments with different types of data such as synthetic data, images, gene expression data, documents, and some other vector-based data sets. By using simple translation of data to proximity graphs, GDVA produces good results for these data sets. The image segmentation results give a powerful visual feedback on the effectiveness for a class of complex problems.

GDVA has two main parameters with a limited number of possible settings which control the core regions extracted from the data. Higher parameter values eliminate more weak cores and therefore produce less number of clusters. These parameters may be used to control the granularity of clusterings to possibly adjust the results from a fine-grained to a coarse-grained clustering. In addition, parameter analysis shows that we should choose appropriate parameter values for GDVA by considering the sparsity and the core assumption satisfaction of the input graph. Specifically, we use high parameter values for sparse graphs and graphs with strong core regions since clusters in these graphs are generally well-separated.

In [45] and [46], we have reported some preliminary successful empirical results of the clustering method on imaging and bioinformatics datasets. Here, we improve the method in many aspects such as: (1) The density of an object is defined as its average similarity with all other objects. Because of this, the cluster's core at the end of the density variation sequence contains only one node, so that the set of core nodes is smaller and more robust to noise. Consequently, partitioning the core set is easier. (2) The minimum density sequence exposes cluster structures under the core assumption as shown in Chapter 3 with Properties 1, 2, and 3. (3) Based on the characteristics of the sequence of minimum densities, the cluster core identification procedure always keeps adjacent core nodes in one group when identifying clusters' cores. Thus, it is easier for selecting a threshold to separate cores because even a high threshold will not break up a real cluster's core. (4) Also, the identification procedure is able to compute automatically a
threshold for partitioning the core graph, even if it is a dense or complete graph. (5) Each node in a clustering result is classified as a core, a certain, or an uncertain node. The cluster core expansion procedure puts aside uncertain nodes when computing the closest cluster for non-core nodes. This adjustment improves clustering results, especially when clusters are not well separated. (6) Parameters of the expansion procedure are determined automatically by a novel density-based validity index DVI. (7) Comparison and evaluation have utilized random graphs. The parameter space has been analyzed to guide the parameter selection based on characteristics of an input graph such as its sparsity and core assumption satisfaction.

1.3. Organization of the thesis

In Chapter 2, we review an important aspect of cluster analysis, namely the similarity measurement using distances between data objects represented in a feature space, and some common graph-based clustering approaches previously introduced in the literature. We also discuss the issue of evaluating results of clustering methods. Chapter 3 explains in detail how the clustering method by graph density variation analysis GDVA works. First, a definition for the density of a node of a graph and characteristics of the sequence of density variation are presented. Then, the main algorithms and their parameters are described along with an illustrative example. Algorithm complexity and structure of clustering results are addressed as well. Chapter 4 introduces novel cluster validity indices DVIs that evaluate the quality of clustering results in order to help determine parameter values automatically. Chapter 5 discusses an evaluation framework by random graphs and gives the results of comparisons of GDVA and several other graph clustering methods. We also illustrate in this chapter the application of GDVA on heterogeneous types of data such as synthetic sets of points on the plane, popular vector-based data sets, gene expression data, documents, and images in experiments with image segmentation. Chapter 6 analyzes the quality of clustering results over the parameter space of GDVA so as to provide a general guidance for choosing suitable parameter values based on some characteristics of input graphs. Finally, Chapter 7 gives the conclusions and states future work.
Chapter 2. Graph-based approaches for cluster analysis

Cluster analysis is a useful exploratory data analysis technique which investigates the structure of data in order to discover hidden groups of data patterns. The output of cluster analysis is usually further studied and analyzed by other methods to produce more detailed results and conclusions. In practice, we, as human beings, by visualization can easily group together similar (i.e., close) data points that are represented in a two or three dimensional space. However, when the number of data points is very large or data is represented in high dimensional spaces or by a graph, one can no longer effectively visualize and perform object grouping, so automatic methods are necessary to carry out the task. Many vector-space and graph-based methods have been proposed and studied in the literature. Here we are focusing on clustering methods for weighted graphs, and therefore one of the essential issues is to construct proximity graphs for feature-based data. In Section 2.2, commonly used similarity measurements are discussed. In Section 2.3, we review several common clustering algorithms for weighted graphs that have been applied to various data analysis problems in computer vision, text mining, and bioinformatics. The issues of selecting which similarity measuring method as well as which clustering method to be applied are very much dependent on the task and the data set under consideration because of the differences in the assumptions used by each clustering method. Advantages and limitations of clustering methods are summarized in Section 2.4. Section 2.5 presents several evaluation methods for clustering results that are used in later experiments.

2.1. Cluster analysis and graph-based clustering

In machine learning, unsupervised learning is a class of methods that explore and determine patterns in a data set of unlabeled data objects. In contrast to supervised learning, where we learn based on a training set of labeled data, unsupervised learning tries to discover useful information in data when a training set is not available. In other words, data patterns are learned without the guidance of labeled data objects. There are several reasons why unsupervised learning is beneficial [20]. Sometimes, labeled data are not available or too costly to obtain, so one may want to work first on an unlabeled data set to find groups of similar data.
objects, and then assign labels to the groups discovered based on supplementary information (usually provided by human experts). This is especially appropriate when the contents of data are not totally known beforehand. On the other hand, the characteristics of patterns in the data can change over time, so unsupervised learning can track the changes and improve the performance of classification methods. Finally, unsupervised methods can be used as a pre-processing step to find useful features for categorization or to perform exploratory data analysis to gain insights into the structure of data.

The most important task in unsupervised learning is cluster analysis, or clustering, which is very common for exploratory data analysis and has many applications in data mining, computer vision, bioinformatics, social network analysis, etc. In a cluster analysis task, we typically have a set of unlabeled data objects and want to partition it into a number of subsets such that the objects of each subset are more similar to each other than to the objects of other subsets. A clustering method is classified as vector-space if it directly processes data in a multi-dimensional feature space where each data object is represented by a point in the space. Therefore, an ideal cluster is a group of close points which are far away from other groups. On the other hand, a clustering method is called graph-based if it only makes use of the pairwise relationships between data objects, i.e., a graph with nodes and edges representing data objects and similarities, respectively. An ideal cluster in this setting is a subgraph of densely connected nodes which are weakly connected to nodes of other subgraphs.

The definition of cluster analysis is quite general as it does not precisely specify how to measure the similarity as well as the quality for a clustering. In fact, one can have many different criteria for measuring and grouping similar data objects. Thus, how to determine whether or not a clustering solution is good is not trivial unless the data is very easy to be clustered according to some user's goal. In general, different solutions may be equally good for the same data set, and different data sets may need different clustering criteria/methods so as to get a good result. Consequently, a very large number of clustering methods have been developed where each method has its own assumptions defined explicitly or implicitly in relation to the characteristics of clusters in data. Methods that are based on a reasonable objective function are usually NP-hard optimization problems. Moreover, there is always a non-trivial question for any artificial objective function, which is whether or not it is a good (or suitable) objective function for some particular data/task.
Specialists have acknowledged that clusters to be found depend very strongly on the data, the user’s goals, and also the degree of granulation [52]. Thus, approximations and heuristics are indispensable for solving clustering problems.

As an example, K-means [49], a very popular vector-space clustering method, partitions a set of points in a feature space into a pre-specified number of clusters so as to minimize the sum of square distances between the points of a cluster and the cluster’s centroid. Even though the K-means procedure does not guarantee to find the globally optimal solution (since the optimization problem is NP-hard), it usually produces good results in practice and has been used widely in clustering applications. After K-means, another popular method is the single-linkage clustering algorithm [31], which is one of many agglomerative hierarchical clustering approaches (see Section 2.3). Single-linkage clustering is a graph-based method and works very well if input data is free from noise. The procedure iteratively groups the two most similar clusters and constructs a dendrogram tree that represents the structure of clusters in data. Unlike K-means, no criterion function is optimized by agglomerative hierarchical methods. Even though there is no rigorous theoretical foundation, due to their simplicity and intuitiveness these method and their variants have become very popular and also shown satisfactory results in many clustering applications.

As mentioned above, vector-space clustering methods deal with data sets represented in a multi-dimensional space where each dimension characterizes some feature of data objects. A difficulty with this representation is known as the "curse of dimensionality", which recognizes the fact that data analysis becomes very challenging when processed in a high dimensional space. In addition, many data sets are not representable in a multi-dimensional feature space but need a graph representation. For example, a set of genes or protein sequences should be represented by a graph showing pairwise similarities between the sequences rather than by data points in a feature space. Likewise, social networks, citation networks, and web data naturally have only the graph representation. Thus, even though there have been many vector-space clustering methods, it is necessary to develop graph-based methods that are able to accept graph data directly as input. It is worth noting that converting from vector-space to graph-based data is relatively straightforward as we will discuss in the next section. On the contrary, the conversion in the opposite
direction is much more difficult if not impossible. Therefore, graph-based methods generally have a wider applicability than vector-space approaches.

2.2. Measuring similarity for feature-based data

Cluster analysis works based on the similarity between data objects, i.e., similar objects should be in the same cluster while dissimilar objects should be in different clusters. Therefore, measuring similarity is essential for any clustering methods. For vector-based data objects, we usually estimate the similarity of objects based on the distance between objects. A high distance implies a low similarity and vice versa. Distances in a multi-dimensional space are commonly calculated by using the Euclidean distance. However, there also exist other distance measures that can be used such as the Manhattan and Chebyshev distances. These metrics can be generalized by using Minkowski distance, which is defined as:

\[
\text{dist}(x, y) = \left( \sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p}
\]

(2.1)

where \(x = (x_1, x_2, \ldots, x_d)\) and \(y = (y_1, y_2, \ldots, y_d)\) \(\in \mathbb{R}^d\) are two points in a \(d\)-dimensional space. In the Minkowski metric, if \(p\) takes on 1, 2, and \(\infty\), we get the Manhattan, Euclidean, and Chebychev distances, respectively.

Since distance metrics measure dissimilarity, we need to transform the metric values to similarity in order to build proximity graphs. There are various methods for this transformation and one of the most common ways is to invert the measure so that the similarity is positive and linearly inversely proportional to the dissimilarity. Additionally, we can normalize the measure to the range of [0, 1]. The similarity between two points \(i\) and \(j\) is calculated as:

\[
W_{ij} = 1 - \frac{\text{dist}(i, j)}{\max_{u,v \in P} \text{dist}(i, j)}
\]

(2.2)

where \(P\) denotes the set of data points, and \(\text{dist}(i, j)\) denotes the distance between \(i\) and \(j\).

Another common way for transforming from distances to similarity measurements is the Gaussian function, defined by:
\[ W_{ij} = e^{-\frac{(\text{dist}(i,j))^2}{\sigma}} \]  

(2.3)

where \( \sigma > 0 \) is a parameter specifying the decreasing rate of the function.

We will only use the function (2.2) in later experiments because of two important advantages: (i) it does not require any parameters, and (ii) it is more straightforward for our visual evaluation due to the linear relationship between spatial distances and edge weights. If exponential functions such as the one defined in (2.3) were used, the data would be transformed with more distortion, even though we could adjust parameters to make clusters more compact in the proximity graph in order to make it easier to get a good clustering result.

In addition to functions (2.2) and (2.3), there are other functions commonly used in the kernel density estimation [35] such as uniform, triangular, Gaussian, cosine, Epanechnikov, biweight, and triweight. All of these functions transform a distance measure into a similarity measure for pairs of points in a multi-dimensional space. Like in kernel density estimation, the window size parameter (i.e., smoothing bandwidth parameter) of these functions is an important factor because it determines the size of a neighborhood area in which the points are considered to have an effect on each other. In other words, this parameter reflects our perception and judgment of closeness and distance and has a big impact on determining the relationship between data points of a data set.

In order to see how the kernel functions above can affect the construction of the proximity graph for a set of points, we examine here several typical cases. For the uniform kernel function, the proximity graph is an unweighted graph with an edge between any pairs of points that are within the window size from each other. The triangular kernel function is actually function (2.2) where similarities are linearly proportional to distances. In (2.2), we build a complete proximity graph because the window size is effectively the maximum distance between the points in the data set. For the triangular function, we usually set the window size to a smaller distance and therefore the proximity graph will become sparser, meaning there are no edges between the points that have a distance greater than the window size parameter. Regarding to the Gaussian kernel function (2.3), the relationship between distance and similarity is non-linear because of the shape of the Gaussian function. Compared to the triangular function, short distances will get more weights.
than long distances. As the distance increases from zero to infinity, the similarity first decreases slowly from one, then very fast, and finally very slowly to approach zero. The actual decreasing rate depends on the value of the parameter $\sigma$ in (2.3). Similarly, other kernel functions such as Epanechnikov, biweight, and triweight are all non-linear transformations. Clearly, the shape of the functions as well as the window size parameter has a decisive effect on the similarity measurements between data points and therefore on clustering results.

Although using distance to measure similarity is common for data sets in a feature space, there are other cases where similarities have to be computed by using cosine or correlation measures. The cosine similarity estimates the cosine of the angle between two vectors. If all the data points are normalized so that their magnitude is one, then the cosine is inversely proportional to the distance between the points. Since the angle between two vectors does not depend on the lengths of the vectors, the cosine similarity should be used for data in which we are not interested in the scale of data points. For example, it is often used in text mining to measure similarity between documents in a feature space, where each feature represents a term and each document is represented by a vector of the importance of the terms with respect to the document in a collection of documents (tf-idf).

For gene expression analysis in bioinformatics, a correlation coefficient is often used to measure the similarity between two genes or tissue samples. The most common correlation coefficient is Pearson correlation coefficient which is computed for two random variables by dividing the covariance of the variables by the product of their standard deviations. The Pearson correlation characterizes linear dependencies between the two variables, so when it is used to measure the similarity of two data points, it indicates how well the features of the data points are linearly correlated. Using Pearson’s correlation in gene expression analysis, two genes are similar if they are co-expressed or co-regulated in different conditions, on the other hand, two samples are similar if their genes have a similar pattern of expressions.

In case data cannot be represented in a multi-dimensional space, one can use the attributes of data objects to measure the similarity. Depending on the properties of clusters needed to be discovered, an appropriate similarity measure should be defined. For example, to discover groups of protein sequences that are possibly evolved from the same ancestor sequence, one can compute a pairwise similarity based on an edit
distance, i.e., the minimum number of operations such as insertions, deletions, and substitutions required to transform one sequence to another. As another example, in order to discover communities in a social network, the similarity of any pair of people may be measured by the number of shared interests or the volume of communication between the two people [11].

2.3. Clustering algorithms on weighted graphs

In this section, we review a number of graph-based clustering methods, which are relevant to later discussions about the GDVA method. Agglomerative hierarchical procedure is very popular and has been the basis for many other methods. The cluster-core-identification procedure in GDVA can make use of the agglomerative hierarchical clustering as an alternative method. Spectral clustering methods for minimizing graph cuts, Markov cluster algorithm, and affinity propagation are also relatively popular clustering approaches for weighted graphs. In Chapter 5, based on randomly generated graphs, we will compare the performance of GDVA with the normalized cut, Markov clustering, and affinity propagation methods. The first step of GDVA applies a greedy procedure similar to the layered clustering method, which finds an order for the nodes of a graph, specifically a node sequence is built out of the set of nodes by iteratively removing the weakest node from the input graph.

2.3.1. Agglomerative hierarchical clustering

Bottom-up or agglomerative hierarchical procedures for clustering a weighted graph [31, 81] are among the best known unsupervised methods because of their conceptual simplicity [20]. Basically, these procedures start by creating one cluster for each node of the input graph. Then the two most similar clusters are identified and merged into one cluster. Therefore, the number of clusters is decreased by one after each merging step and we get $k$ clusters after $n - k +1$ steps, where $n$ is the number of nodes. The merging process continues until a stopping criterion is satisfied, such as the number of clusters $k$ is equal to a pre-specified number or the similarity between the two most similar clusters is smaller than a pre-specified threshold. The output of a hierarchical clustering is usually represented by a dendrogram which is a binary tree that shows how the clusters are merged after each step. In a dendrogram, leaves represent the nodes of the input graph. Each branch represents a cluster and has two sub-branches representing two sub-clusters.
The height of a branch represents the similarity of the two sub-clusters. (See Figure 5.8 for an illustration of a dendrogram.) Since the two most similar clusters are merged at every step, agglomerative hierarchical clustering methods tend to favor growing clusters with minimum variances within each cluster. However, no criterion function is explicitly minimized or maximized by the clustering procedures, so there is no rigorous theoretical foundation for these methods.

To measure the similarity between two clusters in the graph setting, one of functions such as single, complete, and average linkage function can be used. The single-linkage function is most popular and based on the weight of the strongest edge between the nodes of two clusters. On the other hand, the complete-linkage function computes the weight of the weakest edge. Finally, average-linkage functions offer some compromises of the above two functions by using average weights of the edges connecting two clusters. Actually, there exist different variants of the average-linkage function which weight the sizes of the two clusters differently such as unweighted pair group method average (UPGMA), weighted pair group method average (WPGMA), unweighted pair group method centroid (UPGMC), and weighted pair group method centroid (WPGMC) as described in [70].

In case agglomerative hierarchical clustering methods are applied to data sets represented in a multi-dimensional space, there are other possible linkage functions. For example, the similarity between two clusters can be determined based on the Euclidean distance between the means of the two clusters, which turns out to have the same calculation as the UPGMC linkage function [70]. Another example of linkage functions for multi-dimensional data is the Ward distance, which is equal to the sum-of-squared-error (SSE) of the cluster merged from two sub-clusters minus the total of the SSE of the two individual sub-clusters. (SSE is the total of the squared distances between the mean of a cluster and its elements.) Therefore, the Ward method attempts to minimize the within cluster variance of data by a stepwise minimization of the increase in cluster variance [20]. Note that in case of multi-dimensional data, the vector space enables additional studies towards models and criterion functions of the clustering methods. For example, [15] shows that agglomerative clustering methods can be used as a stepwise optimization procedure for solving several model-based clustering problems where data are supposedly generated from a mixture of some specific probabilistic models.
Due to differences in the criteria of available linkage functions, clustering results are likely changed dramatically when different functions are used. The hierarchical clustering method with single-linkage function, i.e. the well-known single-linkage algorithm, produces a maximum spanning tree (equivalent to the popular minimum spanning tree) of the input proximity graph in which one can remove a number of weakest edges to obtain clusters as the remaining connected components. This method tends to discover elongated clusters because any strong edge between two clusters will merge them into one cluster containing two highly similar sub-clusters. Therefore, clusters of any shapes can be discovered by single-linkage method given they are fairly well-separated. At the same time, the method is very sensitive to noise because noisy nodes and edges can falsely merge clusters that are actually far apart (this issue is known as the "chaining effect").

In contrast to the single-linkage algorithm, the complete-linkage method measures the similarity of two clusters based on the weakest edge instead of the strongest edge. As a result, clusters cannot be elongated but must be complete and compact subgraphs. This method is also sensitive to noise and outliers and can only be used when the underlying clusters are densely compact and have comparable sizes. Note that the complete-linkage method should not be used for sparse graphs because it can never merge two clusters if there is no edge between some pair of nodes belonging to the two clusters, thus a sparse graph will be partitioned into many small pieces.

Average-linkage methods attempt to trade off between the single-linkage and complete-linkage methods by using average weights of the edges between two clusters and taking into account the sizes of clusters. They are more appropriate when the underlying clusters are not too elongated or compact. In addition, unweighted average methods (UPGMA and UPGMC) should be used if cluster sizes are comparable, while weighted average methods (WPGMA and WPGMC) are preferable when there are considerable differences in cluster sizes.

An advantage of agglomerative hierarchical clustering methods is that we obtain a multi-level nested cluster solution so that the structure of clusters can be explored and users can choose to extract a fine-grained or coarse-grained clustering based on the tree structure of nested clusters. However, it is usually very hard to decide which linkage-function should be used and to find a good setting for the threshold
parameter or the number of clusters. Moreover, because of employing the maximum and minimum operators when grouping clusters and computing the similarity between clusters, these methods tend to be sensitive to noise and outliers.

2.3.2. Clustering by minimizing graph cuts

Note that agglomerative hierarchical clustering methods presented in the previous section do not minimize or maximize a criterion function. In this section, we turn to some methods that aim to find clustering solutions that optimize a pre-defined criterion function. It is important to note that even though a criterion function will make the clustering problem become well-defined and enable some theoretical analyses, a reasonable criterion function will usually have an NP-hard complexity and so approximation solutions for the optimization problem are often accepted. However, there is usually no guarantee that the approximation solution is close to the optimal solution. Besides, even if the optimal solution can be found, it is still possible that the clustering result is not satisfactory. The reason of this is that a criterion function is merely a way for expressing our assumptions about the cluster structure in data. These assumptions can be true for only a limited number of clustering cases. There can be other cases that these assumptions do not match our expected clustering results because the definition of clustering is very general – that is to group similar objects. Moreover, it can be difficult to interpret a criterion function in terms of data distribution. In other words, it is not easy to clearly describe what kinds of cluster structures are appropriate or inappropriate when applying a particular objective function. That is why in practice simple and intuitive methods that have clear interpretations such as agglomerative hierarchical clustering are very widely used even though they are not based on a mathematical objective function.

In graph clustering, partitioning a graph based on graph cuts seems reasonable. In graph theory, a cut divides the set of nodes of a graph into disjoint subsets and the value of the cut is computed based on the weights of the edges linking nodes of different subsets. Therefore, in a proximity graph, a cut with a small value implies the similarity between the nodes of different subsets is small, and so clustering can be done by finding a minimum cut (i.e., a cut with the minimum value).
Let \( A \) and \( B \) be two subsets of nodes, i.e., \( A \subseteq V \) and \( B \subseteq V \). The sum of the weights of the edges linking the nodes of \( A \) with the nodes of \( B \) is denoted by \( \text{Assoc}(A, B) \):

\[
\text{Assoc}(A, B) = \sum_{u \in A, v \in B} w_{uv}.
\]

If \( A \cup B = V \) and \( A \cap B = \emptyset \), then \( \text{Assoc}(A, B) \) is the same as \( \text{Cut}(A, B) \) which denotes the value of the cut between \( A \) and \( B \):

\[
\text{Cut}(A, B) = \text{Assoc}(A, B).
\]

Wu and Leahy proposed a clustering method that recursively bipartitions a graph by finding minimum cuts so that it produces \( k \) subgraphs and the maximal value of the minimum cuts between any two subgraphs is minimized [78]. As pointed out in [66], a problem of this minimum cut criterion is that isolated nodes will become singleton clusters because the cut between an isolated node and the rest of the graph has the minimum value. A similar problem occurs with another criterion that minimizes the sum of the cuts between each subgraph and the rest of the graph [48, 67]:

\[
\text{Cut}(A_1, \ldots, A_K) = \sum_{i=1}^{K} \text{Cut}(A_i, V \setminus A_i).
\]

To avoid the problem of singleton clusters, one can use other criteria that require clusters to be reasonably large. Shi and Malik proposed the normalized cut criterion [66] which is defined by:

\[
\text{NCut}(A_1, \ldots, A_K) = \sum_{i=1}^{K} \frac{\text{Cut}(A_i, V \setminus A_i)}{\text{Assoc}(A_i, V)}.
\] (2.4)

Note that because \( \text{Cut}(A_i, V \setminus A_i) = \text{Assoc}(A_i, V) - \text{Assoc}(A_i, A_i) \), minimizing the normalized cut criterion is equivalent to maximizing the normalized association criterion which is defined by:

\[
\text{NAssoc}(A_1, \ldots, A_K) = \sum_{i=1}^{K} \frac{\text{Assoc}(A_i, A_i)}{\text{Assoc}(A_i, V)} = K - \text{NCut}(A_1, \ldots, A_K).
\]

In practice, NCut has been applied to computer vision problems such as image and video segmentation [64, 65], and functional brain imaging studies [37]. Other possibilities of criterion functions are the minmax cut criterion [18] and the ratio cut criterion [32] defined as:

\[
\text{MinMaxCut}(A_1, \ldots, A_K) = \sum_{i=1}^{K} \frac{\text{Cut}(A_i, V \setminus A_i)}{\text{Assoc}(A_i, A_i)}.
\] (2.5)
$\text{RatioCut}(A_1, \ldots, A_K) = \sum_{i=1}^{K} \frac{\text{Cut}(A_i, V \setminus A_i)}{|A_i|}.$ \hspace{1cm} (2.6)

Because of the denominators in these criteria, minimizing them will balance the sizes of the clusters so as to avoid clusters that are too small. However, these denominators at the same time create a problem – they favor comparable size clusters [54], which is not necessarily desirable. Another difficulty with optimizing these criteria is that users have to specify the exact number of clusters. Moreover, the optimization problems for these criteria are NP-hard, i.e., it is very computationally expensive to find the exact optimal partitioning that minimizes the normalized cut, minmax cut, and ratio cut. Therefore, to solve these problems, approximation methods are needed, among which the most popular approaches are spectral methods, which will be briefly described next. It should be noted that there also exist other approximation approaches for solving these problems such as kernel K-means method [17] and semi-definite program method [7].

Spectral methods have been well-known for a long time and are able to resolve a relaxed version of the normalized cut, minmax cut, and ratio cut minimization problems. In addition to this graph cut viewpoint, spectral methods have also been analyzed from the random walks point of view and perturbation theory [48, 66]. In the literature, there are a number of different spectral clustering methods, but these algorithms look rather similar to each other as pointed out in [75]. Basically, a spectral clustering method makes use of the eigenvalues and eigenvectors of a Laplacian matrix $L$ (there are several choices for $L$ as described hereafter), and consists of three main steps:

1. Choose and compute a Laplacian matrix $L$.
2. Find a number of smallest eigenvalues and corresponding eigenvectors of the Laplacian matrix $L$.
3. Use the eigenvectors corresponding to the smallest eigenvalues to partition the data set.

Depending on how the Laplacian matrix $L$ is computed, the solution of a spectral method will approximate the solution of one of the cut criteria mentioned above. Specifically, if the Laplacian matrix $L_1$ is equal to $D - W$, where $W$ is the weight matrix of the graph, and $D$ is the diagonal matrix of node degrees given by:

$$d_{ij} = \begin{cases} \sum_{w \in V} w_{iw} & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases}$$

$$d_{ij} = \begin{cases} \sum_{w \in V} w_{iw} & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases}$$
then the spectral method will produce an approximation solution for minimizing the ratio cut [32, 65]. On the other hand, if we use the normalized Laplacian matrix \( L_2 = I - D^{1/2}W \) [51, 65], then the solution of the spectral method will approximate the solution for minimizing the normalized cut and minmax cut. Another way of approximating the normalized cut criterion is to use another normalized Laplacian matrix \( L_3 = I - D^{-1/2}WD^{-1/2} \) [55].

After obtaining a Laplacian matrix \( L \), a number of eigenvectors of \( L \) corresponding to the smallest eigenvalues are computed. Note that computing eigenvalues and eigenvectors for large and dense graphs is very expensive (O(n³)), but if a graph is sparse, this step can be accelerated significantly by using the Lanczos method [65]. Finally, there are different ways to partition the graph based on these eigenvectors. One can get \( k \) clusters by recursively bi-partitioning the graph using the eigenvector of the second smallest eigenvalue [65, 75]. Another possibility is to use \( k \) eigenvectors corresponding the \( k \) smallest eigenvalues to form a set of points in a \( k \) dimensional space (each point corresponds to a node of the graph), and then apply the K-means method to obtain \( k \) clusters of the data [48, 51, 55].

Therefore, there are quite many variants of spectral methods for approximating the solution of the criterion functions (2.4), (2.5), and (2.6). When data is easy to be partitioned (i.e., clusters are well-separated), all these methods would return similar results. However, the results can be very different for real data and it is not easy to decide which method should be used. In general, the normalized cut criterion is more favored than the ratio cut because it minimizes the normalized cut and maximizes the normalized association at the same time [48, 66]. Between the two normalized Laplacian matrices \( L_2 \) and \( L_3 \), Luxburg advocates for using \( L_2 \) when the node degrees in the graph are very different [48]. Regarding the selection between different methods of partitioning the data based on the eigenvectors, there is not a clear winner between the recursive bi-partition and the multiway partition methods [75].

To conclude, spectral methods have been well studied and are popular for producing approximate solutions for clustering based on different graph cut criteria. However, there is no guarantee on the quality of the approximate solution compared to the optimum [48]. Meaning the results of spectral methods can be arbitrarily far from the optimal solution. Moreover, even if we could find the optimum, output clusters would tend to have similar sizes because of the normalizing factors in the criterion functions, which is not
necessarily desirable in practice. Another problem with graph-cut methods is that we have to specify the number of clusters which is unknown in many applications and it is usually not trivial to find a good value for this parameter.

2.3.3. Markov cluster algorithm

In this section, we turn to a clustering method which does not require us to specify the number of clusters. Markov cluster algorithm (MCL) [19] is a graph clustering technique based on flow simulation. It has been widely applied to bioinformatics problems such as detecting paralog/ortholog clusters in genomes [22, 47], and extracting complexes in protein interaction networks [44, 58], where it outperformed other algorithms in the studies such as affinity propagation, restricted neighborhood search clustering, super paramagnetic clustering, and molecular complex detection [10, 76].

The MCL method partitions a proximity graph using a random walk point of view. A similarity matrix is interpreted as a transition matrix where the value of each entry represents a transition probability that a random walk started from one node will visit another node. Therefore, random walks occur more often between nodes connected by high weight edges than between nodes connected by low weight edges. If a graph has dense regions that are interconnected by weak edges, then random walks of a short length are more likely to have both the starting node and the ending node in the same dense region. In other words, we can assume that a random walk that visits a dense cluster does not leave the cluster until many of its nodes have been visited. Based on this assumption, Dongen proposed in his thesis a simple algorithm for simulating flows of a short length from every node to all the nodes of a graph. Strong flows are gradually promoted while weak flows are gradually demoted by an iterative procedure. When the flows are stable, they are generally confined within a cluster and reveal a cluster structure.

Basically, the MCL algorithm operates on a stochastic matrix derived from the similarity matrix to simulate flows in a graph where strong edges carry more flows than weak edges. First, loop edges are possibly added to the nodes of the graph. Then, each entry of the similarity matrix is normalized by the sum of the values on its column. Therefore, the similarity matrix is transformed into a stochastic Markov matrix where each column contains the transition probabilities from one node to all the nodes of the graph (this matrix is
generally not symmetric). Next, flows are simulated by iteratively applying two steps of expansion and contraction on the matrix until it converges to an idempotent matrix. The expansion step simply raises the matrix to the exponent $k = 2$ using matrix multiplication, therefore it simulates flows of length $k$ from each node to all the nodes of the graph. In other words, the result matrix is interpreted a stochastic matrix, where each column represents the flow volumes from one node to all the nodes. More specifically, if the entry $m_{xz}$ in column $z$ is larger than the entry $m_{yz}$, then it implies that the flow of length $k$ from node $z$ to node $x$ is stronger than the flow of length $k$ from node $z$ to node $y$. The contraction step exaggerates further the differences between flow volumes by raising every entry of the matrix to the power $r > 1$, and then renormalizing all the columns so as to keep them stochastic. Therefore, strong flows are boosted while weak flows are additionally diminished. Parameter $r$ is called the inflation parameter and is used to control the number of clusters produced by the algorithm. With a higher value of $r$, the contraction step will demote weak flows more aggressively, and eventually break the graph into more pieces. The two steps of expansion and contraction are iterated until the matrix is sufficiently close to an idempotent matrix, i.e., it does not change by the expansion step. Although there is no convergence proof, the algorithm usually converges in practice. After convergence, remaining flows represented in the idempotent matrix are used to determine a cluster structure. The nodes that have flows to themselves are called attractors. Clusters are firstly formed by grouping together attractors that have flows to each other. Finally, other nodes, which only have flows to some attractors, are assigned to clusters based on their attractors. Note that it can happen that a node has flows to several attractors of different clusters and therefore is assigned to two or more clusters, which results in an overlapping clustering.

In summary, with the advantage that the number of clusters need not be specified, the MCL method has been widely applied in the bioinformatics field and shown to work better than some other clustering methods on several protein interaction network data sets [10, 76]. Clusters are defined by the flows that remain after iterating the two steps of expansion and contraction on a stochastic Markov matrix derived from the similarity matrix until converging to an idempotent matrix. Regarding the method’s limitations, it is mostly suited for sparse graphs with clusters having small diameters. In other words, it mostly works for easy cases when clusters are compact and well-separated, but has a lot of difficulties with dense graphs or graphs in which clusters are not compact. Moreover, it is especially slow on graphs that have many edges.
So to speed up the computation, entries with small values can be removed in a pruning step (after expansion) at the cost of decrease in accuracy.

2.3.4. Affinity propagation

Affinity propagation (AP) is a clustering method that identifies a representative example, called exemplar, for each cluster by passing real-valued messages between nodes until good exemplars and corresponding clusters are found [27]. The message sent from a node $i$ to a candidate exemplar $k$ is a value called "responsibility" $r(i, k)$ indicating how well suited $k$ is the exemplar for $i$. On the other hand, the message sent from $k$ to $i$ is a value called "availability" $a(i, k)$ reflecting the evidence that $i$ should choose $k$ as its exemplar. $r(i, k)$ and $a(i, k)$ are computed iteratively by the following formulas:

$$r(i, k) \leftarrow s(i, k) - \max_{k' \neq k} \{a(i, k') + s(i, k')\},$$
$$a(i, k) \leftarrow \min\{0, r(k, k) + \sum_{i' : i' \neq i, k} \max\{0, r(i', k)\}\},$$

where $s$ denotes the similarity matrix with entry $s(i, k)$ indicating the similarity of node $i$ to node $k$. By using these formulas, AP can work on directed graphs as well, where the similarity matrix $s$ is asymmetric.

Initially, each node is assigned a "preference" value to indicate how likely it is an exemplar. The "preference" values are setting the diagonal of the similarity matrix. If no prior knowledge is available, the preferences of all nodes are set to the same value, which can be varied to control the number of output clusters. "Responsibility" and "availability" messages are iteratively computed and sent back and forth between nodes where a node $i$ is assigned to node $k$ that maximizes the sum $a(i, k) + r(i, k)$ so that $k$ is the exemplar for $i$. If $k = i$, then $i$ is an exemplar. The algorithm stops when it converges to a stable solution where the set of exemplars is unchanged. A damping factor between zero and one is used to avoid numerical oscillations.

The main goal of AP is the same as that of K-centers (aka K-medoids) algorithm which is to find $k$ centers (exemplars) of $k$ clusters so that the distance between points labeled to be in a cluster and the center of that cluster is minimized. In contrast to the K-means algorithm, which uses the means of clusters as the centers, K-centers has to use data points as the centers. In [28], AP has been shown to be faster and able to produce
better results in comparison with K-centers. On the other hand, MCL and AP were compared in [76] for partitioning protein interaction graphs where MCL was more robust than AP on several data sets.

Since AP and K-centers always assign every element to its closest exemplar, an element can be assigned to a cluster even if this element is not similar at all to other elements (non-exemplar) of the cluster. In general, AP and K-centers do not work very well in clustering settings because they only focus on minimizing the distance between exemplars and its neighbors, but not on minimizing the total of distances of all elements within a group and simultaneously maximizing the distances between groups.

2.3.5. Clustering by quasi-concave set function optimization

In this section, we briefly summarize the layered clustering approach proposed by Mirkin and Muchnik. This method is simple and has an interesting property that the global maximum solution for an optimization problem can be easily and efficiently found. The optimization problem is based on a set function defined as the minimum value of the similarity between a set and its elements [53]. Thus, this set function measures the minimum similarity of a set with its elements and can be viewed as the density of the set. Among all the subsets of a finite set, finding the subset that has the maximum density is an optimization problem which is generally very hard to solve. However, there are cases where we can find the densest subset efficiently by a simple greedy procedure. In order to do that, we cannot define the similarity between a set and its elements arbitrarily, but the similarity must be monotonically increasing, i.e., the similarities of a set and its elements never decrease when a new element is added to the set. Note that although a monotone similarity will make the problem of finding the densest subset become easy to solve, it does not necessarily match our intuition because in reality we may expect that the similarities of a set with its elements decrease when we add new elements that are dissimilar to other elements of the set. Therefore, sometimes the densest subset found by this method is not really dense, and the optimum solution does not always lead to a good clustering.

Consider a weighted graph \( G = (V, W) \), we will define a linkage function \( \pi(i, H) \) which measures the similarity of a node \( i \) with a subset of nodes \( H \subseteq V \), where \( i \in H \). Usually, the linkage function \( \pi \) is based on pairwise similarities between \( i \) and the other elements of \( H \), for example \( \pi(i, H) = \sum_{j \in H} w_{ij} \) or \( \pi(i, H) = \max_{j \in H} w_{ij} \). Clearly, these linkage functions are monotonically increasing, i.e., \( \forall i, H_1, H_2: \) if \( i \in H_1 \subseteq H_2 \subseteq \)
V, then $\pi(i, H_1) \leq \pi(i, H_2)$. The density of a subset $H \subseteq V$ is estimated by a set function $F$ defined by $F(H) = \min_{i \in H} \pi(i, H)$. In other words, the density of $H$ is equal to the minimum value of the linkage function over all the elements of $H$. The densest subset of $V$ is the largest subset $H^*$ that has the global maximum value of $F(H)$, i.e., $H^* = \arg\max_{H \subseteq V} F(H)$. The subset $H^*$ is called the maximizer of $V$. It has been shown that the maximizer $H^*$ can be found efficiently by the following greedy algorithm if the function $F(H)$ is quasi-concave, i.e., $\forall H_1, H_2: F(H_1 \cup H_2) \geq \min(F(H_1), F(H_2))$. Moreover, $F(H)$ is quasi-concave if the linkage function $\pi$ is monotonically increasing. Therefore, we can easily find the maximizer if we use linkage functions such as $\pi(i, H) = \sum_{j \in H} w_{ij}$, or $\pi(i, H) = \max_{j \in H} w_{ij}$.

The algorithm to find the maximizer of a graph is a greedy procedure where an iterative process is executed until the graph is empty. At each iteration, we simply find and remove the nodes that have the minimum linkage from the graph. So at the end we have a sequence of minimum linkage values. The maximum value in this sequence can be easily found, and the (largest) subgraph that remains when this value is obtained is proven to be the maximizer.

```
MAXIMIZER_IDENTIFICATION($G = (V, W)$, a monotone linkage function $\pi(i, H)$)

1. $t \leftarrow 1$
2. $H_t \leftarrow V$

While $H_t$ is not empty

1. $F_t \leftarrow \min_{i \in H_t} \pi(i, H_t)$
2. $H_{t+1} \leftarrow H_t \setminus \{ i \mid i \in H_t \land \pi(i, H_t) = F_t \}$
3. $t \leftarrow t + 1$

Return the maximizer which is the largest $H_t$ such that $F_t = \max_i F_t$
```

Figure 2.1. The maximizer identification procedure.
The time complexity for a straightforward implementation of this procedure is $O(|V|^2 \tau)$, where $\tau$ is the time for evaluating the linkage function $\pi$ [72]. However, if the linkage function $\pi(i, H) = \sum_{j \in H} w_{ij}$ is used, then the running time can be improved drastically to $O(|E| + |V| \log |V|)$ by an efficient implementation using a Fibonacci heap data structure [14].

In addition to extracting the densest subgraph, the above procedure can also be used to find a chain nested sequence of sets of nodes $M_0, M_1, M_2, ..., M_L$ where $M_0 = V$ and $\forall i \in \{1, 2, ..., L\}$: $M_i \subset H_{i-1}$ such that $\forall i \in \{1, 2, ..., L\}, M' \subseteq V$: if $M' \cap (V \setminus M_i) \neq \emptyset$, then $F(M_i) > F(M')$. That is the density of $M_i$ is higher than the density of any set $M'$ which is not part of $M_i$. The sets $M_0, M_1, M_2, ..., M_L$ form a sequence of nested sets such that the density of a previous set is smaller than the density of the next sets in the sequence. Therefore, these sets create a layered clustering with increasing density where the most inner layer $M_L$ has the highest density and is the maximizer and the most outer layer $M_0$ has the lowest density and is the set of nodes $V$. The greedy procedure shown in Figure 2.1 produces only the maximizer, but we can add a simple step at the end so as to uncover the layered clustering of a graph. Based on the sequence of linkage values $F_i$, first we find $M_L$ which is the (largest) $H_L$ that has the maximum value of $F_i$, then $M_{L-1}$ is the $H_i$ that properly contains $M_L$ and has the maximum value of $F_i$, then $M_{L-2}$ is the (largest) $H_i$ that properly contains $M_{L-1}$ and has the maximum value of $F_i$, etc. We continue this procedure until we get to the entire set of nodes $V$ which should be assigned to $M_0$, and thus $L$ is determined accordingly. It has been shown that the sets produced by the above procedure form a layered clustering which is unique for a graph [53].

An interesting observation that we can make from the layered clustering is that the greedy procedure iteratively removes the nodes that are the least similar to the remaining set, thus it tends to peel from the

Figure 2.2. The maximizer of a graph in layered clustering.

The...
lowest density to the highest density regions and the densest region is the last region to be removed. We will make use of this observation when discussing the GDVA method in Chapter 3.

Note that the layered clustering does not provide a clustering solution because it does not partition a graph into subgraphs such that nodes in each subgraph are strongly connected while nodes from different subgraphs are weakly connected. In order to produce a clustering, an iterative cluster extraction has been proposed [71, 80]. First, the maximizer of a graph is extracted and each connected component in the maximizer is taken as a cluster of the graph. Then, the remaining graph after removing the maximizer is considered as a new graph and is again processed to find the maximizer. The extraction of clusters is repeated until the graph is empty and therefore a collection of clusters are produced. This clustering procedure has been applied to several data sets in bioinformatics and computer vision using different definitions for the linkage function $\pi$ [73, 80, 82].

An advantage of this clustering method is that it is fully automatic, i.e., no parameters need to be specified by users. If clusters determined by maximizers are satisfactory, then this method is very convenient. However, this is at the same time a disadvantage of the method as it has no flexibility. In reality, the data is usually complex and it is desirable that we can adjust the output if it is not satisfactory. We may also need control parameters to be able to change the granularity of the result to obtain a fine-grained or a coarse-grained clustering. In addition to the lack of flexibility, there are two other serious issues with the above method. First, the density is measured based on a monotone similarity function of a set and its elements $(\pi(i, H) = \sum_{j \in H} w_{ij}$ is often used) which tends to increase whenever more elements are added to the set. Therefore, the densest subgraph (i.e., the maximizer) based on this density is not necessarily strongly connected (traditionally, in a dense subgraph there should be a strong edge between every pair of nodes). Unless the graph is sparse, the density function $F$ tends to decrease when there is a node removed from the graph, so the maximizer tends to cover a very large part of the graph (even the whole graph if it is relatively well connected). Therefore, multiple dense subgraphs that are weakly inter-connected can actually live inside the maximizer. Secondly, removing the maximizer of a graph will likely modify or even destroy the structure of the graph and make satellite regions of the maximizer become isolated. In other words, many nodes and small regions originally connected to the maximizer will become 'orphan' after the removal of
the maximizer. We illustrate the two issues above by a very simple graph (Figure 2.3) which is unweighted and has 12 nodes and 21 edges. Intuitively, this graph should be partitioned into two clusters of the same size. However, the maximizer of this graph is a subgraph of eight nodes consisting of two maximal cliques of four nodes connected to each other by one edge in the middle. Clearly, the maximizer contains two clusters and hence should not be extracted as one cluster. Moreover, if the maximizer is taken out, then the remaining subgraph is composed of four isolated nodes which will become four singleton clusters. This is obviously not an acceptable clustering solution.

2.4. Towards the clustering method based on analyzing graph density variation

The previous section has described several graph-based clustering methods, namely agglomerative hierarchical clustering, graph cuts minimizing, Markov clustering, affinity propagation, and quasi-concave set function optimization. Note that each clustering method has its own assumptions about clusters in the data. These assumptions, which are explicitly or implicitly specified by the methods, are necessary to make the cluster analysis problem, which is defined broadly and generally, more specific and therefore manageable. As a result, the performance of a method on a data set depends on whether or not the data satisfies method's assumptions. It does not exist a method that can universally work well on any data. In order to get good clustering results, we have to select the methods that are appropriate for each particular data set. Therefore, knowing the advantages and limitations of each method is very important when applying cluster analysis.

\[ \pi(i, H) = \sum_{j \in H} w_{ij} \]
Agglomerative hierarchical method based on single-linkage is very popular in cluster analysis and works well if the gaps between adjacent clusters are significantly larger than the gaps between adjacent elements in each cluster. An advantage is clusters can be of any shape, as long as an object is close to any object of a cluster, it is assigned to that cluster. The ability to obtain clusterings at different scales according to the dendrogram may be another advantage. At the same time, forcing data into a hierarchical structure when no real hierarchy exists in the data can be a problem because it can cause false conclusions about the structure of data. The biggest limitation of agglomerative clustering is the noise sensitivity, where a noise data object can make it impossible to separate several well-separated clusters if it happens to be similar to some objects of different clusters, or a few outliers can create their own clusters because they are very dissimilar to others. We can use agglomerative clustering in the cluster-core-identification procedure of the GDVA method because noise and outliers have been eliminated from the set of core nodes extracted from dense regions of clusters by preceding steps. Clearly, separating dense regions of clusters is much easier than separating the clusters themselves if we assume that each cluster has a small dense region.

Clustering by minimizing graph cuts such as normalized cut, minmax cut, and ratio cut are also popular and well-studied methods because their objective criteria are precisely defined. Exact solutions for these graph cuts are difficult to find because the corresponding optimization problems are NP-hard, but spectral methods can be used to find approximate solutions for the problems efficiently. An advantage of these methods is that they are relatively robust to noise since the criteria penalize the existence of small clusters. If these criteria match the actual clusters in a data set, then obviously the result will be very good. However, one difficulty is to select the criterion to be used as it is not easy to interpret and differentiate these criteria and choose a suitable one for a particular data set. Besides, in order to avoid clusters that are too small, the denominators of these criteria are dependent on the sizes of clusters, but this causes a side effect when minimizing the criteria, namely it favors clusters of similar sizes, so it does not work well if clusters in the data are of very different sizes. Another limitation is that the number of clusters has to be specified, which is difficult since we usually do not know this number beforehand. In contrast, the GDVA method can find clusterings with clusters of very different sizes and it does not require specifying the number of clusters.
Markov cluster algorithm (MCL) simulates flows based on edge weights until the flows are stably confined within clusters and therefore reveal cluster structure. An advantage of this approach is the exact number of clusters need not to be known but rather emerges from the clustering process depending on some control parameters. Empirically, MCL works relatively well when graphs are sparse and all inter-cluster edges are weak, i.e., clusters are well-separated. But for dense graphs or graphs with some strong inter-cluster edges MCL has a lot of difficulty to find good clustering solutions because strong flows between clusters as well as weak flows within clusters in the beginning of the flow simulation process can be falsely aggravated in later iterations of the process. Another limitation is the running time becomes extremely slow for dense graphs. In general, MCL is significantly slower than GDVA. For instance, MCL takes more than nine minutes to cluster a graph of five thousand nodes and five million edges, while GDVA takes only three seconds on the same graph.

Affinity propagation (AP) is a heuristic method to partition data so that the total distance between a representative of each cluster and other members of the cluster is minimized. The advantage here is that the objective function is precisely defined and therefore the evaluation of results becomes straightforward. However this objective function is usually not a good criterion for clustering because it does not take into account the mutual similarity between members of the same cluster as well as the dissimilarity between members of different clusters. Therefore, the optimal solution for AP is not necessarily a good clustering solution. Another limitation of AP is that its running time is also slow even though faster than MCL. For the graph with five thousand nodes and five million edges mentioned above, AP takes five minutes to finish compared to three seconds if done by GDVA. In Chapter 5, we compare GDVA with AP, MCL, and normalized cuts using randomly generated weighted graphs.

Clustering by quasi-concave set function optimization, as the predecessor of GDVA, finds and extracts the densest region of a graph (the maximizer) as a cluster, where the density of a region is defined as the minimum in the values of a linkage function for the elements of the region. If the linkage function is monotonic, the maximizer has a nice property, which is its density decreases if we add a node to or remove a node from it. Moreover, finding this global optimum for the densest region of a graph can be done efficiently by a greedy procedure. However, using a monotone linkage function means that the values of
the function for elements of a region cannot decrease if we add another element to the region. So the density defined as above for a region will increase if we expand the region by adding a new element that has just enough connections with the region. Unless a region is really strongly interconnected and very weakly connected with a new element, expanding a region tends to increase its density. As a result, the maximizer tends to cover not only a small strongly connected subgraph but a very large part of a graph if clusters of the graph are not very well-separated. Another limitation is the maximizer usually does not match completely a cluster even if this cluster is very well-separated from the rest because there can be elements that belong to the cluster but have a weaker linkage value than other elements. These elements will be left out by the maximizer and become isolated if the maximizer is removed. Therefore, the course of iteratively extracting the maximizer and then finding another maximizer will eventually create many isolated elements. In contrast, GDVA measures the density locally at each element (as the average similarity to other elements) and search for all regions of high density nodes. These regions form the cores of clusters and are expanded based on element similarity to discover full clusters. GDVA can work on highly connected graphs, where the maximizer usually covers almost the entire graph, and produce much better quality clusterings. Additionally, parameters of GDVA give it the flexibility to adjust clustering results such as generating a fine-grained or coarse-grained granularity clustering.

2.5. Evaluating clustering results

Since there are a large number of clustering methods and each method can possess a number of control parameters, many different clustering results can be produced for the same data set. It is important to evaluate the quality of the results created by different clustering algorithms or by the same algorithms using different parameter values, so that we can compare these algorithms or select good parameter settings. This evaluation task is known as cluster validity and there are three types of validation criteria, namely external, internal, and relative criteria [41, 42, 70]. An external criterion compares a clustering result to a ground truth clustering. An internal criterion determines if a clustering result is intrinsically appropriate for an input data set. A relative criterion compares clustering results by measuring their relative quality based on a pre-specified criterion. External validity indices measure the accuracy of results based on their level of similarity to a known ground truth. In contrast, internal and relative validity indices for the quality of
results are computed without information about ground truth clusterings. In Chapter 4, we will discuss several relative cluster validity indices applicable to graph data such as Davies-Bouldin, Dunn, C index, and Silhouette, and present several new validity indices based on graph density.

For a completely unsupervised clustering problem in reality, data labels are unknown, so there is no ground truth information. However, in experiments, we can make use of the data where data labels are known to evaluate the results of clustering methods. One way to measure the accuracy of a clustering result is to label each cluster by the majority of object’s classes in that cluster and then calculate the number of the objects in the cluster that belong to other classes in order to infer an error rate. However, labeling clusters by the majority of labels is problematic if the number of clusters of output clusterings is not the same as that of the ground truth, e.g., the error rate will likely drop just by increasing the number of clusters outputted by an algorithm. The numbers of clusters in two different clusterings that we want to compare are usually very different, so the similarity of two clusterings is commonly measured based on the number of pairs of objects that are in the same as well as different clusters in two clusterings, such as in Rand, Jaccard, Hubert, Folkes indices, or based on the number of objects in the intersections between clusters of two clusterings, such as in the mutual information index. In the next sections, we consider two widely used indices, namely adjusted Rand index (ARI) and normalized mutual information (NMI), which are used in later experiments for assessing the quality of results. ARI and NMI return a value between zero and one indicating how similar two clusterings are, where zero implies they are entirely different and one means they are the same.

In cluster analysis, these indices can measure the agreement of a clustering result with the known ground truth to estimate the accuracy of the clustering result, e.g., an index value of 0.8 can be considered as an error rate of 0.2. Empirically, we see that the results of ARI and NMI are relatively consistent with each other, meaning they generally do agree on which clustering results are better in comparison with the ground truth.

2.5.1. Rand index and adjusted Rand index

Given a set \( S \) of \( n \) objects and two clusterings \( X \) and \( Y \) on \( S \), Rand index RI measures the similarity between clusterings \( X \) and \( Y \) as:
where $a$ denotes the number of pairs of objects in $S$ that are in the same set in $X$ and in the same set in $Y$, $b$ denotes the number of pairs of objects in $S$ that are in different sets in $X$ and in different sets in $Y$, $c$ denotes the number of pairs of objects in $S$ that are in the same set in $X$ and in different sets in $Y$, and $d$ denotes the number of pairs of objects in $S$ that are in different sets in $X$ and in the same set in $Y$ [69].

Rand index gives a value between zero and one, where zero indicates that the two data clusters do not agree on any pair of points and one indicates that the data clusters are exactly the same.

Since the expected value of the Rand index of two random partitions does not take a constant value, the adjusted Rand index (ARI) corrects the effect of agreement only due to chance between clusterings

\[
ARI = \frac{\text{index} - \text{expected index}}{\text{maximum index} - \text{expected index}}.
\]

Specifically, ARI is defined as [38]:

\[
ARI = \frac{2(ab - cd)}{(a + d)(d + b) + (a + c)(c + b)}.
\]

### 2.5.2. Mutual information and normalized mutual information

Mutual information, which is a symmetric measure to quantify the statistical information shared between two distributions, provides an indication of the shared information between a pair of clusterings. Let $X$ and $Y$ be the random variables described by the cluster labels of two different clusterings. In other words, each cluster represents a possible value of the random variable. The probability of each cluster in a clustering is the ratio of the number of objects in that cluster to the total number of objects. Let $I(X, Y)$ denote the mutual information between $X$ and $Y$, and $H(X)$ and $H(Y)$ denote the entropy of $X$ and $Y$, respectively. There is no upper bound for $I(X, Y)$, so for easier interpretation and comparisons, a normalized version of $I(X, Y)$ that ranges from zero to one is desirable. Several normalizations are possible based on the observation that $I(X,Y) \leq \min(H(X), H(Y))$. For example, the normalized mutual information (NMI), a widely used index, uses the arithmetic or geometric mean of $H(X)$ and $H(Y)$ for normalization [68]:
NMI = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}}.

More specifically, NMI computes the similarity between two clusterings X and Y based on the number of data objects that are in the intersections between clusters of the two clusterings:

NMI = \frac{\sum_{i=1}^{k_x} \sum_{j=1}^{k_y} n_{ij} \log \frac{n_{ij} n_{ij}}{n x_i n y_j}}{\sqrt{\left(\sum_{i=1}^{k_x} n x_i \log \frac{n x_i}{n}\right) \left(\sum_{j=1}^{k_y} n y_j \log \frac{n y_j}{n}\right)}}

where \(k_x\) denotes the number of clusters in clustering X, \(k_y\) denotes the number of clusters in clustering Y, \(n\) denotes the total number of objects, \(n x_i\) denotes the number of objects in cluster \(i\) of clustering X, \(n y_j\) denotes the number of objects in cluster \(j\) of clustering Y, and \(n_{ij}\) denotes the number of objects that are in cluster \(i\) of clustering X as well as in cluster \(j\) of clustering Y.

2.5.3. Evaluation of image segmentation results

Image segmentation is an important problem in computer vision where an image is partitioned into disjoint regions (or segments) based on certain objective. Ideally, pixels of each region are similar or related to each other and dissimilar or not-related to pixels of other regions. Therefore, clustering methods can be applied here to solve the image segmentation problem. Obviously, an issue is how to define the similarity or relationship between pixels. Based on the objective, one has to define some criterion to measure how likely two or more pixels belong to the same segment. Therefore, image proximity graphs, where pixels correspond to nodes and edges correspond to relationships between pixels, can be built and then analyzed by graph-based clustering techniques. Finally, clusters found in an image proximity graph can be translated back to segments of the input image.

Similar to clustering evaluation, the problem of evaluating image segmentation is challenging because the segmenting criterion is also not precisely defined. Obviously, evaluation methods for cluster analysis can also be applied to image segmentation as the tasks are similar. So, if ground truth data is not available, relative cluster validity indices can be used for evaluating segmentation results [33, 41, 70, 74]. However, as input data in this case is an image which is a grid of pixels, validity indices based on characteristics of
the image such as region uniformity, region contrast, line contrast, line connectivity, and texture are more preferable because they are associated more directly to our expectations about segmentation results. Various criteria for image segmentation have been proposed, where each criterion is based on different metrics, but they generally do not perform well when comparing between different algorithms or between human and machine segmentations [62]. On the other hand, if ground truth data is available, one can evaluate the segmentation results by comparing them with the ground truth. This type of comparison can also be done by using ARI and NMI indices as described in the previous sections. However, because each cluster of a segmentation result corresponds to some regions in the image, comparisons are more usually done by matching boundaries of segments between segmentations. For example, [25] has developed software to compute the correspondence between boundary maps of a segmentation result and of a ground truth, and consequently calculate the number of true positives, false positives, and false negatives. From these numbers, the precision and recall scores are computed, and then combined using the harmonic mean into an F-measure score, which is a single evaluation value that gives an indication of the quality of the segmentation result. One of the problems with this type of evaluation is that appropriate ground truth data may be not available for the comparison with the results of segmentation methods. One may build ground truth data by collecting a set of human-labeled segmentations on images. However, the problem with human-labeled segmentations is that clustering methods to be evaluated may not seek for the same goal as humans when segmenting the same image. For example, clustering methods may only group similar pixels based on some specific properties of the pixels such as color, brightness, and location, while humans usually partition an image by recognizing the objects existing in the image, i.e., each segment corresponds to an entire recognizable object even if the intensities and colors of pixels belonging to the same object greatly vary, for instance, a zebra or tiger is labeled by humans as one segment even though the animal's image contains different patches of different colors, some of which may be very alike the image background. Therefore, it would be inaccurate if we compared results of processes that have different objectives. In Chapter 5, clustering methods are applied to find image segments that contain nearby pixels with similar colors or intensities. The segmentation results of GDVA and normalized cut method can be visually compared and assessed based on the objective of grouping nearby similar pixels. As a well-known public benchmark data set, the Berkeley image set [25], which includes 500 natural images with six hand-
labeled segmentations from different human subjects for each image, has been used to evaluate many edge detection techniques\(^1\) in low-level computer vision [88]. However, because most of the natural images in this data set contain objects involving complex subcomponents and textures, the human-labeled segmentations of the images are not appropriate ground truths for evaluating the segmentation results. In general, clustering algorithms with deliberately minimal pre-processing as done in this thesis are not comparable to classification methods specialized for image edge detection based on ground truths being human-recognized image object boundaries.

---

\(^1\) Edge detectors are trained by training images and then used to detect edges in test images (for example, to compute the probability that a pixel belongs to some existing edge in an image). Typically, various cues such as color, contrast, and texture are combined to extract image features to be used in edge detectors. Note that detected edges in an image are generally not connected to form disjoint segments as in image segmentation.
Chapter 3. Graph density variation analysis (GDVA)

Assuming each cluster has a high density region, GDVA method detects the dense regions of clusters called cluster cores and then produces clustering solutions by expanding the cluster cores. The assumption can be applied for data in multi-dimensional space if each cluster is generated from a distribution that has a high peak in its probability density function (in fact each cluster is often assumed to come from a unimodal distribution). Obviously, a clustering task can be solved more easily if we need to consider only the densest regions of clusters. The original clusters may be not well-separated, but if the cluster cores are well-separated then the clusters can be uncovered easily based on their cluster cores. In this chapter, first we introduce the definition of density for a node of a graph and the procedure to construct a sequence of density variation. Then, some properties of the sequence of density variation are presented. The main algorithms of GDVA are described in Section 3.3 and illustrated by a simple example in Section 3.4. In Section 3.5, we briefly discuss the parameter settings of GDVA. More details about parameter analysis will be addressed in Chapter 6. Section 3.6 shows how the cluster structure in clustering results can be examined by using GDVA. Finally, the complexity for each step of the algorithm is given in Section 3.7 with examples of running times on graphs of different sparseness.

3.1. Graph density estimation

If data is represented in a multi-dimensional feature space, the density at any point in the space can be defined based on the number of data objects in a unit volume around that point. Using this notion of density, clustering methods, such as DBSCAN [23] and DHC [43], group into one cluster objects that have the density greater than a threshold and are located to the nearest object of the cluster at a distance smaller than another threshold, although in practice it is difficult to specify these thresholds because the density often varies from one cluster to another. More generally, kernel density estimation is a popular method in statistics which estimates the density at a point by an average of a kernel function of the distances between the point and data objects [35]. A kernel function, such as Gaussian, Epanechnikov, cosine, triangular, biweight, and triweight, is a smooth weight function which gives a higher weight for a smaller distance.
Based on this density estimation, a feature space analysis technique named mean shift [13, 29] can be used for locating the modes of a density function. From a location, mean shift procedure repeatedly moves to a new position located at the weighted mean of the density in a window centered at the previous location. Therefore, it follows a path pointing to higher density direction. The iterative process stops when the weighted mean converges to the previous location, which indicates a possible local maximum of the density function.

In the graph clustering task that we are addressing, data is represented by a proximity graph, i.e., instead of having object positions in a multi-dimensional space, here we have similarities between pairs of objects. This requires a new definition for the density of a graph node. In the literature, the notion for the density of a graph can refer to one of different meanings. For example, it is often defined in graph theory as the average degree of the graph (i.e., the number of edges divided by the number of nodes) for the densest subgraph and the densest k-subgraph problems [4, 5, 12]. Abello et al. defined the density for a subgraph as the ratio of the number of edges of the subgraph to the maximum number of possible edges (i.e., the number of edges in a clique), so as to extract dense subgraphs from a sparse unweighted graph [1]. On the other hand, Mirkin and Muchnik defined the density for a subgraph as the minimum degree of the subgraph for the purpose of finding the densest subgraph [53]. In this paper, the density is defined locally for each node of a subgraph rather than for the subgraph as a whole. If we take the sum of the weights of all the edges connected to a node and normalize it by the number of nodes, we obtain the average similarity of an object to other objects. Intuitively, this is a natural definition for the density of a node of a proximity graph because the density of an object being highly similar (close) to many other objects is higher than the density of an object being dissimilar to (far from) other objects. We call a subset of objects the core of a cluster if this subset is a group of very similar objects that have a high density.

Let us consider an undirected proximity graph \( G = (V, W) \) without loop edges, where \( V \) is the set of nodes and \( W \) is the weight matrix. The entry \( w_{uv} \) of \( W \) contains the weight of the edge between nodes \( u \) and \( v \), and indicates the degree of similarity between two corresponding data objects, \( \forall u, v \in V: w_{uv} \geq 0 \). Since \( G \) is an undirected graph without loop edges, \( W \) is a symmetric matrix of size \( |V| \times |V| \) and we do not consider the entries on the main diagonal of \( W \). The set of non-zero weight edges is denoted by \( E \), so \( |E| \leq |V| (|V|-1) / 2 \).
Given a node $v$ and a subset of nodes $H$ such that $v \in H \subseteq V$, we formally define the density of $v$ with respect to $H$ as:

$$d(v, H) = \begin{cases} 
\frac{1}{|H \setminus \{v\}|} \sum_{u \in H \setminus \{v\}} w_{vu} & \text{if } |H \setminus \{v\}| \geq 1, \\
0 & \text{if } |H \setminus \{v\}| = 0.
\end{cases}$$

(3.1)

In other words, the density of a node $v$ with respect to a subset $H$ is the average weight of all possible edges linking $v$ with the other $|H| - 1$ nodes of $H$. Note that we evaluate the density by taking the average over all edges including zero-weight ones. If $v$ is the only node of $H$, then $d(v, H)$ is taken to be zero. Also, because loop edges are not considered in this setting, we have

$$\frac{1}{|H \setminus \{v\}|} \sum_{u \in H \setminus \{v\}} w_{vu} = \frac{1}{|H| - 1} \sum_{u \in H} w_{vu}.$$

Figure 3.1. The density of node $v$ with respect to subgraph $H$ takes into account the edges linking $v$ with $H$.

We use $D(H)$ to denote the minimum density of $H$ which is the density of the weakest node $m$ of $H$. That is:

$$D(H) = \min_{v \in H} d(v, H),$$

(3.2)

$$m = \arg\min_{v \in H} d(v, H).$$

(3.3)

In case that several nodes simultaneously have the minimum density, one can pick any of them as the weakest node $m$. However, this situation rarely happens in practice for weighted graphs.

### 3.2. The sequence of density variation

The first step of the GDVA clustering method builds a sequence of minimum densities $D_t$ as well as a sequence of corresponding nodes $M_t$ by iteratively finding and removing the weakest node of a graph. Figure 3.2 shows the sequence-construction procedure based on formulas (3.1), (3.2), and (3.3).
In this section, we describe some characteristics of the sequences of $D_t$ and $M_t$, which clarifies the core assumption given in Section 2.3 and explains how GDVA works.

Let $H$ denote the remaining subset of $V$ at some iteration $t$, where $|H| > 2$. Let $m$ denote $M_t$, the weakest node of $H$, and $n$ denote $M_{t+1}$, the weakest node of $H \backslash \{m\}$. Let $\Delta = D_t - D_{t+1} = D(H) - D(H \backslash \{m\})$.

The following property shows that there is a correlation between the value of $\Delta$ and the edge weight $w_{mn}$. If $w_{mn} \leq d(n, H)$, which means the edge between $m$ and $n$ has a weight weaker than the average weight of the edges of $n$ in $H$, then the minimum density does not decrease. On the other hand, a large drop of the minimum density (i.e., $\Delta$ is positive and large) indicates that the edge between $m$ and $n$ is strong.

**Property 1.** Let $H, m, n,$ and $\Delta$ be defined as above.

1. If $w_{mn} \leq d(n, H)$, then $\Delta \leq 0$.  
\[ (3.4) \]

2. If $\Delta > 0$, then $w_{mn} > d(n, H) \geq d(m, H)$.  
\[ (3.5) \]

3. $d(n, H) - d(m, H) = \frac{1}{|H| - 2} (w_{mn} - d(n, H)) - \Delta$.  
\[ (3.6) \]

In this property, if $\Delta > 0$, then $w_{mn}$ is greater than the densities of $n$ and $m$ which are the average weight of edges of $n$ and $m$, respectively. Especially when $\Delta$ is large, the edge between $m$ and $n$ is strong compared to

```
SEQUENCE_CONSTRUCTION(G = (V, W))

H ← V.

For t ← 1, 2, ..., |V|–1, |V|

Compute $D(H)$ and $m$ by using (3.1), (3.2), and (3.3).

$D_t ← D(H)$.

$M_t ← m$.

$H ← H\backslash \{m\}$.

Return the sequences of $D_t$ and $M_t$.
```

Figure 3.2. The sequence-construction procedure.
other edges of \( m \) and \( n \) in \( H \). Using (6), in case \( \Delta > 0 \), the difference between the densities of \( m \) and \( n \) with respect to \( H \) is bounded by

\[
\frac{1}{|H|^{-2}}(w_{mn} - d(n,H)).
\]

In addition, this difference is small if \( \Delta \) is large. In other words, a large drop of the minimum density indicates that the densities of \( m \) and \( n \) are similar before the removal of \( m \). When \( m \) is removed, the density of \( n \) decreases significantly since there is a strong edge between \( m \) and \( n \). If such large drops occur continuously for several consecutive nodes in the \( M \) sequence, then these nodes are strongly interconnected and have a high and comparable density. Thus, they potentially belong to the densest region of a cluster. To see why we have this observation, let assume that \( \Delta \) is large, then the edge between \( m \) and \( n \) is strong and the densities of \( m \) and \( n \) are similar. Let \( x \) be the next weakest node. Note that \( m \) is the weakest node, so the density of \( x \) is higher than the densities of \( m \) and \( n \). When \( n \) is removed, if \( \Delta \) is again large, then we know the edge between \( n \) and \( x \) is strong and the densities of \( n \) and \( x \) are similar. In this case, the edge between \( m \) and \( x \) is at least as strong as the edge between \( m \) and \( n \) because when \( m \) is removed, the density of \( x \) drops to be similar to the density of \( n \). In other words, \( m, n, \) and \( x \) are strongly interconnected.

**Proof of Property 1.** To prove (3.4), (3.5), and (3.6), we show (3.6) first, and then derive (3.4) and (3.5).

Proof of (3.6):

\[
\Delta = D(H) - D(H\setminus\{m\}),
\]

\[
\Delta = d(m,H) - d(n,H\setminus\{m\}),
\]

\[
\Delta = \frac{1}{|H|^{-1}} \sum_{u \in H} w_{mu} - \frac{1}{|H|^{-2}} \sum_{u \in H\setminus\{m\}} w_{mu},
\]

\[
\Delta = \frac{1}{|H|^{-1}} \sum_{u \in H} w_{mu} - \frac{1}{|H|^{-2}} \left( \sum_{u \in H} w_{mu} - w_{mn} \right),
\]

\[
w_{mn} = \sum_{u \in H} w_{mu} - \frac{|H|^{-2}}{|H|^{-1}} \sum_{u \in H} w_{mu} + (|H|^{-2})\Delta, \quad (3.7)
\]

\[
w_{mn} = (|H|^{-1})d(n,H) - (|H|^{-2})d(m,H) + (|H|^{-2})\Delta,
\]

\[
w_{mn} = d(n,H) + (|H|^{-2})(d(n,H) - d(m,H) + \Delta),
\]

\[
d(n,H) - d(m,H) = \frac{1}{|H|^{-2}}(w_{mn} - d(n,H)) - \Delta.
\]
Proof of (3.4) and (3.5):

Since \( m \) is the weakest node of \( H \), \( d(n, H) \geq d(m, H) \) or \( d(n, H) - d(m, H) \geq 0 \). From (3.6), we have 
\[
(w_{mn} - d(n, H)) - \Delta \geq 0.
\]

Thus, (3.4) and (3.5) hold: If \( w_{mn} \leq d(n, H) \), then \( \Delta \leq 0 \). If \( \Delta > 0 \), then \( w_{mn} > d(n, H) \geq d(m, H) \).

The next property shows that if a graph consists of two regions with different densities, then the sequence-construction procedure tends to remove nodes in the sparser region before nodes in the denser region. This property is guaranteed if the density difference is sufficiently high.

**Property 2.** Let \( S \subseteq V \).

If \( \forall i \in \Delta V, j \in S: (|V|-1)d(i, V) < (|S|-1)d(j, S) \),

then the sequence-construction procedure will take out the nodes of \( \Delta V \) before the nodes of \( S \). Moreover,

\[
\forall H: S \subseteq H \subseteq V: D(H) < D(S).
\]

In this property, \( S \) is a dense region since the density of its nodes is higher than that of the nodes of \( \Delta V \). It is possible that a graph has many chain-nested layers of dense regions since a dense region itself can contain an inner denser region. Generally, the nodes in the \( M_t \) sequence are ordered from low-density to high-density regions. That is, weakly connected nodes are located before strongly connected nodes in the \( M_t \) sequence.

**Proof of Property 2.** Condition (3.8) is equivalent to:

\[
\forall i \in \Delta V, j \in S: \sum_{u \in \Delta V} w_{iu} < \sum_{u \in S} w_{ju}.
\]

Therefore, the first weakest node of \( V \) to be removed belongs to \( \Delta V \). Moreover, we have:
∀H: S ⊂ H ⊆ V, ∀i ∈ H \ S, ∀j ∈ S: 
\frac{1}{|H| - 1} \sum_{u ∈ H} w_{iu} < \frac{1}{|H| - 1} \sum_{u ∈ S} w_{ju},

∀H: S ⊂ H ⊆ V, ∀i ∈ H \ S, ∀j ∈ S: 
\frac{1}{|H| - 1} \sum_{u ∈ H} w_{iu} < \frac{1}{|H| - 1} \sum_{u ∈ S} w_{ju},

∀H: S ⊂ H ⊆ V, ∀i ∈ H \ S, ∀j ∈ S: d(i, H) < d(j, H).

This means that after removing any number of nodes that belong to V \ S the density of any node in S is still always higher than the density of any node not in S. Thus, the nodes of V \ S are removed before the nodes of S. In addition, ∀H: S ⊂ H ⊆ V, we have |H| > |S| and also

∀i ∈ H \ S, j ∈ S: \sum_{u ∈ H} w_{iu} < \sum_{u ∈ S} w_{ju}.

Therefore, we get \min_{v ∈ H} d(v, H) < \min_{v ∈ S} d(v, S) and (3.9) ∀H: S ⊂ H ⊆ V: D(H) < D(S) holds. □

In the next property, we analyze a scenario where a graph consists of two dense regions that are linked to each other by weak edges. Under some conditions described below, the sequence-construction procedure takes out all the nodes of one region before the nodes of the other region. Moreover, the minimum density decreases monotonically while the former region is removed.

**Property 3.** Let us assume V = S1 \cup S2, |S2| > |S1| > 1. In the sequence-construction procedure, all the nodes of S1 are removed before the nodes of S2, and the Dt value decreases monotonically during the removal of S1 if:

1. \arg\min_{v ∈ V} d(i, V) ∈ S1, \quad (3.10)
2. \forall j ∈ S2: d(j, S2) ≥ \varepsilon, \quad (3.11)
3. \forall i ∈ S1, j ∈ S2: w_{ij} ≤ \varepsilon, \quad (3.12)
4. \max_{v, u ∈ S1} w_{uv} - \min_{v, u ∈ S1} w_{uv} ≤ \frac{|S2| - |S1|}{(|S1| - 1)(|S2| - 1)} \min_{v, u ∈ S1} w_{uv} - \frac{|S2|}{|S1| - 1} \varepsilon, \quad (3.13)

where \varepsilon ≤ \frac{|S2| - |S1|}{(|S2| - 1)|S2|} \min_{v, u ∈ S1} w_{uv}. \quad (3.14)
Figure 3.4. Graph $G = (V, W)$ consists of two dense regions $S_1$ and $S_2$ connected weakly to each other.

Conditions (3.12) and (3.14) imply that the nodes of $S_1$ connect to each other much more strongly than they connect to the nodes of $S_2$. Moreover, (3.13) means the maximum of the differences between the weights of the edges of $S_1$ is bounded. In other words, the edges of $S_1$ should have a comparable weight. Under these conditions, removing a node of $S_1$ will decrease the densities of other nodes of $S_1$ and, as a result, the $D_t$ value will continually decrease until all the nodes of $S_1$ are removed. This process can be regarded as the collapse of $S_1$. Due to this property, GDVA can detect dense regions in a graph based on the change of minimum densities in the $D_t$ sequence. We assume each cluster in the graph has a dense region that has the properties of $S_1$ and we call this region the cluster's core.

After $S_1$ is removed, the remaining region $S_2$ may itself contain two nested regions that satisfy the Property 3. In this case, the same routine occurs again, which means one of the regions is removed completely first and the $D_t$ value decreases during its removal. However, if $S_2$ contains only one dense region, in other words, $S_2$ does not contain multiple dense regions linked by weak edges, then the $D_t$ value tends to increase until one last node remains, at which time it drops to zero.

**Proof of Property 3.** As before, $H$ denotes the remaining subset of $V$ at some iteration. The weakest node of $H$ is denoted by $m$. We assume that $m$ and all the previously removed nodes belong to $S_1$ and show that under the conditions (3.11), (3.12), (3.13), and (3.14), the weakest node at the next iteration also belongs to $S_1$. Note that because of (3.10) $\arg\min_{i \in V} d(i, V) \in S_1$, the first weakest node of $V$ belongs to $S_1$. By using induction, all the nodes of $S_1$ are taken out before the nodes of $S_2$.

Since we assume all the previously removed nodes are in $S_1$, we have $S_2 \subset H$ and $|S_2| < |H|$. 

\[ G = (V, W) \]
Because of (3.11) \( \forall j \in S_2: d(j, S_2) \geq \varepsilon \), (3.12) \( \forall i \in S_1, j \in S_2: w_{ij} \leq \varepsilon \), \( S_2 \subseteq H \), and \( m \in S_1 \), we can see that
\[ \forall j \in S_2: d(j, H) \leq d(j, H \setminus \{m\}) \]. This means that the densities of all the nodes in \( S_2 \) do not decrease after \( m \) is removed.

Let \( S_1' = S_1 \cap H \), so \( S_1' \subseteq S_1 \). Using (3.7), we see that the next weakest node belongs to \( S_1' \) and \( \Delta > 0 \) if
\[ \forall n \in S_1': w_{mn} > \sum_{u \in H} w_{mu} - \frac{|H| - 2}{|H| - 1} \sum_{u \in H} w_{mu}. \] (3.15)

The reason for this is, when \( m \) is removed, the density of any node of \( S_1' \) decreases while the density of any node of \( S_2 \) does not decrease. Next, we will show that (3.12), (3.13), and (3.14) are sufficient for (3.15) to hold.

Because of (3.12) and \( \forall n, k \in S_1': w_{nk} \leq \max_{v, u \in S_1} w_{vu} \) and \( w_{mk} \geq \min_{v, u \in S_1} w_{vu} \), we can write the following:

\[ \forall n \in S_1': \sum_{u \in H} w_{mu} - \frac{|H| - 2}{|H| - 1} \sum_{u \in H} w_{mu} \leq (|S_1'| - 1) \max_{v, u \in S_1} w_{vu} + |S_2| \varepsilon - \frac{|H| - 2}{|H| - 1} \min_{v, u \in S_1} w_{vu}, \]

\[ \forall n \in S_1': \sum_{u \in H} w_{mu} - \frac{|H| - 2}{|H| - 1} \sum_{u \in H} w_{mu} \leq (|S_1'| - 1) \left( \max_{v, u \in S_1} w_{vu} - \frac{|H| - 2}{|H| - 1} \min_{v, u \in S_1} w_{vu} \right) + |S_2| \varepsilon. \]

Since \(|S_1'| \leq |S_1|\) and \( \max_{v, u \in S_1} w_{vu} - \frac{|H| - 2}{|H| - 1} \min_{v, u \in S_1} w_{vu} > 0 \), we have
\[ \forall n \in S_1': \sum_{u \in H} w_{mu} - \frac{|H| - 2}{|H| - 1} \sum_{u \in H} w_{mu} \leq (|S_1'| - 1) \left( \max_{v, u \in S_1} w_{vu} - \frac{|H| - 2}{|H| - 1} \min_{v, u \in S_1} w_{vu} \right) + |S_2| \varepsilon. \]

Since \(|H| > |S_2|\), we can show that \( \frac{|H| - 2}{|H| - 1} \frac{|S_2| - 2}{|S_2| - 1} \).

Consequently,
\[ \forall n \in S_1': \sum_{u \in H} w_{mu} - \frac{|H| - 2}{|H| - 1} \sum_{u \in H} w_{mu} \leq (|S_1'| - 1) \left( \max_{v, u \in S_1} w_{vu} - \frac{|S_2| - 2}{|S_2| - 1} \min_{v, u \in S_1} w_{vu} \right) + |S_2| \varepsilon. \]

Note that \( \forall n \in S_1': w_{mn} \geq \min_{v, u \in S_1} w_{vu} \). Therefore, a sufficient condition for (3.15) to hold is that
\[ \min_{v, u \in S_1} w_{vu} \geq (|S_1'| - 1) \left( \max_{v, u \in S_1} w_{vu} - \frac{|S_2| - 2}{|S_2| - 1} \min_{v, u \in S_1} w_{vu} \right) + |S_2| \varepsilon. \]

After some simple algebraic manipulations, we get
\[ \max_{v, u \in S_1} w_{vu} - \min_{v, u \in S_1} w_{vu} \leq \frac{|S_2| - |S_1|}{(|S_1'| - 1)(|S_2| - 1)} \min_{v, u \in S_1} w_{vu} - \frac{|S_2|}{|S_1'| - 1} \varepsilon. \]
The right hand side of the above inequality must be positive, so the condition on $\varepsilon$ is that

$$\varepsilon \leq \frac{|S2| - |S1|}{(|S2| - 1)|S2|} \min_{u,v \in S1} w_{vu}.$$ 

### 3.3. Clustering by graph density variation analysis

We assume that each cluster in the graph has one dense region similar to $S1$ in Property 3 where the nodes are densely interconnected by edges that have a high and comparable weight. This region is called the core of a cluster. Besides the core, a cluster can have non-core regions of which the nodes have a lower density, so non-core nodes are strongly connected to some core nodes but may be weakly connected to each other. If a graph contains multiple clusters and the cores of these clusters weakly connect to each other, then the $D_t$ value plunges when nodes of the cores are removed. But if the graph contains only one cluster with one core, then the $D_t$ value tends to increase for the entire process. In other words, if the graph does not have multiple core regions, we will not see precipitous descending sections in the $D_t$ sequence. Note that the last value of the $D_t$ sequence always becomes zero because at that time only one node of the graph remains.

The clustering algorithm by graph density variation analysis (GDVA) consists of four steps. The first step computes the $D_t$ and $M_t$ sequences. The second step identifies nodes belonging to clusters' cores based on descending sections in the $D_t$ sequence. Potential core nodes are nodes in the $M_t$ sequence corresponding to large drops in the $D_t$ sequence. Note that we do not need to extract all the nodes of all the cluster cores. As long as there are some nodes from every cluster core, the set of core nodes can be used to identify the cores of all clusters. In fact, fewer core nodes are better because it will be easier for the third step, which partitions the set of core nodes into groups so that each group corresponds to a cluster's core. The last step produces the final clustering by expanding the clusters' cores into full clusters. As the $M_t$ sequence generally proceeds from weak nodes of sparse regions to strong nodes of dense regions, in order to expand clusters' cores, we scan the $M_t$ sequence in the backward direction and assign non-core nodes to their most similar cluster. So the nodes connected strongly to the cores will get assigned to their clusters before the weakly connected nodes. The similarity between a node and a cluster can be measured by either the average or the maximum weight of the edges linking the node with the cluster.
Figure 3.5 demonstrates the clustering process by a simplified illustration. A graph \( G = (V, W) \) consists of two clusters as shown in (a). \( V = V_1 \cup V_2 \), where \( V_1 \) and \( V_2 \) are two clusters of nodes such that there are only weak edges connecting them. Assume that \( S_1 \) and \( S_2 \) are dense cores of \( V_1 \) and \( V_2 \), respectively. Regions \( V_1 \setminus S_1 \) and \( V_2 \setminus S_2 \) have a low density and are non-core regions of \( V_1 \) and \( V_2 \), respectively. (b) shows the trend of the minimum density sequence. The sparse regions \( V_1 \setminus S_1 \) and \( V_2 \setminus S_2 \) are taken out first while the \( D_t \) value increases as described in Property 2. Next, there remain \( S_1 \) and \( S_2 \) which are dense regions connected weakly to each other. Under the conditions of Property 3, \( S_1 \) is removed before \( S_2 \) and there is a descending section in the \( D_t \) sequence corresponding to the nodes of \( S_1 \). Finally, the nodes of \( S_2 \) are removed and the \( D_t \) value increases until only one node of \( S_2 \) remains, at which time it drops to zero. So there are two descending sections in the \( D_t \) sequence corresponding to the nodes of \( S_1 \) and one node of \( S_2 \). These nodes constitute the set of core nodes. Since the edges inside \( S_1 \) and \( S_2 \) are much stronger than the edges bridging them, this set can be divided easily into two groups of clusters' cores which are \( S_1 \) and a node of \( S_2 \). The clusters' cores are expanded by scanning the \( M_t \) sequence backwards and assigning nodes to the most similar cluster. So the nodes of \( S_2 \) will be assigned to the second cluster. Then, the nodes of \( V_1 \setminus S_1 \) and \( V_2 \setminus S_2 \) will be assigned to the first and the second cluster, respectively, because \( V_1 \setminus S_1 \) and \( V_2 \setminus S_2 \) are better connected to \( S_1 \) and \( S_2 \), respectively, than they are connected to each other.

![Diagram](image)

Figure 3.5. (a) Graph \( G = (V, W) \) has two clusters \( V_1 \) and \( V_2 \) which have dense cores \( S_1 \) and \( S_2 \) respectively. (b) The trend of the minimum density sequence for \( G \) produced by the sequence-construction procedure.
Figure 3.6 outlines GDVA algorithm for clustering a weighted graph $G$ using four control parameters: $\delta$, $\beta$, $\theta$, and $\lambda$, where $\delta$ and $\beta$ are the two main parameters, $\theta$ and $\lambda$ can be computed automatically by heuristics as we will discuss later in this chapter. The algorithm consists of four steps. The first step already discussed in Section 3.2 constructs the sequence of density variation, based on which the second step (Figure 3.7) extracts a set of core nodes $S$. The third step (Figure 3.8) partitions $S$ into a set of clusters’ cores $C$. The last step (Figure 3.9) produces a clustering result $P$ by expanding clusters’ cores in $C$.

Using the $D_s$ and $M_s$ sequences, the core-node-extraction procedure (Figure 3.7) extracts a set of core nodes based on the slopes $\Delta_t$ in the sequence of minimum densities by using two parameters $\delta \in [0, 1]$ and $\beta \in \mathbb{N}$. Parameter $\delta$ determines the value of $\alpha$ by specifying a relative position on the ascending list of all the positive $\Delta_t$. That means if $\delta$ is zero then $\alpha$ takes on the first (smallest) value of the list, and if $\delta$ is one then $\alpha$
takes on the last (largest) value of the list. Then, \( \alpha \) specifies the minimum slope of core nodes that are extracted. Parameter \( \beta \) specifies the minimum size of groups of successive core nodes in the \( M_t \) sequence.

Note that, if some value \( D_k \) is equal to zero, it implies that the corresponding node \( M_k \) is not connected to the remaining graph at the iteration \( k \) in the sequence-construction procedure. This happens when a well-separated cluster is completely isolated from the graph in previous iterations of the procedure so that \( M_k \) is the last node of this cluster to remain. Therefore, we must include a node of this cluster into the set of core nodes regardless of the values of \( \delta \) and \( \beta \). This is also the case for the last node in the \( M_t \) sequence because \( D_{|V|} \) is always equal to zero. In fact, this last node belongs to the core of the strongest cluster. If we increase \( \delta \) and \( \beta \), the number of core nodes is reduced; moreover, only strong core nodes are extracted as they are limited to denser parts of clusters' cores. Generally, we use \( \beta \in \{1, 2, 3, 4, 5\} \) as higher values of \( \beta \) may exclude core nodes belonging to small or weak clusters.

In the cluster-core-identification procedure (Figure 3.8), \( \theta \in [0, \max_{i,j \in V} w_{ij}] \) is a threshold for removing weak edges before finding connected components in the core graph \( G_S \). Regardless of the value of \( \theta \), we always keep together the nodes that relate to the same descending section in the \( D_t \) sequence. Therefore, the maximum number of clusters' cores is the number of descending sections with respect to the set of core nodes. A higher value of \( \theta \) can partition \( G_S \) into a larger number of clusters' cores. If the value of \( \theta \) is very large, for example \( \theta = \max_{i,j \in V} w_{ij} \), then all the descending sections will be separated and each section will correspond to one cluster's core. For lower values of \( \theta \), several descending sections can form one cluster's core if some edge linking nodes of these sections has a weight greater than \( \theta \). If \( \theta \) is zero, then no edges are removed and each connected component of \( G_S \) constitutes a cluster's core. Therefore, if the graph is sparse, we can simply set \( \theta \) to zero so that the procedure just decomposes \( G_S \) into connected components to get clusters' cores. If the graph is dense or complete, \( G_S \) may be connected in one component. In this case, \( \theta \) must be greater than zero so as to partition \( G_S \). We can use the following heuristics to determine the value of \( \theta \). Let \( |V_S| \) be the cardinality of the set of nodes of \( G_S \) (i.e., the number of core nodes \( |S| \)). First, in the set of all the edge weights of the graph, we find the top \( |V_S|^2/2 \) maximum values. Then, the smallest value of these values is taken as the value of \( \theta \). The reason for choosing \( \theta \) as above is that \( |V_S|^2/2 \) approximates the number of edges of a complete graph which has the same number of nodes as the core graph \( G_S \). We set \( \theta \)
to the smallest value among the $|V_S|^2/2$ largest edge weights of $E$, so $\theta$ can differentiate the weights of strong edges inside clusters' cores from the weights of weak edges that link one core to another. Finding the top $k$ maximum values in a list of $n$ values can be done in time $O(n \log k)$ by keeping the largest $k$ values in an efficient data structure such as a Fibonacci heap while scanning the list. Thus, the time complexity for computing $\theta$ as above is $O(|E| \log |V_S|)$.

The cluster-core-identification procedure partitions the core graph $G_S$ to obtain clusters' cores, so this is yet again a clustering problem. However, partitioning $G_S$ is much easier than partitioning $G$ because $|V_S| \ll |V|$ and the separation of clusters' cores in $G_S$ is much better than the separation of original clusters in $G$. The partitioning technique described in Figure 3.9 is simple and based on thresholding and finding connected components. It can be replaced by other graph clustering methods such as the minimum spanning tree (MST)/single linkage clustering (SLC) [31, 81], especially if we want to be able to specify the number of output clusters. In that case, the parameter for the cluster-core-identification procedure is not $\theta$ as explained above, but it is the number of clusters that we want the method to produce. The MST/SLC method is sensitive to noise and outliers; however it can be used here because $G_S$ is composed of nodes from dense parts and avoids outliers and noise nodes of the original graph.

In the cluster-core-expansion procedure (Figure 3.9), we can choose to measure the similarity between a node and a cluster by the average or the maximum weight of the edges between the node and the cluster. This similarity measure does not take into account the nodes of the cluster that are marked as uncertain. We

---

**Cluster_Core_Identification**($G, S, \theta$)

- Extract the core graph $G_S$ which is the subgraph of $G$ induced by the set of core nodes $S$.
- Remove from $G_S$ edges whose weight is smaller than $\theta$. However, we always keep the edges between core nodes that are next to each other in the $M_t$ sequence because they are related to the same descending section in the $D_t$ sequence (i.e., they belong to the same cluster's core and should not be separated in any case).
- Each connected component of $G_S$ represents the core of one cluster.

**Return** the set of clusters’ cores $C$.

Figure 3.8. The cluster-core-identification procedure.
consider a node as uncertain if the ratio of the similarity between the node and the second most similar cluster to the similarity between the node and the most similar cluster is greater than $\lambda \in [0, 1]$. Uncertain nodes usually lie at boundaries between clusters because they connect to several clusters. Thus, nodes of each cluster can be classified into core, certain, and uncertain nodes. As a result, the sub-structures of clusters can be revealed by these classes of nodes. Later we will discuss how to determine the value of $\lambda$ and choose between the average and the maximum weight for measuring the similarity between a node and a cluster.

\begin{verbatim}
CLUSTER_CORE_EXPANSION(G, Ms, C, $\lambda$)

Initialize a clustering $P$ with clusters’ cores $C$.

For $t \leftarrow |V|, |V|-1, ..., 2, 1$

If ($M_i$ of $Ms$ is not yet assigned to a cluster) then

Measure the similarities of $M_i$ to existing clusters based on the average or the maximum weight of the edges linking $M_i$ with each cluster.

If ($M_i$ has a zero similarity to all existing clusters) then

Update $P$: assign $M_i$ to a new cluster.

Continue with the next $t$.

Update $P$: assign $M_i$ to the most similar cluster.

If ($\lambda <$ the ratio of the similarity between $M_i$ and the second most similar cluster to the similarity between $M_i$ and the most similar cluster) then

Mark $M_i$ as an uncertain node so that afterward it will not be taken into account when measuring the similarities between a node and the existing clusters.

Return the clustering result $P$.
\end{verbatim}

Figure 3.9. The cluster-core-expansion procedure.
3.4. Illustrative example

Figure 3.10. A weighted graph with two clusters.

To illustrate how each step of GDVA proceeds, let us use a simple example on a small weighted graph with 9 nodes and 13 edges (Figure 3.10). This graph has two clusters on the left and on the right connecting to each other by three weak edges of weight one in the middle. Each cluster has a few nodes connecting to each other by strong edges of weights three and four, so these nodes form the core for each cluster. Other nodes in the lower part are not core nodes but are connected to core nodes by weaker edges of weights two and one. Applying the sequence-construction procedure to the above graph produces a sequence of nodes and corresponding densities as shown in Figure 3.11, where the number labels of nodes indicate the order of the sequence. The density variation plot exposes two descending segments at nodes 4-5-6 and 8-9. Therefore the maximum set of core nodes for the graph contains three nodes four, five, and eight (Figure 3.12). Clearly, the core set contains two cluster cores corresponding to two connected components of the core graph. Nodes four and five form the cluster core for the left cluster and node eight is the cluster core for the right cluster. In Figure 3.13, green and blue nodes belong to the left and right cluster, respectively.

The last step of GDVA assigns remaining nodes to the closest cluster, so nodes nine, seven, two and one are assigned to the right cluster and nodes six and three are assigned to the left cluster. In this example, because clusters are clearly separated and easily identified, GDVA parameter settings are not a matter as there is only one meaningful clustering solution. In fact, we can vary the parameters and see that GDVA results point to the same solution.
3.5. Parameter settings

In this section, we briefly discuss the parameters of GDVA. Chapter 6 will further analyze the parameter space and present several concrete examples to guide the choice of appropriate parameter settings for a data set. As outlined in Section 3.3, GDVA has parameters: \( \delta, \beta, \theta, \) and \( \lambda \), where \( \delta \in [0, 1], \beta \in \{1, 2, 3, 4, 5\}, \theta \in [0, \max_{j \in V} w_{ij}], \lambda \in [0, 1] \). However, \( \theta \) and \( \lambda \) can be computed heuristically, therefore only parameters \( \delta \in [0, 1] \) and \( \beta \in \{1, 2, 3, 4, 5\} \) need to be specified by users. So the parameter space of GDVA based on \( \delta \) and \( \beta \) is relatively small. Because the core-node-extraction procedure uses \( \delta \) and \( \beta \) to adjust the number of extracted core nodes, these parameters control the number of clusters in the output. Moreover, these
parameters can be used to extract strong or weak clusters. Increasing the values decreases the number of clusters and causes the most salient clusters being discovered. On the other hand, decreasing the values increases the number of clusters by including less salient clusters in the outputs. However, the actual values of $\delta$ and $\beta$ that are good for a particular input graph are also dependent on the sparsity of the graph and on how well-separated clusters of the graph are. Sparse graphs usually have well-separated clusters and therefore we should set $\delta$ and $\beta$ to high values. However, low values of $\delta$ and $\beta$ are used for dense graphs where clusters are not well-separated. In case the input graph is dense but every cluster has a strong core region, i.e., the core assumption holds, then high values for $\delta$ and $\beta$ can be used.

Parameter $\theta \in [0, \max_{i,j \in V} w_{ij}]$ is used to detach connections, if existed, between cluster cores. Therefore, in an easy case where cluster cores are disconnected from each other, $\theta$ can simply be set to zero. When the nodes of input graph are densely connected, the heuristic described in Section 3.3 can be used. For some other problems such as image segmentation, input graphs have some special characteristics that can help us to directly specify the value for $\theta$ instead of computing a heuristic value. Parameter $\lambda \in [0, 1]$ in the cluster-core-expansion procedure of GDVA controls when a node is considered as uncertain, and therefore is not used to compute the cluster assignment for other nodes. Small $\lambda$ results in more uncertain nodes and vice versa. Consequently, this parameter adjusts the border between clusters, where a large value of $\lambda$ produces less uncertain nodes and thus gives weak clusters (i.e., clusters that contain more uncertain nodes) more chance to expand. In contrast, small values of $\lambda$ result in more uncertain nodes and so strong clusters tend to dominate and expand to a greater extend. By default, we set the value for $\lambda$ to an intermediate value of 0.5, so that a node will become uncertain if its affinity with the second most similar cluster is greater than half of its affinity with the most similar cluster. If clusters are well-separated from each other, then the setting of $\lambda$ does not affect the result because most of nodes are not uncertain. In case that clusters are not well-separated, one can use cluster validity indices to find an appropriate setting for $\lambda$. Note that the number of clusters is not affected by $\lambda$, so comparing clustering solutions produced by different values of $\lambda$ is more straightforward. Section 4.3 gives more details on using the density-based cluster validity indices to determine the value of $\lambda$ and choose between the average and the maximum weight when measuring the similarity between a node and a cluster in the cluster-core-expansion procedure.
3.6. Exploring cluster structure in clustering results

Since nodes of each cluster are ordered in the sequence of density variation from low-density to high-density regions, we can investigate the internal structure of clusters produced by GDVA. Outputs of the cluster-core-identification procedure are nodes in cluster cores and are used as cluster representatives. So these are the most important nodes in clustering solutions. In the cluster-core-expansion procedure, cluster cores are expanded into full clusters by measuring the similarities of each node to existing clusters and then assigning the node to the most similar cluster. The confidence degree of these assignments is computed based upon the difference between the similarities of the node with the closest clusters. Therefore, cluster structure can be explored: Certain nodes are those similar to only one existing cluster and have a confidence degree of 100%. Uncertain nodes are those similar to more than one cluster. The location of a node in its clusters can be determined based on the value of its confidence degree. Higher values indicate that the node is located deeper inside the cluster, while lower values indicate the node is located towards the boundary regions of the cluster. In sum, nodes of a clustering solution can be classified into three categories:

1. Core nodes located at centers of clusters.

2. Certain nodes located far away from the boundaries separating clusters. We can also put certain nodes of a cluster into an order based on the sequence of density variation. Nodes towards the beginning of the sequence should be ranked lower than nodes towards the end of the sequence. The highly ranked nodes are generally closer to centers of clusters because they are in a higher density region than the low ranked nodes.

3. Other nodes have connections with neighboring clusters and are associated with a confidence degree indicating the closeness of the nodes to the boundary regions between the clusters.

The cluster structure can also be explored by using different values of parameters $\delta$ and $\beta$, where $\delta \in [0,1]$ and $\beta \in \{1, 2, 3, 4, 5\}$. It is especially interesting to analyze the change of clustering outputs when the value of $\delta$ is varied. It can be seen from the core-node-extraction procedure that if we gradually increase $\delta$, we eliminate more and more core nodes and therefore have less and less number of clusters. The remaining clusters are stronger than the eliminated clusters, therefore we can rank clusters based on how soon the
clusters are eliminated when we increase $\delta$. In many cases, we have been able to see a hierarchical structure of clusters where weak clusters found by lower values of $\delta$ are a part of strong clusters found by higher values of $\delta$. On the other hand, the number of detected clusters increases if we decrease the value of $\delta$. In other words, when decreasing $\delta$ we find some new clusters emerging from inside the clusters obtained by higher values of $\delta$. So based on the changes of clustering outputs, we may obtain a hierarchy which reveals the most separable clusters as well as possible embedded clusters: stronger clusters are obtained by higher values of $\delta$ while weaker clusters are produced with lower values of $\delta$. In other words, adjusting parameter $\delta$ affects the granularity of clustering results – the higher the value of $\delta$, the more coarse-grained the result. Besides, even though the choices for parameter $\beta$ are very limited, it has a similar effect as $\delta$ because when $\beta$ is increased, groups of successive core nodes in the sequence of density variation have to be larger and therefore the number of core nodes decreases. The impact of parameters $\delta$ and $\beta$ on the granularity of clustering results is demonstrated in several examples in Chapter 5.

### 3.7. Algorithm complexity

Our implementation of GDVA requires $O(|V| + |E|)$ space complexity, where $|V|$ and $|E|$ denote the number of nodes and positive weight edges, respectively. Most of the memory usage is for storing the weight matrix of the input graph as the size of this matrix is usually very much bigger than the size of all other data structures in the program. We need to maintain in memory only one copy of the matrix in sparse format, which is the same as the graph adjacency-list representation.

We use a Fibonacci heap data structure [14] so as to efficiently implement the sequence-construction procedure (Figure 3.1). The Fibonacci heap is used to keep the current densities of all nodes. At each iteration, the node with the minimum density is extracted from the heap and then the densities of the nodes adjacent to the node removed are decreased accordingly. As there are $|V|$ nodes and $|E|$ edges, the amortized time cost to extract the minimum at each iteration is $\log |V|$, and the total cost to update keys of the heap for all iterations is $|E|$. Therefore, the procedure has an amortized time cost of $O(|E| + |V| \log |V|)$. The core-node-extraction procedure (Figure 3.6) searches for positive slopes in the $D_\ell$ sequence and sorts them in ascending order, so its worst-case time complexity is $O(|V| \log |V|)$. The cluster-core-identification
procedure (Figure 3.7) removes edges with a weight below a threshold and finds connected components in the core graph $G_S$, so it can be done in time $O(|V_S| + |E_S|)$, where $V_S$ and $E_S$ denote the sets of nodes and edges of $G_S$, respectively. Note that $|V_S| \ll |V|$ and $|E_S| \ll |E|$, so this step is very fast compared to others.

The cluster-core-expansion procedure (Figure 3.8) scans the sequence of nodes $M_t$. For each node that is not yet assigned to a cluster, we scan its neighboring nodes to compute its similarities to existing clusters and find the cluster with maximum similarity. Thus, this procedure requires $O(|E| + nc |V|)$ time, where $nc$ is the number of clusters.

In sum, the overall time complexity of our implementation is $O(|E| + |V| \log |V| + nc |V|)$. Therefore, for dense graphs, where $|E| > O(|V| \log |V|)$, the total running time is $O(|E|)$. On the other hand, it is only $O(|V| \log |V|)$ for sparse graphs, where $|E| \leq O(|V| \log |V|)$ and $nc \leq O(\log |V|)$. Note that the sequence-construction procedure does not have to be re-executed if we change any parameter settings. This is one of the advantages compared to some other clustering techniques (e.g., the Markov cluster algorithm [19]) which have to be restarted from scratch if a parameter is changed.

The stacked bar plot in Figure 3.14 shows the breakdown of execution times of four procedures of GDVA on graphs with sparsity varied from relatively sparse to fully connected. Here, eight graphs were randomly generated with the same number of nodes (10 thousand) but different numbers of edges (1 to 50 million) using a PC with 3GHz Core2 Duo CPU.

![Figure 3.14. Execution times of the steps of the algorithm on random graphs with very different sparseness (10 thousand nodes and from 1 to 50 million edges).](image-url)
Chapter 4. Density-based cluster validity indices (DVIs)

Since many clustering methods are available and each method usually requires some control parameters to be specified, we will obtain different clustering results for the same data set when using different methods or using the same method but with different parameter settings. In reality, when the ground truth clustering is unknown, it is important to measure the quality of different clustering results to find which ones are better than others. A cluster validity index can assess the relative quality of a clustering result according to some pre-specified criterion which generally aims at minimizing inter-cluster connectivity and maximizing intra-cluster connectivity. In the literature, this type of validity indices is referred to as relative criteria [34, 70] to be distinguished with external criteria where we evaluate clustering results based on an apriori ground truth. Specifically, Davies-Bouldin, Dunn, C, and Silhouette are known relative validity indices that can be applied to weighted-graph partitioning [8, 33, 74]. In Sections 4.1 and 4.2, we briefly describe these indices and then propose the density-based validity indices (DVIs) which have the advantage of inexpensive computation complexity, i.e., the indices can be quickly computed in time linear to the number of edges of a graph. Because of this advantage, we can integrate the indices into GDVA to automatically determine the most appropriate values for parameters (Section 4.3). In Section 4.4, we address the problem of evaluating relative validity indices by measuring the rank correlation coefficients between validity scores and similarity scores of clusterings to ground truth based on random graphs.

4.1. Several existing cluster validity indices

4.1.1. Dunn index

Dunn index aims to identify clusterings with compact and well-separated clusters [21]. The index is defined as follows:

\[
Dunn = \frac{\min_{i=1,\ldots,n_c} \min_{j=i+1,\ldots,n_c} d(c_i, c_j)}{\max_{k=1,\ldots,n_c} diam(c_k)},
\]
where $nc$ denotes the number of clusters, $d(c_i, c_j)$ denotes a dissimilarity function between two clusters $c_i$ and $c_j$ based on the minimum dissimilarity of the elements of $c_i$ and $c_j$, i.e., $d(c_i, c_j) = \min_{x \in c_i, y \in c_j} d(x, y)$. $diam(c)$ denotes the diameter of a cluster, which may be considered as a measure of clusters’ dispersion. The diameter of a cluster $C$ can be defined as the maximum dissimilarity of the elements of $C$, i.e., $diam(C) = \max_{x, y \in C} d(x, y)$.

The Dunn index in (4.1) is defined by the ratio between the minimum of the dissimilarities of any pairs of clusters and the maximum of the dissimilarities of the elements of a cluster. Larger values of Dunn $\in [0, \infty]$ indicate better clusterings because if clusters are compact and well-separated, then the diameters of the clusters are small while the distances among the clusters are large. The main disadvantages of the Dunn index include a computational expensiveness and a high sensitiveness to noise (as the minimum cluster distance can be very small while the maximum cluster diameter can be very large when a little noise exists). Several Dunn-like indices have also been proposed [56], making use of different measurements for cluster distance and diameter.

### 4.1.2. Davies-Bouldin index

Similarly to the Dunn index, Davies-Bouldin index is also computed based on the dissimilarities between clusters and the dissimilarities between the elements of a cluster. The Davies-Bouldin index is defined as the average of the similarities of clusters with their most similar cluster, which is computed by the sum of the diameters of the two clusters divided by the dissimilarity between them [16].

$$
\text{Davies-Bouldin} = \frac{1}{nc} \sum_{i=1}^{nc} \max_{i \neq j} \left\{ \frac{diam(c_i) + diam(c_j)}{d(c_i, c_j)} \right\},
$$

(4.2)

where $nc$, $d(c_i, c_j)$ and $diam(c)$ are the same as in (4.1).

Clearly, smaller values of Davies-Bouldin $\in [0, \infty]$ indicate better clusterings because $\frac{diam(c_i) + diam(c_j)}{d(c_i, c_j)}$ is small if two clusters $c_i$ and $c_j$ are compact with small diameters and they are far away from each other.

The disadvantages of Davies-Bouldin index are similar to that of Dunn index, namely the computational expensiveness and the sensitiveness to noise, even though to some extend it is less sensitive to noise than
the Dunn index because the index uses the average instead of the maximum or the minimum of the similarities between clusters and their most similar cluster.

4.1.3. C index

C index [39] evaluates the quality of a clustering result based on the largest and smallest weights of edges and is defined as follows:

\[ C = \frac{S_{\text{max}} - S}{S_{\text{max}} - S_{\text{min}}}, \]  

(4.3)

where \( S \) is the sum of similarities over all pairs of nodes that are in the same cluster. Let \( l \) be the number of those pairs. Let \( S_{\text{max}} \) be the sum of the \( l \) largest similarities over all pairs of nodes regardless they are in the same cluster or not. In contrast, let \( S_{\text{min}} \) be the sum of the \( l \) smallest similarities. C index in (4.3) is computed as the ratio of the difference between \( S_{\text{max}} \) and \( S \) and the difference between \( S_{\text{max}} \) and \( S_{\text{min}} \). Therefore, \( C \in [0, 1] \) is small if nodes in the same cluster are among the most similar nodes and there is a big difference between the similarity of the most similar nodes and the similarity of the least similar nodes. In other words, small values of C index indicate good clusterings and vice versa. In terms of the computation complexity of the C index, we typically need to sort pairwise similarities of all the nodes in order to find the largest and smallest ones.

4.1.4. Silhouette index

Silhouette [59] is a commonly used cluster validity index defined as:

\[ \text{Silhouette} = \frac{1}{nc} \sum_{i=1}^{nc} \left( \frac{1}{|C_i|} \sum_{v \in C_i} s(v) \right), \]  

(4.4)

where \( s(v) \) denotes the Silhouette of a node \( v \), \( nc \) denotes the number of cluster, and \( C_i \) denotes the cluster \( i \). Let \( C^v \) denote the cluster that \( v \) belongs to, let \( C^{\neq v} \) denote the closest cluster to \( v \). The Silhouette of a node \( v \) is computed as:

\[ s(v) = \frac{\text{diss}(v, C^{\neq v}) - \text{diss}(v, C^v)}{\max\left(\text{diss}(v, C^{\neq v}), \text{diss}(v, C^v)\right)}, \]
where \( \text{diss}(v, C) \) denotes the average dissimilarity of node \( v \) with other nodes in its cluster, \( \text{diss}(v, C^i) \) denotes the average dissimilarity of node \( v \) with the nodes of cluster \( C^i \). Since \( C^i \) is the closest cluster to \( v \),

\[
\text{diss}(v, C^i) = \min_{i \in C} \text{diss}(v, C^i).
\]

Note that \( s(v) \in [-1, 1] \) and a large value of \( s(v) \) indicates that \( v \) is in the right cluster because it is close to other nodes of the cluster and remote from the closest cluster. The Silhouette index in (4.4) is computed as the average over the clusters of the averages of the Silhouette values of the nodes in each cluster. Therefore, larger Silhouette values indicate good clusterings. Compared to Dunn and Davies-Bouldin indices, the Silhouette index is less sensitive to noise because it uses average dissimilarities and average Silhouette values of the nodes. In [74], the Silhouette index has been shown to be more effective than several other indices. With regard to computational complexity, the most expensive step of the Silhouette index is to find the closest cluster for each node based on the average dissimilarity of the node with the nodes of each cluster.

### 4.2. Density-based cluster validity indices

In this section, we introduce several cluster validity criteria based on local densities at each node of the graph with respect to the sets of nodes inside and outside its cluster. In contrast to the validity indices shown in the previous section, density-based validity indices do not use maximum and minimum values to avoid the noise sensitiveness and to improve the time complexity. In this section, we describe three density-based validity indices for a clustering result \( P \) on a graph \( G = (V, W) \). The first index is defined as follows:

\[
\text{DVI}_1(P) = \sum_{v \in V} \left( d(v, C^v) - d(v, V \setminus C^v) \right),
\]

(4.5)

where \( C^v \) denotes the set of nodes of the cluster containing \( v \) in \( P \), and \( d(v, H) \) defined in (3.1) represents the density at node \( v \) with respect to a set of nodes \( H \). The \( \text{DVI}_1 \) in (4.5) is defined as the total validity score of the nodes of \( V \), where the validity score at a node \( v \) is computed by the density at \( v \) with respect to the set of nodes of its cluster minus the density at \( v \) with respect to the set of nodes outside its cluster. We can think of \( d(v, C^v) \) and \( d(v, V \setminus C^v) \) as intra-cluster and inter-cluster densities, respectively, at node \( v \) in relation to the clustering \( P \). The value of \( \text{DVI}_1 \) is high when clusters have high internal density nodes and it is low when nodes have a high external density. Therefore, a high \( \text{DVI}_1 \) score suggests a good clustering as nodes are strongly connected to other nodes of their cluster and weakly connected to nodes of other clusters. Our
experiments show that DVI$_1$ is particularly good for comparing different clusterings that have the same number of clusters. Therefore, we use DVI$_1$ in the cluster-core-expansion procedure which determines the boundaries between clusters based on the identified cluster cores.

![Figure 4.1](image.png)

Figure 4.1. Node $v$ has a higher intra-cluster density (i.e., the density w.r.t. nodes inside its cluster) than inter-cluster density (i.e., the density w.r.t. nodes outside its cluster).

The second cluster validity index is also based on intra-cluster and inter-cluster densities of every node, but instead of finding the difference of intra and inter-cluster densities, we compute the ratio between them. The validity index DVI$_2$ is defined as:

$$DVI_2(P) = \frac{\sum_{v\in V} d(v, V \setminus C^v)}{\sum_{v\in V} d(v, C^v)}, \quad (4.6)$$

where $C^v$ and $d(v, H)$ have the same notations as before. The difference between (4.5) and (4.6) is that (4.6) computes the ratio of the total inter-cluster density to the total intra-cluster density, while (4.5) subtract the total inter-cluster density from the total intra-cluster density. A small value of DVI$_2$ indicates a good clustering. Note that because the total intra-cluster density is the denominator in (4.6), it must be greater than zero. In other words, we do not consider the clustering that each node is in a separated cluster which makes the total intra-cluster density to be zero.

The third cluster validity index is also based on ratios between intra and inter-cluster densities. However we compute the sum of the ratios instead of the ratio of the sums. We define DVI$_3$ as the total validity score of the nodes, where the validity score of a node is the ratio of the inter-cluster density to the intra-cluster density of that node. Similar to DVI$_2$, a small value of DVI$_3$ indicates a good clustering.

$$DVI_3(P) = \sum_{v\in V} \frac{d(v, V \setminus C^v)}{d(v, C^v)}. \quad (4.7)$$
Since there is no need for searching the maximum or minimum values, the computation time of the above DVI indices is very fast compared to common validity indices, such as Davies-Bouldin, Dunn, C-index, and Silhouette. Using a graph adjacency-list representation, the DVIs can be evaluated in time $O(|E|)$. The algorithm scans through the set of nodes once and for each node $v$, we compute $d(v, C^v)$ and $d(v, V \setminus C^v)$ by checking non-zero weight edges of $v$. In fact, we can compute all three DVI indices simultaneously by the same pass of the algorithm.

4.3. Using DVI for automatic determination of GDVA parameters

The DVI indices defined in Section 4.2 can help the cluster-core-expansion procedure determine its parameter settings automatically. As described in Section 3.3, the expansion procedure requires us to specify parameter $\lambda \in [0, 1]$ and to choose between the average and the maximum weight for measuring the similarity of a node with a cluster. These parameters have an impact on assigning non-core nodes to clusters and therefore affect the boundaries between clusters. Note that the number of clusters in the final result is generally not affected by these parameters because the number of clusters' cores has been established earlier by the output of the cluster-core-identification procedure. We use DVI to compare clustering results produced by the expansion procedure with different parameter settings. Specifically, with each combination of $\lambda \in \{0, 0.1, 0.2, ..., 1\}$ and the two choices of measuring similarity, we execute the expansion procedure to obtain a clustering output. Then, we assess the quality of the outputs using DVI, and the output with the maximum DVI value is returned as the final clustering result. In this way, parameters of the expansion procedure can be eliminated. Because the time complexity of DVIs is $O(|E|)$, adding the parameter determination processing to GDVA does not change the overall time complexity $O(|E| + |V| \log |V| + nc|V|)$ of the GDVA method.

4.4. Evaluating cluster validity indices

Evaluating and comparing cluster validity indices are challenging tasks. There is little work in the literature addressing these problems. Like evaluating clustering methods, evaluating cluster validity indices is subjective and does not generalize well because it depends on how much a data set conforms to the assumptions on cluster structures implied by each validity index. Although all cluster validations are
somewhat similar in the sense that they all try to reward clusters that contain similar objects and penalize clusters that include dissimilar objects, the actual formula of each index takes a different approach in measuring the quality of clusterings and emphasizes a different aspect of validation. A validity index formula can be regarded as an objective function that we impose on the clustering problem. In general, the definition of a validity index can be arbitrarily complicated so that no efficient optimization methods can find a global or even a local optimum solution. On the other hand, we can compute and compare the index value for some given clusterings easily so as to identify better solutions, thus a good validity index is the one that can help us accurately estimate the quality of clusterings in order to differentiate between good and bad clustering solutions. So we can say, when applying to a collection of clustering results, a good validity index will produce scores that correlate well with the scores of the similarities between the clusterings and the ground truth. If a better validity index value corresponds to a higher similarity between the clusterings and the ground truth, then the index values reflect well the quality of the clusterings. As we have seen before, the similarity between two clusterings can be measured by using ARI or NMI. Therefore we can evaluate the accuracy of a validity index on a data set based on the rank correlations between the index scores on a collection of clusterings and the ARI or NMI scores between those clusterings and the ground truth.

Spearman's rho (ρ) and Kendall's tau (τ) are two popular rank correlation coefficients in statistics [24]. They are used to measure the agreement between different orderings on the same set of objects. Spearman's rho calculates the correlation coefficient based on the differences between the ranks of the same objects in two orderings. On the other hand, Kendall's tau is computed based on the numbers of concordant and discordant pairs of objects in two orderings. A pair of objects is concordant if they are in the same order in both orderings; the objects are discordant if they have different orders in the two orderings. Spearman and Kendall rank correlation coefficients are more suitable than Pearson correlation when we evaluate validity indices because rank correlations measure a monotonic relationship between two variables, while Pearson correlation measures a linear dependence of two variables. In other words, Spearman and Kendall correlations will give a perfect score as long as validity values of two indices are monotonically related, while Pearson correlation does not give a perfect score if the relationship is not linear.
Figure 4.2 illustrates an example of evaluating the rank correlation between NMI and ARI on a sample data set. The sample data is randomly generated with a known clustering ground truth (see Section 5.1.1 for the model to create these random graphs). Several hundred different clustering results are generated and compared with the ground truth using NMI and ARI. We see a relatively strong correlation between the scores measured by NMI and ARI. The rank correlations between NMI and ARI shown in Figure 4.2 measured by Spearman's rho and Kendall's tau are 0.95 and 0.82, respectively.

![Figure 4.2. An example of the correlation between ARI (x-axis) and NMI (y-axis).](image)

To evaluate a validity index, we can use NMI or ARI scores as the ground truth for validity index values and compute the rank correlations between scores of the validity index (e.g., DVI₁, DVI₂, DVI₃) and the ground truth scores. A higher correlation implies that the validity index is a better indication of the quality of clusterings. Figure 4.3 shows an example of the correlations of DVI₁, DVI₂, DVI₃, and Silhouette (x-axis) with NMI (y-axis) based on the above sample data set. In this case, the Spearman rank correlations of DVI₁, DVI₂, DVI₃, and Silhouette with NMI are 0.89, -0.95, -0.97, 0.74, respectively. The Kendall rank correlations on the same data are 0.72, -0.82, -0.86, 0.58, respectively. In Figure 4.4, the correlations of DVI₁, DVI₂, DVI₃, and Silhouette (x-axis) with ARI (y-axis) on the same data are shown. In this case, the Spearman correlations are 0.85, -0.93, -0.93, 0.78, respectively. The Kendall correlations are 0.67, -0.78, -0.77, 0.64, respectively. This method of evaluation validity indices can be applied to any data set for which we have the clustering ground truth. In general, we see that DVI₂ and DVI₃ work relatively well with many synthetic and real data sets. As the computation for these density-based validity indices is very simple and fast, they can be useful for graph-based clustering methods to determine automatically the better clustering results when tuning parameters.
In Tables 4.1 and 4.2, we show the Spearman and Kendal rank correlations between six indices based on graphs generated randomly with a known clustering ground truth (see Section 5.1.1). Values displayed in these tables are the averages of the correlation values between the indices on 50 random graphs. In this experiment, we can see that DVI3 scores are more strongly correlated with the similarities to the ground truth measured by NMI and ARI than the scores of DVI1, DVI2, and Silhouette.

<table>
<thead>
<tr>
<th></th>
<th>NMI</th>
<th>ARI</th>
<th>DVI1</th>
<th>DVI2</th>
<th>DVI3</th>
<th>Silhouette</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>1</td>
<td>0.87</td>
<td>0.74</td>
<td>−0.84</td>
<td>−0.91</td>
<td>0.58</td>
</tr>
<tr>
<td>ARI</td>
<td>0.87</td>
<td>1</td>
<td>0.46</td>
<td>−0.60</td>
<td>−0.71</td>
<td>0.69</td>
</tr>
<tr>
<td>DVI1</td>
<td>0.74</td>
<td>0.46</td>
<td>1</td>
<td>−0.97</td>
<td>−0.91</td>
<td>0.23</td>
</tr>
<tr>
<td>DVI2</td>
<td>−0.84</td>
<td>−0.60</td>
<td>−0.97</td>
<td>1</td>
<td>0.97</td>
<td>−0.35</td>
</tr>
<tr>
<td>DVI3</td>
<td>−0.91</td>
<td>−0.71</td>
<td>−0.91</td>
<td>0.97</td>
<td>1</td>
<td>−0.48</td>
</tr>
<tr>
<td>Silhouette</td>
<td>0.58</td>
<td>0.69</td>
<td>0.23</td>
<td>−0.35</td>
<td>−0.48</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 4.2. Average Kendall rank correlations between NMI, ARI, DVI₁, DVI₂, DVI₃, and Silhouette evaluated on clusterings of random graphs.

<table>
<thead>
<tr>
<th></th>
<th>NMI</th>
<th>ARI</th>
<th>DVI₁</th>
<th>DVI₂</th>
<th>DVI₃</th>
<th>Silhouette</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>1</td>
<td>0.71</td>
<td>0.55</td>
<td>−0.66</td>
<td>−0.74</td>
<td>0.42</td>
</tr>
<tr>
<td>ARI</td>
<td>0.71</td>
<td>1</td>
<td>0.33</td>
<td>−0.45</td>
<td>−0.54</td>
<td>0.52</td>
</tr>
<tr>
<td>DVI₁</td>
<td>0.55</td>
<td>0.33</td>
<td>1</td>
<td>−0.88</td>
<td>−0.76</td>
<td>0.16</td>
</tr>
<tr>
<td>DVI₂</td>
<td>−0.66</td>
<td>−0.45</td>
<td>−0.88</td>
<td>1</td>
<td>0.86</td>
<td>−0.25</td>
</tr>
<tr>
<td>DVI₃</td>
<td>−0.74</td>
<td>−0.54</td>
<td>−0.76</td>
<td>0.86</td>
<td>1</td>
<td>−0.34</td>
</tr>
<tr>
<td>Silhouette</td>
<td>0.42</td>
<td>0.52</td>
<td>0.16</td>
<td>−0.25</td>
<td>−0.34</td>
<td>1</td>
</tr>
</tbody>
</table>
Chapter 5. Clustering experiments and evaluations

GDVA is a general purpose clustering method which can be used for various problems with different types of data. This chapter studies experimental results of GDVA with different applications and data types. Section 5.1 shows evaluation results from controlled experiments with synthetic random graphs under different sparsity conditions. Here, GDVA is compared to normalized cut (NCut), Markov cluster algorithm (MCL), and affinity propagation (AP) in terms of accuracy and scalability. In Section 5.2, we apply GDVA to heterogeneous data including points on the plane, UCI data sets, gene expression data, and text. Finally, Section 5.3 demonstrates GDVA application on image segmentation which indicates GDVA can handle complex problems and is scalable. Segmentation results can be visually assessed as we find segments of image pixels that are nearby and similar in color and/or intensity.

5.1. Evaluating clustering methods using random weighted graphs

A challenging task in cluster analysis is to evaluate the performance of clustering methods. In reality, it is hard to assess the quality of clustering results as there is no ground truth. However, one can make use of experimental data sets in which data labels are known to evaluate results of a clustering method. Data labels are usually given by humans and used as the ground truth of clustering. A problem here is that the ground truth may be not unique because human judgment is subjective and there can be inconsistence between different individuals. However, a bigger problem is that even if we have a consistent and reliable ground truth, due to the nature of the clustering problem, every clustering method has to heuristically make some (different) assumptions about the structure of clusters, so a method can work well on one data set but poorly on another data set. Depending on the structure of the data chosen for experiments, one clustering criterion can be better or worse than others. Evaluations can be performed only on some specific data sets, so a method outperforming others on several data instances does not imply it will always work better than others. For example, the agglomerative hierarchical based on minimum distance will work extremely well if clusters are elongated and there is no noise and the gaps between adjacent clusters are much larger than
the gaps between adjacent elements in each cluster. Clearly, if a data set conforms to the assumption of a method, the method will perform better than others on that data set, however one can always find other data that complies and therefore works with another method better. So experimental comparisons based on a number of data sets do not prove if a method is superior or inferior to others. This is why there are different clustering methods in the literature, and in practice it is essential to determine which method is most appropriate for each specific data set by checking the methods’ assumptions against the data under consideration.

5.1.1. Random weighted graphs

We choose to evaluate clustering methods on randomly generated data so as to alleviate the issues of human subjectivity in evaluations because the ground truth of generated data is objectively known and unique. With generated graphs, we can control the separation between clusters and then evaluate clustering methods over a large number of data sets and a variety of graph structures. The result is a plot of curves of average accuracies rather than a few separated points if only several data sets were used. In addition, most existing public real data sets are not ready in a graph representation and need to be pre-processed and transformed to proximity graphs using some similarity measures. These pre-processing and transformation steps are non-trivial and depend on different assumptions and parameters. The evaluation results will not reflect accurately the quality of a clustering method if they are largely affected by arbitrary pre-processing and transformation steps and their associated parameters. So evaluating directly on graph data will avoid the influence of extra suppositions and parameters on the evaluation results.

![Figure 5.1. Weight matrix of a randomly generated graph of 100 nodes with four pre-specified clusters.](image)
We construct random proximity graphs of 100 nodes numbering from 1 to 100 with four clusters of sizes 10, 20, 30 and 40 as follows. Upon a blank weight matrix of size 100 by 100, the entries in the upper-right triangle are randomly generated from a uniform distribution on the interval [0, 1]. Then, we randomly set one fifth of the entries to zero so that the graph has approximately 4000 edges. We pre-specify that the first, second, third, and fourth clusters consist of the first 10 nodes, the next 20 nodes, the next 30 nodes, and the last 40 nodes, respectively. Next, we keep intra-cluster edges on the generated weight matrix unchanged but decrease weights of inter-cluster edges (Figure 5.1) by multiplying the entries in the inter-cluster region by a parameter $p \in [0, 1]$. Thus, we can adjust the separation between the clusters by varying the value of $p$. If $p = 0$, the clusters are completely separated, while if $p = 1$, the graph is totally random and the clusters are not separable. Finally, small noise is added and the entries of the lower-left triangle are copied down from the upper-right triangle so that the matrix becomes symmetric. In Figure 5.1, (b) illustrates a weight matrix generated using $p = 0.4$. High values are represented by low intensities. We can see that nodes are connected to other nodes in their clusters more strongly than to nodes of other clusters. Since the ground truth is known, the accuracy of clustering results can be evaluated based on NMI and ARI.

5.1.2. Comparisons of GDVA, NCut, MCL, and AP

![Figure 5.2. Comparison of clustering accuracy of GDVA, NCut, MCL, and AP using NMI.](image-url)
We compare GDVA with NCut, MCL, and AP methods, which have been implemented by their authors in C language [84, 85, 86], so these codes are accurate and highly optimized for performance. In AP, similarity values are assumed to be between $-\infty$ and 0, so we subtract one from the weight matrix entries to convert similarities to the interval [-1, 0]. Figures 5.2 and 5.3 show the average accuracy based on NMI and ARI for clustering results of GDVA, NCut, MCL, and AP using $p \in \{0.1, 0.2, ..., 0.9\}$. For each value of $p$, we learn the best parameters for each method by generating a training set of 100 random graphs. We apply clustering methods to the training set using different parameter values and evaluate the clustering results using NMI in order to find the parameter settings that maximize the average value of NMI. Then, we generate a test set of 200 random graphs and calculate the average accuracy of the clustering results for the test set using the settings learned previously. In Figures 5.2 and 5.3, when $p$ is increased, the clusters are less separable; so the averages accuracy decreases. Generally, AP does not work well as it focuses mainly on minimizing distances between each exemplar and its neighbors, but not on minimizing the total of distances of all elements within a group and simultaneously maximizing the distances between groups. Note that random graphs generated as above do not satisfy the core assumption. In the next section, we show GDVA producing better results for data whose clusters have a core region composed of nodes strongly interconnected by edges with a high and comparable weight.

Table 5.1 shows the parameters used by the three methods for each value of $p$. We use the same parameters for GDVA in all the cases. In Chapter 6, we will show the accuracy of results with different values for parameters of GDVA. For NCut, we have to specify the correct number of clusters. For MCL, the inflation parameter $I$ has to be increased when the clusters are less separable (i.e., $p$ is increased). In AP, preference
values of all the nodes are set to the same value since we do not have prior knowledge of possible exemplars.

Table 5.1. Parameters of GDVA, NCut, MCL, and AP for different values of the inter-cluster parameter $p$.

<table>
<thead>
<tr>
<th>$p$</th>
<th>GDVA $\delta$</th>
<th>GDVA $\beta$</th>
<th>NCut #clusters</th>
<th>MCL $I$</th>
<th>AP preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.25</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>$-3.8$</td>
</tr>
<tr>
<td>0.2</td>
<td>0.25</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>$-3.8$</td>
</tr>
<tr>
<td>0.3</td>
<td>0.25</td>
<td>1</td>
<td>4</td>
<td>3.5</td>
<td>$-3.8$</td>
</tr>
<tr>
<td>0.4</td>
<td>0.25</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>$-3.8$</td>
</tr>
<tr>
<td>0.5</td>
<td>0.25</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>$-3.8$</td>
</tr>
<tr>
<td>0.6</td>
<td>0.25</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>$-3.8$</td>
</tr>
<tr>
<td>0.7</td>
<td>0.25</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>$-3.8$</td>
</tr>
<tr>
<td>0.8</td>
<td>0.25</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>$-3.8$</td>
</tr>
<tr>
<td>0.9</td>
<td>0.25</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>$-3.8$</td>
</tr>
</tbody>
</table>

Table 5.2. Comparison of execution times of GDVA, NCut, MCL, and AP on five graphs with different sizes.

<table>
<thead>
<tr>
<th>Graph size</th>
<th>GDVA (seconds)</th>
<th>NCut (seconds)</th>
<th>MCL (seconds)</th>
<th>AP (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>V</td>
<td>$ nodes</td>
<td>$</td>
<td>E</td>
</tr>
<tr>
<td>1,000</td>
<td>200,000</td>
<td>0.1</td>
<td>0.2</td>
<td>13</td>
</tr>
<tr>
<td>2,000</td>
<td>1,000,000</td>
<td>0.7</td>
<td>1.3</td>
<td>80</td>
</tr>
<tr>
<td>3,000</td>
<td>2,000,000</td>
<td>1.3</td>
<td>2.9</td>
<td>190</td>
</tr>
<tr>
<td>4,000</td>
<td>3,000,000</td>
<td>2</td>
<td>4.5</td>
<td>330</td>
</tr>
<tr>
<td>5,000</td>
<td>5,000,000</td>
<td>3</td>
<td>8.5</td>
<td>570</td>
</tr>
</tbody>
</table>

Table 5.2 shows execution times of GDVA, NCut, MCL, and AP on a PC with 3GHz Core2 Duo CPU. Five graphs were generated randomly with pre-specified numbers of nodes and edges. GDVA was programmed in Matlab with some external functions in C. NCut, MCL, and AP were implemented in C by their authors [84, 85, 86]. MCL is very slow on graphs that have many edges, AP is faster than MCL but still very slow compared to GDVA and NCut. For medium and large graphs, GDVA is much faster and more scalable than NCut. In fact, the execution times of GDVA shown in Table 5.2 are mainly spent for
determining parameter values. If all four input parameters are provided, the time required for clustering the last graph with five thousand nodes and five million edges is only about 0.3 seconds.

5.2. Clustering experiments with heterogeneous data

In this section, we illustrate the GDVA clustering method with experiments on synthetic data points on the plane, several UCI data sets, gene expression data, and documents. The next section will show additional experiments with digital images. Because these data sets are not graph-based but feature-based data, a common pre-step is to compute pairwise similarities for the proximity graphs as discussed in Chapter 2. After that, GDVA is applied to find clustering solutions. To evaluate the results, we can compare them to the ground truth from known classes of data objects using NMI or ARI.

5.2.1. Visual examples with data points on the plane

First we illustrate the execution of the GDVA method by visual examples in two-dimensional Euclidean space. Using points on the plane is convenient for visualization, however as the method partitions proximity graphs, the points have to be transformed into a graph representation. This step is crucial because the data can be distorted by the transformation process. Traditionally, each point of the data set is represented by a node of the proximity graph. Edge weights represent the similarity between two points. Edge weights are computed based on Euclidean distances between the points. Many functions can convert distance measures, which represent dissimilarity, into similarity measures. Here, we use a linear weight function defined in (2.2), where \( d_{ij} \) is computed as the Euclidean distance between two points \( i \) and \( j \). For this weight function, the proximity graph is practically a complete graph as all the edges have a positive weight except the edge connecting the two most distant points. Two advantages of this function are: (i) it does not require any parameters, and (ii) it is more straightforward for our visual evaluation because of the linear relation between spatial distances and edge weights. If exponential functions such as the one defined in (2.3) were used, the data would be transformed with more distortion, even though we could adjust parameters to make clusters more compact in the proximity graph in order to make it easier to get a good clustering.
Once the weight matrix is computed, we can apply any clustering algorithm for weighted graphs. In Figure 5.4, (a) shows 80 points in a two-dimensional space generated from three normal distributions with different means and variances. Based on positions of the points on the plane, we can visually divide the points into three clusters of size 20, 25, and 35. Figure 5.1 (b) shows the sequence of minimum densities produced by the sequence-construction procedure on the graph built with the weight function (2.2) for the
point set in (a). Since the clusters of the point set are separated well, we can see that the $D_t$ sequence has three segments corresponding to the three clusters. To compare the difference when no clusters present, Figures 5.5 (b) and 5.6 (b) show the sequences of minimum densities for the sets of points given in Figures 5.5 (a) and 5.6 (a), which are generated from a normal and a uniform distribution, respectively. As these point sets do not have multiple clusters in them, the $D_t$ value tends to increase until the last iteration. In Figure 5.5 (a) we also show the order of the nodes in the $M_t$ sequence by the point intensities. Points with the lowest intensities correspond to nodes at the end of the sequence. It can be seen that the $M_t$ sequence proceeds from outer sparse regions to the central dense region.

Based on the sequences of minimum densities and corresponding nodes, the core-node-extraction procedure extracts a core set that consists of points at the center of clusters which has a higher density than at cluster boundaries. Depending on the values of $\delta$ and $\beta$, we can obtain a larger or smaller core set. In (c) of Figure 5.4, we show the core set for the point set in (a) when $\delta = 0.8$ and $\beta = 2$. It has five points and actually comprises three groups corresponding to three descending sections in the $D_t$ sequence. These groups include two groups of two points from the first and second clusters and one group of one point from the last cluster. If we decrease $\delta$, the core set will be more expanded than in the previous case as shown in (d) and (e). Nevertheless, for any of these core sets, it is clear that partitioning a core set is easier than partitioning the original data because we have a small number of core points and they are from centers of clusters so they are separated much better than the points of the original set. The cluster core identification and cluster-core-expansion procedures partition the core sets into three clusters' cores and expand the cores to produce the desired clustering result as in (f).
In Figure 5.7 (a), we show a set of points on the plane which has two clusters and a few noise points. Two core regions can be seen in the clusters. As before, a weighted graph is built for the data set using weight function (2.2) and the method is applied to the graph. GDVA identifies six core nodes in the core regions as shown in Figure 5.7 (b). The set of core nodes comprises two groups corresponding to the two clusters' cores. The method automatically finds and expands these cores to produce a clustering with two clusters.

By this example, we see that GDVA is not affected by outliers because these points are excluded from the set of core nodes. Therefore, this is an advantage compared to noise-sensitive methods such as agglomerative hierarchical. For example, partitioning the set of data points in Figure 5.7 (a) into two clusters by agglomerative hierarchical methods will result in one of the clusters containing the three outlier
points at the bottom and the other cluster containing the rest of the data. This noise sensitiveness is due to the uses of maximum and minimum operations when grouping clusters as well as computing the similarity between clusters (see Section 2.3.1).

![Figure 5.7](image)

Figure 5.7. (a) A set of points with two clusters, cluster cores, and a few noisy points. For agglomerative hierarchical methods, bi-partitioning the data set produces two clusters, one of which contains the three outlier points at the bottom, and the other cluster contains the rest of the data. (b) Six core points found by the graph density variation analysis (c) Clustering result.

5.2.2. UCI data sets

In this section, we apply GDVA to several publicly available data sets in the UCI machine learning repository [87]. The data sets in UCI are of various types but most of them are multivariate and have class labels associated with data objects. If a data set has all the features of numerical types, it is representable in a multi-dimensional space and hence we can apply vector-based clustering methods like K-means on this data set. The class labels of data objects then can be used as the ground truth in order to measure the quality of clustering results through their similarities to the ground truth by using indices such as NMI or ARI (Section 2.4). For GDVA experiments, we transform data sets into a graph representation by building proximity graphs. As in other experiments with multi-dimensional data sets, we use the weight function (2.2) described in Chapter 2, which is a linear transformation from Euclidean distances to similarity measurements and does not require parameter specification. Table 5.3 shows the NMI scores for the best clustering results produced by K-means and GDVA on several UCI data sets, which are numerical multivariate data from various fields and have different numbers of attributes and instances. The results are obtained by varying parameters to determine the best scores of GDVA and K-means for each data set. In
general, GDVA is able to generate good clusterings compared to K-means, and the number of clusters is not required to be specified. The qualities of the results by GDVA over the parameter space for the Iris data set is shown in Chapter 6.

Table 5.3. Accuracy measured by NMI for K-means and GDVA results on several UCI data sets.

<table>
<thead>
<tr>
<th>UCI data set</th>
<th>K-means</th>
<th>GDVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>0.76</td>
<td>0.76</td>
</tr>
<tr>
<td>Optical digits</td>
<td>0.74</td>
<td>0.79</td>
</tr>
<tr>
<td>Pen-based digits</td>
<td>0.69</td>
<td>0.73</td>
</tr>
<tr>
<td>Ecoli</td>
<td>0.62</td>
<td>0.64</td>
</tr>
<tr>
<td>Segmentation</td>
<td>0.60</td>
<td>0.63</td>
</tr>
<tr>
<td>Libras</td>
<td>0.59</td>
<td>0.57</td>
</tr>
<tr>
<td>Wine</td>
<td>0.43</td>
<td>0.42</td>
</tr>
<tr>
<td>Glass</td>
<td>0.42</td>
<td>0.37</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.27</td>
<td>0.27</td>
</tr>
</tbody>
</table>

5.2.3. Gene expression data

In bioinformatics, clustering techniques have been commonly used for analyzing gene expression data. Clustering applications to gene expression analysis were demonstrated using different methods [3, 6, 30, 79]. Particularly, cancer classification results may be improved by using cluster analysis results to possibly discover previously unknown cancer classes and subclasses [30]. In this section, we show clustering results for two gene expression cancer data sets that are publically available.

1. The colon cancer data set

Based on the colon cancer data set, we try to tackle the problem of tissue clustering which aims to find connections between gene expressions and statuses of tissues and to see if it is possible to predict the status of a tissue based on its gene expressions. The data set used in this experiment is available at [89]. The data contains 62 samples including 40 tumor and 22 normal colon tissues. Each sample consists of a vector of 2000 gene expressions. We will set aside the sample labels (tumor/normal) and cluster the samples based on the similarities between their gene expressions. Ideally, we want to partition the sample set into two clusters such that one contains only tumor tissues and the other contains only normal tissues. The proximity
graph constructed from the gene expression vectors is a complete graph of 62 nodes. Since relative values are more important than absolute values in gene expressions, edge weights that reflect the pairwise similarities of samples are computed based on the Pearson correlation coefficient [24]. Specifically, the weight function is defined by:

$$w_{ij} = \frac{1}{2000} \sum_{k=1}^{2000} \frac{1}{s_i s_j} (i_k - m_i)(j_k - m_j),$$

where $i_k$ and $j_k$ denote the gene expression vectors of samples $i$ and $j$, respectively; $m_i$, $m_j$, $s_i$, $s_j$ denote the means and standard deviations of $i_k$ and $j_k$. Steps 1 and 2 of the method identify 5 core nodes. The dendrogram of these core nodes shown in Figure 5.8 can be separated into two groups. One of which contains three nodes and the other has two nodes. Thus, cutting the dendrogram at the height of 0.1 produces two cluster cores and then expanding these cluster cores yields two clusters. One has 37 samples consisting of 35 tumor and two normal tissues. The other contains 25 samples consisting of five tumor and 20 normal tissues. So the error rate in this case is 7/62 and the NMI measurement of the similarity between the clustering result and the normal-tumor ground truth is 0.49. When looking more carefully into the set of core nodes, we see that it actually contains three descending sections: two sections of two nodes and one section of one node. Therefore, another possibility to partition the core set is to produce three cluster cores instead of two. In this case, expanding cluster cores yields three clusters of 15, 26, and 21 samples. The first cluster contains 14 tumor and one normal tissues, the second cluster has 23 tumor and three normal tissues, and the last cluster includes 18 normal and three tumor tissues. So if we merge the first and second clusters, we get a clustering of two clusters with an error rate of 7/62. The error rate obtained here using GDVA is comparable to previously reported results on the same data set. In [3], a deterministic-annealing algorithm was used to organize the data in a binary tree and then partition it into two clusters, one of which has 19 normal and five tumor tissues, the other has 35 tumor and three normal tissues (error rate = 8/62). In [6], another clustering method called CAST (Cluster Affinity Search Technique) was applied to the same data set and partitioned it into 6 groups. Joining three tumor rich groups together yields a cluster with 36 tumor and three normal tissues. The other three groups contain 19 normal and four tumor tissues. Thus the error rate is 7/62 which is similar to previous results.
2. The leukemia data set

Golub et al. demonstrated the application of clustering on a data set of gene expression levels of leukemia bone marrow samples [30]. These samples were diagnosed as either acute lymphoblastic leukemia (ALL) or acute myeloid leukemia (AML). Thus, one can try to find clusters of samples in the data and compare with these two classes of leukemia. If the clusters found are analogous to the two classes, then the samples of each class are more similar to each other than to the samples of the other class. In other words, if clustering can explore the structure of the data and produce meaningful groups of leukemia samples, it may discover new tumor subtypes. In [30], an interesting result was obtained by clustering a data set of 38 samples consisting of 27 ALL and 11 AML, where each sample is represented by an expression vector of 7129 genes. By using self-organizing maps (SOM), the data set was partitioned into two clusters, in which, one cluster contained 25 samples including 24 ALL and one AML, and another cluster contained 13 samples including ten AML and three ALL. Thus, this clustering result is relatively close to the classes of ALL and AML samples with an error rate of 4/38. The data set is available at [90].

In this example, we explore the above leukemia data using GDVA. First, a complete proximity graph is constructed from the affinities between pairs of samples measured by cosine similarity [63]. Negative affinity values, if they exist, are set to zero. Then, we investigate the clustering results while varying parameters $\delta$ and $\beta$. For all these settings, we find that the method produces only two possibilities of clustering, one of which has two clusters and the other has three clusters. The clustering with two clusters partitions 38 samples into one bigger cluster, which contains 26 samples including 25 ALL and one AML, and one smaller cluster which contains 12 samples including ten AML and two ALL. The clustering with
three clusters is just a fine-grained partitioning of the previous one, in which the bigger cluster with 26 samples is divided into two smaller clusters of 19 and seven samples. Therefore, the clustering result obtained here has an error rate of 3/38 and is comparable with the result reported in [30]. The NMI measurement of the similarity between the result and the ALL-AML ground truth is 0.57 in this case.

5.2.4. Document data set

Document grouping is an application of clustering techniques in information retrieval, where we try to group similar documents together in order to organize a text collection and to extract meaningful categories [50, 77]. In this example, we use a subset of OHSUMED [36] which is a document collection consisting of references from medical journals in MEDLINE – an on-line medical information database. The OHSUMED corpus was used in the filtering track in the Text Retrieval Conference (TREC-9) [83]. In this data set, each reference contains information about a medical article such as the author, the title, the abstract, etc. The data set also includes 63 different queries which were asked by users of MEDLINE. Each query is related to a set of relevant references judged by a group of physicians. The relevance of a reference to a query is given as either definitely relevant or possibly relevant. Therefore, the abstracts of references that are relevant to the same query are more similar to each other than to the abstracts of references relevant to other queries. We extract a subset of the OHSUMED collection based on the first five queries. Specifically, we retrieve from OHSUMED a group of relevant references for each of the first five queries. Overall, our subset contains 200 references consisting of five groups of sizes 20, 30, 40, 50, and 60. We can therefore apply clustering methods to this data set and validate the results based on the known group partitioning.

The similarity between the abstracts of the references is computed as follows. First, stop words are removed from the abstracts. Then, we perform the Porter’s snowball stemming to reduce the words to their stem [57]. Next, unigram and bigram of terms are extracted to form dimensions of a vector feature space [60]. Each abstract is represented by a vector whose values are computed by the classic tf-idf weighting scheme [61]. We measure the affinity between every pair of vectors by the commonly used cosine similarity [63] and thus get a complete proximity graph for the data set (the cosine similarity is between zero and one because all the tf-idf values are non-negative). Finally, we apply GDVA to this graph and
obtain a clustering for the set of abstracts. The quality of the clustering result is evaluated by comparing the output clusters with the five known reference groups. Using $\delta = 0.4$ and $\beta = 2$, the method produces five clusters of sizes 20, 35, 40, 45, and 60. In this result, we find that the clusters of sizes 20, 40, and 60 are identical to three groups of references. The cluster of size 35 contains the group of 30 references plus five references from the group of 50 references. Thus, the method yields a good clustering with an error rate of $5/200$. The NMI measurement of the similarity between the clustering result and the ground truth grouping in this case is 0.95. In Chapter 6, we will show the quality of clustering results produced by using different values in the parameter space for this data.

5.3. Experimental results on image segmentation

Image segmentation is an important yet challenging problem in image processing. Depending on user objectives, different definitions and criteria are proposed and employed for image segmentation. In this chapter, we apply the GDVA clustering method to segment a grayscale or color image into a set of disjoint regions such that each region is composed of nearby pixels having a similar intensity or color. In order to build a proximity graph for an image, each pixel is represented by a node of the graph and the edge weight between two nodes is computed based on the similarity between two corresponding pixels using a similarity function which takes into account properties of corresponding pixels such as their location, brightness, and color. Applying a graph clustering algorithm to the proximity graph of an image will partition it into subgraphs, each of which corresponds to a group of pixels similar to each other but dissimilar to pixels of groups corresponding to other subgraphs. Thus, each cluster in the graph corresponds to a segment in the image. We can evaluate and validate segmentation results by visual inspection and assessment by checking if each segment contains nearby pixels with similar intensity or color. Note that the goal of the image segmentation performed here is not the same as the way that humans usually do the task because segments found by humans usually correspond to recognizable objects. So a segment can contain pixels with very different intensities and colors if they happen to be part of the same object. To detect objects in an image, one may use other computer vision techniques based on various cues such as color, contrast, and texture.
5.3.1. Graph-based representation for images

Given a digital image, we can build a proximity graph \( G = (V, W) \), where each node of \( V \) represents a pixel, the weight \( w_{ij} \) of the edge between two nodes corresponding to pixels \( i \) and \( j \) reflects the likelihood that \( i \) and \( j \) belong to the same segment in the image. Since we want to group nearby pixels that have a similar intensity/color, weights of graph edges are computed by a weight function based on the location and intensity/color of neighboring pixels. In other words, pixel proximities are determined by pixel features (brightness or colors) and pixel positions in an image. Specifically, the weight function for grayscale images is defined as:

\[
w_{ij} = \begin{cases} 
1 - \frac{|I(i) - I(j)|}{s} & \text{if } |I(i) - I(j)| < s \text{ and } \text{dist}(i, j) < r, \\
0 & \text{otherwise},
\end{cases}
\]  

(5.1)

where \( I(i) \) and \( I(j) \) denote the intensities of pixels \( i \) and \( j \), respectively. \( \text{dist}(i, j) \) denotes the Euclidean distance in pixels between \( i \) and \( j \). An edge weight is zero if the distance between two corresponding pixels is more than \( r \) pixels or their intensity difference is greater than \( s \). We find that \( r \in [5, 15] \) and \( s \in [0.1, 0.3] \) generally produce good results. For color images, we use a similar function where the difference between intensities of pixels \( i \) and \( j \) is replaced simply by the distance between their colors in RGB or HSV color space, i.e.:

\[
w_{ij} = \begin{cases} 
1 - \frac{\|C(i) - C(j)\|_2}{\sqrt{3}s} & \text{if } \|C(i) - C(j)\|_2 < \sqrt{3}s \text{ and } \text{dist}(i, j) < r, \\
0 & \text{otherwise},
\end{cases}
\]  

(5.2)

where \( C(i) \) and \( C(j) \) denote 3-component vectors representing the colors of pixels \( i \) and \( j \) in RGB/HSV color space, respectively. Note that \( \|C(i) - C(j)\|_2 \in [0, \sqrt{3}] \). With the weight functions above, an edge exists between two nodes only if the distance between the corresponding pixels is less than \( r \) pixels, as a result each node has approximately \( \pi r^2 \) neighboring nodes. The proximity graph is very sparse since \( \pi r^2 \ll |V| \). High-weight edges exist between nodes whose corresponding pixels are close to each other and have similar intensity/color. Therefore, pixels inside a segment with homogeneous intensity/color (i.e., the inner or core region of a segment) will have corresponding nodes strongly connected with their neighboring
nodes. On the other hand, pixels located at boundaries of the regions have dissimilar neighbor pixels, so their corresponding nodes are weakly connected to each other.

The weight functions in (5.1) and (5.2) are products of two linear functions representing the intensity/color similarity and the nearness between pairs of pixels. In the literature, exponential weight functions are more popularly employed [45, 46, 66], e.g., for grayscale images:

\[
W_{ij} = \begin{cases} 
    e^{-\frac{(I(i)-I(j))^2}{s^2}} \left( \frac{dist(i,j)}{\sigma} \right)^2 & \text{if } dist(i, j) < r, \\
    0 & \text{otherwise},
\end{cases}
\]

and for color images:

\[
W_{ij} = \begin{cases} 
    e^{-\frac{|C(i)-C(j)|^2}{s^2 \sqrt{3}}} \left( \frac{dist(i,j)}{\sigma} \right)^2 & \text{if } dist(i, j) < r, \\
    0 & \text{otherwise},
\end{cases}
\]

where \( s, \sigma, \) and \( r \) are control parameters playing a role similar to the parameters of (5.1) and (5.2). However, compared to (5.3) and (5.4), the linear functions in (5.1) and (5.2) have three advantages similar to the linear function (2.2) that transforms multi-dimensional data to proximity graphs: (i) the number of parameters is reduced compared to exponential weight functions, so investigating and tuning parameter settings can be performed more easily, (ii) it is more straightforward for our visual evaluation because of the linear dependence between edge weights and pixel intensities and positions, and (iii) the computational cost is less expensive. In fact, (iii) is an important advantage because the proximity graphs for images are usually very big graphs with a large number of nodes and edges.

A good result for a graph-based segmentation method obviously depends on whether or not segments of an image are translated into well-separated clusters of the proximity graph. The settings for parameters \( s \) and \( r \) are therefore very important as they determine how images are transformed into proximity graphs. The value of \( s \) is a trade-off between the similarity of intensity/color of pixels in the same segment and the dissimilarity of intensity/color of pixels belonging to different segments. A higher value for \( s \) would allow a higher tolerance for differences of pixel intensity/color within each segment, but it would then be harder to distinguish two segments that have a similar average intensity/color. The setting of \( s \) are dependent on the
contrast of the image, low contrast images may use a smaller value, while high contrast images may use a larger value of $s$. Parameter $r$ specifies how the spatial information is incorporated into the weight function. They determine the likelihood that neighbor pixels belong to the same segment. Higher values of $r$ make a segment span to greater distances over regions of heterogeneous intensity/color. This is a trade-off between detecting weakly separated segments and not breaking a large segment having some heterogeneous regions inside into smaller parts. We may need to increase the value of $r$ for larger images because of the likelihood that large images can contain large segments. In our experiments, we see that $s = 0.2$ and $r = 10$ work well across a wide range of image types.

5.3.2. Image segmentation by GDVA

<table>
<thead>
<tr>
<th>SEGMENTATION(Grayscale or color image $I$, $s$, $r$, $\delta$, $\beta$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build a proximity graph $G$ for input image $I$ based on function (5.1) or (5.2) and parameters $s$ and $r$.</td>
</tr>
<tr>
<td>Apply GDVA to $G$ with parameters $\delta$ and $\beta$.</td>
</tr>
<tr>
<td>Convert the clustering result to an image segmentation result.</td>
</tr>
<tr>
<td><strong>Return</strong> the segmentation result.</td>
</tr>
</tbody>
</table>

Figure 5.9. Image segmentation procedure.

In this section, we describe the segmentation procedure which partitions an image by building and then clustering its proximity graph using the GDVA method. The segmentation procedure consists of three steps as described in Figure 5.9. The first step is to build a proximity graph from an input image using functions (5.1) or (5.2). The next step uses GDVA to obtain a clustering for the proximity graph. The last step converts the clustering result to image segmentation. Input parameters $s$ and $r$ are for building the graphs while parameters $\delta$ and $\beta$ are for GDVA. Other parameters of GDVA can be fixed because the structures of proximity graphs are specialized, namely they are grid graphs and very sparse. Besides, the edge weights are between zero and one and pixels of each core have very similar intensities/colors and also locate very close to each other, so the corresponding core nodes are connected by edges of weights near one. On the other hand, pixels belonging to cores of different segments are usually far apart and dissimilar in intensities/colors, so they are not connected or connected by very weak edges. Therefore, we can simply set $\theta$ to 0.5 to remove all weak edges and extract each connected component in the core graph as a cluster core.
We also typically set $\lambda$ to 0.5, so a pixel is marked as uncertain if its similarity to the second nearest segment is greater than half of its similarity to the nearest segment, where the similarity is computed as the average weight of edges linking a pixel with a segment. Uncertain pixels are located near the boundaries between segments where pixels have different intensities/colors. When segments in an image are well visible, clusters in the proximity graph will be very well-separated as the graph are grid and very sparse. In this case, the number of uncertain pixels is very small and the value of $\lambda$ does not have much effect on the segmentation result.

![Figure 5.10](image.png)

**Figure 5.10.** (a) A 256×256 grayscale image of Matlab. (b) Core pixels and the order of image pixels in the sequence of density variation. (c) Segmentation result. (d) One segment with its core pixels and the order of its pixels in the sequence. The sequence proceeds from boundary pixels to the core pixels.

Figure 5.10 shows an image segmentation example using $r = 10$ and $s = 0.2$. (b) shows the order of nodes in the sequence of density variation for the image in (a). Here, brighter pixels correspond to nodes towards the end of the sequence. In each segment, the sequence proceeds from boundary pixels to core pixels. Using $\delta = 0.95$ and $\beta = 4$, the core-node-extraction procedure extracts a small number of core pixels inside homogeneous regions. The core graph is very small and composed of several groups as shown in (b). These groups are easily identified as connected components by the cluster-core-identification procedure. The segmentation result in (c) is produced by the cluster-core-expansion procedure where different segments
are shown by different colors. We can see that each segment consists of nearby pixels with similar intensities.

![Image](image.png)

**Figure 5.11.** Segmentation results on noise-added images.

To illustrate the performance of GDVA when noise is present, Figure 5.11 shows segmentations on three images created by adding noise to the image in Figure 5.10 (a). The intensity of each pixel is added by a value randomly generated from a normal distribution $N(\mu = 0, \sigma)$, where $\sigma$ takes on 0.1, 0.2, and 0.3 for the left, middle, and right image, respectively. The segmentation results are obtained by the same settings as before (i.e., $r = 10$, $s = 0.2$, $\delta = 0.95$, $\beta = 4$).

![Image](image.png)

**Figure 5.12.** Segmentation results of the image in Figure 5.10 (a) with $\delta = 0.99$, 0.9, and 0.8.
Regarding the effect of parameters $\delta$ and $\beta$ when applying GDVA to image segmentation, $\beta$ is an integer having the role of eliminating noisy pixels in the potential core pixels that have a high rate of decrease of density. Usually we use $\beta \in \{3, 4\}$ to exclude small cores which represent weak clusters. So the main parameter here is $\delta$. Changing $\delta$ can result in increasing or decreasing the number of segments. In general, we set $\delta$ to a value higher than 0.8. Because proximity graphs of images are very sparse, clusters here are very well-separated and therefore we use high values of $\delta$ and $\beta$ to get the strong clusters only and avoid unreliable core pixels. For images that contain a mixture of strong and weak segments, we can adjust $\delta$ to obtain a coarse-grained or fine-grained segmentation. When $\delta$ is decreased, we get more core pixels from weak segments. Consequently, the number of segments is increased and the segmentation result is more fine-grained as more weak segments emerge. Figure 5.12 shows segmentations for the image in Figure 5.10 (a) with three $\delta$ values: 0.99, 0.9, and 0.8 (other parameters are unchanged). Similarly, Figure 5.13 (b), (c), and (d) show the segmentations for the grayscale image in (a) with different values of $\delta$. Clearly, higher $\delta$ values produce a more coarse-grained segmentation and vice versa. These examples demonstrate that core pixels can be arranged in an order such that the pixels having a high rate of decrease in the sequence of density variation likely belong to the cores of strong segments.

Figure 5.14 shows examples of image segmentation by GDVA on natural color images of the Berkeley image data set [88]. Here, we use images that have little texture and contain well-distinguished objects. So we can visually check the segmentation results to see how close to image objects the segments are. Due to the shape and the variance in brightness/color of an object, we usually see that combining several segments of the result can create a match with an object of the image.
In this experiment, we compare the results of GDVA and NCut methods on segmenting images. MCL and AP methods are not suitable for image segmentation experiments because they are too slow for proximity graphs of images which are usually very large. Also, MCL and AP are not likely able to produce satisfactory results for the image segmentation task. So they are not included in the comparison. NCut has

Figure 5.14. Image segmentation examples on several natural color images of the Berkeley image data set.
been applied to computer vision problems such as image and video segmentation [64, 65], and functional brain imaging studies [37]. NCut requires the exact number of clusters \( k \) to be specified. This is problematic in image segmentation because the number of segments is highly variable from image to image. The number of segments of an image is not something that we know or think about in the first place, but rather it is a natural result of the perception process. Using GDVA, we do not specify the number of clusters but can adjust the granularity of segmentation results. A more serious problem with minimizing the NCut criterion function (2.4) is that the normalizing factors of the function make the method favor cutting the graph into clusters of similar sizes. As a result, small clusters are left out easily while large clusters are often split up into smaller parts. For instance, the image in Figure 5.15 (a) contains two regions: the oval at the center and the outside region. If (a) is segmented by a two-way NCut, we get the segmentation in (b) where the cut divides the image into two segments of similar size. It fails to find the central oval as one segment because of the disparity between the number of pixels inside and outside the oval. On the other hand, GDVA finds the correct segmentation even if noise is added as shown in (c), (d), (e) and (f). Figure 5.16 shows another example where the grayscale image (a) is segmented into 2, 3, 4, 5, 6 and 7 segments by NCut. The segments in the results are of roughly similar size as shown in (g), (h), (i), (j), (k), and (l). Moreover, different values of \( k \) produce very different segmentations, i.e., the result is not stable when the parameter setting is changed. For the same image, GDVA produces better results as shown in (b), (c), (d), (e), and (f), where we use \( \beta = 3 \) and \( \delta = 0.99, 0.97, 0.95, 0.93, \) and 0.9, respectively. Note that higher values of \( \delta \) yield more coarse-grained segmentations with less number of segments, but the results are very consistent in their sensitivity to perturbations of the different parameter settings. In Figure 5.17, the second and third columns show the segmentation results of GDVA and NCut, respectively, for the grayscale images in the first column. Again, GDVA yields much better results than NCut which tends to break large segments into several parts of similar sizes.
Figure 5.15. (b) shows the segmentation result by NCut on the image in (a). Images (c) and (e) are created by adding different kinds of noise to (a). (d) and (f) show the segmentations by GDVA on (c) and (e), respectively.

Figure 5.16. Segmentations with different parameter settings. (a) is a grayscale image. (b), (c), (d), (e), and (f) show segmentations by GDVA with $\delta = 0.99, 0.97, 0.95, 0.93$, and 0.9, respectively. (g), (h), (i), (j), (k), and (l) show segmentations of NCut on the same proximity graph with $k = 2, 3, 4, 5, 6$ and 7, respectively.

Figure 5.17. Comparison of segmentation results of GDVA and NCut.
Regarding algorithm complexity, Table 5.4 shows the execution times on different image proximity graphs of GDVA and NCut using a PC with a CPU of Core2 Duo 2.4GHz and 2GB RAM. Note that the time to build proximity graphs is not included. It can be seen that the execution time of GDVA is roughly linear to the number of edges of the graphs and is much faster than NCut. The running time of NCut is dependent on the parameter of the number of clusters, of which a higher value usually requires a longer running time.

The execution times for NCut shown in Table 5.4 are the average time for partitioning the graphs into two, three, and four segments. For the last three cases where the graph has more than 105 thousand nodes and 18 million edges, NCut fails to execute because of an "Out of memory" error. An additional advantage of GDVA over NCut is that if we need to change parameter settings, we do not need to re-execute the first step of GDVA, while changing the value of parameter \( k \) for NCut will result in re-computing the cut from scratch.

Table 5.4. Execution times of the NCut and GDVA on image's proximity graphs of different sizes.

<table>
<thead>
<tr>
<th>Image’s proximity graph size</th>
<th>NCut (seconds)</th>
<th>GDVA (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#nodes (thousands)</td>
<td>#edges (millions)</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>4.7</td>
<td>9</td>
</tr>
<tr>
<td>45</td>
<td>7.1</td>
<td>16</td>
</tr>
<tr>
<td>60</td>
<td>10.7</td>
<td>30</td>
</tr>
<tr>
<td>75</td>
<td>12.2</td>
<td>60</td>
</tr>
<tr>
<td>90</td>
<td>14.8</td>
<td>106</td>
</tr>
<tr>
<td>105</td>
<td>17.6</td>
<td>146</td>
</tr>
<tr>
<td>120</td>
<td>18.7</td>
<td>n/a</td>
</tr>
<tr>
<td>500</td>
<td>50.1</td>
<td>n/a</td>
</tr>
<tr>
<td>1000</td>
<td>70.1</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Chapter 6. Parameter analysis

Parameter settings are essential in cluster analysis because clustering results of a method are strongly dependent on the parameters that control the method. For instance, if the number of clusters is one of the parameters, it usually has a great impact on the output clustering. In practice, users generally prefer methods that have a small parameter space because the number of possible parameter values is limited and therefore it is less expensive to tune the parameters for a better result. Also, methods that have a clear interpretation of parameters are more preferable because it is easier to understand how to control the methods when doing cluster analysis. In order to show the effect of the parameters of GDVA on clustering results, this chapter will analyze the parameter space of GDVA using various data sets and demonstrate the relationship between parameter settings and the core assumption satisfaction of input graphs. Section 6.1 shows the accuracies of clustering results over the parameter space for data sets such as Iris, OHSUMED documents, and random graphs. Section 6.2 discusses some practical ways to select appropriate parameter values when applying GDVA to a data set.

6.1. The parameter space of GDVA

GDVA method has two main parameters $\delta \in [0, 1]$ and $\beta \in \{1, 2, 3, 4, 5\}$. Changing these parameters will affect the number of core nodes extracted. First, $\delta$ specifies the minimum rate of decrease for the density of core nodes in the sequence of density variation. Because the rate of decrease of density of a node implies the importance of the node in comparison with other nodes in the neighbor, nodes with higher rates belong to stronger core regions. Secondly, $\beta$ specifies the minimum number of required successive nodes that have the rate of decrease greater than $\delta$. So, $\beta$ enforces the minimum size of a descending segment in the density variation sequence which can be extracted as core. Note that a large descending segment indicates a big core. Therefore, increasing $\delta$ and $\beta$ will reduce the core set and make core nodes be extracted from denser and larger regions. On the other hand, decreasing $\delta$ and $\beta$ will expand the core set. In general, higher values of $\delta$ and $\beta$ produce a smaller number of clusters’ cores and consequently a smaller number of output
clusters. In practice, to cluster a new data set, we often start with high values to get coarse-grained clusterings and then lower the parameters to uncover more clusters in more fine-grained clusterings. The best values for $\delta$ and $\beta$ are also dependent on the sparsity of the input graph: Higher values should be used for sparse graphs while lower values are used for dense graphs. The reason for this adjustment is that the separation of clusters is much higher in sparse graphs than in dense graphs, and when clusters are well-separated as in a sparse graph we should use high values of $\delta$ and $\beta$ to detect strong clusters. On the other hand, clusters in dense graphs are usually not so well-separated, so we have to use lower values of $\delta$ and $\beta$ to detect weaker clusters. In general, we try to avoid setting these parameters too low because this tends to add many nodes into the core set including nodes that are unreliable.

Regarding parameter $\theta \in [0, \max_{i,j \in V} w_{ij}]$, if $\theta$ is set to zero, no edges between core nodes are removed, and thus each connected component of the core graph constitutes a cluster core. Note that if the input graph is complete or very dense, all core nodes may be connected in one cluster core. In this case, we need to set $\theta$ to some value higher than zero to be able to decompose the core graph. In general, we can compute a value for $\theta$ heuristically as discussed in Section 3.3. The higher value of $\theta$ is, the more cluster cores are produced. Because we always keep nodes in the same descending segment in the density variation sequence together, if $\theta$ is set to $\max_{i,j \in V} w_{ij}$, each descending segment will end up as one cluster core, and thus the number of output clusters will reach the maximum which is equal to number of descending segments. Regarding parameter $\lambda \in [0, 1]$, the change of $\lambda$ is for fine tuning of the clustering results. Basically, increasing $\lambda$ will reduce the number of uncertain nodes so weak clusters can expand more easily. On the other hand, decreasing $\lambda$ will increase the number of uncertain nodes so strong clusters tend to dominate and expand to a greater extent. Thus, this parameter usually affects the boundaries between clusters but not the number of clusters. We typically vary $\lambda$ and use DVI$_3$ to evaluate the quality of the clustering outputs and find the best value for $\lambda$.

Next, we investigate the change of the quality of clustering results over the parameter space for different data sets. Since the number of possible core nodes is limited, the parameter space is discrete and finite. Figure 6.1 shows the Iris data set [87] with three classes projected onto the first two principle components.
Even though the ground truth in this case consists of three clusters, we can see there are only two well-separated clusters. Clustering this data by GDVA as performed in Section 5.2.3 persistently produces these two clusters. Therefore, we get 0.76 as the result accuracy measured by comparing to the ground truth using NMI. Figure 6.2 shows the accuracy of clustering results over the parameter space with $\delta \in [0, 1]$ and $\beta \in \{1, 2, 3, 4\}$ where we can see that the clustering result is stable, i.e., for this data set the method is not sensitive to parameter settings. Note that the accuracy is zero if the values of $\delta$ and $\beta$ are set so high that all the data is grouped in one cluster (e.g., $\delta \in [0.9, 1]$ and $\beta = \{3, 4\}$). Similarly for the OHSUMED document grouping example described in Section 5.2.4, Figure 6.3 shows the accuracy of clustering results over the parameter space. Here, the best result is obtained when $\delta \in [0.2, 0.4]$ and $\beta = 2$.

![Figure 6.1](image1)

Figure 6.1. Projection of the Iris data set onto the first two principal components.

![Figure 6.2](image2)

Figure 6.2. Accuracy of results over the parameter space when clustering the Iris data set.
Figure 6.4 shows the average accuracy over parameter spaces with $\delta \in [0, 1]$ and $\beta \in \{1, 2, 3\}$ for graphs generated randomly using parameter $p \in \{0.1, 0.3, 0.5, 0.7\}$ as described in Section 5.1. Each random graph has four clusters of sizes 10, 20, 30, and 40 nodes. For each value of $p$, we generate 50 graphs and compute the average accuracy of the results. Note that the average accuracy is maximal when parameter values are low, i.e., $\delta \in [0.2, 0.3]$ and $\beta = 1$, because clusters in these graphs do not have a dense core. However, the range of $\delta$ and $\beta$ for the best clustering is more expanded when clusters are more separated (i.e., $p = 0.1$). In other words, choosing parameter settings is easier for data sets in which clusters are well-separated. Finally, Figure 6.5 shows the average accuracy over parameter spaces for random graphs generated similarly to the above case but each graph has four clusters of the same size of 25 nodes instead of different sizes. Similar to Figure 6.4, in this case clusterings with the best accuracy are obtained by low values of parameters.
Average accuracy of clustering results for random graphs generated using $p = 0.1$

Average accuracy of clustering results for random graphs generated using $p = 0.3$

Average accuracy of clustering results for random graphs generated using $p = 0.5$

Average accuracy of clustering results for random graphs generated using $p = 0.7$

Figure 6.4. Average accuracy over the parameter space when clustering randomly generated graphs with four clusters of sizes 10, 20, 30, and 40.
Average accuracy of clustering results for random graphs generated using $p = 0.1$

Average accuracy of clustering results for random graphs generated using $p = 0.3$

Average accuracy of clustering results for random graphs generated using $p = 0.5$

Average accuracy of clustering results for random graphs generated using $p = 0.7$

Figure 6.5. Average accuracy over the parameter space when clustering randomly generated graphs with four clusters of size 25.
There is a strong relation between the characteristics of an input graph and the best range of parameter values, which helps us select appropriate parameter settings for a data set. As seen before, when data has well-separated clusters, we obtain stable and good clustering results with high values of parameters. In the image segmentation experiments, we also use high parameter values, i.e., \( \delta \in [0.8, 1] \) and \( \beta \in \{3, 4, 5\} \), because the proximity graphs of images are very sparse and each segment has a dense core consisting of adjacent pixels with very similar color/intensity. For random graphs in which clusters do not have a dense core, \( \delta \) and \( \beta \) need to be set to low values, but when the graphs satisfy the core assumption, \( \delta \) and \( \beta \) should be set to higher values. Figure 6.6 shows the average accuracy over the parameter spaces when clustering graphs with clusters having and not having a dense core. In Figure 6.6 (a), graphs are generated randomly with four clusters of 25 nodes and with \( p = 0.5 \). Note that the average accuracy is maximal at low values of \( \delta \) and \( \beta \). In Figure 6.6 (b), graphs are generated similarly but then four nodes of each cluster are interconnected completely by edges with the same high weight so that each cluster has a small dense core region. In this case, the average accuracy of clustering results is maximal at high values of \( \delta \) and \( \beta \). Also,
the range of parameter values for good results is more expanded and the accuracy is more stable because cluster cores exist.

In sum, we should select parameter settings based on the sparsity of an input graph and on how strong clusters' cores are: Higher values for sparser graphs and clusters with stronger cores and vice versa. On the other hand, if the ground truth partitioning of a data set is known, we can verify whether or not the core assumption holds for the data by analyzing the quality of clustering results over the parameter space. If we are able to get a stable and good-quality clustering by using high values of parameters, then it implies that the data satisfies the core assumption and contains strong clusters' cores.
Chapter 7. Conclusions and future work

An important task in machine learning and data mining is cluster analysis, whose objective is to partition a data set into subsets such that objects of a subset are similar to each other while objects of different subsets are not similar. Cluster analysis has a wide range of applications such as image segmentation, document grouping, gene expression analysis, paralog/ortholog detection, social network analysis, etc. Graph theoretic approaches for cluster analysis work based on pairwise similarities of objects rather than positions of objects in a multi-dimensional space. Generally, graph-based clustering methods are widely applicable because pairwise similarities can be computed straightforwardly from the features of objects, even though this translation has to be carefully performed as it has a great impact on clustering results. This thesis focuses on graph-based approaches of cluster analysis, particularly a novel method called GDVA that performs clustering based on graph density variation analysis and density-based cluster validation. The density of a node is defined reasonably as the average similarity to other nodes in a graph. If clusters of a proximity graph contain a core consisting of high-density nodes strongly interconnected by edges that have a high and comparable weight, then the clusters can be discovered by identifying and expanding the clusters' cores. The clustering process is carried out in four steps: (1) construct the sequence of density variation, (2) extract core nodes, (3) identify cluster cores, and (4) expand cluster cores to full clusters. Under the core assumption, the sequence of density variation exposes two interesting characteristics: (1) Nodes in the sequence is ordered from sparse to dense regions and (2) descending sections of minimum densities in the sequence indicate core nodes. Therefore, a small set of core nodes can be extracted from the data to eventually determine cluster cores, each of which is a partition of the set of core nodes. Once identified, cluster cores can be expanded by scanning backwards the sequence of nodes from dense to sparse regions and assigning every node to the most similar cluster. This method of clustering can help to explore the cluster structure, i.e., besides the dense cores of clusters, nodes of a cluster are ordered according to their density, and nodes close to the boundaries between clusters are identifiable based on the difference of the similarities to the most and the second most similar clusters of the nodes. Overall, the
GDVA method takes \( O(|E| + |V|) \) storage space and \( O(|E| + |V| \log |V|) \) running time, which is roughly equal to \( O(|V| \log |V|) \) for sparse graphs and to \( O(|E|) \) for dense graphs.

Another aspect of cluster analysis is to validate a clustering result so as to compare different results and methods. Generally, the quality of a clustering result is measured based on the intra-cluster and inter-cluster connectivity of the nodes. A good result should have high total intra-cluster connectivity and low total inter-connectivity. Therefore, we defined several cluster validity indices called DVI which use the densities of the nodes of the graph with respect to their cluster and to the other clusters. These intra-cluster and inter-cluster densities are combined in DVI indices to quantitatively evaluate the quality of a clustering result and therefore results obtained by different values of parameters can be compared. An important advantage of DVIs is that the computational time is very fast, thus these validity indices can be integrated into a clustering algorithm to select appropriate settings for parameters. In GDVA, we use DVI in the last step of cluster core expansion to reduce the number of parameters needed to be specified by users.

GDVA is a general purpose clustering method which can be applied to data from various fields. We have showed experimental results of GDVA with heterogeneous data such as random graphs, multi-dimensional data, images, text and gene expression data. For these data sets, different similarity measurements such as Euclidean distance, tf-idf, correlation coefficients, and user-defined functions were used to build proximity graphs which then were partitioned into clusters. In case the labels of data objects are known, we can evaluate the quality of clustering results by measuring the similarity between the results and the ground truth clustering of data labels using NMI and ARI indices. Generally, GDVA performs very well in comparison to other methods. In case of experiments with images, the results of segmentation can be assessed visually, which show that GDVA produces more satisfactory results than the normalized cut method. The most noticeable advantage of GDVA is its low time complexity. It runs much faster than the other clustering methods, especially on large graphs.

Evaluating clustering methods is a very challenging task because each method makes its own assumptions on the structure of clusters. Due to the heavily data-dependent nature of the clustering task, comparisons based on a number of specific data sets cannot be generalized to assert a method is universally better than another. Furthermore, ground truths, which are the references for evaluating the quality of clustering
results, are usually created by humans. Since different people can have different judgments on the same data set, the ground truths are subjective and not unique, which complicates the evaluation of clustering results. For instance, clustering the same data set at different scales (fine or coarse-grained) generally gives different solutions, all of which are good. So comparing the quality of the outputs of different methods is not trivial. To alleviate this issue of human subjectivity in evaluations, we used graphs generated randomly based on a known ground truth. Besides, using random graphs has two additional advantages. First, there is no pre-processing to transform vector-based to graph-based data, so evaluation results are applied directly to clustering methods and not affected by any data transformations. Second, we can generate a large number of random graphs and also control the degree of separation between clusters, thus a wider variety of graph structures can be evaluated and compared. Consequently, the results are presented as curves of average accuracies rather than only few separated points if only several data sets were used. In Chapter 5, we used random graphs generated based on a simple model with a controlling parameter specifying the degree of cluster separation. It is worth noting that using random graphs in evaluation can promote a particular algorithm if the graphs are created deliberately according to the data assumption of that algorithm. To cover a wide range of graph structures, we used a straightforward generation model with as few parameters as possible. In fact, the core assumption used in GDVA is not satisfied on these random graphs because edge weights in each cluster are highly varied. Nevertheless, evaluations based on these graphs showed good clustering results of GDVA compared to NCut, MCL, and AP methods. One may ask how the evaluation result will change if clusters have a core region. Thus, we compared the results of GDVA on random graphs without cluster cores and random graphs with a small core injected in each cluster. As expected, the clustering results obtained in the latter case were much more stable and reliable than that in the former case over possible parameter settings. So good clustering results are obtained easily when GDVA is applied to data in which clusters have dense cores.

Parameter selection is a common issue for clustering methods since it is usually not easy for users to decide which values of parameters are suitable for a data set whose structure is unknown. For example, some methods may require to specify the number of clusters, or a threshold, etc. which can be hard to know. Clearly, it is necessary for users to understand the effect of parameters on the outputs of clustering methods. The main parameters of GDVA consist of $\delta \in [0, 1]$ and $\beta \in \mathbb{N}$, which can control the number of
extracted core nodes. We investigated the parameter space formed by $\delta$ and $\beta$ for different data sets and demonstrated that these parameters can be adjusted to change the granularity of clustering results. Specifically, higher values of $\delta$ and $\beta$ produce more coarse-grained results and vice-versa. We found that the clustering results of GDVA are much better and, more importantly, less dependent on parameters when the underlying clusters in data contain a dense core or are well-separated. Moreover, the study of parameters $\delta$ and $\beta$ showed that generally we should consider higher values for $\delta$ and $\beta$ when input graphs are sparse and/or clusters have strong cores and vice versa. On the other hand, parameter analysis also shows that an analysis of the parameter space can be performed to examine whether or not cluster cores exist in data. Specifically, in case the ground truth is known and we obtain a good-quality clustering with high values of $\delta$ and $\beta$, then it implies the core assumption holds. When the ground truth is unknown but a stable clustering is produced by high values of $\delta$ and $\beta$, it also suggests clusters have dense cores.

An important advantage of GDVA is that users do not have to specify the number of clusters. In reality, cluster structure is often complex (e.g., segments in images), so the number of clusters should not be set beforehand; rather it should be just the outcome of a clustering result, i.e., the number of clusters is not unique and can take on very different values because one data set can have many acceptable ways of partitioning. For example, we have the ground truth with six clusters and three partitioning solutions: one with six clusters, one with ten clusters, and one with three clusters. The partitioning with six clusters is not necessarily a good solution because it can be very different from the partitioning of the ground truth. In contrast, the solution with ten clusters can be better if it is a more fine-grained partitioning such that we can combine some clusters to get the ground truth. The partitioning with three clusters can also be a good solution if it is a coarse-grained partitioning of the ground truth. In GDVA, $\delta$ and $\beta$ can be adjusted to uncover a coarse-grained or a fine-grained clustering.

Another advantage of GDVA is that it does not only partition input graphs but also provides additional information about cluster structure. Under the core assumption, GDVA puts nodes into an order from sparse to dense regions, so nodes within each cluster can be sorted according to this order. Core nodes extracted from the densest regions of clusters can be used as representative data objects for the clusters. The number of these representatives is very small compared to the size of input data. In addition, they are
mostly free from noise data, so in many occasions it is very convenient to deal with the core nodes instead of the entire data. Non-core nodes are associated with a confidence degree which indicates how close the nodes are to other clusters in comparison to their cluster. Nodes near the boundaries between clusters have a low confidence degree so that they are uncertain nodes and not taken into account when assigning nodes to the most similar cluster.

Although this thesis focuses only on clustering weighted graphs, the GDVA approach can be modified to perform clustering for unweighted graphs and possibly directed graphs. We briefly discuss here GDVA modifications needed in these settings. In an unweighted graph, since all edges have the same weight, the core assumption can be interpreted as a region where the nodes are completely or almost completely connected. Non-core nodes should be more strongly connected to the core nodes of their clusters than to nodes of other clusters. The density of a node defined in (3.1) is equal to the ratio of the number of adjacent nodes to the number of nodes in the graph. Thus, a node attains the maximum density, which is one, if it is connected to all other nodes. Based on this definition, Steps 1 and 2 of GDVA to construct the sequence of density variation and extract core nodes need not be modified. However, Step 3 to identify cluster cores cannot use a threshold to remove weak edges because all edges are of the same weight. If cluster cores are disconnected, we can simply assign each connected component in the core graph to a cluster core. Otherwise, we can use one of the clustering methods for unweighted graphs to partition the core graph. (See [26] for a recent review of the methods that are able to find clusters in unweighted graphs.) Similar to the case of weighted graphs, the separation of cluster cores in the core graph is much better than that of clusters in the input graph. An additional alternative to find cluster cores when they are not well-separated is to apply Steps 1 and 2 once again to the core graph to further reduce the number of core nodes and therefore make cluster cores more separable before finding connected components. Finally, in Step 4, we may not measure the similarity between a node and a cluster by the average or the maximum weight of edges linking the node and the cluster, whose values are always the same. Here, the most apparent substitution method for measuring similarity is to use the number of edges linking the node with the cluster (i.e., the number of adjacent nodes in the cluster), or alternatively the ratio of the number of edges to the number of nodes of the cluster. Finally, the cluster validity indices DVI are based on the density defined in (3.1) and therefore are still applicable. Now, let us turn to the case of clustering a directed graph. In order to
apply GDVA, we need to find an efficient and reasonable way to define the density of a node with respect to a graph while taking into account directed edges. The easiest way may be to convert directed graphs to undirected graphs by, for instance, replacing all directed edges between a pair of nodes by an undirected edge with a weight equal to the average weight of the directed edges. However, whether or not this simple approach makes sense in terms of estimating the density is still an open question. On the other hand, it may also be possible to estimate the density based on the number of paths between a node and a graph, or the number of cycles involving a node and the nodes of a graph, etc. These latter approaches may be more reasonable for directed graphs; however, their computational complexity is relatively expensive. Besides, different clustering tasks and data sets may need different ways of estimating density to suit the description of clusters to be found in the data. In sum, many possibilities for extending GDVA and DVI can be explored for clustering unweighted and/or directed graphs. Once these extensions are implemented, the question of how they perform on synthetic and real data will need to be addressed.

This final chapter has summarized the designs, advantages, and experimental results of the GDVA method and DVI indices for clustering weighted graphs. In the future, the clustering method and validity indices can continue to be improved for a better performance and utility. For example, an open question is how to apply the clustering algorithm if the input graph does not fit into memory. Is it possible to divide a large graph into smaller parts and apply the method to each part and combine the results? The performance will be improved tremendously if we can parallelize all or part of the algorithm so that it is able to use multiple computers for clustering an input graph. In addition, GDVA may benefit by making use of minimum cuts that optimize a graph cut criterion when partitioning the core set or detecting cluster boundaries based on cluster cores. To facilitate the GDVA method's usage, the method may be made fully automatic by adding a technique for analyzing the parameter space of $\delta$ and $\beta$ and determining automatically appropriate parameter values based on ranges of settings where the outputs are stable. As discussed earlier, if there is a large region in the parameter space where the output clusters are stable, we can assert that the data contain strong clusters. For more empirical evaluation results, we can perform experiments with new data sets in bioinformatics such as gene expressions and protein sequences for gene expression clustering and paralog/ortholog detection. Also, it will be interesting to apply GDVA to image segmentation where proximity graphs are computed by incorporating additionally texture or other object information, so that
output segments reflect more naturally the objects in images, especially when their surface involves textures. Regarding the density-based cluster validity indices DVI, evaluations can be performed on real data in addition to random graphs to study the performance of these indices and compare them with other indices in the literature on different data types and cluster structures. Finally, the GDVA method and DVI indices can be extended for clustering unweighted and/or directed graphs as described previously. Then it is interesting to study the effect of data transformation, i.e., how final results are changed if we transform data represented by weighted graphs into unweighted graph representation before doing analyses. It will be necessary to evaluate these extensions with reference to existing methods (there are many more partitioning algorithms for unweighted graphs than those for weighted graphs). Many publically available data sets in social networks and networks of communication, collaboration, interaction, citation, etc. can be used for this purpose.
References


