

HONGLEI ZHANG

# Graph Analysis and Applications in Clustering and Content-based Image Retrieval



HONGLEI ZHANG

Graph Analysis and  
Applications in Clustering and  
Content-based Image Retrieval

ACADEMIC DISSERTATION

To be presented, with the permission of  
the Faculty of Information Technology and Communication Sciences  
of Tampere University,  
for public discussion in the Auditorium S3  
of the Sähköotalo building, Korkeakoulunkatu 3, Tampere,  
on 9 August 2019, at 12 o'clock.

ACADEMIC DISSERTATION

Tampere University, Faculty of Information Technology and Communication Sciences  
Finland

<i>Responsible supervisor or/and Custos</i>	Prof. Dr. Moncef Gabbouj Tampere University Finland	
<i>Supervisor</i>	Prof. Dr. Serkan Kiranyaz Qatar University Qatar	
<i>Pre-examiners</i>	Prof. Dr. Hichem Frigui University of Louisville USA	Prof. Dr. Amel Benazza-Benyahia University of Carthage Tunisia
<i>Opponents</i>	Prof. Dr. Mikko Kivelä Aalto University Finland	

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

Copyright ©2019 author

Cover design: Roihu Inc.

ISBN 978-952-03-1183-4 (print)  
ISBN 978-952-03-1184-1 (pdf)  
ISSN 2489-9860 (print)  
ISSN 2490-0028 (pdf)  
<http://urn.fi/URN:ISBN:978-952-03-1184-1>

PunaMusta Oy – Yliopistopaino  
Tampere 2019

## ABSTRACT

About 300 years ago, when studying Seven Bridges of Königsberg problem - a famous problem concerning paths on graphs - the great mathematician Leonhard Euler said, “This question is very banal, but seems to me worthy of attention”. Since then, graph theory and graph analysis have not only become one of the most important branches of mathematics, but have also found an enormous range of important applications in many other areas. A graph is a mathematical model that abstracts entities and the relationships between them as nodes and edges. Many types of interactions between the entities can be modeled by graphs, for example, social interactions between people, the communications between the entities in computer networks and relations between biological species. Although not appearing to be a graph, many other types of data can be converted into graphs by certain operations, for example, the  $k$ -nearest neighborhood graph built from pixels in an image.

Cluster structure is a common phenomenon in many real-world graphs, for example, social networks. Finding the clusters in a large graph is important to understand the underlying relationships between the nodes. Graph clustering is a technique that partitions nodes into clusters such that connections among nodes in a cluster are dense and connections between nodes in different clusters are sparse. Various approaches have been proposed to solve graph clustering problems. A common approach is to optimize a predefined clustering metric using different optimization methods. However, most of these optimization problems are NP-hard due to the discrete set-up of the hard-clustering. These optimization problems can be relaxed, and a sub-optimal solution can be found. A different approach is to apply data clustering algorithms in solving graph clustering problems. With this approach,

one must first find appropriate features for each node that represent the local structure of the graph. Limited Random Walk algorithm uses the random walk procedure to explore the graph and extracts efficient features for the nodes. It incorporates the embarrassing parallel paradigm, thus, it can process large graph data efficiently using modern high-performance computing facilities. This thesis gives the details of this algorithm and analyzes the stability issues of the algorithm.

Based on the study of the cluster structures in a graph, we define the authenticity score of an edge as the difference between the actual and the expected number of edges that connect the two groups of the neighboring nodes of the two end nodes. Authenticity score can be used in many important applications, such as graph clustering, outlier detection, and graph data preprocessing. In particular, a data clustering algorithm that uses the authenticity scores on mutual  $k$ -nearest neighborhood graph achieves more reliable and superior performance comparing to other popular algorithms. This thesis also theoretically proves that this algorithm can asymptotically find the complete recovery of the ground truth of the graphs that were generated by a stochastic  $r$ -block model.

Content-based image retrieval (CBIR) is an important application in computer vision, media information retrieval, and data mining. Given a query image, a CBIR system ranks the images in a large image database by their “similarities” to the query image. However, because of the ambiguities of the definition of the “similarity”, it is very difficult for a CBIR system to select the optimal feature set and ranking algorithm to satisfy the purpose of the query. Graph technologies have been used to improve the performance of CBIR systems in various ways. In this thesis, a novel method is proposed to construct a visual-semantic graph—a graph where nodes represent semantic concepts and edges represent visual associations between concepts. The

constructed visual-semantic graph not only helps the user to locate the target images quickly but also helps answer the questions related to the query image. Experiments show that the efforts of locating the target image are reduced by 25% with the help of visual-semantic graphs.

Graph analysis will continue to play an important role in future data analysis. In particular, the visual-semantic graph that captures important and interesting visual associations between the concepts is worthy of further attention.





## PREFACE

Firstly, I would like to express my deep gratitude to my supervisor Prof. Dr. Moncef Gabbouj for his continuous support, his patience, encouragement, and guidance along the years of my study. Prof. Gabbouj is not only a great supervisor but also a great person from whom I have learned so much. I have enjoyed every single moment working with him. I would also like to thank Prof. Dr. Serkan Kiranyaz for his valuable guidance, encouragement and inspiration during my Ph.D. study.

My sincere thanks also give to the pre-examiners Prof. Dr. Amel Benazza-Benyahia and Prof. Dr. Hichem Frigui for their careful review of the draft of this thesis, the valuable comments, and insightful suggestions.

I would especially like to thank my dear colleagues Dr. Stefan Uhlmann, Dr. Jenni Raitoharju, Dr. Guanqun Cao, Dr. Kaveh Samiee, Dr. Iftikhar Mohamed, Mr. Waris Adeel Mohamad, Dr. Ezgi Ozan, Mr. Morteza Zabihi, Mr. Anton Murvev, Dr. Alexandros Iosifidis, Mr. Dat Tranthanh, Ms. Lei Xu, and Mr. Mohammad Fathi Al-Sa'd for great working atmosphere, generous help and excellent teamwork. I would always remember the moment we spend together and the cherished friendship you have given me. I would also like to thank other colleagues who are working or had been working in the MUVIS team for their great support and kind help.

I would also like to thank my beloved wife Dr. Jinghuan Wang and my son Yuhao Zhang for their support and incent towards my goal. I'd like to give special thanks to my father Shudong Zhang, my mother Xiulan Dong, my brother Hongsheng Zhang and my sister Hongbo

Zhang. They have helped me through my life and answered my requests whenever I needed them.

Last but not least I would like to thank D2I and CVDI projects for giving financial support, CSC and TCSC for providing computing service to complete my research work.

Tampere, July 2019

Honglei Zhang

# CONTENTS

Abstract . . . . .	i
Preface . . . . .	v
List of Symbols . . . . .	ix
List of Abbreviations . . . . .	xi
List of Publications . . . . .	xiii
Author's Contribution . . . . .	xv
1. Introduction . . . . .	1
1.1 Brief history of graph theory . . . . .	1
1.2 Introduction of graph analysis in machine learning . . . . .	3
1.3 Objectives and thesis overview . . . . .	7
2. Graph Theory . . . . .	11
2.1 Basic concepts and graph representations . . . . .	11
2.2 Graph attributes and definitions . . . . .	12
2.3 Metrics . . . . .	14
2.4 Random graph generation models . . . . .	17
3. Graph Clustering and Graph-based Data Clustering . . . . .	21
3.1 Graph clustering . . . . .	21
3.2 Evaluation of graph clustering algorithms . . . . .	22
3.3 Graph clustering methods . . . . .	29

3.4	Random walk-based graph clustering . . . . .	35
3.5	Graph-based data clustering . . . . .	45
3.6	Summary . . . . .	59
4.	Content-based Image Retrieval with Graph Techniques . . . . .	63
4.1	Introduction to content-based image retrieval . . . . .	63
4.2	Visual-semantic graph . . . . .	70
4.3	Using visual-semantic graph in CBIR systems . . . . .	81
4.4	Summary . . . . .	89
5.	Conclusions . . . . .	93
	Bibliography . . . . .	96
	Publications . . . . .	123

## LIST OF SYMBOLS

$ S $	cardinality of set $S$
$ x $	absolute value of variable $x$
$\ \cdot\ _1$	$L_1$ norm
$\cap$	intersection of two sets
$\cup$	union of two sets
$\binom{n}{k}$	the number of $k$ -combinations of a set of $n$ elements
$A$	adjacency matrix
$A \setminus B$	relative complement of set $A$ in $B$
$A(e_{ij})$	authenticity score of edge $e_{ij}$
$\overline{ab}$	edge that connects nodes $a$ and $b$
$a, b, c, \dots$	nodes in a graph
$D$	degree matrix
$D^{out}$	out-degree matrix
$d_i$	the degree of node $n_i$
$E$	the set of edges in a graph
$E^+$	the set of edges that two ends nodes are in the same cluster
$E^-$	the complementary set of $E^+$
$E(\cdot)$	the expected value of a random variable
$e_{ij}$	edge that connects nodes $i$ and $j$
$G(V, E)$	graph $G$ with the set of node $V$ and the set of edge $E$
$H(X)$	entropy of random variable $X$
$H(X Y)$	conditional entropy of random variable $X$ given random variable $Y$
$H(X, Y)$	joint entropy of random variables $X$ and $Y$
$I(X, Y)$	mutual information of two random variables $X$ and $Y$
$K^c$	complement of set $K$
$k_a$	the degree of node $a$
$L$	Laplacian matrix
$L_{sym}$	normalized Laplacian matrix

$n_i$	node $i$
$N_a$	the neighboring nodes of node $a$
$O(\cdot)$	Big O notation indicates that a function is as the order of another function
$P$	transition matrix of a Markov process
$V$	the set of nodes in a graph
$x^{(t)}$	a vector of variables at time $t$
$x^{(*)}$	a fixed-point of variable $x$
$\pi$	equilibrium state of a Markov process
$\emptyset$	empty set

## LIST OF ABBREVIATIONS

APL	Average Path Length
ARI	Adjusted Rand Index
BRkNN	Binary Relevance $k$ -NN
CBIR	Content-based Image Retrieval
CNN	Convolutional Neural Network
DBSCAN	Density-based Spatial Clustering of Applications with Noise
EBGM	Elastic Bunch Graph Matching
gCBIR	Graph-enhanced CBIR
GDL	Graph Degree Linkage
GN	Girvan Newman Algorithm
HPC	High-performance Computing
ID	Identification
IRD	Inverse Relative Density
KNN	$k$ -Nearest Neighbor
LDA	Linear Discriminant Analysis
LLE	Local Linear Embedding
LRW	Limited Random Walk
MCL	Markov Cluster Algorithm
MCVS	Microsoft Clickture-lite Visual Semantic
MIR	Mean Inverse Rank
MIT	Massachusetts Institute of Technology
MKNN	Mutual $k$ -Nearest Neighbor
MLkNN	Multilabel $k$ -NN
N-Cut	Normalized Cut
NMI	Normalized Mutual Information
PA	Preference Attachment
PCA	Principle Component Analysis
RI	Rand Index

RRW      Repeated Random Walk



## LIST OF PUBLICATIONS

- [P1] Honglei Zhang, Jenni Raitoharju, Serkan Kiranyaz, and Moncef Gabbouj. Limited random walk algorithm for big graph data clustering. *Journal of Big Data*, 3(1):26, 2016.
- [P2] Honglei Zhang, Serkan Kiranyaz, and Moncef Gabbouj. Outlier edge detection using random graph generation models and applications. *Journal of Big Data*, 4(1):11, April 2017.
- [P3] Honglei Zhang, Serkan Kiranyaz, and Moncef Gabbouj. A k-nearest neighbor multilabel ranking algorithm with application to content-based image retrieval. In *2017 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2017 - Proceedings*, pages 2587–2591. IEEE, 2017.
- [P4] Honglei Zhang, Serkan Kiranyaz, and Moncef Gabbouj. Data Clustering Based on Community Structure in Mutual k-Nearest Neighbor Graph. *International Conference on Telecommunications and Signal Processing (TSP)*, 2018.
- [P5] Honglei Zhang and Moncef Gabbouj. Feature Dimensionality Reduction with Graph Embedding and Generalized Hamming Distance. *IEEE International Conference on Image Processing (ICIP)*, 2018.



## AUTHOR'S CONTRIBUTION

Publication [P1] presents the Limited Random Walk (LRW) graph clustering algorithm that achieves the state-of-the-art accuracy and can be implemented in an embarrassing parallel paradigm. Honglei Zhang conceived the idea, performed the implementation and computation, and wrote the first draft of the manuscript.

Publication [P2] studies the authenticity of the edges in a graph and gives the definition of the authenticity score. It also shows various applications that benefit from the analysis of the authenticity scores. Honglei Zhang conceived the idea, developed the theorem and perform the experiments. Honglei Zhang wrote the manuscript in consultation with the co-authors.

Publication [P3] presents a multilabel ranking algorithm and its application in content-based image retrieval. The proposed algorithm is evaluated using a big image dataset and shows significant improvement compared to the other instance-based multilabel ranking algorithms. Honglei Zhang conceived the idea, perform the experiments and drafted the manuscript.

Publication [P4] extends the idea of data clustering using authenticity scores and presents a method to determine the number of clusters. Honglei Zhang conceived the idea, performed the experiments and drafted the manuscript.

Publication [P5] presents a dimensionality reduction method using graph embedding and generalized Hamming distance. The method uses information embedded in multilabel data and gives better performance compared to other competing methods. Honglei Zhang conceived the idea, perform the experiments and drafted the manuscript.



# 1. INTRODUCTION

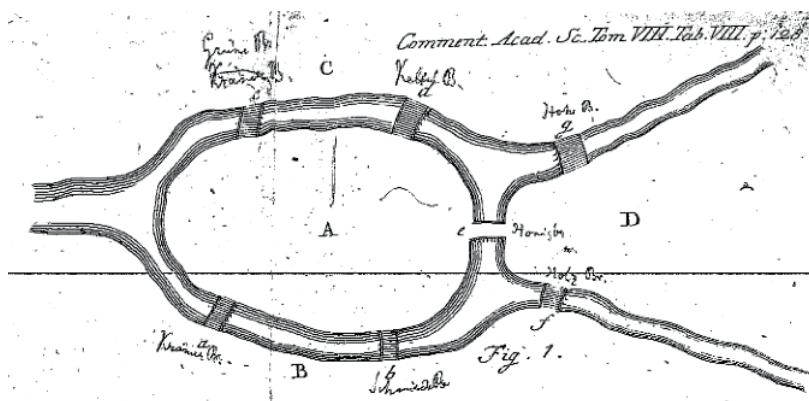
## 1.1 Brief history of graph theory

The study of graph theory dated back to Leonhard Euler in the sixteenth century when he studied the problem of the Seven Bridges of Königsberg—finding a route by which one can visit every part of the city and cross each of the seven bridges once and only once [1, 2]. Euler said [3]:

“This question is so banal, but seemed to me worthy of attention in that [neither] geometry, nor algebra, nor even the art of counting was sufficient to solve it.”

In the past hundreds of years, with the help of the great efforts by mathematicians and scientists in many fields, graph theory has not only become an important branch of mathematics but also a fundamental tool in areas such as physics, biology, social science, and information technology [4, 5, 6, 7]. Besides the “Seven Bridges” problem, many other famous problems and conjectures, including the four color theorem [8] and traveling salesman problem [9], have greatly aroused the interests of scientists in graph theories and made graph theory one of the most active topics in mathematics [4, 5, 7]. One of the recent and important advances in graph theory is the random graph model, which is a combination of graph theory and probability theory [10].

A graph is a mathematical model that abstracts entities as vertices



**Figure 1.1** A sketch of the Seven Bridges of Königsberg by Euler (E53 of MAA Euler Archive [2])

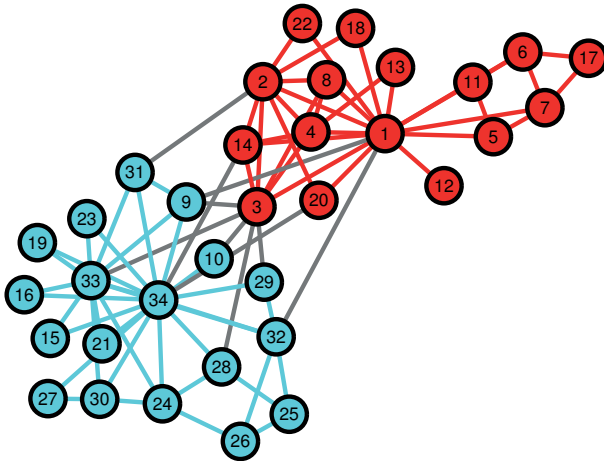
(nodes) and their relations as edges (links). Graphs can be categorized in many different ways, such as directed and undirected graph, weighted and unweighted graph, finite and infinite graph [7]. Tree and forest can also be considered as special types of graphs.

Because of the huge volumes of data in various applications that can be modeled in a graph structure, graph analysis has become more and more important in fields other than mathematics. For example, in social science, the relationships between people form social networks where each vertex represents a person and each edge indicates the relationship between the two connected people [11, 12, 13]. Similarly, a transport system can be modeled as a graph where the vertices are the cities and the edges are the roads [14]. Even though the data being analyzed are not explicitly organized as graphs, very often they can be transformed, and the graph techniques can be used to analyze the data. For example, a metabolic system is a very complex system that is comprised of organs, hormones, and enzymes. This system can be represented by a directed bipartite graph where the vertices are reactions and the chemicals produced and/or consumed by reactions, and the directed edges indicate whether a metabolite is a substrate

(input) or a product (output) of a reaction [7]. The Reactome pathway knowledge base is a collection of relations between human proteins and reactions. The relations have been modeled as graphs for the public to discover information in gene expression pattern or somatic mutation from tumor cells [15].

## 1.2 Introduction of graph analysis in machine learning

The combination of graph analysis and machine learning is essential and beneficial since they both study the relations between given entities. For many applications, especially when the data is in a graph structure, it is sometimes difficult to separate the two methodologies. For example, graph clustering in graph analysis and data clustering in machine learning have a similar target. Techniques from one field were thus used to solve problems in the other field. In many real-world graphs, especially in social networks, the vertices form communities—the links among the vertices inside a community are much denser than the links connecting vertices in different communities [16]. Finding these communities is important to understand the underlying relations among the vertices [P1, 17, 18, 19, 20, 21]. Graph clustering is a technique to find communities in big networks. Fig. 1.2 shows the graph of Zachary’s karate club, which was used in the earliest researches in this subject [22]. Each vertex represents a member of the club and the links indicate the members have interactions outside of the club. The club split into two groups due to the conflicts between the two administrators. The color of the vertices shows how the club was split at that time. Many graph clustering algorithms have been used to interpret and predict this split [P1, 18, 23]. Graph partition, which is closely related to graph clustering, aims to partition a big graph into smaller components of roughly equal sizes such that the links between any two partitions are minimized. Graph partition is crucial for processing and analyzing big graph data in a distributed



*Figure 1.2* Graph of Zachary's karate club [22]

system [24, 25, 26].

Data clustering is an important task used in many fields, such as machine learning, pattern recognition, information retrieval [27]. Various methods that use graph clustering techniques have been proposed to solve the problem [28, 29, 30, 31]. Given the data samples, these algorithms first construct a  $k$ -nearest neighbor graph (KNN) or mutual  $k$ -nearest neighbor graph (MKNN). Then graph clustering techniques can be applied to find the clusters in the constructed graph. In [P2], Zhang et al. proposed an approach to split the graph into components by iteratively removing the edges according to their authenticity scores. The number of clusters can be determined by analyzing the sizes or the properties of the components that are formed during the collapsing process.

Graph analysis has also found further use in an important machine learning problem, namely the embedding method for dimensionality reduction. Yan et al. unified different dimensionality reduction methods, including Principle Component Analysis (PCA), Linear Discrim-



inant Analysis (LDA), Local Linear Embedding (LLE) and IsoMap, within a common framework called graph embedding [32]. An intrinsic graph and a penalty graph are constructed from the data samples. The objective of the algorithms is to find lower dimension representations of the data samples that preserve the relationships of the vertices in the intrinsic graph and the penalty graph. This framework has inspired many researchers to develop new dimensionality reduction algorithms [33, 34, 35]. For example, Zhang et al. constructed an intrinsic graph using generalized Hamming distance as the weights to model the similarity of the labels in multi-label data analysis [P5].

Social networks are important graph structure because of their broad and important applications [36, 37, 38]. One of the most important applications is to find the strategy that maximizes the influence of an idea through social networks [39, 40]. Since a decision made by an individual is frequently affected by people connected to him/her, the effectiveness of propagating an idea through the social network is greatly affected by the strategy of selecting the target individuals. Increasing the acceptance of those influential people may lead to faster and broader acceptance through the whole society. A related topic is to slow down or stop the propagation of a virus through a network, for example, an infectious disease among social networks or computer virus through computer networks [41, 42]. Node centrality [7] and other measurements [39, 40] have been used to evaluate the influential power of each individual. PageRank is a measurement that evaluates the importance of a web page according to its relations to other web pages in the web graph—a graph whose nodes are web pages and edges are the hyperlinks between the pages [43]. Ranking entities according to certain criteria is what learning to rank, another important application in machine learning [44], deals with. PageRank and other graph-based features have proved to be effective for this application [45].

Graph techniques have also been extensively used in computer vision tasks [46]. Wiskott et al. developed an elastic bunch graph matching (EBGM) algorithm to recognize human faces [47]. First, feature vectors of fiducial points (eyes, mouth, etc) are extracted using Gabor filters. Then the bunch graph is constructed from these fiducial points and a face is recognized using a similarity function that combines the similarity of the fiducial points and the distortion of the image grid in the graph. Dealing with the image segmentation problem, Boykov et al. used graph cuts to minimize the energy function defined for a segmentation, where the graph is constructed from the pixel lattice of an image with the addition of two terminal nodes [48]. To estimate human pose in an image that contains multiple people, Cao et. al. applied graph matching technique for part association [49]. Zhang and Shah modeled the human parts by a relational graph and a hypothesis graph and used a tree-based optimization method to estimate the human pose from a sequence of the frames in a video [50].

Content-based image retrieval (CBIR) system helps users to efficiently retrieve information from a large image dataset based on the content of query images [51, 52]. It is another important application in machine learning and computer vision [53]. Graph techniques have also shown great importance in CBIR systems. Cai et. al. constructed image graph based on the hyperlinks between the web pages on the Internet and proposed to represent an image by its visual feature, textual feature and graph-based feature [54]. Using these three representations, images retrieved from a search engine can be clustered into semantic clusters and presented to the users for better clarity, simplicity, and consistency. Graph-ranking model, for example, the aforementioned PageRank, decides the importance of a vertex according to the graph structure [55]. Xu developed a graph-based ranking model called Efficient Manifold Ranking that can efficiently construct the image graph and compute the ranking scores for a CBIR system [56].

Deep neural networks have undoubtedly gained the most attention in the area of machine learning for the last couple of years [57, 58]. An artificial neural network can be modeled by a directed graph where each node represent an artificial neuron and each edge indicates the connection from the output of a neuron to the input of another neuron [59]. Graph-based methods are used to study linearly separable Boolean functions, which an important problem in the research of neural networks [60, 61]. Another important combination of graph theory and neural networks is to apply neural networks, in particular, convolutional neural networks, on the data that are represented in graph structures. Kipf and Welling proposed a graph-based semi-supervised classification method to classify the node in a graph where only a small subset of nodes are annotated [62]. To execute convolution operations on a graph, Niepert et. al. construct a node sequence via a graph labeling procedure and a graph normalization that imposes an order of the neighborhood graph [63]. The proposed method can be used in the graph classification problem where each graph structure is assigned to a label. Zhang et. al. applied an evolutionary method to find efficient graph structures of deep convolutional networks for image classification problems. The top-performing graph structures found during the evolution show some properties of the graph structure that greatly affect the performance of a deep convolutional neural network (CNN) [64].

The combination of graph-based techniques and machine learning is far beyond what has been discussed above. Graph theory has gained great attention from the researchers in machine learning and becomes a fundamental mathematical tool in this field.

### 1.3 Objectives and thesis overview

During the last decades, with the rapid development of the Internet and computer technologies, the size of data to be processed has increased dramatically. For example, a visual-semantic graph build for a CBIR application may contain millions of nodes and tens of millions of edges. New algorithms are required to tackle the difficulties caused by the large graph data to efficiently use graph techniques in different applications.

The objectives of this thesis are to develop a novel mathematical formulation for graph clustering, graph analytics, and graph-based data clustering for large graph data and use these formulations to improve graph-based CBIR system's efficiency compared to traditional CBIR approaches. The main research questions the thesis aims to answer are: what new insights graph-based approaches can provide us when dealing with large-scale datasets when the latter are represented by graphs? how can graph analytics solve image content-based indexing and retrieval in large-scale databases?

The 5 publications included in this thesis answer these research questions from different directions. Publication [P1] answers the research questions by presenting an efficient graph clustering algorithm, named the Limited Random Walk algorithm, for large graph data that achieves the state-of-the-art accuracy and can be implemented in an embarrassing parallel manner. Publication [P2] studies the authenticity of edges in a graph and shows various applications that benefit from this analysis, in particular when dealing with large-scale graphs. Publication [P4] extends the idea of [P2] and presents a method to cluster data using graph techniques. Publication [P3] and [P5] answer the research question by presenting a multilabel ranking algorithm and a dimensionality reduction method that are based on graph techniques and serve as critical enablers for content-based image retrieval in large-

scale databases.

The rest of the thesis is organized as follows:

Chapter 2 gives a brief introduction of some basic concepts and definitions used in graph theory. Some attributes and properties for nodes and edges are also described in this chapter. Metrics that evaluate density, centrality, and authenticity are discussed. The last section of this chapter describes some important random graph generation models that are used in the remaining chapters.

Chapter 3 describes some important algorithms for graph clustering and graph-based data clustering problems. It first explains different types of graph clustering problems and some metrics to evaluate, either externally or internally, the performance of graph clustering algorithms. Then an overview of spectral graph clustering, fitness function optimization, data-based, and model-based clustering techniques are given. This chapter gives a detailed explanation of random walk-based clustering algorithms. The stability and complexity of the limited random walk algorithm are discussed. Next, the data clustering algorithms that use graph analysis techniques are described. This chapter also gives a detailed discussion about the data clustering algorithm that is based on the authenticity scores of the edges in a mutual  $k$ -nearest neighbor graph. The sufficient condition that guarantees the complete recovery of the ground truth is proved.

Chapter 4 discusses the benefits of using graph techniques in the area of CBIR. It first describes some challenges that a general CBIR system faces, for example, the difficulty of annotating a large dataset, and the ambiguity of the intention of a query. Then the system architecture of a CBIR system is briefly described. To capture the visual relations of semantic concepts, this chapter shows a method to construct a visual-semantic graph from a large database of clicktiture data. Later, a graph-

enhanced CBIR (gCBIR) system is described and the performance is compared to the tradition CBIR systems.

Finally, the conclusion of the thesis and the expected directions of future research about graph and data clustering, as well as gCBIR systems are discussed in Chapter 5.

## 2. GRAPH THEORY

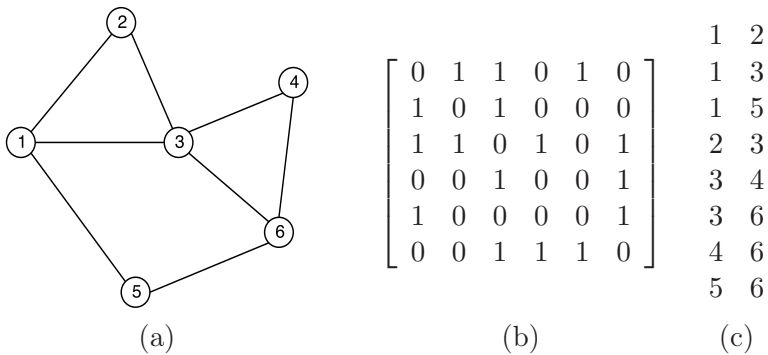
### 2.1 Basic concepts and graph representations

A graph is a data structure that represents the relationship between objects. Let  $G(V, E)$  be a graph, where  $V$  is the set of nodes and  $E$  is the set of edges. Let  $n_i \in V$  be the a node and  $e_{ij} \in E$  be the edge that connects nodes  $n_i$  and  $n_j$ , where  $i, j = 1, 2, \dots, N$ . For simplicity, we also use italic letters to denote nodes and an overline above two italic letters to denote the edge that connects the two end nodes. For example,  $a$  represents node  $a$  and  $\overline{ab}$  is the edge that connects nodes  $a$  and  $b$ . Note that parallel edges—edges that have the same end nodes—are not allowed in our definition.

Depending on the data, graphs can be categorized in different ways. If each edge is associated with a weight, the graph is a weighted graph. The weights can be either integer or real numbers. If each edge is associated with a direction, the graph is a directed graph. A bipartite graph is a graph that contains two disconnected sets of nodes. For example, the relationship between customers and products can be represented by a bipartite graph that contains the nodes of customers and the nodes of products. An edge links a customer node and a product node if the customer purchased the product. Trees and forests can also be considered a type of graph—a graph without cycles. A hypergraph is a generalization of the graph, where an edge may link two sets of nodes. Unless otherwise stated, a graph refers to an undirected and

unweighted graph in this thesis.

A graph is normally represented by its adjacency matrix. The columns and the rows represent the nodes in a graph. The elements in the adjacency matrix indicate whether the two nodes are connected by an edge. For a weighted graph, the value of the elements indicate the weights of the edges. Other than the adjacency matrix, edge list (or adjacency list) is also used to represent a graph. Each row in an edge list is a pair of nodes that are connected by an edge. For a weighted graph, the weights of the edges are shown in the third column. Fig. 2.1 shows a graph, the adjacency matrix representation, and the edge list representation. The adjacency matrix is used in graph analysis because of the mathematical advances in matrix analysis. While the edge list is more suitable for storage for its compact form.



**Figure 2.1** A unidirected graph (a), its adjacency matrix (b) and edge list (c) representations

## 2.2 Graph attributes and definitions

In this section, some basic definitions and attributes that are used in this thesis are explained.

The degree of a node is defined as the number of edges that are connected to the node. Given the adjacency matrix  $A$ , the degree of node



$n_i$  can be calculated by

$$d_i = \sum_{j=1}^N A_{ij}. \quad (2.1)$$

For a directed graph, in-degree and out-degree are defined as the number of edges that point to the node or leave from the node. Eq. 2.1 can be generalized to define the weighted degree of a node in a weighted graph. Weighted in-degree and weighted out-degree can be defined in a similar way.

Walk, trial and path are useful terms when studying the movement of an agent or the distance between the nodes in a graph. A walk is a sequence of nodes where the adjacent nodes in the sequence must be connected by an edge. A walk can be considered as the record of the visited nodes and edges when an agent travels on the graph. A trial is a walk without duplicate edges. A path is a trial without duplicate nodes. The length of a path is the number of edges in a path. The distance of two nodes is defined as the length of the shortest path that connects the two nodes.

A connected graph is a graph that any pair of nodes in the graph is connected by at least one path. A subgraph of graph  $G(V, E)$  is a graph in which the set of nodes and the set of edges are subsets of  $V$  and  $E$  respectively. An induced graph of  $G(V, E)$  is a subgraph of  $G(V, E)$  that contain all the edges in  $E$  that link the nodes in the induced graph. A component of graph  $G(V, E)$  is a connected induced graph of  $G(V, E)$ , and there is no edge in  $E$  that connects a node in the component to a node that is not in the component. The concept of the component is mainly used to analyze a disconnected graph since a connected graph has one component, which is the graph itself.

Ego-graph of a node is the induced graph that contains the node and

its neighboring nodes. Edge-ego-graph is the induced graph of the two end nodes of an edge and all the neighboring nodes of these two end nodes.

The Laplacian matrix of graph  $G$  is defined as

$$L = D - A, \quad (2.2)$$

where  $D$  is the degree matrix which is a diagonal matrix and the diagonal elements of this matrix are the degrees of the nodes as defined in Eq. 2.1, and  $A$  is the adjacency matrix. Laplacian matrix is a representation of a graph that has been extensively used in graph analysis, especially spectral graph theory and graph clustering [65].

### 2.3 Metrics

A number of attributes and metrics have been defined to describe or evaluate the properties of a graph.

The density of a graph is defined as

$$\text{density}(G) = \frac{|E|}{|V|(|V| - 1)}. \quad (2.3)$$

The diameter of a graph is the longest distance of any pair of nodes in a graph. Note that the diameter of a disconnected graph can be undefined or defined as infinite.

The connectivity of a graph determines the efficiency of information diffusion on a graph. The algebraic connectivity of a graph is defined as the second smallest eigenvalue of its Laplacian matrix, whose smallest eigenvalue being zero [66]. The larger the algebraic connectivity is, the better a graph is connected. If the graph is not connected, the

value of its algebraic connectivity is zero.

Clustering or community structure is a very common phenomenon in social networks [16, 67, 68]. As often seen in daily life, a group of closely related people is likely to be mutual friends. Similarly, for many types of graphs, the connections among the nodes in a cluster are much denser than the connections between nodes in different clusters.

When the role of a node is studied, its centrality is an important property [69]. A node with a larger centrality value plays a more important role when information flows on the graph. A simple measurement of the node centrality is to use its degree. However, a node with a large degree value is not necessarily a critical node. For example, node 7 in graph (a) in Fig. 2.2 has a low degree, but all information flows between the left and right side of the graph has to pass through it. For this reason, other centrality metrics have been introduced. Closeness centrality is defined as the average shortest path length of a node to other nodes in the graph. Betweenness centrality of a node is defined as the number of times that the shortest path of any pair of nodes passes through the node. Another important measurement is PageRank centrality. PageRank centrality of a node in a directed graph is defined as

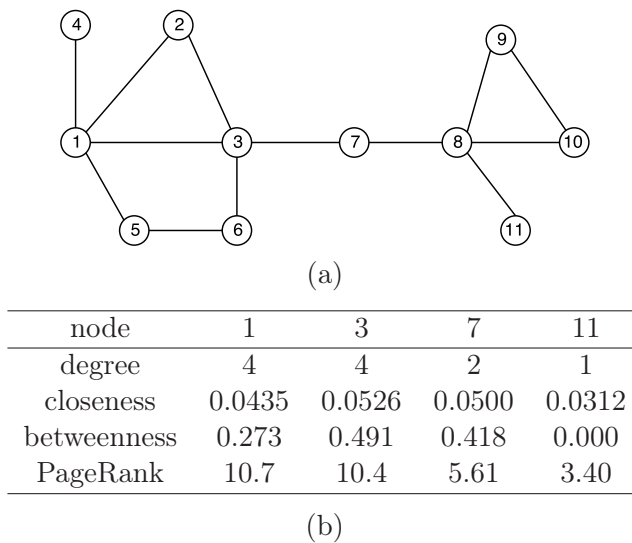
$$x_i = \alpha \sum_{j \in N^{in}(i)} A_{ij} \frac{x_j}{k_j^{out}} + \frac{1 - \alpha}{N}, \quad (2.4)$$

where  $x_i$  and  $x_j$  are the PageRank value of nodes  $i$  and  $j$ , respectively,  $N^{in}(i)$  is the set of nodes that connect to node  $i$  by a outgoing edge,  $k_j^{out}$  is the out-degree of node  $j$ ,  $N$  is the number of nodes, and  $\alpha$  is a damping factor that controls the scope of neighboring nodes that contribute to the PageRank value. Eq. 2.4 has the analytical solution as follows:

$$x = D^{out} (D^{out} - \alpha A)^{-1} \cdot \mathbf{1}, \quad (2.5)$$

where  $x = [x_1, x_2, \dots, x_N]^T$  is the vector of PageRank values of each node,  $D^{out}$  is a diagonal matrix of out-degrees. If the graph is an undirected graph and the damping factor  $\alpha$  is set to 1, the PageRank measurement is simply the degree centrality [7].

Fig. 2.2 shows a graph and the centrality measurement of some nodes. Notice that the PageRank centrality is roughly proportional of the degree centrality on an undirected graph [70].



**Figure 2.2** A graph and the centrality measurements of some nodes. The PageRank values are calculated with a damping factor  $\alpha = 0.85$ .

The centrality of the nodes is another important property that has been extensively studied. However, there have been very limited studies on the centrality or other measurements for edges [71]. Some centrality measurements for nodes can be applied to edges. For example, the betweenness centrality of an edge is defined in the same way as the betweenness centrality of nodes. Meo et. al. [72] defined  $k$ -path

centrality for node as

$$C^k(n_i) = \sum_{s \in V} \frac{\sigma_s^k(n_i)}{\sigma_s^k}, \quad (2.6)$$

where  $\sigma_s^k(n_i)$  is the number of  $k$ -paths (a path of length  $k$ ) originating from node  $s$  and passing through node  $n_i$ , and  $\sigma_s^k$  is the total number of  $k$ -paths originating from node  $s$ .  $k$ -path centrality of an edge is defined by replacing the node with an edge in the previous definition. Corresponding to the degree centrality of a node, the degree product of an edge—the product of the degrees of the two end nodes of the edge—is used as a centrality measurement for edges.

Authenticity measures whether an edge follows the clustering properties in a graph or not. Zhang et. al. [P2] defined the authenticity score as

$$a_{e_{ij}} = m_{e_{ij}} - e_{e_{ij}}, \quad (2.7)$$

where  $m_{e_{ij}}$  is the actual number of edges that connect the two groups of the neighboring nodes of the two end nodes of edge  $e_{ij}$ , and  $e_{e_{ij}}$  is the expected number of edges between these two groups of nodes. Edge authenticity can be used for graph clustering, outlier detection, and graph data preprocessing [P2, P4].

## 2.4 Random graph generation models

Random graph generation model has been an important research topic for the last several decades [73]. Numerous models have been proposed to generate graphs by stochastic processes to simulate or mimic real-world graphs. The most important aspect of a good random graph generation model is that the generated graphs show similar properties as those real-world graphs. This section will not give a thorough review of these models. Instead, we will only discuss some graph generation

models that have a huge impact on the research in this field and the models that are used in other sections of this thesis.

Erdős–Rényi model is the first random graph generation model and the most well-studied one [10]. A commonly used Erdős–Rényi model is  $G(n, p)$  model where a graph of  $n$  nodes is generated by randomly connecting two nodes by an edge with probability  $p$ . Many important properties have been found about the graphs generated by the Erdős–Rényi model. For example,  $\frac{\ln(n)}{n}$  is the sharp threshold of the connectedness of graph  $G(n, p)$ . If  $p > \frac{\ln(n)}{n}$ ,  $G(n, p)$  is almost sure to be connected. Otherwise, it is almost surely to be disconnected. However, random graphs generated by the Erdős–Rényi model lack many important properties that a real-world graph has. One significant shortcoming is that the generated graphs do not show clustering structure. Another important defect is that the degree distribution of the nodes is binomial, whereas the degree distribution of real-world graphs often follows the power law [74]. To overcome these shortcomings, many other random graph generation models have been proposed.

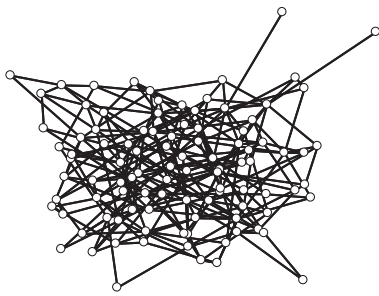
Preferential attachment, also named as “the rich get richer” or “cumulative advantage”, is a principle that an entity gets more of a certain asset if it has already possessed more of this asset. Barabási and Albert applied this principle to the random graph growth process in their model [75, 76]. The generation process starts from a single node and the nodes are added one by one. Every time a node is added, the probability that the new node is connected to node  $n_i$  is proportional to the degree of node  $n_i$ . The degree distribution of the graph generated by the Barabási–Albert model follows the power law in the form of  $p(k) = k^{-3}$ . The Barabási–Albert model is a very simple process that whenever a node is added to the graph, edges can only be added to connect the newly added node. Also, whenever an edge is added,

it can not be removed from the graph anymore. This is obviously not the case in many real-world graphs such as social network or web page graphs. Many extensions have been proposed to address these limitations [77, 78].

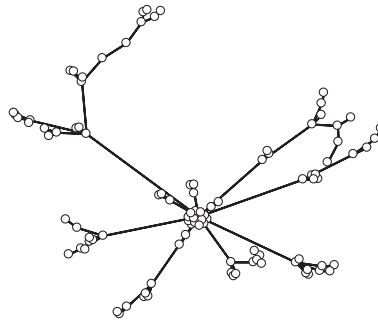
Small-world is another interesting property that many real-world graphs have [38, 79]. Small-world means that the average distance between any pair of nodes in a graph is limited by a small number. It is also known as “six degrees of separation” for social networks. A small-world graph is a graph where the average path length (APL) of any pair of nodes is proportional to  $\log(N)$ . APL of a graph generated by the Erdős–Rényi model follows  $APL_{ER} \propto \frac{\log(N)}{\log(k)}$ , where  $k$  is the average number degree. APL of a graph generated by the Barabási–Albert model follows  $APL_{BA} \propto \frac{\log(N)}{\log \log(N)}$  [79, 80, 81]. Watts and Strogatz proposed a model that generate graphs with not only the small-world property but also a constant clustering coefficient. With the Watts–Strogatz model, a ring type of lattice graph is first constructed. Then a process, similar to the Erdős–Rényi model, is applied to randomly rewire the edges with a certain probability.

Stochastic block model aims at generating graphs with another important property of real-world graphs: clustering. With this model, nodes are first divided into  $r$  groups. Then a  $r \times r$  probability matrix  $P$  is defined such that the element  $P_{ij}$  defines the probability of an edge is generated to connect nodes between group  $i$  and  $j$ . This model can be viewed as a generalization of the Erdős–Rényi model where all elements in matrix  $P$  are identical.

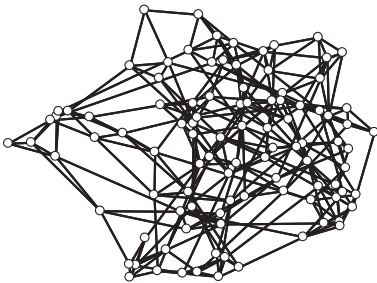
Fig. 2.3 shows the samples of random graphs generated by different random graph generation models.



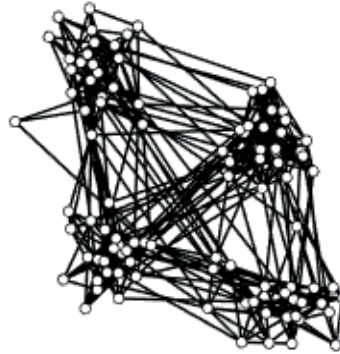
(a) Erdős-Rényi model



(b) Barabási-Albert model



(c) Watts-Strogatz model



(d) Stochastic block model

**Figure 2.3** Random graphs generated by different generation models



## 3. GRAPH CLUSTERING AND GRAPH-BASED DATA CLUSTERING

### 3.1 Graph clustering

Most graph data, especially graphs of social networks, are heterogeneous—the graph contains communities such that the density of edges in a community is much higher than the overall density of the whole graph [12, 18, 82, 83, 84]. Finding these communities is not only important to understand the underlying relationship between nodes, but also beneficial to computation and storage. Graph clustering is a technique to organize nodes into clusters such that the densely connected nodes are assigned to the same cluster and the connections between different clusters are sparse.

Graph clustering is a general term for many related techniques. Graph partition is a technique that divides nodes into a number of components such that the components are balanced—each component contains roughly the same number of nodes [24, 85, 86]. Graph partition becomes important in a distributed system when the graph data is too big to fit into the resource of a single computing unit. In this situation, the big graph data will be partitioned into a certain number of components and each component is processed separately by a single unit. To minimize the communication cost between different units, the links between the components must be minimized [87, 88].

If each node belongs to only one cluster, this type of clustering is called hard clustering. There are also situations when one node may belong to more than one cluster, this clustering technique is called soft clustering (also referred to as fuzzy clustering) [89]. Graph partition is normally hard clustering. Sometimes, some nodes may not belong to any cluster, this technique is called graph clustering with outlier detection. The nodes that are not associated with any cluster are recognized as outliers. Within the scope of this thesis, we only discuss the techniques related to hard clustering.

In many applications, it is not necessary to cluster the whole graph, or it is impractical to cluster the whole graph due to its size. Instead, we may be only interested in finding the cluster that contains a certain node. This technique is called local graph clustering (or community detection in some literature) [90, 91, 92, 93, 94].

The next section will discuss how to evaluate the performance of graph clustering algorithms and later show how the techniques are used in graph clustering.

## 3.2 Evaluation of graph clustering algorithms

It is a challenging task to evaluate the performance of different graph clustering algorithms [95, 96, 97]. The definition of the cluster structure in a graph is heuristic and many different mathematical models have been developed and applied. Depending on the actual application, one has to choose suitable models that match the expected cluster structure. We can use two types of methods to evaluate graph clustering algorithms depending on whether or not the ground truth data is available: external evaluation and internal evaluation.

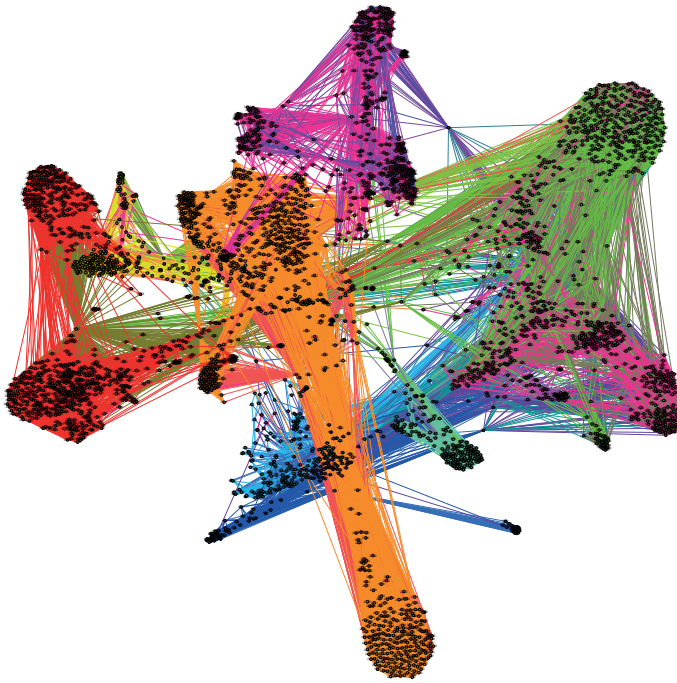
### 3.2.1 External evaluation

For some applications, clusters in the graph may be known from other sources [98]. For example, the graph structure of a synthetic graph is known if the graph is generated by a predefined clustering generation model, such as a stochastic block model (see Section 2.4) or caveman graph model [99]. For a real-world graph, the graph structure may be annotated by human experts or the structure is revealed by other hints. For example, it is well known that the clustering structure of Zachary’s karate club graph (as described in Section 1.2) is the same as how the club was split. The social network service website Facebook encourages its users to organize their friends into “circles”. Ego-Facebook is the graph data that were collected from the circles of 10 Facebook end users. The 10 clusters as the data were collected are used as the ground truth of this graph data. Fig 3.1 shows the ground truth of ego-Facebook graph.

When the ground truth data is available, clustering results can be evaluated using external evaluation methods. The external evaluation compares the partition of the nodes given by the clustering algorithms to the partition in the ground truth. Let  $X = \{X_1, X_2, \dots, X_r\}$  be the partition of the nodes given by a clustering algorithm, where  $X_i$  is the set of the nodes in cluster  $i$ , and  $r$  is the number of clusters. Each node  $n_i \in G(V, E)$  belongs to one of the partitions in  $X$ . Similarly, let  $Y = \{Y_1, Y_2, \dots, Y_s\}$  be the partition of the nodes in the ground truth and  $s$  is the number of clusters. The performance of a graph clustering algorithm can be evaluated by the following metrics.

- Rand index

Rand index uses pairs of nodes to evaluate partition  $X$  and the ground truth  $Y$  [100]. For every pair of nodes in  $V$ , if



*Figure 3.1* Ground truth of the ego-facebook graph data

the assignment of the two nodes agrees in  $X$  and  $Y$  (for example they are both in the same cluster in  $X$  and  $Y$ , or they are separated in different clusters both in  $X$  and  $Y$ ), the pair is marked as a correct assignment. If the assignment does not agree (for example, they are in the same cluster in  $X$ , but separated in  $Y$ ), the pair is marked an incorrect assignment. Rand index is defined as the percentage of correctly assigned pairs. Let  $C(X) = \{(n_i, n_j) | n_i \in X_k, n_j \in X_k, k \leq r\}$  be the set of node pairs that are assigned in the same cluster of partition  $X$ . Let  $D(X) = \{(n_i, n_j) | n_i \in X_k, n_j \in X_l, k \neq l, k \leq r, l \leq r\}$  be the set of node pairs that the two nodes are assigned in different clusters in partition  $X$ . Rand index is defined as

$$RI = \frac{|C(X) \cap C(Y)| + |D(X) \cap D(Y)|}{\binom{N}{2}}, \quad (3.1)$$

where  $N$  is the number of nodes.

- Adjusted Rand index

One drawback of the Rand index is that a randomly partitioned result may yield a high Rand index score because of the large value of the disagreed pairs [100, 101]. To avoid this problem, the adjusted Rand index is normally used. Adjusted Rand index is defined as

$$ARI = \frac{RI - E(RI)}{\max(RI) - E(RI)}, \quad (3.2)$$

where  $E(RI)$  is the expected Rand index value of a random partition and  $\max(RI)$  is the maximum Rand Index value. Adjusted Rand index has a value close to zero for random partitions.

- Normalized mutual information

Mutual information [102] is a concept that is used in probability theory to measure the dependency of two random variables. By

the definition of mutual information, it can be written as

$$\begin{aligned} I(X, Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \quad , \quad (3.3) \\ &= H(X) + H(Y) - H(X, Y) \end{aligned}$$

where  $I(X, Y)$  is the mutual information of random variables  $X$  and  $Y$ ,  $H(\cdot)$  is the marginal entropy of a random variable,  $H(X|Y)$  is the conditional entropy of random variable  $X$  given random variable  $Y$ , and  $H(X, Y)$  is the joint entropy of the two random variables  $X$  and  $Y$ . Mutual information gives a value of how much extra information is required to encode a random variable by knowing another random variable. The value of mutual information is between zero and  $H(X)$  or  $H(Y)$ .

The normalized mutual information (NMI) is defined as

$$R = \frac{I(X, Y)}{H(X) + H(Y)}. \quad (3.4)$$

NMI is commonly used as a measure for the performance of data clustering [103, 104] and graph clustering algorithms [P1, P2, 68, 105].

To evaluate a graph clustering algorithm, given the confusion matrix, NMI can be calculated by

$$NMI = \frac{-2 \sum_{i=1}^{C_A} \sum_{j=1}^{C_G} N_{ij} \log(N_{ij}N/N_{i-}N_{-j})}{\sum_{i=1}^{C_A} N_{i-} \log(N_{i-}/N) + \sum_{j=1}^{C_G} N_{-j} \log(N_{-j}/N)}, \quad (3.5)$$

where  $C_A$  is the number of clusters that the graph clustering algorithm determines,  $C_G$  is the number of clusters in the ground truth,  $N_{ij}$  is the element in row  $i$  and column  $j$  in the confusion matrix,  $N_{i-}$  is the sum of the  $i$ -th row,  $N_{j-}$  is the sum of the  $j$ -th column and  $N$  is the total number of nodes in the graph.

### 3.2.2 Internal evaluation

Very often the ground truth of the clustering structure is not available for real-world data [82, 83]. To evaluate the performance of graph clustering algorithms without the ground truth, internal evaluation metric can be used. Internal evaluation is based on the general definition of the clustering structure in a graph such that the clusters found by an algorithm shall be compact and the link between the clusters shall be sparse. Obviously, the choice of an internal evaluation metric depends on the actual application and the graph data. It shall be noted that many graph clustering algorithms find clusters by optimizing a predefined internal evaluation metric using various optimization methods. Thus comparing different graph clustering algorithms using internal evaluation metrics is often inappropriate. Furthermore, comparing the clustering algorithms that use the same fitness function is actually comparing the underlying optimization methods incorporate in the algorithms.

- Density-based metrics

Given a measurement of the density of a cluster in a graph, the performance of a graph clustering algorithm can be evaluated by the average density of the clusters that the algorithm finds. One typical definition of a cluster density, similar to graph density, is defined as

$$d(C) = \frac{2|E(C)|}{|C| \cdot (|C| - 1)}, \quad (3.6)$$

where  $C \subset V$  is the set of nodes in a cluster,  $E(C) \in E$  is the set of edges of the induced subgraph of the nodes in cluster  $C$ , and  $|\cdot|$  is the cardinality of a set.

One major limitation of this metric is that the score is greatly impacted by the size of the cluster [68]. In particular, small cliques,

which have a density of 1— the largest value of this metric, may greatly hinder the evaluation. Thus, an algorithm may simply improve the average density score by separating small cliques from a cluster.

- Cut-based metrics

Because of the limitations of the density-based metrics, cut-based metrics often give more reliable evaluations. Let  $K \subset V$  be the set of nodes of a cluster. Let  $K^c = V \setminus K$  be the complement set of  $K$ . The total degrees (or weight for weighted graphs) of cluster  $K$  can be calculated by

$$a(K) = \sum_{i \in K} \sum_{j \in V} A_{ij}. \quad (3.7)$$

The degrees (or weight for weighted graphs) of cut  $(K, K^c)$  is defined as

$$c(K) = \sum_{i \in K} \sum_{j \in K^c} A_{ij}. \quad (3.8)$$

The conductance of cut  $(K, K^c)$  is defined as

$$\varphi(K, K^c) = \frac{c(K)}{\min(a(K), a(K^c))}. \quad (3.9)$$

Besides conductance, other measurements of a cut are also commonly used. For example, normalized cut [65] is defined as

$$ncut(K, K^c) = \frac{c(K)}{a(K)} + \frac{c(K)}{a(K^c)}. \quad (3.10)$$

A lower conductance value or normalized cut value indicates a better cut of the graph.



Inverse relative density (IRD) [P1] is defined as

$$ird(K, K^c) = \frac{|E| - a(K) + c(K)}{a(K) - c(K)}. \quad (3.11)$$

Both the density-based metrics and the cut-based metrics are defined to evaluate a single cluster. These metrics can also be used to evaluate the performance of local graph clustering algorithms. For global graph clustering problems, the value of each cluster is first calculated, then the overall performance is evaluated by the statistics of all clusters, for example, the mean or the extreme value of all clusters.

### 3.3 Graph clustering methods

Various methods have been proposed for graph clustering problems. In this section, we briefly review some of the commonly used methods. In the next section, we will give details of random walk-based methods, which is not reviewed in this section. Note that even methods that fall into different categories, they are still closely linked with each other.

#### 3.3.1 Spectral graph clustering

Spectral clustering is one of the most popular graph clustering algorithms [65, 106, 107]. It can be easily implemented and often gives satisfactory results. To derive the spectral graph clustering method, we start from the normalized cut measurement defined by Eq. 3.10 as the fitness function. We further define an assignment vector  $x$ , such that

$$x_i = \begin{cases} \frac{1}{|K|} & \text{if } i \in K \\ \frac{1}{|K^c|} & \text{if } i \in K^c \end{cases}. \quad (3.12)$$

It can be derived from Eq. 3.10 that

$$ncut(K, K^c) = \frac{x^T Lx}{x^T Dx}, \quad (3.13)$$

where  $L$  is the Laplacian matrix as defined in Eq. 2.2 and  $D$  is the degree matrix of the graph. Minimizing Eq. 3.13 with regard to  $x$  in the discrete manner is NP-hard [65]. However, we can relax the problem by letting  $x$  to be a real valued vector with the constraint that  $x^T D\mathbf{1} = 0$ , where  $\mathbf{1}$  is the vector where all elements are 1s. This problem can be solved as the generalized eigenvector problem  $Lx = \lambda Dx$ . Let  $y = D^{1/2}x$ . It can be seen that  $y$  is the second eigenvector of the normalized Laplacian

$$L_{sym} = D^{-1/2}LD^{-1/2}. \quad (3.14)$$

After the eigenvector is calculated, we can cluster the nodes according to their corresponding values in the eigenvector. This principle can easily be extended to  $k$  clusters. Instead of using only one eigenvector, multiple eigenvectors can also be used [106]. Algorithm 1 shows the details of the spectral graph clustering algorithm.

Normally,  $k$ -means clustering is used to cluster the nodes after the feature vectors are calculated. However, any data clustering algorithm can be used in this step [108, 109].

It should be noted that if the target is to minimize the ratio cut measurement, which is defined as

$$rcut(K, K^c) = \frac{c(K)}{|K|} + \frac{c(K^c)}{|K^c|}. \quad (3.15)$$

Applying the same procedure as normalized cut, one ends up finding the eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix  $L$  of graph  $G(V, E)$  [110].

**given** adjacency matrix  $A$  of graph  $G(V, E)$  and the number of clusters  $k$

- Compute normalized Laplacian matrix  $L_{sym}$  by Eq. 3.14
- Compute the first  $k$  eigenvectors  $v_1, v_2, \dots, v_k$  of  $L_{sym}$
- Let  $V$  be the matrix with the columns  $v_1, v_2, \dots, v_k$
- Normalize the rows of  $V$  to have norm of 1
- Using the rows of  $V$  as features and cluster the data points into  $k$  clusters
- Assign the nodes to the corresponding cluster given by the data clustering algorithm

**Algorithm 1:** Spectral graph clustering using multiple eigenvectors

### 3.3.2 Fitness function optimization

A large number of graph clustering algorithms choose to optimize a predefined fitness function using different optimization strategies. Newman et al. [83, 111] studied the modularity of social networks and defined the modularity as

$$Q = \sum_i (e_{ii} - a_i^2), \quad (3.16)$$

where  $i$  is the index of a community,  $e_{ii}$  is the fraction of the edges that connect the nodes in community  $i$ , and  $a_i = \sum_j e_{ij}$  is the fraction of the expected number of edges that connect the nodes in community  $i$ . Spielman and Teng opted to take the conductance (Eq. 3.9) as the fitness function [94]. Starting from the similarity and dissimilarity of two nodes, Veldt et al. studied the disagreement of the clusters and

defined the fitness function as

$$\sum_{i,j \in E^+} (1 - \lambda)x_{ij} + \sum_{i,j \in E^-} \lambda(1 - x_{ij}), \quad (3.17)$$

where  $\lambda$  is a predefined weight,  $x_{ij} \in \{0, 1\}$  is the indicator of whether nodes  $i$  and  $j$  are in the same cluster,  $E^+$  is the set of edges that the two end nodes are in the same cluster, and  $E^- = E \setminus E^+$  is the complementary set of  $E^+$  [112]. In [113], Görke et al. evaluated graph clustering algorithms that use different density measurements as their fitness function using two greedy heuristics: vertex moving and cluster merging. They show that the vertex moving approach produces more reliable results than the cluster merging approach. They also show the limitations of using different density based measurement as the fitness function.

As a matter of fact, finding the optimal value of the fitness function defined earlier is NP-hard. However, different optimization strategies have been applied to find a locally optimal solution.

- Greedy Agglomeration [82, 83, 113, 114]

This method starts off with the state that each node is in its own cluster. At each iteration, the algorithm evaluates the improvement of the fitness function by merging each pair of the clusters and merges the pair with the largest improvement. This procedure is repeated until no improvement can be achieved by merging any pair of clusters. This method adopts the greedy optimization strategy and has a complexity of  $O((m+n)n)$ , where  $m$  is the number of edges and  $n$  is the number of nodes. Obviously, this approach does not guarantee global optimization. A clear limitation of this approach is that once a node is assigned to a cluster, it will not be moved to another one. To address this limitation, in [115], Blondel et al. alternatively apply two

phases. In the first phase, nodes are moved between communities to optimize the modularity. In the second phase, a new graph is constructed by merging the communities in order to improve the fitness function.

- Relaxation

Optimizing the fitness functions such as those defined in Eqs. 3.9 3.10 3.15 3.16, in a discrete manner is a NP-hard problem [116]. However, these problems can be relaxed by converting hard assignment into fuzzy assignment, similar to the strategy used in spectral clustering algorithms [65, 106, 107]. In [112], the authors studied the correlation clustering problem, whose objective is to minimize Eq. 3.17. This optimization problem is an integer linear programming problem and it can be relaxed to a linear programming problem and the solution can be found in polynomial time [117].

- Other optimization techniques

Given a fitness function, an optimization technique can be applied to find the optimal or a suboptimal solution. In [118, 119], simulated annealing is used to optimize Eq. 3.16. Duch and Arenas adopted extremal optimization approach in optimizing the modularity fitness function [120]. In their approach, nodes are first ranked according to their contribution to the fitness function before moving to other clusters. Then nodes are moved based on the probability  $P(q) \sim q^\tau$ , where  $q$  is the rank of the node and  $\tau$  is the extremal optimization constant. In [121], mean field annealing is used to optimize Eq. 3.16.

### 3.3.3 Data clustering-based methods

Data clustering is an important topic in machine learning and has been comprehensively studied for many decades. Many graph clustering

algorithms take advantage of the results achieved in data clustering. These algorithms first extract features for the nodes in the graph, and then apply a suitable data clustering method, such as  $k$ -means, to find clusters of nodes. The most critical part of these methods is to find suitable feature representations. Zhang et al. generate feature vectors for the nodes using the concept of limited random walks [P1]. In [122], Tian et al. learn node representations using a sparse deep autoencoder and take the activations of the last hidden layer as the features.

### 3.3.4 Model-based methods

Model-based methods assume that the graph is generated from a mathematical model and find clusters by optimizing a fitness function derived from the model. Chen et.al model a graph using a stochastic block model by which a random graph is generated with a higher probability to link in-cluster nodes than between-cluster nodes [123]. With the relaxation of the cluster matrix—a matrix that indicates whether the two nodes are assigned to the same cluster, the graph clustering problem is converted to a convex optimization problem and can be solved effectively. The Potts model [124] studies interacting spins positioned on a lattice structure. Each spin can be in one of the  $q$  states, where  $q$  is a predefined integer. Note that the Potts model is a generalized Ising model [125], in which spin can be in one of the two states. An energy function (Hamiltonian) is defined for the system. In [126, 127],  $q$ -state The Potts model is used to detect communities in a graph by minimizing the Hamiltonian function.

## 3.4 Random walk-based graph clustering

A random walk is a stochastic process in which one or multiple imaginary agents travel randomly on the nodes of a graph [20]. At each step,

an agent randomly selects a neighboring node with a certain probability and moves to it. The structure of the graph can be revealed by analyzing the probability distribution of the agent on each node or by studying the route that the agents traveled.

### 3.4.1 Random walk and Markov Chain

A random walk is normally modeled as a discrete-time Markov chain, where each node is considered as a state. Let  $t = 0, 1, 2, \dots, T$  be the time stamp and  $x^{(t)} \in \mathbb{R}^N$  be the vector where the element indicates the probability of the agent at each node at time  $t$ . The Markov chain can be written as

$$x^{(t+1)} = Px^{(t)}, \quad (3.18)$$

where  $P$  is the transition matrix whose element  $P_{ij}$  is the probability that the agent moves from node  $i$  to  $j$ . Given the probability vector  $x$  at time 0, Eq. 3.18 can be written as

$$x^{(t)} = P^t x^{(0)}. \quad (3.19)$$

Note that the Markov chain defined by Eqs. 3.18 or 3.19 is time-homogeneous—the transition matrix  $P$  is the same during the whole process.

There are multiple ways to define the transition matrix for a random walk on a graph. For an unweighted graph, the most common way is to assign equal probability to the neighboring nodes [20, 128]. In this approach, the transition matrix is written as

$$P_{ij} = \frac{A_{ij}}{\sum_{k=1}^n A_{kj}} \quad (3.20)$$

or

$$P = AD^{-1}, \quad (3.21)$$

where  $A$  is the adjacency matrix of the graph and  $D$  is the degree matrix. In [P1], Zhang et al. assign the equal probability to the current node such that the agent has the same probability to move to a neighboring node or stay in the current node. This is equivalent to adding a self-loop edge to each node of the graph. The transition matrix becomes

$$P = (I + A)(I + D)^{-1}. \quad (3.22)$$

This idea can be easily extended by assigning a different probability to stay at the current node and to move to a neighboring node. In this case,

$$P = (\alpha I + A)(\alpha I + D)^{-1}, \quad (3.23)$$

where  $\alpha$  can be considered as the weight of the self-loop edge. This transition matrix fixes the aperiodic problem that a graph may have [P1]. A random walk defined by Eq. 3.23 is also called lazy random walk when  $\alpha = \frac{1}{2}$ .

In [129], the authors assume that an agent travels to a neighboring node with a higher probability if the two nodes share more common neighbors. They define the transition matrix as

$$P_{ij} = \frac{1}{K_i} A_{ij} (c_{ij} + 1)^\gamma, \quad (3.24)$$

where  $c_{ij}$  is the number of common neighbors of nodes  $i$  and  $j$ ,  $K_i = \sum_k A_{ik} (c_{ik} + 1)^\gamma$  is a normalization term, and  $\gamma$  controls the bias towards the nodes with more common neighbors. If  $\gamma = 0$ , Eq. 3.24 is equivalent to Eq. 3.21.

Note that these transition matrices defined for unweighted graphs can be easily extended to weighted graphs [130].



### 3.4.2 Equilibrium state

We first introduce two important definitions for Markov processes. A Markov process is irreducible if for any two states  $i, j$ , there exists an integer  $t$  such that  $P_{ij}^t > 0$ . This means that it is possible to reach any state from another state. For the random walks on an undirected graph, this means that the graph is connected. Let  $T(x) = \{t \geq 1 | P^t(x, x) > 0\}$  be the set of times that the Markov process return to its starting state  $x$ . The period of state  $x$  is defined to be the greatest common divisor of elements in  $T(x)$ , which is denoted by  $\gcd(T(x))$ . A Markov process is called aperiodic if the period of all states are 1.

**Proposition 3.1** ([131]). *If  $P$  is irreducible,  $\gcd(T(x)) = \gcd(T(y))$  for all state  $x$  and  $y$ .*

Since the random walk on a connected graph is irreducible, it is easy to derive the next proposition.

**Proposition 3.2.** *A random walk on a connected graph that contains a self-loop is aperiodic.*

**Theorem 3.1** ([131]). *If  $P$  is irreducible, there exists a unique distribution  $\pi$  that satisfies  $\pi = P\pi$ .*

**Theorem 3.2** ([131, 132]). *If  $P$  is irreducible and aperiodic, with stationary distribution  $\pi$ , the distribution vector  $x$  of the Markov process converges to  $\pi$ , regardless of the initial distribution  $x^{(0)}$ .*

The distribution  $\pi$  is called the equilibrium distribution of the Markov process.

**Proposition 3.3.** *For an undirected, unweighted and aperiodic graph with transition matrix  $P = AD^{-1}$ , the equilibrium distribution is  $\pi =$*

$\frac{d}{2m}$ , where  $d$  is the degree vector of the graph and  $m$  is the number of total edges.

This proposition can be easily verified by the definition of equilibrium distribution.

According to Theorem 3.2, any random walk, regardless of the starting node, would end up with the equilibrium distribution if the walk lasts long enough. According to Proposition 3.3, the equilibrium distribution simply gives the degree centrality measurement. For a graph clustering task, we are more interested in the local structure of a graph rather than the global attributes of each node. In particular, when a walk starts from a seed node, we hope the walk will stay near the seed node and show the local structure around the seed node. Next, we show some techniques that can be used to limit the scope of a walk and use these techniques to study the local structure.

### 3.4.3 Variants of the Random Walk

After an agent starts its walk from a seed node, it is likely to visit these nodes in the cluster that the seed node belongs to more often than the nodes in other clusters because the number of connections between the clusters is smaller than that within the cluster. However, as the walk continues, the agent will escape from its own cluster and the probability distribution will converge to the equilibrium distribution. The idea of restricting the scope of a random walk is to limit the distance that an agent can travel. This technique makes it also suitable for local graph clustering problem—finding the community that includes the seed node without exploring the whole graph structure [68]. Next, we describe some techniques to restrict the range of the walk.

### Walks with a fixed number of steps

A straight-forward approach of restricting the scope of the walk is to set a fixed number of steps that an agent can travel. In [128], the authors studied the probability that the agent is at node  $j$  when started from node  $i$  after a fixed walking distance  $T$ . With the Markov chain defined by Eq. 3.19, this probability is  $P_{ji}^T$ . They define a distance measurement of two nodes based on the observation that two nodes should see the other nodes in a “similar way” if the two nodes are in the same community. Given the distance measurement of the nodes, they apply a greedy agglomerative method to group nodes into clusters. Similar techniques are also used in [130]. One major difficulty of using this technique is to decide the walking distance  $T$ . If  $T$  is too small, the local structure is not sufficiently explored. If  $T$  is too large, the agent may escape the current cluster and the probability distribution will reflect the global structure. If the density of the clusters varies greatly, it is impossible to find a  $T$  that is suitable for every node in the graph. To overcome this difficulty, in [21], the authors developed a non-homogeneous random walk where the walking length is determined stochastically. At each step, the probability of continuing the walk is determined by a probability function that is parameterized by the degree of the node.

### Walks with teleport

Another interesting approach is to apply a teleport operation during the walk—the agent can be teleported back to the starting node (or state) with a predefined probability at each step [133, 134, 135, 136]. This approach is also called PageRank random walk because of the great similarities to the PageRank algorithm [24]. The Markov chain

with teleporting is defined by

$$x^{(t+1)} = \alpha x^{(0)} + (1 - \alpha)Px^{(t)}, \quad (3.25)$$

where  $\alpha$  defines the probability of teleporting to the starting state.

**Proposition 3.4** ([133]). *For any fixed value of  $\alpha \in (0, 1]$ , the equilibrium state of a random walk defined by Eq. 3.25 has a unique equilibrium distribution*

$$\pi = \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t \left( P^t x^{(0)} \right). \quad (3.26)$$

Many global and local graph clustering algorithms have been developed based on this approach. In [133], the authors sweep over the approximate equilibrium distribution vector  $\pi$  to find cuts with nearly optimal conductance for local clustering task. The Repeated Random Walk (RRW) algorithm let the walks with teleport start from every node of the graph and the equilibrium distributions, named as affinity vectors, are collected [135]. Clusters are formed by associating the nodes with high affinity values into the same cluster. In [134], the authors first find initial clusters using walks with teleport and then merge the clusters if their overlapping coefficient is above a predefined threshold.

### Limited random walks

A novel approach, named Markov Cluster Algorithm (MCL), of using random walks to find clusters in a graph was introduced in [137]. MCL algorithm lets the agents start walking from every node in the graph simultaneously. The probability distribution of the agents can

be calculated by

$$X^{(t+1)} = PX^{(t)}, \quad (3.27)$$

where each column of  $X$  is the probability distribution of an agent at the nodes in the graph. The initial matrix  $X^{(0)} = I$ , where  $I$  is an identity matrix. If the process continues as defined in Eq. 3.27, this is a normal random walk and  $X^{(\infty)}$  would be a matrix that each column is the equilibrium distribution of the Markov chain. In MCL, the random walk is considered a flow over the graph. The node that absorbs more flows from other nodes is considered as a contractor. MCL algorithm tries to enhance the local contractor in each cluster during the random walk process. The walking process will end up in a state that all nodes in a cluster are attracted by the contractor nodes after a number of steps. To achieve this, MCL applies an expansion and an inflation operation at each step. Let  $X_{*,j}$  be the  $j$ -th row of matrix  $X$ . The expansion operation is a normal random walk step and matrix  $X$  is updated by Eq. 3.27. The inflation operation is simply an arithmetic operation on each element of  $X_{*,j}$  that enhances the larger values and decreases the lower values of this vector. After inflation, matrix  $X$  is normalized such that each column sums to 1. It should be noted that MCL is not a stationary Markov chain because of the nonlinear operation involved in the inflation operation.

MCL is an important approach in graph clustering because of its novelty in using random walks. However, there are certain limitations when using this method to find clusters in a graph. First, at each step, the system calculates the product of two matrices as specified in 3.27. This calculation is expensive if the number of nodes is large. Second, the algorithm cannot be efficiently parallelized since the expansion has to be executed after all agents completed the walking step. Third, the MCL algorithm tends to over cluster data when there is more than one contractor in a cluster.

In [P1], the authors developed a Limited Random Walk algorithm (LRW) by adopting the inflation and normalization scheme used in the MCL algorithm. LRW starts random walks from every node in the graph. After each step, the inflation and normalization procedure is applied to the probability distribution of each distribution vector. The walking continues until the equilibrium state is reached. LRW is a stationary Markov chain where the transition matrix stays the same during the whole walking process. This makes the walk of each agent possible to execute independently. Therefore, the implementation of LRW can be done in an embarrassingly parallel manner and thus the algorithm can use the computing resources of a high-performance computing (HPC) system efficiently. The walking procedure is called exploring phase since agents randomly walk on the graph to explore the local structure.

Let  $x$  be the probability distribution vector. The inflation operation is an element-wise super-linear function — a function grows faster than a linear function, for example a power function with exponent greater than 1. Let  $f(x)$  be the inflation function defined as

$$f(x) = [x_1^r, x_2^r, \dots, x_n^r]^T, r > 1. \quad (3.28)$$

The normalization operation normalizes vector  $x$  to have a sum of 1, defined as

$$g(x) = \frac{x}{\|x\|_1}, \quad (3.29)$$

where  $\|x\|_1 = \sum_{i=1}^n |x_i|$  is the L1-norm of vector  $x$ .

Given the transition matrix  $P$ , the walking step of LRW can be written as

$$x^{(t+1)} = g\left(f\left(Px^{(t)}\right)\right) \quad (3.30)$$

Note that the inflation operation defined in Eq. 3.28 is a nonlinear

function. The LRW process defined by Eq. 3.30 is not a canonical Markov chain. Thus, there does not exist an equilibrium state as seen in Theorem 3.1. However, the following theorem shows the existence of the fixed-point.

**Theorem 3.3.** *There exists a fixed-point  $x^*$  such that  $x^* = g(f(Px^*))$ .*

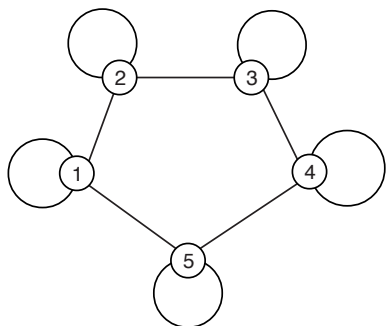
The proof of this theorem can be found in [P1].

Since LRW is a non-linear discrete dynamic system, there are no effective mathematical tools to fully understand the behavior of this procedure. However, when  $r = 1$ , LRW is simply a Markov chain process, in which the fixed-point  $x^*$  is the unique equilibrium state  $\pi$  (the global attractor). When  $r > 1$ , a fixed-point can be an unstable state and LRW may have limit cycles that oscillate around the fixed-point. Also, the system may contain more than one fixed-points. Fig. 3.2 shows an example of an oscillation occurring between two states in the extreme case where  $r \rightarrow \infty$ . Notice that the system may oscillate in different states depending on the initial state.

In practice, we chose  $r$  from  $(1, 2]$ . This makes LRW behave close to a linear system and oscillations are extremely rare.

The inflation and normalization operations defined in Eqs. 3.28 and 3.29 limit the scope that an agent explores. When  $r$  is large, the local attractors near the source node get boosted quickly thus the system will not explore far away from the source node. When  $r$  is close to 1, the agent explores further and get attracted to the attractors far away from the starting node.  $r$  plays a similar role as  $\alpha$  in Eq. 3.25.

The LRW algorithm first explores the graph using random walks starting from every node. Let  $x^{(*,i)}$  be the final probability vector for a random walk initiated from node  $i$ . We may treat each  $x^{(*,i)}$  as a fea-



(a) a circle graph

$$P = \begin{bmatrix} 1/3 & 1/3 & 0 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 0 & 1/3 & 1/3 \end{bmatrix}$$

(b) the transition matrix

$$x^{(a)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(c) oscillating state 1

$$x^{(b)} = \begin{bmatrix} 1/3 \\ 1/3 \\ 0 \\ 0 \\ 1/3 \end{bmatrix}$$

(d) oscillating state 2

**Figure 3.2** An example of oscillation during LRW. The circle graph shown in (a) has the transition matrix  $P$  as shown in (b). When  $r \rightarrow \infty$ , LRW may oscillate between two states  $x^{(a)}$  and  $x^{(b)}$  as seen in (c) and (d). It is easily seen that the system has a fixed-point  $x^* = [1/5, 1/5, 1/5, 1/5, 1/5]^T$ .



ture vector for node  $i$  and apply any data clustering algorithm, such as  $k$ -means, to cluster the nodes. However, since the nodes in a cluster share the common attractors, we can cluster the nodes by simply comparing the contracting nodes. With this approximation, the nodes can be clustered with a complexity of  $O(n)$  [P1].

#### 3.4.4 Computational complexity comparison

The computational complexity of a graph clustering algorithm is often dependent on the structure of the graph. If the graph is sparse and has clear cluster structure, many graph clustering algorithms, such as MCL and LRW, can find the clusters in an order of  $O(n)$ , despite the complexity of  $O(n^3)$  in the worst-case scenario. The computational complexity of data clustering-based algorithms is determined by the chosen data clustering algorithm. Table 3.1 shows some popular graph clustering algorithms, their techniques, and complexities.

### 3.5 Graph-based data clustering

Data clustering is an important topic that is extensively used in machine learning and pattern recognition [27]. Both data clustering and graph clustering can be categorized as unsupervised learning and the two subjects are highly related. As discussed in Section 3.3.3, data clustering techniques have been extensively used to solve graph clustering problems. Meanwhile, graph-based techniques are also often used for data clustering problems [P2, P4, 29, 30, 31, 130]. This section will present some data clustering algorithms that are based on graph techniques.

**Table 3.1** Comparison of the computational complexity of some popular graph clustering algorithms

algorithm	reference	technique	complexity
GN	[83]	fitness optimization	$O(nm^2)$
Clauset	[82]	fitness optimization	$O(n \log^2 n)$
Louvain	[115]	fitness optimization	$O(m)$
Guimera	[118]	fitness optimization	parameter dependent
Reichardt	[127]	model-based	parameter dependent
SAE	[122]	data clustering-based	$O(nki)^*$
N-Cut	[65]	spectral	$O(ni)^{**}$
MCL	[137]	random walk	$O(nk^2)^{***}$
LRW	[P1]	random walk	$O(nk)^{****}$

\* Complexity is determined by  $k$ -means algorithm, where  $k$  is the number of clusters and  $i$  is the number of iterations

\*\*  $i$  is the number of iteration used in matrix-free method to calculate the second eigenvector [138].

\*\*\*  $k$  is graph dependent. The worst-case complexity is  $O(n^3)$ .

\*\*\*\*  $k$  is graph depended. The worst-case complexity is  $O(n^3)$ .

### 3.5.1 Overview

To apply graph techniques for data clustering problems, the first step is to construct a graph from the input data. Normally the  $k$ -nearest neighbor (KNN) graph or the mutual  $k$ -nearest neighbor (MKNN) graph is used. We first take each data point as a node. For a KNN graph, two nodes are connected if one is in the set of  $k$ -nearest neighbors of the other. And for an MKNN graph, two nodes are connected only if both nodes are in the set of the  $k$ -nearest neighbors of the other node. Note that for a KNN graph, every node has a degree of  $k$ ; and for an MKNN graph, the maximal value of the degree is  $k$ . KNN graph is always connected, while MKNN may be disconnected.

After the graph is constructed, any graph clustering algorithm can be applied to partition the nodes into clusters. In [30] and [139], the authors utilize the “max flow-min cut” theorem and find clusters in a graph by finding the minimal cut that separated the communities in the graph. The method is plausible because the “max flow” problem can be solved in polynomial time [140]. The pseudo-code of this method is shown in Algorithm 2.

**given** a KNN or MKNN graph  $G(V, E)$  and parameter  $\alpha$

- Add a sink node  $t$  to graph  $G$  and let  $t$  connect to all the nodes in  $G$  with weight  $\alpha$ . Let  $G'$  be this expanded graph.
- Computer min-cut tree  $\mathcal{T}^*$  using Gomory-Hu algorithm [140]
- Remove  $t$  from  $\mathcal{T}^*$
- The components left in  $\mathcal{T}^*$  are the clusters found in  $G$

**Algorithm 2:** Max-flow(min-cut) data clustering algorithm

In [31], Zhang et al. studied the affinity of a node to a cluster and proposed the Graph Degree Linkage (GDL) algorithm. They define the

affinity measurement between a node and a cluster to be the product of the in-degree and out-degree of the node. The affinity measurement between two clusters is the sum of the affinity values of the nodes from one cluster to the other. A greedy optimization strategy is applied to find the partitions of the graph by minimizing the affinity between the clusters.

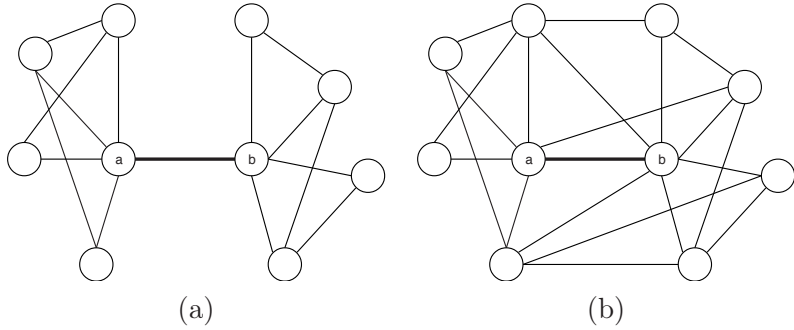
In [141], Zhang et al. propose to use path integral as the affinity measurement of a cluster. The path integral, which is a concept used in quantum mechanics [142], can be calculated as the sum of the weights of all paths in a cluster normalized by the square of the size of the cluster. The proposed method adopts the agglomerative approach and maximizes the overall path integral using a greedy search.

One of the advantages of using graph clustering techniques in data clustering problems is that many of these algorithms are able to handle noisy data—find clusters and detect outliers at the same time [P2, P4, 31, 141].

### 3.5.2 Data clustering using authentic score

Zhang et al. studied the authenticity of the edges in social networks [P2] and proposed a novel approach for data clustering problems. The idea was inspired by a common phenomenon in social networks that if the link between two people is authentic, the friends of these two people are likely to be connected as well. Now, consider edge  $\overline{ab}$  in the two graphs in Fig. 3.3. The edge  $\overline{ab}$  in graph (b) is more authentic than the one in graph (a), because the neighboring nodes of nodes  $a$  and  $b$  are more closely related in graph (b).

Next, two definitions that are used to study the neighboring nodes of an edge are given.



**Figure 3.3** The edge  $\overline{ab}$  in graph (b) is more authentic than the one in graph (a), since the neighboring nodes of nodes  $a$  and  $b$  are more closed related.

**Definition 3.4.** An edge-ego-network is the induced subgraph that contains the two end nodes of the edge, the neighboring nodes of these two end nodes, and the edges that link these nodes.

$S_{a \setminus b} = N_a \cup \{a\} \setminus \{b\}$  be the set of nodes that includes node  $a$  and its neighboring nodes except node  $b$ . Similarly,  $S_{b \setminus a} = N_b \cup \{b\} \setminus \{a\}$ .

**Definition 3.5.** A supporting edge of edge  $\overline{ab}$  is an edge that connects a node in set  $S_{a \setminus b}$  and a node in set  $S_{b \setminus a}$ .

Using the random graph generation model, the authenticity of an edge is defined as

$$a_{\overline{ab}} = m_{\overline{ab}} - e_{\overline{ab}}, \quad (3.31)$$

where  $m_{\overline{ab}}$  is the number supporting edges in the graph and  $e_{\overline{ab}}$  is the expected number of supporting edges if the graph is generated by a stochastic model. A high authenticity score indicates that the edge is likely to be authentic; meanwhile, a low score indicates that the edge is more likely to be an outlier.

To resemble the true structure of a social network, we use Preference

Attachment (PA) model to calculate  $e_{\overline{ab}}$ . Using the PA model, we get

$$e_{\overline{ab}} = \frac{1}{2m} \sum_{c \in S_a, d \in S_b} k_c k_d \quad (3.32)$$

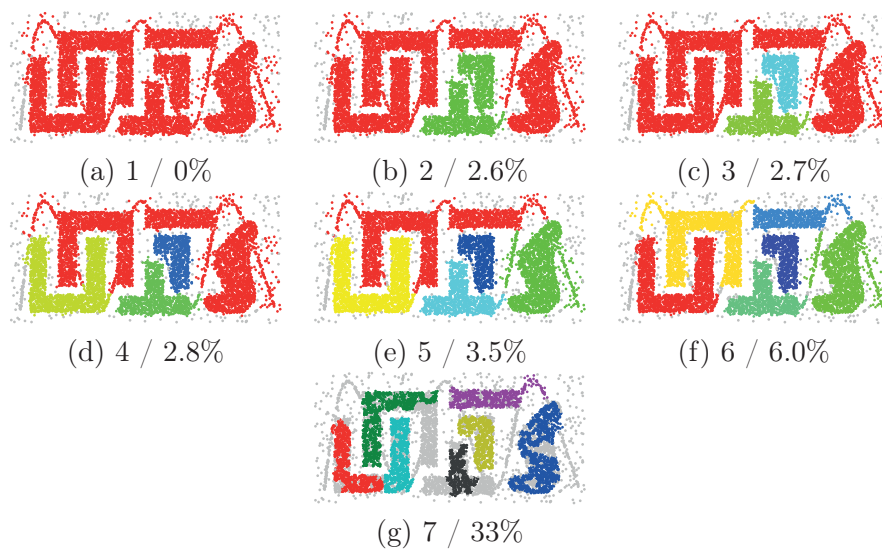
where  $m$  is the number of edges and  $k_c$  and  $k_d$  are the degree of nodes  $c$  and  $d$  respectively [P2].

Since a graph generated by a PA model may contain self-loops and duplicate edges, Eq. 3.31 is inaccurate, especially when  $e_{\overline{ab}}$  is large. To compensate for this bias, Eq. 3.31 can be modified as

$$a_{\overline{ab}} = m^{\frac{\gamma}{ab}} - e_{\overline{ab}}, \quad (3.33)$$

where  $\gamma$  is a real number greater than 1. In practice, we normally choose  $\gamma$  to be 2.

If a graph has a cluster structure, edges that link nodes in a cluster have a higher authenticity score than edges that link nodes in different clusters. In [P2], Zhang et al. developed a splitting approach to find clusters in a graph by gradually removing the edges from the graph according to their authenticity scores in ascending order. When a sufficient number of edges are removed, the graph will break into smaller components and these components can be used to identify clusters. A large number of edges have to be removed to break a real cluster into smaller components. Thus the number of clusters can be decided by analyzing the sizes of the components or the number of edges that have been removed. Fig. 3.4 shows this collapsing process of an MKNN graph generated from a toy data with cluster structure. The clusters in the data can be identified by the big components detected during the collapsing process. This example also shows that one can break the links between the clusters by only removing a small number of edges.



**Figure 3.4** The collapsing process of an MKNN graph when the edges are removed by the ascending order of their authenticity scores. For clarity, the edges are not shown in the graph. Big components identified during this collapsing process are shown with different colors. The numbers below the figure show the number of big components in the graph and the percentage of edges that have been removed

Zhang et al. proposed to use two methods to determine the clusters from this collapsing process [P2]. One method simply uses the sizes of the big components. When a cluster is broken during the collapsing process, it generates many small components and isolated nodes. If the number of clusters is known, we can find the optimal partition by maximizing the minimal component size. However, if the number of clusters is unknown, we can use the maximal conductance to determine the best partitions. The pseudo-code of these two methods is presented in Algorithm 3.

**given** MKNN graph  $G(V, E)$

- **calculate** authenticity scores of the edges in  $G$  by Eqs. 3.31 and 3.32
- **collapse** graph  $G$  by gradually removing the edges according to their authenticity scores
- **find** the big components from the collapsed graphs
- **assign** the isolated nodes and small components to the big components according to their authenticity scores
- **determine** the best partition by the size of the big components or the conductance values of the detected clusters

**Algorithm 3:** Data clustering by collapsing the MKNN graph using the edge authenticity scores

Experiments show that this algorithm is able to find clusters of complex shape. More important, it is robust to the density variations of clusters and outliers in the input data [P2, P4].

Next, we show that Algorithm 3 can reveal the true clustering structure in a graph generated by a stochastic block model (as described in Section 2.4), if certain conditions are met.

**Definition 3.6.** Stochastic  $r$ -block model is a stochastic block model



that generates a random graph with  $r$  equal-sized clusters with between-cluster edge probability  $p_b$  and within-cluster edge probability  $p_c$ .

A random graph generated by a stochastic  $r$ -block model is called a  $r$ -block random graph. Note that 2-block random graph is also called bisection random graph in the literature [143]. We use  $G(n_c, r, p_c, p_b)$  to represent a random graph generated by a stochastic  $r$ -block model, where each cluster contains  $n_c$  nodes,  $p_c$  and  $p_b$  are in the range of  $[0, 1]$ . Obviously, this graph contains  $n_c \cdot r$  nodes.

It should be noted that the subgraph of each cluster is generated by a Erdos-Renyi model  $G(n_c, p_c)$ . If  $p_c > p_b$ , the generated graph will show a clear cluster structure.

**Definition 3.7.** If a partition of the graph matches the block structure of how the graph was generated, the partition is an exact recovery of the ground truth.

**Definition 3.8.** If a partition of the graph satisfies the condition that any two nodes in a partition belong to the same cluster in the ground truth, the partition is an incomplete recovery of the ground truth.

Next, we first study some theoretical results of the authenticity scores on graphs generated by a stochastic  $r$ -block model. Later, we will show a sufficient condition that the authenticity scores can be used to completely recover the ground truth of a  $r$ -block random graph.

Let subscript  $c$  denote an edge within a cluster and subscript  $b$  denote an edge between clusters. According to Eq. 3.31, the authenticity score for an edge within a cluster is  $a_c = m_c - e_c$ . Similarly, the authenticity score for an edge between clusters is  $a_b = m_b - e_b$ .

**Proposition 3.5.** For a random graph  $G(n_c, r, p_c, p_b)$  generated by a stochastic  $r$ -block model,  $E(m_c) > E(m_b)$  if  $n_c > r$ ,  $n_c > 3$ , and

$p_c > \frac{(n_c-1)^2}{(n_c-2)(n_c-3)}p_b$ , where  $E(\cdot)$  is the expected value.

*Proof.* We first list all cases that a supporting edge is generated by the stochastic  $r$ -block mode. Let edge  $\overline{ab}$  be the edge to be investigated. A supporting edge is created in one of the following two cases:

Case 1: nodes  $c$  and  $d$  are randomly selected and edges  $\overline{ac}$ ,  $\overline{cd}$  and  $\overline{db}$  are created according to the corresponding probabilities. Edge  $\overline{cd}$  is a supporting edge for edge  $\overline{ab}$ .

Case 2: node  $c$  is randomly selected and edges  $\overline{ac}$  and  $\overline{cb}$  are created according to the corresponding probabilities. Edges  $\overline{ac}$  and  $\overline{cb}$  are both supporting edges for edge  $\overline{ab}$ .

From these two cases, we can list all scenarios that a supporting edge for edge  $\overline{ab}$  is generated. For a within-cluster edge, where nodes  $a$  and  $b$  are in the same cluster, we have the following scenarios for case 1:

- Scenario w1: nodes  $c$  and  $d$  are in the same cluster as nodes  $a$  and  $b$ .
- Scenario w2a: node  $c$  is in the same cluster as nodes  $a$  and  $b$ ; node  $d$  is in a different cluster
- Scenario w2b: node  $d$  is in the same cluster as nodes  $a$  and  $b$ ; node  $c$  is in a different cluster
- Scenario w3: nodes  $c$  and  $d$  are in a cluster other than the cluster that contains nodes  $a$  and  $b$
- Scenario w4: nodes  $c$  and  $d$  are in different clusters, and none of them are in the same cluster as nodes  $a$  and  $b$

And the following scenarios are for Case 2:

- Scenario w5: node  $c$  is in the same cluster as nodes  $a$  and  $b$
- Scenario w6: node  $c$  is in a different cluster than nodes  $a$  and  $b$

Similarly, for a between-cluster edge, where nodes  $a$  and  $b$  are in different clusters, we have the following scenarios for Case 1:

- Scenario b1: node  $c$  is in the same cluster of node  $a$ ; node  $d$  is in the same cluster as node  $b$
- Scenario b2a: node  $c$  is in the same cluster as node  $a$ ; node  $d$  is in a cluster that is different from nodes  $a$  and  $b$
- Scenario b2b: node  $d$  is in the same cluster as node  $b$ ; node  $c$  is in a cluster that is different from nodes  $a$  and  $b$
- Scenario b3: nodes  $a$ ,  $b$ ,  $c$ , and  $d$  are all in a different cluster
- Scenario b4: nodes  $c$  and  $d$  are in the same cluster, that is different from the clusters containing node  $a$  or  $b$

And the following scenarios are for Case 2:

- Scenario b5a: node  $c$  is in the same cluster as node  $a$
- Scenario b5b: node  $c$  is in the same cluster as node  $b$
- Scenario b6: node  $c$  is in a cluster other than the one containing nodes  $a$  or  $b$

Next, we calculate the expected number of edges for each scenario. Since  $E_c(m_{\overline{ab}})$  can be calculated in a similar manner for each scenario, we only provide the procedure for Scenario w1 in this thesis.

- Scenario w1: We randomly select nodes  $c$  and  $d$  from the cluster containing nodes  $a$  and  $b$ . The probability of generating edges  $\overline{ac}$ ,  $\overline{bd}$ , or  $\overline{cd}$  is  $p_c$ . Note that edge  $\overline{cd}$  is also a supporting edge if edges  $\overline{ad}$ ,  $\overline{bc}$  and  $\overline{cd}$  are all generated. So the probability that edge  $\overline{cd}$  is created as a supporting edge is  $2p_c^3$ . The number of ways of selecting nodes  $c$  and  $d$  is calculated by combinations  $\binom{n_c-2}{2}$ . Thus the expected number of supporting edges in this scenario is  $E_c(m_{\overline{ab}}) = 2\binom{n_c-2}{2}p_c^3$ .

Table 3.2 shows all scenarios that a supporting edge is created and the expected value of the number of supporting edges for that scenario. To simplify the notation, we use bracket  $[\cdot]$  to indicate the association of the nodes: nodes in the same bracket mean that they are in the same cluster, and nodes in different brackets are in different clusters. For example  $[a, b], [c, d]$  mean that nodes  $a$  and  $b$  are in the same cluster and nodes  $c$  and  $d$  are in another cluster. Notice that the corresponding scenarios for within-cluster edge and between-cluster edges are aligned in the same row. We will see that this alignment helps us to compare the expected values of the supporting edges in each row.

Given  $n_c > r$  and  $p_c > \frac{(n_c-1)^2}{(n_c-2)(n_c-3)}p_b$ , we can easily verify that  $E_c(m_{\overline{ab}}) > E_b(m_{\overline{ab}})$  in each row. By the way of how the supporting edges are created, the expected value of the number of supporting edges for edge  $\overline{ab}$  is the sum of  $E(m_{\overline{ab}})$  of each scenario in Table 3.2. Thus given  $n_c > r$ ,  $n_c > 3$ , and  $p_c > \frac{(n_c-1)^2}{(n_c-2)(n_c-3)}p_b$ , we have  $E(m_c) > E(m_b)$ .  $\square$

**Proposition 3.6.**  $E(e_c) = E(e_b)$  for an  $r$ -block random graph.

*Proof.* According to Eq. 3.32,  $e_{\overline{ab}}$  is a polynomial of  $k_i$ , where  $k_i$  is the degree of a node in the edge-ego-network of  $e_{\overline{ab}}$ . Since  $E(k_i)$  is same for all nodes in a  $r$ -block random graph, we have  $E(e_c) = E(e_b)$ .  $\square$

**Table 3.2** The scenarios that supporting edges are created and the expected value of the number of supporting edges in this scenario. Each row represents one scenario for a within-cluster edge and a between-cluster edge. The second column is the description of the scenario for a within-cluster edge. The third column is the expected value of the number of supporting edges for this scenario. The fifth column is the description of the scenario for a between-cluster edge. And the last column is the expected value of the number of supporting edges for this scenario.

scenario	within-cluster edge	$E_c(m_{ab}^-)$	scenario	between-cluster edge	$E_b(m_{ab}^-)$
w1	$[a, b, c, d]$	$2 \binom{n_c-2}{2} p_c^3$	b1	$[a, c], [b, d]$	$(n_c - 1)^2 p_c^2 p_b$
w2	$[a, b, c], [d]$ or $[a, b, d], [c]$	$2(n_c - 2)(n - n_c) p_c p_b^2$	b2	$[a, c], [b], [d]$ or $[a], [c], [b, d]$	$2(n_c - 1)(n - 2n_c) p_c p_b^2$
w3	$[a, b], [c], [d]$	$2 \binom{r-1}{2} n_c^2 p_b^3$	b3	$[a], [b], [c], [d]$	$2 \binom{r-2}{2} n_c^2 p_b^3$
w4	$[a, b], [c, d]$	$2(r - 1) \binom{n_c}{2} p_c p_b^2$	b4	$[a], [b], [c, d]$	$2(r - 2) \binom{n_c}{2} p_c p_b^2$
w5	$[a, b, c]$	$2(n_c - 2) p_c p_b$	b5	$[a, c], [b]$ , or $[a], [b, c]$	$2(n_c - 1) p_c p_b$
w6	$[a, b], [c]$	$2(n - n_c) p_c^2$	b6	$[a], [b], [c]$	$2(n - 2n_c) p_c^2$

**Proposition 3.7.**  $E(a_c) > E(a_b)$  for an  $r$ -block random graph if  $n_c > r$ ,  $n_c > 3$  and  $p_c > \frac{(n_c-1)^2}{(n_c-2)(n_c-3)}p_b$ .

*Proof.* This statement is obvious given Propositions 3.5, 3.6 and the definition of authenticity score in Eq. 3.31.  $\square$

**Theorem 3.9.** *Authenticity scores can asymptotically find the complete recovery of a  $r$ -block random graph if  $p_c > p_b$ .*

*Proof.* Since  $r$  is a constant and  $n_c = n/r$ ,  $n_c > r$  when  $n > r^2$ . It is also obvious that

$$\lim_{n \rightarrow \infty} \frac{(n_c - 1)^2}{(n_c - 2)(n_c - 3)} = 1.$$

According to Proposition 3.5, we have  $E(m_c) > E(m_b)$  if  $n_c > r$ ,  $n_c > 3$  and  $p_c > \frac{(n_c-1)^2}{(n_c-2)(n_c-3)}p_b$ . By the law of large numbers [144], we can find a value  $T$  in the range of  $(E(m_c), E(m_b))$ , such that

$$\lim_{n \rightarrow \infty} \Pr(m_c > T) = 1, \quad (3.34)$$

and

$$\lim_{n \rightarrow \infty} \Pr(m_b > T) = 0. \quad (3.35)$$

Considering Proposition 3.6, we have

$$\lim_{n \rightarrow \infty} \Pr(a_c > T) = 1, \quad (3.36)$$

and

$$\lim_{n \rightarrow \infty} \Pr(a_b > T) = 0. \quad (3.37)$$

From Eqs. 3.36 and 3.37, if  $n$  is sufficiently large, by removing all the

edges whose authenticity scores are below  $T$ , we can split the graph into  $r$  components such that each component corresponds to a cluster in the ground truth. By this partition, we find the complete recovery of the ground truth.  $\square$

Theorem 3.9 indicates that using the authenticity score one can find the complete recovery of a  $r$ -block random graph if  $n$  is sufficiently large. Note that Proposition 3.7 shows that the expected value of the authenticity scores of within-cluster edges is higher than that of the between-cluster edges. By removing the edges according to their authenticity scores, we are able to break the links between the clusters first. The large components of the collapsed graph are subsets of the nodes in one cluster. Another technique to improve the performance is to re-calculate the authenticity scores as edges are removed. However, our experiments show that a straightforward implementation is sufficient to find good clusters of the input data [P4].

### 3.5.3 Computational complexity analysis

The first step in using graph techniques for data clustering is to construct a KNN or MKNN graph from the input data. Let  $k$  be the number of neighbors when constructing a KNN or MKNN graph. The computational complexity of using a brute-force method is in the order of  $O(n^2)$ . Callahan et al. showed that, theoretically, a KNN graph construction takes  $O(n \log n + nk)$  [145]. Fast methods to approximate a MKNN graph are also available [146]. For example, Connor et al. claimed a method with a complexity of  $O(nk \log k)$  [147].

Table 3.3 shows the computational complexities of some data clustering algorithms using graph techniques.

**Table 3.3** Computational complexity comparison of some data clustering algorithms that are based on graph techniques

Algorithm	Complexity
GDL [31]	$O(n^2)$
k-means [148, 149, 150]	$O(ni)$ *
a-link [148, 150]	$O(n^2 \log n)$
N-Cut [65]	$O(ni)$ **
DBSCAN [151, 150]	$O(n \log n)$
authenticity score [P2, P4]	$O(k^3n + n \log n)$ ***

\*  $i$  is the number of iterations. In practice,  $i$  is difficult to estimate and in the worst case it is super-polynomial [152].

\*\*  $i$  is the number of iterations. Shi and Malik claimed that  $i$  is typically less than  $O(n^{1/2})$  [65].

\*\*\*  $k$  is the number of neighbors when constructing the KNN or MKNN graph.

### 3.6 Summary

Graph clustering is an important task in graph analysis and data mining. Given a clustering metric, one can apply different optimization techniques to find a good partition of the graph [83, 94, 111, 112]. Spectral clustering algorithms relax this discrete optimization problem by using continuous values and show that the spectral components (eigenvectors) of the Laplacian can be used to find clusters of the graph [65]. However, graph spectral clustering algorithms experience difficulties in computation when the size of the graph is large. Most of the existing algorithms are based on sequential optimization techniques. Thus they cannot efficiently use the resources of high-performance computing facilities. In [P1], Zhang et al. proposed a random walk-based graph clustering algorithm that can easily be implemented in an embarrassing parallel paradigm.

The focus of this thesis is on static graph clustering. However, in real-



world applications, a graph data may change over time: new nodes and edges are added to the graph; existing nodes and edges are removed from the graph; the properties of nodes and edges are changed [153]. Graph clustering for dynamic graph data is an important research topic with broad applications. The proposed LRW graph clustering algorithm [P1] focuses on the association of nodes that are close to each other. A change in a graph only affects the exploration results of the neighboring nodes and the merging phase can be executed efficiently. Thus the algorithm is also suitable for dynamic graph clustering. The study of graph clustering on dynamic graphs will be one of the focus in the future.

Graph clustering and data clustering are similar tasks for data analysis and pattern recognition. The two techniques are closely related. When using data clustering techniques for graph related problems, features for each node are extracted based on the structure of the graph. Random walk or graph structure embedding are normally used for this purpose [P1, 122]. After the features are extracted, any data clustering technique can be used to cluster the nodes. When applying graph techniques in data clustering, we first construct a KNN or an MKNN graph out of the input data. Then, any graph clustering algorithm can be used to find the clusters in the KNN or MKNN graph. In [P2, P4], the authors showed a simple splitting method based on the authenticity of the edges. In this section, we proved that algorithms using the authenticity scores can asymptotically find the complete recovery of the clustering structure of the graph generated by a stochastic  $r$ -block model. Experiments show that this method gives consistently satisfactory results regardless of the density of the cluster, the size of the cluster and the presence of noise.



## 4. CONTENT-BASED IMAGE RETRIEVAL WITH GRAPH TECHNIQUES

### 4.1 Introduction to content-based image retrieval

Retrieving useful information from a large dataset is an important task in this era of data explosion. Using web search engines, users have got accustomed to retrieving useful web pages and documents by text-based queries. However, retrieving multimedia content, such as image, audio, and video, is a more challenging task. To retrieve information from a large multimedia database using a text-based search engine, users have to enter queries in text format (keywords, sentences of a natural language, or statements of a certain query language) to describe the content he/she is interested in. This approach heavily relies on the text-based retrieval techniques. However, it is difficult to precisely and completely describe the content of multimedia items using any language. Another difficulty is that this approach also requires detailed descriptions of the items in the multimedia database. However, for a large multimedia database, there is always inadequate text descriptions. Earlier attempts of annotating large multimedia databases have shown that the annotations are ambiguous, erroneous and deficient [154, 155, 156].

Because of the difficulties faced by text-based search technologies, content-based image retrieval (CBIR) has become a critical topic in computer vision and multimedia content retrieval. With a CBIR sys-

tem, a user provides one or multiple query images and the system returns similar images according to the content of the query image [52, 56]. The retrieval results are normally presented to the user in the order of their relevance to the query image(s).

CBIR systems can be categorized into two types. One is designed to serve for a specific use case, for example, a CBIR system that retrieves pictures of furniture that matches the style of the furniture in a given query image, or a system that retrieves pictures of animals of the same breed as the one in a query image. Another type of CBIR system serves for general use cases, similar to search engines on the Internet. For better user experience, a general CBIR system needs to understand the purpose of the query—the intention of the user and the expectation of the retrieved result. Unlike text-based system, where the intention can be elaborated by providing more description to the query, for example, “running Labrador dog” and “layered birthday cake”, a CBIR system has to determine the intention using clues from the query images and knowledge gained from previous records.

#### **4.1.1 Purpose of a query**

The purposes of a query can be summarized into two broad categories depending on whether the user knows what they are looking for.

- Finding “similar” images or a specific group of images from the dataset

In this category, the user has a clear definition of “similarity” when he/she triggers the search. The results returned from the CBIR system must match the definition of “similarity” and the CBIR system has to be able to rank images according to this definition. The majority of researches on CBIR systems fall into

this category [52, 157, 158, 159]. The target of the search can be a concrete concept or an abstract concept.

- The target can be a concrete concept, such as “the specific dog breed in this picture”, “this type of vehicle” or “more pictures of this person”. When the target is clearly defined and understood, the retrieval problem becomes a classification problem and the CBIR system will return the pictures of the same class as the query images. With the fast development of pattern recognition, especially recent advances in deep learning technologies, systems that understand 1000 classes can perform in a level close to or even better than human beings [160, 161]. Automatic face recognition was also reported to outperform human accuracy [162, 163]. A deep learning based CBIR system normally performs well in this case, as long as the concept is precisely known.
  - It is also possible that the target concept is abstract and cannot be matched to a concrete class, for example, “pictures of the same artistic style of the query image”, “pictures as peaceful as this one”, or “pictures arousing similar sentiment”. Since the definition of “similarity” is abstract, it is difficult to apply the aforementioned classification approach. One has to clearly define a measurement of “style” or “sentiment” and find corresponding features to rank images. There has been limited related research on this topic [164, 165] and the performance of CBIR systems are often unsatisfactory in this case.
- Seeking the answers to a question related to the query image

In this category, the user is looking for images that may help him/her to answer a question related to the query image. Although this type of use case is common, there has been little

research in this area from CBIR perspective [166, 167]. One of the difficulties of this type of use case is that there are numerous questions that can be initiated from an image. Fig. 4.1 shows a query image and some questions related to the image.



- What is this event?
- Why a military vehicle is on the street?
- Where is this place?
- Who are the reporters in the picture?
- What is the model of the vehicle in this picture?

*Figure 4.1 A query image and a number of questions related to the image*

With the query image itself, it would be impossible for a system to figure out which question to answer. The user may elaborate the query with additional text information, or the system can present the results in groups and each group answers a specific question. An ideal system may answer these questions directly in text form. However, presenting the relevant images is often more convenient and brings extra information to the user.

The purpose of a query also affects how the results should be presented to the user. Sometimes, a user may expect to retrieve either a certain group of images or one specific image, for example, when querying “this breed of dog” or “this specific dog”. In another situation, the user may want to extend the scope of the query image and expect to see more diverse results, for example, “show me different people with

this type of hairstyle”. In the latter case, the diversity of the retrieved images must be kept so that the results are not “too similar” to the query image.

#### 4.1.2 Gaps in CBIR systems

Researchers have realized certain gaps that a CBIR system suffers from to fulfill the requirement of users [168]. The most important one is the semantic gap that describes the disparity between a user and a CBIR system when interpreting an image [159, 169]. With different purposes of a query, the interpretations of the images may be totally different and the definition of similarity is also different. For example, given the query image in Fig. 4.1, a user may interpret the image as a serious public security event, whereas the CBIR system interprets the image as a vehicle. As discussed in the previous section, it is difficult for a CBIR system to reduce this gap without understanding the purpose of the query.

Feature gap (as known as sensor gap [170]) refers to the situation that a CBIR system does not have effective features to evaluate similarities between images, even when the concept of “similarity” is clearly defined. This may be due to the limitation of the technology in image understanding, such as inferior performance in image classification under certain conditions [171]. More often, the functionality of a CBIR system cannot cover all possible use cases. A system that is designed to recognize different breed of dogs may not be able to distinguish different types of vehicles.

In [168], the authors also defined the performance gap and the usability gap. These two gaps address the same question of how a user can easily and quickly find and locate images that he/she is interested in from the results provided by a CBIR system.

### 4.1.3 Architecture of a CBIR system

A typical CBIR system consists of the following key components [159].

- Feature extraction

This component is responsible for extracting relevant features for the images in the dataset and the query images. The features can be low-level features (such as color features, texture features, and shape features[159]), middle-level features (such as SIFT and HOG features [172]) or high-level features (such as class-specific representations of an image given by a deep neural network [57]). Feature extraction is normally executed offline and the features are quantized and stored in a database for fast access [173]. Feature extraction for query images is executed online.

- Similarity measurement

To measure the similarity between a query image and images in the dataset, a metric needs to be defined. The most common measurement is the Minkowski metric, that is defined as

$$d(x, y) = \left( \sum_{i=1}^d |x_i - y_i|^r \right)^{1/r}, \quad (4.1)$$

where  $x$  and  $y$  are features of two images,  $d$  is the dimension of the feature,  $r$  is a constant, and  $r \geq 1$ . In particular, Eq. 4.1 is the Euclidean distance when  $r = 2$  and the Manhattan distance when  $r = 1$ . When multiple features are used, the similarity measurements can be assembled using a statistical method to generate an overall similarity score. Retrieved images are ordered according to their similarities to the query image. In a large-scale CBIR system, involving datasets with millions or billions of images. The CBIR system may just return the  $k$  nearest



images for fast retrieval instead of ranking all images. To further speed up the process, an approximation of the  $k$ -nearest neighbor search is often used [174].

- Presentation of retrieval results

After the  $k$  nearest images are retrieved from the dataset, a CBIR system normally displays the results as a list of images ordered by their similarities. To reduce the semantic gap and improve the user experience, some CBIR systems incorporate relevance feedback—a technique that refines the results with the help of users' feedback [175]. Some other systems allow a user to either select features that are used for similarity comparison or refine retrieval results from the keywords that the user give to a query image [176, 177]. In [178], the CBIR system organizes images in the dataset into a tree structure using a clustering algorithm and allows users to browse the retrieval results in a hierarchical view. This approach combines the visual content and the semantics, thus making it easier for users to locate the target images.

A good CBIR system shall use not only the visual content of the images but also all available information, such as tags, annotation, date, location and surrounding texts. A key requirement for a general CBIR system is the ability to handle large and continuously evolving dataset. All operations, including feature extraction and similarity measurement, must be fast and efficient to provide real-time response. The retrieved results shall be presented with good user experience and a method shall be available for users to refine the query and locate the target images quickly. The next subsection will show how graph techniques can help improve the user experience of CBIR systems.

## 4.2 Visual-semantic graph

Graph-based techniques have been extensively used in CBIR systems. In [179], the authors built a visual similarity graph where the nodes are the images and the weight of an edge is the similarity score of the two images. Taking the relevance feedback input to the system, they rank an image based on the probability of a random walker hitting a relevant image before hitting a non-relevant image, given that the walk started from that image. In [180], the authors used multi-graph learning to incorporate user feedback and generated the re-ranked results. Given the initial retrieval result and the relevant feedback, the system first constructs multiple visual similarity graphs—each using a different feature. Then multi-graph learning with inter-graph and intra-graph constraints is applied to generate a fused graph and the retrieval results are reordered. Sometimes, an image dataset may contain annotations or tags. In [181] the authors constructed two types of similarity graphs: a visual similarity graph where nodes are the images and the weights of the edges are the similarity values calculated from the image descriptors (features); a semantic similarity graph where nodes are the term-sets (or tags) and the weights of the edges are the association level of the term-sets. Then the two graphs are unioned together and the system allows the user to browse the dataset using a paradigm of a random walk with teleport as described in 3.4.3.

These approaches attempt to tackle the disadvantage of low-level features by either embedding various features, incorporating user feedbacks, or exploiting semantic information. However, they have not been able to address the challenges of capturing the semantic relations among a large number of concepts, for example, general topics at the Internet level. Nor are they able to provide good service to users when the intention of the query is unknown.

To address the ambiguity of the purpose of a query, in [182], the

authors proposed the “mental image search” method, where a user can locate target images by a series of relative feedback or a visual composition of the mental image. Instead of presenting the query result as a ranked list, the “mental image search” approach applies an interactive method to present the images in the database. A major limitation of this approach is that the system does not take advantage of the visual and conceptual relations of the images in the database. Furthermore, users have difficulty to navigate the presented images because they are lack of clear logic.

The next subsection will show a technique that can capture visual-focused relations among a large number of veracious concepts at the Internet level. When used in a CBIR system, this graph can greatly reduce the users’ efforts in finding the target images when the intention of the query is unspecified.

#### **4.2.1 Click-through data**

Annotating or tagging images of a large image dataset has always been a bottleneck for a general CBIR system. The tremendous amount of work and the inadequate quality of annotation prevent the industry from building a good general CBIR system for a big image dataset. However, search engines that provide image search services on the Internet based on the title or surrounding text have collected a huge amount of data that can be used as annotations and help improve the quality of a general CBIR system. Text-based image search engines let users enter a query text and then rank images in the dataset according to the title, image file name or surrounding text of the web page that the image is embedded. Thumbnails of the images are presented to the user ordered by their relevance. The user then clicks on the image that he/she is interested in to view the full-size image or get extra information about the image. When the user clicks an image in the

**Table 4.1** Summary of the training dataset of the Microsoft Clickture-lite Dataset

images	triads	query texts	avg. query texts per image	avg. images per query text	avg. clicks per image
1M	23M	11,7M	23.1	2.0	82.3

result list, he/she gives his consensus on the association of the query text and the image. The number of clicks received for each image and query text pair is recorded by the system. This type of data is named as click-through data [183]. Each item in the click-through data is a triad that contains the query text, the image, and the number of clicks. The click-through data used in this thesis is the Microsoft Clickture-lite Dataset [184, 185]. Table 4.1 shows the summary of the training dataset of this dataset.

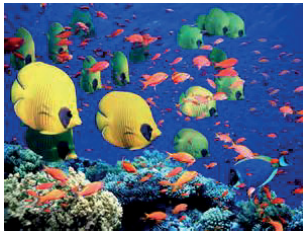
Fig. 4.2 shows some examples of the click-through data. Note that an image may be associated with multiple query texts and vice versa.

The click-through data have the following properties:

- The majority of click-through triads are correct associations and the query texts with a high number of clicks describe the most important aspect of the image. However, there are also cases that the associations between the image and the query text are incorrect.
- The raw query texts contain different forms of the text for the same concept, such as plural forms of nouns, different order of words, and synonyms. Many query texts also contain typos, which dramatically increase the number of query texts associated with an image.



bike (239)  
 picture of bikes (19)  
 pictures of bicycles (8)  
 bike riding (9)  
 kids on bike (1)  
 animated kids bicycles (1)  
 2 children (1)



great barrier reef (312)  
 the great barrier reef (73)  
 great barrier reef animals (18)  
 tropical fish (8)  
 galapagos fish (2)  
 underwater pictures (1)



mexican revolution (5)  
 mexico in 1900s (2)  
 mexican cartel (2)  
 guns of mexican revolution (1)  
 mexican military cap (1)



paul ryan bow (10)  
 paul ryan bowhunting (10)  
 paul ryan with bow (2)  
 paul ryan bow and arrow (2)  
 paul ryan archery (1)

(a)

(b)

**Figure 4.2** Examples of click-through data. (a) the image; (b) the associated query text and number of clicks (shown in brackets).

- One image can be described by multiple valid query texts. For example, the first image in Fig. 4.2 contains both children and bikes. Thus, the query texts related to “children”, “bike”, and the combination of the two are all valid descriptions.
- Some query texts contain general words that do not describe the content of an image, for example, irrelevant nouns such as “picture” and “photo”, prepositions, conjunction, and determiners.

### 4.2.2 Query text cleaning

Since the raw click-through data, in particular, the query texts, are noisy, we first apply text processing to clean up the input data. Text processing first converts each query text into a unique semantic ID—a set of word stems—in order to merge different forms of the same query into one entry. This procedure includes the following steps:

1. Remove the triads in which the query texts contain non-ASCII characters.
2. Split each query text into words and perform part-of-speech tagging [186]. After tagging, nouns, verbs, adjectives, and adverbs are kept. All other word types are discarded.
3. Lemmatize words using WordNet engine [187] so that a word is represented only by its stem.
4. Remove words that do not describe the content of an image. Our blacklist includes “image”, “picture”, “free”, “photo”, etc.

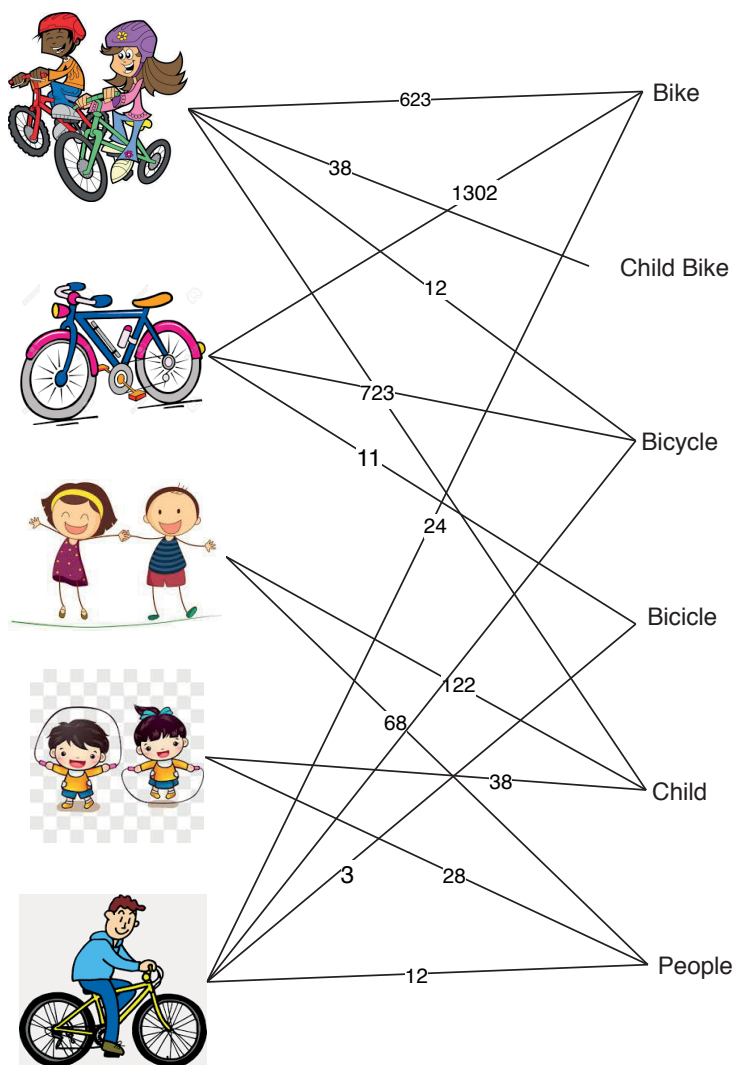
After the query text cleaning, the number of semantic IDs is significantly reduced compared to the number of query texts. For example, query texts such as “picture of bikes”, “the free pictures of bike”, “bike

picture”, and “image with bikes” are all converted into the same semantic ID “bike”. However, typos are not corrected during the text processing, because many contemporary words used on the Internet are deliberately misspelled to achieve a sense of humor or for obfuscation. For example “doge” and “cate” shall not be considered as typos for “dog” and “cat”.

### 4.2.3 Build the visual-semantic graph

After the query texts are converted into semantic IDs, the triads with the same image and the same semantic ID are merged. The number of clicks of the new triad is the sum of clicks of the merged triads. The merged triads can be represented as a bipartite graph (called image-semantic bipartite graph) where one type of node represents images, the other type of node represents semantic ID, and the weight of an edge that links an image node and a semantic ID node is the number of clicks. This image-semantic bipartite graph is similar to the one used in [181]. An induced subgraph of this graph is shown in Fig 4.3. Note that the graph contains duplicate concepts, such as “bike” and “bicycle”, and typos, such as “bicicle” and “bicycel”.

Because of the diversity of query texts, semantic IDs are still redundant in the image-semantic bipartite graph. For example, the semantic ID “obama”, “barrack obama”, “president barrack obama”, and typos of these IDs all refer to the same concept although with different IDs. We note that semantic IDs are associated with the same group of images if the meanings of the semantic IDs are identical. Meanwhile, images that are associated with the same group of semantic IDs are also conceptually close to each other. With this observation, we can construct a visual-semantic graph that contains nodes with explicit concepts to further reduce the redundancy in this bipartite graph.



*Figure 4.3* A bipartite graph that shows images, semantic IDs, and the number of clicks associated with the two elements.



A visual-semantic graph is a graph where each node is the collection of the images and the semantic IDs that are conceptually identical. For example, the semantic node “bike” contains the images of bikes and the semantic IDs such as “bike” and “bicycle”. Let  $P$  be the set of images and  $Q$  be the set of semantic IDs in an image-bipartite graph. Let  $p \in P$  be an image,  $q \in Q$  be a semantic ID, and  $c_{pq}$  be the number of clicks that is associated with  $p$  and  $q$ . Let  $s_i = (P_i, Q_i, c_i)$  be a node in the visual-semantic graph where  $P_i$  is the set of images belonging to  $s_i$ ,  $Q_i$  is the set of semantic IDs belonging to  $s_i$  and  $c_i$  is the total number of clicks that are associated with items in  $P_i$  and  $Q_i$ . We can merge image  $x$  to node  $s_i$  if image  $x$  is more associated with  $S_i$  than any other node. For example, merge  $x$  to  $s_i$  if  $\sum_{q \in Q_i} c_{xq} > \tau \sum_{q \in Q} c_{xq}$ , where  $\tau \in (0, 1)$  is the threshold that controls the level of association. If  $\tau$  is too small, we may merge loosely related items into one concept. For example, the images of Obama’s family may be merged into the node of Obama. If  $\tau$  is too big, images with the same concepts may end up with different nodes. In practice, we simply select  $\tau = 0.5$ , which guarantees the majority rule if the assignment is considered as a voting event. Similarly semantic ID  $y$  is merged to  $S_i$  if  $\sum_{p \in P_i} c_{py} > \tau \sum_{p \in P} c_{py}$ . A pivot-based algorithm is used to construct the nodes in the visual-semantic graph from the triads that the image-semantic graph represents. The algorithm first selects the triad with the largest number of clicks as the pivot node, and then merges images and semantic IDs to the pivot node alternatively. Algorithm 4 shows the details of this method.

After the nodes are constructed, we can build the visual-semantic graph by linking the two nodes with an edge whose weight represents the level of association of the two concepts. Let  $s_i = (P_i, Q_i, c_i)$  and  $s_j = (P_j, Q_j, c_j)$  be two nodes in the visual-semantic graph. The weight

**given** input triad set  $T$  with elements  $(p_t, q_t, c_t)$ , where  
 $t = 1, 2, \dots, |T|$ ,  $p_t \in P$  and  $q_t \in Q$   
**initiate** node set  $S = \emptyset$  and completed triad set  $U = \emptyset$   
**while**  $T \setminus U$  is not empty  
    select  $t^* = \operatorname{argmax}_t (c_t)$  from set  $T \setminus U$   
    let  $s_i = (P_i, Q_i, c_i)$  where  $P_i = \{p_{t^*}\}$ ,  $Q_i = \{q_{t^*}\}$  and  $c_i = c_{t^*}$   
    remove  $p_{t^*}$  from  $P$  and remove  $q_{t^*}$  from  $Q$   
    repeat until  $P_i$  and  $Q_i$  does not change anymore  
    for  $x \in P$   
        if  $\sum_{q \in Q_i} c_{xq} > \tau \sum_{q \in Q} c_{xq}$   
            add  $x$  to  $P_i$   
            remove  $x$  from  $P$   
            add the triads in  $T$  that contain  $(x, q)$  to  $U$  for  
all  $q \in Q_i$   
        for  $y \in Q$   
            if  $\sum_{p \in P_i} c_{py} > \tau \sum_{p \in P} c_{py}$   
                add  $y$  to  $Q_i$   
                remove  $y$  from  $Q$   
                add the triads in  $T$  that contain  $(p, y)$  to  $U$  for  
all  $p \in P_i$   
        add  $s_i$  to  $S$   
**return**  $S$

**Algorithm 4:** Construct nodes in a visual-simantic graph from input triads

**Table 4.2** Statistics of the visual-semantic graph generated from Microsoft Clickture dataset

nodes	edges	density	average degree	diameter	clustering coefficient
729K	5.32M	2.0E-5	14.6	14*	0.273**

\* estimated using 100 randomly selected nodes

\*\* estimated using 10k randomly selected nodes

of the edge that links node  $s_i$  and  $s_j$  is defined as

$$w_{ij} = \sum_{p \in P_i} \sum_{q \in Q_j} c_{pq} + \sum_{p \in P_j} \sum_{q \in Q_i} c_{pq}. \quad (4.2)$$

If  $w_{ij} = 0$ , no edge is added.

Note that there is a weight associated with each node in the visual-semantic graph. The weight of the node indicates the popularity of the concept. For example, the weight of node “dog” is much larger than the weight of node “1989 ford engine” since the first concept is more popular in the search history.

#### 4.2.4 Properties of the visual-semantic graph

In this section, we study some properties of the visual-semantic graph built from Microsoft Clickture-lite dataset using Algorithm 4. The generated graph is called Microsoft Clickture-lite Visual-Semantic (MCVS) graph. Table 4.2 shows some statistics of the MCVS graph.

The MCVS graph contains 729k nodes that are generated from 11.7M query texts. Each node represents a semantically exclusive concept and is associated with the images of that concept. The concepts are similar to the labels used in an image dataset for machine learning tasks. However, unlike most of the image datasets where the labels

are objects [188], labels in the MCVS graph show great variation, since they contain:

- general objects such as “dog”, “table”, and “car”.
- names of people and group of people such as “Albert Einstein”, “Justin Bieber”, and “One Direction”.
- names of places or entities such as “White House”, “Florida”, and “MIT”.
- names of events such as “911”, “the Vietnam war” and “American civil war”.
- a specific part of an object or a specific style of some object such as “Ford engine”, “Bob hairstyle”, “rose tattoo” and “dinosaur color page”.
- feelings or descriptions such as “funny”, “peaceful”, and “fast”.
- actions or subjects in action such as “running”, “horse riding”, and “man riding a bike”.

Because of the great variety of the labels, it is impossible to categorize the type of links between the labels as normally used in a semantic graph. However, since the visual-semantic graph is generated from an image dataset, links between the nodes are determined by the visual content of the nodes. For example, a strong link between “cat” and “dog” is due to the fact that they are often shown together in a picture. Similarly, a link between “fish” and “water” does not indicate “fish lives in the water”, but fish and water often appear together in a picture. Next, we can examine some subgraphs of the MCVS graph and see the relations between the nodes.

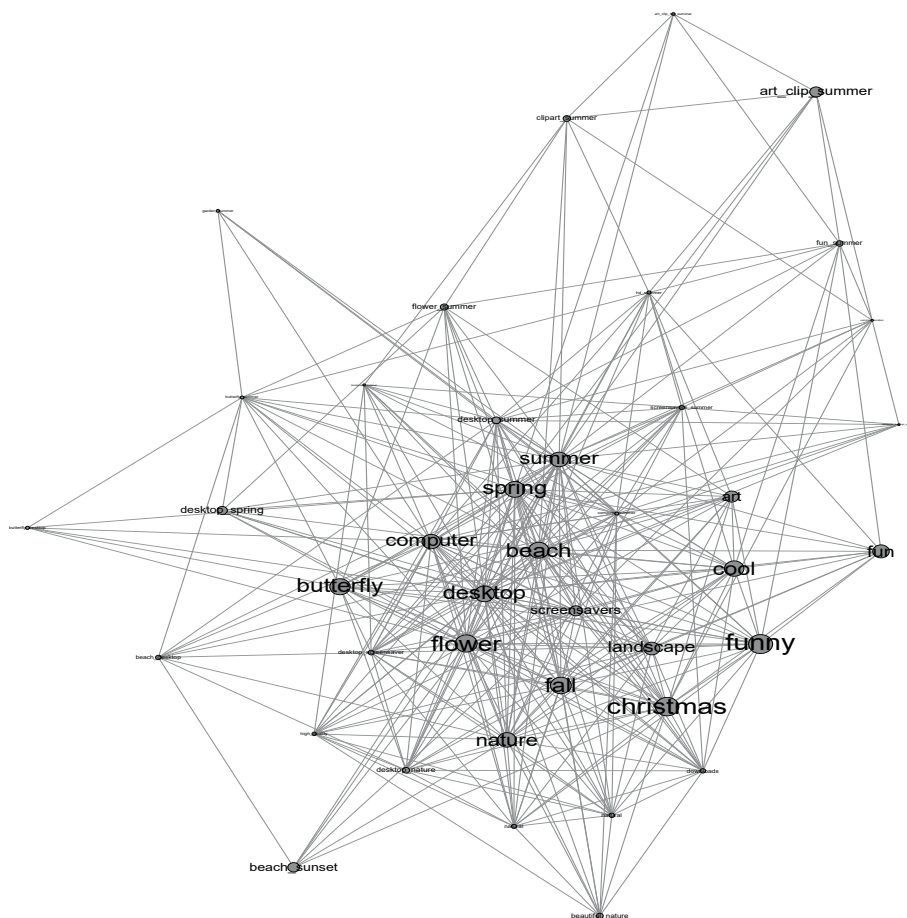
Figures 4.4, 4.5 and 4.6 shows three subgraphs of the MCVS graph. Each subgraph contains a seed node, 10 neighboring nodes of the seed node and 30 second-level neighboring nodes. The neighboring nodes are selected according to the weight of the edge that connects them to the seed node. The size of a node indicates the weight of the node. For clarity, edges with small weight are not shown in the graph.

### 4.3 Using visual-semantic graph in CBIR systems

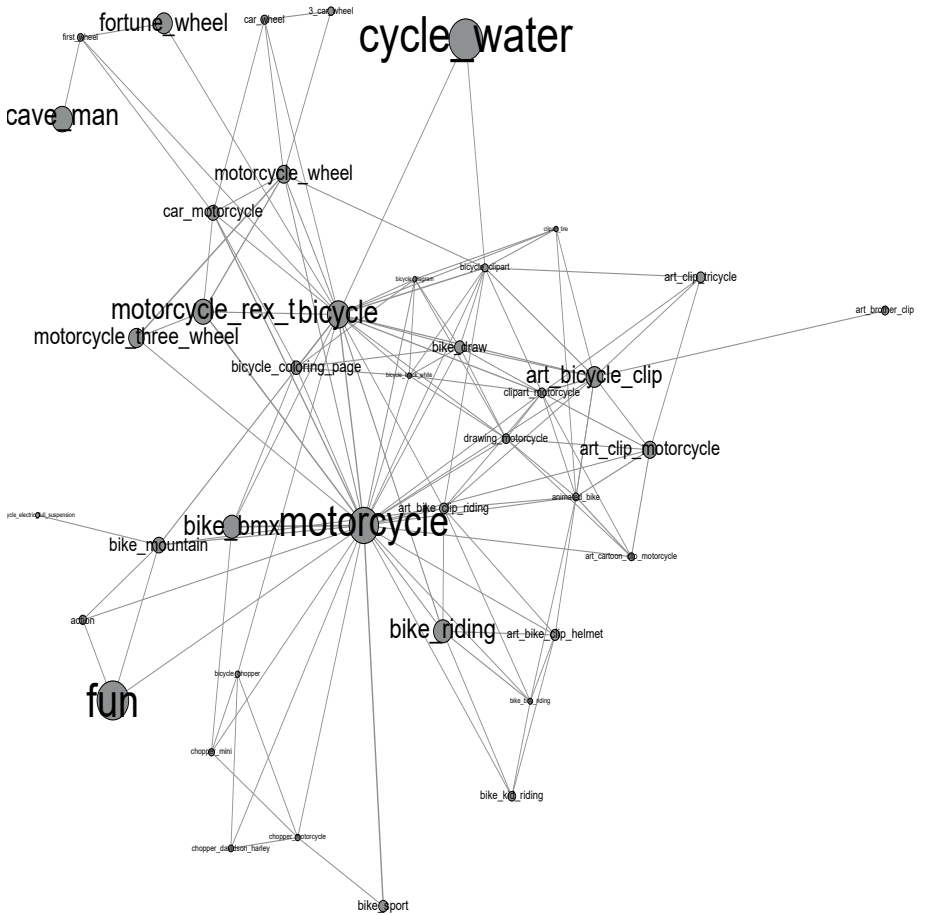
#### 4.3.1 Graph-enhanced CBIR system

As discussed in Section 4.1, the biggest challenge that a CBIR system faces is the difficulty to understand the intention of a query. Even if the system provides users options to choose the most relevant features for that query [176], it is challenging for users to match their intention to the most efficient feature set. In particular, if users are looking for the answer to a question related to the query image, showing visually similar images rarely provides information to answer the question. This section presents a graph-enhanced CBIR system (gCBIR system) that addresses these difficulties with the help of graph techniques using the visual-semantic graph.

A visual-semantic graph organizes a large variety of concepts into a graph structure and the links between nodes capture the visual affinity of different concepts. Since images are more convenient in describing complex concepts and are easier for users to understand, users are able to find their target images faster by navigating through visual relations. Given a query image, the proposed gCBIR system first ranks the nodes in the visual-semantic graph using a multi-label ranking algorithm. The ranked nodes are presented to the user as a list. When a user clicks the name of a node, the neighboring nodes of the selected one are shown to the user ordered by their relevance to the query



**Figure 4.4** The subgraph of seed node “summer”. In the subgraph, there are nodes that are naturally related to “summer” such as “beach”, “flower”, and “butterfly”. “spring” and “fall” are also in the subgraph. Note that “winter” is also linked with “summer” but the edge between the two nodes is not shown because its weight is too low. It is surprising to see that “Christmas” is in the graph. “Christmas” and “Summer” are linked through the node “funny” because of the humor effect (know as surreal humor) created by the illogical combination of “summer” and “Christmas”.



**Figure 4.5** The subgraph of seed node “bicycle”. This subgraph contains many clearly related items such as “motorcycle”, “bike\_riding” and “mountain\_bike”. The subgraph also shows that “bicycle” and “motorcycle” are related to “fun”. It is also interesting to see that the node “cave\_man” in the graph. “cave\_man” and “bicycle” are linked with node “first\_wheel”. In common sense, “cave\_man” invented the first wheel that is naturally related to bicycle.



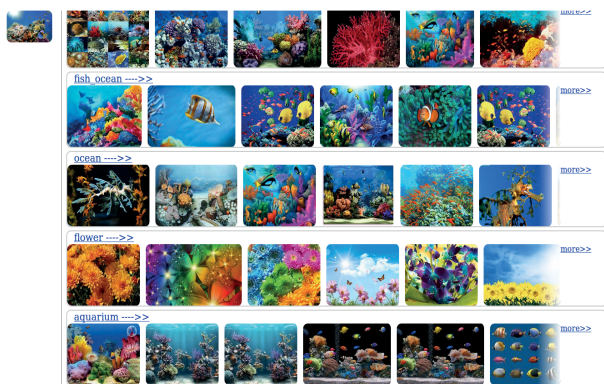
**Figure 4.6** The subgraph of seed node “rain”. In the subgraph, there are nodes that are naturally related to “rain” such as “cloud”, “rain\_drop”, “forest”, and “precipitation”. “dance” is linked to “rain” mainly because of the famous scene in the American movie “Singin’ in the rain” [189]. Similarly, node “purple” appears in the graph because of the famous album and film “purple rain” [190, 191]. It is also interesting to see that “kiss\_rain” links node “rain” and “love\_poem” together.



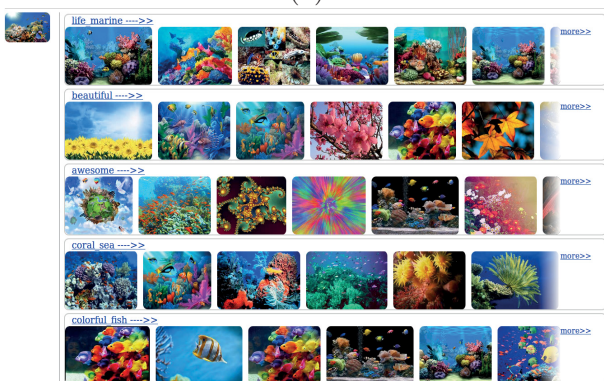
image. Users can also choose to view all the images in the node. Fig. 4.7 shows the screenshots of a typical use case.

After a query image is received, the gCBIR system first ranks the nodes in the visual-semantic graph according to their similarities to the query image. This is a typical multilabel ranking problem that has been studied for decades [P3, P5, 192, 193]. Different approaches have been proposed to solve this problem. Tsoumakas et al. categorized the algorithms into three groups: problem transformation methods, algorithm adaptation methods and ensemble methods [193]. The problem transformation methods take the binary classification methods as the basis and use either a one-against-all or one-against-one strategy to get the multilabel classification results. The algorithm adaptation methods modify the existing algorithms for binary classification problems to handle multiple labels. The ensemble algorithms apply a set of basic classifiers to the subsets of the samples and the labels. The results are aggregated using sum, voting or other appropriate rules [194]. However, when the number of labels is large, previous algorithms are either infeasible or perform poorly.

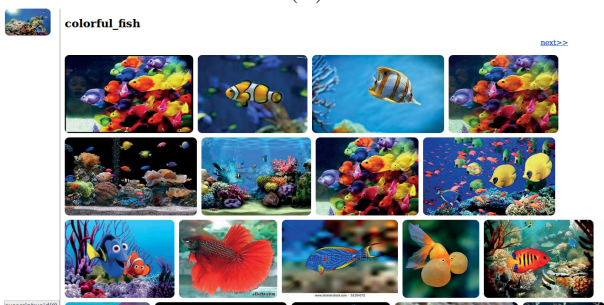
To effectively tackle the problem with a large number of labels, Zhang et al. proposed a multilabel ranking algorithm that is based on the k-nearest neighbor paradigm [P3]. The proposed algorithm treats labels as the properties of the samples and models the probability of having a certain property as an exponential function of the distance. Taking all the positive samples around a query sample into consideration, the probability of not having the property can be calculated using the product rule. The proposed method shows superior performance on the multilabel datasets generated from the Microsoft Clickture-lite Dataset compared to other instance-based algorithms such as MLkNN [195], BRkNN and BRkNN-w [196].



(a)



(b)



(c)

**Figure 4.7** A typical use case of the proposed *gCBIR* system using the visual-semantic graph. (a) Given a query image (shown in the top-left corner), the proposed *gCBIR* system shows the list of nodes ordered by their relevance to the query image. (b) After the user selected “aquarium”, the *gCBIR* system shows the neighboring nodes of node “aquarium” in the ordered of their relevances to the query image. The user can continue browsing the graph by selecting a next neighboring node. (c) The user selects “colorful\_fish” and the proposed *gCBIR* system shows images associated with node “colorful\_fish”.

After the nodes are ordered according to their affinity to the query image, the query result is presented to the user as described in Fig. 4.7. The gCBIR system only sorts the nodes once when the system receives the query image. To further improve the user experience, nodes that have already been presented in the previous screens will not be shown again when the user navigates through the visual-semantic graph.

### 4.3.2 Experimental results of the gCBIR system

Next, we evaluate the effectiveness of the gCBIR system using the MCVS graph by comparing its performance to traditional CBIR systems. We suppose a user tries to find the images of a certain concept by providing a query image. The gCBIR system ranks the nodes in the visual-semantic graph according to their relevance to the query image and shows the result to the user as a graph. The system performance can be evaluated by the effort (for example, the number of checked items) used to find the correct node. When the results are presented as a list, we can assume that the user takes the same amount of effort to check each item in the list. With this assumption, the mean inverse rank (MIR), also known as the mean reciprocal rank in statistics [197], can be used to evaluate the performance of a ranking algorithm [P3]. However, when the results are presented as a graph, it requires extra effort to navigate to the neighboring nodes of a node. Thus extra penalties should be applied when the user navigates the graph further. Suppose that the target node for the query image  $i$  is found with the navigation of the sequence of nodes  $n_1, n_2, \dots, n_{K_i}$  and the rank of each node is  $r_1, r_2, \dots, r_{K_i}$  in the list of neighboring nodes of the previous node. Let  $c$  be the penalty of navigating to a neighboring node. Let  $n$  be the number of query images in the test set. The MIR of navigating through a graph is defined as

$$MIR = \frac{1}{n} \sum_{i=1}^n \frac{1}{\sum_{j=1}^{K_i} r_j + (K_i - 1)c}. \quad (4.3)$$

The larger MIR score is, the less effort a user needs to find the correct node, thus the better a CBIR system performs.

As described in Section 4.2.1, the training dataset of the Microsoft Clickture-lite Dataset contains 1M images and 11M query texts. The development set contains 21893 pairs of image and query text. The MCVS graph generated using Algorithm 4 contains 729k nodes. We map each query text to a node in the MCVS graph and generate 21893 image and node pairs from the development set. A traditional CBIR system, where the query results are presented as a list, is used as the baseline to compare with the proposed gCBIR system. Let  $k$  be the maximum number of items in a list that can be checked. For the baseline system, if the correct node does not appear in the first  $k$  items, we mark the query as a failed item and the MIR score for this item is set to 0. For the gCBIR system, we assume that the user will only navigate to the neighboring nodes once ( $K_i = 2$  in Eq 4.3). Thus for the gCBIR system, the maximum number of items that can be checked is  $k^2$ . If the correct node does not appear in the  $k^2$  items, the query is marked as failed and the MIR score is set to 0. The MIR score for the baseline system is calculated with  $k = 100$ . The proposed gCBIR system was evaluated using both  $k = 100$  and  $k = 10$ . Note that when  $k = 10$ , the total number of nodes that can be checked is the same as the baseline. Table 4.3 shows the MIR scores and the number of failed queries of the baseline system and the proposed gCBIR system using different parameters.

Table 4.3 shows that the proposed gCBIR system clearly outperforms the baseline with regard to the MIR score. Note that when  $k = 10$ , the gCBIR system presents the same amount of items to the user as

**Table 4.3** *Experimental results of the proposed gCBIR system*

	baseline (k=100)	gCBIR (k=100, c=1)	gCBIR (k=10,c=1)	gCBIR (k=10, c=10)
MIR	0.081	0.102	0.094	0.082
failed queries	14719	9149	15895	15895

the baseline. The larger MIR score seen in this setup indicates that a user is able to find the target node more easily than the baseline. It is also noticed that the number of failed queries of the gCBIR system is larger than the baseline when  $k = 10$ . This is expected since when the user navigates to the neighboring nodes, the nodes are further from the query image. Actually, this behavior is advantageous if the user is looking for answers to a question related to the query image. Since he/she is able to quickly find more related items in a broader range.

#### 4.4 Summary

Content-based image retrieval is an important application of computer vision and data mining. As the proverb goes “a picture is worth a thousand words”. People are capable of capturing a great amount of information and details in a short time from an image than the description in text format. Retrieving valuable information from a large image or multimedia dataset using a query image is intuitive and efficient. However, because of the enormous variety of the intentions of a query, it is difficult for a CBIR system to select the most efficient feature set and ranking method to provide a satisfying retrieval result. This difficulty is coined as “semantic gap” and it has been one of the most critical research topics regarding the CBIR systems [52].

With the fast growth of the Internet and the advances in electronic devices, a large volume of multimedia content such as images, videos,

and audio clips are generated. Retrieving important information from these data is crucial and challenging. Previous efforts of annotating big image datasets using tags have been proved to be costly and unreliable since they often generate noisy annotations. More important, the content and the focus on the Internet is continuously changing and it is impossible to apply human resources to provide timely annotations.

People have been relying on search engines, mostly text-based, to retrieve information—including multimedia content—from the Internet. When a user uses a text-based image search engine, images are ranked according to the title or surrounding texts and shown as thumbnail images. From the given thumbnail images, the user can click on the image he/she is interested in to get either the full-size image or further information about that image. The number of clicks that an image receives with regard to the query text gives a reliable evaluation of the association between the two items. Microsoft Clickture-lite dataset is this type of dataset that contains a large number of triads of query text, image and the number of clicks that have been collected from a text-based image search engine [183]. From this kind of datasets, we can construct visual-semantic graphs that capture the visual relations between different semantic concepts. The weight of the link indicates the strength of the association from the visual perspective.

With the help of the visual-semantic graph, we can effectively address the difficulties that a CBIR system deals with. Instead of showing the retrieval results as a list of ranked images, we can show the graph of the ranked semantic concepts. The user is able to navigate the visual-semantic graph to quickly locate the target images. Furthermore, with the visual-semantic graph, the user can explore related information quickly and thus find answers to questions that are related to the query image. The experimental results show that the proposed gCBIR system can find the target concept in a large dataset more efficiently

compared to traditional CBIR systems.

The proposed gCBIR system can be applied to any large multimedia database that contains the conceptual and visual relations between the entities. However, the industry has not made such kind of large databases publically available other than the Microsoft Clickture-lite database. The proposed algorithm will be verified on other databases when they become available. In a real-world system, images and concepts are changing continuously. Constructing the visual-semantic graph and using it in a gCBIR system for a large dynamic multimedia database is an interesting topic for future research.

Unlike links in a semantic graph that indicate the semantic relations between concepts, links in a visual-semantic graph are closely related to the visual content of the concepts. They show more diversity than the relations in a semantic graph. By examining the visual-semantic graph, we notice some interesting links among the concepts. The use of visual-semantic graph shall not be limited to CBIR systems. It embeds important and interesting information about the relations of a large variety of concepts and is worthy of further attention.





## 5. CONCLUSIONS

Over the past hundreds of years, graph theory has attracted great attention from not only mathematicians but also researchers and engineers in many other fields. Any data that represent relations among identities can be modeled in a graph structure by nodes and edges. From the “Seven Bridges of Königsberg” problem [2] and the “Four-color theorem” [8] to recent advances in random graph models [10] and big graph analysis [7], graph theory has become an important branch in mathematics and a fundamental tool in computer science, which helps solve numerous scientific and engineering problems. Despite numerous topics in graph analysis, this thesis discussed a few topics that the author studied over the years when working on social networks and multimedia data.

Nodes and edges are the constructional elements of a graph. Various attributes have been defined to evaluate the importance of nodes or edges, or to determine the roles that nodes or edges play in a graph. Comparing to the study of attributes for nodes, there has been little research on attributes or properties of edges in a graph. In [P2, P4], we studied the clustering structure of social networks and proposed an authenticity score to measure the truthfulness of an edge. Edges with low authenticity scores are likely to be either outliers in a graph or links that connect nodes in different clusters in a graph. Numerous applications can benefit from the study of edge authenticity, such as outlier detection, graph clustering, data clustering, and graph data

preprocessing.

Cluster structure is a common phenomenon in social networks. Finding clusters in a big graph helps to understand the relations among nodes and extract useful information from the structure of a network. In [P1], we developed the Limited Random Walk (LRW) algorithm in which the scope of the walking agents is limited using inflation and normalization operators. Using the LRW procedure, we can extract features for each node in a big graph using an embarrassingly parallel implementation. After features are obtained, any suitable data clustering algorithm can be applied to find clusters in the graph.

The LRW algorithm and the research of edge authenticity score provide us new insights into a graph data structure and answer our first research question raised in Chapter 1.

This thesis also showed how graph techniques can be used to improve the user experience of content-based image retrieval (CBIR) systems. Given a query image, a CBIR system ranks images in a large image dataset according to their similarity to the query image and the retrieved images are presented to the user by this order. However, because of the ambiguity of the intention of a query and the limitation of the computer vision technologies, a CBIR system has to deal with the challenges, coined as “semantic gap”, that the retrieved images do not meet users’ expectations. The definition of “similarity” may be different with respect to the intentions of the query and it is always difficult to choose the most suitable feature set that is optimal to a specific definition of “similarity”. This thesis proposed a method to construct a visual-semantic graph—a graph where each node represents an independent semantic concept and each link represents the visual association between two concepts—from clicktiture datasets that contain triads of query text, image, and the number of clicks. Different

from normal semantic graphs where links represent the logical relations of different concepts, links in a visual-semantic graph captures the visual relations of different objects and concepts. The graph-enhanced CBIR system (gCBIR) presented in this thesis significantly improves the efficiency of retrieving target images from large image datasets and provides answers to the second research question raised in Chapter 1.

The studies about the visual semantic graph and the gCBIR system are far from complete. For future work, new algorithms should be developed to construct the visual-semantic graphs with a better abstraction of semantic concepts. More importantly, since the content on the Internet is continuously changing, fast algorithms are required to update the visual-semantic graph continuously. When new concepts appear, their links to the existing concepts shall be predicted. Furthermore, combining natural language processing and voice processing techniques with the gCBIR system can provide a solution with better efficiency. This is also an important direction for future research work.



## BIBLIOGRAPHY

- [P1] Honglei Zhang, Jenni Raitoharju, Serkan Kiranyaz, and Moncef Gabbouj. Limited random walk algorithm for big graph data clustering. *Journal of Big Data*, 3(1):26, 2016.
- [P2] Honglei Zhang, Serkan Kiranyaz, and Moncef Gabbouj. Outlier edge detection using random graph generation models and applications. *Journal of Big Data*, 4(1):11, April 2017.
- [P3] Honglei Zhang, Serkan Kiranyaz, and Moncef Gabbouj. A k-nearest neighbor multilabel ranking algorithm with application to content-based image retrieval. In *2017 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2017 - Proceedings*, pages 2587–2591. IEEE, 2017.
- [P4] Honglei Zhang, Serkan Kiranyaz, and M. Gabbouj. Data Clustering Based on Community Structure in Mutual k-Nearest Neighbor Graph. *International Conference on Telecommunications and Signal Processing (TSP)*, 2018.
- [P5] Honglei Zhang and Moncef Gabbouj. Feature Dimensionality Reduction with Graph Embedding and Generalized Hamming Distance. *IEEE International Conference on Image Processing (ICIP)*, 2018.
- [1] David S. Richeson. *Euler's Gem: The Polyhedron Formula and the Birth of Topology*. Princeton University Press, 2008.
- [2] The Euler Archive. Available at <http://eulerarchive.maa.org/>.
- [3] Brian Hopkins and Robin J. Wilson. The truth about Königsberg. *The College Mathematics Journal*, 35(3):198–207, 2004.

- [4] P. Erdős. Graph theory and probability. II. *CANAD. J. MATH*, 1960.
- [5] John Harris, Jeffrey L. Hirst, and Michael Mossinghoff. *Combinatorics and Graph Theory*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 2 edition, 2008.
- [6] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 1 edition edition.
- [7] Mark Newman. *Networks: An Introduction*. Oxford University Press, Oxford ; New York, 1 edition edition, May 2010.
- [8] Professor Cayley. On the Colouring of Maps. *Proceedings of the Royal Geographical Society and Monthly Record of Geography*, 1(4):259–261, 1879.
- [9] E.L. Lawler. *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley & sons, 1985.
- [10] Paul Erdős and Alfréd Rényi. On Random Graphs I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.
- [11] Jérôme Kunegis. KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion*, pages 1343–1350, 2013.
- [12] Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. Community detection in Social Media. *Data Mining and Knowledge Discovery*, 24(3):515–554, June 2011.
- [13] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In

- Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42. ACM, 2009.
- [14] Vito Latora and Massimo Marchiori. Is the Boston subway a small-world network? *Physica A: Statistical Mechanics and its Applications*, 314(1):109 – 113, 2002.
- [15] David Croft, Antonio Fabregat Mundo, Robin Haw, Marija Milacic, Joel Weiser, Guanming Wu, Michael Caudy, Phani Garapati, Marc Gillespie, Maulik R. Kamdar, Bijay Jassal, Steven Jupe, Lisa Matthews, Bruce May, Stanislav Palatnik, Karen Rothfels, Veronica Shamovsky, Heeyeon Song, Mark Williams, Ewan Birney, Henning Hermjakob, Lincoln Stein, and Peter D’Eustachio. The Reactome pathway knowledgebase. *Nucleic Acids Research*, page gkt1102, November 2013.
- [16] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006. arXiv: physics/0602124.
- [17] Austin R. Benson, David F. Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, July 2016.
- [18] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002.
- [19] David Hallac, Jure Leskovec, and Stephen Boyd. Network Lasso: Clustering and Optimization in Large Graphs. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’15, pages 387–396, New York, NY, USA, 2015. ACM.
- [20] R. Lambiotte, J. C. Delvenne, and M. Barahona. Random Walks, Markov Processes and the Multiscale Modular Organization of

- Complex Networks. *IEEE Transactions on Network Science and Engineering*, 1(2):76–90, July 2014.
- [21] Yu Xin, Zhi-Qiang Xie, and Jing Yang. The adaptive dynamic community detection algorithm based on the non-homogeneous random walking. *Physica A: Statistical Mechanics and its Applications*, 450:241–252, 2016.
- [22] Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.
- [23] Mason A. Porter, Jukka-Pekka Onnela, and Peter J. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.
- [24] R. Andersen, Fan Chung, and K. Lang. Local Graph Partitioning using PageRank Vectors. In *47th Annual IEEE Symposium on Foundations of Computer Science, 2006. FOCS '06*, pages 475–486, October 2006.
- [25] Aydin Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent Advances in Graph Partitioning. *CoRR*, abs/1311.3144, 2013.
- [26] Huaijun Qiu and Edwin R. Hancock. Graph matching and clustering using spectral partitions. *Pattern Recognition*, 39(1):22–34, 2006.
- [27] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, June 2010.
- [28] Zhen Hu and Raj Bhatnagar. Clustering algorithm based on mutual K-nearest neighbor relationships. *Statistical Analysis and Data Mining*, 5(2):100–113, April 2012.



- [29] Kohei Ozaki, Masashi Shimbo, Mamoru Komachi, and Yuji Matsumoto. Using the Mutual K-nearest Neighbor Graphs for Semi-supervised Classification of Natural Language Data. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, pages 154–162, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.
- [30] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, November 1993.
- [31] Wei Zhang, Xiaogang Wang, Deli Zhao, and Xiaoou Tang. Graph Degree Linkage: Agglomerative Clustering on a Directed Graph. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, number 7572 in Lecture Notes in Computer Science, pages 428–441. Springer Berlin Heidelberg, 2012.
- [32] Shuicheng Yan, Dong Xu, Benyu Zhang, and Hong-Jiang Zhang. Graph embedding: a general framework for dimensionality reduction. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 830–837 vol. 2, June 2005.
- [33] Bin Cheng, Jianchao Yang, Shuicheng Yan, Yun Fu, and Thomas S. Huang. Learning with l1-graph for image analysis. *Trans. Img. Proc.*, 19(4):858–866, April 2010.
- [34] Haiping Lu, Konstantinos N. Plataniotis, and Anastasios N. Venetsanopoulos. A survey of multilinear subspace learning for tensor data. *Pattern Recognition*, 44(7):1540 – 1551, 2011.
- [35] A. Sharma, A. Kumar, H. Daume, and D. W. Jacobs. Generalized Multiview Analysis: A discriminative latent space. In *2012 IEEE*

- Conference on Computer Vision and Pattern Recognition*, pages 2160–2167, June 2012.
- [36] Jure Leskovec and Julian J. McAuley. Learning to Discover Social Circles in Ego Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 539–547. Curran Associates, Inc., 2012.
- [37] Jure Leskovec and Andrej Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection*. June 2014.
- [38] Stanley Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.
- [39] Shishir Bharathi, David Kempe, and Mahyar Salek. Competitive Influence Maximization in Social Networks. In Xiaotie Deng and Fan Chung Graham, editors, *Internet and Network Economics*, pages 306–311, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [40] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the Spread of Influence Through a Social Network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.
- [41] J. O. Kephart and S. R. White. Directed-graph epidemiological models of computer viruses. In *Proceedings. 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 343–359, May 1991.
- [42] B. Aditya Prakash, Hanghang Tong, Nicholas Valler, Michalis Faloutsos, and Christos Faloutsos. *Virus Propagation on Time-Varying Networks: Theory and Immunization Algorithms*.

- [43] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999.
- [44] Tie-Yan Liu. Learning to Rank for Information Retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.
- [45] Matthew Richardson, Amit Prakash, and Eric Brill. Beyond PageRank: machine learning for static ranking. In *Proceedings of the 15th international conference on World Wide Web*, pages 707–715. ACM, 2006.
- [46] Abraham Kandel, Horst Bunke, and Mark Last, editors. *Applied Graph Theory in Computer Vision and Pattern Recognition*. Studies in Computational Intelligence. Springer-Verlag, Berlin Heidelberg, 2007.
- [47] L. Wiskott, N. Krüger, N. Kuiger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, July 1997.
- [48] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001.
- [49] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime Multi-Person 2d Pose Estimation using Part Affinity Fields. *CoRR*, abs/1611.08050, 2016.
- [50] Dong Zhang and Mubarak Shah. A Framework for Human Pose Estimation in Videos. *CoRR*, abs/1604.07788, 2016.
- [51] S. Kiranyaz, K. Caglar, E. Guldogan, O. Guldogan, and M. Gabbouj. MUVIS: a content-based multimedia indexing and retrieval

- framework. In *Seventh International Symposium on Signal Processing and Its Applications, 2003. Proceedings.*, volume 1, pages 1–8 vol.1, July 2003.
- [52] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.
- [53] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based Multimedia Information Retrieval: State of the Art and Challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, February 2006.
- [54] Deng Cai, Xiaofei He, Zhiwei Li, Wei-Ying Ma, and Ji-Rong Wen. Hierarchical Clustering of WWW Image Search Results Using Visual, Textual and Link Information. In *Proceedings of the 12th Annual ACM International Conference on Multimedia, MULTIMEDIA '04*, pages 952–959, New York, NY, USA, 2004. ACM.
- [55] Rada Mihalcea. Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions, ACLdemo '04*, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.
- [56] B. Xu, J. Bu, C. Chen, C. Wang, D. Cai, and X. He. EMR: A Scalable Graph-Based Ranking Model for Content-Based Image Retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):102–114, January 2015.
- [57] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [58] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

- [59] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Advanced Texts in Econometrics. Clarendon Press, 1995.
- [60] S. Muroga, I. Toda, and M. Kondo. Majority Decision Functions of up to Six Variables. *Mathematics of Computation*, 16(80):459–472, 1962.
- [61] Y. Rao and X. Zhang. Characterization of Linearly Separable Boolean Functions: A Graph-Theoretic Perspective. *IEEE Transactions on Neural Networks and Learning Systems*, 28(7):1542–1549, July 2017.
- [62] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR*, abs/1609.02907, 2016.
- [63] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning Convolutional Neural Networks for Graphs. *CoRR*, abs/1605.05273, 2016.
- [64] Honglei Zhang, Serkan Kiranyaz, and Moncef Gabbouj. Finding Better Topologies for Deep Convolutional Neural Networks by Evolution. *ArXiv e-prints*, September 2018.
- [65] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.
- [66] Mark Newman. *Networks: An Introduction*. Oxford University Press, Oxford ; New York, 1 edition edition, May 2010.
- [67] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Experiments on Graph Clustering Algorithms. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003*, number 2832 in Lecture Notes in Computer Science, pages 568–579. Springer Berlin Heidelberg, January 2003.

- [68] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.
- [69] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. Villas Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, 2007.
- [70] N. Perra and S. Fortunato. Spectral centrality measures in complex networks. *pre*, 78(3):036107, September 2008.
- [71] Yuhua Qian, Yebin Li, Min Zhang, Guoshuai Ma, and Furong Lu. Quantifying edge significance on maintaining global connectivity. *Scientific Reports*, 7:45380 EP –, 2017.
- [72] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Angela Ricciardello. A Novel Measure of Edge Centrality in Social Networks. *CoRR*, abs/1303.1747, 2013.
- [73] Béla Bollobás. *Random Graphs*. Cambridge University Press, Cambridge ; New York, 2 edition edition, October 2001.
- [74] Mark EJ Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary physics*, 46(5):323–351, 2005.
- [75] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, January 2002.
- [76] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: the topology of the worldwide web. *Physica A: Statistical Mechanics and its Applications*, 281(1):69–77, 2000.
- [77] Réka Albert and Albert-László Barabási. Topology of Evolving Networks: Local Events and Universality. *Phys. Rev. Lett.*, 85(24):5234–5237, December 2000.

- [78] P. L. Krapivsky, G. J. Rodgers, and S. Redner. Degree Distributions of Growing Networks. *Physical Review Letters*, 86:5401–5404, June 2001.
- [79] D. J. Watts and S. H. Strogatz. Collective dynamics of ‘small-world’ networks. *nat*, 393:440–442, June 1998.
- [80] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509, October 1999.
- [81] A. Fronczak, P. Fronczak, and J. A. Holyst. Average path length in random networks. *eprint arXiv:cond-mat/0212230*, December 2002.
- [82] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.
- [83] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.
- [84] Mark EJ Newman. Random graphs with clustering. *Physical Review Letters*, 103(5), July 2009. arXiv: 0903.4009.
- [85] Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 25–32. ACM, 2001.
- [86] J. Y. Zien, M. D. F. Schlag, and P. K. Chan. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(9):1389–1399, September 1999.

- [87] Konstantin Andreev and Harald Räcke. Balanced Graph Partitioning. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '04, pages 120–124, New York, NY, USA, 2004. ACM.
- [88] H. Meyerhenke, P. Sanders, and C. Schulz. Parallel Graph Partitioning for Complex Networks. *IEEE Transactions on Parallel and Distributed Systems*, 28(9):2625–2638, September 2017.
- [89] Kai Yu, Shipeng Yu, and Volker Tresp. Soft Clustering on Graphs. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, pages 1553–1560, Cambridge, MA, USA, 2005. MIT Press.
- [90] Fan Chung and Mark Kempton. A Local Clustering Algorithm for Connection Graphs. In *Algorithms and Models for the Web Graph*, pages 26–43. Springer, 2013.
- [91] Peter Macko, Daniel Margo, and Margo Seltzer. Local clustering in provenance graphs. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 835–840. ACM, 2013.
- [92] Satu Elisa Schaeffer. Stochastic Local Clustering for Massive Graphs. In Tu Bao Ho, David Cheung, and Huan Liu, editors, *Advances in Knowledge Discovery and Data Mining*, number 3518 in Lecture Notes in Computer Science, pages 354–360. Springer Berlin Heidelberg, January 2005.
- [93] Julian Shun, Farbod Roosta-Khorasani, Kimon Fountoulakis, and Michael W. Mahoney. Parallel Local Graph Clustering. *Proc. VLDB Endow.*, 9(12):1041–1052, August 2016.
- [94] Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *arXiv preprint arXiv:0809.3232*, 2008.



- [95] Z. Yang, J. I. Perotti, and C. J. Tessone. Hierarchical benchmark graphs for testing community detection algorithms. *ArXiv e-prints*, August 2017.
- [96] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4), October 2008. arXiv: 0805.4770.
- [97] Sylvain Brohée and Jacques van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7(1):488, November 2006.
- [98] Jaewon Yang and Jure Leskovec. Defining and Evaluating Network Communities based on Ground-truth. *arXiv:1205.6233 [physics]*, May 2012. arXiv: 1205.6233.
- [99] Peter Kareiva. Small Worlds: The Dynamics of Networks between Order and Randomness. Duncan J. Watts. *The Quarterly Review of Biology*, 76(1):65–65, March 2001.
- [100] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985.
- [101] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. *J. Mach. Learn. Res.*, 11:2837–2854, December 2010.
- [102] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.
- [103] L.N.F. Ana and A.K. Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings*, volume 2, pages II–128–II–133 vol.2, June 2003.

- [104] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.
- [105] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3–5):75–174, February 2010.
- [106] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an Algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS’01, pages 849–856, Cambridge, MA, USA, 2001. MIT Press.
- [107] A. Pothen, H. Simon, and K. Liou. Partitioning Sparse Matrices with Eigenvectors of Graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, July 1990.
- [108] Kevin Lang. Fixing two weaknesses of the Spectral Method. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 715–722. MIT Press, 2006.
- [109] Boaz Nadler, Stéphane Lafon, Ronald R. Coifman, and Ioannis G. Kevrekidis. Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS’05, pages 955–962, Cambridge, MA, USA, 2005. MIT Press.
- [110] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, September 1992.
- [111] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

- [112] Nate Veldt, David F. Gleich, and Anthony Wirth. A Correlation Clustering Framework for Community Detection. In *Proceedings of the 2018 World Wide Web Conference, WWW '18*, pages 439–448, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.
- [113] Robert Görke, Andrea Kappes, and Dorothea Wagner. Experiments on Density-Constrained Graph Clustering. *J. Exp. Algorithmics*, 19:3.3:1.1–3.3:1.31, January 2015.
- [114] L. Waltman and N. J. Eck. A smart local moving algorithm for large-scale modularity-based community detection. *Eur Phys J B*, 86, 2013.
- [115] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.
- [116] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation Clustering. *Machine Learning*, 56(1-3):89–113, July 2004.
- [117] Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavl'tsev. Near Optimal LP Rounding Algorithm for Correlation Clustering on Complete and Complete k-partite Graphs. *arXiv:1412.0681 [cs]*, December 2014. arXiv: 1412.0681.
- [118] R. Guimera, M. Sales-Pardo, and L. A. N. Amaral. Module identification in bipartite and directed networks. *Physical Review E*, 76(3), September 2007. arXiv: physics/0701151.
- [119] Claire P. Massen and Jonathan P. K. Doye. Identifying "communities" within energy landscapes. *Physical Review E*, 71(4), April 2005. arXiv: cond-mat/0412469.

- [120] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical review E*, 72(2):027104, 2005.
- [121] Sune Lehmann and Lars Kai Hansen. Deterministic Modularity Optimization. *The European Physical Journal B*, 60(1):83–88, November 2007. arXiv: physics/0701348.
- [122] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning Deep Representations for Graph Clustering. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1293–1299, Qu&#233;bec City, Qu&#233;bec, Canada, 2014. AAAI Press.
- [123] Y. Chen, S. Sanghavi, and H. Xu. Improved Graph Clustering. *IEEE Transactions on Information Theory*, 60(10):6440–6455, October 2014.
- [124] F. Y. Wu. The Potts model. *Reviews of Modern Physics*, 54(1):235–268, January 1982.
- [125] Sacha Friedli and Yvan Velenik. *Statistical Mechanics of Lattice Systems: A Concrete Mathematical Introduction*. Cambridge University Press, 1 edition, November 2017.
- [126] I. Ispolatov, I. Mazo, and A. Yuryev. Finding mesoscopic communities in sparse networks. *Journal of Statistical Mechanics (Online)*, 9:p09014, September 2006.
- [127] Jörg Reichardt and Stefan Bornholdt. Detecting Fuzzy Community Structures in Complex Networks with a Potts Model. *Physical Review Letters*, 93(21):218701, November 2004.
- [128] P. Pons and M. Latapy. Computing communities in large networks using random walks (long version). *ArXiv Physics e-prints*, December 2005.

- [129] Haijun Zhou and Reinhard Lipowsky. Network Brownian Motion: A New Method to Measure Vertex-Vertex Proximity and to Identify Communities and Subcommunities. In Marian Bubak, Geert Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, *Computational Science - ICCS 2004*, pages 1062–1069, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [130] David Harel and Yehuda Koren. On Clustering Using Random Walks. In Ramesh Hariharan, V. Vinay, and Madhavan Mukund, editors, *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science*, number 2245 in Lecture Notes in Computer Science, pages 18–41. Springer Berlin Heidelberg, 2001.
- [131] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, Providence, R.I, 1 edition edition, December 2008.
- [132] J.R. Norris. *Markov Chains | Applied probability and stochastic networks*, 1998.
- [133] Reid Andersen, Fan Chung, and Kevin Lang. Using pagerank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007.
- [134] B. Cai, H. Wang, H. Zheng, and H. Wang. An improved random walk based clustering algorithm for community detection in complex networks. In *2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2162–2167, October 2011.
- [135] Kathy Macropol, Tolga Can, and Ambuj K. Singh. RRW: repeated random walks on genome-scale protein networks for local cluster discovery. *BMC bioinformatics*, 10(1):283, 2009.
- [136] Zeyuan A. Zhu, Silvio Lattanzi, and Vahab Mirrokni. A local algorithm for finding well-connected clusters. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 396–404, 2013.

- [137] Stijn Dongen. *Graph clustering by flow simulation*. PhD thesis, Universiteit Utrecht, Utrecht, The Netherlands, May 2000.
- [138] A. Knyazev. Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method. *SIAM Journal on Scientific Computing*, 23(2):517–541, January 2001.
- [139] Gary William Flake, Robert E. Tarjan, and Kostas Tsioutsoulis. Graph Clustering and Minimum Cut Trees. *Internet Mathematics*, 1(4):385–408, 2003.
- [140] R. Gomory and T. Hu. Multi-Terminal Network Flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, December 1961.
- [141] Wei Zhang, Deli Zhao, and Xiaogang Wang. Agglomerative clustering via maximum incremental path integral. *Pattern Recognition*, 46(11):3056–3065, November 2013.
- [142] Hagen Kleinert. *Path integrals in quantum mechanics, statistics, polymer physics, and financial markets*. World scientific, 2009.
- [143] Elchanan Mossel, Joe Neeman, and Allan Sly. Stochastic Block Models and Reconstruction. *arXiv:1202.1499 [math-ph]*, February 2012. arXiv: 1202.1499.
- [144] William Feller. *An Introduction to Probability Theory and Its Applications, Vol. 1, 3rd Edition*. Wiley, 3rd edition edition, 1968.
- [145] Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM (JACM)*, 42(1):67–90, 1995.

- [146] Wei Dong, Charikar Moses, and Kai Li. Efficient K-nearest Neighbor Graph Construction for Generic Similarity Measures. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 577–586, New York, NY, USA, 2011. ACM.
- [147] M. Connor and P. Kumar. Fast construction of k-nearest neighbor graphs for point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):599–608, July 2010.
- [148] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.
- [149] Dongkuan Xu and Yingjie Tian. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2(2):165–193, June 2015.
- [150] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.
- [151] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.
- [152] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153. ACM, 2006.
- [153] Robert Görke, Tanja Hartmann, and Dorothea Wagner. Dynamic Graph Clustering Using Minimum-Cut Trees. In Frank Dehne, Marina Gavrilova, Jörg-Rüdiger Sack, and Csaba D. Tóth, editors, *Algorithms and Data Structures*, Lecture Notes in Computer Science, pages 339–350. Springer Berlin Heidelberg, 2009.

- [154] Rob Fergus, Yair Weiss, and Antonio Torralba. Semi-supervised learning in gigantic image collections. In *Advances in neural information processing systems*, pages 522–530, 2009.
- [155] Mark J. Huiskes, Bart Thomee, and Michael S. Lew. New trends and ideas in visual concept detection: the MIR flickr retrieval evaluation initiative. In *Proceedings of the international conference on Multimedia information retrieval*, pages 527–536. ACM, 2010.
- [156] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. LabelMe: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.
- [157] N. W. U. D. Chathurani, S. Geva, V. Chandran, and V. Cynthujah. An effective Content Based Image Retrieval system based on global representation and multi-level searching. In *2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS)*, pages 158–163, December 2015.
- [158] John P. Eakins and Margaret E. Graham. *Content-based image retrieval, a report to the JISC Technology Applications programme*. 1999.
- [159] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, January 2007.
- [160] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *arXiv:1605.07146 [cs]*, May 2016. arXiv: 1605.07146.
- [161] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. *arXiv:1603.05027 [cs]*, March 2016. arXiv: 1603.05027.



- [162] A. Blanton, K. C. Allen, T. Miller, N. D. Kalka, and A. K. Jain. A Comparison of Human and Automated Face Verification Accuracy on Unconstrained Image Sets. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 229–236, June 2016.
- [163] P. Jonathon Phillips, Amy N. Yates, Ying Hu, Carina A. Hahn, Eilidh Noyes, Kelsey Jackson, Jacqueline G. Cavazos, Géraldine Jeckeln, Rajeev Ranjan, Swami Sankaranarayanan, Jun-Cheng Chen, Carlos D. Castillo, Rama Chellappa, David White, and Alice J. O’Toole. Face recognition accuracy of forensic examiners, superrecognizers, and face recognition algorithms. *Proceedings of the National Academy of Sciences*, page 201721355, May 2018.
- [164] Xiaohui SHEN, Zhe Lin, Shu Kong, and Radomir Mech. Utilizing deep learning for rating aesthetics of digital images, October 2017.
- [165] Z. Wang, D. Liu, S. Chang, F. Dolcos, D. Beck, and T. Huang. Image aesthetics assessment using Deep Chatterjee’s machine. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 941–948, May 2017.
- [166] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.
- [167] Lin Ma, Zhengdong Lu, and Hang Li. Learning to Answer Questions from Image Using Convolutional Neural Network. In *AAAI*, volume 3, page 16, 2016.
- [168] Thomas M. Deserno, Sameer Antani, and Rodney Long. Ontology of Gaps in Content-Based Image Retrieval. *Journal of Digital*

- Imaging: the official journal of the Society for Computer Applications in Radiology*, 22(2):202–215, April 2009.
- [169] Malay Kumar Kundu, Manish Chowdhury, and Samuel Rota Bulò. A graph-based relevance feedback mechanism in content-based image retrieval. *Knowledge-Based Systems*, 73:254–264, January 2015.
- [170] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.
- [171] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.
- [172] Ehab Salahat and Murad Qasaimeh. Recent advances in features extraction and description algorithms: A comprehensive survey. In *Industrial Technology (ICIT), 2017 IEEE International Conference on*, pages 1059–1063. IEEE, 2017.
- [173] Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 931–938, 2014.
- [174] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.
- [175] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297.

- [176] Savvas A. Chatzichristofis, Konstantinos Zagoris, Yiannis S. Boutalis, and Nikos Papamarkos. Accurate image retrieval based on compact composite descriptors and relevance feedback information. *International Journal of Pattern Recognition and Artificial Intelligence*, 24(02):207–244, 2010.
- [177] Prof Dr Kai-Uwe Barthel; Jonas Hartmann; Nico Hezel; Mike Krause; Anja Sonnenberg. akiwi - a keywording tool.
- [178] Ilaria Bartolini. Content Meets Semantics: Smarter Exploration of Image Collections Presentation of Relevant Use Cases. 2012.
- [179] Samuel Rota Bulò, Massimo Rabbi, and Marcello Pelillo. Content-based image retrieval with relevance feedback using random walks. *Pattern Recognition*, 44(9):2109–2122, September 2011.
- [180] C. Deng, R. Ji, D. Tao, X. Gao, and X. Li. Weakly Supervised Multi-Graph Learning for Robust Image Reranking. *IEEE Transactions on Multimedia*, 16(3):785–795, April 2014.
- [181] Barbara Poblete, Benjamin Bustos, Marcelo Mendoza, and Juan Manuel Barrios. Visual-semantic Graphs: Using Queries to Reduce the Semantic Gap in Web Image Retrieval. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10*, pages 1553–1556, New York, NY, USA, 2010. ACM.
- [182] Simon P. Wilson, Julien Fauqueur, and Nozha Boujemaa. Mental Search in Image Databases: Implicit Versus Explicit Content Query. In Matthieu Cord and Pádraig Cunningham, editors, *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*, Cognitive Technologies, pages 189–204. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

- [183] Yingwei Pan, Ting Yao, Tao Mei, Houqiang Li, Chong-Wah Ngo, and Yong Rui. Click-through-based Cross-view Learning for Image Search. 2014.
- [184] Clickture. Available at <https://www.microsoft.com/en-us/research/project/clickture/>.
- [185] Xian-Sheng Hua, Linjun Yang, Jingdong Wang, Jing Wang, Ming Ye, Kuansan Wang, Yong Rui, and Jin Li. Clickage: Towards Bridging Semantic and Intent Gaps via Mining Click Logs of Search Engines. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 243–252, New York, NY, USA, 2013. ACM.
- [186] Eugene Charniak. Statistical Techniques for Natural Language Parsing. *AI Magazine*, 18(4):33–33, December 1997.
- [187] George A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38:39–41, 1995.
- [188] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [189] AFI|Catalog - Singin' in the Rain. Available at <https://catalog.afi.com/Catalog/moviedetails/50652>.
- [190] Albert Magnoli. Purple Rain, August 1984. Available at <http://www.imdb.com/title/tt0087957/>.
- [191] Prince And The Revolution - Purple Rain. Available at <https://www.discogs.com/Prince-And-The-Revolution-Purple-Rain/release/194021>.
- [192] Eva Gibaja and Sebastián Ventura. A Tutorial on Multilabel Learning. *ACM Comput. Surv.*, 47(3):52:1–52:38, April 2015.

- [193] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.
- [194] Jesse Read, Luca Martino, and David Luengo. Efficient monte carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition*, 47(3):1535–1546, March 2014.
- [195] Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, July 2007.
- [196] Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, Ioannis Yiannis Kompatsiaris, Grigorios Tsoumakas, and Ioannis Vlahavas. A comprehensive study over vlad and product quantization in large-scale image retrieval. *IEEE Transactions on Multimedia*, 16(6):1713–1728, 2014.
- [197] Dragomir R. Radev, Hong Qi, Harris Wu, and Weiguo Fan. Evaluating Web-based Question Answering Systems. In *LREC*, 2002.



## PUBLICATIONS





# PUBLICATION

I

**Limited random walk algorithm for big graph data clustering**

H. Zhang, J. Raitoharju, S. Kiranyaz and M. Gabbouj

*Journal of Big Data* 3.1 (2016), 26

DOI: 10.1186/s40537-016-0060-5

**Publication reprinted with the permission of the copyright holders**




RESEARCH

Open Access



# Limited random walk algorithm for big graph data clustering

Honglei Zhang<sup>1\*</sup> , Jenni Raitoharju<sup>1</sup>, Serkan Kiranyaz<sup>2</sup> and Moncef Gabbouj<sup>1</sup>

\*Correspondence:

honglei.zhang@tut.fi

<sup>1</sup> Department of Signal Processing, Tampere University of Technology, Korkeakoulunkatu 1, 33101 Tampere, Finland  
Full list of author information is available at the end of the article

## Abstract

Graph clustering is an important technique to understand the relationships between the vertices in a big graph. In this paper, we propose a novel random-walk-based graph clustering method. The proposed method restricts the reach of the walking agent using an inflation function and a normalization function. We analyze the behavior of the limited random walk procedure and propose a novel algorithm for both global and local graph clustering problems. Previous random-walk-based algorithms depend on the chosen fitness function to find the clusters around a seed vertex. The proposed algorithm tackles the problem in an entirely different manner. We use the limited random walk procedure to find attractor vertices in a graph and use them as features to cluster the vertices. According to the experimental results on the simulated graph data and the real-world big graph data, the proposed method is superior to the state-of-the-art methods in solving graph clustering problems. Since the proposed method uses the embarrassingly parallel paradigm, it can be efficiently implemented and embedded in any parallel computing environment such as a MapReduce framework. Given enough computing resources, we are capable of clustering graphs with millions of vertices and hundreds millions of edges in a reasonable time.

**Keywords:** Graph clustering, Random walk, Big data, Community finding

## Background

Graph data are important data types in many scientific areas, such as social network analysis, bioinformatics, and computer and information network analysis [1]. In recent years, the size of graph data has grown dramatically. For example, a typical social network graph may contains millions of vertices and hundreds of million of edges. Further more, these graphs may continuously evolve over time. Processing these dynamic big graph data is very challenging and time consuming. In general, big graphs are normally heterogeneous. They have such non-uniform structures that edges between vertices in a group are much denser than edges connecting vertices in different groups. Graph clustering (also named as “community detection” in the literature) algorithms aim to reveal the heterogeneity and find the underlying relations between vertices [2]. This technique is critical for understanding the properties, predicting dynamic behavior and improving visualization of big graph data.

Graph clustering is a computationally challenging and difficult task, especially for big graph data. Many algorithms have been proposed over the last decades [2–4]. The

criteria-based approaches try to optimize clustering fitness functions using different optimization techniques. Newman defined a modularity measurement based on the probability of the link between any two vertices. He applied a greedy search method to minimize this modularity fitness function in order to partition a graph into clusters [5]. Blondel et al. used the same fitness function but combined it with other optimization techniques [6–8]. Spielman and Teng opted the graph conductance measurement as the fitness function [9]. Other than criteria-based methods, spectral analysis has also been widely adapted in this area [10, 11]. Random-walk-based methods tackle the problem from a different angle [12–14]. These methods use the Markov chain model to analyze the graph. Each vertex represents a state and the edges indicate transitions between the states. The probability values that are distributed among the states (vertices) reveal the graph structure.

For big graph data, the problem becomes more challenging or even intractable. Very often, people are only interested in finding the cluster for a given seed vertex. This problem is called local clustering problem [9, 15, 16]. For example, from an end user's perspective, finding the closely connected friends around him or her is more important than revealing the global user clusters of a large social network. It is unnecessary to explore the whole graph structure for this problem. Recently, random walk methods have gained great attention on this local graph clustering problem, since a walk started from the seed vertex is more likely to stay in the cluster where the seed vertex belongs. Comparing to the criteria-based methods, the random-walk-based methods are capable of extracting local information from a big graph without the knowledge of the whole graph data. In [17–19], a random walk is first applied to find important vertices around the seed vertex. Then a sweep stage is involved to select the vertices that minimize the conductance of the candidate clusters.

The accuracy of any criteria-based clustering method (or those combined with the random walk procedures) is greatly affected by the chosen clustering fitness function. Furthermore, most local clustering algorithms use the criteria that are more suitable for the global graph clustering problem. These choices greatly degrade the performance of these algorithms when the graph is big and highly uneven. Also the majority of the graph clustering algorithms are designed in sequential computing paradigm. Therefore, they do not take advantage of modern high-performance computing systems.

In this paper, we propose a novel random-walk-based graph clustering algorithm—the limited random walk (LRW) algorithm. First of all, the LRW algorithm does not rely on any clustering fitness function. Furthermore, the proposed method can efficiently tackle the computational complexity using a parallel programming paradigm. Finally, as a unique property among many graph clustering methods, LRW can be adapted to both global and local graph clustering in an efficient way.

The rest of the paper is organized as follows: basics of random walk procedure and the proposed LRW algorithm are explained in "Methodology" section; an extensive set of experiments on the simulated and real graph data, along with both numerical and visual evaluations are given in "Experiments" section; finally, the conclusions and future work are discussed in "Conclusions" section.

## Methodology

### Basic definitions and the random walk procedure

Let  $G(V, E)$  denote a graph of  $n$  vertices and  $m$  edges, where  $V = \{v_i | i = 1, \dots, n\}$  is the set of vertices and  $E = \{e_i | i = 1, \dots, m\}$  is the set of edges. Let  $A \in \mathcal{R}^{n \times n}$  be the adjacency matrix of the graph  $G$  and  $A_{ij}$  are the elements in the matrix  $A$ . Let  $D \in \mathcal{R}^{n \times n}$  be the degree matrix, which is a diagonal matrix whose elements on the diagonal are the degrees of each vertex. In this paper, we assume the graph is undirected, unweighted and does not contain self-loops.

Clustering phenomenon is very common in big graph data. A cluster in a graph is a vertex set where the density of the edges inside the cluster is much higher than the density of edges that link the inside vertices and the outside vertices.

Random walk on a graph is a simple stochastic procedure. At the initial state, an agent stays on a chosen vertex (seed vertex). At each step, the agent randomly picks a neighboring vertex and moves to it. The agent repeats this movement and there is certain probability that the agent lands on a vertex after each movement.

Let  $x_i^{(t)}$  denote the probability that the agent is on vertex  $v_i$  after step  $t$ , where  $i = 1, 2, \dots, n$ .  $x_i^{(0)}$  is the probability of the initial state. Let  $s$  be the seed vertex. We have  $x_s^{(0)} = 1$ , and  $x_i^{(0)} = 0$  for  $i \neq s$ . Let  $x^{(t)} = [x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)}]^T$  be the probability vector, where the superscript  $T$  denotes the transpose of a matrix or a vector. By the definition of the probability, it is easy to see that  $\sum_{i=1}^n x_i^{(t)} = 1$  or  $\|x^{(t)}\|_1 = 1$ .

The random walk procedure is equivalent to a discrete-time stationary Markov chain process. Each vertex is corresponding to a state in the Markov chain and each edge indicates a possible transition between the two states. The Markov transition matrix  $P$  can be obtained by normalizing the adjacency matrix to have each column sum up to 1, e.g.

$$P_{ij} = \frac{A_{ij}}{\sum_{k=1}^n A_{kj}} \quad (1)$$

or

$$P = AD^{-1}. \quad (2)$$

Other forms of the transition matrix  $P$  can also be used, for example the lazy random walk uses transition matrix  $P = \frac{1}{2}(I + AD^{-1})$ , where  $I$  is the identity matrix. Given the transition matrix  $P$ , we can calculate  $x^{(t+1)}$  from  $x^{(t)}$  using the equation:

$$x^{(t+1)} = Px^{(t)}. \quad (3)$$

A closed walk is a walk on a graph where the ending vertex is same as the seed vertex. The period of a vertex is defined as the greatest common divisor of the lengths of all closed walks that start from this vertex. We say a graph is aperiodic if all of its vertices have periods of 1.

For an undirected, connected and aperiodic graph, there exists an equilibrium state  $\pi$ , such that  $\pi = P\pi$ . This state is unique and irrelevant to the starting point. By iterating Eq. 3,  $x^{(t)}$  converges to  $\pi$ . More details about the Markov chain process and the equilibrium state can be found from [20].

**Limited random walk procedure**

**Definitions**

We first define the transition matrix  $P$ . We assign the same probability to the transition that the walking agent stays in the current vertex and the transition that it moves to any neighboring vertex. We add an identity matrix to the adjacency matrix and then normalize the result to have each column sum to 1. The transition matrix can be written as

$$P = (I + A)(I + D)^{-1}. \tag{4}$$

Comparing to the transition matrix in Eq. 2, this is similar to adding self-loops to the graph, but increasing the degree of each vertex by 1 instead of 2. This modification fixes the periodicity problem that the graph may have [20]. It greatly improves the stability and accuracy of the algorithm in graph clustering.

At each walking step, the probability vector  $x^{(t)}$  is computed using Eq. 3. Note that, in general, elements in  $x^{(t)}$  that are around the seed vertex are non-zeros and the rest are zeros. So we do not need the full transition matrix to calculate the probability vector for the next step.

Starting from the seed vertex, a normal random walk procedure will eventually explore the whole graph. To reveal a local graph structure, different techniques can be used to limit the scope of the walks. Harel and Koren fix the number of walking steps by a predefined constant [21]. Xin et al. use a stochastic method to determine if a walk should be continued and set the maximum number of walking steps to be 6 according to the principle of “six degrees of separation” [14]. In [13, 17, 22], the random walk function is defined as

$$x^{(t+1)} = \alpha x^{(0)} + (1 - \alpha)Px^{(t)}, \tag{5}$$

where  $\alpha$  is called the teleport probability. The idea is that there is a certain probability that the walking agent will teleport back to the seed vertex and continue walking.

Inspired by the Markov clustering algorithm (MCL) algorithm [12], we adapt the inflation and normalization operation after each step of the transition. The inflation operation is an element-wise super-linear function—a function that grows faster than a linear function. Here we use the power function

$$f(x) = [x_1^r, x_2^r, \dots, x_n^r]^T, \tag{6}$$

where the exponent  $r > 1$ . Since  $x$  indicates the probability that the agent hits each vertex,  $x$  must be normalized to have a sum of 1 after the inflation operation. The normalization function is defined as

$$g(x) = \frac{x}{\|x\|_1}, \tag{7}$$

where  $\|x\|_1 = \sum_{i=1}^n |x_i|$  is the  $L_1$  norm of the vector  $x$ . Since  $x_i \geq 0$  and  $\sum_{i=1}^n x_i = 1$ , Eq. 7 can also be written in a vector form as

$$g(x) = \frac{x}{x^T \cdot \mathbf{1}}, \tag{8}$$

where  $\mathbf{1} = [1, 1, \dots, 1]^T$ . The inflation and normalization operation enhance large values and depress small values in the vector  $x$ .

We call the aforementioned procedure the limited random walk (LRW) procedure. Comparing to the normal random walk procedure defined in "Basic definitions and the random walk procedure" section, LRW involves inflation and normalization operations in each walking step. These nonlinear operations limit the agent to walk around the neighborhood of the seed vertex, especially if there is a clear graph cluster boundary.

The MCL algorithm simulates flow within a graph. It uses the inflation and normalization operation to enhance the flow within a cluster and reduce the flow between clusters. The MCL procedure is a time-inhomogeneous Markov Chain in which the transition matrix varies over time. The MCL algorithm starts the random walk from all vertices simultaneously—there are  $n$  agents walking on the graph at the same time. The walking can only continue after all agents have completed a walking step and the result probability matrix has been inflated and normalized. Unlike in the MCL algorithm, the LRW procedure is a time-homogeneous Markov Chain. We initiate random walk from a single seed vertex, and do the inflation on the probability values of this walking agent. This design has many advantages. First, it avoids unnecessary walks since the graph structure around the seed vertex may be exposed by a single walk. Second, the procedure is suitable for the local clustering problems because it does not require the whole graph data. Third, if multiple walks are required, each walk procedure can be executed independently. Thus the algorithm is fully parallelizable.

The LRW procedure involves a nonlinear operation, thus it is difficult to analyze its properties on a general graph model. Next we study the equilibrium of the LRW procedure.

#### **Equilibrium of the LRW procedure**

We first prove the existence of equilibrium of the LRW procedure. Let  $X$  be the set of values of the probability vector  $x$ . We have

$$X = \{(x_1, x_2, \dots, x_n) \mid 0 \leq x_1, x_2, \dots, x_n \leq 1 \text{ and } x_1 + x_2 + \dots + x_n = 1\}. \quad (9)$$

The LRW procedure defined by Eqs. 3, 6 and 7 is a function that maps  $X$  to itself. Let  $\mathcal{L} : X \rightarrow X$ , such that

$$\mathcal{L}(x) = g(f(Px)). \quad (10)$$

**Theorem 1** *There exists a fixed-point  $x^*$  such that  $\mathcal{L}(x^*) = x^*$ .*

*Proof* We use the Brouwer fixed-point theorem to prove this statement.

Given  $0 \leq x_1, x_2, \dots, x_n \leq 1$ , the set  $X$  is clearly bounded and closed. Thus  $X$  is a compact set.

Let  $u, v \in X$  and  $w = \lambda u + (1 - \lambda)v$ , where  $\lambda \in \mathcal{R}$  and  $0 \leq \lambda \leq 1$ . So  $w_i = \lambda u_i + (1 - \lambda)v_i$  for  $i = 1, 2, \dots, n$ . Obviously  $0 \leq w_i \leq 1$ .

Further,

$$\begin{aligned} \sum_{i=1}^n w_i &= \sum_{i=1}^n (\lambda u_i + (1 - \lambda)v_i) \\ &= \lambda \sum_{i=1}^n u_i + (1 - \lambda) \sum_{i=1}^n v_i \\ &= 1 \end{aligned}$$

Thus  $w \in X$ . This indicates that the set  $X$  is convex.

Since function  $f(x)$  is continuous over the set  $X$  and function  $g(x)$  is continuous over the codomain of function  $f(x)$ , function  $\mathcal{L}$  is continuous over the set  $X$ .

Given  $\mathcal{L}$  is a continuous function that maps a convex set to itself, according to the Brouwer fixed-point theorem, there is a point  $x^*$  such that  $\mathcal{L}(x^*) = x^*$ .  $\square$

Theorem 1 shows the existence of fixed-point of the LRW procedure, i.e., the LRW procedure will not escape from a fixed-point whenever the point is reached. Since the LRW procedure is a non-linear discrete dynamic system, it is difficult to analytically investigate the system behavior. However, when  $r = 1$ , the LRW procedure is simply a Markov chain process, in which the fixed-point  $x^*$  is the unique equilibrium state  $\pi$  and the global attractor. In another extreme case when  $r \rightarrow \infty$ , a fixed-point can be an unstable equilibrium and the LRW procedure may have limit cycles that oscillate around a star structure in the graph. In one state of the oscillation, the probability value of the center of a star structure is close to one. In practice, we chose  $r$  from (1, 2]. This makes the LRW procedure close to a linear system and oscillations are extremely rare. In this case, the fixed-points of the LRW procedure are stable equilibriums.

#### **Limited random walk on general graphs**

Without any prior knowledge of the cluster formation, we normally start the LRW procedure from an initial state where  $x_s = 1$ ,  $x_i = 0$  for  $i \neq s$  and  $s$  is the seed vertex. During the LRW procedure, there are two simultaneous processes—the spreading process and the contracting process. When the two processes can balance each other, a stationary state is reached.

During the spreading process, the probability values spread as the walking agent visits new vertices. The number of visited vertices increases exponentially at first. The growth rate depends on the average degree of the graph. The newly visited vertices will always receive the smallest probability values. If the graph has an average degree of  $d$ , it is not difficult to see that the expected probability value of a newly visited vertex at step  $t$  is  $(1/d)^t$ . As the walking continues, the probability values tend to be distributed more evenly among all visited vertices.

The other ongoing process during the LRW procedure is the contracting process. During this process, the probability values of the visited vertices contract to some vertices. Since the graph is usually heterogeneous, some vertices (and groups of vertices) will receive higher probability values as the procedure continues. The inflation operation further enhances this contracting effect. The degrees of a vertex and its surrounding vertices determine whether the probability values concentrate to or diffuse from these vertices. Some vertices, normally the center of a star structure, receive larger probability values than others. We call these vertices attractor vertices and they can be used to represent the structure of a graph.

Because the density of edges inside a cluster is higher than that of linking the vertices inside and outside the cluster, the probability that a walking agent visits vertices outside the cluster is small. Thus, the LRW procedure will find attractor vertices that the seed vertex is associated. We can use these vertices as features to cluster the vertices.

The larger the inflation exponent  $r$  is, the faster the algorithm converges to the attractor vertices. The LRW procedure tries to find the attractor vertices that are near the seed vertex. However, if  $r$  is too large, the probability values concentrate to the nearest attractor



vertex (or the seed vertex itself) before the graph is sufficiently explored. If  $r$  is too small, the probability values will concentrate to the attractor vertices that may belong to other clusters. The performance of the LRW algorithm depends on choosing a proper inflation exponent  $r$ . From this aspect, it is similar to the MCL algorithm. In practice,  $r$  is normally chosen between 1 and 2 and the value 2 was found to be suitable for most graphs.

### LRW for global graph clustering problems

In this section, we propose how to LRW in global graph clustering problems. Our algorithm is divided into two phases—graph exploring phase and cluster merging phase. To improve the performance on big graph data, we also propose a multi-stage strategy.

#### Graph exploring phase

In the graph exploring phase, the LRW procedure is started from several seed vertices. At each iteration, the agent moves one step as defined in Eq. 3. Then the probability vector  $x$  is inflated by Eq. 6 and normalized by Eq. 7. The iteration stops when the probability vector  $x$  converges or the predefined maximum number of iterations is reached. Let  $x^{(*,i)}$  denote the final probability vector of a random walk that was started from the seed vertex  $v_i$ . As described in the previous section, the LRW procedure explores the vertices that are close to the seed vertex. Thus, the vector  $x^{(*,i)}$  has non-zero elements only on these neighboring vertices.

Algorithm 1 illustrates the graph exploring from a seed vertex set  $Q$ . Note that for small graph data, we can set the seed vertex set  $Q = V$  (i.e. the whole graph). In such case, the LRW procedure is executed on every vertex of the graph and the multi-stage strategy is not used.

---

#### Algorithm 1 Limited random walk graph exploring from a vertex set

---

given adjacency matrix  $A$  of the graph  $G(V, E)$ , the seed vertex set  $Q$ , the exponent  $r$  of the inflation function 6, maximum number of iterations  $T_{max}$ , a small value  $\epsilon$  to limit the number of visited vertices, and a small value  $\xi$  for the termination condition

initialize the transition matrix  $P$  by Eq. 4

for each vertex  $i$  in the vertex set  $Q$

- 1 initialize  $x^{(0,i)}$  such that  $x_k^{(0,i)} = \begin{cases} 1 & k = i \\ 0 & \text{otherwise} \end{cases}$  and  $t = 1$
  - 2 repeat if  $t < T_{max}$ 
    - (a)  $x^{(t,i)} = Px^{(t-1,i)}$
    - (b) if  $x_k^{(t,i)} < \epsilon$ , set  $x_k^{(t,i)}$  to 0
    - (c)  $x_k^{(t,i)} = f(x_k^{(t,i)}) = x_k^{(t,i)^r}$
    - (d)  $x_k^{(t,i)} = \frac{x_k^{(t,i)}}{\|x^{(t,i)}\|_1}$
    - (e) if  $\|x^{(t,i)} - x^{(t-1,i)}\|_2 < \xi$ , exit the loop
    - (f)  $t \leftarrow t + 1$
  - 3 output  $x^{(t,i)}$  as the attractor vector  $x^{(*,i)}$  for vertex  $i$
- 

Note that the threshold  $\epsilon$  limits the number of nonzero elements in the probability vector  $x$ . It is easy to prove that the number of nonzero elements in  $x^{(t,i)}$  is less than  $1/\epsilon$ . A larger  $\epsilon$  eliminates very small values in  $x^{(t,i)}$  and prevent unnecessary computing efforts. However,  $\epsilon$  does not impose a limit on the largest cluster we can find. Further, the choice of  $\epsilon$  has little impact on the final clustering results because either the LRW procedure finds the most dominant attractor vertices in a cluster or the small clusters are merged in the cluster merging phase.

### Cluster merging phase

After the graph has been explored, we will find the clusters in the cluster merging phase. We treat each  $x^{(*,i)}$  as the attractor vector for the vertex  $v_i$ . Vertices belonging to the same cluster have attractor vectors that are close to each other. Any unsupervised clustering algorithm, such as k-means or single linkage clustering method, can be applied to find the desired number of ( $k$ ) clusters. Because of the computational complexity of these clustering algorithms, we design a fast merging algorithm that can efficiently cluster vertices according to their attractor vectors.

Each element  $x_j^{(*,i)}$  in  $x^{(*,i)}$  is the probability value of the stationary state that the walking agent hits the vertex  $v_j$  when the seed vertex is  $v_i$ . The attractor vector  $x^{(*,i)}$  is determined by the graph structure of the cluster that the initial vertex  $v_i$  has. Thus, vertices in the same cluster should have very similar attractor vectors. We first find the vertex that has the largest value in the vector  $x^{(*,i)}$ . Suppose  $m = \arg \max_j (x_j^{(*,i)})$ , we call  $v_m$  the attractor vertex of vertex  $v_i$ . Grouping vertices by their attractor vertex can be done in a fast way (complexity of  $O(1)$ ) using a dictionary data structure. After the grouping, each vertex is assigned to a cluster that is identified by the attractor vertex. However, it is possible that some vertices in one cluster do not have the same attractor vertex. This may happen when the cluster is large and the edge density in the cluster is low. We then apply the following cluster merging algorithm to handle this overclustering problem.

The vertices that have large values, which are determined by a threshold relative to  $x_m^{(*,i)}$ , in  $x^{(*,i)}$  are called significant vertices for vertex  $v_i$ . If two vertices have large enough overlaps of their significant vertices, they should be grouped into the same cluster. From this observation, we first collect significant vertices for the found clusters. Then we merge clusters if their significant vertices overlap more than a half. Note that the attractor vertex and the significant vertices are always in the same cluster as the seed vertex. This is very useful when we use the multi-stage graph strategy.

Algorithm 2 shows the details of the merging phase of the LRW algorithm. Note that, for small graph data, we set the seed vertex set  $Q = V$  and the initial clustering dictionary  $\mathcal{D}$  to be empty.

---

#### Algorithm 2 LRW cluster merging phase

---

given feature vectors  $x^{(*,i)}$  of the seed vertex set  $Q$ , where  $i \in Q$ , threshold  $\tau$  such that  $0 < \tau < 1$ , and initial clustering dictionary  $\mathcal{D}$

for each feature vector  $x^{(*,i)}$  where  $i \in Q$

- 1 find  $m = \arg \max_j (x_j^{(*,i)})$  and add  $i$  to an empty vertex set  $S$
- 2 find all  $j$  such that  $x_j^{(*,i)} > \tau \cdot x_m^{(*,i)}$  and add them to an empty vertex set  $\mathcal{F}$
- 3 if  $\mathcal{D}$  does not contain the key  $m$ 
  - add the pair  $\langle S, \mathcal{F} \rangle$  as the value associated with the key  $m$  to dictionary  $\mathcal{D}$
  - else
    - find the value  $\langle S_m, \mathcal{F}_m \rangle$  that is associated with the key  $m$
    - add  $i$  to set  $S_m$  and merge  $\mathcal{F}$  to  $\mathcal{F}_m$  by  $\mathcal{F}_m = \mathcal{F}_m \cup \mathcal{F}$
    - update the value  $\langle S_m, \mathcal{F}_m \rangle$  of the key  $m$  in the dictionary  $\mathcal{D}$ .

end

for each pair of keys  $m_1$  and  $m_2$  in the dictionary  $\mathcal{D}$

- 1 get the value pairs of  $\langle S_{m_1}, \mathcal{F}_{m_1} \rangle$  and  $\langle S_{m_2}, \mathcal{F}_{m_2} \rangle$  that are associated with the key  $m_1$  and  $m_2$
- 2 get the union of the two vertex sets  $\mathcal{U} = \mathcal{F}_{m_1} \cap \mathcal{F}_{m_2}$
- 3 if  $|\mathcal{U}| > \frac{1}{2} \min(|\mathcal{F}_{m_1}|, |\mathcal{F}_{m_2}|)$ 
  - (a) merge  $S_{m_2}$  to  $S_{m_1}$ ,  $S_{m_1} = S_{m_1} \cup S_{m_2}$
  - (b) merge  $\mathcal{F}_{m_2}$  to  $\mathcal{F}_{m_1}$ ,  $\mathcal{F}_{m_1} = \mathcal{F}_{m_1} \cup \mathcal{F}_{m_2}$
  - (c) update  $\langle S_{m_1}, \mathcal{F}_{m_1} \rangle$  to the key  $m_1$
  - (d) delete the key  $m_2$  and its associated value

for each key  $m$  in  $\mathcal{D}$ , output  $S_m$  as the clustering result

---

### Multi-stage strategy

For small graph data, we can do the LRW procedure on every vertex of the graph. So the seed vertex set  $Q = V$ . The graph clustering is completed after a graph exploring phase and a cluster merging phase. However, when the graph data is large, it is time-consuming to perform the LRW procedure from every vertex of the graph. A multi-stage strategy can be used to greatly reduce the number of required walkings. First, we start the LRW procedure from a randomly selected vertex set. After the first round of the graph exploring, some clusters can be found after the cluster merging phase. Next we generate a new seed vertex set by randomly selecting vertices from those vertices that have not been clustered. Then we do the graph exploration from the new seed vertex set. We repeat this procedure until all vertices are clustered.

Algorithm 3 shows the global graph clustering algorithm using the multi-stage strategy.

---

#### Algorithm 3 Global graph clustering algorithm using multi-stage strategy

---

given adjacency matrix  $A$  of the graph  $G(V, E)$ , the exponent  $r$  for the inflation function, threshold value  $\tau$ , maximum number of iterations  $T_{max}$  and a small enough value  $\epsilon$   
 initialize the transition matrix  $P$  by Eq. 4, vertex set  $B = V$  and clustering dictionary  $\mathcal{D} = \emptyset$   
 while  $B$  is not empty  
   1 generate a vertex set  $Q \subset B$  by randomly select vertices from  $B$   
   2 do the graph exploring from the vertex set  $Q$  using algorithm 1 to get feature vectors  $x^{(*,i)}$  for  $i \in Q$   
   3 given  $x^{(*,i)}$  and  $\mathcal{D}$ , do the graph cluster merging using algorithm 2 to update the clustering dictionary  $\mathcal{D}$   
   4 for each key  $m$  in  $\mathcal{D}$ , merge the attractor vertex set  $\mathcal{F}_m$  to the cluster vertex set  $S_m$ ,  $S_m = S_m \cup \mathcal{F}_m$   
   5 get clustered vertex set  $R = \bigcup_{m \in \mathcal{D}} S_m$   
   6 let  $B = B \setminus R$   
 for each key  $m$  in  $\mathcal{D}$ , output  $S_m$  as the clustering result

---

### LRW for local graph clustering problems

For the local graph clustering problems, the LRW procedure can efficiently find the cluster from a given seed vertex. To achieve this, we first perform graph exploring from the seed vertex in the same way as described in "Graph exploring phase" section. Let  $x^{(*)}$  be the probability vector after the graph exploration. If a probability value in  $x^{(*)}$  is large enough, the corresponding vertex is assigned to the local cluster without further computation. Similar to the global graph clustering algorithm, we use a relative threshold  $\eta$  that is related to the maximum value in  $x^{(*)}$ . Vertices whose probability values are greater than  $\eta \cdot \max(x_j^{(*)})$  are called significant vertices. The significant vertices are assigned to the local cluster directly. A small value of  $\eta$  will reduce the computational complexity, but may decrease the accuracy of the algorithm. Suitable values of  $\eta$  were experimentally found to be between 0.3 and 0.5.

The vertices with low probability values can either be outside the cluster or inside the cluster but with relatively low significance. Unlike [9, 15, 16], which involve a sweep operation and a cluster fitness function, we do another round of graph exploring from these insignificant vertices. After the second graph exploring is completed, we apply the cluster merging algorithm described in "Cluster merging phase" section.

Algorithm 4 presents the LRW local clustering algorithm.

**Algorithm 4** LRW local graph clustering algorithm

---

given graph  $G(V, E)$ , seed vertex  $v$ , threshold values  $\eta$  and  $\tau$ , where  $0 < \eta < 1$  and  $0 < \tau < 1$ , the exponent  $r$  of the inflation function, and maximum number of iterations  $T_{max}$

- 1 do graph exploration starting from the seed vertex  $v$  as described in algorithm 1 and get the attractor vector  $x^{(*,v)}$
- 2 find vertices in  $x^{(*,v)}$  for which  $x_i^{(*,v)} > 0$  and collect the vertices in to set  $S$
- 3 find  $m = \arg \max_j (x_j^{(*,v)})$
- 4 split the set  $S$  into two sets  $S_1$  and  $S_2$  such that  $S_1 = \{j | x_j^{(*,v)} \geq \eta x_m^{(*,v)}\}$  and  $S_2 = \{j | x_j^{(*,v)} < \eta x_m^{(*,v)}\}$
- 5 for each vertex in  $S_2$  do graph exploration to get feature vectors  $x^{(*,i)}$ , where  $i \in S_2$
- 6 do clustering merging according to the feature vectors of the vertex  $v$  and the vertices in  $S_2$  as described in algorithm 2 and find the cluster  $S_3$  that contains the seed vertex  $v$
- 7 return  $S_1 \cup S_3$

---

**Computational complexity**

We first analyze the computational complexity of the LRW algorithm for the global graph clustering problem. We assume the graph  $G(V, E)$  has clusters. Let  $\bar{n}_c$  be the average cluster size—the number of vertices in the cluster, and  $C$  is the number of clusters. We have  $\bar{n}_c \cdot C = n$ . Note  $C \ll n$ . The most time-consuming part of the algorithm is the graph exploring phase. For each vertex, every iteration involves a multiplication of the transition matrix  $P$  and the probability vector  $x$ . The LRW procedure visits not only the vertices in the cluster but also a certain amount of vertices close to the cluster. Let  $\gamma$  be the coefficient that indicates how far the LRW procedure explores the graph before it converges. Notice the maximum number of nonzero elements in a probability vector is  $1/\epsilon$ . Let  $J$  denote the number of vertices that the LRW procedure visits in each iteration, thus  $J = \min(\gamma \bar{n}_c, 1/\epsilon)$ . Thus the transition step at each iteration has complexity of  $O(J\bar{n}_c)$ . The inflation and normalization steps, which operate on the probability vector  $x$ , have the complexity of  $O(J)$ . Let  $K$  be the number of iterations for the LRW procedure to converge. So, the computational complexity for a complete LRW procedure on each vertex is  $O(KJ\bar{n}_c)$ . For a global clustering problem when performing the LRW procedure on every vertex, the graph exploration phase has a complexity of  $O(KJ\bar{n}_c n)$ . In the worst case, the algorithm has a complexity of  $O(n^3)$ . This is an extremely rare case and it only happens when the graph is small; does not have a cluster structure; and the edge density is high. This worst case scenario is identical to the MCL algorithm [12]. Notice that the variables  $J$  and  $K$  have upper bounds and  $\bar{n}_c$  is determined by the graph structure, the algorithm has a complexity of  $O(n)$  for big graph data.

The computational complexity of the cluster merging phase involves merging clusters that were found using the attractor vertices. This merging requires  $\binom{C}{2}$  times of set comparison operations, where  $C$  is the number of clusters found by the attractor vertices. The complexity of this phase is roughly  $O(C^2)$ . This does not impose a significant impact to the overall complexity of the algorithm, since  $C \ll n$ . The time spent in this phase is often negligible. Experiments show that the clusters found using the attractor vertices are close to the final results. For applications where speed is more important than accuracy, the cluster merging phase can be left out.

When the LRW algorithm is used in local graph clustering problems, the first graph exploration (started from the seed vertex) has a complexity of  $O(KJ\bar{n}_c)$ . After the first graph exploration, there are  $LJ$  vertices to be further explored, where  $L$  is related to the

threshold  $\eta$  and  $L < 1$ . The overall complexity of the LRW local clustering algorithm is thus  $O(LKJ^2\bar{n}_c)$ .

The LRW algorithm is a typical example of embarrassingly parallel paradigm. In the graph exploring phase, each random walk can be executed independently. Therefore it can be entirely implemented in a parallel computing environment such as a high-performance computing system. The time spent for graph exploring phase decreases roughly linearly with respect to the number of available computing resources. The two-phase design also fits the MapReduce programming model and can easily be adapted into any MapReduce framework [23].

## Experiments

The LRW algorithm uses the following parameters: inflation exponent  $r$ , maximum number of iterations  $T_{max}$ , small value  $\epsilon$ , merging threshold  $\tau$  and local clustering threshold  $\eta$ . In practice, except the inflation exponent  $r$ , the values of the other parameters have little impact to the final results. The inflation power  $r$  should be chosen according to the density of the graph. A sparse graph should use a smaller value of  $r$ , though  $r = 2$  is suitable for most real world graphs. In our experiments, we chose  $r = 2$  unless otherwise specified. The other parameters have been set as:  $T_{max} = 100$ ,  $\epsilon = 10^{-5}$  and  $\tau = \eta = 0.3$ . We will show the impact of some parameters in "The sensitivity analysis of the parameters" section.

### Simulated data for global graph clustering problem

We first show the performance of the LRW algorithm using simulated graph data. The simulated graph is generated using the Erdos-Renyi model [24] with some modifications to generate clusters. Using the ground truth of the cluster structure, we can evaluate the performance of graph clustering algorithms. This kind of simulated data are widely used in the literature [5, 25–27].

The graphs are generated by the model  $G(n, p, c, q)$  where  $c$  is the number of clusters,  $n$  is the number of vertices,  $p$  is the probability of the link between two vertices, and  $q = d_{in}/d_{out}$  is the parameter that indicates the strength of the cluster structure, where  $d_{in}$  is the expected number of edges linking one vertex to other vertices inside the same cluster, and  $d_{out}$  is the expected number of edges linking a given vertex to other vertices in other clusters. Larger  $q$  indicates stronger cluster structure. When  $q = 1$ , each vertex has equal probability that it links to vertices that are inside and outside the cluster—the graph has a very weak cluster structure. Let  $d$  be the expected of degree of a vertex. So,  $d = d_{in} + d_{out} = p(n - 1)$ . We use this model to generate graphs that consist of  $c$  clusters and each cluster has the same number of vertices. For each pair of vertices, we link them with the probability  $\frac{qpc(n-1)}{(q+1)(n-c)}$  if they belong to the same cluster, and the probability of  $\frac{pc(n-1)}{n(q+1)(c-1)}$  if they belong to different clusters.

We use the normalized mutual information (NMI) to evaluate the clustering result against the ground truth [28, 29]. We first calculate the confusion matrix where each row is a cluster found by the clustering algorithm and each column is a cluster in the ground truth. The entries in the confusion matrix are the cardinality of the intersect set of the row cluster and the column cluster. Let  $N_{ij}$  be the values at the  $i$ -th row and the  $j$ -th column,  $N_{i-}$  the sum of the values at the  $i$ -th row,  $N_{-j}$  the sum of the values at the

$j$ -th column,  $N$  the total number of vertices,  $C_A$  the number of clusters that the clustering algorithm found (number of rows), and  $C_G$  the number of clusters in the ground truth (number of columns). The NMI is calculated as follows:

$$NMI = \frac{-2 \sum_{i=1}^{C_A} \sum_{j=1}^{C_G} N_{ij} \log(N_{ij}N/N_{i-}N_{-j})}{\sum_{i=1}^{C_A} N_{i-} \log(N_{i-}/N) + \sum_{j=1}^{C_G} N_{-j} \log(N_{-j}/N)}, \quad (11)$$

where  $0 \leq NMI \leq 1$ . If the clustering algorithm returns the exact same cluster structure as the ground truth,  $NMI = 1$ . Notice, NMI is not a symmetric evaluation metric. If an algorithm assigns all vertices into one cluster ( $C_A = 1$ ), then NMI value is 0. On the other hand, if an algorithm assigns each vertex to its own cluster ( $C_A = N$ ), then  $NMI > 0$ .

We generated graphs by choosing  $n = 128$  and  $d = 16$ . The number of the generated clusters is 4 and each cluster contains 32 vertices. We varied the ratio  $q$  and evaluated the performance of the LRW algorithm against Girvan-Newman (GN) [27], Louvain [6], Infomap [30] and MCL [12] algorithms. GN clusters a graph by iteratively removing edges according to their "betweenness" measures. Louvain optimizes the modularity measure of a graph using a greedy search paradigm. Infomap is another modularity-based algorithm. It starts with each vertex in its own cluster and iteratively merges the clusters, moves vertices between clusters or splitting a cluster until no better modularity measure can be found. MCL is a random walk based algorithm that also involves inflation operation. The differences between the MCL algorithm and the proposed one are explained in "Definitions" section.

Two simulated graphs are shown in Fig. 1, where the clusters are colored differently and the graphs are visualized by force-directed algorithms.

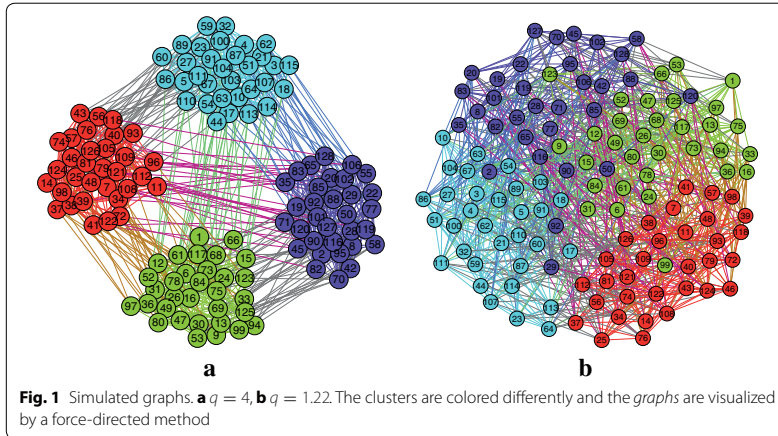
The comparative results are given in Table 1, where the number of clusters found by the algorithms is placed between parentheses.

From the results, Louvain is the best performing algorithm and the LRW algorithm comes as the second. It can be seen that the LRW algorithm can find the correct structure if the graph has a strong cluster structure. When the cluster structure diminishes as  $q$  decreases, the walking agents quickly spread to the whole graph before the contraction dominates. Thus, the LRW algorithm returns the whole graph as one cluster. This behavior is beneficial when we need to find the true clusters in a big graph. The GN and Louvain algorithms are more like graph partition algorithms. They optimize certain cluster fitness functions using the whole graph data. They tend to partition the graph into clusters even though the cluster structure is weak. That explains their better NMI scores in Table 1 when  $q$  is small.

We use real graph data to evaluate the performance of the LRW global clustering algorithm on heterogeneous graphs. Details of the experiments and the results are given in "Real world data" section.

### Simulated data for local graph clustering problems

In this section, we compare the LRW algorithm with other local clustering algorithms. The test graphs are generated using the protocol defined in [26]. To simulate the data that are close to real world graphs, the vertex degree and the cluster size are chosen to follow the power law. Each test graph contains 2048 vertices. The vertex degree has the



**Table 1** The NMI values and the numbers of clusters of the clustering results on simulated graph data

$q$	GN	Louvain	Infomap	MCL	LRW
4.0	0.975 (4)	1.0 (4)	1 (4)	1 (4)	1.0 (4)
3.0	1 (4)	1.0 (4)	1 (4)	1 (4)	1.0 (4)
2.33	0.950 (4)	1.0 (4)	1 (4)	0.860 (7)	1.0 (4)
1.86	0.900 (4)	1.0 (4)	1 (4)	0.478 (95)	1.0 (4)
1.5	0.890 (4)	1.0 (4)	0 (1)	0.453 (119)	0.975 (4)
1.22	0.593 (4)	0.771 (5)	0 (1)	0.444 (128)	0 (1)
1	0.232 (4)	0.304 (7)	0 (1)	0.444 (128)	0 (1)

minimum value of 16 and the maximum value of 128. The minimum and maximum cluster sizes are 16 and 256, respectively. Similar to the previous section, the inbound-outbound ratio  $q$  defines the strength of the cluster structure.

The competing algorithms are criteria-based algorithms that optimize a fitness function using either the greedy search or the simulated annealing optimization method. Let vertex set  $K$  be the cluster that contains the seed vertex.  $K^c = V \setminus K$  is the complement vertex set of  $K$ . Let function  $a(\cdot)$  be the total degree of a vertex set, that is

$$a(S) = \sum_{i \in S, j \in V} A_{ij}, \tag{12}$$

where  $A_{ij}$  are the entries of the adjacency matrix. The cut of the cluster  $K$  is defined as

$$c(K) = \sum_{i \in K, j \in K^c} A_{ij}. \tag{13}$$

The following are the definitions of the fitness functions.

Cheeger constant (conductance):

$$f(K) = \frac{c(K)}{\min(a(K), a(K^c))} \tag{14}$$

Normalized cut:

$$f(K) = \frac{c(K)}{a(K)} + \frac{c(K)}{a(K^c)} \quad (15)$$

Inverse relative density:

$$f(K) = \frac{|E| - a(K) + c(K)}{a(K) - c(K)} \quad (16)$$

Different local clustering algorithms are used to find the cluster that contains the seed vertex. The Jaccard index is used to evaluate the performance of each algorithm. Let  $K$  be the set of vertices that an algorithm finds and  $\mathcal{T}$  be the ground truth cluster that contains the seed vertex. The Jaccard index is defined as

$$J = \frac{|K \cap \mathcal{T}|}{|K \cup \mathcal{T}|} \quad (17)$$

We generated 10 test graphs for each inbound-outbound ratio  $q$ . From each generated graph, we randomly picked 20 vertices as seeds. For each algorithm and each inbound-outbound ratio  $q$ , we computed the Jaccard index for each seed and took the average of the 200 Jaccard indices. The results are shown in Table 2, where “Che” stands for the Cheeger constant (conductance) fitness function; “NCut” stands for the normalized cut fitness function; “IRD” stands for the inverse relative density; the ending letter “G” stands for the greedy search method; and the ending letter “S” stands for the simulated annealing method.

From the results, it is obvious that the LRW algorithm clearly outperforms other methods when the graph has a clear cluster structure. For the same reason explained in “[Simulated data for global graph clustering problem](#)” section, it does not give good result if the cluster structure is weak. This is the main difference between the LRW algorithm and graph partition algorithms.

### Real world data

In this section, we evaluate the performance of the LRW algorithm on some real world graph data.

#### *Zachary's karate club*

We first do clustering analysis on the Zachary's karate club graph data [31]. This graph is a social network of friendship in a karate club in 1970. Each vertex represents a club member and each edge represents the social interaction between the two members. During the study, the club split into two smaller ones due to the conflicts between the administrator and the coach. The graph data have been regularly used to evaluate the performance of the graph clustering algorithms [27, 30, 32]. The graph contains 34 vertices and 78 edges. We applied the LRW algorithm on this graph and the result shows two clusters that are naturally formed. Figure 2 shows the clustering result, where clusters are illustrated using different colors.



**Table 2** Jaccard index of local graph clustering results on the simulated graphs

q	CheG	CheS	NCutG	NCutS	IRDG	IRDS	LRW
4.0	0.753	0.840	0.752	0.820	0.753	0.830	<i>0.945</i>
3.0	0.671	0.812	0.671	0.801	0.671	0.798	<i>0.927</i>
2.33	0.668	0.774	0.668	0.758	0.668	0.776	<i>0.880</i>
1.86	0.593	0.650	0.593	0.681	0.593	0.684	<i>0.823</i>
1.5	0.492	0.630	0.493	0.629	0.492	0.609	<i>0.660</i>
1.22	0.444	0.544	0.437	<i>0.549</i>	0.444	0.529	0.504
1	0.298	0.410	0.296	<i>0.452</i>	0.298	0.424	0.295

As the figure shows, the LRW algorithm finds the two clusters of the Zachary's karate club. Actually the two clusters perfectly match the ground truth—how the club was split in 1970.

Clustering results of the GN, Louvain, Infomap and MCL algorithms are given in Additional file 1.

#### *Ego-Facebook graph data*

The second data we used is the ego-Facebook graph data [33]. The social network website Facebook allows users to organize their friends into “circles” or “friend lists” (for example, friends who share common interests). This data was collected from volunteer Facebook users for researchers to develop automatic circle finding algorithms. Ego-network is the network of an end user's friends. The ego-Facebook graph is a combination of ego-networks from 10 volunteer Facebook users. There are 4039 vertices and 88234 edges in the graph.

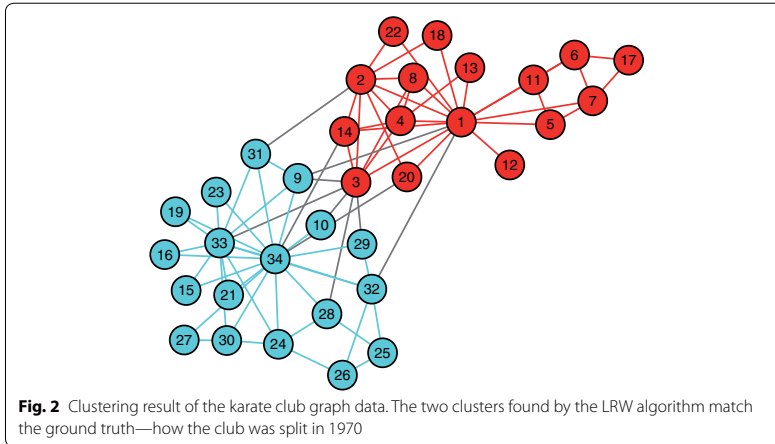
We applied the LRW, GN [27], Louvain [6], Infomap [30] and MCL [12] graph clustering algorithms to this data. To compare the results, we generated the ground truth clustering by combining the vertices in the “circles” of each volunteer user. So, the ground truth contains 10 clusters. If a vertex appears in the circles of more than one volunteer, we assign the vertex to all of these ground truth clusters. We evaluated the number of clusters and the NMI values of the results that each competing algorithm generated.

We also calculated the mean conductance (MC) value of the clustering results. The conductance value of a cluster is calculated using Eq. 14. We then took the mean of all the conductance values of the clusters that an algorithm finds. Smaller MC values indicate better clustering results. Note that MC tends to favor smaller numbers of clusters in general. If the numbers of clusters are roughly the same, MC values give good evaluation of the clustering results. It is also worth noting that MC value is capable of evaluating clustering algorithms without the ground truth. We shall use this metric in later experiments where the ground truth is not available.

The MC scores, NMI scores and the number of clusters found by each algorithm are reported in Table 3. A italic font indicates the best score among all competing algorithms.

The results show that the random-walk-based algorithms—LRW and MCL—are able to find the correct cluster structure of the data. Other criteria-based algorithms are sensitive to trivial disparities of the graph structure and are likely to overcluster the data.

The clustering result of the LRW algorithms is shown in Fig. 3.



**Table 3** Global graph clustering results on the ego-Facebook graph

	GN	Louvain	Infomap	MCL	LRW
Mean conductance	0.156	0.133	0.397	0.0882	<i>0.0770</i>
NMI	0.778	0.796	0.723	0.908	<i>0.910</i>
Number of clusters	16	19	76	10	10

Clustering results of other algorithms are shown in the Additional file 1.

#### **Heterogeneous graph data**

To evaluate the performance of the LRW algorithm on real heterogeneous graph data, we selected 5 graph data from the collection of the KONECT project [34]. The graph data are selected from different categories and the size of the graph data varies from small to medium. The properties and the references of the test graphs are shown in Table 4.

Since there is no ground truth available for these test data, we evaluated each clustering algorithm by the mean conductance (MC) values. The results are in Table 5. The best MC scores are shown in a italic font. The numbers of clusters found by each algorithm are placed between parentheses. We also plot the clustered graphs in which the vertices are located using a force-directed algorithm and colored according to their associated clusters. These clustered graphs are given in the Additional file 1 for subjective evaluation.

The reactome and the infectious graphs have low density. We chose the inflation exponent  $r = 1.2$  to prevent overclustering the data. For other graph data, the default value  $r = 2$  is used.

Based on the MC scores and the visualized clustering results, the LRW algorithm achieves a superior clustering performance in most of the cases. Note that the reactome data has a weak cluster structure, thus the LRW algorithm has difficulty to find a good partition for it.



### The sensitivity analysis of the parameters

The proposed LRW algorithm depends on a number of parameters to perform global and local graph clustering. In this section, we perform the sensitivity analysis on the parameters.

We use both simulated and real-world graphs in our experiments. Test graph G1, G2 and G3 are similar to those used in "Simulated data for global graph clustering problem" section except that we vary the density of each graph. The expected degree  $d$ , which is a measure of the graph density, of graph G1, G2 and G3 are 12, 16 and 20 respectively and  $q$  is set to be 1.86 for all test graphs. Test graph G4 is generated in the same way as described in "Simulated data for local graph clustering problems" section. The

**Table 4 Properties of the heterogeneous graphs used for testing**

	Vertices	Edges	Category	Reference
Dolphins	62	156	Animal	[35]
Arenes-jazz	198	2742	Human social	[36]
Infectious	410	2765	Human contact	[37]
Polblogs	1490	19,090	Hyperlink	[38]
Reactome	6229	146,160	Metabolic	[39]

**Table 5 Global graph clustering results on the real heterogeneous graph data**

	GN	Louvain	Infomap	MCL	LRW
Dolphins	0.425 (4)	0.440 (5)	0.487 (6)	0.675 (12)	0.347 (4)
Arenes-jazz	0.485 (4)	0.455 (4)	0.577 (7)	0.529 (5)	0.364 (4)
Infectious	0.162 (5)	0.214 (6)	0.465 (17)	0.673 (40)	0.175 (5)
Polblogs	0.524 (12)	0.501 (11)	0.727 (36)	0.777 (45)	0.427 (11)
Reactome	0.108 (110)	0.099 (114)	0.315 (248)	0.478 (352)	0.221 (191)

ego-Facebook graph data in "[Ego-Facebook graph data](#)" section is used as an example of real-world graphs. We performed global graph clustering on these test graphs using the LRW algorithm with different parameters. The NMI scores are used to evaluate the performance of the algorithm. The experiments using simulated graph data were repeated 10 times and the average NMI scores and the average number of clusters are reported.

As described in "[Limited random walk on general graphs](#)" section, the most important parameter of the LRW algorithm is the inflation exponent  $r$ . We first set  $T_{max} = 100$ ,  $\epsilon = 10^{-5}$ ,  $\tau = 0.3$  and vary the inflation exponent  $r$ . Table 6 shows the NMI scores and the number of clusters reported by the LRW algorithm with different values of  $r$ .

The test results show the relationship among the inflation exponent  $r$ , the density of the test graphs and the performance the LRW algorithm. A large inflation exponent  $r$  may overcluster the data as the results on graph G1 and G2 shows. It can be easily noticed that the LRW algorithm is not sensitive to the choice of  $r$  for test graph G4 and the ego-Facebook graph. These graphs, and almost all real-world graphs, are more heterogeneous than the simulated graphs G1, G2 and G3. The LRW algorithm performs better on this type of graph since the attractor vertices and significant vertices are more stable on these graphs.

The parameter  $T_{max}$  sets a limit on the number of iterations for the LRW procedure to converge. According to our experiments, value 100 is large enough to ensure the convergence of almost all cases. For example, only 3 out of 88,234 LRW procedures do not converge within 100 iterations on the ego-Facebook graph. A few exceptional cases has no impact on the final clustering results. The parameter  $\epsilon$  is used to remove small values in the probability vector thus decrease the computational complexity. It has no impact on the final clustering result as long as the value is small enough, for example  $\epsilon < 10^{-4}$ .

We also conducted the experiments by varying the threshold value  $\tau$  from 0.1 to 0.5. The NMI scores and the number of clusters found by the LRW algorithm with different  $\tau$  values are almost identical to the values in the corresponding cells in Table 6. This indicates that the choice of  $\tau$  has very little impact on the clustering performance.

**Table 6** The NMI scores and the number of clusters by the different inflation exponent values

<i>r</i>	G1	G2	G3	G4	Facebook
1.2	0 (1)	0 (1)	0 (1)	0.155 (5)	0.902 (10)
1.4	0 (1)	0 (1)	0 (1)	0.927 (22.7)	0.906 (10)
1.6	0 (1)	0 (1)	0 (1)	1.0 (23)	0.908 (11)
1.8	1 (4)	1 (4)	1 (4)	1.0 (20.8)	0.910 (10)
2	0.971 (4.6)	1 (4)	1 (4)	1.0 (25)	0.910 (10)
2.4	0.868 (7.0)	0.990 (4.2)	1 (4)	1.0 (21.8)	0.910 (10)
3	0.822 (6.3)	0.962 (4.8)	0.988 (4.2)	1.0 (24.3)	0.910 (10)

According to these results, one only needs to choose a proper inflation exponent  $r$  to use the LRW algorithms. Other parameters can be chosen freely from a wide range of reasonable values.  $r = 2$  is suitable for most of graphs and is preferable because of the computational advantage.

### Big graph data

In this section, we apply the LRW algorithm on real-world big graph data and show the computational advantage of its parallel implementation. The test graphs were received from the SNAP graph data collection [40, 41]. These graphs are from major social network services and E-commerce companies. We use the high quality communities that either created by users or the system as ground truth clusters. The details of the high quality communities are described in [41]. The Rand index is used to evaluate the results of the proposed clustering algorithm. To generate positive samples, we randomly picked 1000 pairs of vertices, where the vertices in each pair come from the same cluster in the ground truth. Negative samples consist of 1000 pairs of vertices, where the vertices from each pair come from different clusters in the ground truth. The Rand index is defined as

$$RI = \frac{TP + TN}{N}, \quad (18)$$

where  $TP$  is the number of true positive samples,  $TN$  is the number of true negative samples, and  $N$  is the total number of samples.

Since none of the competing algorithms used in previous sections can complete this task due to the large size of the data, we only report the results from the LRW algorithm. Table 7 shows the size of the test graphs, the time spent on the graph exploration phase, the number of CPU cores and the amount of memory used for graph exploration, the time spent on cluster merging phase, the number of clusters that the LRW algorithm finds and the Rand index of the clustering results. In this experiment, multiple CPU cores were used for graph exploration and one CPU core was used for clustering merging.

Table 7 shows that the LRW algorithm is able to find clusters from large graph data with a reasonable computing time and memory usage. The Rand index values indicate that the clusters returned by the LRW algorithm match well the ground truth. The time spent on the graph exploration phase is inversely proportional to the number of CPU

**Table 7 Clustering performance of real-world big graph data**

	com-Amazon	com-Youtube	com-LiveJournal	com-Orkut
Vertices	334,863	1,134,890	3,997,962	3,072,441
Edges	925,872	2,987,624	34,681,189	117,185,083
CPU cores (graph exploration)	12	96	96	96
Memory per CPU core	4G	4G	8G	8G
Graph exploration (in hours)	0.83	3.08	17.4	24.0
Cluster merging (in hours)	0.20	1.34	9.44	2.53
Clusters	37,473	170,569	381,246	165,624
Rand index	0.908	0.755	0.951	0.751

cores. Computational time can be further reduced if more computing resources are available. The proposed algorithm can efficiently handle graphs with millions of vertices and hundreds of millions of edges. For even larger graphs that exceed the memory limit for each computing process, a mechanism that retrieves part of the graph from a central storage can be used. Since the LRW procedure is capable of exploring a limited number of vertices that are near a seed vertex, the algorithm can cluster much larger graphs if such a mechanism is implemented.

### Conclusions

In this paper, we proposed a novel random-walk-based graph clustering algorithm, the so-called LRW. We studied the behavior of the LRW procedure and developed the LRW algorithms for both global and local graph clustering problems. The proposed algorithm is fundamentally different from previous random-walk-based algorithms. We use the LRW procedure to find attracting vertices and use them as features to cluster vertices in a graph. The performance of the LRW algorithm was evaluated using simulated graphs and real-world big graph data. According to the results, the proposed algorithm is superior to other well-known methods.

The LRW algorithm can be efficiently used in both global and local graph clustering problems. It finds clusters from a big graph data by only locally exploring the graph. This is important for extreme large data that may not even fit in a single computer memory. The algorithm contains two phases—the graph exploring phase and the cluster merging phase. The graph exploring phase is the most critical part and also the most time-consuming part of the algorithm. This phase can be implemented in embarrassingly parallel paradigm. The algorithm can easily be adapted to any MapReduce framework.

From our experiments, we also noticed the limitations of the LRW algorithm. First, when used as a global clustering algorithm, the computational complexity can be high, especially when the graph cluster structure is weak. This is due to the fact that the graph may be analyzed multiple times during the graph exploration phase, if we perform the LRW procedure from every vertex of the graph. However, using the multi-stage strategy can dramatically reduce the computation time. Second, if the cluster structure is weak, the LRW algorithm may return the whole graph as one cluster—though this behavior is desired in many cases.

The experiments show that the performance of the proposed LRW graph clustering algorithm is not sensitive to any parameter except the inflation exponent  $r$ , especially

when the graph is not heterogeneous. For future research, we will further improve the LRW algorithm so that it can optimally select the inflation function that best suits the problem at hand.

### Additional file

**Additional file 1.** Clustering results on graphs used in the experiments of various methods.

### Abbreviations

GN: Girvan-Newman; LRW: limited random walk; MCL: Markov clustering algorithm; MC: mean conductance; NMI: normalized mutual information.

### Authors' contributions

HZ carried out the conception and design of the study, participated in the analysis and interpretation of data, and was involved in drafting and revising the manuscript. JR, SK and MG made substantial contributions to the design of the study, the analysis and interpretation of the data, and were involved in critically reviewing the manuscript. All authors read and approved the final manuscript.

### Author details

<sup>1</sup> Department of Signal Processing, Tampere University of Technology, Korkeakoulunkatu 1, 33101 Tampere, Finland.

<sup>2</sup> Electrical Engineering Department, College of Engineering, Qatar University, 2713, Al Hala St, Doha, Qatar.

### Competing interests

The authors declare that they have no competing interests.

Received: 23 September 2016 Accepted: 15 November 2016

Published online: 01 December 2016

### References

- Benson AR, Gleich DF, Leskovec J. Higher-order organization of complex networks. *Science*. 2016;353(6295):163–6.
- Schaeffer SE. Graph clustering. *Comput Sci Rev*. 2007;1(1):27–64.
- Lambiotte R, Delvenne JC, Barahona M. Random walks, Markov processes and the multiscale modular organization of complex networks. *IEEE Trans Netw Sci Eng*. 2014;1(2):76–90.
- He P, Xu X, Hu K, Chen L. Semi-supervised clustering via multi-level random walk. *Pattern Recognit*. 2014;47(2):820–32.
- Newman ME. Fast algorithm for detecting community structure in networks. *Phys Rev E*. 2004;69(6):066133.
- Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. *J Stat Mech Theory Exp*. 2008;2008(10):10008.
- Clauset A, Newman ME, Moore C. Finding community structure in very large networks. *Phys Rev E*. 2004;70(6):066111.
- Waltman L, van Eck NJ. A smart local moving algorithm for large-scale modularity-based community detection. *Eur Phys J B*. 2013;86(11):1–14.
- Spielman DA, Teng SH. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. [arXiv:0809.3232](https://arxiv.org/abs/0809.3232); 2008.
- Qiu H, Hancock ER. Graph matching and clustering using spectral partitions. *Pattern Recognit*. 2006;39(1):22–34.
- Spielman D, Teng S. Spectral sparsification of graphs. *SIAM J Comput*. 2011;40(4):981–1025.
- Dongen S. Graph clustering by flow simulation. PhD thesis, Universiteit Utrecht, Utrecht, The Netherlands; 2000.
- Macropol K, Can T, Singh AK. RRW: repeated random walks on genome-scale protein networks for local cluster discovery. *BMC Bioinform*. 2009;10(1):283.
- Xin Y, Xie Z-Q, Yang J. The adaptive dynamic community detection algorithm based on the non-homogeneous random walking. *Phys A Stat Mech Appl*. 2016;450:241–52.
- Chung F, Kempton M. A local clustering algorithm for connection graphs. In: *Algorithms and models for the web graph*; 2013. p. 26–43.
- Macko P, Margo D, Seltzer M. Local clustering in provenance graphs. In: *Proceedings of the 22nd ACM international conference on information & knowledge management*; 2013. p. 835–840.
- Andersen R, Chung F, Lang K. Using pagerank to locally partition a graph. *Internet Math*. 2007;4(1):35–64.
- Buhler T, Rangapuram SS, Setzer S, Hein M. Constrained fractional set programs and their application in local clustering and community detection. [arXiv:1306.3409](https://arxiv.org/abs/1306.3409); 2013.
- Zhu ZA, Lattanzi S, Mirrokhi V. A local algorithm for finding well-connected clusters. In: *Proceedings of the 30th international conference on machine learning (ICML-13)*; 2013. p. 396–404.
- Norris JR. Markov chains applied probability and stochastic networks; 1998.
- Harel D, Koren Y. On clustering using random walks. In: Hariharan R, Vinay V, Mukund M, et al., editors. *Theoretical computer science, Lecture notes in computer science*. Berlin: Springer; 2001. p. 18–41.

22. Cai B, Wang H, Zheng H, Wang H. An improved random walk based clustering algorithm for community detection in complex networks. In: 2011 IEEE international conference on systems, man, and cybernetics (SMC); 2011. p. 2162–2167.
23. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. *Commun ACM*. 2008;51(1):107–13.
24. Newman M. *Networks: an introduction*. 1st ed. New York: Oxford; 2010.
25. Danon L, Díaz-Guilera A, Arenas A. The effect of size heterogeneity on community identification in complex networks. *J Stat Mech Theory Exp*. 2006;2006(11):11010.
26. Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. *Phys Rev E*. 2008;78(4):046110.
27. Newman ME, Girvan M. Finding and evaluating community structure in networks. *Phys Rev E*. 2004;69(2):026113.
28. Ana LNF, Jain AK. Robust data clustering. In: *Proceedings 2003 IEEE computer society conference on computer vision and pattern recognition*, vol. 2; 2003. p. 128–1332.
29. Danon L, Diaz-Guilera A, Duch J, Arenas A. Comparing community structure identification. *J Stat Mech Theory Exp*. 2005;2005(09):09008.
30. Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. *Proc Natl Acad Sci*. 2008;105(4):1118–23.
31. Zachary WW. An information flow model for conflict and fission in small groups. *J Anthropol Res*. 1977;1:452–73.
32. Sahai T, Speranzon A, Banaszuk A. Hearing the clusters of a graph: a distributed algorithm. *Automatica*. 2012;48(1):15–24.
33. Leskovec J, McAuley JJ. Learning to discover social circles in ego networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. *Advances in neural information processing systems 25*; 2012. p. 539–547.
34. Kunegis J. Konect—the Koblenz network collection. In: *Proceedings of international conference on World Wide Web companion*; 2013. p. 1343–1350.
35. Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behav Ecol Sociobiol*. 2003;54(4):396–405.
36. Gleiser P, Danon L. Community structure in jazz. *Adv Complex Syst*. 2003;06(04):565–73.
37. Isella L, Stehlé J, Barrat A, Cattuto C, Pinton JE, van den Broeck W. What's in a crowd? Analysis of face-to-face behavioral networks. *J Theor Biol*. 2011;271(1):166–80.
38. Adamic LA, Glance N. The political blogosphere and the 2004 US election: divided they blog. *Proceedings of the 3rd international workshop on link discovery*, LinkKDD '05 New York, NY, USA; 2005. p. 36–43.
39. Croft D, Mundo AF, Haw R, Milacic M, Weiser J, Wu G, Caudy M, Garapati P, Gillespie M, Kamdar MR, Jassal B, Juepe S, Matthews L, May B, Palatnik S, Rothfels K, Shamovsky V, Song H, Williams M, Birney E, Hermjakob H, Stein L, D'Eustachio P. The reactome pathway knowledgebase. *Nucleic Acids Res*. 2013;1102:472–7.
40. Leskovec J. Stanford large network dataset collection.
41. Yang J, Leskovec J. Defining and evaluating network communities based on ground-truth. [arXiv:1205.6233](https://arxiv.org/abs/1205.6233); 2012.



# PUBLICATION

II

**Outlier edge detection using random graph generation models and applications**

H. Zhang, S. Kiranyaz and M. Gabbouj

*Journal of Big Data* 4.1 (2017), 11

DOI: 10.1186/s40537-017-0073-8

**Publication reprinted with the permission of the copyright holders**



RESEARCH

Open Access



# Outlier edge detection using random graph generation models and applications

Honglei Zhang<sup>1\*</sup> , Serkan Kiranyaz<sup>2</sup> and Moncef Gabbouj<sup>1</sup>

\*Correspondence:  
honglei.zhang@tut.fi  
<sup>1</sup> Department of Signal  
Processing, Tampere  
University of Technology,  
Finland, Korkeakoulunkatu 1,  
FI-33101 Tampere, Finland  
Full list of author information  
is available at the end of the  
article

## Abstract

Outliers are samples that are generated by different mechanisms from other normal data samples. Graphs, in particular social network graphs, may contain nodes and edges that are made by scammers, malicious programs or mistakenly by normal users. Detecting outlier nodes and edges is important for data mining and graph analytics. However, previous research in the field has merely focused on detecting outlier nodes. In this article, we study the properties of edges and propose effective outlier edge detection algorithm. The proposed algorithms are inspired by community structures that are very common in social networks. We found that the graph structure around an edge holds critical information for determining the authenticity of the edge. We evaluated the proposed algorithms by injecting outlier edges into some real-world graph data. Experiment results show that the proposed algorithms can effectively detect outlier edges. In particular, the algorithm based on the Preferential Attachment Random Graph Generation model consistently gives good performance regardless of the test graph data. More important, by analyzing the authenticity of the edges in a graph, we are able to reveal underlying structure and properties of a graph. Thus, the proposed algorithms are not limited in the area of outlier edge detection. We demonstrate three different applications that benefit from the proposed algorithms: (1) a preprocessing tool that improves the performance of graph clustering algorithms; (2) an outlier node detection algorithm; and (3) a novel noisy data clustering algorithm. These applications show the great potential of the proposed outlier edge detection techniques. They also address the importance of analyzing the edges in graph mining—a topic that has been mostly neglected by researchers.

**Keywords:** Outlier detection, Graph mining, Outlier edge

## Background

Graphs are an important data representation, which have been extensively used in many scientific fields such as data mining, bioinformatics, multimedia content retrieval and computer vision. For several hundred years, scientists have been enthusiastic about graph theory and its applications [1]. Since the revolution of the computer technologies and the Internet, graph data have become more and more important because many of the “big” data are naturally formed in a graph structure or can be transformed into graphs.

Outliers almost always happen in real-world graphs. Outliers in a graph can be outlier nodes or outlier edges. For example, outlier nodes in a social network graph may

include: scammers who steal users' personal information; fake accounts that manipulate the reputation management system; or spammers who send free and mostly false advertisements [2–4]. Researchers have been working on algorithms to detect these malicious outlier nodes in graphs [5–8]. Outlier edges are also common in graphs. They can be edges that are generated by outlier nodes, or unintentional links made by normal users or the system. Outlier edges are not only harmful but also greatly increase the system complexity and degrade the performance of graph mining algorithms. In this paper, we will show that the performance of the community detection algorithms can be greatly improved when a small amount of outlier edges are removed. Outlier edge detection can also help evaluate and monitor the behavior of end users and further identify the malicious entities. However, in contrast to the focus on the outlier node detection, there have been very few studies on outlier edge detection.

In this paper, we first propose an authentic score of an edge using the clustering property of social network graphs. The authentic score of an edge is determined by the difference of the actual and the expected number of edges that link the two groups of nodes that are around the investigating edge. We use random graph generation models to predict the number of edges between the two groups of nodes. The edges with low authentic scores, which are also called weak links in this paper, are likely to be outliers. We evaluated the outlier edge detection algorithm that is based on the authentic score using injected edges in real-world graph data.

Later, we show the great potentials of the outlier edge detection technique in the areas of graph mining and pattern recognition. We demonstrate three different applications that are based on the proposed algorithms: (1) a preprocessing tool for graph clustering algorithms; (2) an outlier node detection algorithm; (3) a novel noisy data clustering algorithm.

The rest of the paper is organized as follows: the prior art is reviewed in "[Previous work](#)"; the methodology to determine the authentic scores of edges is in "[Methods](#)"; evaluation of the proposed outlier edge detection algorithms are given in "[Evaluation of the proposed algorithms](#)"; various applications that use or benefit from outlier edge detection algorithms are presented in "[Applications](#)"; and finally, conclusions and future directions are included in "[Conclusions](#)".

## **Previous work**

Outliers are data instances that are markedly different from the rest of the data [9]. Outliers are often located outside (mostly far way) from the normal data points when presented in an appropriate feature space. It is also commonly assumed that the number of outliers is much less than the number of normal data points.

Outlier detection in graph data includes outlier node detection and outlier edge detection. Noble and Cook studied substructures of graphs and used the Minimum Description Length technique to detect unusual patterns in a graph [6]. Xu et al. considered nodes that marginally connect to a structure (or community) as outliers [10]. They used a searching strategy to group the nodes that share many common neighbors into communities. The nodes that are not tightly connected to any community are classified as outliers. Gao et al. also studied the roles of the nodes in communities [11]. Nodes in a community tend to have similar attributes. Using the Hidden Markov Random Field

technique as a generative model, they were able to detect the nodes that are abnormal in their community. Akoglu et al. detected outlier nodes using the near-cliques and stars, heavy vicinities and dominant heavy links properties of the ego-network- the induced network formed by a focal node and its direct neighbors [12]. They observed that some pairs of the features of normal nodes follow a power law and defined an outlier score function that measures the deviation of a node from the normal patterns. Dai et al. detected outlier nodes in bipartite graphs using mutual agreements between nodes [7].

In contrast to proliferative research on outlier node detection, there have been very few studies on outlier edge detection in graphs. Liu et al. find outlier pairs in a complex network by evaluating the structural and semantic similarity of each pair of the connected nodes [13]. Chakrabarti detected outlier edges by partitioning nodes into groups using the Minimum Description Length technique [14]. Edges that link the nodes from different groups are considered as outliers. These edges are also called weak links or weak ties in literature [15]. Obviously this method has severe limitations. First, one shall not classify all weak links as outliers since they are part of the normal graph data. Second, many outlier edges do not happen between the groups. Finally, many graphs do not contain easily partitionable groups.

Detection of missing edges (or link prediction) is the opposite technique of outlier edge detection. These algorithms find missing edges between pairs of nodes in a graph. They are critical in recommendation systems, especially in e-commerce industry and social network service industry [16, 17]. Such algorithms evaluate similarities between each pair of nodes. A pair of nodes with high similarity score is likely to be connected by an edge. One may use the similarity scores to detect outlier edges. The edges whose two end nodes have a low similarity score are likely to be the outlier edges. However, in practice, these similarity scores do not give satisfactory performance if one uses them to detect outlier edges.

## Methods

### Notation

Let  $G(V, E)$  denote a graph with a set of nodes  $V$  and a set of edges  $E$ . In this article, we consider undirected, unweighted graphs that do not contain self-loops. We use lower case  $a, b, c$ , etc., to represent nodes. Let  $\overline{ab}$  denote the edge that connects nodes  $a$  and  $b$ . Because our graph  $G$  is undirected,  $\overline{ab}$  and  $\overline{ba}$  represent the same edge. Let  $N_a$  be the set of neighboring nodes of node  $a$ , such that  $N_a = \{x | x \in V, \overline{ax} \in E\}$ . Let  $S_a = N_a \cup \{a\}$  (i.e.  $S_a$  contains node  $a$  and its neighboring nodes). Let  $k_a$  be the degree of node  $a$ , so that  $k_a = |N_a|$ . Let  $A$  be the adjacency matrix of graph  $G$ . Let  $n = |V|$  be the number of nodes and  $m = |E|$  be the number of edges of graph  $G$ .

Freeman defines the ego-network as the induced subgraph that contains a focal node and all of its neighboring nodes together with edges that link these nodes [18]. To study the properties of an edge, we define the edge-ego-network as follows:

**Definition 1** An edge-ego-network is the induced subgraph that contains the two end nodes of an edge, all neighboring nodes of these two end nodes and all edges that link these nodes.

Let  $G_{\overline{ab}} = G(V_{\overline{ab}}, E_{\overline{ab}})$  denote the edge-ego-network of edge  $\overline{ab}$ , where  $V_{\overline{ab}} = S_a \cup S_b$  and  $E_{\overline{ab}} = \{\overline{xy} | x \in V_{\overline{ab}}, y \in V_{\overline{ab}} \text{ and } \overline{xy} \in E\}$ .

**Motivation**

Graphs representing real-world data, in particular social network graphs, often exhibit the clustering property- nodes tend to form highly dense groups in a graph [19]. For example, if two people have many friends in common, they are likely to be friends too. Therefore, it is common for social network services to recommend new connections to a user using this clustering property [16]. As a consequence, social network graphs display an even stronger clustering property compared to other graphs. New connections to a node may be recommended from the set of neighboring nodes with the highest number of common neighbors to the given node. The common neighbors (CN) score of node  $a$  and node  $b$  is defined as

$$s_{CN} = |N_a \cap N_b|. \tag{1}$$

Common neighbors score is the basis of many node similarity scores that have been used to find missing edges [16]. Some common similarity indices are:

- Salton index or cosine similarity (Salton)

$$s_{Salton} = \frac{S_{CN}}{\sqrt{k_a k_b}} \tag{2}$$

- Jaccard index (Jaccard)

$$s_{Jaccard} = \frac{S_{CN}}{|N_a \cup N_b|} \tag{3}$$

- Hub promoted index (HPI)

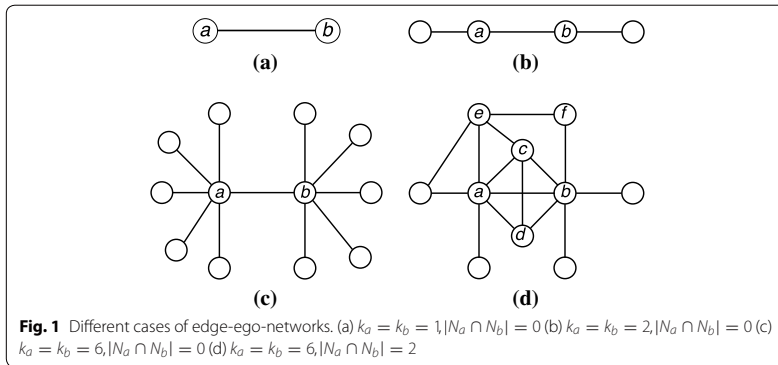
$$s_{HPI} = \frac{S_{CN}}{\min(k_a, k_b)} \tag{4}$$

- Hub depressed index (HDI)

$$s_{HDI} = \frac{S_{CN}}{\max(k_a, k_b)} \tag{5}$$

Next we shall investigate how to detect outlier edges in a social network using the clustering property. According to this property, if two people are friends, they are likely to have many common friends or their friends are also friends of each other. If two people are linked by an edge, but do not share any common friends and neither do their friends know each other, we have good reason to suspect that the link between them is an outlier. So, when node  $a$  and node  $b$  are connected by edge  $\overline{ab}$ , there should be edges connect the nodes in set  $S_a$  and the nodes in set  $S_b$ . However, the number of connections should depend on the number of nodes in these two groups. Let us consider the different cases as shown in Fig. 1.

In these four cases, edge  $\overline{ab}$  is likely to be a normal edge in case (d) because nodes  $a$  and  $b$  share common neighboring nodes  $c$  and  $d$ , and there are connections between



neighboring nodes of  $a$  and those of  $b$ . In the case of (a), (b) and (c),  $|N_a \cap N_b| = 0$ , which implies that nodes  $a$  and  $b$  do not share any common neighboring nodes. However edge  $\overline{ab}$  in case (c) is more likely to be an outlier edge because nodes  $a$  and  $b$  have each many neighboring nodes but there is no connection between any two of these neighboring nodes. In case (a) and (b) we do not have enough information to judge whether edge  $\overline{ab}$  is an outlier edge or not. If we apply the node similarity scores to detect outlier edges, we find that  $S_{CN} = 0$  for cases (a), (b) and (c). Thus, the node similarity scores defined by Eqs. (1), (2), (3), (4) and (5) all equal to 0. For this reason, these node similarity scores cannot effectively detect outlier edges.

In case (c), edge  $\overline{ab}$  is likely to be an outlier edge because the expected number of edges between node  $a$  together with its neighboring nodes and node  $b$  together with its neighboring nodes is high, whereas the actual number of edges is low. So, according to the clustering property, we propose the following definition for the authentic score of an edge:

**Definition 2** The authentic score of an edge is defined as the difference between the number of actual edges and the expected value of the number of edges that link the two sets of neighboring nodes of the two end nodes of the given edge. That is:

$$s_{\overline{ab}} = m_{\overline{ab}} - e_{\overline{ab}}, \tag{6}$$

where  $m_{\overline{ab}}$  is the actual number of edges that links the two sets of nodes- one set is node  $a$  together with its neighboring nodes and the other set is node  $b$  together with its neighboring nodes, and  $e_{\overline{ab}}$  is the expected number of edges that link the aforementioned two sets of nodes.

We can rank the edges by their authentic scores defined in Eq. (6). The edges with low scores are more likely to be outlier edges in a graph.

Let  $\alpha(S, T) = |\overline{ab}|_{a \in S, b \in T \text{ and } \overline{ab} \in E}$  denote the number of edges that links the nodes in sets  $S$  and  $T$ . We suppose the graph  $G$  is generated by a random graph generation model. Let  $\epsilon(S, T)$  denote the expected value of the number of edges that links the nodes in sets  $S$  and  $T$  by the generation model. "Expected number of edges between two

sets of nodes" describes two generation models and the functions of calculating  $\epsilon(S, T)$ . Obviously  $\alpha(S, T)$  and  $\epsilon(S, T)$  are symmetric functions. That is:

**Theorem 1**  $\alpha(S, T) = \alpha(T, S)$  and  $\epsilon(S, T) = \epsilon(T, S)$ .

Let  $P_{a,b}$  and  $R_{a,b}$  be the two sets of nodes that are related to end nodes  $a$  and  $b$ . Node set  $R_{a,b}$  depends on set  $P_{a,b}$ . The actual number of edges and the expected number of edges of the sets of nodes related to the two end nodes may vary when we switch the end nodes  $a$  and  $b$ . We use the following equations to calculate  $m_{ab}^-$  and  $e_{ab}^-$ :

$$m_{ab}^- = \frac{1}{2} (\alpha(P_{a,b}, R_{a,b}) + \alpha(P_{b,a}, R_{b,a})); \tag{7}$$

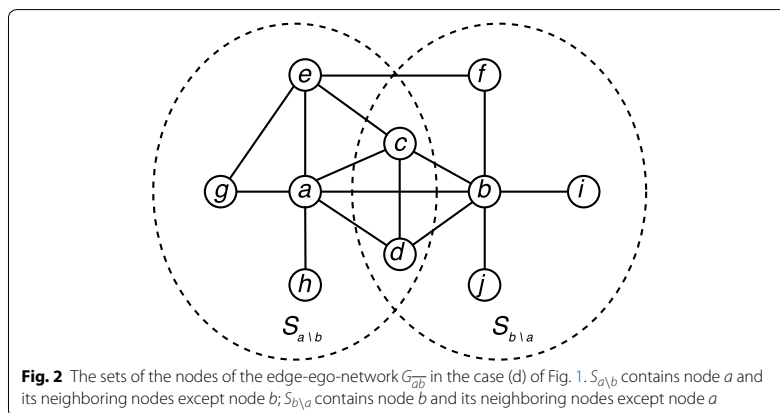
$$e_{ab}^- = \frac{1}{2} (\epsilon(P_{a,b}, R_{a,b}) + \epsilon(P_{b,a}, R_{b,a})). \tag{8}$$

**Schemes of node neighborhood sets**

For a ego-network, Coscia and Rossetti showed the importance of removing the focal node and all edges that link to it when studying the properties of ego-networks [20]. It is more complicate to study the properties of an edge-ego-network since there are two ending nodes and two sets of neighboring nodes involved. Considering the common nodes of the neighboring nodes and the end nodes of the edge being investigated, we now define four schemes that capture different configurations of these two sets.

Let  $S_{a \setminus b} = S_a \setminus \{b\}$  be the set of nodes that contains node  $a$  and its neighboring nodes except node  $b$ . Let  $N_{a \setminus b} = N_a \setminus \{b\}$  be the set of nodes that contains the neighboring nodes of  $a$  except node  $b$ . Obviously  $S_{a \setminus b} = N_{a \setminus b} \cup \{a\}$ . Fig. 2 shows the edge-ego-network  $G_{ab}^-$  and the two sets of nodes  $S_{a \setminus b}$  and  $S_{b \setminus a}$  corresponding to case (d) in Fig. 1.

We first define two sets of nodes that are related to node  $a$  and its neighboring nodes:  $N_{a \setminus b}$  and  $S_{a \setminus b}$ . Next, we define two sets of nodes that are related to node  $b$  and its neighboring nodes with regard to the sets of nodes  $N_{a \setminus b}$  and  $S_{a \setminus b}$ :  $S_{b \setminus a} \setminus S_{a \setminus b}$  and  $S_{b \setminus a}$ . In Fig. 2,  $N_{a \setminus b} = \{c, d, e, g, h\}$ ,  $S_{a \setminus b} = \{a, c, d, e, g, h\}$ ,  $S_{b \setminus a} \setminus S_{a \setminus b} = \{b, f, i, j\}$  and





$S_{b \setminus a} = \{b, c, d, f, i, j\}$ . In the case of a social network graph,  $N_{a \setminus b}$  would consist of friends of user (node)  $a$  except  $b$ ;  $S_{a \setminus b}$  consists of  $a$  and friends of  $a$  except  $b$ ;  $S_{b \setminus a} \setminus S_{a \setminus b}$  consists of  $b$  and friends of  $b$  except  $a$  and those who are friends of  $a$ ;  $S_{b \setminus a}$  consists of  $b$  and friends of  $b$  except  $a$ .

Based on the set pairs of nodes  $a$  and  $b$ , we define the following four schemes and their meanings in the case of a social network graph. We use superscript (1), (2), (3) and (4) to indicate the four schemes respectively.

- Scheme 1 :  $P_{a,b}^{(1)} = N_{a \setminus b}$  and  $R_{a,b}^{(1)} = S_{b \setminus a} \setminus S_{a \setminus b}$   
How many of  $a$ 's friends know  $b$  and his friends outside of the relationship with  $a$ ?
- Scheme 2 :  $P_{a,b}^{(2)} = N_{a \setminus b}$  and  $R_{a,b}^{(2)} = S_{b \setminus a}$   
How many of  $a$ 's friends know  $b$  and his friends?
- Scheme 3 :  $P_{a,b}^{(3)} = S_{a \setminus b}$  and  $R_{a,b}^{(3)} = S_{b \setminus a} \setminus S_{a \setminus b}$   
How many of  $a$  and his friends know  $b$  and his friends outside of the relationship with  $a$ ?
- Scheme 4 :  $P_{a,b}^{(4)} = S_{a \setminus b}$  and  $R_{a,b}^{(4)} = S_{b \setminus a}$   
How many of  $a$  and his friends know  $b$  and his friends?

For the edge-ego-network  $G_{\overline{ab}}$  shown in Fig. 2, scheme 1 examines edges  $\overline{ef}$ ,  $\overline{cb}$  and  $\overline{db}$ ; scheme 2 examines edges  $\overline{ef}$ ,  $\overline{ec}$ ,  $\overline{cb}$ ,  $\overline{cd}$ ,  $\overline{dc}$  and  $\overline{db}$ ; scheme 3 examines edges  $\overline{ab}$ ,  $\overline{ef}$ ,  $\overline{cb}$  and  $\overline{db}$ ; scheme 4 examines edges  $\overline{ab}$ ,  $\overline{ac}$ ,  $\overline{ad}$ ,  $\overline{ef}$ ,  $\overline{ec}$ ,  $\overline{cb}$ ,  $\overline{db}$ ,  $\overline{dc}$  and  $\overline{cd}$ .

Next we study the symmetric property of these four schemes.

**Theorem 2**  $\alpha \left( P_{a,b}^{(2)}, R_{a,b}^{(2)} \right) = \alpha \left( P_{b,a}^{(2)}, R_{b,a}^{(2)} \right)$  and  $\alpha \left( P_{a,b}^{(4)}, R_{a,b}^{(4)} \right) = \alpha \left( P_{b,a}^{(4)}, R_{b,a}^{(4)} \right)$

The proof of this theorem is given in Appendix. Theorem 2 shows that the number of edges that link the nodes from the two groups defined in scheme 2 and scheme 4 are symmetric. That is the values remains the same if the two end nodes are switched. We can use  $m_{\overline{ab}}^{(2)} = \alpha \left( P_{a,b}^{(2)}, R_{a,b}^{(2)} \right)$  and  $m_{\overline{ab}}^{(4)} = \alpha \left( P_{a,b}^{(4)}, R_{a,b}^{(4)} \right)$  instead of Eq. 7.

**Theorem 3**  $\epsilon \left( P_{a,b}^{(4)}, R_{a,b}^{(4)} \right) = \epsilon \left( P_{b,a}^{(4)}, R_{b,a}^{(4)} \right)$

This theorem can be directly derived from  $P_{a,b}^{(4)} = R_{b,a}^{(4)}$ ,  $R_{a,b}^{(4)} = P_{b,a}^{(4)}$  and Theorem 1. So  $e_{\overline{ab}} = \epsilon \left( P_{a,b}^{(4)}, R_{a,b}^{(4)} \right)$ . Note scheme 4 is symmetric in calculating both of the actual and expected number of edges of the two groups.

#### Expected number of edges between two sets of nodes

With the four schemes described above, we get the number of edges that connect nodes from the two sets using Eq. 7. To calculate the authentic score of an edge by Eq. (6), we should find the expected number of edges between these two sets of nodes. Next we will use random graph generation models to determine the expected number of edges between these two sets of nodes.

**Erdős-Rényi random graph generation model**

The Erdős-Rényi model, often referred as  $G(n, m)$  model, is a basic random graph generation model [21]. It generates a graph of  $n$  nodes and  $m$  edges by randomly connecting two nodes by an edge and repeat this procedure until the graph contains  $m$  edges.

Suppose we have  $n$  nodes in an urn and predefined two sets of nodes  $S$  and  $T$ . We randomly pick two nodes from the urn. Note, the intersection of sets  $S$  and  $T$  may not be empty. The probability of picking the first node from set  $S \setminus T$  is  $\frac{|S| - |S \cap T|}{n}$  and the probability of picking the first node from set  $S \cap T$  is  $\frac{|S \cap T|}{n}$ . If the first node is from set  $S$ , the probability of picking the second node from set  $T$  is  $\frac{|S| - |S \cap T|}{n} \frac{|T|}{n-1} + \frac{|S \cap T|}{n} \frac{|T| - 1}{n-1}$ . Since the graph is undirected, we may also pick up a node from set  $T$  first and then pick up the second node from set  $S$ . So, the probability that we generate an edge that connects a node set  $S$  and a node set  $T$  by randomly picking is:

$$p(S, T) = (|S||T| - |S \cap T|) \frac{2}{n(n-1)}. \tag{9}$$

We repeat this procedure  $m$  times to generate a graph, where  $m$  is the number of edges in graph  $G$ . The expected number of edges that connect the nodes in set  $S$  and the nodes in set  $T$  is:

$$e(S, T) = (|S||T| - |S \cap T|) \frac{2m}{n(n-1)}. \tag{10}$$

Note, here we ignore the duplicate edges during this procedure. This has little impact on the final results for real-world graphs where  $m \ll n(n-1)$ . In Eq. (10), let

$$d_G = \frac{2m}{n(n-1)}, \tag{11}$$

where  $d_G$  is the density (or fill) of graph  $G$ .

Next we will find the expected number of edges under the four schemes defined in "Schemes of node neighborhood sets". Since edge  $ab$  is already fixed, we should repeat the random procedure  $m - 1$  times. For real-world graphs where  $m \gg 1$ , we can safely approximate  $m - 1$  by  $m$ .

Now we can apply Eq. (10) under the four schemes. Let  $k_a$  and  $k_b$  be the degrees of nodes  $a$  and  $b$ . Let  $k_{ab} = |N_a \cap N_b|$  be the number of common neighboring nodes of nodes  $a$  and  $b$ . The expected number of edges for each scheme is:

- Scheme 1:

$$e_{ab}^{(1)} = \left( k_a k_b - \frac{1}{2} (k_a + k_b) (1 + k_{ab}) + k_{ab} \right) d_G \tag{12}$$

- Scheme 2:

$$e_{ab}^{(2)} = \left( k_a k_b - \frac{1}{2} (k_a + k_b) - k_{ab} \right) d_G \tag{13}$$

- Scheme 3:

$$e_{ab}^{(3)} = \left( k_a k_b - \frac{1}{2} (k_a + k_b) k_{ab} \right) d_G \tag{14}$$

- Scheme 4:

$$e_{ab}^{(4)} = (k_a k_b - k_{ab}) d_G \tag{15}$$

**Preferential attachment random graph generation model**

The Erdős-Rényi model generates graphs that are lacking some important properties of real-world data, in particular the power law of the degree distribution [1]. Next we introduce a random graph generation model using a preferential attachment mechanism that generates a random graph in which degrees of each node are known. Our preferential attachment random graph generation model (PA model) is closely related to the modularity measurement that evaluates the community structure in a graph. Newman defines the modularity value as the difference of the actual number of edges and the expected number of edges of two communities [22]. The way of calculating the expected number of edges between two communities follows preferential attachment mechanism instead of using the Erdős-Rényi model. In the Erdős-Rényi model, each node is picked with the same probability. However, by the preferential attachment mechanism, the nodes with high degrees are picked with high probabilities. Thus an edge is more likely to link nodes with a high degree.

We can apply the preferential attachment strategy to generate a random graph with  $n$  nodes,  $m$  edges and each node has a predefined degree value. We first break each edge into two ends and put all the  $2m$  ends into an urn. A node with degree  $k$  will have  $k$  entities in the urn. At each round, we randomly pick two ends (one at a time with substitution) from the urn, link them with an edge and put them back into the urn. We repeat this procedure  $m$  times. We call this procedure Preferential Attachment Random Graph Generation model, or PA model in short. Note, we may generate duplicate edges or even self-loops with this procedure. Thus the expected number of edges estimated by this model is higher than a model that does not generate duplication edges and self-loops. This defect can be ignored when  $k_a$  and  $k_b$  are small. Later we will show a method that can compensate this bias, especially when  $k_a$  and  $k_b$  are large.

If we have two nodes  $a$  and  $b$ , the probability that an edge is formed in each round is:

$$p_{ab} = \frac{k_a k_b}{2m^2}. \tag{16}$$

Then the expected number of edges that link the nodes  $a$  and  $b$  after  $m$  iterations is:

$$e_{ab} = \frac{k_a k_b}{2m}. \tag{17}$$

If we have two sets of nodes  $S$  and  $T$ , the expected number of edges that link the nodes in set  $S$  and the nodes in set  $T$  is:

$$\epsilon(S, T) = \sum_{a \in S} \sum_{b \in T} e_{ab} = \frac{1}{2m} \sum_{a \in S} \sum_{b \in T} k_a k_b. \tag{18}$$

Applying Eq. (18) to the four schemes defined in "Schemes of node neighborhood sets", we get the expected number of edges for each scheme is

- Scheme 1:

$$e_{ab}^{(1)} = \frac{1}{4m} \left( \sum_{i \in P_{a,b}^{(1)}} \sum_{j \in R_{a,b}^{(1)}} k_i k_j + \sum_{i \in P_{b,a}^{(1)}} \sum_{j \in R_{b,a}^{(1)}} k_i k_j \right) \tag{19}$$

- Scheme 2:

$$e_{ab}^{(2)} = \frac{1}{4m} \left( \sum_{i \in P_{a,b}^{(2)}} \sum_{j \in R_{a,b}^{(2)}} k_i k_j + \sum_{i \in P_{b,a}^{(2)}} \sum_{j \in R_{b,a}^{(2)}} k_i k_j \right) \tag{20}$$

- Scheme 3:

$$e_{ab}^{(3)} = \frac{1}{4m} \left( \sum_{i \in P_{a,b}^{(3)}} \sum_{j \in R_{a,b}^{(3)}} k_i k_j + \sum_{i \in P_{b,a}^{(3)}} \sum_{j \in R_{b,a}^{(3)}} k_i k_j \right) \tag{21}$$

- Scheme 4:

$$e_{ab}^{(4)} = \frac{1}{4m} \left( \sum_{i \in P_{a,b}^{(4)}} \sum_{j \in R_{a,b}^{(4)}} k_i k_j + \sum_{i \in P_{b,a}^{(4)}} \sum_{j \in R_{b,a}^{(4)}} k_i k_j \right) \tag{22}$$

**Authentic score using the PA model**

**Authentic score compensation**

We may apply Eqs. (19), (20), (21) or (22) to Eq. (6) to calculate the authentic score of an edge. As mentioned in "[Preferential attachment random graph generation model](#)", the PA model generates graphs with duplicate edges and self-loops. Thus the estimated expected number of edges that link two sets of nodes are higher than an accurate model. The gap is even more significant when the number of edges is large. To compensate for this bias, we refine the authentic score function for the PA model as

$$s_{ab} = m_{ab}^\gamma - e_{ab} \tag{23}$$

where  $\gamma > 1$ . The power function of the first term increases the value, especially when  $m_{ab}$  is large. This eventually compensates the bias introduced in the second term. In practice, we normally choose  $\gamma = 2$ .

**Matrix of degree products**

To get  $e_{ab}$  using Eqs. (19), (20), (21) or (22), we should find the sum of  $k_a k_b$  for every pair of nodes in the corresponding edge-ego-network. We can store the values of  $k_a k_b$  for every pair of nodes to prevent unnecessary multiplication operations and thus reduce the processing time. However, storing this information would require a storage space in the order of  $n^2$ , which is not applicable when  $n$  is large. We observe that we do not need

to calculate the product of the degrees for every pair of nodes in graph  $G$ . What we need is the pair of nodes that appear together in every edge-ego-network.

The distance of two nodes in a graph is defined as the length of the shortest path between them. It is easy to see that the maximum distance of two nodes in an edge-ego-network is 3. Next, we use the property of the adjacency matrix to find the pairs of nodes that appear together in edge-ego-networks.

Let  $d_{ij}$  be the distance of node  $i$  and node  $j$ . Let  $B(k) = A^k$ , where  $A$  is the adjacency matrix of graph  $G$  and  $k$  is a natural number. Let  $B_{ij}(k)$  be the element of the matrix  $B(k)$ . Then  $B_{ij}(k)$  is the number of walks with length  $k$  between node  $i$  and node  $j$ . If  $B_{ij}(k) = 0$ , there is no walk with length  $k$  between nodes  $i$  and  $j$ .

**Proposition 3.1** *If  $d_{ij} = k$ ,  $B_{ij}(k) \neq 0$*

*Proof* If  $d_{ij} = k$ , there exists at least one path with length  $k$  from node  $i$  to node  $j$ . Since a path of a graph is a walk between two nodes without repeating nodes, there exists at least one walk with length  $k$  between the node  $i$  and the node  $j$ . So  $B_{ij}(k) \neq 0$ .

**Theorem 4** *Let  $K(k) = B(1) + B(2) + \dots + B(k)$ . If  $d_{ij} \leq k$ ,  $K_{ij}(k) \neq 0$*

*Proof* Let  $d_{ij} = l$ , where  $l \leq k$ . From Proposition 3.1,  $B_{ij}(l) \neq 0$ . Since  $B(k)$  is a nonnegative matrix where  $B_{ij}(k) \geq 0$ , we have  $K_{ij}(k) = B_{ij}(1) + \dots + B_{ij}(l) + \dots + B_{ij}(k) \neq 0$ .

According to Theorem 4, to find the pairs of nodes with a distance of 3 or less, we need to find the nonzero elements in matrix  $K(3)$ . Let  $I$  be the indicator matrix whose elements indicate whether the distance between a pair of nodes is equal to or less than 3. Such that:

$$I_{ij} = \begin{cases} 1 & \text{if } K_{ij}(3) \neq 0 \\ 0 & \text{if } K_{ij}(3) = 0 \end{cases} \quad (24)$$

Let matrix  $D$  denote the degree matrix whose diagonal elements are the degree of each node, that is:

$$D_{ij} = \begin{cases} k_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (25)$$

Let

$$E = \frac{1}{2m} \left( (DI) \circ (DI)^T \right), \quad (26)$$

where  $\circ$  denotes the Hadamard product of two matrices. The value of the nonzero elements in matrix  $E$  is the expected number of edges between the two nodes under the PA model. Using matrix  $E$ , we can easily calculate the authentic score for each scheme. For example the authentic score of the edge  $\overline{ab}$  using scheme 1 and the score function defined by Eq. (6) is:

$$s_{ab}^{(1)} = \frac{1}{2} \left( \sum_{i \in P_{a,b}^{(1)}} \sum_{j \in R_{a,b}^{(1)}} (A_{ij} - E_{ij}) + \sum_{i \in P_{b,a}^{(1)}} \sum_{j \in R_{b,a}^{(1)}} (A_{ij} - E_{ij}) \right). \quad (27)$$

### Evaluation of the proposed algorithms

In this section we evaluate the performance of the proposed outlier edge detection algorithms. Due to the availability of the datasets with identified outlier edges, we generate test data by injecting random edges to real-world graphs. This experimental setup is effective to evaluate algorithms that detect outliers, since the injected edges are random thus do not follow the actual principle that generated the real-world graph. We also evaluate the proposed outlier detection algorithms by measuring the change of some important graph properties when outlier edges are removed. In next section, we will show that the proposed algorithms are not only effective in simulated data but also powerful in solving real-world problems in many areas.

We first inject edges to a real-world graph data by randomly picking two nodes from the graph and linking them with an edge, if they are not linked. The injected edges are formed randomly, and thus they do not follow any underlying rule that generated the real-world graph. An outlier edge detection algorithm returns the authentic score of each edge. Given a threshold value, the edges with lower scores are classified as outliers.

With multiple algorithms, we vary the threshold value and record the true positive rates and the false positive rates of each algorithm. We use the receiver operating characteristic (ROC) curve—a plot of true positive rates against false positive rates at various threshold values—to subjectively compare the performance of different algorithms. We also calculate the area under the ROC curve (AUC) value to quantitatively evaluate the competing algorithms.

### Comparison of different combinations of the proposed algorithm

The proposed algorithm involves two random graph generation models and four schemes. Two authentic score functions are proposed for the PA Model. With the first experiment, we study the performance of different combinations using real-world graph data.

We take the Brightkite graph data as the test graph [23]. Brightkite is a social network service in which users share their location information with their friends. The Brightkite graph contains 58, 228 nodes and 214, 708 edges. The data was received from the KONECT graph data collection [24].

We injected 1000 random “false” edges to the graph data. If an algorithm yields the same authentic scores to multiple edges, we randomly order these edges. We compare the detection results of the algorithms using the Erdős-Rényi (ER) model and the PA model with the combination of the four schemes explained in “Schemes of node neighborhood sets” and the two score functions defined in Eqs. (6) and (23). Table 1 shows the AUC values of the ROC curves of all combinations. Italic font indicates the best score among all of them.

From the experimental results, we see that the performance of the PA model with score function defined by Eq. (23) is clearly better than that of the score function defined by Eq. (6). The term  $m^\gamma$  in Eq. (23) increases the value even more when  $m$  is large. After

**Table 1 AUC values of the ROC curves using Brightkite graph Data**

	ER model		PA model	
	Eq. (6)	Eq. (23)	Eq. (6)	Eq. (23)
Scheme 1	0.885	0.885	0.880	0.904
Scheme 2	0.885	0.885	0.882	<i>0.905</i>
Scheme 3	0.878	0.878	0.873	0.902
Scheme 4	0.879	0.879	0.878	0.903

*Italic indicates the best score of each experiment*

the bias of the PA model is corrected, the performance of the outlier edge detection algorithm is greatly improved. The choice of the score function defined by Eqs. 6 and 23 has little impact to the ER model based algorithms.

The results also show that the combination of the PA model and the score function defined by Eq. (23) is superior than other combinations by a significant margin. Scheme 2 gives better performance than the other schemes, especially for ER Model based algorithms. In the rest of this paper, we use scheme 2 for the ER Model based algorithm. With the combination of the PA Model and the score function defined by Eq. 23, the difference between each scheme is insignificant. Because of the symmetric property of scheme 4, we use it for the PA model with the score function defined by Eq. 23.

#### Comparison of outlier edge detection algorithms

In this section we perform comparative evaluation of the proposed outlier edge detection algorithms against other algorithms. All test graphs originate from the KONECT graph data collection. Table 2 shows some parameters of the test graph data. The density of a graph is defined in Eq. (11). GCC, which stands for the global clustering coefficient, is a measure of clustering property of a graph. It is the ratio of the number of closed triangles and the number of connected triplet nodes. The higher GCC value is, the stronger clustering property a graph has.

We compared the performance of the two proposed algorithms [ER model combined with scheme 2 and the score function defined by Eq. (6) and PA model combined with scheme 4 and the score function defined by Eq. (23)] with three other algorithms that use node similarity scores for missing edge detection. We use the Jaccard Index and Hub Promoted Index (HPI) as defined in Eqs. (3) and (4). We also use the preferential

**Table 2 Test graph data for comparing outlier edge detection Algorithms**

	Nodes	Edges	Density	GCC (%)	Reference
Advogato	6.5 k	51 k	$1.2 \times 10^{-3}$	9.2	[25]
Twitter-icwsm	465 k	835 k	$3.9 \times 10^{-6}$	0.06	[26]
Brightkite	58 k	214 k	$1.3 \times 10^{-4}$	11	[23]
Facebook-wosn	63 k	817 k	$4.0 \times 10^{-4}$	14.8	[27]
Ca-cit-HepPh	28 k	4.6 m	$8.0 \times 10^{-3}$	28	[28]
Youtube-friend	1.1 m	3.0 m	$4.6 \times 10^{-6}$	0.6	[29]
Web-Google	875 k	5.1 m	$6.7 \times 10^{-6}$	5.5	[30]

attachment index (PAI) that is another missing edge detection metric that works for outlier edge detection. The PAI for edge  $\overline{ab}$  is defined as

$$s_{PAI} = k_a k_b. \tag{28}$$

Figure 3 shows the ROC curves of different algorithms on the Brightkite graph data. For reference, the figure also shows an algorithm that randomly orders the edges by giving random scores to each edge.

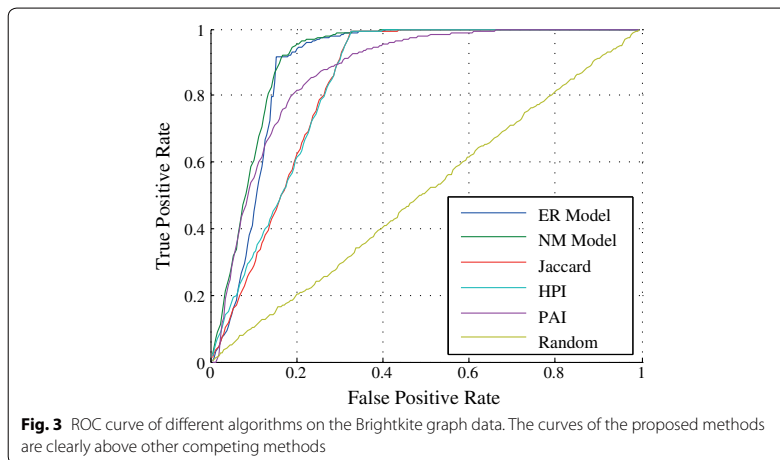
As Fig. 3 shows, the ROC curve of the algorithm that gives random scores is roughly a straight line from the origin to the top right corner. This line indicates that the algorithm cannot distinguish between an outlier edge and a normal edge, which is expected. The ROC curve of an algorithm that can detect outlier edges should be a curve above this straight line, as all algorithms used in this experiment. As mentioned in "Motivation", the Jaccard Index and HPI both use the number of common neighbors. Thus their scores are all 0 for edges that connect two end nodes that do not share any common neighbors. In real-world graphs, a large amount of edges have a Jaccard Index or HPI value 0, especially for graphs that contain many low degree nodes.

The PAI value is the product of the degrees of the two end nodes of an edge. Sorting edges with their PAI values just puts the edges with low degree end nodes to the front. The figure shows that the PAI value can detect outlier edges with fairly good performance. This indicates that most of the injected edges connecting the nodes with low degrees. Considering most of the nodes in a real-world graph are low degree nodes, this is an expected behavior.

Figure 3 indicates that the proposed outlier edge detection algorithms are clearly superior to the competing algorithms. The algorithm based on the PA model performs better than the one based on the ER model.

Table 3 shows the AUC values of the ROC curves on all test graph data. Italic font shows the best AUC values for each test graph.

The comparison results show that the PA model algorithm gives consistently good performance regardless of the test graph data. The experiment also shows the correlation



**Fig. 3** ROC curve of different algorithms on the Brightkite graph data. The curves of the proposed methods are clearly above other competing methods



**Table 3** AUC values of the ROC curves on different graph data

	ER	PA	Jaccard	HPI	PAI
Advogato	0.887	<i>0.893</i>	0.858	0.859	0.877
Twitter-icwsm	0.531	0.942	0.527	0.530	<i>0.997</i>
Brightkite	0.885	<i>0.905</i>	0.833	0.827	0.873
Facebook-wosn	0.968	<i>0.970</i>	0.947	0.946	0.878
Ca-cit-HepPh	0.970	0.967	<i>0.993</i>	0.991	0.888
Youtube-friend	0.770	0.842	0.731	0.738	<i>0.898</i>
Web-Google	0.985	<i>0.992</i>	0.944	0.945	0.859

*Italic indicates the best score of each experiment*

between the performance of the algorithms that are based on the random graph generation model and the GCC value of the test graph. For example, the ER model and PA model algorithms works better on Facebook-Wosn and Brightkite graph data, which have high GCC values as shown in Table 2. Performance of the ER model algorithm degrades considerably on graphs with a very low GCC value, such as the twitter-icwsm graph. This result agrees with the fact that both the ER model and the PA model algorithms use the clustering property of graphs. We also observe that PAI works better on graphs with low GCC values. We estimate that these graphs contain many star structures and two nodes with low degrees are rarely linked by an edge. The large number of claw count (28 billion) and small number of triangle count (38 k) in twitter-icwsm graph data partially confirm our estimation.

#### Change of graph properties

The proposed outlier edge detection algorithms are based on the clustering property of graphs. Since outlier edges are defined as edges that do not follow the clustering property, removing them should increase the coefficients that measure this property. On the other hand, some outlier edges (also called weak links in this aspect) serves an important role to connect remote nodes or nodes from different communities. Removing such edges should also extensively increase the distance of the two end nodes. Thus the coefficients that measure the distance between the nodes of a graph shall increase when outlier edges are removed. In this experiment, we verify these changes caused by the removal of the detected outlier edges.

The global clustering coefficient (GCC) and the average local clustering coefficient (ALCC) are the de facto measures of the clustering property of graphs. GCC is defined in "[Comparison of outlier edge detection algorithms](#)". Local clustering coefficient (LCC) is the ratio of the number of edges that connect neighboring nodes of a node and the number of all possible edges that connect these neighboring nodes. The LCC of node  $a$  can be expressed as

$$c_a = \frac{|\{\overline{ij} | i \in N_a, j \in N_a, \overline{ij} \in E\}|}{k_a(k_a - 1)}. \quad (29)$$

Average local clustering coefficient is the average of the local clustering coefficients of all nodes in the graph.

We use diameter, the 90-percentile effective diameter (ED) and the mean shortest path (MSP) length as distance measures between the nodes in a graph. Diameter is the

maximum shortest path length between any two nodes in a graph. 90-percentile effective diameter is the number of edges that are needed on average to reach 90% of other nodes. The mean shortest path length is the average of the shortest path length between each pair of nodes in the graph. Note, if the graph is not connected, we measure the diameter, ED and MSP of the largest component in the graph.

In this experiment, we removed 5% of the edges with the lowest authentic score. Table 4 shows the GCC, ALCC, Diameter, ED and MSP values before and after the outlier edges were removed. For comparison, we also calculated values of these coefficients after same amount of edges are randomly removed 5% from the graph.

The results show that removing the detected outlier edges clearly increases the GCC and ALCC values, while random edge removal slightly decreases the values. This confirms the enhancement of the clustering property after outlier edges are removed. The diameter, ED and MSP values all increase when the detected outlier edges were removed. This increase is much more significant than when random edges were removed. This also confirms the theoretical prediction.

## Applications

In this section, we demonstrate various applications that benefit from the proposed outlier edge detection algorithms. In these applications, we use the algorithm of the PA model combined with scheme 4 and the score function defined by Eq. 23.

### Impact on graph clustering algorithms

Graph clustering is an important task in graph mining [31–33]. It aims to find clusters in a graph- a group of nodes in which the number of inner links between the nodes inside the group is much higher than that between the nodes inside the group and those outside the group. Many techniques have been proposed to solve this problem [34–37].

The proposed outlier edge detection algorithms are based on the graph clustering property. They find edges that link the nodes in different clusters. These edges are also called weak links in the literature. With the proposed techniques, we can now remove detected outlier edges before applying a graph clustering algorithm. This should improve the graph clustering accuracy and reduce the computational time.

In this application, we evaluate the performance impact of the proposed outlier edge detection technique on different graph clustering algorithms. We use simulated graph data with cluster structures as used in [36, 38–40]. We generated test graphs of 512 nodes. The average degree of each node is 24. The generated cluster size varies from 16 to 256. Let  $d_{out}$  be the average number of edges that link a node from the cluster to

**Table 4** Graph properties changes after noise edges removal

	Original	ER model	PA model	Random
GCC	0.111	0.121	0.120	0.105
ALCC	0.172	0.180	0.183	0.158
Diameter	18	19	20	18
ED	5.91	6.78	6.36	5.95
MSP	3.92	4.10	4.10	3.95

nodes outside the cluster. Let  $d$  be the average degree of the node. Let  $\mu = \frac{d_{out}}{d}$  be the parameter that indicates the strength of the clustering structure. The smaller  $\mu$  is, the stronger the clustering structure is in the graph. We varied  $\mu$  from 0.2 to 0.5. Note, when  $\mu = 0.5$ , the graph has a very weak clustering structure, i.e. a node inside the cluster has an equal number of edges that link it to other nodes inside and outside the cluster.

We use the Normalized Mutual Information (NMI) to evaluate the accuracy of a graph clustering algorithm. The NMI value is between 0 and 1. The larger the NMI value is, the more accurate the graph clustering result is. An NMI value of 1 indicates that the clustering result matches the ground truth. More details of the NMI metric can be found in [35, 41].

We first apply graph clustering algorithms to the test graph data and record their NMI values and computational time. Then we remove 5% of the detected outlier edges from the test graph data, and apply these graph clustering algorithms again to the new graph and record their NMI values and computational time. The differences of the NMI values and the computational time show the impact of the outlier edge removal on the graph clustering algorithms.

The evaluated algorithms are LRW [42], GN [36], SLM [43], Danon [38], Louvain [34] and Infomap [44]. MCL [45] is not listed since it failed to find the cluster structure from this type of test graph data.

We repeated the experiment 10 times and calculated the average performance. Table 5 shows the NMI values before and after outlier edges were removed. The first number in each cell shows the NMI values of the clustering result on the original graph and the second number shows the NMI values of the clustering result on the graph after the outlier edges were removed.

Table 6 shows the NMI value changes in percentage. A positive value indicates that the NMI value has increased.

The results show that outlier edge removal improves the accuracy of most graph clustering algorithms. The clustering accuracy of the SLM algorithm and the Louvain algorithm decrease slightly in some cases.

Table 7 shows the computational time changes in percentage before and after outlier edges are removed. Negative values indicate that the computational time is decreased.

These results show that outlier edge removal decreases the computational time of most algorithms used in the experiment. In some cases, SLM and the Louvain algorithms show significant gains in computation time. Note further that the increase of the

**Table 5** The NMI values before and after outlier edges were removed

$\mu$	LRW	GN	SLM	Danon	Louvain	Infomap
0.2	1.0/1.0	0.99/1.0	1.0/1.0	0.99/1.0	1.0/1.0	1.0/1.0
0.25	0.97/1.0	0.98/0.99	1.0/1.0	0.99/0.98	1.0/1.0	1.0/1.0
0.3	0.89/0.95	0.93/0.97	1.0/1.0	0.95/0.98	1.0/1.0	0.92/1.0
0.35	0.78/0.82	0.74/0.72	0.96/0.94	0.66/0.84	0.90/0.86	0.36/0.91
0.4	0.80/0.86	0.66/0.70	0.83/0.81	0.67/0.70	0.84/0.81	0.78/0.83
0.45	0.25/0.73	0.53/0.52	0.71/0.67	0.51/0.55	0.68/0.60	0.22/0.43
0.5	0.03/0.61	0.39/0.47	0.58/0.56	0.39/0.49	0.51/0.53	0/0.47

**Table 6** Changes of normalized mutual information on graph clustering algorithms in percentage

$\mu$	LRW	GN (%)	SLM	Danon (%)	Louvain	Infomap
0.2	0	0.8	0	1.0	0	0
0.25	3.3%	1.5	0	-1.0	0	0
0.3	7.3%	5.0	0	3.5	0	9.1%
0.35	5.7%	-2.2	-2.1	26	-4.9%	155%
0.4	8.5%	6.7	-2.2	4.8	-3.0%	5.8%
0.45	190%	-1.1	-6.2	8.4	-12%	95%
0.5	1730%	19	-4.4	26	2.4%	$\infty$

**Table 7** Changes of computational time on graph clustering algorithms in percentage

$\mu$	LRW (%)	GN (%)	SLM (%)	Danon (%)	Louvain (%)	Infomap (%)
0.2	-52	-11	-36	-3.1	-33	-47
0.25	-23	-18	1.0	-1.0	-41	-16
0.3	-8.9	-9.3	7.7	-1.4	-31	-13
0.35	-11	-0.3	-21	-3.5	-35	31
0.4	-11	-5.7	-5.3	-3.0	-20	17
0.45	-16	2.8	-14.4	2.1	-41	33
0.5	-21	-6.7	-1.9	-3.4	-39	55

computational time in the Infomap algorithm leads to a crucial improvement of the clustering accuracy.

#### Outlier node detection in social network graphs

As mentioned in "Previous work", many algorithms have been proposed to detect outlier nodes in a graph. In this section we present a technique to detect outlier nodes using the proposed outlier edge detection algorithm.

In a social network service, if a user generates many links that do not follow the clustering property, we have good reasons to suspect that the user is a scammer. To detect this type of outlier nodes, we can first detect outlier edges. Then we find nodes that are the end points of these outlier edges. Nodes that are linked to many outlier edges are likely to be outlier nodes.

In this application, we use Brightkite data for outlier node detection. In the experiment, we rank the edges according to their authentic scores. We take the first 1000 edges as outlier edges and rank each node according to the number of outlier edges that it is connected to.

Table 8 shows the top 8 detected outlier nodes: the node ID, the number of outlier edges that the node links, the degree of the node, the rank of the degree among all nodes and LCC values of the node.

The results show that the detected outlier nodes tend to have large degree values. In particular, the LCC values of the detected outlier nodes are extremely low comparing to the ALCC value (0.172) of the graph. This shows that the neighboring nodes of the detected outlier nodes have very weak clustering property.

**Table 8** Outlier node detection results on Brightkite graph

Node id	Outlier edges	Degree	Degree rank	LCC
41	21	1134	1	0.005
458	16	1055	2	0.001
115	9	838	4	0.004
175	7	270	39	0.001
989	7	270	40	0.015
2443	7	379	16	0.010
36	5	467	11	0.005
158	5	833	5	0.004

### Clustering of noisy data

Clustering is one of the most important tasks in machine learning [46]. During the last decades, many algorithms have been proposed, i.e. [47–49]. The task becomes more challenging when noise is present in the data. Many algorithms, especially connectivity-based clustering algorithms, fail over such data. In this section we present a robust clustering algorithm that uses the proposed outlier edge detection techniques to find correct clusters in noisy data.

Graph algorithms have been successfully used in clustering problems [50, 51]. To cluster the data, we first build a mutual  $k$ -nearest neighbor (MKNN) graph [52, 53]. Let  $x_1, x_2, \dots, x_n \in R^d$  be the data points, where  $n$  is the number of data points and  $d$  is the dimension of the data. Let  $d(x_i, x_j)$  be the distance between two data points  $x_i$  and  $x_j$ . Let  $N_k(x_i)$  be the set of data points that are the  $k$ -nearest neighbors of the data point  $x_i$  with respect to the predefined distance measure  $d(x_i, x_j)$ . Therefore, the cardinality of the set  $N_k(x_i)$  is  $k$ . A MKNN graph is built in the following way. The nodes in the MKNN graph are the data points. Two nodes  $x_i$  and  $x_j$  are connected if  $x_i \in N_k(x_j)$  and  $x_j \in N_k(x_i)$ . The constructed MKNN graph is unweighted and undirected.

With a proper distance function, data points in a cluster are close to each other whereas data points in different clusters are far away from each other. Thus, in the constructed MKNN graph, a node is likely to be linked to other nodes in the same cluster while the links between the nodes in different clusters are relatively less. This indicates that the MKNN graph has the clustering property similar to social network graphs.

Outlier data points are normally far away from the normal data points. Some outlier nodes form isolated small components in the MKNN graph. However, the outlier nodes that fall between the clusters form bridges that connect different clusters. These bridges greatly degrade the performance of connectivity-based clustering algorithms, such as single-linkage clustering algorithm and complete-linkage clustering algorithm [46].

Based on these observations, we propose a hierarchical clustering algorithm by iteratively removing edges (weak links) according to their authentic scores. When a certain amount of outlier edges is removed, different clusters form separate large connected components—a connected component in a graph that contains a large proportion of the nodes, and it is straightforward to find them in the graph. A breadth-first search or a depth-first search algorithm can find all connected components in a graph with the complexity of  $O(n)$ , where  $n$  is the number of nodes. At each iteration step, we find large

connected components in the MKNN graph and the data points that do not belong to any large connected components are classified as outliers.

Using the proposed algorithm, we cluster a dataset taken from [54]. Figure 4 shows some results of different number of detected clusters. Outliers are shown in light gray color and data points in different clusters are shown in different colors.

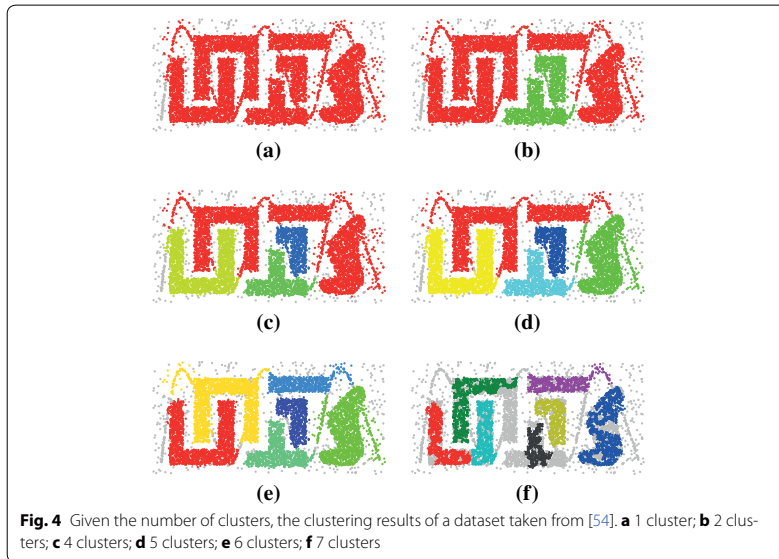
As the Fig. 4 shows, the proposed algorithm cannot only classify outliers and normal data points but also find clusters in the data points. As more and more edges are removed from the MKNN graph, the number of clusters increases.

Next we show how to determine the true number of clusters. Table 9 shows the number of removed edges and the number of detected clusters of this dataset.

As the result shows, removing a small amount of edges is enough to find correct clusters in the data. One has to remove a large amount of edges to break a genuine cluster into smaller components. We can simply define a threshold and stop the iteration if the number of clusters does not increase any more.

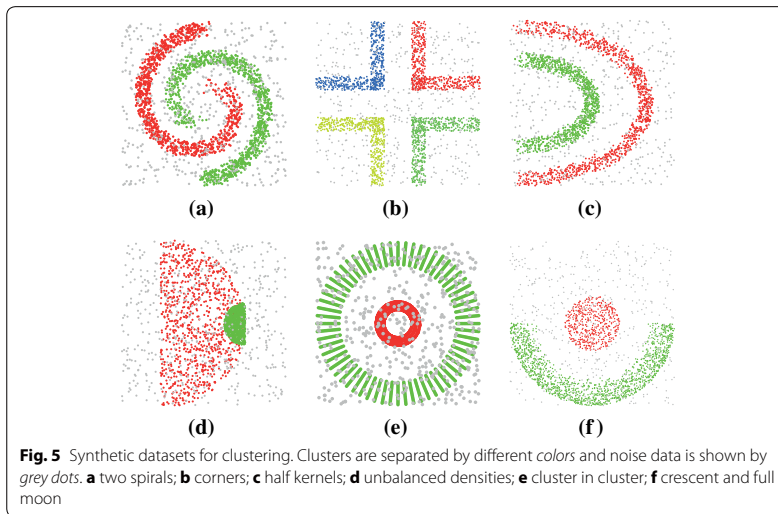
To illustrate the performance of the proposed clustering algorithm, we use synthetic data that are both noisy and challenging. Figure 5 shows the test datasets. We used tools from [55] to generate the normal data points and added random data points as noise.

In our experiments, we use the Euclidean distance function. The number of nearest neighbors is 30. At each iteration step, we remove 0.1% of total number of edges according to their authentic scores. A large connected component is a component whose size is



**Table 9** Percentage of the removed edges and the number of detected clusters

Removed edges	2.6%	2.7%	2.8%	3.5%	6%	33.3%
Number of clusters	2	3	4	5	6	7



larger than 5% of the total number of nodes. The clustering termination threshold is set as 10% of the total number of edges.

We compare the proposed clustering algorithm with the *k*-means [46], the average-linkage (a-link) [46], the normalized cuts (N-Cuts) [56] and the graph degree linkage (GDL) [49] clustering algorithms. Since the competing algorithms cannot detect the number of clusters, we use the value from the ground truth. Table 10 shows the NMI scores of the proposed algorithm and the competing algorithms.

The results show that the *k*-means and the average linkage clustering algorithms fail on complex-shaped clusters. GDL and the proposed algorithms are all graph-based clustering algorithms. They are able to find clusters with arbitrary shapes. From the NMI scores, the proposed algorithm is clearly superior to the competing clustering algorithms.

## Conclusions

In real-world graphs, in particular social network graphs, there are edges.

generated by scammers, malicious programs or mistakenly by normal users and the system. Detecting these outlier edges and removing them will not only improve the efficiency of graph mining and analytics, but also help identify harmful entities. In this article, we introduce outlier edge detection algorithms based on two random graph

**Table 10** Clustering of noisy data results

Dataset	k-Means	a-Link	N-Cuts	GDL	Proposed
(a)	0.031	0.099	0.053	0.650	<i>0.672</i>
(b)	0.743	0.743	0.743	0.743	<i>0.848</i>
(c)	0	0.004	0.559	0.654	<i>0.755</i>
(d)	0.208	0.161	0.367	0.553	<i>0.619</i>
(e)	0.001	0.133	0.680	0.701	<i>0.744</i>
(f)	0.001	0.162	0.627	0.612	<i>0.714</i>

*Italics* indicates the best score of each experiment

generation models. We define four schemes that represent relationships of two nodes and the groups of their neighboring nodes. We combine the schemes with the two random graph generation models and investigate the proposed algorithms theoretically. We tested the proposed outlier edge detection algorithms by experiments on real-world graphs. The experimental results show that our proposed algorithms can effectively identify the injected edges in real-world graphs. We compared the performance of our proposed algorithms with other outlier edge detection algorithms. The proposed algorithms, especially the algorithm based on the PA model, give consistently good results regardless of the test graph data. We also evaluated the changes of graph properties caused by the removal of the detected outlier edges. The experimental results show an increase in both the clustering coefficients and the increase of the distance between the nodes in the graph. This is coherent with the theoretical predictions.

Further more, we demonstrate the potential of the outlier edge detection using three different applications. When used with the graph clustering algorithms, removing outlier edges from the graph not only improves the clustering accuracy but also reduces the computational time. This indicates that the proposed algorithms are powerful preprocessing tools for graph mining. When used for detecting outlier nodes in social network graphs, we can successfully find outlier nodes whose behavior deviates dramatically from that of normal nodes. We also present a clustering algorithm that is based on the edge authentic scores. The clustering algorithm can efficiently find true data clusters by excluding noises from the data.

Outlier edge detection has great potentials in numerous Big Data applications. In the future, we will apply the proposed outlier edge detection algorithms in applications in other fields, for example computer vision and content-based multimedia retrieval in the Big Visual Data. We observed that nodes and edges outside edge-ego-network also contain valuable information in outlier detection. However, using this information dramatically increases the computational cost. We will work on fast algorithms that can efficiently use the structural information of the whole graph.

#### Abbreviations

ALCC: average local clustering coefficient; AUC: area under the curve; CN: common neighbors; ED: effective diameter; ER: Erdős-Rényi; GCC: global clustering coefficient; GDL: graph degree linkage; GN: Girvan-Newman; HDI: hub depressed index; HPI: hub promoted index; LCC: local clustering coefficient; LRW: limited random walk; MC: mean conductance; MCL: Markov Clustering Algorithm; MKNN: mutual k-nearest neighbor; MSP: mean shortest path; NMI: Normalized Mutual Information; N-Cuts: normalized cuts; PA: preferential attachment; PAI: preferential attachment index; ROC: receiver operating characteristic; SLM: smart local moving.

#### Authors' contributions

HZ carried out the conception and design of the study, participated in the analysis and interpretation of data, and was involved in drafting and revising the manuscript. SK and MG made substantial contributions to the design of the study, the analysis and interpretation of the data, and were involved in critically reviewing the manuscript. All authors read and approved the final manuscript.

#### Author details

<sup>1</sup> Department of Signal Processing, Tampere University of Technology, Finland, Korkeakoulunkatu 1, FI-33101 Tampere, Finland. <sup>2</sup> Electrical Engineering Department, College of Engineering, Qatar University, Qatar, 2713, Al Hala St, Doha, Qatar.

#### Competing interests

The authors declare that they have no competing interests.

#### Funding

Achademy of Finland supported this research.



**Appendix: Proof of Theorem 2**

**Proposition 6.1**  $\alpha(S \cup T, R) = \alpha(S, R) + \alpha(T, R)$  if  $S \cap T = \emptyset$ .

*Proof* Let  $A$  be the adjacency matrix of an unweighted and undirected graph  $G$ . We have  $\alpha(S, T) = \sum_{i \in S} \sum_{j \in T} A_{ij}$ . Given  $S \cap T = \emptyset$ ,

$$\begin{aligned} \alpha(S \cup T, R) &= \sum_{i \in S \cup T} \sum_{j \in R} A_{ij} \\ &= \sum_{i \in S} \sum_{j \in R} A_{ij} + \sum_{i \in T} \sum_{j \in R} A_{ij} \\ &= \alpha(S, R) + \alpha(T, R) \end{aligned}$$

Next we prove Theorem 2.

*Proof* For scheme 4,  $P_{a,b}^{(4)} = S_{a \setminus b}, R_{a,b}^{(4)} = S_{b \setminus a}, P_{b,a}^{(4)} = S_{b \setminus a}$  and  $R_{b,a}^{(4)} = S_{a \setminus b}$ . Using Theorem 1, we can easily get  $\alpha(P_{a,b}^{(4)}, R_{a,b}^{(4)}) = \alpha(P_{b,a}^{(4)}, R_{b,a}^{(4)})$ .

To prove Theorem 2 for scheme 2, we divide the nodes in edge-ego-network  $G_{\overline{ab}}$  into five mutually exclusive sets:

- $V_1 = \{x | x \in N_a \text{ and } x \notin S_b\}$ ;
- $V_2 = \{x | x \in N_b \text{ and } x \notin S_a\}$ ;
- $V_3 = \{x | x \in N_a \text{ and } x \in N_b\}$ ;
- $V_4 = \{a\}$ ;
- $V_5 = \{b\}$ .

From the definition, we have

$$\begin{aligned} P_{a,b}^{(2)} &= N_{a \setminus b} = V_1 \cup V_3, \\ R_{a,b}^{(2)} &= S_{b \setminus a} = V_2 \cup V_3 \cup V_5, \\ P_{b,a}^{(2)} &= N_{b \setminus a} = V_2 \cup V_3, \\ R_{b,a}^{(2)} &= S_{a \setminus b} = V_1 \cup V_3 \cup V_4. \end{aligned}$$

Using the definition of  $\alpha(S, T)$  and Proposition 6.1, we get

$$\begin{aligned} \alpha(P_{a,b}^{(2)}, R_{a,b}^{(2)}) &= \alpha(V_1 \cup V_3, V_2 \cup V_3 \cup V_5) \\ &= \alpha(V_1, V_2) + \alpha(V_1, V_3) + \alpha(V_1, V_5) \\ &\quad + \alpha(V_3, V_2) + \alpha(V_3, V_3) + \alpha(V_3, V_5) \end{aligned} \tag{30}$$

and

$$\begin{aligned} \alpha(P_{b,a}^{(2)}, R_{b,a}^{(2)}) &= \alpha(V_2 \cup V_3, V_1 \cup V_3 \cup V_4) \\ &= \alpha(V_2, V_1) + \alpha(V_2, V_3) + \alpha(V_2, V_4) \\ &\quad + \alpha(V_3, V_1) + \alpha(V_3, V_3) + \alpha(V_3, V_4) \end{aligned} \tag{31}$$

Taking the fact that  $\alpha(V_1 \cap V_5) = 0$ ,  $\alpha(V_2 \cap V_4) = 0$ , and  $\alpha(V_3, V_4) = \alpha(V_3, V_5)$ , the right hand side of Eqs. 30 and 31 are equal. Thus  $\alpha\left(P_{a,b}^{(2)}, R_{a,b}^{(2)}\right) = \alpha\left(P_{b,a}^{(2)}, R_{b,a}^{(2)}\right)$ .

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 8 February 2017 Accepted: 13 April 2017

Published online: 26 April 2017

## References

- Newman M. Networks: an introduction. 1st ed. New York: Oxford; 2010.
- Jiang M, Cui P, Beutel A, Faloutsos C, Yang S. CatchSync: catching synchronized behavior in large directed graphs. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '14. New York: ACM; 2014. p. 941–50.
- Beutel A, Xu W, Guruswami V, Palow C, Faloutsos C. CopyCatch: stopping group attacks by spotting lockstep behavior in social networks. In: Proceedings of the 22nd international conference on World Wide Web, 2013. p. 119–130.
- Yu R, Qiu H, Wen Z, Lin C, Liu Y. A survey on social media anomaly detection. SIGKDD Explor Newsletter. 2016;18(1):1–14.
- Akoglu L, Tong H, Koutra D. Graph based anomaly detection and description: a survey. Data Mining Knowl Discov. 2015;29(3):626–88.
- Noble CC, Cook DJ. Graph-based anomaly detection. In: Proceedings of the Ninth ACM SIGKDD international conference on knowledge discovery and data mining. KDD '03, Washington, D.C. New York: ACM; 2003. p. 631–636. doi:10.1145/956750.956831.
- Dai H, Zhu F, Lim EP, Pang H. Detecting anomalies in bipartite graphs with mutual dependency principles. In: 2012 IEEE 12th international conference on data mining (ICDM). IEEE; 2012. p. 171–80.
- Henderson K, Gallagher B, Eliassi-Rad T, Tong H, Basu S, Akoglu L, Koutra D, Faloutsos C, Li L. Rolx: structural role extraction & mining in large graphs. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining. ACM; 2012. p. 1231–9.
- Hodge VJ, Austin J. A survey of outlier detection methodologies. Artif Intell Rev. 2004;22(2):85–126.
- Xu X, Yuruk N, Feng Z, Schweiger TAJ. SCAN: a structural clustering algorithm for networks. Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '07. New York: ACM; 2007. p. 824–33.
- Gao J, Liang F, Fan W, Wang C, Sun Y, Han J. On community outliers and their efficient detection in information networks. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '10. New York: ACM; 2010. p. 813–22.
- Akoglu L, McGlohon M, Faloutsos C. oddball: spotting anomalies in weighted graphs. In: Zaki MJ, Yu JX, Ravindran B, Pudi V, eds. Advances in knowledge discovery and data mining. Lecture notes in computer science. 2010. pp. 410–421.
- Liu L, Zuo WL, Peng T. Detecting outlier pairs in complex network based on link structure and semantic relationship. Expert Syst Appl. 2017;69:40–9.
- Chakrabarti D. AutoPart: parameter-free graph partitioning and outlier detection. In: Boulicaut JF, Esposito F, Giannotti F, Pedreschi D, eds. Knowledge discovery in databases: PKDD 2004. Lecture notes in computer science. 2004. p. 112–24.
- Easley D, Kleinberg J. Networks, crowds, and markets: reasoning about a highly connected world. Cambridge University Press; 2010.
- Lu L, Zhou T. Link prediction in complex networks: a survey. Physica A Stat Mech Appl. 2011;390(6):1150–70.
- Barbieri N, Bonchi F, Manco G. Who to follow and why: link prediction with explanations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '14. New York: ACM; 2014. p. 1266–75.
- Freeman LC. Centered graphs and the structure of ego networks. Math Soc Sci. 1982;3(3):291–304.
- Watts DJ, Strogatz SH. Collective dynamics of 'small-world' networks. Nature. 1998;393(6684):440–2.
- Coscia M, Rossetti G, Giannotti F, Pedreschi D. DEMON: a local-first discovery method for overlapping communities. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '12. New York: ACM; 2012. p. 615–23.
- Bollobás B. Random graphs. 2 ed. Cambridge: New York; 2001.
- Newman MEJ. Modularity and community structure in networks. Proc Natl Acad Sci. 2006;103(23):8577–82.
- Cho E, Myers SA, Leskovec J. Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining. New York: ACM; 2011. p. 1082–90.
- Kunegis J. Konec: the koblenz network collection. In: Proceedings of the 22nd international conference on World Wide Web. New York: ACM; 2013. p. 1343–50.
- Massa P, Salvetti M, Tomasoni D. Bowling alone and trust decline in social network sites. In: IEEE International conference on dependable, autonomous and secure computing, 2009. DASC'09 Eighth. 2009. p. 658–63.
- De Choudhury M, Lin YR, Sundaram H, Candan KS, Xie L, Kelliher A. How does the data sampling strategy impact the discovery of information diffusion in social media? ICWSM. 2010;10:34–41.

27. Viswanath B, Mislove A, Cha M, Gummadi KP. On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM workshop on online social networks. New York: ACM; 2009. p. 37–42.
28. Leskovec J, Kleinberg J, Faloutsos C. Graph evolution: densification and shrinking diameters. *ACM Trans Knowl Discov Data*. 2007;1(1):2.
29. Yang J, Leskovec J. Defining and evaluating network communities based on ground-truth. *Knowl Inform Syst*. 2015;42(1):181–213.
30. Leskovec J, Lang KJ, Dasgupta A. Statistical properties of community structure in large social and information networks. In: Proceedings of the 17th international conference on World Wide Web. New York: ACM; 2008. pp. 695–704.
31. Fortunato S. Community detection in graphs. *Phys Rep*. 2010;486(3–5):75–174.
32. Coscia M, Giannotti F, Pedreschi D. A classification for community discovery methods in complex networks. *Stat Anal Data Min*. 2011;4(5):512–46.
33. Papadopoulos S, Kompatsiaris Y, Vakali A, Spyridonos P. Community detection in social media. *Data Min Knowl Discov*. 2011;24(3):515–54.
34. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. *J Stat Mech Theory Exp*. 2008;2008(10):10008.
35. Danon L, Diaz-Guilera A, Duch J, Arenas A. Comparing community structure identification. *J Stat Mech Theory Exp*. 2005;2005(09):09008.
36. Newman ME, Girvan M. Finding and evaluating community structure in networks. *Phys Rev E*. 2004;69(2):026113.
37. Schaeffer SE. Graph clustering. *Comput Sci Rev*. 2007;1(1):27–64.
38. Danon L, Diaz-Guilera A, Arenas A. The effect of size heterogeneity on community identification in complex networks. *J Stat Mech Theory Exp*. 2006;2006(11):11010.
39. Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. *Phys Rev E*. 2008;78(4):046110.
40. Newman ME. Fast algorithm for detecting community structure in networks. *Phys Rev E*. 2004;69(6):066133.
41. Ana LN, Jain AK. Robust data clustering. In: Proceedings 2003 IEEE computer society conference on computer vision and pattern recognition, 2003. vol. 2, p. 128–1332.
42. Zhang H, Raitoharju J, Kiranyaz S, Gabbouj M. Limited random walk algorithm for big graph data clustering. *J Big Data*. 2016;3(1):26.
43. Waltman L, Eck NJV. A smart local moving algorithm for large-scale modularity-based community detection. *The. Eur Phys J B*. 2013;86(11):1–14.
44. Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. *Proc Natl Acad Sci*. 2008;105(4):1118–23.
45. Dongen S. Graph clustering by flow simulation. PhD thesis, Utrecht: Universiteit Utrecht; 2000.
46. Theodoridis S, Koutroumbas K. Pattern recognition. 4 ed. Amsterdam; 2008.
47. Jain AK, Murty MN, Flynn PJ. Data clustering: a review. *ACM Comput Surv*. 1999;31(3):264–323.
48. Lloyd S. Least squares quantization in PCM. *IEEE Trans Inform Theory*. 1982;28(2):129–37.
49. Zhang W, Wang X, Zhao D, Tang X. Graph degree linkage: agglomerative clustering on a directed graph. In: Fitzgibbon A, Lazebnik S, Perona P, Sato Y, Schmid C, editors. Computer Vision - ECCV 2012. Lecture Notes in Computer Science. 2012. p. 428–41.
50. Harel D, Koren Y. On clustering using random walks. In: Hariharan R, Vinay V, Mukund M, editors. FST TCS 2001: Foundations of software technology and theoretical computer science. Lecture notes in computer science. 2001. pp. 18–41.
51. Dong X, Frossard P, Vandergheynst P, Nefedov N. Clustering with multi-layer graphs: a spectral perspective. *IEEE Trans Sign Process*. 2012;60(11):5820–31.
52. Brito MR, Chávez EL, Quiroz AJ, Yukich JE. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Stat Probab Lett*. 1997;35(1):33–42.
53. Ozaki K, Shimbo M, Komachi M, Matsumoto Y. Using the mutual k-nearest neighbor graphs for semi-supervised classification of natural language data. In: Proceedings of the fifteenth conference on computational natural language learning. 2011. p. 154–62.
54. Karypis G, Han E-H, Kumar V. Chameleon: hierarchical clustering using dynamic modeling. *Computer*. 1999;32(8):68–75.
55. 6 functions for generating artificial datasets - File Exchange - MATLAB Central. <http://se.mathworks.com/matlabcentral/fileexchange/41459>. Accessed 23 Feb 2017.
56. Shi J, Malik J. Normalized cuts and image segmentation. *IEEE Trans Pattern Anal Mach Intell*. 2000;22(8):888–905.



# PUBLICATION

## III

**A k-nearest neighbor multilabel ranking algorithm with application to  
content-based image retrieval**

H. Zhang, S. Kiranyaz and M. Gabbouj

*2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)2017,*  
2587–2591

DOI: 10.1109/ICASSP.2017.7952624

**Publication reprinted with the permission of the copyright holders**



# A K-NEAREST NEIGHBOR MULTILABEL RANKING ALGORITHM WITH APPLICATION TO CONTENT-BASED IMAGE RETRIEVAL

Honglei Zhang\*, Serkan Kiranyaz\*<sup>†</sup>, Moncef Gabbouj\*

\* Signal Processing  
Tampere University of Technology

<sup>†</sup> Electrical Engineering  
Qatar University

## ABSTRACT

Multilabel ranking is an important machine learning task with many applications, such as content-based image retrieval (CBIR). However, when the number of labels is large, traditional algorithms are either infeasible or show poor performance. In this paper, we propose a simple yet effective multilabel ranking algorithm that is based on k-nearest neighbor paradigm. The proposed algorithm ranks labels according to the probabilities of the label association using the neighboring samples around a query sample. Different from traditional approaches, we take only positive samples into consideration and determine the model parameters by directly optimizing ranking loss measures. We evaluated the proposed algorithm using four popular multilabel datasets. The proposed algorithm achieves equivalent or better performance than other instance-based learning algorithms. When applied to a CBIR system with a dataset of 1 million samples and over 190 thousand labels, which is much larger than any other multilabel datasets used earlier, the proposed algorithm clearly outperforms the competing algorithms.

**Index Terms**— Multilabel Learning, k-Nearest Neighbor, Content-Based Image Retrieval

## 1. INTRODUCTION

Multilabel ranking algorithms deal with the problems that each sample can be assigned to multiple labels [1, 2]. Labels used in multilabel learning are not mutually exclusive. This is different from the classes used in traditional multiclass classification where a sample can only be assigned to one class. Multilabel data is very common in many applications such as text categorization, bioinformatics and multimedia content retrieval. For example, an image may be labeled by keywords “cat”, “animal” and “funny”. Given a query sample, multilabel ranking algorithms give scores to each label and sort them from the most relevant to the least relevant.

Different approaches have been developed to solve the multilabel learning problems. Tsoumakas et al. categorized the algorithms into three groups: problem transformation methods, algorithm adaptation methods and ensemble methods [2]. The problem transformation methods take the binary

classification methods as basis and use either one-against-all or one-against-one strategy to get the classification results. The algorithm adaptation methods modify the existing binary classification algorithms to handle multiple labels. The ensemble algorithms apply a set of basic classifiers to subsets of samples and labels and the results are aggregated using sum, voting or other appropriate rules [3]. However, when the number of labels is large, previous algorithms are either infeasible or perform poorly.

In this paper, we propose an instance-based learning algorithm that can effectively handle the problem of the large number of labels. We compared the proposed algorithm with other instance-based multilabel ranking algorithms on four popular benchmark datasets. More importantly, we evaluated the performance of the proposed algorithm using a real-world content-based image retrieval (CBIR) system with a dataset of one million samples and over 190 thousand labels. To our best knowledge, this dataset is much larger than any other multilabel datasets used earlier [1, 2, 4].

## 2. BACKGROUND AND RELATED WORKS

### 2.1. Notations

Let  $L = \{L_1, L_2, \dots, L_q\}$  be the set of labels, where  $q$  is the number of labels. Let  $x_i \in R^d$ ,  $i = 1, 2, \dots, n$  be the feature vector of the samples, where  $R$  is the field of real numbers,  $d$  is the dimension of the feature vector and  $n$  is the number of samples. Let  $y_i = (y_i^{(1)}, y_i^{(2)}, \dots, y_i^{(q)}) \in \{0, 1\}^q$ ,  $i = 1, 2, \dots, n$  be the set of labels to which sample  $x_i$  is assigned, where  $y_i^{(l)} = 1$  if  $x_i$  is assigned to label  $L_l$ . We call  $Y_i = \{l | y_i^{(l)} = 1\}$  the relevant label set, and  $\bar{Y}_i = \{l | y_i^{(l)} = 0\}$  the irrelevant label set. Let  $T = \{(x_i, y_i) | i = 1, 2, \dots, m\}$  be the training set.

The score function for label  $L_k$  is defined as  $f_k(x) : R^d \rightarrow R$ ,  $k = 1, 2, \dots, q$ . The labels are ranked according to these scores such that  $rank(x, L_i) < rank(x, L_j)$  if  $f_i(x) > f_j(x)$ , where  $rank(L_i)$  is the rank of label  $L_i$ . We aim to learn a set of score functions  $\mathcal{F} = \{f_1, f_2, \dots, f_q\}$  that optimize a predefined objective function.

## 2.2. Evaluation measures

Different measures have been proposed to evaluate the performance of multilabel ranking algorithms.

Ranking loss evaluates the fraction of label pairs that have been ranked in a wrong order. The evaluation function is defined as

$$\text{ranking loss} = \frac{1}{n} \sum_{i=1}^n \frac{|D_i|}{|Y_i| |\bar{Y}_i|}, \quad (1)$$

where and  $D_i = \{(k, l) \mid f_k(x) > f_l(x), \text{rank}(L_k) > \text{rank}(L_l)\}$  is the set of labels pairs that have been ranked in a wrong order.

Average precision evaluates average fraction of the labels that are ranked above a true label that is actually in the relevant label set. The metric is defined as

$$\text{average precision} = \frac{1}{n} \sum_{i=1}^n \frac{1}{|Y_i|} \sum_{l \in Y_i} \frac{|B_{i,l}|}{\text{rank}(x_i, L_l)}, \quad (2)$$

where  $B_{i,l} = \{k \mid \text{rank}(x_i, L_k) > \text{rank}(x_i, L_l), k \in Y_i\}$ .

In this paper we also use one error and coverage as evaluation measures. Details can be found in [2].

Note, smaller values indicate better performance for all measures except average precision.

## 2.3. Related work

Binary Relevance (BR) methods apply one against all strategy and learn a binary classifier for each label [2]. For prediction, the binary classifiers for each label are applied independently. Read et al. further developed Classifier Chain (CC) [5] and Classifier Trellis (CT) [6] such that binary classifiers are connected by extending the feature space with the output of other classifiers. Label Power-set (LP) methods learn binary classifiers for sets of labels with different combinations. These methods can effectively deal with the correlation between labels [2]. However, the number of classifiers explodes as the number of labels increases. Most machine learning algorithms for binary classes have been adapted for the multilabel learning problems, for example ML-C4.5 [7], RFML-4.5 [8], and rank-SVM [9]. All of these algorithms learn certain number of classifiers or models from the training set. They show difficulties to handle the problems of large number of labels, large dataset or changes in the training set. For this reason, instance-based learning algorithms are more appropriate for some applications.

Zhang et al. developed the Multilabel  $k$ -NN (MLkNN) algorithm from the traditional  $k$ -nearest neighbor ( $k$ -NN) classification method [10]. MLkNN gather statistical information (the counts of the labels around a sample) for each label from the training set. For prediction, the maximum a posteriori (MAP) approach is applied to determine the set of labels. DMLkNN extends MLkNN method by using not only the statistical information from positive samples, but also negative

samples [11]. However neither MLkNN nor DMLkNN perform well when the number of labels is large, since there is insufficient training data to achieve reliable statistical information. Cheng and Hullermerier developed IBLR-ML method by combining linear regression and  $k$ -NN algorithms [12]. IBLR-ML method contains  $q$  classifiers, similar to BR methods, and thus suffers from the big label set problem too.

Spyromitros et al. combined BR methods with  $k$ -NN method (BRkNN) by using the count of label  $L_l$  in the set of neighbors as the confidence score [4]. MLC-WkNN improves BRkNN by giving weights to each sample according to its distance to the query sample [13]. The weights are the coefficients of a linear model learned by approximating the query sample from its  $k$  nearest neighbors. BR-kNN and MLC-WkNN can handle the large label problem. But their performance suffers due to the simple models they have applied.

## 3. K-NEAREST NEIGHBOR MULTILABEL RANKING ALGORITHM

### 3.1. Positive sample model

In our approach, we treat labels as the properties of a sample. The closer two samples are, the more likely they share same labels. In an extreme case, if two samples are identical, they will have the same set of labels.

Let  $i$  be the query sample,  $t$  be a sample that has label  $L_l$  and  $d(i, t)$  be the distance between sample  $i$  and  $t$ . Let  $E_i^{(l)}$  denote the event that sample  $i$  has label  $L_l$ . We model the probability of  $E_i^{(l)}$  as a function of the distance between sample  $i$  and  $t$ , and define it by

$$P \left( E_i^{(l)} \mid E_t^{(l)}, d(i, t) \right) = \exp(-z \cdot d(i, t)), \quad (3)$$

where  $z$  is a constant number. When  $d(i, t) = 0$  (sample  $i$  and  $t$  are identical), the probability of sample  $i$  to have label  $L_l$  is 1. In such a case, the prediction of label  $L_l$  is determined. Note, when  $d(i, t) \mapsto \infty$ , the probability function in Eq. 3 returns 0, which shall not be interpreted as the probability of  $E_i^{(l)}$  is 0. It actually indicates that sample  $t$  does not give any information to infer the association of label  $L_l$ .

Taking all positive samples into consideration, we make each of them contribute to the association of label  $L_l$ . A sample is not associated with label  $L_l$  if none of the positive sample is in favor of it. Thus we derive the probability of  $E_i^{(l)}$  given the training set  $T$  as

$$P \left( E_i^{(l)} \mid T \right) = 1 - \prod_{j \in T^{(l)}} (1 - \exp(-z \cdot d(i, t))), \quad (4)$$

where  $T$  is the training set and  $T^{(l)} = \{j \mid (x_j, y_j) \in T, l \in Y_j\}$ .

Because the samples located far from the query sample do little contribution to the probability function in Eq. 4, we can



apply the  $k$ -nearest neighbor paradigm. Let  $N_k(i)$  be the set of  $k$  neighboring samples, we have

$$P\left(E_i^{(l)}|N_k(i)\right) = 1 - \prod_{j \in N_k^{(l)}(i)} (1 - \exp(-z \cdot d(i, t))), \quad (5)$$

where  $N_k^{(l)}(i)$  is the set of samples that are associated with label  $L_l$  in  $N_k(i)$ , which is the set of  $k$  nearest neighbors of sample  $i$ .

Both our approach and IBLR-ML algorithm use exponential function to model the label association. The fundamental difference between our approach and IBLR-ML (and other similar algorithms) is the way we treat negative samples. IBLR-ML algorithm uses both positive and negative samples to learn binary classifiers or regressors; whereas in our approach, only positive samples contribute to the label association. When the number of labels is large, the training set can only be labeled loosely such that the relevant label set for a sample is incomplete even if the labels are accurate. Thus the models using both positive and negative samples often get confused because of the wrong negative samples in training set.

### 3.2. Model fitting

Since only positive samples are used for prediction, we cannot use maximum likelihood (ML), maximum a posteriori (MAP), or other classification techniques to fit the parameters. However, we can directly optimize any ranking measure defined in Section 2.2. In this paper, we choose the ranking loss as our objective function since it gives a complete evaluation of a ranking result.

Although an analytical solution is hard to find and the objective function is not convex, we can obtain a suboptimal solution using grid search or stochastic gradient descent method because of the model we incorporate. Subsampling techniques can be applied when the size of the training data is large.

Note that the parameter  $z$  acts like the smoothing parameter (bandwidth) used in kernel density estimation method. When  $z$  is small, a positive sample has a wide impact to the feature space around it; and when  $z$  is large, the impact of a positive sample is small. Considering the model in Eq. 4, when  $z \rightarrow 0$ , the proposed algorithm assign the labels according to their empirical prior probabilities. When  $z \rightarrow +\infty$ , the proposed algorithm is equivalent to BRkNN algorithm.

## 4. EXPERIMENTAL RESULTS ON BENCHMARK DATASETS

In this section, we evaluate the performance of the proposed algorithm on four popular benchmark datasets: Emotions, Scene, Yeast and Mediamill. The datasets are collected

from [14] and have been widely used to evaluate the performance of multilabel learning algorithms [1, 6, 10, 15]. Each of the datasets is divided into training and test set by the data providers [14].

We first show the impact of the parameter  $z$  using the Yeast dataset. We fix the number of neighbors to be 40 and vary  $z$  from 0.01 to 1. The values of the ranking loss against the parameter  $z$  are shown in Fig. 1.

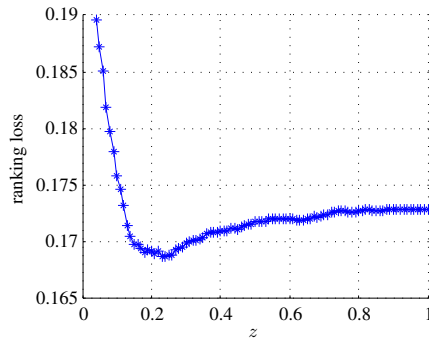


Fig. 1. Graph of ranking loss against the parameter  $z$

The result shows that even though the objective function of the proposed algorithm is not convex, the problem is not ill-posed. Using grid search or stochastic gradient descent method, a good solution can be found. The result also illustrates that the ranking loss converge to a certain value as  $z$  increases. According to Section 3.2, the proposed method converges to BRkNN algorithm.

Next, we compare the results of the proposed algorithm with other instance-based methods. Results of MLkNN, IBLR-ML, BRkNN and DMLkNN are obtained using Mulan multilabel learning toolbox [14] and parameter  $k$  is selected by optimizing the ranking loss on the test sets. The results are shown in Table 1, where r.l. stands for ranking loss, o.e. stands for one error, cov. stands for coverage and a.p. stands for average precision. The best values are marked using bold font.

Except the Scene dataset, the proposed method performs better than most of the other competing algorithms. Next, we will show the performance of the proposed algorithm in a real-world application with significant higher number of samples and labels.

## 5. APPLICATION TO CONTENT-BASED IMAGE RETRIEVAL

Given a query image, a CBIR system tries to find “similar” images in a large dataset and present the retrieved images in the order of relevance [16]. However, the concept of “similarity” is ambiguous since it totally depends on the user and the purpose of the query. A query result might be totally irrelevant if the purpose is not correctly justified. One aspect

**Table 1.** Comparison with other instance-based algorithms

	r.l.	o.e.	cov.	a.p.
Emotions				
MLkNN	<b>0.145</b>	<b>0.252</b>	1.787	0.818
IBLR-ML	<b>0.145</b>	0.262	<b>1.752</b>	<b>0.821</b>
BRkNN	0.150	0.262	1.792	0.818
DMLkNN	0.148	0.252	1.802	0.817
Ours	<b>0.145</b>	0.257	1.772	0.817
Scene				
MLkNN	0.082	0.252	0.512	0.852
IBLR-ML	<b>0.081</b>	<b>0.237</b>	<b>0.508</b>	<b>0.859</b>
BRkNN	0.101	0.278	0.607	0.826
DMLkNN	0.083	0.242	0.513	0.855
Ours	0.093	0.259	0.564	0.843
Yeast				
MLkNN	0.170	0.243	6.336	0.753
IBLR-ML	0.166	0.233	6.338	0.763
BRkNN	0.169	0.237	6.314	0.757
DMLkNN	0.169	0.249	6.371	0.757
Ours	<b>0.160</b>	<b>0.226</b>	<b>6.116</b>	<b>0.771</b>
Mediamill				
MLkNN	0.051	0.181	18.277	0.716
IBLR-ML	<b>0.050</b>	0.181	17.988	0.718
BRkNN	0.054	0.197	19.047	0.709
DMLkNN	<b>0.050</b>	0.178	17.924	0.719
Ours	0.051	<b>0.169</b>	<b>16.963</b>	<b>0.739</b>

of this problem has been coined as “semantic gap”, which describes the mismatching of the visual feature and the semantic intention of the query [17–19]. Since an image can be described by multiple labels and the purpose of the query is unknown, the CBIR system we used first rank the labels for the query image and then group the retrieved images accordingly. This requires a large number of labels that can describe an unknown image well and a large amount of images that have been labeled.

In this experiment, we converted the MSR-bing challenge dataset into a multilabel ranking dataset [20]. The VGG deep neural network is used to extract features [21].

Because neither the training set nor the test set are fully labeled, we are not able to use the measures defined in Section 2.2. We propose to use the mean inverse rank (MIR) as the evaluation metric. MIR is defined as

$$MIR = \frac{1}{n} \sum_{i=1}^n \frac{1}{|\tilde{Y}_i|} \sum_{l \in \tilde{Y}_i} \frac{1}{rank(x_i, L_l)}, \quad (6)$$

where  $\tilde{Y}_i$  is the label set given in the test set for sample  $x_i$ . Note,  $\tilde{Y}_i \subseteq Y_i$ , where  $Y_i$  is the set of ground truth labels. MIR is similar to average precision defined in Eq. 2. But it does not require the full set of ground truth labels. The larger a MIR value is, the better the algorithm performs.

Other than MIR, we also use success rate as a measure. A query is marked as success if the rank of a true label is less than a predefined value. Let  $SR@r$  denote a success rate at position  $r$ . Large success rate value indicates better performance of an algorithm.

In this experiment, we compared the proposed algorithm with BRkNN, BRkNN-w and MLkNN algorithms. BRkNN-w algorithm differs from standard BRkNN method by applying a weight to each sample in the nearest neighbor set. In our experiments, the inverse of the distance is used as the weight. For all algorithms,  $k$  is set to be 100. The experiment result is shown in Table 2.

**Table 2.** Experiment results on MSR-bing multilabel dataset

	BRkNN	BRkNN-w	MLkNN	Ours( $z=10$ )
MIR	0.0807	0.0930	0.0551	<b>0.123</b>
SR@5	0.103	0.117	0.0736	<b>0.159</b>
SR@10	0.152	0.176	0.1075	<b>0.216</b>
SR@100	0.327	0.328	0.2160	<b>0.333</b>

Table 2 shows that the proposed algorithm significantly outperforms the competing algorithms. The results of BRkNN-w and BRkNN method show that using the distance information between the query sample and the neighboring sample can clearly improve the performance. MLkNN method does not incorporate this information. Since of the number of labels is large, there is not enough samples to collect statistical information for each label. That also accounts for the poor performance of MLkNN algorithm.

## 6. CONCLUSIONS

In this paper, we have proposed a simple yet effective instance-based multilabel ranking algorithm to tackle the problem of the large number of labels for content-base image retrieval in large scale. We treat labels as the properties of samples and model the probability of having a certain property as an exponential function of the distance. Taking all the positive samples around a query sample into consideration, we calculate the probability of not having the property using a product rule. Unlike traditional methods, we use only positive samples in our method. For this reason, we choose to optimize the ranking loss function directly. Because of the simplicity of our model, grid search or stochastic gradient descent method can effectively find a suboptimal solution.

We compared the performance of the proposed algorithm with other instance-based algorithms on four benchmark datasets. The proposed algorithm is either the best or close to the best on three datasets. We used the proposed algorithm in a CBIR system with the MSR-bing challenge multilabel dataset, which contains 1M samples and over 190 thousand labels. The proposed algorithm clearly outperforms other methods by a significant margin.

## 7. REFERENCES

- [1] E. Gibaja and S. Ventura, "A Tutorial on Multilabel Learning," *ACM Comput. Surv.*, vol. 47, no. 3, pp. 52:1–52:38, Apr. 2015.
- [2] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.
- [3] J. Read, L. Martino, and D. Luengo, "Efficient monte carlo methods for multi-dimensional learning with classifier chains," *Pattern Recognition*, vol. 47, no. 3, pp. 1535–1546, Mar. 2014.
- [4] E. Spyromitros, G. Tsoumakas, and I. Vlahavas, "An empirical study of lazy multilabel classification algorithms," in *Hellenic conference on Artificial Intelligence*. Springer, 2008, pp. 401–406.
- [5] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, no. 3, pp. 333–359, Jun. 2011.
- [6] J. Read, L. Martino, P. M. Olmos, and D. Luengo, "Scalable multi-output label prediction: From classifier chains to classifier trellises," *Pattern Recognition*, vol. 48, no. 6, pp. 2096–2109, Jun. 2015.
- [7] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2001, pp. 42–53.
- [8] D. Kocev, C. Vens, J. Struyf, and S. Džeroski, "Ensembles of multi-objective decision trees," in *European Conference on Machine Learning*. Springer, 2007, pp. 624–631.
- [9] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Advances in neural information processing systems*, 2001, pp. 681–687.
- [10] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, Jul. 2007.
- [11] Z. Younes, F. Abdallah, T. Denoeux, and H. Snoussi, "A dependent multilabel classification method derived from the k-nearest neighbor rule," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 1, pp. 1–14, 2011.
- [12] W. Cheng and E. Hüllermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Machine Learning*, vol. 76, no. 2-3, pp. 211–225, 2009.
- [13] J. Xu, "Multi-Label Weighted k-Nearest Neighbor Classifier with Adaptive Weight Estimation," in *Neural Information Processing*, ser. Lecture Notes in Computer Science, B.-L. Lu, L. Zhang, and J. Kwok, Eds. Springer Berlin Heidelberg, Nov. 2011, no. 7063, pp. 79–88.
- [14] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2411–2414, 2011.
- [15] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognition*, vol. 45, no. 9, pp. 3084–3104, Sep. 2012.
- [16] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, Dec. 2000.
- [17] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, Jan. 2007.
- [18] M. K. Kundu, M. Chowdhury, and S. Rota Bulò, "A graph-based relevance feedback mechanism in content-based image retrieval," *Knowledge-Based Systems*, vol. 73, pp. 254–264, Jan. 2015.
- [19] B. Xu, J. Bu, C. Chen, C. Wang, D. Cai, and X. He, "EMR: A Scalable Graph-Based Ranking Model for Content-Based Image Retrieval," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 1, pp. 102–114, Jan. 2015.
- [20] X.-S. Hua, M. Ye, and J. Li, "Mining knowledge from clicks: MSR-Bing image retrieval challenge," in *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–4.
- [21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

# PUBLICATION

## IV

### **Data Clustering Based on Community Structure in Mutual k-Nearest Neighbor Graph**

H. Zhang, S. Kiranyaz and M. Gabbouj

*2018 41st International Conference on Telecommunications and Signal Processing (TSP)2018,*

1-7

DOI: 10.1109/TSP.2018.8441226

**Publication reprinted with the permission of the copyright holders**

# Data Clustering Based on Community Structure in Mutual $k$ -Nearest Neighbor Graph

Honglei Zhang\*, Serkan Kiranyaz<sup>†</sup>, and Moncef Gabbouj\*

\*Department of Signal Processing, Tampere University of Technology, Tampere, Finland

<sup>†</sup>Electrical Engineering Department, Qatar University, Doha, Qatar

**Abstract**—Data clustering is a fundamental machine learning problem. Community structure is common in social and biological networks. In this article we propose a novel data clustering algorithm that uses this phenomenon in mutual  $k$ -nearest neighbor (MKNN) graph constructed from the input dataset. We use the authentic scores—a metric that measures the strength of an edge in a social network graph—to rank all the edges in the MKNN graph. By removing the edges gradually in the order of their authentic scores, we collapse the MKNN graph into components to find the clusters. The proposed method has two major advantages comparing to other popular data clustering algorithms. First, it is robust to the noise in the data. Second, it finds clusters of arbitrary shape. We evaluated our algorithm on synthetic noisy datasets, synthetic 2D datasets and real-world image datasets. Results on the noisy datasets show that the proposed algorithm clearly outperforms the competing algorithms in terms of Normalized Mutual Information (NMI) scores. The proposed algorithm is the only one that does not fail on any data in the the synthetic 2D dataset, which are specifically designed to show the limitations of the clustering algorithms. When testing on the real-world image datasets, the best NMI scores achieved by the proposed algorithm is more than any other competing algorithm. The proposed algorithm has computational complexity of  $O(k^3n + kn \log(kn))$  and space complexity of  $O(kn)$ , which is better than or equivalent to the most popular clustering algorithms.

**Keywords**—data clustering, authentic score, graph

## I. INTRODUCTION

Data clustering is a fundamental machine learning problem [1]. Given a set of data points, a clustering algorithm groups the data points according to certain similarity measurements and clustering criteria. Over the last several decades, numerous clustering methods have been proposed, such as: centroid-based algorithms like K-means and K-medians [1]; connectivity-based algorithms like single-linkage and average-linkage [1]; and density-based algorithms like DBSCAN [2]. Each algorithm has its advantages and limitations. One has to choose an appropriate method based on the input datasets and the requirements of a particular application.

Graph-based method is an important approach for data clustering [3], [4], [5], [6], [7]. These algorithms construct a graph from the given data points and take advantage of graph theories and technologies to partition the graph into components. Hu and Bhatnagar proposed an agglomerative algorithm using the constructed mutual  $K$ -nearest neighbor (MKNN) graph from the data points [8]. Their algorithm finds initial clusters from dense areas in the graph, and then merges these clusters according to the connectivity measurement until

predefined criteria is reached. Zhang et al. studied directed  $K$ -nearest neighbor (DKNN) graph and presented an effective affinity measurement using the product of the average in-degrees and the average out-degrees [6]. They applied a greedy approach to minimize the affinity measurement and partition the DKNN graph. Zhang et al. studied another agglomerative method that merge clusters based on the incremental path integral—an affinity measurement that measures the stability of a dynamic system built from a random walk model [7].

Aforementioned graph-based data clustering methods are agglomerative and focus on the affinities of nodes and clusters. In this paper, we present a novel divisive approach using MKNN graph. Our paradigm is different from traditional clustering methods: instead of grouping similar data points into clusters, the algorithm tries to find appropriate boundaries to split the data points. Our proposed method uses the common phenomenon in social and biological networks that a community has dense connections among the nodes inside but sparse connections to the nodes outside [9]. Zhang et al. studied this phenomenon and introduced the authentic score of an edge based on random graph generation models [10]. Authentic score measures the strength of an edge from the neighboring nodes around it. To find the cores of the clusters, the proposed algorithm collapses a weighted MKNN by gradually removing edges in the ascending order of their authentic scores.

## II. BACKGROUNDS

### A. Mutual $k$ -Nearest Neighbor Graph

Let  $X = \{x_1, x_2, \dots, x_n\}$  be the set of input data points, where  $x_i \in \mathcal{R}^d$  is a  $d$ -dimensional vector and  $n$  is the number of data. Let  $d(x_i, x_j)$  be the distance between data points  $x_i$  and  $x_j$ . In this paper, we assume that Euclidean distance is used. Let  $N_k(x_i)$  be the set of the  $k$  nearest neighbors of data point  $x_i$ . Obviously  $|N_k(x_i)| = k$ . Let  $G(V, E)$  be the graph constructed from the input dataset, where each node is a data point from  $X$ ,  $V$  is the set of nodes and  $E$  is the set of edges. We connect node  $x_i$  and  $x_j$  with the edge  $\overline{x_i x_j}$  if both of the two end data points are in the set of the  $K$ -nearest data points of the other one. That is:  $\overline{x_i x_j} \in E$  if  $x_j \in N_k(x_i)$  and  $x_i \in N_k(x_j)$ . The graph  $G(V, E)$  is called mutual  $k$ -nearest neighbor (MKNN) graph of dataset  $X$ . Let  $A$  be the adjacency matrix of graph  $G(V, E)$ . Let  $m = |E|$  be the number of edges of the graph. Note that MKNN graphs are undirected.

A component of a graph is a subgraph that every pairs of the nodes in the subgraph are linked by at least one path, but

no path links any node in the subgraph to a node outside of the subgraph.

### B. The Linking Strength of Edges

Zhang et al. studied outlier edges in social networks [10] and pointed that an edge is likely to be an outlier if the neighboring nodes of the two end nodes do not form a dense cluster. For example, in Fig. 1, the edge  $\overline{x_i x_j}$  in (b) is more authentic than  $\overline{x_i x_j}$  in (a) since there are more connections of the neighboring nodes of the two end nodes  $x_i$  and  $x_j$  in (b).

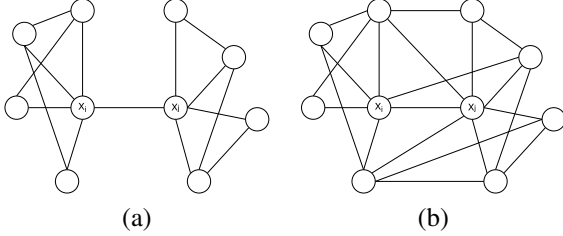


Fig. 1. Strength of the edges in a graph

Using random graph generation models, Zhang et al. define the authentic score  $s_{\overline{x_i x_j}}$  of edge  $\overline{x_i x_j}$  as

$$s_{\overline{x_i x_j}} = r_{\overline{x_i x_j}} - e_{\overline{x_i x_j}}, \quad (1)$$

where  $r_{\overline{x_i x_j}}$  is the actual number of edges that link the two sets of the neighboring nodes of node  $x_i$  and  $x_j$ , and  $e_{\overline{x_i x_j}}$  is the expected number of edges that link these two sets of nodes. The lower the authentic score is, the more likely an edge is an outlier. Their research shows that better performance can be achieved using the Preferential Attachment random graph generation model with the authentic score function:

$$s_{\overline{x_i x_j}} = r_{\overline{x_i x_j}}^2 - e_{\overline{x_i x_j}}. \quad (2)$$

Let  $N(x_i)$  be the set of neighboring nodes of node  $x_i$ . Let  $S_{x_i} = \{x_i\} \cup N(x_i) \setminus \{x_j\}$ , and  $S_{x_j} = \{x_j\} \cup N(x_j) \setminus \{x_i\}$ . The actual number of edges that link  $S_{x_i}$  and  $S_{x_j}$  is

$$r_{\overline{x_i x_j}} = \sum_{a \in S_{x_i}, b \in S_{x_j}} A_{ab}. \quad (3)$$

Using the Preferential Attachment model, the expected number of edges that link  $S_{x_i}$  and  $S_{x_j}$  is

$$e_{\overline{x_i x_j}} = \frac{1}{2m} \sum_{a \in S_{x_i}, b \in S_{x_j}} k_a k_b, \quad (4)$$

where  $k_a$  and  $k_b$  are the degrees of node  $a$  and  $b$ .

The authentic scores of the edges that link the nodes in the same cluster are higher than the scores of the edges that link the nodes from different clusters. Zhang et al. demonstrated an effective clustering algorithm based on outlier edge detection [10]. After constructed a MKNN graph, they remove edges one by one in the ascending order of their authentic scores. At each step, the number of components is detected. It would require a large portion of edges to be removed to break a genuine cluster into smaller pieces. Thus a threshold can be defined to find the right split of the MKNN graph.

This method is robust to noise and also insensitive to the variation of the data density of clusters. But, it can not handle the data in which the sizes of clusters vary significantly. A large cluster would normally dominate the process and lead to wrong split being detected. Another disadvantage is that the generated MKNN does not incorporate the distance of the nodes. This may lead to inaccurate clustering results. Next, we present a novel clustering method that addresses these limitations.

## III. METHODOLOGY

### A. Weighted MKNN Graph

We first construct a MKNN graph from the dataset as defined in Section II-A. To facilitate the distance between the data points, we first convert the distance values into affinity values and assign the affinity values as weights to the edges of the MKNN graph.

An major advantage using MKNN graphs is that the algorithm is insensitive to the density of the data points. To retain this advantage, we use a linear function instead of the popular exponential affinity conversion function [6], [7]. Recall that the distance of two data points  $x_i$  and  $x_j$  is  $d(x_i, x_j)$ , our affinity function for data points  $x_i$  and  $x_j$  is defined as

$$a(x_i, x_j) = C - d(x_i, x_j), \quad (5)$$

where  $C$  is a constant that is large enough to make the affinity values nonnegative. In practice, we simply choose  $C = \max_{x_i, x_j \in X} d(x_i, x_j)$ , which is the largest distance of the data points in the input dataset.

Assigning the affinity values as weights to the edges in the MKNN graph, we get the weighted MKNN graph.

### B. Authentic Scores of the Edges in a Weighted MKNN Graph

Let  $A$  be the adjacency matrix of the weighted MKNN graph, where the elements of  $A$  are the affinity values of edges. So,  $A_{x_i x_j} = a(x_i, x_j)$  if node  $x_i$  and  $x_j$  are linked by an edge; otherwise  $A_{x_i x_j} = 0$ . The actual number of edges that link the nodes in sets  $S_{x_i}$  and  $S_{x_j}$  can be calculated by Eq. 3. To calculate the expected number of edges that link the nodes in sets  $S_{x_i}$  and  $S_{x_j}$  using Eq. 4, we need the degrees of each node. For a weighted MKNN graph, we use the generalized degree of a node:

$$k_a = \sum_{b \in N(a)} A_{ab}. \quad (6)$$

Using Eq. 6 and the adjacency matrix  $A$ , we are able get the authentic scores for each edge by Eqs. 2, 3 and 4.

### C. Partitions of a Weighted MKNN Graph

In a MKNN graph, data points from a cluster form a dense structure like a community in social networks, where the edges of the nodes inside the community are stronger than those that links to the nodes outside. To find these communities in a MKNN graph, we gradually remove edges according to their authentic scores where edges with low authentic scores are removed first. When certain number of the edges are removed,

the MKNN graph splits into large components since the links between genuine clusters are removed first. During this graph collapsing procedure, small components or isolated nodes may also appear. They are mostly border nodes or noisy data. As more and more edges are removed, large components split into smaller ones. When all the edges are removed, the graph eventually becomes  $n$  isolated nodes. Zhang et al. noticed that a large number of edges have to be removed to break a real community into pieces [10]. Instead, we can cluster the input data points by analyzing how the MKNN graph breaks.

Let  $\mathcal{P} = \{P_1, P_2, \dots, P_q\}$  be a partition of graph  $G(V, E)$ , where  $q$  is the number of components,  $P_i$  is the set of nodes in the  $i$ -th component. We sort  $P_i$  in the descending order by their cardinalities. Let  $s_i = |P_i|$  be the cardinality of set  $P_i$ . We have  $i < j$  if  $s_i > s_j$ . Obviously  $\cup_{i=1}^q P_i = V$ .

We use superscript to denote the step in this iterating procedure. At step  $t$ , the partition of the MKNN graph is  $\mathcal{P}^{(t)}$  and the corresponding sizes of each component is  $S^{(t)} = [s_1^{(t)}, s_2^{(t)}, \dots, s_{q^{(t)}}^{(t)}]$ , where  $[\cdot]$  denotes a list. During the collapsing procedure, we get a list of partitions of the MKNN graph. When  $t = 0$ , we have the original MKNN graph, that is  $S^{(0)} = [n]$ . When  $t = m$ , we have  $S^{(m)} = [1, 1, \dots, 1]$ .

If we can find an optimal partition of the MKNN graph, the large components in this partition are the core of the data clusters. We will give details of finding the optimal partition of the MKNN graph in Section III-D. Let  $\mathcal{P}^{(o)}$  be the optimal partition, and  $r$  be the number of clusters. Besides the top  $r$  largest components in  $\mathcal{P}^{(o)}$ , we may have small components  $P_{r+1}^{(o)}, P_{r+2}^{(o)}, \dots, P_{q^{(o)}}^{(o)}$ . To achieve a partition of the whole dataset, we simply add the removed edges back in the reversed order and assign a small component to the first large component that it links in this reuniting procedure.

Algorithm 1 shows the details of the proposed algorithm when the number of cluster is given.

#### D. Optimal Partition Determination

Given the number of clusters  $r$  and a list of partitions  $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \dots, \mathcal{P}^{(m)}$ , we will show two methods to determine the optimal partition. Note that  $\mathcal{P}^{(k)} = \{P_1^{(k)}, P_2^{(k)}, \dots, P_{q^{(k)}}^{(k)}\}$ , where  $P_i^{(k)}$  is the set of nodes of the  $i$ -th component,  $s_i^{(k)}$  is the cardinality of set  $P_i^{(k)}$ , and  $i < j$  if  $s_i > s_j$ .  $P_1^{(k)}, P_2^{(k)}, \dots, P_r^{(k)}$  are the cores of the data clusters.

1) *Maximize the Minimal Cluster Size*: If the sizes of clusters are balanced, the sizes of the cores should be close to each other. When a core breaks during the collapsing procedure, it generates much smaller components. With this observation, we can simply choose the partition that maximizes the minimal size of the core among all partitions. We call this method Max-Min method. Note that  $s_1^{(k)}, s_2^{(k)}, \dots, s_{q^{(k)}}^{(k)}$  are sorted in descending order,  $s_r^{(k)}$  is the smallest core component. Given  $s_r^{(1)}, s_r^{(2)}, \dots, s_r^{(m)}$ , the optimal partition is expressed as

$$o = \arg \max_{k=1, \dots, m} (s_r^{(k)}). \quad (7)$$

Note, if  $q^{(k)} < r$ , we set  $s_r^{(k)} = 0$ .

2) *Minimize the Maximal Conductance Value*: The Max-Min method described in the previous section is fast and efficient when the clusters are balanced. However, if this assumption does not hold, Max-Min method does not guarantee to find a good solution. Without any prior knowledge, we can determine the optimal partition by evaluating the conductance values of the candidate partitions.

The conductance of a cluster  $C$  is defined as

$$\varphi(C) = \frac{a(C, \bar{C})}{\min(a(C), a(\bar{C}))}, \quad (8)$$

where  $\bar{C} = V \setminus C$  is the complement of cluster  $C$ ,  $a(C) = \sum_{i \in C} \sum_{j \in V} A_{ij}$  is the weight of cluster  $C$ , and  $a(C, \bar{C}) = \sum_{i \in C} \sum_{j \in \bar{C}} A_{ij}$  is the weight of the cut of  $C$  and  $\bar{C}$ . A small conductance value indicates that the cluster is well separated from the rest of the data points.

Given a partition  $\mathcal{P}^{(k)} = (P_1^{(k)}, P_2^{(k)}, \dots, P_{q^{(k)}}^{(k)})$ , we first merge small components  $P_{r+1}^{(k)}, P_{r+2}^{(k)}, \dots, P_{q^{(k)}}^{(k)}$  to the candidate cores  $P_1^{(k)}, P_2^{(k)}, \dots, P_r^{(k)}$  by the reuniting procedure described in Section III-C. This generates  $r$  clusters  $C_1^{(k)}, C_2^{(k)}, \dots, C_r^{(k)}$  for each  $k$ . The optimal partition is the one that minimize the maximal conductance value of the candidate partitions. That is

$$o = \arg \min_{k=1, \dots, m} \left( \max_{j=1, \dots, r} \left( \varphi(C_j^{(k)}) \right) \right). \quad (9)$$

We set  $\varphi(C_r^{(k)}) = +\infty$  if  $q^{(k)} < r$ . This eliminates the partitions that the number of big components is less than  $r$ . We call this method Min-Max method. This method generates better clustering result with the cost of increasing computational complexity. Later we will describe the batch mode of the proposed algorithm that greatly reduces the computational complexity.

#### E. Noise Detection

Outliers are data points that lie far from the normal data points. Normally, an outlier node is either isolated or weakly connected to the normal data nodes in a MKNN graph. Since two normal data points in a cluster often share many common neighbors, the edge that links these nodes has high authentic score. During the graph collapsing procedure, edges that link different clusters are removed first. Then the edges that link outliers to normal data points are removed. The edges that link normal data points inside a cluster are removed last. If the normal data clusters have clear borders, the authentic scores of the edges inside a cluster is much larger than the scores of the edges that link an outlier and normal data points. We use this to distinguish outliers from normal data points.

Let the authentic scores of the last removed edge for the partitions  $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \dots, \mathcal{P}^{(m)}$  be  $c^{(1)}, c^{(2)}, \dots, c^{(m)}$ . Let  $\mathcal{P}^{(o)}$  be the optimal partition. We first find partitions that contain same core components as  $\mathcal{P}^{(o)}$ . Assume these partitions start

---

**Algorithm 1** Data clustering based on authentic scores of the edges in the weighted MKNN graph

---

**given** adjacency matrix  $A$  of the weighted MKNN graph  $G(V, E)$  and the number of clusters  $r$

**initialize**  $\mathcal{P}^{(0)} = \{G\}$ ,  $H = G$ ,  $J = [\ ]$ , and  $k = 1$

**calculate** the authentic scores of the edges in graph  $G$  using Eqs. 2, 3, 4 and 6

**sort** the edges in the ascending order of their authentic scores and store the result in list  $\mathcal{L}$

**for each** edge  $e$  in  $\mathcal{L}$

1) remove  $e$  from graph  $H$

2) do breath-first search or depth-first search to find all components in  $H$

3) sort the components by their size and store the result in  $\mathcal{P}^{(k)} = \{P_1^{(k)}, P_2^{(k)}, \dots, P_{q^{(k)}}^{(k)}\}$ , where  $q^{(k)}$  is the number of components,  $P_i^{(k)}$  is the set of nodes in the  $i$ -th component and  $|P_i^{(k)}| \geq |P_{i+1}^{(k)}|$  for  $i = 1, 2, \dots, q^{(k)} - 1$

4) add  $e$  to  $J$

5)  $k \leftarrow k + 1$

**find** the optimal partition  $\mathcal{P}^{(o)}$  of graph  $G$  from  $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \dots, \mathcal{P}^{(m)}$

**for each**  $e$  in  $J$  in reverse order

1) add  $e$  to  $H$

2) if  $e$  links a component in  $\{P_{r+1}^{(o)}, P_{r+2}^{(o)}, \dots, P_{q^{(o)}}^{(o)}\}$  and a component in  $\{P_1^{(o)}, P_2^{(o)}, \dots, P_r^{(o)}\}$ , merge the small component into the large one.

3) if all components in  $\{P_{r+1}^{(o)}, P_{r+2}^{(o)}, \dots, P_{q^{(o)}}^{(o)}\}$  are merged

break

**return**  $P_1^{(o)}, P_2^{(o)}, \dots, P_r^{(o)}$  as the clustering result

---

at index  $u$  and end at index  $v$ . We have  $u \leq o \leq v$ . Let  $d^{(i)} = c^{(i+1)} - c^{(i)}$ , for  $i = u, u + 1, \dots, v - 1$ . Let

$$w = \arg \max_{i=u, u+1, \dots, v-1} (d^{(i)}). \quad (10)$$

Then the partition  $\mathcal{P}^{(w)}$  is the optimal partition for outlier detection. In this case, the core components of  $\mathcal{P}^{(w)}$  are the normal data cluster, and the nodes that do not belong to any core component are outliers.

#### F. Batch Mode of the Proposed Algorithm

The number of iterations in Algorithm 1 equals to the number of edges in the MKNN graph, which can be large when the dataset is big. In practice, we can collapse the MKNN graph by removing the edges in a batch mode. We simply define a constant number  $T$ . At each iteration,  $m/T$  edges are removed. After the graph collapsing stage, we have only  $T$  candidate partitions to deal with. This greatly decreases the computational complexity, since the number of iterations does not depend on the dataset size. In this study we simply choose  $T$  to be 100.

#### G. Complexity Analysis

Next we analyze the computational complexity and memory requirements of the proposed algorithm. The proposed algorithm contains several stages.

First, we build a MKNN graph from the given dataset. The computational complexity of building up a MKNN graph by a brutal-force method is  $O(n^2)$ . However, Callahan et al. showed that theoretically the KNN graph construction takes  $O(n \log n + nk)$  [11]. Fast methods to approximate MKNN graph are also available [12]. For example, Connor et al.

claimed a method with the complexity of  $O\left(\left\lceil \frac{n}{p} \right\rceil k \log k\right)$ , where  $p$  is the number of threads [13].

Second, we calculate the authentic scores for each edge in the MKNN graph. The complexity of calculating authentic scores is  $O(mk^2)$  [10]. Note that  $m \leq \frac{1}{2}kn$  for a MKNN graph. Sorting  $m$  authentic scores requires computational time  $O(m \log m)$  or  $O(kn \log(kn))$ . So the complexity for calculating the authentic scores is  $O(k^3n + kn \log(kn))$ .

Third, we apply the graph collapsing procedure in the batch mode described in Algorithm 1 and Section III-F. As a batch of the edges are removed, we run breadth-first search or depth-first search to find all components. The computational complexity of these algorithms are  $O(n + m)$ . Since we need to do the search  $T$  times, the complexity for the whole procedure is  $O(T(n + m))$  or  $O(Tkn)$ .

The forth stage of the algorithm is to determine the optimal partition. The complexity of the Max-Min method is  $O(T)$ . The complexity of the Min-Max method is  $O(Tkn)$ .

The last stage is the graph reuniting stage. For every edge we restore, we simply update the clustering identification of the two end nodes. The maximum number of edges that we need to restore is  $m$ . The computational complexity of this stage is  $O(m)$  or  $O(kn)$ .

For outlier detection, determination of the optimal partition involves two steps. The first step is to check if  $P_1^{(k+1)}, P_2^{(k+1)}, \dots, P_r^{(k+1)}$  contain the same core component as  $P_1^{(k)}, P_2^{(k)}, \dots, P_r^{(k)}$  and we repeat this checking until we find all the same partitions around the optimal partition  $o$ . The complexity of checking each pair of the partition is  $O(n)$ . In the worst case that we need to check all the partitions, the overall complexity is  $O(Tn)$ . The second step is to find the



solution of Eq. 10, which has the complexity of  $O(T)$ .

Taking all the above stages into consideration and ignoring constants and insignificant terms, the overall complexity of the proposed algorithm is  $O(k^3n + kn \log(kn))$ . The memory requirement of the proposed algorithm, which mainly involves manipulating the MKNN graph, is  $O(kn)$ . Note that the previous analysis suppose that the degree of each node is  $k$  (KNN graph). In reality, the average degree of each node in a MKNN graph is less than  $k$ .

Table I compares the computational complexity of the proposed algorithm and other popular clustering algorithms. In regards to  $n$ , the proposed algorithm has computational complexity of  $O(n \log n)$  which is one of the best among all the algorithms. However, when  $n$  is small and  $k$  is large, the proposed algorithm may be slower than other algorithms since the term  $O(k^3n)$  dominates.

TABLE I. COMPUTATIONAL COMPLEXITY OF THE PROPOSED ALGORITHMS AND OTHER POPULAR CLUSTERING ALGORITHMS

	Complexity
GDL [6]	$O(n^2)$
k-means [1], [14], [15]	$O(ni)$ *
a-link [1], [15]	$O(n^2 \log n)$
N-Cut [16]	$O(ni)$ #
DBSCAN [2], [15]	$O(n \log n)$
Ours	$O(k^3n + n \log n)$

\*  $i$  is the number of iterations. In practice,  $i$  is difficult to estimate and in worst case it is superpolynomial [17].

#  $i$  is the number of iterations. Shi and Malik claimed that  $i$  is typically less than  $O(n^{1/2})$  [16].

## IV. EXPERIMENTS

### A. Synthetic noisy toy datasets

In this experiment, we evaluate the efficiency of the proposed algorithm using generated noisy toy datasets. Each dataset contains 2000 normal data points and 500 uniformly distributed noisy data points <sup>1</sup>. Fig. 2 shows the synthetic noisy toy datasets. Different clusters and noisy data points are colored differently.

We compared the proposed algorithm with a set of popular clustering algorithms: graph degree linkage (GDL) [6],  $k$ -means[1], average linkage (a-link)[1], normalized cuts (N-Cut) [16], self-tuning spectral clustering(STSC) [18] and DBSCAN (DBS) [2]. The optimal partition is obtained by the Min-Max method described in Section III-D2.

To evaluate the performance of different algorithms, we choose the Normalized Mutual Information (NMI) as the performance metric [19], [20]. NMI values are between 0 and 1. A larger NMI value indicates a better clustering result. Value 1 means that the algorithm gives the same clustering as the ground truth. For GDL and the proposed algorithm, we use  $k = 30$  to build the MKNN graph. Table II shows the performance of the selected algorithms. The best NMI values are illustrated with bold font.

<sup>1</sup>Normal data points were generated using the tools from <http://www.mathworks.com/matlabcentral/fileexchange/41459>

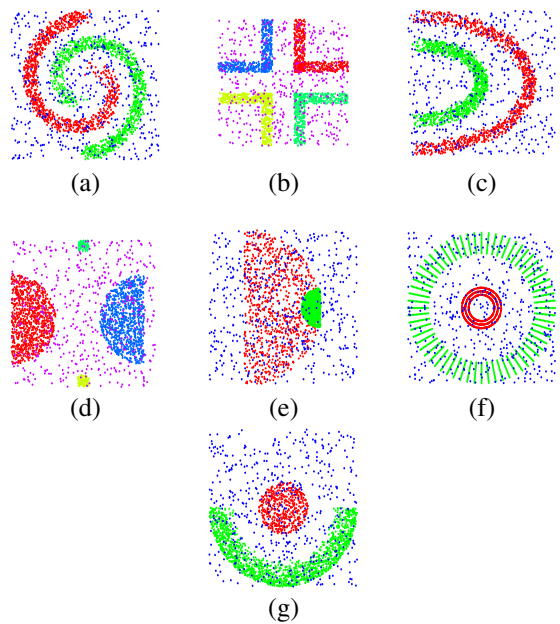


Fig. 2. Synthetic noisy datasets

TABLE II. NMI VALUES OF THE SELECTED ALGORITHMS ON SYNTHETIC NOISY DATASETS

	GDL	$k$ -means	a-link	N-Cut	STSC	DBSCAN	Ours
(a)	0.650	0.031	0.099	0.053	0.022	0.724	<b>0.734</b>
(b)	0.743	0.743	0.743	0.743	0.743	<b>0.847</b>	0.753
(c)	0.654	0.000	0.004	0.559	0.654	0.749	<b>0.754</b>
(d)	0.646	0.454	0.717	0.691	0.483	0.819	<b>0.823</b>
(e)	0.553	0.208	0.161	0.367	0.517	0.287	<b>0.646</b>
(f)	0.701	0.001	0.133	0.680	0.714	0.429	<b>0.750</b>
(g)	0.612	0.001	0.162	0.627	0.641	<b>0.749</b>	<b>0.749</b>

As Table II shows, the proposed algorithm achieves the best scores on all of the testing datasets except dataset (b).  $k$ -means fails on almost all of the datasets since it can not handle complex cluster shapes [21]. Poor performance observed in the average linkage method is due to the fact that the algorithm is very sensitive to noise. Normalized cuts, GDL and STSC shows better performance than  $k$ -means and average linkage methods. But these methods do not detect noisy data points. DBSCAN is a density-based clustering algorithm. It can find clusters with complex shapes and detect noise data points. DBSCAN shows similar performance as the proposed algorithm on dataset (a), (b), (c), (d) and (g). However, DBSCAN is not able to handle the clusters that their densities vary a lot, as shown in the results of datasets (e) and (f). The results clearly indicate that the proposed algorithm can cluster data with complex shape, detect noisy data points and is robust to the variation of the density of the clusters.

### B. Synthetic 2D Datasets

In this section we evaluate the performance of the proposed algorithm using 2D datasets that have been widely used in other studies <sup>2</sup>: (a) [22]; (b) [23]; (c) and (d) [24]; (e) [25];

<sup>2</sup>The datasets were downloaded from <https://www2.uef.fi/en/sipu/data-and-software>

(f) [26]; (g) and (h) [27]. Fig. 3 shows the testing datasets.

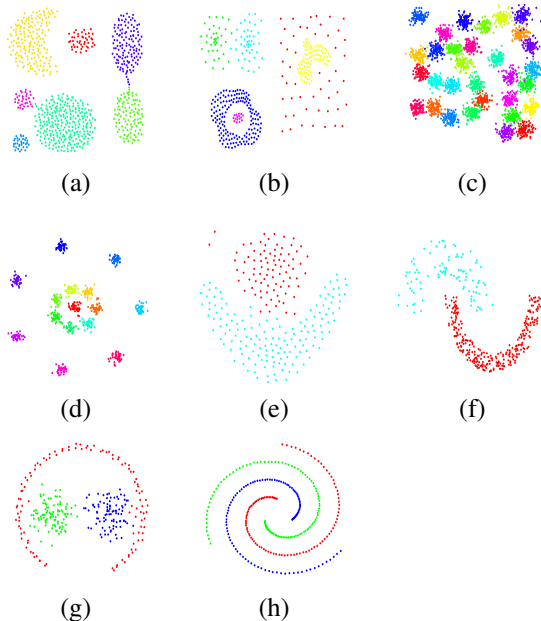


Fig. 3. Synthetic 2D datasets: (a) [22]; (b) [23]; (c) and (d) [24]; (e) [25]; (f) [26]; (g) and (h) [27].

Because of the small size of the clusters in these datasets, we set  $k = 10$  for the proposed algorithm. Noisy data detection is not performed for these datasets. Other parameters are same as the ones used in the experiments in Section IV-A. Table III shows the NMI scores of the competing algorithms. The scores with underline are clearly below the best score thus indicate the failure of the method on that dataset.

TABLE III. NMI VALUES OF THE SELECTIVE ALGORITHMS ON SYNTHETIC 2D DATASETS

	GDL	k-means	a-link	N-Cut	STSC	DBSCAN	Ours
(a)	0.993	0.880	<b>1.000</b>	0.980	0.975	0.890	0.985
(b)	0.850	<u>0.738</u>	0.837	<u>0.762</u>	<u>0.763</u>	0.918	<b>0.987</b>
(c)	0.955	0.949	0.952	<b>0.967</b>	0.882	<u>0.756</u>	0.942
(d)	<b>0.994</b>	<b>0.994</b>	0.992	<b>0.994</b>	0.941	<u>0.768</u>	0.976
(e)	<b>1.000</b>	0.399	0.483	0.927	0.564	0.048	0.927
(f)	<b>1.000</b>	<u>0.367</u>	<u>0.698</u>	0.224	<u>0.661</u>	0.820	<b>1.000</b>
(g)	<u>0.583</u>	<u>0.547</u>	<u>0.522</u>	<b>0.879</b>	<u>0.584</u>	0.046	0.769
(h)	0.007	0.000	0.003	0.224	0.330	<b>1.000</b>	<b>1.000</b>

The results show that the proposed algorithm and GDL work better on clusters with complex shapes, such as (e) and (f). Normalized cuts and  $k$ -means works better on clusters of normal distributed data points, such as (c) and (e). The proposed algorithm is the only one that works on dataset (b) where the density of the clusters varies significantly. Note that the proposed algorithm is the only one that gives satisfactory performance on all the datasets.

The authentic score of an edge is computed within a limited range of its neighboring nodes. This affects the accuracy of a few data points that lie on the border of the clusters and slightly worsen the performance on datasets (c), (d) and (e).

### C. Clustering on Real Image Datasets

In this section we show the performance of the proposed algorithms on real-world image datasets. We use the digital

image datasets MNIST and USPS <sup>3</sup>, object image datasets COIL-20 and COIL-100 <sup>4</sup> and face image datasets UMist, FRGC, CMU-PIE and YTF. In this experiment, we use raw pixel values as the feature vector.

Table IV shows the NMI scores of the selective algorithms using raw pixel value as feature vector:  $k$ -median, graph-based average linkage (GLink), N-Cut, STSC, GDL and the proposed algorithm. The proposed algorithm achieves either the best or the second best results on all datasets except the CMU-PIE.

Yang et al. proposed a supervised approach that jointly learn deep representations and image clusters. Using the deep representations, the clustering result can be greatly improved. Table V shows the NMI scores of different clustering algorithms using the deep representations as feature vector.

The results are conformance to other experiments. The proposed algorithm gives best NMI scores on 5 out of 8 real-world image datasets.

## V. CONCLUSIONS

Community structure is a common phenomenon in social and biological networks. Zhang et.al. studied this phenomenon and proposed authentic scores to measure the strength of the edges. In this paper we propose a novel and efficient data clustering algorithm using the authentic scores of the edges in the weighted MKNN graph. The proposed algorithm collapses the MKNN graph by gradually removing edges in the ascending order of their authentic scores. During the collapsing procedure, we detect the components in the graph by either breadth-first search or depth-first search algorithm. The optimal partition can be determined by either maximizing the minimal cluster size (Max-Min) method or minimizing the maximal conductance value (Min-Max) method. We also show that the proposed method can detect noise from the input dataset. We evaluated the proposed algorithm over both synthetic and real-world image datasets. The results clearly show that the proposed algorithm is superior to the competing algorithms. The proposed algorithm has the computational complexity of  $O(k^3n + kn \log(kn))$  and memory requirement of  $O(kn)$ .

The proposed algorithm is fast and efficient. It is able to find clusters of complex shape and is insensitive to density variations of the clusters. The results of the experiments also show some limitations. The proposed algorithm is not able to separate clusters that are connected by a strong bridge. This limitation also applies to other density and graph-based algorithms, such as GDL and DBSCAN. Because the authentic scores are calculated from a limited range of the neighboring nodes, the data points on the border of a cluster may be misplaced. Addressing this problem in an effective way will be the topic of our future study.

<sup>3</sup>MNIST and USPS datasets were downloaded from <http://www.cs.nyu.edu/~roweis/data.html>

<sup>4</sup>COIL-20 and COIL-100 datasets were downloaded from <http://www.cs.columbia.edu/CAVE/software/>

TABLE IV. NMI SCORES OF THE SELECTIVE ALGORITHMS ON IMAGE DATASETS

	COIL-20	COIL100	USPS	MNIST	UMist	FRGC	CMU-PIE	YTF
k-means	0.775	0.822	0.447	0.528	0.609	0.389	0.549	0.761
G-Link	0.710*	0.706*	0.732*	0.808*	-	-	-	-
N-Cut	0.884	0.823	0.675	0.735	0.782	0.285	0.411	0.742
STSC	0.895	0.858	0.726	0.756	0.611	0.431	0.581	0.620
GDL	0.937	0.929	0.824	0.844	0.755	0.351	<b>0.934</b>	0.622
SC-LS	0.877	0.833	0.681	0.756	0.810	<b>0.550</b>	0.788	0.759
AC-Zell	0.911	0.913	0.799	0.768	0.755	0.351	0.910	0.733
AC-PIC	0.950	<b>0.964</b>	0.840	<b>0.853</b>	0.750	0.415	0.902	0.697
ours	<b>0.979</b>	0.957	<b>0.854</b>	0.848	<b>0.893</b>	0.457	0.744	<b>0.811</b>

The NMI scores of the algorithms marked with \* are taken from [6], other scores except those of the proposed algorithm are taken from [28].

TABLE V. NMI SCORES OF THE SELECTIVE ALGORITHMS ON IMAGE DATASETS USING DEEP REPRESENTATIONS

	COIL-20	COIL100	USPS	MNIST	UMist	FRGC	CMU-PIE	YTF
k-means	0.926	0.919	0.758	0.908	0.871	0.636	0.956	0.835
N-Cut	0.963	0.900	0.705	0.910	0.877	0.640	0.995	0.823
STSC	0.959	0.922	0.741	0.911	0.847	0.651	0.938	0.741
GDL	<b>1</b>	0.985	0.913	<b>0.915</b>	0.870	0.574	<b>1</b>	<b>0.842</b>
SC-LS	0.950	0.905	0.780	0.912	<b>0.879</b>	0.639	0.950	0.802
AC-Zell	<b>1</b>	0.989	0.910	0.893	0.870	0.551	<b>1</b>	0.821
AC-PIC	<b>1</b>	0.990	0.914	0.909	0.870	0.553	<b>1</b>	0.829
ours	<b>1</b>	<b>0.991</b>	<b>0.915</b>	0.912	0.877	<b>0.658</b>	<b>1</b>	0.824

## REFERENCES

- [1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.
- [2] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, 1996, pp. 226–231.
- [3] D. Harel and Y. Koren, "On Clustering Using Random Walks," in *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science*, ser. Lecture Notes in Computer Science, R. Hariharan, V. Vinay, and M. Mukund, Eds. Springer Berlin Heidelberg, 2001, no. 2245, pp. 18–41.
- [4] K. Ozaki, M. Shimbo, M. Komachi, and Y. Matsumoto, "Using the mutual k-nearest neighbor graphs for semi-supervised classification of natural language data," in *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2011, pp. 154–162.
- [5] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: theory and its application to image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1101–1113, Nov. 1993.
- [6] W. Zhang, X. Wang, D. Zhao, and X. Tang, "Graph Degree Linkage: Agglomerative Clustering on a Directed Graph," in *Computer Vision ECCV 2012*, ser. Lecture Notes in Computer Science, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Springer Berlin Heidelberg, 2012, no. 7572, pp. 428–441.
- [7] W. Zhang, D. Zhao, and X. Wang, "Agglomerative clustering via maximum incremental path integral," *Pattern Recognition*, vol. 46, no. 11, pp. 3056–3065, Nov. 2013.
- [8] Z. Hu and R. Bhatnagar, "Clustering algorithm based on mutual K-nearest neighbor relationships," *Statistical Analysis and Data Mining*, vol. 5, no. 2, pp. 100–113, Apr. 2012.
- [9] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, Jun. 2002.
- [10] H. Zhang, S. Kiranyaz, and M. Gabbouj, "Outlier edge detection using random graph generation models and applications," *Journal of Big Data*, vol. 4, no. 1, p. 11, Apr. 2017.
- [11] P. B. Callahan and S. R. Kosaraju, "A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields," *Journal of the ACM (JACM)*, vol. 42, no. 1, pp. 67–90, 1995.
- [12] W. Dong, C. Moses, and K. Li, "Efficient K-nearest Neighbor Graph Construction for Generic Similarity Measures," in *Proceedings of the 20th International Conference on World Wide Web*, ser. WWW '11. New York, NY, USA: ACM, 2011, pp. 577–586.
- [13] M. Connor and P. Kumar, "Fast construction of k-nearest neighbor graphs for point clouds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 4, pp. 599–608, Jul. 2010.
- [14] D. Xu and Y. Tian, "A Comprehensive Survey of Clustering Algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, Jun. 2015.
- [15] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005.
- [16] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, 2000.
- [17] D. Arthur and S. Vassilvitskii, "How slow is the k-means method?" in *Proceedings of the twenty-second annual symposium on Computational geometry*. ACM, 2006, pp. 144–153.
- [18] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Advances in neural information processing systems*, 2004.
- [19] L. Ana and A. Jain, "Robust data clustering," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings*, vol. 2, Jun. 2003, pp. II–128–II–133 vol.2.
- [20] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 09, p. P09008, 2005.
- [21] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, Jun. 2010.
- [22] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 4, 2007.
- [23] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *Computers, IEEE Transactions on*, vol. 100, no. 1, pp. 68–86, 1971.
- [24] C. J. Veenman, M. J. Reinders, and E. Backer, "A maximum variance cluster algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 9, pp. 1273–1280, 2002.
- [25] L. Fu and E. Medico, "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data," *BMC bioinformatics*, vol. 8, no. 1, p. 3, 2007.
- [26] A. K. Jain and M. H. Law, "Data clustering: A users dilemma," in *Pattern Recognition and Machine Intelligence*. Springer, 2005, pp. 1–10.
- [27] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 191–203, 2008.
- [28] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5147–5156.

# PUBLICATION

## V

### **Feature Dimensionality Reduction with Graph Embedding and Generalized Hamming Distance**

H. Zhang and M. Gabbouj

*2018 25th IEEE International Conference on Image Processing (ICIP)2018, 1083–1087*

DOI: 10.1109/ICIP.2018.8451089

**Publication reprinted with the permission of the copyright holders**



# FEATURE DIMENSIONALITY REDUCTION WITH GRAPH EMBEDDING AND GENERALIZED HAMMING DISTANCE

Honglei Zhang, Moncef Gabbouj

Signal Processing  
Tampere University of Technology  
Tampere, Finland

## ABSTRACT

Principal component analysis (PCA) and linear discriminant analysis (LDA) are the most well-known methods to reduce the dimensionality of feature vectors. However, both methods face challenges when used on multilabel data—each data point may be associated to multiple labels. PCA does not take advantage of label information thus the performance is sacrificed. LDA can exploit class information for multiclass data, but cannot be directly applied to multilabel problems. In this paper, we propose a novel dimensionality reduction method for multilabel data. We first introduce the generalized Hamming distance that measures the distance of two data points in the label space. Then the proposed distance is used in the graph embedding framework for feature dimension reduction. We verified the proposed method using three multilabel benchmark datasets and one large image dataset. The results show that the proposed feature dimensionality reduction method consistently outperforms PCA and other competing methods.

### *Index Terms*—

dimensionality reduction, graph embedding, multilabel

## 1. INTRODUCTION

Feature dimensionality reduction using graph embedding paradigm is a common approach [1]. Yan et al. showed that Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Isomap and many other dimensionality reduction methods can be unified under a general graph embedding paradigm by using different intrinsic and penalty graph. Dimensionality reduction method using PCA aims to represent the data better in the new lower-dimension feature space [2]. The intrinsic graph of PCA algorithm is a fully connected graph where all edges have the same weight. LDA algorithm is applicable for multiclass data, where each sample belongs to one of the classes [3]. The features are projected into a lower-dimensional space aiming to maximize the effectiveness of discriminant.

However, many real world problems involves multilabel data, where each data point is associated with multiple la-

bels—in contrast to multiclass data, where each data point is assigned to one class label [4]. The classes in multiclass data are mutually exclusive. However the relations of the labels in multilabel data can be more complicate. For example, an image dataset may have many labels, such as “dog”, “cat”, “animal”, “running”, “funny”, “Labrador”, “black”, “pink”. One may use any combination of these labels to describe an image, although some combination may be rare, such as “pink Labrador”. Because of the complexity of the labels, LDA can not be directly applied for dimensionality reduction.

Previous researches have combined the feature dimensionality reduction with classification task by optimizing a joint objective function [5, 6]. These approaches have obvious limitation that the dimensionality reduction algorithm can be not combined with other classification or ranking algorithms. Thus the system accuracy is bounded by the chosen joint objective function.

In this paper, we propose the generalized Hamming distance metric that catches the relations between the labels. We show that Hamming distance is a special case of the proposed metric when the labels are mutually independent. We use this metric in dimensionality reduction methods under the graph embedding framework. The proposed algorithm can be used together with any classification or ranking algorithm.

## 2. NOTATIONS AND PREVIOUS WORK

### 2.1. Notations

Given a multilabel dataset, let  $L = \{L_1, L_2, \dots, L_q\}$  be the set of labels where  $q$  is the number of labels. Let  $X = \{x_1, x_2, \dots, x_N\}$  be the set of the samples where  $x_i \in R^M$  and  $M$  is the dimension of the features. Let  $Y = \{y_1, y_2, \dots, y_N\}$  be the labels assigned to the corresponding data points. We use binary code to represent the labels of each data point, such that  $y_i \in \{0, 1\}^q$ . Let  $y_i(k)$  be the  $k$ -th element of vector  $y_i$ . We have  $y_i(k) = 1$  if data point  $x_i$  is associated with label  $L_k$ ; otherwise  $y_i(k) = 0$ .

To find a lower-dimensional representation of the data points, our target is to learn a projection  $z = f(x)$  where  $z \in R^P$ ,  $P$  is dimension of the new feature space and

$P < M$ . In this paper, we study the linear projection of the feature space, such that  $z = Wx$ , where  $W \in R^{P \times M}$  is the projection matrix.

## 2.2. Previous Work

The basic idea of dimensionality reduction with graph embedding is to manipulate the distances between the pairs of data points in the projected feature space. Different weights are assigned to the distances and the algorithm tries to optimize the sum of the weighted distances. Let  $A_{ij}$  be the weight for the distance between data point  $x_i$  and  $x_j$  in the projected feature space. The objective function is defined as

$$J = \sum_{i,j,i \neq j} \|Wx_i - Wx_j\|^2 A_{ij}. \quad (1)$$

By maximizing this objective function with regard to the projection matrix  $W$ , two data points with a larger weight are expected to have larger distance in the projected space. However, to make the optimization tractable, we apply a regularization term

$$x^T W^T B W x = I, \quad (2)$$

where  $I$  is the identity matrix and  $B$  is a predefined matrix. When  $B$  is an identical matrix, this regularization term forces the data points to lie on a unit sphere in the projected feature space.

Next, we can generate a complete graph in which each node is a data point and the edge connecting node  $x_i$  and  $x_j$  has the weight  $A_{ij}$ . Let  $A$  be the weighted adjacency matrix. Let  $D$  be a diagonal matrix with the weighted degree of each node on its diagonal, such that  $D_{ii} = \sum_{j=1}^N A_{ij}$ . The Laplacian matrix of the graph is defined as

$$L = D - A. \quad (3)$$

Yan et al. showed that PCA, LDA and other dimensionality reduction methods can be unified by this formation by choosing different matrix  $A$  and regularization matrix  $B$  [1]. The graph defined by matrix  $A$  is called intrinsic graph and the graph defined by matrix  $B$  is called penalty graph. For example, in the formation of PCA,  $A_{ij} = \frac{1}{N}$  and  $B = I$ ; in the formation of LDA,  $A_{ij} = \delta(y_i, y_j)$  and  $B = 1 - \frac{1}{N}ee^T$ , where  $e$  is the  $N$  dimensional vector of 1 and  $\delta(y_i, y_j)$  is defined by:

$$\delta(y_i, y_j) = \begin{cases} 1 & y_i = y_j \\ 0 & \text{otherwise} \end{cases}. \quad (4)$$

PCA tries to represent the data better in a lower-dimensional feature space, while LDA tries to maximize the discriminant in the new feature space. LDA normally achieves better performance for the classification tasks. However, for the multilabel problems that deal with the data that can be associated with multiple labels, LDA can not be applied directly since function  $\delta(y_i, y_j)$  takes effect only if the two data

points have exactly same labels. Furthermore, because of the complex relationship between the labels, LDA algorithm can not capture the dependencies between the labels. Next, we introduce a dimensionality reduction method using a novel weight definition. The proposed algorithm can be applied to multilabel data and take the dependencies between the labels into consideration.

## 3. METHODOLOGY

As discussed in Section 2.2, the weight  $A_{ij}$  in Eq. 1 preserve the distance (or similarity) of the data points in the lower-dimensional feature space. For the multilabel classification problem or ranking problem, it is obvious that the data points sharing many common labels should be close to each other in projected feature space; while those data points that do not share common labels should be separated far away. With this intention, the following metrics may be used as the weight:

- Euclidean distance:

$$A_{ij} = \|y_i - y_j\|^2 \quad (5)$$

- Hamming distance:

$$A_{ij} = \text{count}(y_i \oplus y_j), \quad (6)$$

where  $\oplus$  is the XOR operator of two binary vector and  $\text{count}(\cdot)$  calculates the number of 1s in a binary vector. Note, hamming distance calculate number of labels that differs in  $y_i$  and  $y_j$

However, these commonly used metrics ignore the relationship between the labels and assume all the labels are independent. For multilabel problems, the relationship between the labels are complicate. For example, the correlation between label “dog” and “puppy” is obviously higher than that of “dog” and “table”. Thus the data points labeled with “dog” and “puppy” should be closer than data points labeled “dog” and “table” in the projected feature space. With this observation, we propose to use the mutual information to capture the correlations between labels and define a novel intrinsic graph that is suitable for dimension reduction of multilabel data.

### 3.1. Normalized mutual information of labels

In information theory, mutual information measures the relations between two random variables. Mutual information of two random variables  $X$  and  $Y$  is defined as

$$I(X; Y) = \sum_y \sum_x p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \quad (7)$$

Note that mutual information is greater than zero but not bounded from above [7]. We use the following normalized

mutual information to capture the correlation between the random variables:

$$NI(X, Y) = \frac{I(X; Y)}{\min(H(X), H(Y))}, \quad (8)$$

where  $H(X)$  and  $H(Y)$  are the marginal entropies of variable  $X$  and  $Y$ . Eq. 8 is equivalent to the total correlation, which is the Kullback-Leibler divergence from distribution  $p(X, Y)$  to  $p(X)p(Y)$ . It is easy to see that  $0 \leq NI(X, Y) \leq 1$ , where 0 is reached if the two random variables are independent and 1 is reached if they are linearly correlated.

We consider each label  $L_i$  as a random variable. Given the set of multilabel data  $X$  and the set of corresponding labels  $Y$ , empirical probability of  $L_i$  can be calculated by

$$p(L_i) = \frac{1}{N} \sum_{k=1}^N y_k(i), \quad (9)$$

and the joint probability of two labels  $L_i$  and  $L_j$  by

$$p(L_i, L_j) = \frac{1}{N} \sum_{k=1}^N y_k(i)y_k(j). \quad (10)$$

From Eqs. 7, 8, 9 and 10, we can calculate the empirical normalized mutual information matrix  $F$  where element  $F_{ij} = NI(L_i, L_j)$ . Note that matrix  $F$  is symmetric.

We consider  $y_i$  lies in a label space where the basis are the labels. If the labels are correlated, the basis of the label space are nonorthogonal. Given matrix  $F$ , next we define a novel metric to measure the distance of two data points in label spaces.

### 3.2. Generalized hamming distance

Hamming distance defined in Eq. 6 is the number of different labels that are associated to data points  $x_i$  and  $x_j$ . Eq. 6 can be written as:

$$A_{ij} = \text{count}(y_i \vee y_j) - \langle y_i, y_j \rangle, \quad (11)$$

where “ $\vee$ ” is the “or” operator of two binary vectors. Because of the correlations between labels, the basis of label space is not orthogonal. The inner product of two vectors  $y_i$  and  $y_j$  with nonorthogonal basis is defined as

$$\langle y_i, y_j \rangle = \sum_l \sum_m y_i(l)y_j(m) \langle \mathbf{e}_l, \mathbf{e}_m \rangle, \quad (12)$$

where  $\mathbf{e}_l$  and  $\mathbf{e}_m$  are the basis vectors. From Eqs. 11 and 12, we define the generalized Hamming distance of two samples  $x_i$  and  $x_j$  with label  $y_i$  and  $y_j$  in label space as the following:

$$A_{ij} = \text{count}(y_i \vee y_j) - y_i^T F y_j, \quad (13)$$

where  $F$  is the normalized mutual information matrix defined in Section 3.1. The first term of Eq. 13 counts the number of labels that data points  $x_i$  and  $x_j$  are associated. The second term is the inner product of vector  $y_i$  and  $y_j$  in the label space.

**Theorem 1.** *Generalized Hamming distance becomes Hamming distance if labels are mutually independent.*

*Proof.* Let  $L_i$  and  $L_j$  be two independent label variable. We have  $I(L_i, L_j) = 0$ . Thus  $NI(L_i, L_j) = 0$ . Since  $I(L_i, L_i) = H(L_i) - H(L_i|L_i) = H(L_i)$ , we have  $NI(L_i, L_i) = 1$ . Thus matrix  $F$  in Eq. 13 is an identical matrix when the label variables are mutually independent. According to Eqs. 6 and 13, the theorem is proved.  $\square$

### 3.3. Solving the optimization problem

When the weights  $A_{ij}$  are known, our target is to find the optimal solution of the objective function defined in Eq. 1. Such that

$$W^* = \arg \max_W \sum_{i,j,i \neq j} \|Wx_i - Wx_j\|^2 A_{ij} \quad (14)$$

subject to  $x_i^T W^T W x_i = 1$  for  $i = 1, 2, \dots, N$

The solution of optimization problem 14 can be obtained by solving the eigenvalue problem

$$\tilde{L}w = \lambda w,$$

where  $\tilde{L} = X^T L X$  and  $L$  is the Laplacian matrix of the intrinsic graph [8]. By keeping the first  $P$  (the dimension of the projected feature space) eigenvectors of matrix  $\tilde{L}$  with the largest eigenvalues, we get the matrix  $W^*$ .

## 4. EXPERIMENTS

In this section, we evaluate the proposed dimension reduction method using some of the most popular benchmark datasets for multilabel ranking tasks.

### 4.1. Small datasets

We first evaluate the proposed algorithm using three small datasets taken from [9]. The statistics of these datasets are shown in Table 1.

**Table 1.** Statistics of the testing datasets

Name	domain	instance	label	dimension	cardinality
Yeast [10]	biology	2417	14	103	4.24
Scene [11]	image	2407	6	294	1.07
Emotions [12]	music	593	6	72	1.87

In our first experiment, we used the Yeast dataset [10] and applied the proposed dimensionality reduction method to reduce the dimension from 103 to 1, 2, 4, 8 and 16. After dimensionality reduction, we applied MLkNN method [13] on the multilabel ranking problem and recorded the ranking loss [14] values. We compared 5 results using different weight definitions in the intrinsic graph :



- PCA:  $A_{ij} = \frac{1}{N}$ .
- Hamming distance (Eq. 6).
- Euclid. Y: Euclidean distance on labels (Eq. 5).
- Euclid. X:  $A_{ij} = \|x_i - x_j\|^2$ . This is the Euclidean distance of the training data in its original feature space.
- GH: Generalized Hamming distance (Eq. 13).

Table 2 shows the results of the competing methods. Best score among all methods are shown in bold font.

**Table 2.** Raking loss values of dimensionality deduction methods and ML-kNN on Yeast data

Dimension	1	2	4	8	16
PCA	<b>0.209</b>	<b>0.202</b>	0.196	0.18	<b>0.172</b>
Hamming	0.212	0.205	0.198	<b>0.177</b>	0.174
Euclid. Y	0.21	0.208	0.198	<b>0.177</b>	0.173
Euclid. X	<b>0.209</b>	0.204	0.197	0.179	0.173
GH	<b>0.209</b>	<b>0.202</b>	<b>0.195</b>	<b>0.177</b>	<b>0.172</b>

As Table 2 shows, the dimensionality reduction using generalized Hamming distance achieves the best results on all dimensions.

Next we executed the experiments on Yeast, Scene and Emotions datasets. To evaluate the general performance of different methods, we recorded the average ranking loss of dimensions of 1, 2, 4, 8 and 16. We used the competing dimensionality reduction methods and different multilabel ranking methods: MLkNN [13], IBLR\_ML [15], BRkNN[16], DMLkNN [17] and RLkNN [18]. The results are shown in Tables 3, 4 and 5.

**Table 3.** Average ranking loss on Yeast dataset

method	MLkNN	IBLR_ML	BRkNN	DMLkNN	RLkNN
PCA	0.1918	0.1917	0.194	<b>0.1986</b>	<b>0.2015</b>
Hamming	0.1932	0.192	0.1952	0.2022	0.2015
Euclid. Y	0.1932	0.1916	0.195	0.2015	0.2015
Euclid. X	0.1924	0.1919	0.195	0.1992	0.202
GH	<b>0.191</b>	<b>0.1887</b>	<b>0.1922</b>	0.1995	0.2019

**Table 4.** Average ranking loss on Scene dataset

method	MLkNN	IBLR_ML	BRkNN	DMLkNN	RLkNN
PCA	0.1542	0.1482	0.16	0.1614	0.1551
Hamming	0.1537	0.1474	<b>0.1572</b>	0.1586	0.1529
Euclid. Y	0.1537	0.1468	0.158	0.159	<b>0.152</b>
Euclid. X	0.159	0.1534	0.1638	0.1646	0.1569
GH	<b>0.153</b>	<b>0.1462</b>	0.1575	<b>0.1574</b>	<b>0.152</b>

As the results show, the dimensionality reduction methods using generalized hamming distance consistently achieves better performance than other competing methods. It should also be noted that the proposed algorithm achieves the best average ranking loss score on all the datasets.

**Table 5.** Average ranking loss Emotions dataset

method	MLkNN	IBLR_ML	BRkNN	DMLkNN	RLkNN
PCA	0.2204	0.2019	0.2006	0.2352	0.1831
Hamming	<b>0.2138</b>	0.2011	0.1983	0.2302	0.1823
Euclid. Y	0.2161	<b>0.198</b>	0.1968	<b>0.223</b>	0.1828
Euclid. X	0.2282	0.1996	0.1996	0.2465	0.1881
GH	0.2147	0.2035	<b>0.1948</b>	0.2336	<b>0.18</b>

## 4.2. Big dataset

Next we evaluated the proposed method using a large image dataset–NUS-WIDE 128 [19]. The NUS-WIDE 128 dataset has 269648 instances and 61 labels. The original dimension of the feature is 128.

Since the number of samples is large, 5000 instances of data from the training dataset were randomly selected to calculate the projection matrix  $W$ . We reduced the dimension of the feature from 128 to 4 and 32 and recorded ranking loss of different multilabel classification algorithms. The results are shown in Table 6 and Table 7.

**Table 6.** Ranking loss on NUS-WIDE 128 dataset - feature dimension is 4

Dimension	MLkNN	IBLR_ML	BRkNN	DMLkNN	RLkNN
PCA	0.098	0.106	0.128	0.097	0.1422
Hamming	<b>0.096</b>	<b>0.103</b>	<b>0.126</b>	<b>0.095</b>	0.1396
Euclid. Y	0.097	0.104	0.126	<b>0.095</b>	0.1403
Euclid. X	0.098	0.106	0.128	0.097	0.142
GH	<b>0.096</b>	<b>0.103</b>	<b>0.126</b>	<b>0.095</b>	<b>0.1394</b>

**Table 7.** Ranking loss on NUS-WIDE 128 dataset - feature dimension is 32

Dimension	MLkNN	IBLR_ML	BRkNN	DMLkNN	RLkNN
PCA	0.093	0.097	0.127	0.091	0.1234
Hamming	0.092	0.097	<b>0.123</b>	<b>0.089</b>	<b>0.1195</b>
Euclid. Y	0.092	<b>0.096</b>	0.124	0.09	0.1202
Euclid. X	0.093	0.097	0.127	0.091	0.1233
GH	<b>0.091</b>	0.097	<b>0.123</b>	<b>0.089</b>	<b>0.1195</b>

As the results show, the proposed feature reduction method using generalized Hamming distance achieves the best results when combined with all multilabel ranking algorithms.

## 5. CONCLUSIONS

In this paper, we introduced the generalized hamming distance as a measurement to capture the correlations between the labels in a multilabel dataset. We applied the proposed metric to the graph embedding dimensionality reduction framework. We evaluated the proposed methods using three small benchmark datasets and a large image dataset that have been widely used for evaluating multilabel ranking algorithms. The results show that the proposed method consistently outperforms other dimensionality reduction methods.

## 6. REFERENCES

- [1] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and S. Lin, “Graph Embedding and Extensions: A General Framework for Dimensionality Reduction,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40–51, 2007.
- [2] Sergios Theodoridis and Konstantinos Koutroumbas, *Pattern Recognition, Fourth Edition*, Academic Press, Amsterdam, 4 edition edition, Nov. 2008.
- [3] William S. Rayens, “Discriminant Analysis and Statistical Pattern Recognition,” *Technometrics*, vol. 35, no. 3, pp. 324–326, Aug. 1993.
- [4] Eva Gibaja and Sebastián Ventura, “A Tutorial on Multilabel Learning,” *ACM Comput. Surv.*, vol. 47, no. 3, pp. 52:1–52:38, Apr. 2015.
- [5] Shuiwang Ji and Jieping Ye, “Linear Dimensionality Reduction for Multi-label Classification,” in *IJCAI*, 2009, vol. 9, pp. 1077–1082.
- [6] Yao-nan Chen and Hsuan-tien Lin, “Feature-aware Label Space Dimension Reduction for Multi-label Classification,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1529–1537. Curran Associates, Inc., 2012.
- [7] Thomas M. Cover and Joy A. Thomas, *Elements of information theory*, John Wiley & Sons, 2012.
- [8] Fan RK Chung, *Spectral graph theory*, vol. 92, American Mathematical Soc., 1997.
- [9] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas, “Mulan: A java library for multi-label learning,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2411–2414, 2011.
- [10] André Elisseeff and Jason Weston, “A kernel method for multi-labelled classification,” in *Advances in neural information processing systems*, 2001, pp. 681–687.
- [11] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown, “Learning multi-label scene classification,” *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.
- [12] Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P. Vlahavas, “Multi-Label Classification of Music into Emotions,” in *ISMIR*, 2008, vol. 8, pp. 325–330.
- [13] Min-Ling Zhang and Zhi-Hua Zhou, “ML-KNN: A lazy learning approach to multi-label learning,” *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, July 2007.
- [14] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski, “An extensive experimental comparison of methods for multi-label learning,” *Pattern Recognition*, vol. 45, no. 9, pp. 3084–3104, Sept. 2012.
- [15] Weiwei Cheng and Eyke Hüllermeier, “Combining instance-based learning and logistic regression for multilabel classification,” *Machine Learning*, vol. 76, no. 2-3, pp. 211–225, 2009.
- [16] Jianhua Xu, “Multi-Label Weighted k-Nearest Neighbor Classifier with Adaptive Weight Estimation,” in *Neural Information Processing*, Bao-Liang Lu, Liqing Zhang, and James Kwok, Eds., number 7063 in Lecture Notes in Computer Science, pp. 79–88. Springer Berlin Heidelberg, Nov. 2011.
- [17] Zoulficar Younes, Fahed Abdallah, Thierry Denoeux, and Hichem Snoussi, “A dependent multilabel classification method derived from the k-nearest neighbor rule,” *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 1, pp. 1–14, 2011.
- [18] Honglei Zhang, Serkan Kiranyaz, and Moncef Gabbouj, “A k-nearest neighbor multilabel ranking algorithm with application to content-based image retrieval,” in *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, Mar. 2017.
- [19] Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, Ioannis Yiannis Kompatsiaris, Grigorios Tsoumakas, and Ioannis Vlahavas, “A comprehensive study over vlad and product quantization in large-scale image retrieval,” *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1713–1728, 2014.



