

**HIGH PERFORMANCE DECENTRALISED
COMMUNITY DETECTION ALGORITHMS FOR
BIG DATA FROM SMART COMMUNICATION
APPLICATIONS**

AMHMED ABDULSALAM BHIH

A thesis submitted in partial fulfilment of the requirements of
Liverpool John Moores University for the degree of
Doctor of Philosophy

January 2018

ABSTRACT

Many systems in the world can be represented as models of complex networks and subsequently be analysed fruitfully. One fundamental property of the real-world networks is that they usually exhibit inhomogeneity in which the network tends to organise according to an underlying modular structure, commonly referred to as community structure or clustering. Analysing such communities in large networks can help people better understand the structural makeup of the networks. For example, it can be used in mobile ad-hoc and sensor networks to improve the energy consumption and communication tasks. Thus, community detection in networks has become an important research area within many application fields such as computer science, physical sciences, mathematics and biology.

Driven by the recent emergence of big data, clustering of real-world networks using traditional methods and algorithms is almost impossible to be processed in a single machine. The existing methods are limited by their computational requirements and most of them cannot be directly parallelised. Furthermore, in many cases the data set is very big and does not fit into the main memory of a single machine, therefore needs to be distributed among several machines.

The main topic of this thesis is about network community detection within these big data networks. More specifically, in this thesis, a novel approach, namely Decentralized Iterative Community Clustering Approach (DICCA) for clustering large and undirected networks is introduced. An important property of this approach is its ability to cluster the entire network without the global knowledge of the network topology. Moreover, an extension of the DICCA called Parallel Decentralized Iterative Community Clustering approach (PDICCA) is proposed for efficiently processing data distributed across several machines. PDICCA is based on MapReduce computing platform to work efficiently in distributed and parallel fashion.

In addition, the real-world networks are usually noisy and imperfect with missing and false edges. These imperfections are often difficult to eliminate and highly affect the quality and

accuracy of conventional methods used to find the community structure in the network. However, in real-world networks, node attribute information is also available in addition to topology information. Considering more than one source of information for community detection could produce meaningful clusters and improve the robustness of the network. Therefore, a pre-processing approach that considers attribute information, shared neighbours and connectivity information aspects of the network for community detection is presented in this thesis as part of my research.

Finally, a set of real-world mobile phone usage data obtained from Cambridge Laboratories (Device Analyzer) has been analysed as an exploratory step for viability to apply the algorithms developed in this thesis.

All the proposed approaches have been evaluated and verified for feasibility using real-world large data set. The evaluation results of these experimentations prove very promising for the type of large data networks considered.

Keyword: Community analysis, community detection algorithms; decentralized clustering algorithm; networks; graph; distributed algorithms.

Acknowledgements

Praise be to Allah, the most gracious and the most merciful. Without his blessing my accomplishment would never have been possible.

This thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study. It is a pleasure to convey my gratitude to them all in my humble acknowledgment.

First of all, I would like to thank my research supervisor, Dr. Princy Johnson for giving me the opportunity to be part of her research group and for providing me the right balance of guidance and independence in my research. Without her guidance, vast knowledge and persistent help this achievement would have been faraway.

My thanks also go to my second and third supervisors Dr. Trung Nguyen and Dr. Martin Randles for providing invaluable suggestions and necessary information regarding this research from different views. Their constructive feedback and suggestions greatly improved this thesis.

Special thanks to my office mates for creating a good atmosphere, interesting discussions, and great technical chats making the Ph.D. period such a friendly environment.

Last but not least, truly on top of all, I am heartily thankful to my parents and all my family members. Without their prayers, support, trust, and understanding I would not have been able to seek a single word.

Amhmed

Publications

International Conferences Publications

Bhiih, A.A., Johnson, P. and Randles, M., 2016, June. Diversity in Smartphone Usage. In *Proceedings of the 17th International Conference on Computer Systems and Technologies 2016*, Palermo, Italy (pp. 81-88). ACM.

Eiza, M.H., Randles, M., Johnson, P., Shone, N., Pang, J. and **Bhiih, A.**, 2015, October. Rail Internet of Things: An Architectural Platform and Assured Requirements Model. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference*, Liverpool, UK (pp. 364-370). IEEE.

Bhiih, A.A., Johnson, P. and Randles, M., October 2017, June. Decentralized Iterative Community Clustering Approach (DICCA). IEEE 28th Annual International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC, Montreal, QC, Canada (pp.1-7). IEEE.

Journals Publications

Bhiih, A.A., Johnson, P. and Randles, M., (2015), EM Clustering Approach for Multi-Dimensional Analysis of Big Data Set. *International Journal of Engineering Research & Technology (IJERT)*. ISSN: 2278-0181, Vol. 4 (pp.553-557).

Contents

ABSTRACT	1
Acknowledgements	iii
Publications	iv
Contents	v
List of Tables	x
List of Figures	xi
List of Symbols and Abbreviations	xiii
CHAPTER 1 INTRODUCTION	1
1.1 Introduction	1
1.2 Impact of the Research and its Impact	3
1.2.1 Social networks	3
1.2.2 Impact on WWW	4
1.2.3 Routing in Ad-hoc and Wireless Sensor Networks	4
1.3 Research Challenges	5
1.4 Aim and Research Objectives	7
1.5 Scope of Research	9
1.6 Contributions of the research to state of the art	9
1.7 Thesis Structure.....	10
CHAPTER 2 LITERATURE REVIEW	12
2.1 Basic concepts of graph theory	12
2.2 Community Detection Algorithms	16
2.2.1 Link-Centrality-Based Algorithms	17
2.2.2 Modularity Optimisation Algorithms	18
2.2.3 Spectral Algorithms	21
2.2.4 Random-Walk-Based Algorithms.....	22
2.2.5 Information-Based Algorithms	22
2.3 Parallelisation of Centrality Algorithms	23
2.3.1 MapReduce	24

2.4	Summary	26
CHAPTER 3 NETWORK MODELS AND STATISTICAL METHODS FOR COMPARISON OF NETWORKS.....28		
3.1	Topology of Real Networks	28
3.1.1	The Small-World effect	28
3.1.2	Degree Distribution.....	29
3.1.3	Community Effects.....	29
3.2	Overview of Validity Evaluation	31
3.2.1	Cluster Quality Metrics.....	31
3.2.1.1	Coverage.....	31
3.2.1.2	Conductance	32
3.2.1.3	Modularity	33
3.2.2	External Evaluation Metrics	33
3.2.2.1	Rand Index.....	33
3.2.2.2	Adjusted Rand Index	34
3.2.2.3	Normalized Mutual Information (NMI)	35
3.2.3	Computational complexity.....	36
3.2.4	Visualization for Cluster Validation	36
3.2	Artificial Networks.....	37
3.2.1	Girvan and Newman (GN) Benchmark Networks.....	38
3.2.2	LFR Benchmark Networks	39
3.3	Research Methodology.....	40
3.4	Summary	42
CHAPTER 4 DECENTRALIZED ITERATIVE COMMUNITY CLUSTERING APPROACH (DICCA).....44		
4.1	Related Literature and Previous Studies	44
4.2	Description of the Proposed DICCA.....	46
4.3	Experimentation and Results.....	52
4.3.1	LFR Synthetic Dataset (network)	52
4.3.2	Evaluation Metric.....	52
4.3.3	Parameter Selection Strategy	53
4.3.3.1	Time to Live	53

4.3.3.2	Threshold Value	57
4.3.3.3	Automated Identification of Appropriate Threshold Value	59
4.4	Analysis of Results and Discussion	63
4.4.1	Results for Each Iteration of Clustering.....	64
4.4.2	Clustering Results for Increasing Network Size.....	66
4.4.3	Evaluating Repeatability of the Algorithm’s Performance.....	66
4.4.4	Evaluation of Message Complexity of the DICCA Algorithm.....	67
4.4.5	Evaluation of Clustering Performance Using Mixing Parameter	69
4.4.6	Evaluation of Clustering Performance Using Adjacency Matrix Representations 70	
4.5	Summary	73
 CHAPTER 5 PARALLEL DECENTRALIZED ITERATIVE COMMUNITY		
CLUSTERING APPROACH (PDICCA)		
5.1	Introduction	74
5.2	Description of the Proposed PDICCA Approach.....	76
5.2.1	Framework of the PDICCA Approach	76
5.2.2	Partitioning of the Network Nodes Set	80
5.2.3	How to Calculate the Parameters	81
5.3	Matlab Implementation of PDICCA Approach for Distributed Memory Systems... 81	
5.4	Parallel Algorithms Using MapReduce Model	83
5.4.1	Description of Algorithm in MapReduce Model	84
5.5	Analysis of Results and Discussion	85
5.5.1	Environment Setup.....	85
5.5.2	Experimental Evaluation.....	85
5.5.2.1	Horizontal Scalability in Relation to the Number of Parallel Cores	85
5.5.2.1.1	Quality.....	85
5.5.2.1.2	Message Complexity of the PDICCA Algorithm	86
5.5.2.2	Clustering Results for Increasing Network Size.....	89
5.5.2.2.1	Quality.....	89
5.5.2.2.2	Evaluating Repeatability of the Algorithm’s Performance	89
5.5.2.2.3	Evaluation of Complexity of the PDICCA Approach	90
5.5.2.3	Evaluation of Clustering Performance Using Mixing Parameter	91

5.6	Summary	92
CHAPTER 6 A PRE-PROCESSING APPROACH FOR ROBUST COMMUNITY CLUSTERING TECHNIQUES BASED ON COLLABORATIVE INFORMATION SOURCES		
		94
6.1	Introduction	94
6.2	Related Literature and Contribution.....	96
6.3	Experimental Datasets.....	97
6.4	Correlation Analysis.....	98
6.4.1	Shared Neighbours.....	98
6.4.2	Correlation of Communities and Attributes.....	99
6.5	Description of the Proposed Approach	103
6.5.1	The Parameter Learning Phase	106
6.5.1.1	Attribute Similarity Metric	107
6.5.1.2	Effect of α and β on the Quality of Community Structure	111
6.5.2	Information Aggregation Phase	115
6.6	Experimentation and Results.....	115
6.6.1	Experimental Setup.....	115
6.6.2	Experimental Results and Discussion.....	116
6.6.2.1	Evaluation of Attribute Weighting Method.....	116
6.6.2.2	Model Performance	120
6.6.2.2.1	Number of Community Clusters	120
6.6.2.2.2	Modularity	123
6.7	Summary	124
CHAPTER 7 A CASE STUDY IN TELECOMMUNICATION INDUSTRY OF SMARTPHONE USAGE.....		
		126
7.1	Introduction	126
7.2	Related Literature.....	127
7.3	Proposed Methodology	129
7.3.1	Datasets.....	129
7.3.2	Characterisation Methodology.....	130
7.4	Results, Analysis and Discussion.....	131
7.4.1	Calls via Time.....	131

7.4.2	Text Messaging via Time.....	132
7.4.3	Mobile Data Traffic Distribution via Time.....	132
7.4.4	Percentage of Calls, Text Messaging and Mobile Data Traffic Over the Days of Week	132
7.4.5	Percentage of Calls, Text Messaging and Mobile Data Traffic via Different Time Zones.....	133
7.5	Summary	133
CHAPTER 8 CONCLUSION AND FUTURE WORK.....		137
8.1	Summary of Contributions	137
8.2	Recommendations and Future Works	140
REFERENCES.....		143
APPENDIX.....		153
Appendix A: Additional Results		153
A.1	Additional Results for DICCA described in chapter 4.....	153
A.2	Additional Results for PDICCA described in chapter 5.....	155
A.3	Additional results for pre-processing approach described in chapter6.....	156
Appendix B: Permission to Reuse IEEE Material		168

List of Tables

Table 4.1 Comparison of the algorithms.....	46
Table 4.2 The experimental results obtained by the DICCA algorithm on a small network of 50 nodes	58
Table 4.3 The LFR benchmark graph parameters.	63
Table 5.1 Comparison between DICCA and PDICCA.....	75
Table 5.2 Comparison with message exchanged locally in hosts and messages exchanged between master and hosts.....	88
Table 5.3 Experimental results of the PDICCA approach for increasing number of nodes in the network.....	91
Table 6.1 Results for four dataset	119

List of Figures

Figure 1.1 A simple graph with three communities that are represented by different colours.	3
Figure 2.1 An example of unweighted undirected graph and its adjacency matrix.....	14
Figure 2.2 Architecture of MapReduce framework (Dean and Ghemawat, 2008).....	25
Figure 3.1 The way of benchmarking the algorithm using a network with ground-truth communities.....	33
Figure 4.1 Illustrates the concept of the algorithm	50
Figure 4.2 Performance of the DICCA algorithm using different TTL values.....	54
Figure 4.3 Comparison between computing time and the message complexities over different TTL values.....	54
Figure 4.4 Performance of DICCA algorithm using adaptive termination via different TTL values	57
Figure 4.5 Community detection result for a small network with 50 nodes as extracted by the proposed DICCA algorithm using TTL=3 and with different threshold values. (a) threshold value =0, (b) threshold value =0.1, (c) threshold value =0.2, (d) threshold value =0.3, (e) threshold value =0.4, (f) threshold value =0.5, (g) threshold value =0.6, (h) threshold value =0.7, (i) threshold value >=0.8, (j) ground truth clusters, (k) Modularity via threshold value. The values of the other parameters were fixed: $\gamma=2$, $\beta=1$	60
Figure 4.6 The community structures of the ground truth communities and those extracted by the proposed DICCA algorithm on the LFR benchmark networks with 50 nodes using TTL=3 and threshold value =0.223xt ¹	62
Figure 4.7 Community detection result for each iteration on a small network of 50 nodes using the proposed DICCA algorithm with TTL=3, threshold value =0.223 *t, and $\gamma=1$, $\beta=2$	65
Figure 4.8 NMI, Q-DICCS and Ground truth Q scores (y-axis) as number of nodes (x-axis) changes.....	66
Figure 4.9 Standard deviation of final modularity/NMI with network sizes.....	67
Figure 4.10 Total number of exchanged messages (y-axis) as number of nodes (x-axis) changes.....	68
Figure 4.11 Percentage of Message exchanged per each iteration. (a) number of node in the network is 500, (b) number of node in the network is 1,000.....	68
Figure 4.12 Performance of the proposed algorithm using Mixing parameter. (a) Number of node in the network is 500, (b) Number of node in the network is 1,000.	69
Figure 4.13 Spy plot for the connections of the nodes.....	72
Figure 5.1 Framework of the PDICCA approach.	78
Figure 5.2 Examples of eight nodes with two community clusters	80
Figure 5.3 Parfor mechanism.	82
Figure 5.4 PDICCA workflow and architecture.	84
Figure 5.5 NMI, Q-PDICCS and Ground truth Q scores (y-axis) as number of workers (x-axis) changes number of nodes: (a) 500 (b) 1,000.....	86
Figure 5.6 Number of Message exchanged in each iterations and for each worker with respect to the number of workers varied from 2 to 4 (a, b, c) for number of nodes 500 (d, e, f) for number of nodes 1,000.	87
Figure 5.7 Average percentage of Message exchanged per each iteration with number of cores varied from 1 to 4 workers (a, b, c) network size 500 (d, e, f) network size 1,000.....	88

Figure 5.8 NMI, Q-DICCS and Ground truth Q scores (y-axis) as number of nodes (x-axis) changes.....	89
Figure 5.9 Standard deviation of final modularity/NMI with network sizes.....	90
Figure 5.10 (a) Total number of exchanged messages (y-axis) as number of nodes (x-axis) changes. (b) .Running-time scalability of proposed algorithm in seconds.....	91
Figure 5.11 Performance of the proposed algorithm using Mixing parameter μ . (a) Number of node in the network is 500, (b) Number of node in the network is 1,000.....	92
Figure 6.1 Visualization results of node clustering coefficient for subset of four datasets (should be viewed in colour).....	99
Figure 6.2 Visualization of correlations between attributes and communities for Reed dataset. (a) Communities based on attributes: nodes are coloured the same if they have the same value for the corresponding attribute; nodes with a missing value for an attribute are white. (b) Communities based on community clustering algorithm: nodes are coloured the same if they belong to the same community.....	101
Figure 6.3 Agreement of different community detection algorithms with each attribute, for a subset of four datasets.....	102
Figure 6.4 System architecture for the proposed approach.....	104
Figure 6.5 (a-b) Modularity value achieved by four community clustering algorithm dataset using different value of α and β on: (a) Caltech36 (b) Reed98 dataset.....	113
Figure 6.6 (c-d) Modularity value achieved by four community clustering algorithm dataset using different value of α and β on: (c) Harvord76 (d) Vassar85 dataset.....	114
Figure 6.7 Attribute weights for four datasets.....	117
Figure 6.8 Robustness of weighting method to the edge removal.....	118
Figure 6.9 Number of community clusters for: (a) Caltech36 university dataset, (b) Reed98 university dataset (c) Haverford76 university dataset, (d) Vassar85 dataset.....	121
Figure 6.10 Average Community size for: (a) Caltech36 university dataset, (b) Reed98 university dataset (c) Haverford76 university dataset, (d) Vassar85 dataset.....	122
Figure 6.11 Modularity index vis missing edges for: (a) Caltech36 university dataset, (b) Reed98 university dataset (c) Haverford76 university dataset, (d) Vassar85 dataset.....	124
Figure 7.1 Data mining process	131
Figure 7.2 Number of calls via hours of day.....	131
Figure 7.3 Number of text messages as a function of time of day.....	132
Figure 7.4 Mobile data traffic as a function of time of day	132
Figure 7.5 Percentage of calls, text messaging and mobile data traffic via days of week	133
Figure 7.6 Percentage of calls, text messaging and mobile data traffic via different time zones	133

List of Symbols and Abbreviations

A	Adjacency matrix
A_{ij}	Connection weight of node pair $i; j$ given adjacency matrix A .
AR	Rand Index
ARI	Adjusted Rand Index
$A_{sim}(i,j)$	The attribute similarity between a pair of nodes (i, j) in network $G = (V, E, A)$.
C	A partition of V
CDC	Connectivity-based Decentralized Node Clustering scheme
C_i	Clustering coefficient of a given node i
CPU	Central Processing Unit
CSV	Comma separated values
dc	The total degree of nodes in C .
DICCA	Decentralized Iterative Community Clustering Approach
DiDiC	Distributed Diffusive Clustering algorithm
E	Set of edges (links)
e_{ij}	Fraction of edges in the network that connect nodes in group i to those nodes in group j
FA	Fast Modularity algorithm
GN benchmark	Girvan and Newman benchmark
$G = (V, E, A)$	Graph/network consisted of set of nodes (V) and set of edges (E) and each node $V_i \in V$ is associated with an attribute vector (A_i^1, \dots, A_i^d) . Where d is the attribute dimension and i represents the node ID
$G(V, E)$	Graph/network consisted of set of nodes (V) and set of edges (E)
GPU	Graphics Processing Unit
$H_{sim}(i, j)$	Hybrid similarity matrix
J	Jaccard similarity
k	The number of clusters
\bar{K}	Mean degree of each node
k_i	Degree of node i
k_{max}	Maximum degree

LW_d^i	The local attribute weight for cluster i with N nodes each with d attributes
LA	Louvain algorithm
L_c	The total number of edges joining nodes of cluster C
LE	Leading eigenvector
L_i	Number of edges between neighbours of node i
M	Number of edges $ E $ in the network.
μ	Mixing parameter
n	Number of nodes $ V $ in the network.
$N_{\Delta}(i)$	Number of triangles involving node i
$N_3(i)$	Number of connected triples having i as the central node.
$Nbr(V_i)$	Neighbours of node V_i
NMI	Normalized mutual information
OnID	Originator node ID
PC	Personal computer.
P_k	Degree distribution
PCT	Parallel Computing Toolbox
PDICCA	Parallel Decentralized Iterative Community Clustering approach
Q	Newman-Girvan modularity.
RW	Random Walk
$SN(i,j)$	The shared neighbours between node i and j .
$SN_{sim}(i,j)$	Shared neighbours similarity between nodes i and j .
$SN_{sim}(i,j)$	The shared neighbours similarity between nodes i and j
t	the iteration number
T	Transitivity
TTL	Time to Live
V	Set of nodes (vertices)
VLSI	Very Large Scale Integration
$W(V_m, V_i)$	The weight of the edge between V_m and V_i
WMsg	Message Weight
WSNs	Wireless Sensor Networks

WWW	World wide web network
β	Exponent of community size distribution
γ	Exponent of the degree distribution
α	Exponent of the power-law degree distribution
$\Gamma(i)$	Neighbourhood of node i
$\delta(\cdot)$	Kronecker delta function, $\delta(x; y) = 1$ if and only if all variables in the argument are equal and $\delta(x; y) = 0$ otherwise.
$\Phi(C_i)$	The conductance of given cluster C_i

CHAPTER 1

INTRODUCTION

1.1 Introduction

Many systems in the world can be represented as networks (also referred to as graphs in much of the mathematical literature) composed of nodes (vertices) and links (edges) in which network links represent relationships between the interrelating parts (nodes) of the systems. Examples include technological networks such as the Internet (Faloutsos, Faloutsos and Faloutsos, 1999) and the World Wide Web (WWW) (Albert, Jeong and Barabási, 1999), biological networks e.g., Neuronal networks, metabolic networks, protein-protein interaction networks and food webs (Vocaturo and Veltri, 2017), and distribution networks (Newman, 2003) like postal delivery routes, citation networks, social networks, organisational networks (Newman, 2003) and even political elections (Adamic and Glance, 2005) etc.

Recently, it has become common to analyse interactions in the real-world by looking at the networks that underlie these interactions (Chen, Zaiane and Goebel, 2009). However, real-world networks are not random networks, they usually exhibit inhomogeneity and reveal a high level of order and organisation (Mahata and Patra, 2016). An interesting feature that real-world networks usually present is the community structure property, under which the topology of network is organised into modules commonly called communities or clusters (Fortunato, 2010).

The process of discovering the cohesive groups or clusters in the network is known as community detection (Bedi and Sharma, 2016), it is also known as the graph partition problem in modern graph theory, and as the graph clustering or dense subgraph discovery problem in the graph mining area (Wang et al, 2015).

The problem of community or graph clustering is not well defined and the concepts of community do not have a universally accepted definition. Highlighting the difficulties of the problem, in his recent work, Fortunato stated that “the definition often depends on the specific system at hand and/or application one has in mind” (Fortunato, 2010). Considering social network as an example, community can be defined using many natural properties. Whether the nodes representing people in a community should know each other, the community should have a high edge density or each detectable community ought to have a unique identity (Shah and Zaman, 2010).

Informally, a cluster is usually defined as a set of entities that are closer to each other than with the rest of the entities in the data set (Jain, Murty and Flynn, 1999). The notion of closeness is based on a similarity measure that is usually defined with the use of a mathematical objective function. The task of clustering is also referred to as “unsupervised learning where the aim is to group together similar data set without resorting to any a priori knowledge about the clusters (Schaeffer, 2007). In the case of networks, the similarity is usually measured either based on the structural similarity which considers the topological features or the attribute features related to the nodes or edges of the graph, or both of them (Malliaros and Vazirgiannis, 2013).

There are several definitions of the community detection problem. In general, the community detection algorithms aim to divide a network into sub-communities. The general principle on which most community definitions are based is the tendency for the nodes to divide into clusters with dense connections within clusters and only sparser connections between them (Newman, 2004a). However, communities may overlap as nodes belong to multiple clusters simultaneously. The overlapping community is very common in real-world networks for example, in a social network, a person may belong to more than one social group such as friend group and family group which are known as overlapping nodes (Amelio and Pizzuti, 2014). More detailed definitions of community are presented in another work (Fortunato, 2010).

Figure 1.1 shows a small network of 12 nodes that illustrates this idea of network structure. The network has three communities denoted by the circles in which a set of nodes are densely connected internally and loosely connected to the rest of the network.

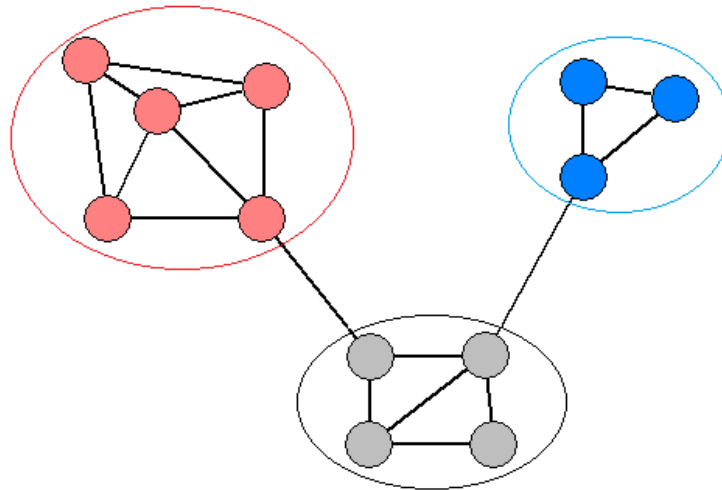


Figure 1.1 A simple graph with three communities that are represented by different colours.

1.2 Impact of the Research and its Impact

1.2.1 Social networks

Community structure is a common and important topological characteristic of many real-world complex networks. Nodes belonging to a tight-knit community are more than likely to have other properties in common (Danon et al, 2005). The determination of communities in the networks can help to better understand the structural makeup of the networks, provide powerful insights about the structure of networks, and help analyse complex phenomena at different scales (Orman, Labatut and Cherifi, 2011; Borgatti, Everett and Johnson, 2013). Thus, this research topic has applications in many fields such as biology, social science, physics, computer science, business science, etc. (Schaeffer, 2007; Orman, Labatut and Cherifi, 2011).

In social networks, for example, analysis of community detection is extremely useful in the context of many applications, including customer segmentation, vertex labelling,

recommendations and link inference (Khatoon and Banu, 2015). Also could be used to estimating unknown features of users in social networks. If a given user does not give a certain piece of information (like the school he/she went to), but a reasonable number in his/her community do, the missing information can be imputed with a reasonable degree of confidence.

1.2.2 Impact on WWW

Community structure is important not only on social networks, but also on various other networks. For the famous example of the Internet, determination of community structure can address questions such as, how to route data as packets in an efficient way, how to reduce the time consumption for such traffic and what is the fast and safe path to consider reaching the destination etc. It can go further in depth, by elucidating questions like how computer viruses are spreading through the Internet, and what mechanisms they follow to hit organisations etc. Also in dark networks, community structure can reveal the hidden relationships between individual terrorists and help develop effective disruptive strategies. (Warnke, 2016). Similarly, in the case of the world wide web (WWW), pages related to the same subject are typically organised into communities, so that the identification of these communities can help the task of seeking for identifying the category of the network as well as understanding its dynamic evolution and organisation (Costa et al, 2007).

1.2.3 Routing in Ad-hoc and Wireless Sensor Networks

Clustering without global knowledge is an important technique in mobile ad-hoc and sensor networks (Gehweiler and Meyerhenke, 2010) for the improvement of certain management e.g. energy consumption and communication tasks.

In wireless sensor networks (WSNs), nodes are usually consist with limited and non-rechargeable energy resources. Thus in WSNs, energy consumption is the most critical problem and large number of clustering routing protocols have been developed for WSNs to reduce

communications, efficiently optimize the energy of sensor nodes, organize messages among the cluster head and their node members and optimize the network life-time (Liu, 2012).

In clustering routing protocols, the sensing field of sensor network is divided into number of clusters where each cluster has a leader called cluster head. The cluster head collects the data from its node members and transfer it to the destination (base station). Yu and Chong (2005) reported that the cluster structure is an effective topology that could provide many benefits in the context of wireless sensor networks (WSNs). It could be used to increase the system capacity by spatial reuse of resources. Furthermore, it improves routing performance, since the set of cluster-heads and cluster gateways can normally form a virtual backbone for inter-cluster routing, and thus the generation and spreading of routing information can be restricted to this set of nodes. Additionally, they stated that the cluster structure makes an ad hoc network appear smaller and more stable in the view of each mobile terminal, this is because in WSNs when a mobile node changes its attaching cluster, only mobile nodes residing in the corresponding clusters need to update the information.

For more information, interested readers may refer to Yu and Chong's survey (Yu and Chong, 2005).

1.3 Research Challenges

In recent years, the problem of network clustering has received growing attention as an important analytical technique and has been actively investigated in a variety of fields, from computer science and statistical physics (Newman, 2004b; Newman and Girvan, 2004) to data mining (Moghaddam et al, 2010). Therefore, a rich and diverse list of methods and algorithms has been generated.

In the current Big Data era, the amount of generated data is huge, existing in various formats, from a continuously increasing number of sources. The real-world networks can be very large

in size, even reaching billions of nodes. However, most of the community detection algorithms in the literature are classified as global algorithms, which require access to the entire information of the network and are designed to work on a single machine.

As the data size is scaling up, the need for computing power is exponentially increasing. In many such situations, it has become difficult for the stand-alone community detection algorithms to find communities in large-scale networks (Li et al, 2015) and the required processing power far exceeds the processing capabilities of single machines. However, most of the existing community detection algorithms cannot be directly parallelised. Furthermore, in many such cases the large-scale data set does not fit into the main memory of a single machine and needs to be distributed among several machines. These demanding requirements make existing community clustering algorithms even more limited than before, and so more powerful and scalable clustering tools for big data analysis seem to be in urgent need.

Additionally, in many real-world networks, node attribute is also available in addition to topology information. It is pointed out that nodes containing similar content of communication are much likely to belong to the same community (McPherson, Smith-Lovin and Cook, 2001; Traud et al, 2011). Traud et al (2011) show that a set of nodes' attributes can act as the primary organising principle of the communities. An overwhelming majority of conventional approaches to community detection focus on topology information and largely ignore the attribute information. However, the collected topology information for networks is usually noisy when there are missing edges. This makes the task of community detection for incomplete networks very challenging.

To summarise, Big data exhibits different characteristics such as 'volume, variety, velocity, value, thus it is very difficult to analyse Big data and obtain information with traditional techniques (Hu et al, 2014).

Given these scenarios, there is the emergence of a new research direction to develop a powerful and scalable community clustering method for big data analysis, which will make use of the relationship between the attribute and link information to improve the robustness of the existing community clustering methods in unreliable environments (incomplete or noisy networks).

1.4 Aim and Research Objectives

The main goal of this thesis is to design and implement novel techniques and algorithms for the problem of clustering and community detection in large and undirected networks. In the light of the above discussed research challenges, the main objectives and motivations of this research work are summarised below:

1. To design and implement an efficient community-detection approach that could work at the local level and does not require any global knowledge of the network.

As the networks being operated on become larger and larger, the ability to process them in the main memory of a single machine becomes impractical due to both time and memory constraints. Moreover, community detection algorithms are often computationally expensive and are not scalable to large networks with hundreds of millions or even billions of nodes and billions of edges.

The above issues motivated me to design, implement, and evaluate an efficient community-detection solution for large-scale networks. More specifically, the proposed approach works at the local level and does not require any global knowledge of the network. From the heuristic point of view, it is worth noting that the optimisation of global clustering methods, when only restricted to the local knowledge, is more difficult. That is why most of the existing approaches and algorithms make use of global knowledge.

2. To extend the proposed approach for large-scale networks to work in parallel and in a distributed fashion.

Being a localised algorithm, it can be run in parallel or in a distributed fashion among clusters when the size of the input network or the computation complexity is beyond the resources of a single computer.

3. To design and implement a community clustering approach considering both attribute information and topological structure information to improve the performance of existing community detection algorithms.

Since in many real-world networks, the nodes and links in the networks may contain attribute information, this attribute information has important significance in completely presenting the community structure of the network and could improve the robustness of community detection algorithms in unreliable environments.

4. To analyse a set of real-world mobile phone usage data as an exploratory step for viability to apply the algorithms developed in this thesis.

The smart phones in the telecommunication industry generate a massive amount of data. These data usually include call details, data and network details. The amount of data is so big that manual management and analysis of these data is almost impossible. From this perspective to explore the viability of applying the proposed method and algorithms to analyse the big data sets generated by smart phones. A real-life big data (Device Analyzer) set from Cambridge Laboratories is used for this proposed objective.

5. To propose a set of broad guidelines and future design from the understanding gained.

Under this objective, the potential usage of the developed approaches proposed in this thesis will be demonstrated. Also, recommendations, guidance information, and suggestions to improve the effectiveness of the developed algorithm will be made.

1.5 Scope of Research

This thesis studies in the scope of community detection in big networks. In other words, the main goal of this thesis is to design and implement novel techniques and algorithms for the problem of clustering and community detection in large and undirected networks. The approaches proposed in this thesis all assume that the given network structure is needed to be divided into communities in such a way that every node belongs to one of the communities (non-overlapping communities). Although doing some modifications of the proposed approaches can achieve overlapping communities, the focus of this thesis is on non-overlapping communities.

1.6 Contributions of the research to state of the art

This thesis aims to design and implement methods for the problem of extracting non-overlapping communities in large networks. However, since the global community clustering approaches demand shared memory to access global information, they are inappropriate for this goal. Thus, in this work attention is given to the local community clustering as it is more accessible for parallelization.

The following summary provides a short overview of the four key contributions of this work that address all of the challenges introduced in the previous sections:

1. A novel Decentralized Iterative Community Clustering Approach (DICCA) to extract an efficient community structure for large networks is proposed. An important property of this approach is its ability to cluster the entire network without the global knowledge of the network topology. This ability means that the entire network does not need to be loaded into one memory and DICCA could be easily adapted to run in parallel on as many processors as available to find community clusters in big networks. This cannot be done in the majority of the existing community detection algorithms as they

implicitly assume that the entire structure of the big network is known and is available. Another perspective of DICCA approach is reducing the problem size by aggregating the nodes in the network, allowing the approach to cluster the large-scale data set efficiently.

2. A Parallel Decentralized Iterative Community Clustering Approach (PDICCA), which does not require any global knowledge of the graph topology is proposed. PDICCA is a distributed memory parallel processing approach that transforms the serial steps of DICCA approach into parallel tasks. It is scalable and will work with a range of computer architecture platforms (e.g. cluster of PCs, multi-core distributed memory servers, GPUs).
3. A pre-processing approach for existing community detection algorithms is proposed to improve the robustness of community detection algorithms in unreliable environments. The proposed approach is applicable to the existing weighted community detection algorithms and it seeks to improve their performance by considering attribute information, shared neighbours information and connectivity between nodes in the network. Therefore, if either attribute information or topological structure information is noisy or missing, the other could make up for it.
4. Using a set of real-life android smartphone usage datasets, the different features of mobile phone usage is analysed.

1.7 Thesis Structure

The thesis contains eight chapters, which are organised as follows. The **present chapter** gives an overall picture of the thesis, highlights the importance of the field of community detection in the networks and states the challenges, aim, objectives and the contributions of the research. The rest of the thesis is organised as follows:

Chapter 2 gives some basic definitions of graph theory, which are used in further chapters. Furthermore, the literature review of state-of-the-art community detection algorithms and related work in the area of parallelisation techniques for the community detection algorithms are also discussed.

Chapter 3 presents some specific structural properties and models of real networks. Additionally, the current work available in literature for models that generate synthetic networks with community structures along with the most popular quality metrics for assessing the network clustering results are discussed.

Chapter 4 addresses the first technical objective of the research. It gives a detailed description of my proposed Decentralized Iterative Community Clustering Approach, for detecting community and then the effectiveness and efficiency of the DICCA approach is evaluated.

Chapter 5 centres around the design and implementation of the parallel framework version of DICCA approach named PDICCA. In this chapter, the principle and implementation of the proposed PDICCA approach is detailed and its performance is evaluated.

Seeking to improve the robustness of existing community detection algorithms rather than looking to identify communities in the network based just on topological structure information, a new pre-processing approach that considers attribute information, shared neighbours information and connectivity between nodes in the network is presented in **chapter 6**. **Chapter 7** shows the data analysis of the datasets from the real-world telecom network.

Finally, **chapter 8** concludes the research activities within this thesis by summarising the contributions and proposing a set of possible suggestions for future work.

CHAPTER 2

LITERATURE REVIEW

This chapter introduces some fundamental concepts that are widely used throughout this thesis, and reviews existing work on the community clustering and distributed techniques. It starts with a short introduction into the basics of graph theory, including the concepts required to understand further chapters. This is followed by a discussion of the definitions and concepts around community clustering. Then a detailed literature survey on the state-of-the-art in community approaches and the parallelisation techniques for extracting network clusters is presented.

2.1 Basic concepts of graph theory

Many practical problems in various fields of study such as scientific computing, data analysis etc, can be modelled in their essential form by graphs and solved using appropriate graph algorithms. In graph theory, a simple graph $G = (V, E)$ is defined as an abstract representation of a set of nodes (or vertices) $V = \{1, \dots, n\}$ and a set of edges (or links) $E = \{(i, j) | i, j \in V\}$ which connect pairs of nodes together. A pair (i, j) belongs to E if there is an interaction between the nodes i and j and the cardinality of the set E . The number of nodes in the graph is $n = |V|$ and the number of edges $m = |E|$. In some graphs it is possible to find an edge that connects a node to itself, $(i, i) \in E$, it is called a **self-loop** (Silva and Zhao, 2016).

The edges in the graph can be assigned with a weight, which represents the strength of connection between two nodes; in this case, the graph is called a weighted graph. If each edge has unit weight, the graph is called an unweighted graph (Silva and Zhao, 2016). Considering the nature of the edges, the graphs can be classified into two: undirected and directed graph. A graph is called directed (also referred to as digraph) if the orientation of the edges is important for the task (Silva and Zhao, 2016). A directed graph $G = (V, E)$ consists of a non-empty set of

nodes V and a set of directed edges E . Each edge $e:(u, v)$ of E is specified by an ordered pair of nodes (u, v) and comes out from node u , namely the origin (or tail), and reaches a destination v (or head).

Directed graphs arise in many real-world applications such as the web graph whose node represents a web host and each directed edge represents the hyperlinks. These hyperlinks are one-way from web pages on the source host to web pages on the destination host (Canright and Engø-Monsen, 2008). On the other hand, in undirected graphs, the edges have no orientation and the graph has edges that represent symmetric relationships in which whenever the edge (u, v) exists in an undirected graph then so does the edge (v, u) (Costa et al, 2007). For example, in friendship networks where each relationship is considered reciprocal in the sense that if you are friends with someone, then they are friends with you.

From the mathematical point of view an undirected unweighted graph $G = (V, E)$ can be represented by a matrix A called adjacency matrix $A \in \{0,1\}^{n \times n}$.

Definition 2.1 Adjacency Matrix: The adjacency matrix A of a graph $G = (V, E)$ is an $|V| \times |V|$ matrix, such that:

$$A_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise} \end{cases} \quad (2.1)$$

The adjacency matrix for an undirected graph is symmetric, This fact implies that $A_{(i,j)} = A_{(j,i)}$. However for a directed graph, the adjacency matrix may not be symmetric (Silva and Zhao, 2016).

Throughout this thesis, the terms “graph” and “network” are used interchangeably. In the same spirit, the data relationships that make up a graph are termed structure or topology of the network. Unless stated otherwise, a graph $G = (V, E)$ is unweighted, undirected and consists of

a set of nodes V and a set of E edges. Nodes and vertices convey the same type of information and are used interchangeably and the same principle applies to edges and links.

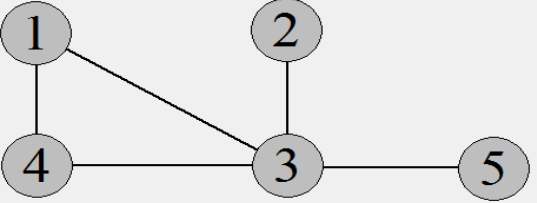
Labeled graph	Adjacency matrix
	$A = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$

Figure 2.1 An example of unweighted undirected graph and its adjacency matrix.

Definition 2.2 Degree of a node: The degree K_i of a node ‘i’ in undirected graph $G = (V;E)$ is equal to the number of edges connecting to node i (Silva and Zhao, 2016). Given an adjacency matrix A , the degree of node i is the sum of row entries corresponding to node i, which can be expressed as:

$$K_i = \sum_{j=0}^n A_{ij} \quad (2.2)$$

However, for directed graphs, the concept of degree is split into two categories: out-degree and in-degree.

Definition 2.3 In-degree and out-degree: The out-degree of a node ‘i’ in a directed graph is the number of edges that leave the node i, and the in-degree is the number of edges that enter the node i (Silva and Zhao, 2016).

Definition 2.4 A completely connected (fully connected) graph: In undirected graph G the fully connected graph is a graph in which every pair of distinct nodes is connected by a unique edge. Thus the total number of edges in a completely connected graph with n number of nodes is equal to $n(n-1)/2$ (Tomassini, 2010).

Definition 2.5 A triangle: In graph $G = (V, E)$ a triangle (Δ) is a three node subgraph with $V = \{v_1, v_2, v_3\} \subset V$ and $E = \{(v_1, v_2), (v_2, v_3), (v_3, v_1)\} \subset E$ (Schank and Wagner).

Definition 2.6 A triple: In graph $G = (V, E)$ a triple $N_3(i)$ at node 'i', is a path length of two for which i is the centre node (Schank and Wagner). For undirected graph, the number of triples of node i is defined as:

$$N_3(i) = \binom{K_i}{2} = \frac{K_i [K_i - 1]}{2} \quad (2.3)$$

and the number of triples in graph G is defined as the summing of triples of all nodes in the graph:

$$N_3 = \sum_{i=1}^n N_3(i) \quad (2.4)$$

To illustrate the concept of triangle and triples, the network in Figure 2.1 has 1 triangle and 8 connected triples.

Definition 2.7 Reachability: In graph theory, reachability refers to the ability to get from one node to another within a graph. Given a graph $G(V, E)$, it is said that $V_2 \in V$ is reachable from $V_1 \in V$ if there is at least a walk that starts from V_1 and ends at V_2 (Silva and Zhao, 2016).

Definition 2.8 Homophily:

Apart from the previous patterns that concern network architecture, there are also some other patterns that relate to how links depend on other characteristics of nodes. For instance, if nodes are people, then they have some attributes such as age, gender, ethnicity, profession, political attitudes, their hobbies and so forth. In real-world networks, it has been shown that the similar nodes in terms of their characteristics tend to be more frequently linked to each other than to nodes that are less similar to themselves in characteristics. This is referred to as homophily, as originally named by Lazarsfeld and Merton (McPherson, Smith-Lovin and Cook, 2001; Jackson, 2010).

Definition 2.9 Hierarchical structure: Another important aspect related to community structure is the hierarchical organisation (multiscale or multilevel) exhibited in most real-world

networks in which communities contain smaller communities that may be further divided into sub-communities. (Fortunato, 2010)

2.2 Community Detection Algorithms

The problem of unveiling the community structure of a network is called community detection. Community detection is an active area of network science research and over the years, a wide variety of community detection algorithms have been proposed to find the communities in the network. Community detection is also named as graph partitioning in much of the literature (Aggarwal and Wang, 2010; Wang et al, 2015). It is tempting to suggest that this community detection and graph partitioning are really addressing the same question; in both, their aim is to identify groups of nodes in a network that are better connected to each other than to the rest of the network. However, it is very important to stress that the task of graph partitioning and community detection can be distinguished from one another based on whether the experimenter fixes the number and size of the groups or it is unspecified (Newman, 2010). Graph partitioning is the problem of partitioning a graph into a predefined number and size of clusters. It has been pursued particularly in computer science and related fields with applications in parallel computing and very-large-scale integration (VLSI) design. However, in the community detection, which has been pursued by sociologists and more recently by physicists and applied mathematicians, with applications especially to social and biological networks the number and size of clusters are unspecified. Furthermore, the goal in the former is usually to identify the best division of a network regardless of whether or not a good division existed. In case there are no good divisions exist, the least bad one will be done as a solution. On the other hand, in community detection, the algorithm only divides the network when good divisions exist and leave the network undivided in case there are no existing good divisions (Newman, 2010).

Community structure identification has been an important research topic in complex networks. Given the number and range of community definitions, it is not a surprise that the number of

methods proposed for detecting and revealing the community structures in networks are even larger. Furthermore, the community detection algorithms can be classified in different ways, and depending on the selected criteria, one algorithm can belong to more than one category. A brief summary of existing community detection algorithms is introduced in the sections below. The algorithms are classified based on methodological principles as presented in Orman, Labatut and Cherifi (2011) in which most of the existing community detection algorithms mainly fall into the following categories:

2.2.1 Link-Centrality-Based Algorithms

The centrality measures such as degree centrality (Silva and Zhao, 2016) and betweenness (Girvan and Newman, 2002) are used to rank how important an edge (or node) is in the structure of the network. Thus, the link-centrality-based algorithms are usually hierarchical divisive approaches that start with a single community comprising all the nodes of the network. Then repeatedly removing/cutting edges and dividing the network progressively into smaller and smaller disconnected subnetworks that are viewed as communities until further splitting is no longer worthwhile. The centrality measures are used for the selection of the links to be cut, which are links connecting the communities and not those within them (Orman, Labatut and Cherifi, 2011).

The first and most known algorithm using this approach is the Girvan-Newman algorithm introduced in Girvan and Newman (2002). The algorithm estimates the centrality of a link by considering the edge betweenness measure, which is defined as the number of shortest paths between pairs of nodes that go through an edge in a graph. The algorithm is based on the fact that edges connecting communities are expected to have high edge betweenness. Thus, by iteratively removing these edges, the network is separated into groups from one another and the underlying community structure of the network is revealed. Though the algorithm obtains good results, it is very slow and highly complex thus it is not well suited for very large networks.

2.2.2 Modularity Optimisation Algorithms

The most popular method widely used to find community in the network relies on the optimisation of a quantity called modularity. Modularity (Q) is a prominent measure for the quality of a community structure introduced by Newman and Girvan in (Newman and Girvan, 2004) and it has become a widely accepted quality of measure for community detection.

The general concept of modularity optimisation algorithms is to detect the best community structure in terms of modularity by searching over possible divisions of a network that have high modularity.

Definition 2.10 Modularity (Q)

Modularity is based on the idea that a random graph is not expected to have a cluster structure, so it quantifies the community strength by comparing the fraction of edges that fall within a community with the expected fraction value of the same quantity of edges failing at random.

Let e_{ij} be the fraction of edges in the network that connect nodes in group i to those nodes in group j , then the modularity score Q for a clustering is given by the following equation (Newman and Girvan, 2004):

$$Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] \quad (2.5)$$

Formally, modularity can be defined as (Fortunato, 2010):

$$Q = \frac{1}{2|m|} \sum_{ij} \left[A_{ij} - \frac{K_i K_j}{2|m|} \right] \delta_{c_i c_j} \quad (2.6)$$

Where A_{ij} is an element of the adjacency matrix, K_i is the degree of node i . m is the total number of edges in the network. $\delta_{c_i c_j}$ is the Kronecker delta symbol, which is equal to 1 if $c_i=c_j$ and 0 otherwise, and c_i is the label of the community to which node i is assigned.

The modularity can also be equivalently defined as (Fortunato, 2010):

$$Q = \sum_{c=1}^k \left[\frac{L_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right] \quad (2.7)$$

Here, k is the number of clusters, L_c the total number of edges joining nodes in community c and d_c is the total degree of nodes in c .

The higher the value of Q in the network, the better its community strength. Networks with high modularity have dense connections between nodes within the same communities and sparse connections between nodes from different communities. Thus, a Q value close to 0 indicates that fraction of edges within communities is no better than for a random case. Values other than 0 indicate deviations from randomness. However, Newman et.al reported that in real networks the modularity values typically fall in the range from about 0.3 to 0.7, and values 0.3 or more, usually indicate good divisions (Newman and Girvan, 2004).

Fortunato and Barthélemy (2007) pointed out that the modularity measure suffers from serious resolution limits, and claimed that the size of the detected community, by enforcing modularity optimisation Q , depends on the size of the whole network, which may fail to identify modules smaller than a certain size. The main reason is that the modularity index does not consider the information of the number of nodes in a community, and the choice of partition is highly sensitive to the total number of edges in the network.

However, despite the fact that modularity is subject to a resolution limit, it is still one of the most popularly accepted metrics for measuring the quality of community structure as well as an optimisation criterion used by some algorithms to identify communities in networks (Newman, 2016). In the following paragraphs, two modularity optimisation algorithms are considered in some detail.

Fastgreedy algorithm is an agglomerative hierarchical clustering method proposed by Newman (Newman, 2004b). The algorithm greedily maximises the modularity function Q , and

starts the process by assigning a different community to each node in the network. Then at each stage in the process, the pair of clusters that yields greatest increase of modularity or smallest decrease is merged until only one cluster remains containing all nodes in the network. The whole procedure can be represented by a dendrogram (hierarchical tree) that illustrates the order of the mergers. Cuts through the dendrogram at different levels give different partitions into communities. The optimal community cluster can be found by cutting the dendrogram at the level of maximum Q .

Louvain algorithm is a hierarchical agglomerative optimisation method proposed by Blondel et al and attempts to optimise the modularity of a partition of the network. The optimisation is performed in two steps that are repeated iteratively (Blondel et al, 2008).

This algorithm starts with each node in the network belonging to its own community. Then in the first step and for each node in the network, the algorithm uses the local moving heuristic to obtain an improved community structure by moving each node from its own community to its neighbours' community and evaluating the gain of modularity associated with the moving of the node. The node is then placed in the community for which the modularity change is the most positive. If none of these modularity changes is positive, the node stays in its original community. This process is applied repeatedly and sequentially for each node until all the nodes in the network are considered, and no further improvement can be achieved. This concludes the first step. The second step of the algorithm consists of building a new network from the communities discovered in the first step. Therefore, the individual nodes in the new network are the individual communities from the first step. In this new network, there will be an edge between two nodes if there were edges between the corresponding two communities in the previous step. The weights of those new edges are the sum of the weights of the edges between nodes in the corresponding two communities. The edges between nodes of the same community in the first step will lead to self-loops for this community node in the new network. Once the

second step is completed, it is possible to replay the first step and iterate again if necessary. The two steps repeat iteratively and stop when there is no more change in the modularity gain and consequently a maximum modularity is obtained.

2.2.3 Spectral Algorithms

The spectral algorithms are mostly based on the analysis of the eigenvectors of matrices derived from the networks and designed to find the partition minimising the links lying in between the node groups. **Leading eigenvector** is one of the effective spectral algorithms proposed by Newman (2006b). The algorithm is based on the spectral optimisation of modularity. Newman showed that the modularity could be expressed in terms of the eigenvectors of a characteristic matrix for the network, called modularity matrix, and therefore spectral techniques for the optimisation process could be applied. He exploits the spectral properties of the modularity matrix by using the leading eigenvectors (associated with the largest eigenvalues) of the modularity matrix to maximise the modularity in his proposed algorithm. The algorithm initially divides the network by assigning all the nodes into two communities according to the signs of the leading vector elements of the modularity matrix. The negative signs clustered in one group and positive signs in the other. The algorithm then runs recursively on each subnetwork to divide those parts, and so forth. At any stage when there is no division of a subgraph that will increase the modularity of the network the algorithm leaves the corresponding subgraph undivided. This happens when all the elements in the eigenvector of the proposed split subgraph have the same sign, and when the entire network has been decomposed into indivisible subgraphs the algorithm ends. For the interested readers, Newman (2006b) discusses the algorithm in more detail.

However, there are two drawbacks in the spectral algorithm described above. First, it only takes the leading eigenvector of the modularity matrix to generate the solution and ignores all the information provided by the other eigenvectors. Second, it splits a network into more than two

communities by recursive partitioning instead of getting all the communities directly in a single step (Chen and Hero, 2015).

2.2.4 Random-Walk-Based Algorithms

Random walk is a process of traversing nodes at random and it has been widely used to partition the network into communities. There are several algorithms which have been proposed in literature based on the random walk. An example includes **Walktrap** (WT) algorithm which is proposed by Pons and Latapy (2006).

The walktrap algorithm is based on the principle that random walks on a network tend to get “trapped” into densely connected parts defining the communities. In this method, the authors propose using a node similarity measure based on short walks to capture structural similarities between nodes instead of modularity to identify community via hierarchical agglomeration. The algorithm starts by assigning each node to its own community and the distance for every pair of communities is computed. Communities are merged according to the minimum of their distances and the process iterated. After $n-1$ steps, the algorithm finishes and gives a hierarchical structure of communities called a dendrogram. The best partition is then considered to be the one that maximises modularity.

2.2.5 Information-Based Algorithms

Information-Based algorithms are also known as compression-based approaches. These approaches use the concept of information theory to find community clusters in the network. They basically consider the community structure as a set of regularities in the network topology, which can be used to represent the whole network in a more compact way than the whole adjacency matrix (Orman, Labatut and Cherifi, 2012). **Infomap algorithm** is an example of information theoretic algorithms proposed by Rosvall and Bergstrom (2008). Infomap algorithm characterises the problem of finding the optimal community clustering in the network as the problem of finding the most compressed (shortest) description length of the

random walks on the network. It uses a random walk as a proxy for information flow in a network and minimises a map equation, which measures the description length of a random walker, over all the network clusters to reveal its community structure. To represent the community structure, the algorithm uses a two-level nomenclature based on Huffman coding: a level to distinguish communities in the network and the other to distinguish nodes in the community.

In practice, the random walker is likely to stay longer inside communities, therefore in the process of finding a community containing few inter-community links, only the second level is needed to describe its path, leading to a compact representation. However, even though Infomap is a competitive community detection algorithm and shows a very good performance across several benchmarks (Fortunato, 2010), it cannot handle big networks with millions and billions of edges that are becoming commonplace with the advent of Big Data (Bae et al, 2017).

For a more thorough discussion of community detection methods and algorithms and their principles, please refer to the work done by Fortunato who is one of the major authorities in the field of community detection (Fortunato, 2010) and Schaeffer (Schaeffer, 2007).

2.3 Parallelisation of Centrality Algorithms

Presently, the real-world networks are often complicated and accompanied by extremely large sizes. Using conventional algorithms to analyse the networks is almost impossible to process in a single machine and they usually require specialised processing methods, especially parallel ones. Furthermore, many data parallelisation methods are proposed to extend storage capabilities and to improve performance by distributing data and related tasks into disparate hardware (Hu et al, 2014). MapReduce (Dean and Ghemawat, 2008) is one of the most popular distributed computation frameworks that is being widely applied to large scale data-intensive processing.

2.3.1 MapReduce

MapReduce is a distributed computing model proposed by Google in 2004 for processing massive data sets with a parallel distributed algorithm using a large number of computers in an efficient and fault tolerant manner (Dean and Ghemawat, 2008). Nowadays, MapReduce is widely used as an efficient distributed computation tool in many applications e.g., search, clustering, analysis of social networks, log analysis and matrix multiplication to name but a few (Derbeko et al, 2016).

The computation of MapReduce takes a set of input key/value pairs, and produces a set of output key/value pairs. The computation of MapReduce is expressed as two functions written by the user: Map and Reduce. One iteration of map and reduce functions is called MapReduce Job. MapReduce computation could be simply described as the following steps (Dean and Ghemawat, 2008):

1. Input data is read from the disk and converted to Key-Value pairs.
2. The map function takes an input pair of data separately, processes it and produces a list of intermediate key/value pairs.

$$(Key1, Value1) \rightarrow list(Key2, Value2) \quad (2.8)$$

3. The reduce function takes intermediate *Key2* with a list of *Values* and processes them to form a new list of values.

$$(Key2, list(Value2)) \rightarrow list(Value3) \quad (2.9)$$

4. Once all input pairs have been processed, the output of the Reduce function is then written to the disk as *Key-Value* pairs.

MapReduce runs in a cluster of nodes; one node acts as a master node and the others act as workers. The master node is responsible for assigning tasks to idle workers whereas the worker nodes are responsible for running map and reduce tasks. A block diagram of the MapReduce framework is shown in Figure 2.2.

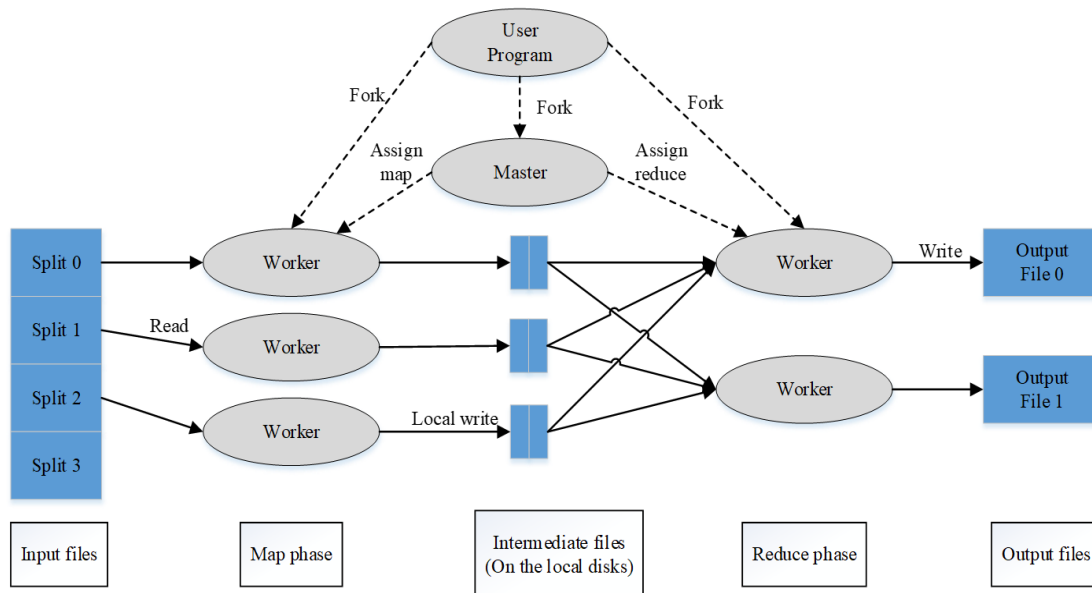


Figure 2.2 Architecture of MapReduce framework (Dean and Ghemawat, 2008)

There are some existing open source implementations of MapReduce such as Hadoop (Hadoop, 2016), which has been widely used by many organisations such as Facebook, Yahoo!, LinkedIn. However, despite the popularity of MapReduce and being extensively used by both academia and industry, the MapReduce has also been the object of severe criticism (Doulkeridis and Nørnvåg, 2014; Fernández et al, 2014; Mohebi et al, 2016), mainly due to its performance limitations, which arise in various complex processing tasks such as lack of loop-aware task scheduling. MapReduce does not support multi-staging of tasks in a single run. Whenever new MapReduce jobs are executed, the input data has to be reloaded from the disk every time during iterations and regardless whether or not the input has changed from the previous iterations.

Recently, some researchers proposed several frameworks that support asynchronous execution, which is not allowed in MapReduce. For example, some approaches provide support for

iterative algorithms that use MapReduce execution models such as: Twister (Ekanayake et al, 2010), HaLoop (Bu et al, 2010) and iMapReduce (Zhang et al, 2012).

2.4 Summary

Since the terminologies networks and graphs share the same definition, the first part of this chapter introduces the basic concepts of graph theory that are used in further chapters. This includes the definitions of adjacency matrix, degree of a node, completely connected graph, triangle, triple, reachability, homophily and hierarchical structure.

This is followed by the literature review of state-of-the-art community detection algorithms and the discussion of different categories of clustering algorithms. The field of community detection is very rich and several algorithms to detect communities in networks are proposed. As an overview, the community detection algorithms could be classified based on methodological principles into five categories: link-centrality-based algorithms, modularity optimisation algorithms, spectral algorithms, random-walk-based algorithms and information-based algorithms. For a more thorough discussion of community detection methods and algorithms and their principles, please refer to the work done by Fortunato who is one of the major authorities in the field of community detection (Fortunato, 2010) and Schaeffer (Schaeffer, 2007).

Most of the community detection algorithms in the literature are classified as global algorithms and are designed to work on a single machine. However, in large-scale network scenarios which will not fit within a single machine, it is impossible for such community detection algorithms to find communities. Parallelizing the algorithms is one way to improve the scalability of community detection. However, it is worth noting that community detection algorithms, which use global information, are not suitable for parallelization. Hence, a Decentralized Iterative Community Clustering approach (DICCA) is proposed in this research.

The last part of this chapter addresses the parallelisation techniques that have been used to parallelise the community detection algorithms. Though there are several techniques available for implementing parallelisation, most of the algorithms used for big data scenario employ MapReduce scheme. This is due to its salient features that include scalability, flexibility, fault-tolerance and simplicity. So, I have incorporated MapReduce scheme in parallelising the Decentralized Iterative Community Clustering approach (PDICCA).

CHAPTER 3

NETWORK MODELS AND STATISTICAL METHODS FOR COMPARISON OF NETWORKS

In the previous chapter, the basic concepts of community detection methods were introduced. In this chapter, the empirical properties of real-world networks are discussed. Following this, general metrics to evaluate the performance of community clustering algorithms and cluster quality on the networks are presented. . Then a comprehensive study to benchmark approaches for community detection in the networks is conducted. Finally, research methodology used in this work is discussed.

3.1 Topology of Real Networks

As it has been noted in the first chapter of this thesis, many real-world systems can be represented as complex networks. However, the real-world networks are non-random and they usually present interesting patterns and properties conveying that their inherent structure is not governed by randomness. Researchers have concentrated particularly on a few properties that seem to be common to many networks (the small-world effect, degree distribution and community effects), which will be discussed in the following subsections.

3.1.1 The Small-World effect

The small-world concept in simple terms describes the fact that even if the network has many nodes, there exists a relatively small number of intermediate steps (short path) connecting any pair of nodes within the network (Newman, 2003). It was first introduced in the 1960s by Stanley Milgram through a series of experiments (Travers and Milgram, 1967; Travers and Milgram, 1969).

The network is said to show a small-world effect if the value of the mean geodesic distance, scales logarithmically or slower with network size for fixed mean degree (Newman, 2003). However, nowadays, the small-world effect has been studied and verified directly in a large number of different networks such as, the well-known “six-degrees of separation” in social networks (Newman, 2003).

3.1.2 Degree Distribution

In real-world networks, not all the nodes in the network have the same number of edges. The spread in the node degrees is characterised by a distribution function P_k . The degree distribution P_k is defined as the fraction of nodes in the network with a degree k (Newman, 2003). Degree distribution of the network gives important information about topological characterisation of the network. For example, many networks, such as the internet (Faloutsos, Faloutsos and Faloutsos, 1999), citation networks (Redner, 1998), telephone call networks (Aiello, Chung and Lu, 2000) have all been shown to display power-law degree distribution $P_k \sim k^{-\alpha}$ where the constant α is known as the exponent of the power-law with a scaling between $2 \leq \alpha \leq 3$ (Newman, 2010).

3.1.3 Community Effects.

A number of measures have been developed for testing this tendency in the network. One of them is the clustering coefficient which measures the degree to which nodes in a network tend to cluster together. However, there are two well-known definitions of the clustering coefficient of an unweighted network: the local clustering coefficient and the global clustering coefficient (also referred as transitivity) (Newman, 2001; Costa et al, 2007).

The local clustering coefficient is a local property, introduced by Watts and Strogatz (1998a) and used to describe the network structure of nodes that are close to each other.

Consider a node i in a network G , the clustering coefficient of a node i , C_i , is defined as the ratio of the number of edges connecting the neighbours of i to the total possible number of such edges of i .

$$C_i = \frac{2L_i}{K_i[K_i-1]} \quad (3.1)$$

Where, L_i is the number of edges between neighbours of node i , K_i is the degree of node i (Costa et al, 2007).

The clustering coefficient for the whole network is the average of the local values C_i .

$$C = \frac{1}{n} \sum_{i=1}^n C_i \quad (3.2)$$

Where n is the number of nodes in the network (Costa et al, 2007).

An alternative definition of the clustering coefficient of a given node i is:

$$C_i = \frac{N_{\Delta}(i)}{N_3(i)} \quad (3.3)$$

where $N_{\Delta}(i)$ is the number of triangles involving node i and $N_3(i)$ is the number of connected triples having i as the central node (Costa et al, 2007).

The global clustering coefficient is defined as the tendency among two nodes to be connected if they share a mutual neighbour (if $a \leftrightarrow b$ and $b \leftrightarrow c$, then heightened probability that $a \leftrightarrow c$ and forming a triangle). The global clustering coefficient is based on the relative number of triangles in the network, compared to total number of connected triples of nodes and can be written as (Newman, 2001):

$$T = \frac{3*N_{\Delta}}{N_3} \quad (3.4)$$

Where: N_{Δ} is the number of triangles in the network and N_3 is the number of connected triples.

In real networks, it is shown that the small-world property is often associated with the presence of clustering, denoted by high values of the clustering coefficient (Watts and Strogatz, 1998a).

3.2 Overview of Validity Evaluation

Since there is no universally accepted definition of what a community is, assessing the validity of community detection algorithms is a hard task and several validity approaches have been developed in literature to evaluate the performance of the community clustering algorithms. However, until this day, there is no formalisation of the problem of comparing and validation of community structure. In this section, the most commonly used cluster validity metrics are discussed. The cluster validity metrics could be classified into two types, cluster quality metrics and external evaluation metrics.

3.2.1 Cluster Quality Metrics

3.2.1.1 Coverage

Coverage (Emmons et al, 2016) is one of the simplest quality functions, which compares the fraction of intra-cluster edges in the graph to the total number of edges in the graph. Coverage is given by:

$$Coverage = \frac{\sum_{i,j} A_{ji} \delta(S_i S_j)}{\sum_{i,j} A_{ji}} \quad (3.5)$$

Where S_i is the cluster to which node i is assigned and $\delta(a; b)$ is 1 if $a = b$ and 0 otherwise.

Coverage values usually range between 0 and 1. Higher values of coverage mean that there are more edges inside the clusters than edges linking different clusters. However, coverage metric does not take into account the internal cluster density and causes a strong bias toward partitions with a smaller number of clusters. Thus, it leads to a trivial clustering in which all nodes are assigned to the same cluster.

3.2.1.2 Conductance

In contrast to coverage, which measures only the accumulated edge weight within clusters, the conductance, which is also known as Cheeger constant (Arias-Castro, Pelletier and Pudlo, 2012) is based on the idea that two clusters should have a small degree of connectivity between each other and in the ideal case they are disconnected. More formally, it computes the ratio of the number of inter-cluster edges for the cluster and either, the number of edges with an endpoint in the cluster or the number of edges that do not have an endpoint in the cluster, whichever is smaller (Kannan, Vempala and Vetta, 2004).

Consider a cut that divides G into C non-overlapping clusters C_1, C_2, \dots, C_k . The conductance of any given cluster $\Phi(C_k)$ is denoted by (Kannan, Vempala and Vetta, 2004):

$$\Phi(C_k) = \frac{\sum_{i \in C_k, j \notin C_k} A_{ij}}{\min\{A(C_k), A(\overline{C_k})\}} \quad (3.6)$$

Where: $A(C_k) = \sum_{i \in C_k, j \in V} A_{ij}$ which determine the total degrees of C_k , $\overline{C_k}$ denotes the complement of C_k in graph G and A is the adjacency matrix of the graph G .

The conductance of the graph G is (Kannan, Vempala and Vetta, 2004):

$$\Phi(G) = \min(\Phi(C_k)) \quad (3.7)$$

Conductance is widely used to capture quantitatively the notion of a good network community as a set of nodes that has better internal- than external-connectivity. The lower the conductance the better is the clustering (Leskovec, Lang and Mahoney, 2010). However, as more clusters in the network will probably lead to more cut-edges, it is pointed out that the conductance has a tendency of giving better scores to partitioning with fewer clusters (Almeida et al, 2011).

3.2.1.3 Modularity

As presented in chapter 2, modularity is one of the most popular validation metrics for topological clustering and it is used as an optimisation method for detecting community structure in networks. Modularity states that a good cluster should have a bigger than expected number of connections between the nodes within modules and a smaller than expected number of connections between nodes in different modules. The higher the value of modularity the better its community strength.

3.2.2 External Evaluation Metrics

When working with a network that has well-defined clusters of “ground truth”, it is possible to evaluate a specific clustering algorithm by comparing the computed solution provided by the algorithm with this “ground truth” solution as shown in Figure 3.1. In the following subsection, the common indices that are used for measuring “goodness” of a clustering result comparing to ground truth” solution are discussed.

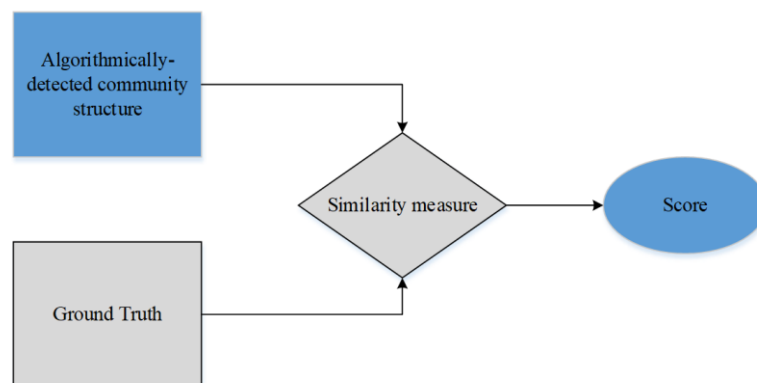


Figure 3.1 The way of benchmarking the algorithm using a network with ground-truth communities

3.2.2.1 Rand Index

The Rand Index (RI) is a statistical measure developed by Rand to measure the similarity between two clustering solutions (Rand, 1971). It is based on the relationship between pairs of nodes and requires two labels for each node. One label is corresponding to its true community

and the other one is corresponding to the predicted community. If X and Y are community clustering assignments for each node in the network, Rand Index is defined as the fraction of pairs of nodes that are correct to all possible pairs of nodes. A pair of nodes is considered correct either if the nodes share the same cluster in both clustering processes X and Y or if they are in different clusters in both solutions. The Rand Index is then given by the equation:

$$RI(X, Y) = \frac{a_{00} + a_{11}}{a_{00} + a_{11} + a_{01} + a_{10}} = \frac{a_{00} + a_{11}}{\binom{n}{2}} \quad (3.8)$$

Where:

a_{11} : i and j are assigned to the same cluster in both X and Y.

a_{00} : i and j are assigned to different clusters in both X and Y.

a_{10} : i and j are assigned to the same cluster in X but to different clusters in Y.

a_{01} : i and j are assigned to different clusters in X but to the same cluster in Y.

n: number of nodes in the network.

RI gives a measure of similarity with a value ranging from 0, when there is no pair classified in the same way under both data clusters, to 1 when data clusters are exactly the same. In practice, the RI often lies within the narrow range of [0.5, 1]. However, RI is highly sensitive to the number of clusters considered in each clustering solution and has a tendency to give higher values as the number of clusters increases (Wagner and Wagner, 2007).

3.2.2.2 Adjusted Rand Index

The Adjusted Rand Index (ARI) is the chance-corrected version of the RI proposed by Hubert and Arabie and it is known to be less sensitive to the number of clusters (Hubert and Arabie, 1985). ARI is equal to the normalised difference of the Rand Index and its expected value under the null hypothesis. The expression for ARI takes the general form (index - expected index)/

(maximum index - expected index). More formally the Hubert-Arabie's formulation of the adjusted Rand index is (Amodio et al, 2015):

$$ARI(X, Y) = \frac{2(a_{00} a_{11} - a_{01} a_{10})}{(a_{00} + a_{01})(a_{01} + a_{11}) + (a_{00} + a_{10})(a_{10} + a_{11})} \quad (3.9)$$

Like the RI, the adjusted Rand Index equals to 1 when both partitions are exactly similar. Because it is chance-corrected, a value equal to 0 represents the fact that the similarity between X and Y is equal to expected value under the generalised hypergeometric distribution assumption for randomness. However, negative values are possible and they indicate less agreement than expected value. For further detailed description of ARI, the reader is referred to Hubert and Arabie (1985).

3.2.2.3 Normalized Mutual Information (NMI)

Normalized Mutual Information (NMI) is a similarity measure for comparing two partitions based on the information theory concept. It is introduced in the community detection domain by Danon et al. and since then it has been widely used to evaluate the accuracy of community detection algorithms (Danon et al, 2005).

For an n-node network with two partitions $X = \{X_1, X_2, X_3, \dots, X_k\}$ and $Y = \{Y_1, Y_2, Y_3, \dots, Y_{\bar{k}}\}$ where X and Y represent the real communities and found communities respectively, the normalized mutual information $NMI(X, Y)$ of two divisions X and Y of a network is defined as follows (Labatut, 2015):

$$NMI(X, Y) = \frac{-2 \sum_{K=1}^k \sum_{\bar{K}=1}^{\bar{k}} P(K, \bar{K}) \text{Log} \left[\frac{P(K, \bar{K})}{P(K)P(\bar{K})} \right]}{\sum_{K=1}^k P(K) \text{Log}[P(K)] + \sum_{\bar{K}=1}^{\bar{k}} P(\bar{K}) \text{Log}[P(\bar{K})]} \quad (3.10)$$

Where: $(K, \bar{K}) = \frac{X_K \cap Y_{\bar{K}}}{n}$, $P(K) = \frac{X_K}{n}$ and $P(\bar{K}) = \frac{Y_{\bar{K}}}{n}$

If the found partition by the algorithm is identical to the real community, then NMI takes its maximum value of 1. If the partition found is totally independent of the real partition then $NMI=0$ (Labatut, 2015).

3.2.3 Computational complexity

Computational complexity theory is the study of the scalability of algorithms. The term scalability involves both the number of computation steps needed and the number of memory units that need to be allocated to run the computation. In the case of a graph, the number of nodes n and/or the number of edges m is usually used to indicate the complexity of algorithm. Big O notation is a symbolism used in complexity theory, computer science, and mathematics to describe the asymptotic behaviour of functions. It tells you how fast a function grows or decreased (Fortunato, 2010).

3.2.4 Visualization for Cluster Validation

Applying metrics is one way to evaluate the quality and correctness of the detected communities but “a picture is worth a thousand words”. Visualising networks is the most direct way of understanding them. However, large networks, particularly dense ones are very difficult to visualise due to inherent visual clutter caused by many edge crossings (Kang et al, 2014).

Different graphical representations for data associated with networks and their layout algorithms to give an impression of graph layout issues and limitations with regard to scalability have been proposed. These algorithms include Yifan Hu (Hu, 2005), ForceAtlas (Jacomy et al, 2014), Barnes-Hut Algorithm (Barnes and Hut, 1986) and OpenOrd layout algorithm (Martin et al, 2011). How to design appropriate graph visualization technique depends on many factors, including the type of graph describing the data and the analytical task at hand.

An alternative visualisation method is to use the adjacency matrix representations. In an adjacency matrix, nodes are displayed twice, on the abscissa and on the ordinate. An edge between the two corresponding nodes in the network is represented by a non-zero entry. However, since each edge in the network is defined by itself in a non-shared space, there is no edge-crossing problem. According to studies performed by Ghoniem et al. the adjacency matrix outperforms the node-link diagram when the considered graph becomes large and dense (Ghoniem, Fekete and Castagliola, 2004).

Furthermore, using adjacency matrix representations, coherent rectangular areas (blocks) appear in ordered matrix plots whenever strongly connected nodes are present in the underlying topology. In network analysis scenarios, these blocks would be referred to as clusters. Hence, with these representations, clear block patterns help counting clusters and identify larger and smaller clusters (Behrisch et al, 2016). The adjacency matrix representation has been used in many domains including: social science, artificial intelligence, biology, supply management, neurology and transportation (Behrisch et al, 2016).

In this research, I have used the matrix reordering visualisation technique for representing the community clusters.

However, the research in this work focuses on the problem of community detection in the networks and does not touch the visualization technique. For more information, interested readers may refer to (Herman, Melançon et al. 2000) and (Von Landesberger, Kuijper et al. 2011).

3.2 Artificial Networks

When evaluating the performance of community detection algorithms, there are two approaches that could be used. The first approach is to test against the real-world networks

with prior information about communities and the second approach is to test against an artificial network whose community structure is already known, which is usually termed as ground truth.

Among the former, Zachary's karate club (Zachary, 1977) and the college football network (Girvan and Newman, 2002) have been extensively used. However, due to the complexity of data collection and costs, real-world benchmarks are usually small-sized networks (Yang, Algesheimer and Tessone, 2016). Furthermore, obtaining a real network with a ground truth is not only difficult, but also costly in economic terms and time. Moreover, since it is not possible to control all the different features of a real network (e.g. average degree, degree distribution, community sizes, etc.), the algorithms could only be tested with a limited set of features. On the other hand, artificially generated networks can overcome most of these limitations. Thus, the literature has given much attention to algorithms' performance on benchmark networks and there are a number of models available to produce synthetic networks. The following subsections discuss the most well-known benchmarks that generate networks with ground truth.

3.2.1 Girvan and Newman (GN) Benchmark Networks

The Girvan and Newman benchmark (GN) is one of the first benchmarks proposed for community detection algorithms by Girvan and Newman in (Girvan and Newman, 2002). The GN benchmark network consists of 128 nodes that are divided equally into 4 communities of 32 nodes each. The strength of the community (λ) is given by the fraction of the edges placed between two communities to the total number of edges in the network. The lower value of this parameter will result in networks with clear separable communities. However, the GN benchmark has some limitations such as: all the nodes of the network have essentially the same degree, the communities are all of the same size and the network is small.

Since the real-world networks are characterised by heterogeneity in the distributions of node degrees and of community sizes, which is not the case in the GN benchmark, this benchmark is not entirely suitable for real-world network clustering (Newman, 2003).

3.2.2 LFR Benchmark Networks

The LFR benchmark model was proposed by Lancichinetti et al. to generate undirected and unweighted networks that closely resemble real-world networks with community structure (Lancichinetti, Fortunato and Radicchi, 2008). LFR model has become a popular choice for assessing the performance of community detection algorithms and the model was subsequently extended to generate weighted and/or directed networks, with the possibility of overlapping communities. However, in this work, the focus is given to the undirected unweighted networks with non-overlapping communities.

The LFR model is proposed to address most characteristics of real networks, e.g., size of the network and heterogeneous degree distribution. In the LFR benchmark, both the node degrees of a network and the size of each community are controlled by a power-law distribution with exponent γ and β respectively. However, it has been observed that real-world graphs have such a power-law degree distribution (Newman, 2003) with typical values of: $2 \leq \gamma \leq 3$, $1 \leq \beta \leq 2$ (Lancichinetti, Fortunato and Radicchi, 2008).

An important parameter of the LFR model is the mixing parameter μ , which represents the ratio between the external degree of each node with respect to its community and the total degree of the node. Each node shares a fraction $1 - \mu$ of its links with the other nodes of its community and a fraction μ with the other nodes of the network. Essentially this parameter can be viewed as the amount of noise in the graph. The larger the μ value of a network is, the harder it is to detect communities in it. If $\mu > 0.5$ then each node shares more than half of its edges with nodes in other communities, $\mu = 0$ means all edges are within community edges and $\mu = 1$ means all edges are between nodes in different communities. The model also allows controlling directly the following parameters: number of nodes and maximum degrees. The code of LFR mode is publicly made available by the authors (Fortunato).

3.3 Research Methodology

The aim of the research is to develop an accurate and effective community clustering approaches for large-scale networks. This section presents research methodology for achieving the objectives of this thesis. Figure 3.2 shows the research methodology framework used to achieve these objectives. Each stage of the methodology for this research is explained briefly in the following lines.

Studying the background information and a careful review of the relevant literature (presented in chapter 2 and 3), revealed the insufficiencies of existing community detection techniques. This provided the direction for the research and helped me to formulate the problem definition along with the research objectives that listed in section 1.4. However, to achieve these objectives three approaches are proposed and evaluated extensively.

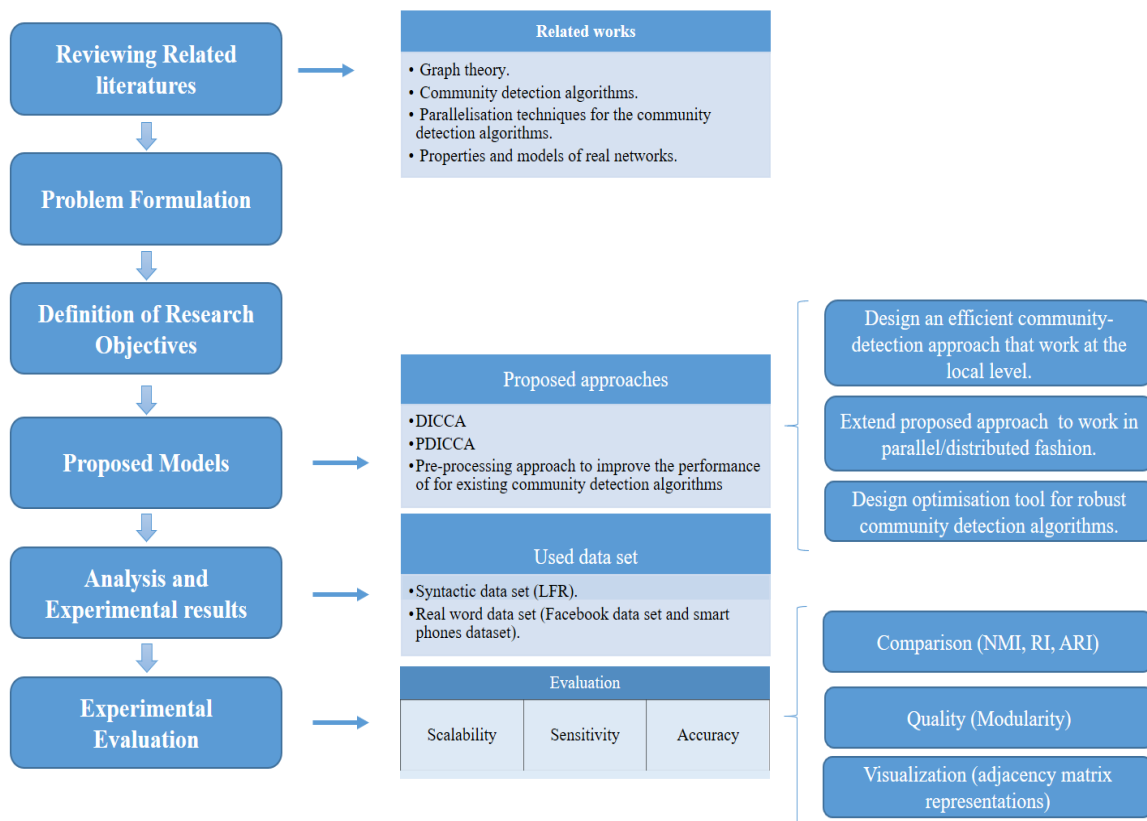


Figure 3.2 Research methodology framework

1- Decentralized Iterative Community Clustering Approach (DICCA)

A novel Decentralized Iterative Community Clustering Approach to extract an efficient community structure for large social networks are proposed. The proposed approach works at the local level and does not require any global knowledge of the network. It based on random walk and reachability, which is done by message propagation between neighbours.

2- Parallel Decentralized Iterative Community Clustering Approach (PDICCA)

PDICCA is a distributed memory parallel processing approach that transforms the serial steps of the DICCA approach into parallelised tasks.

3- An optimization approach for improving the robustness of community detection in the existing weighted community detection algorithms, especially in networks with missing information is proposed. This is done through considering attribute information, shared neighbours' information and connectivity between nodes in the network, for the detection process.

The following chapters (chapter 4, 5 and 6) explain in details about these three proposed approaches.

For implementation of the proposed approaches, list of software were used in the process:

- Matlab software
- Igraph (R) software packages

In this work, the synthetic dataset is generated by the LFR benchmark model along with their ground-truth communities in order to be able to evaluate the effectiveness of the proposed community detection approaches on a range of network-structural properties and network sizes.

In addition, anonymised Facebook datasets are used to evaluate the effectiveness of the Prepressing approach (3rd proposed approach).

Evaluating the validity of community detection algorithms based on a single measure alone can lead to misleading conclusions. Thus, in this work, a range of performance measurements, Normalized Mutual Information (NMI), modularity (Q) and Adjusted Rand Index (ARI) have been applied as evaluation criteria to evaluate the quality of community clusters. These three performance measurements are based on three different approaches. The ARI is performed on pair counting whereas, NMI is based on the information theory approach. The third approach is the modularity measure, which relies strictly on the network topology. This modularity measure allows to quantify the quality of a community structure in a blind way and without the use of a reference (ground-truth).

Going a step further, the matrix reordering visualisation is used as a visual representation for networks by encoding visually an adjacency matrix to show community clusters in the network.

3.4 Summary

Real-word networks have specific topological features, which characterize their connectivity. Measurements of the connectivity are essential to describe, analyse, model, validate the networks and exploit network structure to achieve certain aims. In this chapter, the empirical properties of real-word networks that describe the structure of the network are presented. This specifically focuses on the statistical properties of networks that have received particular attention, including the small-world effect, degree distribution and community effects.

Furthermore, in this chapter various performance measures for assessing the quality of community clustering algorithms are discussed. This includes, cluster quality metrics such as coverage, conductance and modularity, and some external evaluation metrics such as Rand

index, adjusted Rand index and Normalized mutual information. Also, adjacency matrix representation is discussed.

Finally, a comprehensive benchmarking study on the approaches for community detection in the networks is conducted. Girvan and Newman (Lancichinetti, Fortunato and Radicchi, 2008) and LFR Benchmark models (Lancichinetti, Fortunato and Radicchi, 2008) that are proposed to generate synthetic networks to mimic the real-world networks are discussed in more detail. The GN benchmark has some limitations such as, all the nodes of the network have essentially the same degree, the communities are all of the same size and the network size is small. Since the real-world networks are characterised by heterogeneity in the distributions of node degrees and of community sizes, this benchmark is not entirely suitable for real-world network clustering. So in this work, the synthetic dataset is generated by the LFR benchmark model along with their ground-truth communities is used in order to be able to evaluate the effectiveness of the proposed community detection approaches on a range of network-structural properties and network sizes.

CHAPTER 4

DECENTRALIZED ITERATIVE COMMUNITY

CLUSTERING APPROACH (DICCA)

In this chapter, a novel Decentralized Iterative Community Clustering approach (DICCA) for detecting communities in complex networks is proposed. The DICCA approach is based on the random walk procedure and reachability of nodes in the network. An important property of this approach is its ability to cluster the entire network without the global knowledge of the network topology. This ability means that this method could be easily adapted to any parallel/distributed processing to find community clusters in big networks.

Some parts of this chapter are published in the proceedings of the IEEE 28th Annual International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC, Montreal, QC, Canada (pp.1-7) in October 2017. However, in reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [Liverpool John Moores University]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

4.1 Related Literature and Previous Studies

The problem of network clustering has received considerable attention from researchers in recent years and the list of proposed algorithms is rich and diverse. Among them, those based on modularity maximization form the most prominent family of community detection algorithms closely followed by the category of algorithms based on random walks (Fortunato,

2010). However, most of the research on community detection algorithms has been designed to work on a single machine employing a form of basic random access to the entire network, so they require access to the entire network at all times (Fortunato, 2010).

In the modern era of technology, a tremendous amount of data is generated at an incredible speed from everywhere. As the data size is scaling up, the need for computing power is exponentially increasing. In many such situations, the required processing power far exceeds the processing capabilities of single machines. Furthermore, in many such cases the large-scale data set does not fit into the main memory of a single machine and needs to be distributed among several machines. These demanding requirements have led to the need for parallel and distributed algorithms for big data analysis.

In this chapter, a novel Decentralized Iterative Community Clustering Approach (DICCA) for accurately clustering networks is presented. This scheme is completely decentralized and does not require the global knowledge of the network. Apart from DICCA, there exist some other algorithms that operate based on partial information. For example, the Distributed Diffusive Clustering algorithm (DiDiC) is proposed by Joachim and Henning (Gehweiler and Meyerhenke, 2010), based on the method of disturbed diffusion, which is designed to eliminate all the global operations for assigning nodes to partitions. However, the nodes executing DiDiC algorithm need to communicate with their direct neighbours and DiDiC requires knowledge of all the neighbouring nodes.

Another algorithm somewhat similar to the proposed DICCA is Connectivity-based Decentralized Node Clustering scheme (CDC) proposed by Ramaswamy et.al (Ramaswamy, Gedik and Liu, 2005). The CDC algorithm adopts some ideas from the diffusion-based models, and is particularly designed for peer-to-peer networks. Even though the algorithm assumes that each node has a limited view of the entire network, similar to the DiDiC algorithm, CDC

algorithm requires knowledge about all the neighbouring nodes. Another distributed graph partitioning algorithm, called Ja-be-Ja, proposed in (Rahimian et al, 2013) is a decentralized local algorithm that does not require any global knowledge of the graph topology. To compute the partitioning, the node only requires some local information about its neighbouring nodes, and a small subset of random nodes in the graph. However, unlike the proposed DICCA approach, the algorithm produces partitions of equal sizes. In fact, it tends to find balanced size partitions rather than good-shaped partitions, and therefore, the number and size of yielded partitions is controlled, and does not depend on the topology of the input graph. Therefore, the outcome does not match the real-life scenario.

Table 4.1 Comparison of the algorithms

Algorithm	Short name	Concept of the algorithm	Features	Comments
Distributed Diffusive Clustering algorithm	DiDiC	Uses the concept of disturbed diffusion to identify dense graph regions	Requires knowledge of all the neighbouring nodes	DiDiC initially was implemented to balance the loads on virtual P2P supercomputers
Connectivity-based Decentralized Node Clustering scheme	CDC	The central idea in the CDC scheme is to simulate flow in the network where every edge considered as a road between two points	Requires knowledge about all the neighbouring nodes	Model is suitable for discovering connectivity-based clusters in peer to peer network and handle highly dynamic nodes
Ja-be-Ja	Ja-be-Ja	It is a distributed edge partitioner that creates balanced partitions while reducing the vertex cut	Does not require any global knowledge of the graph topology	The algorithm produces partitions of equal sizes. However, this is usually not the case for real networks.
Decentralized Iterative Community Clustering approach	DICCA	The algorithm is based on the random walk procedure and reachability of nodes in the network	Able to cluster the entire network without the global knowledge of the network topology	The algorithm adaptable to any parallel/ distributed processing to find community clusters in big networks when the size of the input network or the computation complexity is beyond the resources of a single computer.

4.2 Description of the Proposed DICCA

DICCA is an agglomerative clustering algorithm, it starts with every node belonging to a community cluster on its own and iteratively merging the clusters that have high similarity with each other. DICCA is based on random walk and reachability by broadcasting messages

through the network to compute similarity between community clusters and identify clusters in the network.

The pseudo code outlining the entire procedure is listed in Algorithm 4.1 below and it consists of two phases that run in an iterative fashion. The first phase, named local clustering, is to define originators, one for each community cluster and associate each node to the best-fit originator. The second phase, named network reduction, is used to build a new network based on the detected communities in the first phase.

In the local clustering phase of each round of the iteration, one node is selected randomly as the originator. Then this originator node sends a message (Msg) to all its neighbours. The message contains the following three fields: Originator node ID (OnID), Time to Live (TTL) and Message Weight (WMsg). OnID is used for uniquely identifying the originator node. TTL is the maximum number of hops that the Msg can be recirculated before being discarded. The message weight field (WMsg) is the weight carried by the message. The Weight represents the estimated probability of reaching any node in the network starting from the originator node. However, the WMsg is initialised to one and assigned to the originator itself, to avoid the originator being assigned to any other clusters. The function used to calculate the weight of message sent from the originator O_i to its neighbouring node V_i depends on the edges between the originator O_i and the node V_i and is defined as:

$$WMsg(O_i, V_i) = \frac{W(O_i, V_i)}{\sum_{V_j \in Nbr(O_i)} W(O_i, V_j)} \quad (4.1)$$

Each node in the network maintains a set of values, represented as Total Message Weight, originator ID. The Total Message Weight value represents the sum of the weights of all the messages that reached N_i and has the same Originator node ID. When the node V_i receives a message Msg, it updates the total weight function corresponding to the message originator node. Then, the receiving node V_i checks whether or not the TTL of the message is greater than zero.

If so, the node decrements TTL value by one, updates WMsg of the Msg and forwards the updated message to all its neighbours. The updated weight of the new message $WMsg(V_i, V_k)$ being re-sent from node V_i to its neighbouring node V_k is defined as:

$$WMsg(V_i, V_k) = WMsg \times \frac{W(V_i, V_k)}{\sum_{V_j \in Nbr(V_i)} W(V_i, V_j)} \quad (4.2)$$

However, Node V_k halts the message circulation if TTL is zero or WMsg becomes insignificantly low. When the TTL reaches zero, the message will no longer be forwarded and the nodes join the community led by the originator node O_i that has received total weight values greater than the specified threshold. However, if the total weight values received for some nodes lie below a predefined threshold, then those nodes will remain as outliers.

In the next step, the algorithm adds one more originator node, by randomly selecting one of the nodes from the outliers that do not belong to any community. Then the new originator repeats the same process that was carried out by the former originator and updates communities and their corresponding originator as well as the outlier nodes list. The algorithm keeps iteratively adding one more originator, and updating communities and outlier nodes until each node is joined to a community, and there is no outlier node remaining. However, each node in the network may receive multiple messages generated from different originator nodes. In that case, the node joins the community led by the originator node that has the highest total weight.

The second phase of the algorithm consists of building a new network from the communities discovered in the first phase where the individual nodes in the new network are the individual communities from the first step. In this new network, there will be an edge between two nodes if there were edges between the corresponding two communities in the previous step. The weights of those new edges are the sum of the weights of the edges between nodes in the corresponding two communities. The edges between nodes of the same community in the first step will lead to self-loops for this community node in the new network.

The two phases mentioned above are repeated with the rebuilt network iteratively and the process stops when there is no more change in the communities and consequently optimised community clusters are obtained.

Although the exact computational complexity of DICCA is harder to formalize, this algorithm behaves as $O(m \log((n \cdot m)^2))$, in which n is the total number of nodes in the network and m the number of edges. However, the most effort is in the first phase of the algorithm.

The proposed concept is shown in Figure 4.1. The figure illustrates how the proposed algorithm works at different stages of execution of the algorithm with 11 nodes labelled from 1 to 11 and 17 unweighted edges. The algorithm process is initiated by choosing node 4 as originator in the first iteration and threshold value is set to 0.25. Messages in the figure are defined by three fields that provide information about the messages representing the originator, TTL and current weight of the message respectively. For example, if the field value of the message received by node 5 is {4:2: 0.25}, it means that the message data was originated by node 4 and the weight of current message is 0.25 with TTL=2.

By compiling the notions above, a community cluster in the proposed algorithm can be described as:

1. The nodes and only these nodes which are mutually densely-connected, belong to the same cluster.
2. If node V does not have many neighbours and it is reachable from one or several nodes, then V belongs to the cluster that is more densely connected.
3. If V does not have any neighbours, then V does not belong to any cluster.
4. The obtained communities are not overlapping and consequently, they define a partition C of n such that $V = \cup_{i=1}^k C_i$ and $C_i \cap C_j = \emptyset$ for any $i \neq j$.

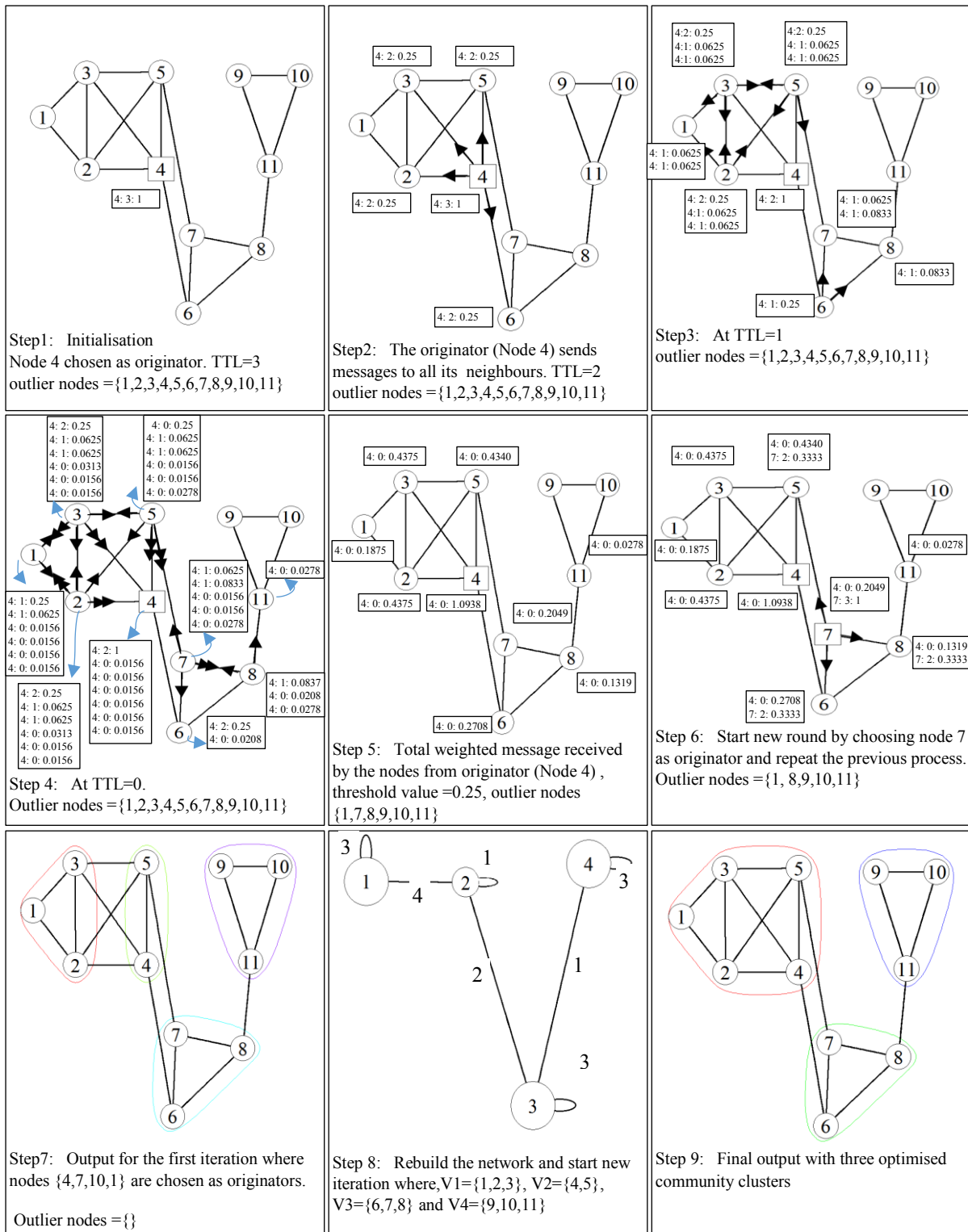


Figure 4.1 Illustrates the concept of the algorithm

Algorithm 4.1. The proposed method

Input: underlying network graph G , time_to_live and threshold value

Output: C communities as a final division of G .

Repeat

Outlier list \leftarrow **all nodes** // local clustering phase

While outlier list $\neq \{\}$

$O_i \leftarrow$ Rand select (outlier list) // choose a node randomly to be an originator.

 //creat new message (Msg)

 OnId \leftarrow O_i // originator ID

 TTL \leftarrow time_to_live

 WMsg \leftarrow 1

 Msg \leftarrow { OnId , TTL, WMsg }

While TTL \geq 0

 Total_weight (O_i, V_i) = **sendmessages**($G, O_i, \text{OnId}, \text{TTL}, \text{Msg}$) // Total

 //weight between O_i and its neightbout nodes (V_i)

 TTL \leftarrow TTL-1

$O_i \leftarrow V_i$

 Msg \leftarrow { OnId , TTL, Total_weight (O_i, V_i) }

end while

for each Node $V_i \in G$

if Total_weight(V_i, onID) \geq threshold **then**

$C(V_i) \leftarrow$ Join the cluster lead by max onID

else

 Remain outlier

end if

end

end while

$\hat{G} = \text{Aggregate}(G, C)$ // Network reduction phase “Compact each community to one
 // new node and build new network”

if ($C_{\text{current}} = C_{\text{previous}}$) // no membership change

break;

return C // return the final division of G

end Algorithm

Function sendmessages ($G, O_i, \text{OnId}, \text{TTL}, \text{Msg}$)

for each Node $V_i \in \text{Nbr}(O_i)$ **do**

 Send WMsg to $V_i \leftarrow$ $W_{\text{Msg}}(O_i, V_i) = W_{\text{Msg}}(O_i, V_i) * W(O_i, V_i) / \sum_{V_j \in \text{Nbr}(V_i)} W(V_i, V_j)$

If N_i have seen message from onID before **then**

 Total_weight(V_i, O_i) \leftarrow Total_weight (V_i, O_i) + WMsg

else

 Total_weight(V_i, O_i) \leftarrow WMsg

end if

end

Return Total_weight(V_i, O_i)

end function

4.3 Experimentation and Results

4.3.1 LFR Synthetic Dataset (network)

Many real-world complex networks such as the Internet, social networks, biological networks, infrastructure networks etc. are heterogeneous and show a power-law degree distribution (Newman, 2003). In such networks not all their components such as nodes, links and subgraphs carry the same role or importance in the network, which has crucial effects on the resulting performance of the algorithms deployed. Consequently, the performance of any community detection algorithm varies depending on the network's characteristics. Furthermore, to analyse the efficiency of the community detection algorithm, one needs to apply it to networks which have ground truth communities (the actual partitions), and then the performance of the algorithm needs to be measured as the accuracy in recognising the ground truth communities.

Due to the scarce availability of real networks that have ground truth communities, and in order to measure the performance of the proposed community detection algorithm on both network-structural properties and network size, the synthetic dataset is generated by the LFR benchmark model along with their ground-truth communities and used to test the proposed algorithm in this work.

4.3.2 Evaluation Metric

Since the true community structure is known for the benchmark network, the proposed algorithm is evaluated by comparing the obtained partition in the experiments with the ground truth provided by the LFR benchmark. Normalized mutual information (NMI) metric is used to quantify the accuracy of community detection methods by evaluating the level of correspondence between detected and ground-truth communities. In addition, modularity measurement is used to evaluate how effective the algorithm is in terms of modularity optimisation.

4.3.3 Parameter Selection Strategy

The proposed algorithm uses two parameters, which are ‘time to live’ and ‘threshold value’; if these two parameters are optimally set, then, it will highly improve the performance of the algorithm. So some strategies about the choice of these two initial parameters are discussed in this section.

4.3.3.1 Time to Live

TTL is a parameter used by the algorithm to control the number of nodes visited in the network. TTL value must be a positive integer greater than zero. In reality, choosing an appropriate TTL value is not an obvious task. On one hand, small time-to-live may expire before reaching many relevant nodes which are further away. On the other hand, high time to live means more nodes than needed are visited, thus increasing both the message load on the network and the running time of the algorithm. Therefore, in the proposed algorithm, rebuilding the network before starting a new iteration is considered as a solution for this issue. For example, with a small value of TTL, some nodes (V_f) that are densely connected with the neighbours of the originator (intermediate nodes between them and the originator node) cannot receive messages from the originator O_i as the TTL value might have expired in the current iteration. Then in the following iteration, the intermediate nodes will be merged with the originator node making them as one node. Then in the next iteration these V_f nodes will be reached by the originator O_i with a small value of TTL.

In order to determine the effect of TTL value on the community clustering accuracy, the TTL value ranging from 1 to 4 has been used in this evaluation. Figure 4.2 indicates the accuracy values of synthetic networks with 500 and 1000 nodes. In this work, modularity and NMI have been used to evaluate the quality of community detection. In order to give a condensed picture

of the results, the computing time in seconds and the message complexity results as a function of the TTL are presented in Figure 4.3.

From the figure, it is clear that there is a correlation between TTL and both computing time and message complexity. The smaller the TTL, the faster the algorithm. This can be qualified

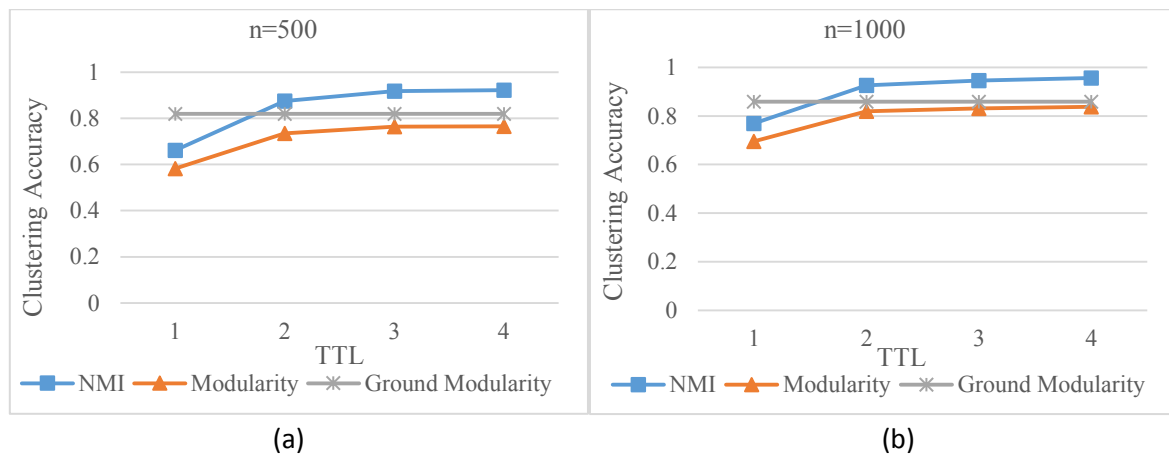


Figure 4.2 Performance of the DICCA algorithm using different TTL values

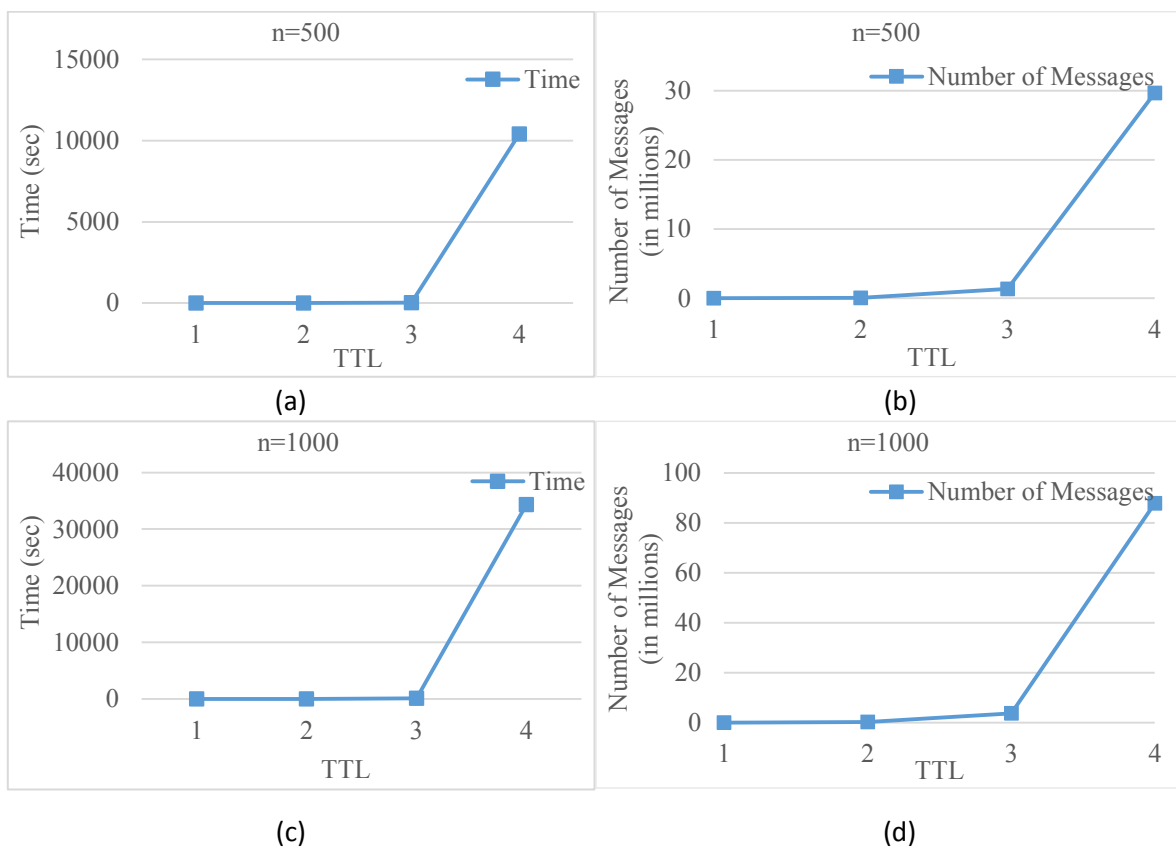


Figure 4.3 Comparison between computing time and the message complexities over different TTL values

by the fact that the run time of the DICCA algorithm depends on the total number of exchanged messages which in turn is affected by the total number of hops that a message is permitted to travel before being discarded (TTL).

However, the proposed algorithm in this work is implemented in Matlab from scratch, which is not optimised for speed. Therefore, the total number of exchanged messages (Message Complexity) will be computed as a score for running time in this work.

The graphs in Figure 4.2 demonstrate that the algorithm yields good community clusters when the TTL is set to be 3. Furthermore, recall from chapter 3 that big networks from real-world applications are often small-world networks (Watts and Strogatz, 1998b) (Silva and Zhao, 2016), so increasing the TTL value does not have significant impact on the quality of community detection but may result in a very high communication load. However, selecting a small TTL value can reduce the broadcast overhead but will compromise the accuracy. For example, when $TTL = 1$ is used, the WMsg message is only being propagated once from originator to its neighbour, which means only the direct originator's neighbour nodes could be merged in that iteration. For this scenario, the NMI and total number of messages generated by the algorithm for $N \in \{500; 1000\}$ were $\{0.661; 0.769\}$ and $\{4832; 9019\}$ and respectively. On the other hand when a value of $TTL=3$ was used for $n \in \{500; 1000\}$, the NMI results were $\{0.918; 0.946\}$ and the total number of messages were $\{1,347,024; 3,735,475\}$. Furthermore, when $TTL = 4$, the NMI scores were $\{0.922; 0.956\}$ which are almost same as the NMI yielded by the algorithm when TTL is 3. On the contrary, the total number of messages generated were $\{29,680,547; 87,794,210\}$ which are significantly higher than that generated when TTL was 3. Based on the above discussion, it is clear that the algorithm will stabilize very fast on the networks with small value of TTL, but quality is worse in most cases. On the contrary, using a

large value of TTL can ensure that all nodes will receive the message, but introduces unnecessary broadcast messages for nodes beyond the target-clustering region.

The number of messages sent during an iteration clearly depends on the number of nodes in the network and on the size of the n-neighbourhoods of the nodes (network structure). This means high communication load is required for extracting clusters and may result in a scalability problem in large and dense network environments. This scalability issue greatly hinders the application of module extraction to network analysis where most of the networks consist of high number of nodes. However, in big networks, the message weight becomes extremely low compared to a threshold value. A node's decision to join a cluster is based on the total weight of the messages from the originator to the node exceeding the threshold value. Consequently, extremely low message weight does not affect the accuracy of clusters and the process could be halted.

To avoid an excessive number of messages being forwarded, adaptive termination technique has been implemented in the DICCA approach. When the message weight becomes insignificantly low, the message is discarded by the received node even though the TTL may still be greater than zero. In this work the minimum value of message weight (Min_VALUE) is specified to be three hundred less than threshold value.

By comparing Figures 4.2-4.3 with Figure 4.4, it can be observed that there are negligible differences between the performance of the algorithm in terms of NMI and Modularity scores.

Considering message complexity and running time, the performance of the algorithm when the Min_VALUE is applied is by far better than its performance when Min_VALUE is not applied.

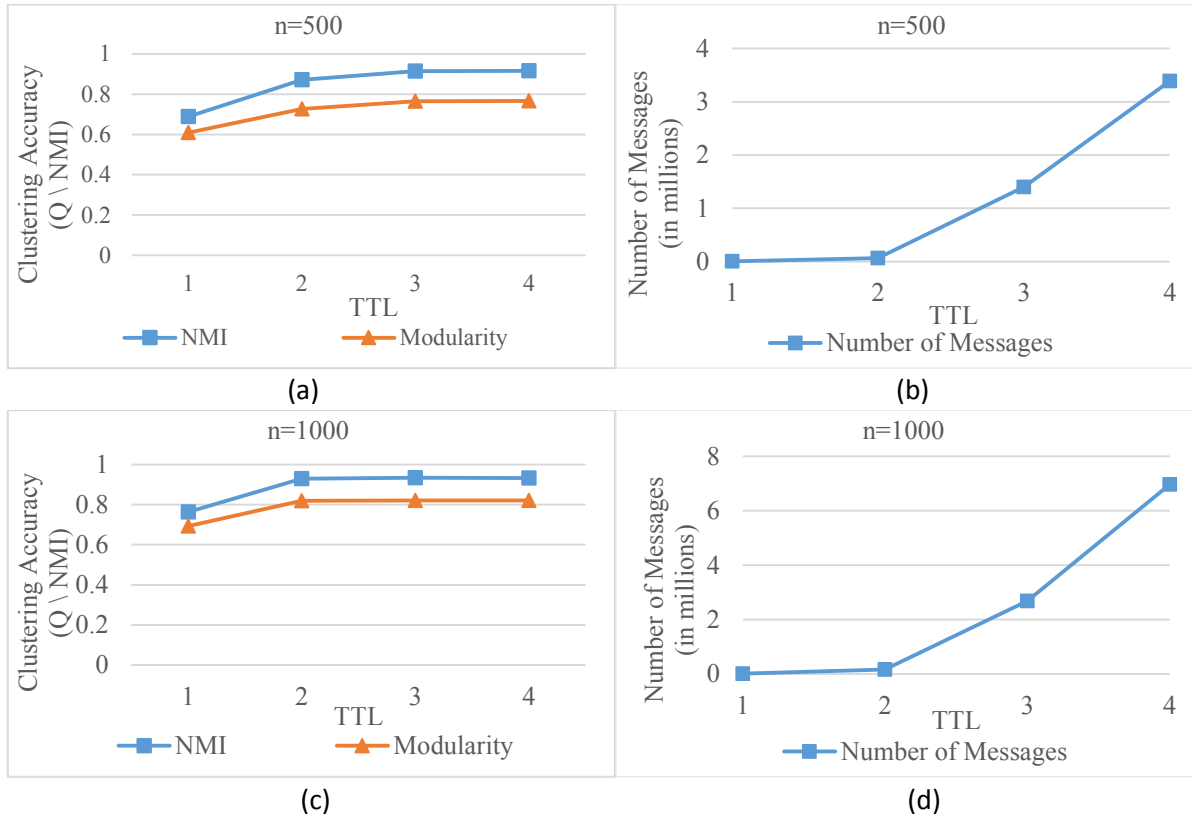


Figure 4.4 Performance of DICCA algorithm using adaptive termination via different TTL values

4.3.3.2 Threshold Value

The threshold is a numerical value ranging between 0 and 1, which defines the minimum weight of the message required to join a cluster. It is defined by the user at the beginning of the process. The node is allowed to join the community cluster led by originator O_i , if the total weight of the message received by the node from O_i is equal to or greater than the threshold value. As the threshold value increases, the difficulty of merging communities also increases. Thus, the size of the community clusters depends on the threshold value. If a high threshold is set, more small-size communities are detected. On the contrary, setting a lower threshold leads to fewer but large size detected clusters. Therefore, the size of the community clusters produced by the proposed algorithm could be controlled using the threshold parameter. The threshold value is

in the range of $\{0; 1\}$, 0 yielding a single community and 1 producing clusters of singleton nodes. Tuning this parameter could be seen as a possible practical remedy to control the desired size and the number of communities.

In order to understand how the threshold value affects the accuracy, size and the number of community clusters, the effect of different threshold values has been studied on a small network with 50 nodes and 83 edges. The results presented in Table 4.2 show that when the threshold value increases, more small-sized communities are detected. In contrast, lower threshold value leads to larger detected clusters. For example, when the threshold value is 0.1, three clusters have been detected and the biggest detected cluster has 21 members. That number of clusters becomes 5 when the threshold parameter is changed to 0.7. That is because larger threshold value means more strict requirements in community intra-connectivity and only strongly connected nodes can belong to the same cluster.

Table 4.2 The experimental results obtained by the DICCA algorithm on a small network of 50 nodes

Threshold value	NMI	Number of clusters	Modularity (Q)	Min N.of members	Max N.of members	Avg N.of members
0	0	1	0	50	50	50
0.1	0.664672	3	0.623675	14	21	16.66667
0.2	0.810166	5	0.674046	5	21	10
0.3	0.88515	6	0.717521	5	16	8.333333
0.4	0.85165	9	0.658151	1	10	5.555556
0.5	0.900606	12	0.622587	1	9	4.166667
0.6	0.900606	16	0.622587	1	9	3.125
0.7	0.723512	39	0.18682	1	5	1.282051
0.8	0.670295	50	-0.02584	1	1	1
0.9	0.670295	50	-0.02584	1	1	1
1	0.670295	50	-0.02584	1	1	1
0.223xt¹	0.950701	9	0.68907	2	10	5.555556

Figure 4.5 shows the visualization of synthetic network with 50 nodes and the detected clusters when the threshold parameter is varied from 0 to 1 in steps of 0.1. The layout for all the different

visualizations of the network is kept constant to be able to draw conclusions easily by looking at the figures. Members in the same community are represented with the same colour.

Using the proposed DICCA algorithm the maximum modularity is obtained when the threshold value is 0.3 by the partition in 6 communities achieving $Q=0.71$ (graph d). However, the ground truth partitioning is 8 communities with $Q=0.717$. DICCA merged three communities into one. Beside this, there are 5 communities classified correctly with the exception of one node (node 23) which is misclassified.

Clearly, the success of the algorithm is heavily dependent on the proper tuning of the threshold value. However, there is no standard prescription for threshold value for all type of data sets and applications. The most appropriate threshold value for a given data set is usually derived experimentally, defined by the user according to their knowledge or estimated on the basis of data from previously completed similar projects.

4.3.3.3 Automated Identification of Appropriate Threshold Value

Although the threshold value controls the number and the size of clusters that will be extracted, which could be considered as an advantage of the algorithm, choosing the right threshold without a priori knowledge of the network structure is a challenging task. Furthermore, generating a priori knowledge requires human expertise and is time consuming since real networks are usually big and contain huge amounts of information (De, 2016). In this work, based on the above observation, a mathematical model is proposed to automatically calculate the threshold value. The model calculates the optimal threshold value based on the size, density and layout structure of the network. Equations 4.3 to 4.5 present the threshold calculation model for undirected networks designed by the author to help calculate the threshold value

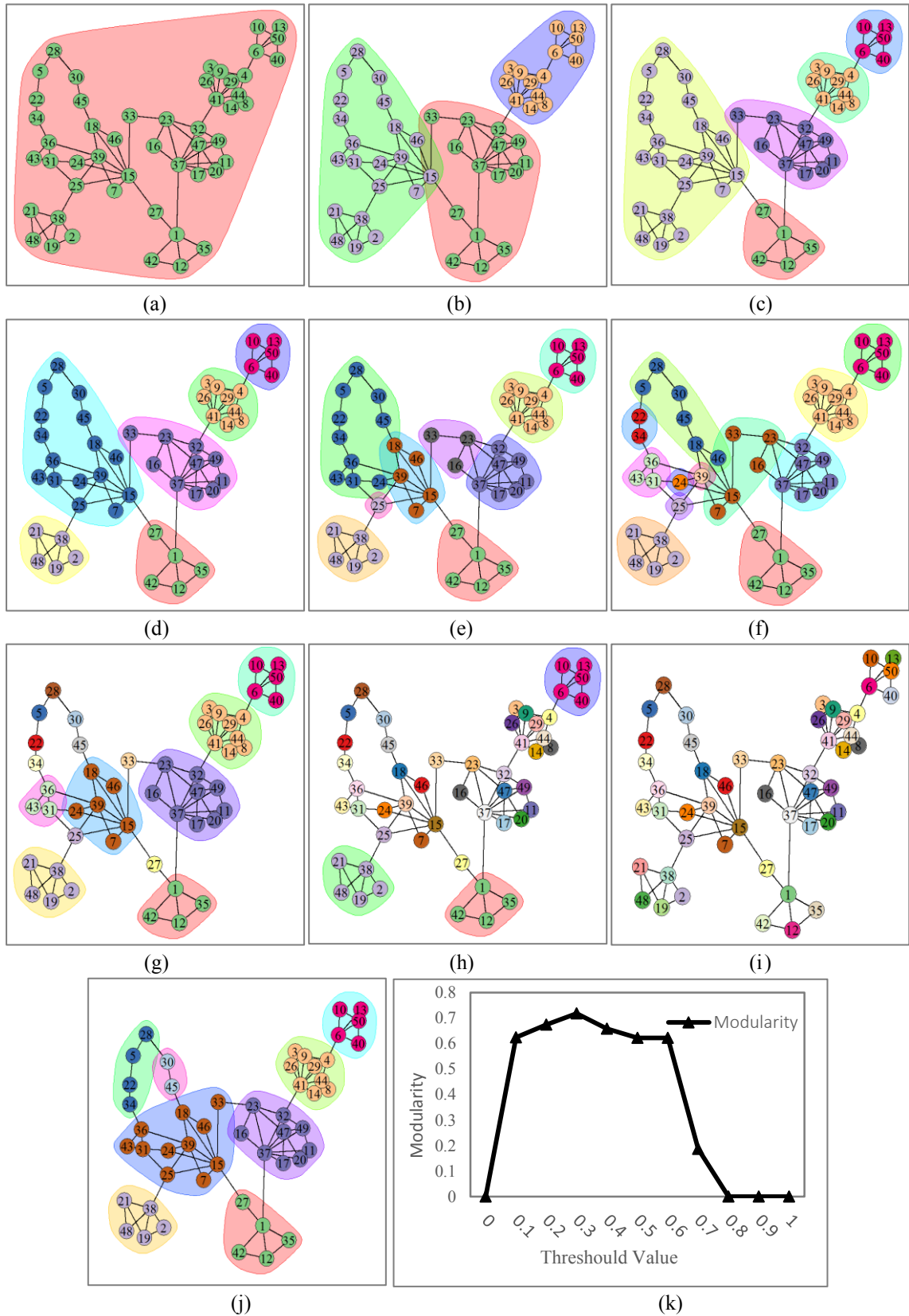


Figure 4.5 Community detection result for a small network with 50 nodes as extracted by the proposed DICCA algorithm using $TTL=3$ and with different threshold values. (a) threshold value =0, (b) threshold value =0.1, (c) threshold value =0.2, (d) threshold value =0.3, (e) threshold value =0.4, (f) threshold value =0.5, (g) threshold value =0.6, (h) threshold value =0.7, (i) threshold value ≥ 0.8 , (j) ground truth clusters, (k) Modularity via threshold value. The values of the other parameters were fixed: $\gamma=2$, $\beta=1$.

when the users have no knowledge of the community properties of the network. Threshold value calculation for specific networks and applications may require specific concepts and considerations.

In undirected network, the threshold value is defined as follows:

$$\text{Threshold value} = avg_t + (t - 1)x(1 - C) x avg_t \quad (4.3)$$

$$avg_t = \frac{\log(\log(n))}{\log(n)} \sum_{i=1}^n \left(\frac{1}{k(i)} + \frac{K_i-1}{K_i^2} + \frac{K_i-2}{K_i^3} \right) \quad (4.4)$$

$$K_i = \sum_{j=0}^n A_{ij} \quad (4.5)$$

where, t is the iteration number, K_i is the degree of node i, n is the total number of nodes in the network, A is the adjacency matrix and C is network clustering coefficient which is defined as:

$$C = \frac{1}{n} \sum_{i=1}^n \frac{2L_i}{K_i[K_i-1]} \quad (4.6)$$

where L_i is the number of edges between neighbours of node i (Costa et al, 2007).

Given a network with n nodes, a complete network (fully connected network) is a simple undirected graph in which every pair of distinct nodes is connected by a unique edge. Based on the graph theory the network clustering coefficient for a fully connected network is 1 and the degree of each node is defined as:

$$K_i = n - 1 \quad (4.7)$$

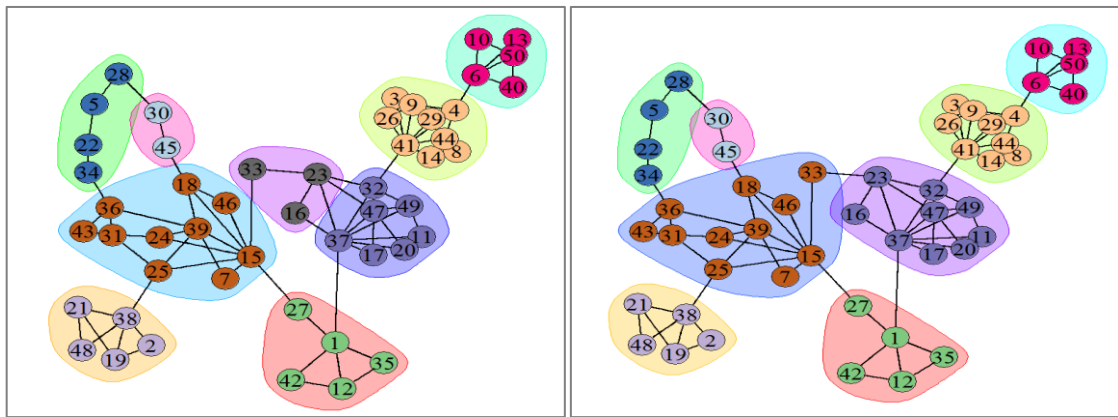
Thus, the total edges of the network having n nodes will be:

$$\sum_{i=0}^n K_i = n(n - 1) \quad (4.8)$$

Using equation (4.3) to calculate the threshold value:

$$\text{Threshold Value} = \frac{\log(\log(n))}{\log(n)} \sum_{i=1}^n \left(\frac{1}{K_i} + \frac{K_i-1}{K_i^2} + \frac{K_i-2}{K_i^3} \right) \quad (4.9)$$

Here, the value of $\sum_{i=1}^n (\frac{1}{K_i} + \frac{K_i-1}{K_i^2} + \frac{K_i-2}{K_i^3})$ represents the maximum weight of messages received by node i when the TTL=3 and since $\frac{\log(\log(n))}{\log(n)}$ is always less than 1, if the proposed algorithm with adapted equation (4.3) for threshold parameter is used to extract clusters in the complete network, the algorithm will merge all nodes in one cluster from the first iteration. This result is acceptable since there is no obvious cluster structure in a fully connected network. It is worthwhile mentioning that, in each iteration, the threshold value is stepwise increased by $(t-1) \times (1-C) \times \text{avg}_t$ as seen in equation (4.3), so that it becomes progressively difficult for clusters that are not so densely connected to join with each other. Only the strongly connected ones will be able to merge. Additionally, the maximum threshold value cannot be larger than 1. By using the proposed model, the threshold value at the first iteration for a small network of 50 nodes as considered in Table 4.2 is derived as $0.223 \times t^1$, where t^1 refers to the first iteration.



(a) Communities detected with proposed algorithm.

(b) Ground truth communities

Figure 4.6 The community structures of the ground truth communities and those extracted by the proposed DICCA algorithm on the LFR benchmark networks with 50 nodes using TTL=3 and threshold value $=0.223 \times t^1$.

Figure 4.6 shows the visualization of the ground-truth community structure of 50 nodes and the detected clusters result using the DICCA algorithm when the threshold value parameter was calculated using equation (4.3). The DICCA algorithm gives a near optimal partitioning. It identifies nine clusters, one more than the ground truth partition, which has difficulty in extracting the cluster containing nodes 33, 23 and 16.

Based on the above argument, in all the experimentations performed in this work as discussed below, threshold value is defined using equation (4.3) and to achieve good trade-off between high modularity and low message complexity (running time), TTL is set to a value of 3.

4.4 Analysis of Results and Discussion

In this section, the results from the experiments conducted using synthetic networks are presented, analysed and discussed in detail. The proposed DICCA approach was implemented using Matlab, which is not optimised for speed on the windows system with ® Core™ i7 6700K CPU 4.00GHz and 16 RAM available memory.

A set of undirected networks were generated using the LFR benchmark graph. The default benchmark parameter values are used as the benchmark parameters for the exponents of the degree distribution and community size, viz. $\gamma = 2$, $\beta = 1$. The mixing parameter is varied from 0.1 to 0.75 and the number of nodes is varied from 500 to 5000. The average degree and the maximal degree are 25 and 50, respectively. Table 4.3 outlines the parameters used to generate the LFR benchmark graph.

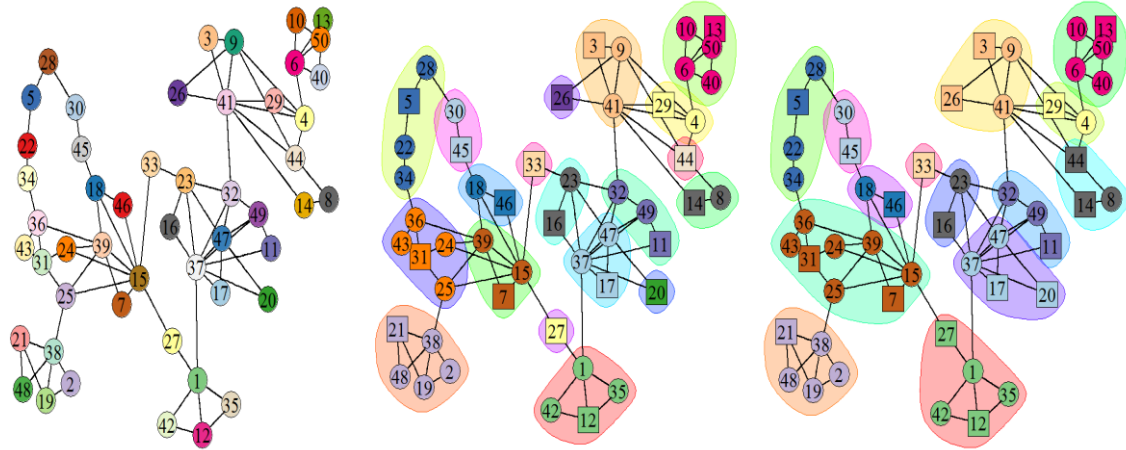
Table 4.3 The LFR benchmark graph parameters.

Variable	Value	Description
n	$n \in \{500, 1000, \dots, 5000\}$	number of nodes in the network
\bar{K}	25	mean degree of each node
k_{max}	50	maximum degree
μ	$\mu \in \{0.1, 0.15, \dots, 0.75\}$,	mixing parameter
β	1	exponent of community size distribution (typically $1 \leq \beta \leq 2$ in real-world networks)
γ	2	exponent of degree distribution (typically $2 \leq \gamma \leq 3$ in real-world networks)

For each combination of parameter values, five instances of network were generated to check for consistency. Furthermore, to eliminate the effect of randomness of choosing originators in the proposed DICCA method, the algorithm was run 20 times on the five instances of network datasets, so, the experimental results presented are the average of 100 simulation runs.

4.4.1 Results for Each Iteration of Clustering

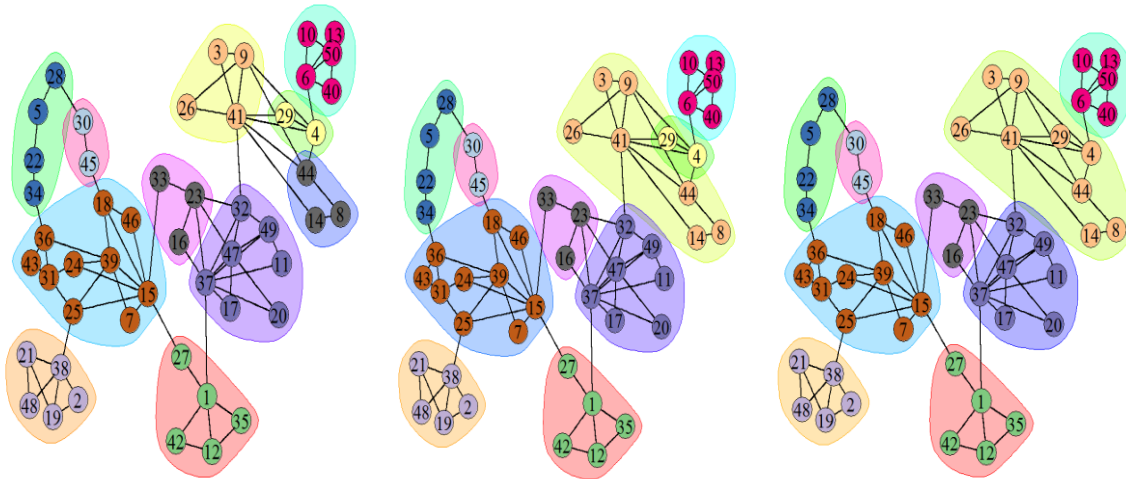
Figure 4.7 shows iteration results of the algorithm for a small network with 50 nodes. Nodes in the same community are labelled in the same colour. In the first iteration, originator nodes are represented by rectangular shape. It is worth mentioning that due to the reduction phase of DICCA, which consists of merging nodes in the same community into one node to create a new graph, nodes in the figure that are shaded together with the same colour represent one node in the following iteration process of the algorithm. Each iteration results in a network with a different number of community clusters, and the number of communities becomes smaller and smaller until the convergence of clusters is achieved. For example, in the initialisation stage, each node is a cluster on its own, therefore there are as many clusters as the number of nodes in the network. After initialisation, in the first iteration, 15 communities are identified followed by 14 and 11 communities during the second and third iterations respectively. The random initial originator nodes are transferred into meaningful clustering in iteration 5. Graph (g) in Figure 4.7, illustrates the convergence of the clusters, where there is no change in cluster membership of clusters with subsequent iterations (iteration 5). To be able to analyse the intermediate results of the algorithm the value of modularity and NMI via the iteration are calculated and shown in graph (h) in Figure 4.7, which reveals that at each iteration, the measure of both Modularity and NMI are improved progressively until the convergence is reached.



(a) Initialisation

(b) Iteration#1

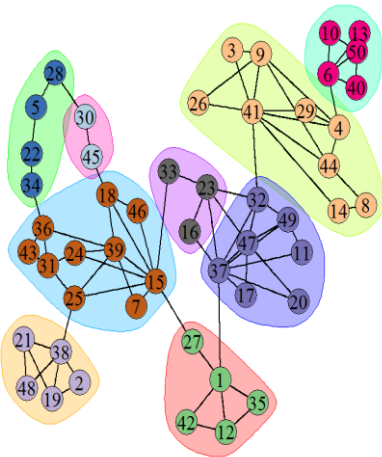
(c) Iteration#2



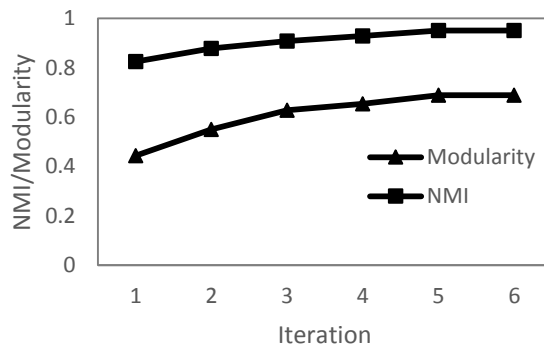
(d) Iteration#3

(e) Iteration#4

(f) Iteration#5



(g) Convergence



(h) Performance via iteration

Figure 4.7 Community detection result for each iteration on a small network of 50 nodes using the proposed DICCA algorithm with TTL=3, threshold value = $0.223 * t$, and $\gamma=1$, $\beta=2$.

4.4.2 Clustering Results for Increasing Network Size

To check how the performance of the proposed algorithm is affected by the network size, the algorithm was evaluated using the previously discussed synthetic network with varying number of nodes, viz. $n \in \{500, 1000, \dots, 5000\}$. The obtained community structure is compared with the ground truth communities using the previously discussed NMI and modularity measures.

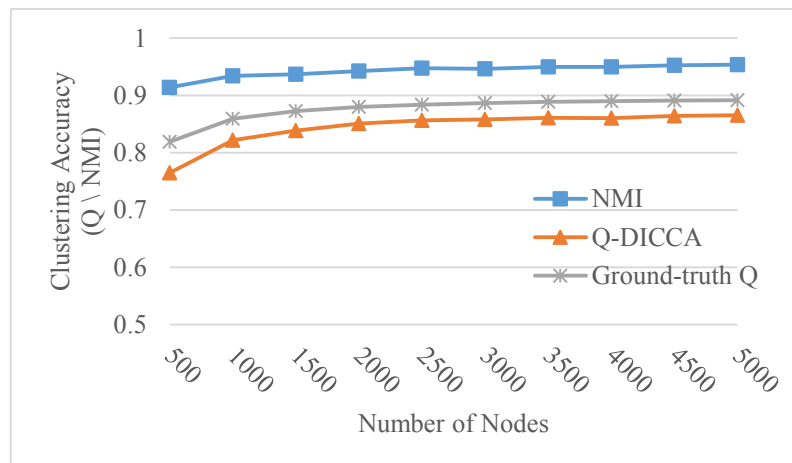


Figure 4.8 NMI, Q-DICCS and Ground truth Q scores (y-axis) as number of nodes (x-axis) changes.

Figure 4.8 shows the clustering accuracy of the proposed DICCA algorithm when the network size is varied from 500 nodes to 5,000 nodes. The algorithm performs very well and the communities detected are very close to the reference (value of 1) with an average NMI value of above 0.9. However, the modularity index (Q) of clustering results obtained by the DICCA algorithm is slightly lower compared to that of the ground truth network.

4.4.3 Evaluating Repeatability of the Algorithm's Performance

It is important to mention that several clustering methods are sensitive to random starts of algorithm (Weber and Robinson, 2016) and the resulting clusters depend on the initial random starts where the algorithm does not yield the same result with each run. However, to further investigate the ability of the DICCA clustering algorithm to produce consistent results across random starts, the standard deviation of the clustering results is measured where the algorithm is run 100 times each time with different random initialisation. The lower values of standard

deviation indicate lower output changes and are always preferable. Results of the standard deviation value of both NMI and modularity are displayed in Figure 4.9. As an overview, the most notable phenomenon that can be observed from the results is that the overall value of standard deviation is negligible, indicating that the DICCA algorithm does not have stability issues and is able to successfully reproduce stable output when the experiment is repeated.

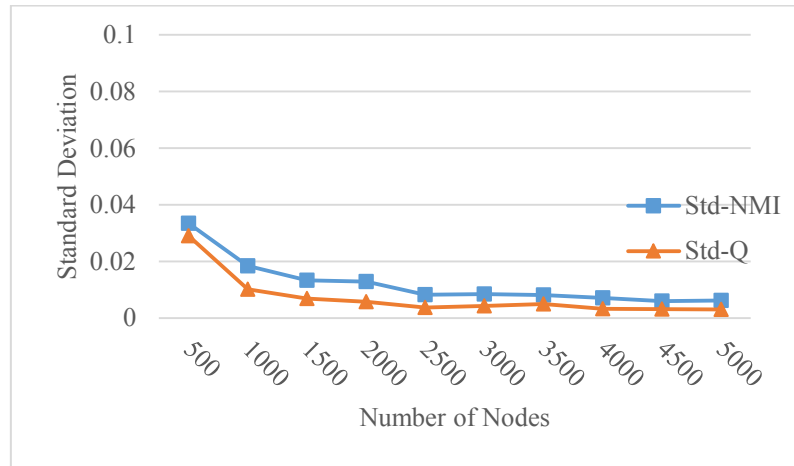


Figure 4.9 Standard deviation of final modularity/NMI with network sizes.

4.4.4 Evaluation of Message Complexity of the DICCA Algorithm

Performance of the proposed algorithm was evaluated in terms of the total number of exchanged messages for different network size, as an indirect measure of processing capability required for increasing network size. At the outset, the curve in Figure 4.10 shows a linear increase in the number of exchanged messages with increasing size of the network.

However, more in-depth analysis as shown in Figure 4.11, which shows the average percentage of exchanged messages in each iteration tells a different story. It can be observed from the figure that data exchange for the DICCA algorithm is much greater at the first stage of iteration when each node is in its own cluster. Just after 2 to 3 initial iterations, most nodes have their cluster labels and the algorithm has merged the nodes belonging to the same cluster to be one node. In fact, on average more than 90% of the data exchange happens in the first iteration for a network size of 1,000 nodes. As seen in Figure 4.11, the percentages of total exchanged

messages in the first three iterations are 99.59 % and 98.66% for network size of 500 and 1,000 nodes respectively. Hence, it can be safely concluded that though the proposed approach may tend to have an increasing number of generated messages for increasing network size, it does not require more iterations before the clusters converge. Most of the data exchange is in the first 2 or 3 iterations due to the sheer number of nodes exchanging data with each other. The average number of iterations is slightly increased from 5 to 7 as the number of nodes increased from 500 to 5,000 (See table A.1.1 in Appendix A.1).

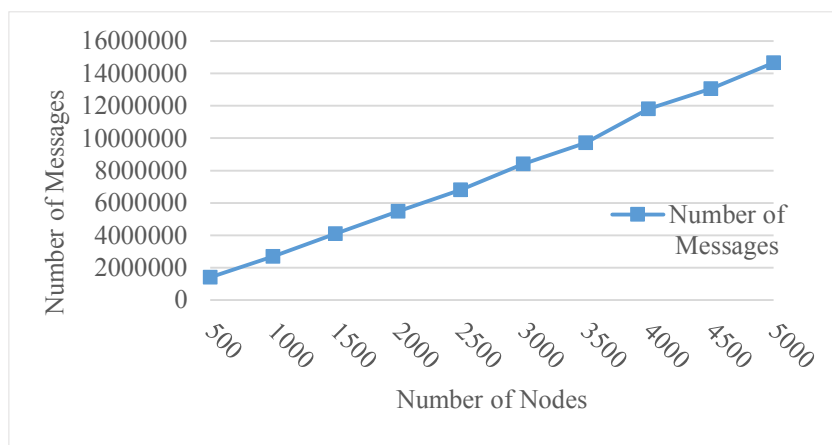


Figure 4.10 Total number of exchanged messages (y-axis) as number of nodes (x-axis) changes

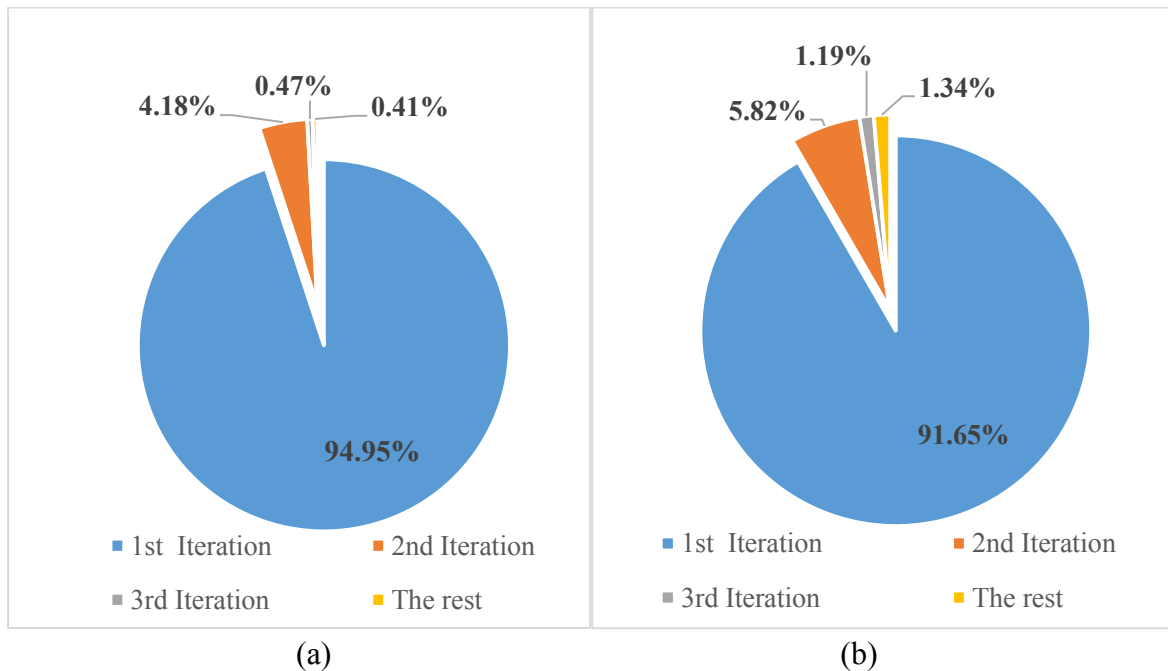


Figure 4.11 Percentage of Message exchanged per each iteration. (a) number of node in the network is 500, (b) number of node in the network is 1,000.

4.4.5 Evaluation of Clustering Performance Using Mixing Parameter

The DICCA algorithm was evaluated with varying values of mixing parameter between 0.1 and 0.75, $\mu \in \{0.1, 0.15, \dots, 0.75\}$, and keeping the number of nodes constant, $n \in \{500, 1000\}$. Figure 4.12 shows the mean values of all the obtained results for NMI and Q.

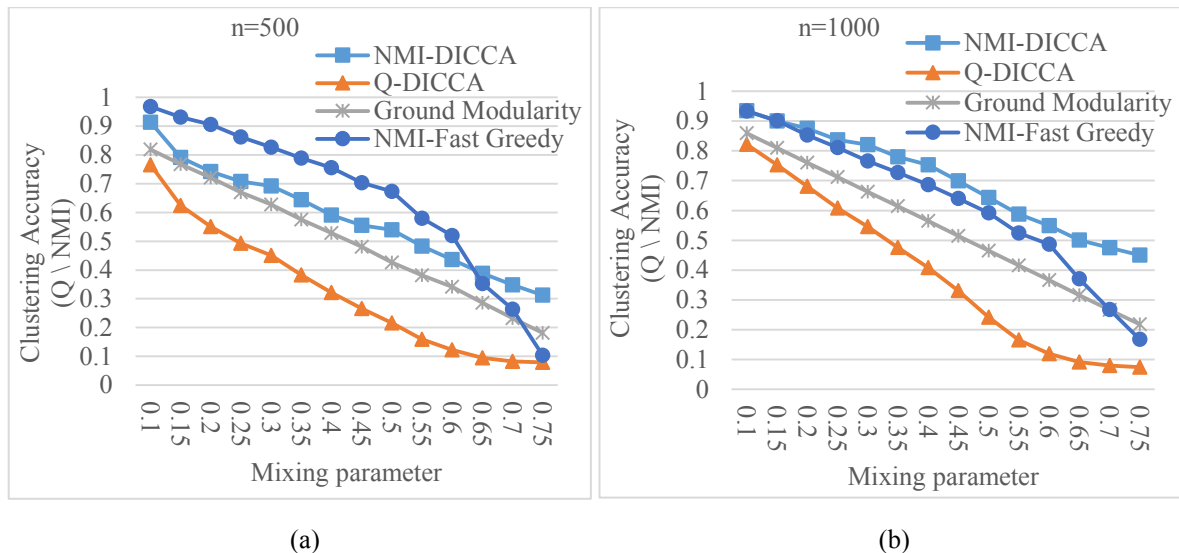


Figure 4.12 Performance of the proposed algorithm using Mixing parameter. (a) Number of node in the network is 500, (b) Number of node in the network is 1,000.

In Figure 4.12, the mean modularity score and the NMI of the partitions compared with the ground truth communities as a function of the mixing parameter are shown. As can be seen, the proposed algorithm has a similar performance for both networks of size 500 and 1,000. However, on a closer look, the algorithm performs very well for the mixing parameter value ≤ 0.5 and provides a good match to the ground truth. In contrast, for mixing parameter values ≥ 0.5 , its performance drops with respect to both NMI and the modularity scores of its network partitions.

Also, it should be noticed that with increasing value of mixing parameter, the modularity of both the DICCA algorithm and ground truth network is decreasing. This can be justified by the fact that when the mixing parameter becomes more than 0.5 many of the edges will fall outside the communities and so the communities become rather indistinguishable. In other words, for smaller μ the network exhibits a clear community structure, as per the definition of a

community in a strong sense that each node should have more connections within the community than with the rest of the graph (Silva and Zhao, 2016). Therefore, for higher μ , the network starts to show a multipartite structure and it most closely resembles the network that does not display any community structure. However, the modularity index of clustering results obtained by the proposed algorithm is gradually lowering compared to the ground truth network modularity index.

Furthermore, the modularity of both the ground-truth clustering network and the results achieved by the proposed DICCA algorithm are shown along with the clustering obtained using the fast greedy modularity optimisation proposed by Clauset, Newman and Moor (Clauset, Newman and Moore, 2004). This comparison reveals that the poor performance of the proposed DICCA algorithm for mixing parameter value ≥ 0.5 is not due to the failure of the algorithm but rather due to the network structure.

4.4.6 Evaluation of Clustering Performance Using Adjacency Matrix

Representations

To further investigate the quality of the clustering performance of the DICCA algorithm, the spy plot of the input networks and the community clusters obtained by the DICCA algorithm are shown as examples in Figure 4.13 for network size of 500, 2,500 and 5,000 nodes respectively. Graphs (a, d, g) in Figure 4.13 show the spy plot for the connections of the input networks where the graph structure is hardly visible. Graphs (b, e, h) in Figure 4.13 show the spy plot obtained after rearranging the network according to ground truth community structure and graphs (c, f, i) in Figure 4.13 present the spy plot obtained after rearranging the network according to the clusters that they were assigned to by the proposed DICCA algorithm. Note that the clusters are ordered based on the number of nodes in the community cluster where the cluster with the most nodes is located on the top.

In the Figure 4.13, each blue dot corresponds to an element of the adjacency matrix that has the value one, the white areas correspond to elements with the value zero. It can be easily observed from the plot that the adjacency matrix visualizes strong clusters as solid rectangles and the DICCA algorithm performs quite well in arranging the nodes into different clusters. The algorithm discovered 13, 74 and 150 cluster structures with modularity values of 0.776, 0.857 and 0.864 for the final clustering result of 500, 2,500 and 5,000 network size respectively, which corresponds to a very good community structure between the nodes. The number of clusters in the actual partitions for the corresponding networks (500, 2,500 and 5,000) are 13, 91 and 171 respectively.

To further assess the similarity of the solutions, another metric called ARI was considered. ARI is based on pair counting. Although this metric has different bias compared to NMI, which is based on information theory, in general, the results show the same trend as NMI. The results are included in the appendix A.1 along with the exact values of the NMI and Q performance measures.

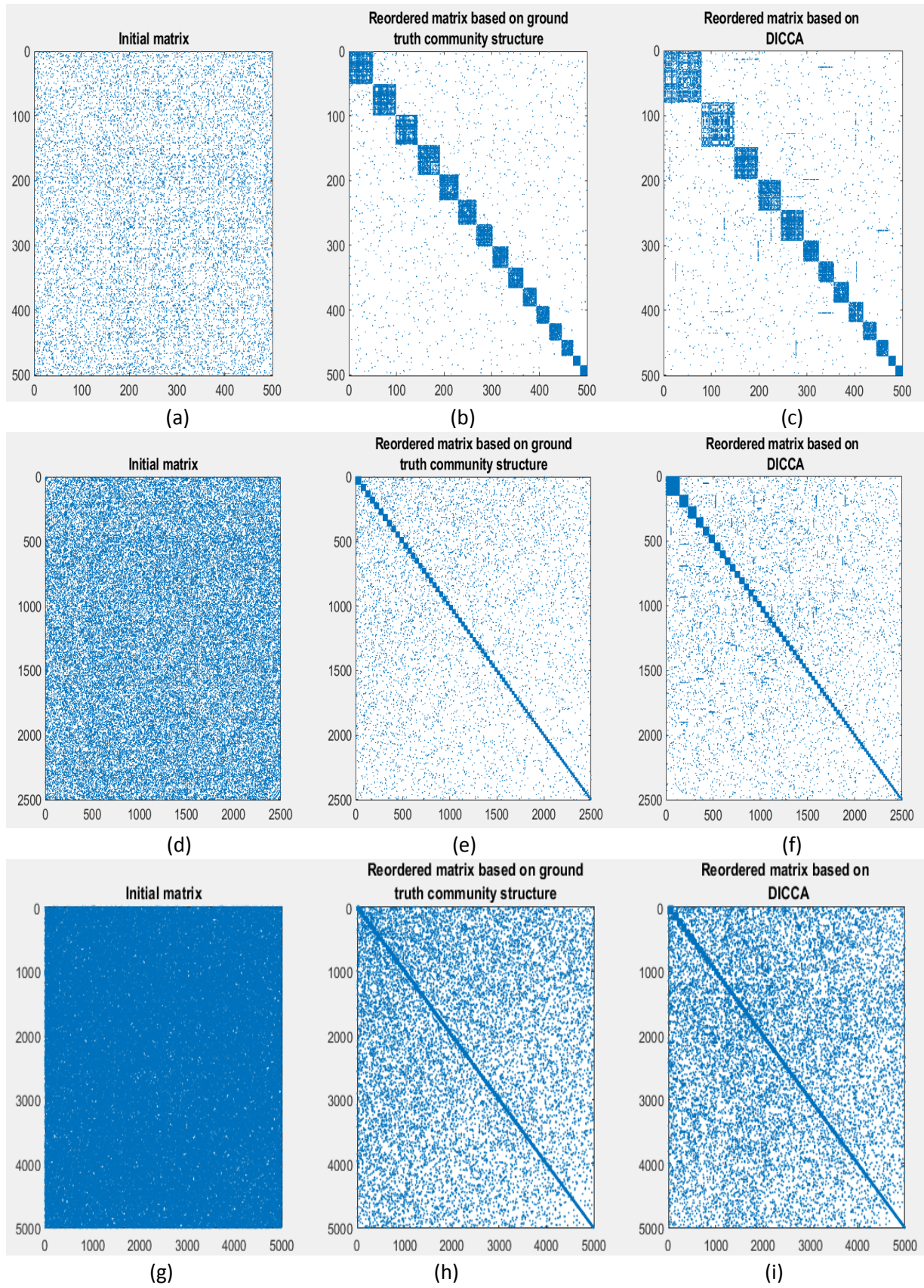


Figure 4.13 Spy plot for the connections of the nodes.

4.5 Summary

In this chapter, a novel Decentralized Iterative Community Clustering Approach (DICCA) to extract an efficient community structure for large social networks has been presented. DICCA is based on random walk and reachability, which is done by message propagation between neighbours. The algorithm consists of two phases that are run in an iterative fashion. First, it must determine all originators in the network, which could be seen as cluster centres, and assign each node to the community whose originator is densely connected. The second phase is to build new networks based on the detected communities in the first phase where each community becomes a node and the edges in the new network are representing the sum of the edges between two communities. The DICCA algorithm uses two parameters named threshold value and time to live (TTL). The threshold value should be ideally specified by the expert according to domain knowledge. However, when this knowledge is not available, optimum parameter values should be estimated. In this work, the mathematical model to obtain optimal threshold value based on the characters of the networks is presented. In addition, the optimal value of the TTL parameter is discussed. The DICCA algorithm is demonstrated with an artificial network and the output shows very promising results.

Regardless of the threshold calculation method, the algorithm is simple and its concept does not require any global knowledge. Being a localised algorithm, it can be run in parallel or in a distributed fashion among clusters when the size of the input network or the computation complexity is beyond the resources of a single computer. In the following chapter the main challenges to be addressed when designing and implementing the distributed framework version of the algorithm is discussed.

CHAPTER 5

PARALLEL DECENTRALIZED ITERATIVE COMMUNITY CLUSTERING APPROACH (PDICCA)

In the previous chapter, a standalone approach named DICCA has been proposed for identifying community clusters, which is self-organised and does not require any global information of the network. In this chapter, an extended version of the DICCA called Parallel Decentralized Iterative Community Clustering approach (PDICCA) is proposed. The PDICCA approach is parallel in that it does not require any global knowledge of network structure when the data is distributed across several machines and strict synchronization between the distributed datasets is not required.

5.1 Introduction

Faced with the challenge of a big dataset, many researchers pay great attention to parallel and distributed clustering algorithms that would improve the bottleneck of traditional clustering methods on a single machine. To cope with this scenario, a distributed and parallel computing model is needed to process a large dataset by scaling the dataset out to multiple machines across a cluster and process it. Some novel parallel computing frameworks shine, of which MapReduce is one of the most popular (Dean and Ghemawat, 2008).

In this chapter, a Parallel Decentralized Iterative Community Clustering approach (PDICCA) is proposed. The design of the PDICCA approach follows master/worker configuration, with one master serving as coordinator of many workers. In this case, of master/worker configuration, the master is not required to do the job allocations nor does it need to have the overview of the data itself. The purpose of the master in this configuration is to purely compile the results from the slave workers at the end of each iteration. These features allow PDICCA

to be easily adapted to a distributed graph processing system from data centres to fully distributed networks.

The PDICCA transforms the operations of the DICCA approach which is a serial process, into a parallelised approach. The PDICCA is a pipelined parallel implementation and maintains the overall structure of the serial method (DICCA) presented in the previous chapter. The novelty of the design comes from the following fact: even though the PDICCA solves the same problem and maintains the overall structure as does the serial method, the proposed approach is distinguished due to the features of exploiting the use of distributed memory and extracting parallelism under the MapReduce framework. The proposed algorithm does not require any global knowledge of the network topology, and is scalable and will work with a range of computer architecture platforms (e.g. cluster of PCs, multi-core distributed memory servers, GPUs), where, the master and slave workers could represent either different threads in a single machine or different machines in a computing cluster. Also, one of the main contributions of this chapter is to take advantage of the graph partitioning when performing parallel community clustering in order to speed up the process by minimizing the communication between slave-workers. Furthermore, a parallel implementation of PDICCA based on the most popular MapReduce model to accelerate processing in large-scale networks is proposed.

Table 5.1 Comparison between DICCA and PDICCA

Algorithm	DICCA	PDICCA
Process approach	Serial process approach	Parallelised process approach
Concept of the algorithm	Based on the random walk procedure and reachability of nodes in the network	Based on the random walk procedure and reachability of nodes in the network
Framework	Consists of two phases: local clustering and network reduction phase that run in an iterative fashion	Consists of three phases: clustering, re-clustering and rebuilding phase that run in an iterative fashion
worker schemes	Work in one single machine	The approach consists of two worker schemes: master and slave-clustering workers
Mismatching node	Not applicable	Use cluster strength to find best result for mismatching node
Parameters	Uses two parameters, Time To Live and threshold value	Uses two parameters, Time To Live and threshold value

5.2 Description of the Proposed PDICCA Approach.

The core idea of my proposed approach is to divide the dataset into blocks, and then iteratively repeat the following three phases: clustering, re-clustering and rebuilding phase: the clustering phase is responsible for finding local community clusters for each block independently and in parallel. In the second phase, the local clusters thus extracted from the individual blocks are aggregated to find the initial community clustering for the entire network. The third phase involves building a new, but smaller network for each block of data based on the initial community clustering. Each cycle of this process through all the three phases is referred to as an iteration. The three phases iterate until the old and the new community-clustering list does not converge anymore.

5.2.1 Framework of the PDICCA Approach

The PDICCA approach consists of two worker schemes: master and slave-clustering workers. The master worker creates the blocks as it reads the dataset, and passes them to slave-clustering workers. The master worker is also responsible for receiving and aggregating the cluster assignment results from all the slave-clustering workers, perform some computation, assign the overlapped nodes into the best community and return the final solution. On the other hand slave-clustering worker's functionality is to identify local communities by going through its own data set and applying the first phase of the DICCA approach proposed in chapter 4, named local clustering phase. The overview of PDICCA approach is shown in Figure 5.1.

Slave-clustering worker runs in parallel and stores the community clustering lists in its local memory. However, since each slave-clustering worker has some part of the data and does not have a global knowledge of the network, consequently, different slave-clustering workers could cluster the same node into different communities. Thereby, when all the blocks are clustered

and the local communities have been identified, the master worker loads the local community-clustering lists to aggregate.

Since the PDCCA approach is proposed to find non-overlapping clusters then the partition C of n nodes should form a partition such that $n = \cup_{i=1}^k C_i$ and $C_i \cap C_j = \emptyset$ for any $i \neq j$. So, the master worker is responsible for finding the set of overlapping nodes. The overlapping node list is then sent back to the slave workers to calculate the strength of clustering solutions for each overlapped node among different machines. This is then sent back to the master worker for the re-clustering phase. In the re-clustering phase, the master worker finds out the best solution for overlapped nodes, the solution corresponding to the highest strength of clustering, and updates the community-clustering list. At the end of the re-clustering phase, the network is partitioned into a number of communities.

Next step is the re-build phase, which involves building a new network by each of slave-clustering workers. Using the same method presented in section 4.2 where the nodes in the new network are the communities from the re-clustering phase. The weight of the link between two nodes in this new network is the total weight of the links between the nodes of the two corresponding communities in the original network. The links between the nodes of the same community become self-loops of the corresponding node in the new network.

The iteration is then repeated until a stable set of community clusters (fulfilling the convergence condition) is obtained.

It is to be noted that each slave-clustering worker has its own private non-shareable memory and there are no communications between the workers in the clustering phase. Thus, each slave-clustering worker operation is independent of the others and each of the slave-clustering worker's operations can be performed in parallel.

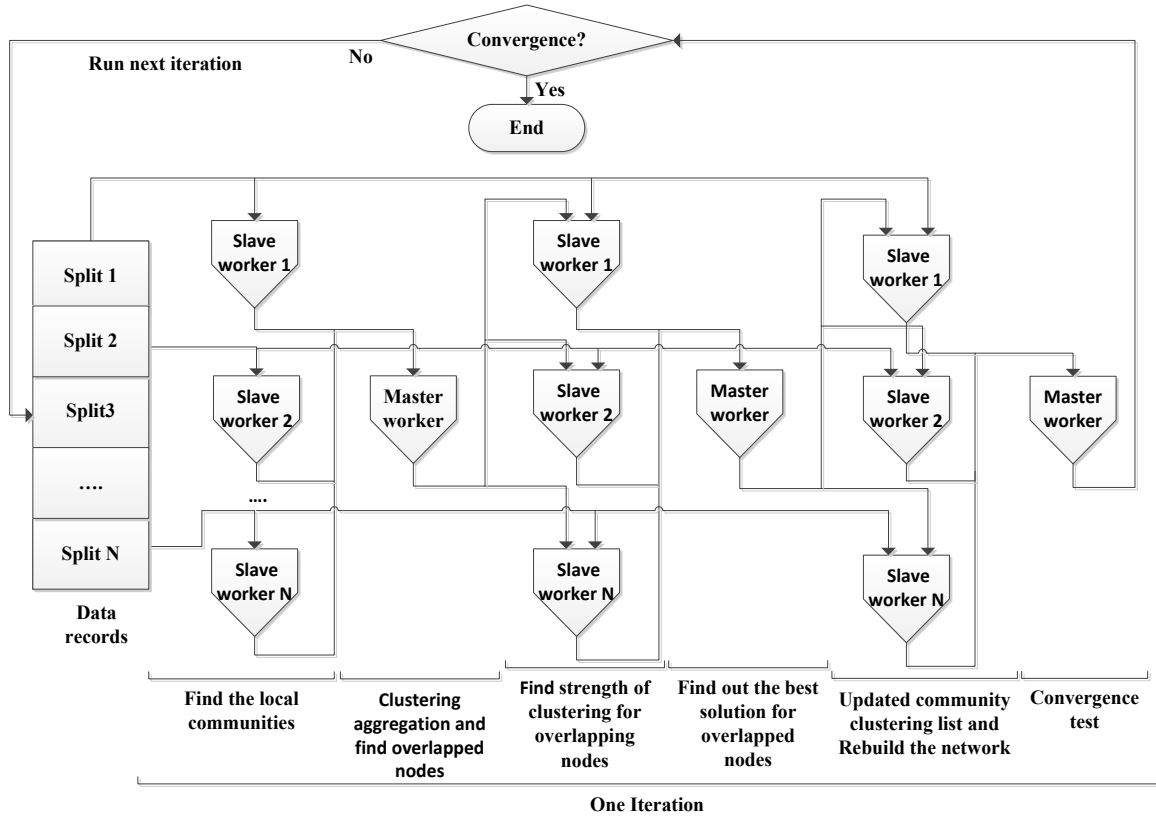


Figure 5.1 Framework of the PDICCA approach.

To calculate the strength of overlapped nodes, the clustering strength of overlapped node V_m is formalised in the following definition:

Definition 5.1 Cluster strength

Given a network set $G = (V, E)$, with $n = |V|$ nodes and $m = |E|$ edges is presented. During the clustering phase, each slave-clustering worker clusters these nodes into C clusters and assigns V_m node to different communities. To find the best community that fits V_m node, the proposed scheme carries out the following two steps:

First, the node V_m obtains two sets of information from each of its neighbours, namely, the degree of the neighbour node and the cluster to which it belongs to, and then calculates the neighbour attraction between V_m and its neighbour V_i , which is defined as:

$$Nbr \text{ Attraction } V_m (V_i) = \frac{W(V_m, V_i)}{\sum_{V_k \in Nbr(V_i)} W(V_i, V_k)} \quad (5.1)$$

Where $W(V_m, V_i)$ represents the weight of the edge between V_m and V_i .

Then the strength value of V_m for all the clusters (C) where V_m belongs to is calculated by computing the sum of the attractions for V_m towards its neighbours (Nbr Attraction) within these C clusters.

The pseudocode for the cluster strength of V_m to the cluster C_1 is shown in Algorithm 5.1 and it is calculated as follows:

$$\text{Cluster strength } (V_m, C_1) = \sum_{V_i \in C_1 \& V_i \in \text{Nbr}(V_m)} \text{Nbr Attraction } V_m(V_i) \quad (5.2)$$

Algorithm 5.1 The Cluster strength

Function Cluster strength

Input: underlying network graph G , V_m (overlapped node)

Output: Cluster_Id community as a final division of V_m .

Function Cluster strength (G, V_m)

for each Node $V_i \in \text{Nbr}(V_m)$ **do**

 Nbr Attraction $V_m(V_i) \leftarrow W(V_m, V_i) / \sum_{V_k \in \text{Nbr}(V_i)} W(V_i, V_k)$

end

for each C **do** // C is the Community clusters

 Cluster strength (V_m, C_i) $\leftarrow W \sum_{V_i \in C_i \& V_i \in \text{Nbr}(V_m)} \text{Nbr Attraction } V_m(V_i)$

end

 Cluster_Id = Max {Cluster strength (N_m, C_i)}

Return Cluster_Id

end function

The proposed scheme calculates how strongly the mismatching node V_m is connected to each of the existing clustering solutions and then V_m joins the cluster with the highest cluster strength value.

Refer to Figure 5.2, node ' V_1 ' has neighbour nodes (' V_2 ' and ' V_3 '), and belongs to the cluster ' C_1 ' and has one node ' V_6 ' that belongs to cluster ' C_2 ' then the neighbour attraction between node ' V_1 ' and its neighbour is:

$$\text{Nbr Attraction } V_1(V_2) = \frac{W(V_1, V_2)}{\sum_{V_k \in \text{Nbr}(V_2)} W(V_2, V_k)} = \frac{1}{3} ; \text{ where } V_k = \{V_1, V_4, V_5\}$$

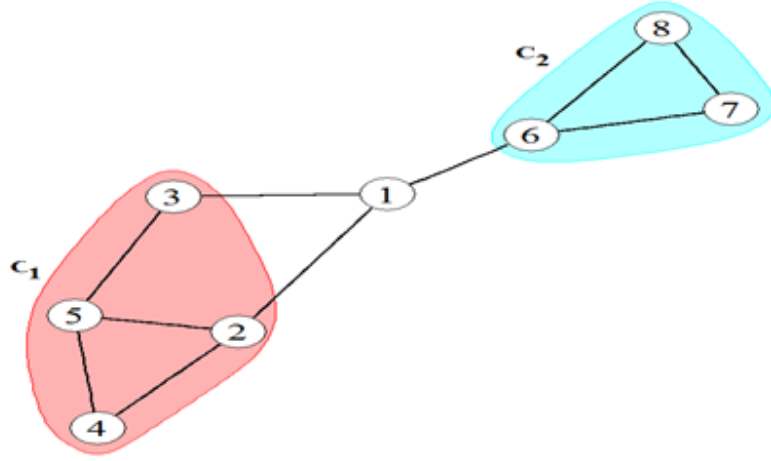


Figure 5.2 Examples of eight nodes with two community clusters

$$\text{Nbr Attraction } V_1 (V_3) = \frac{W(V_1, V_3)}{\sum_{V_k \in \text{Nbr}(V_3)} W(V_3, V_k)} = \frac{1}{2} ; \text{ where } V_k = \{V_1, V_5\}$$

$$\text{Nbr Attraction } V_1 (V_6) = \frac{W(V_1, V_6)}{\sum_{V_k \in \text{Nbr}(V_6)} W(V_6, V_k)} = \frac{1}{3} ; \text{ where } V_k = \{V_1, V_7, V_8\}$$

The cluster strength of V_1 to the cluster C_1 is calculated as follows:

$$\text{Cluster strength } (V_1, C_1) = \sum_{V_i \in C_1 \& V_i \in \text{Nbr}(V_1)} \text{Nbr Attraction } V_1 (V_i) = \frac{1}{3} + \frac{1}{2} = 0.8333$$

The cluster strength of V_1 to the cluster C_2 is calculated as follows:

$$\text{Cluster strength } (V_1, C_2) = \sum_{V_i \in C_2 \& V_i \in \text{Nbr}(V_1)} \text{Nbr Attraction } V_1 (V_i) = \frac{1}{3} = 0.3333$$

Based on the cluster strength value, the node V_1 chooses to join the cluster with higher strength, which is cluster C_1 in this example.

5.2.2 Partitioning of the Network Nodes Set

It is worth mentioning that in this work, for the purpose of computation, network nodes are partitioned with the same size and they are assigned to different workers. This enables the workers to serve a similar size of network.

It would be beneficial for the nodes close to each other to be processed on the same worker, since this will increase the local computing and decrease network transfer (cost of bandwidth) caused by overlapped nodes (Kajdanowicz, Kazienko and Indyk, 2014). Unfortunately, the

network partitioning requires a priori knowledge of the global picture of network structure, which is a resource-consuming task, especially for large network structures. For this reason, in this work the partitioning aspect of the network is done randomly with the consideration that the number of edges in each partition should be the same.

5.2.3 How to Calculate the Parameters

As mentioned in the previous chapter, DICCA approach uses two parameters to be defined. The first parameter ‘Time To Live’ (TTL) is defined as the number of hops that a message is permitted to travel before being discarded. The next parameter is threshold value that determines the difficulty of merging communities and is defined by the equation presented in the previous chapter. However, in the PDICCA approach, TTL is set to be 3 (optimal value obtained from chapter 4) and the threshold value for each worker is calculated based on its local view of data and using the equation 4.3 presented in chapter 4.

5.3 Matlab Implementation of PDICCA Approach for

Distributed Memory Systems

To implement the PDICCA approach in a parallel manner, the Parallel Computing Toolbox (PCT) available in the Matlab software platform is used (MATLAB, Release 2017a). PCT enables computational solution of data intensive problems using multicore CPUs, GPUs and computer clusters. In PCT to start a parallel processing, the MATLAB pool is opened to reserve a collection of MATLAB worker sessions that run separately on the local machine or on a remote cluster. In the PCT toolbox the loop command “parfor” is included. By using parfor, for each worker a separate process is created with its own memory and own CPU usage. The workers are headed by a client process which creates and manages them. When parfor is executed, the MATLAB client coordinates with the MATLAB workers which form a parallel

pool. The code within the parfor loop is distributed to workers and it executes in parallel in the pool. The required data needed by workers to do the computations is sent from the client to all the workers and the results from all the workers are collected back by the client as shown in Figure 5.3.

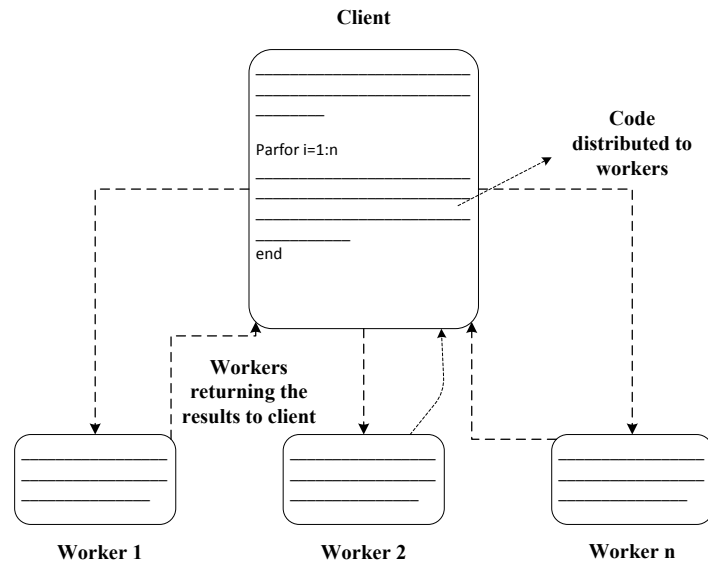


Figure 5.3 Parfor mechanism.

In this work, the algorithm is implemented on a multi-core machine to which two or more independent processors are attached. The client divides the work among multiple processors by allocating different data to the different processors (called workers). The processors run their job independently of each other and no communication can occur between workers during the execution of the loop. Each processor executes the same program but working on different sets of data, so each worker maintains its own memory stack. Furthermore, since the implementation relies on partitioning data into a number of blocks, the number of data blocks equals the number of available workers (processors) in which each worker has only one block of data to process and does not have access to the whole data.

The client loads the outputs from each worker and aggregates the outputs to do some processing, submits new instructions to workers and makes final clustering when stable condition has been reached.

5.4 Parallel Algorithms Using MapReduce Model

Since in MapReduce model it is not possible to share any information among different slave machines while running map or reduce functions, not all of graph clustering based algorithms can be fitted into the MapReduce model. However, since the idea of the PDICCA approach follows master/worker configuration, with one master serving as the coordinator of many workers, this algorithm can be directly applied to work on top of the MapReduce computing framework. As shown in Figure 5.1, the PDICCA approach is an iterative process, where each iteration can be expressed in three step MapReduce jobs. To begin with, the client submits the job to the master node of a machine cluster where the master machine will partition the input data into several parts and arrange a number of slave machines to process these input data partitions in map functions. The output of each map function will be sorted, shuffled and then routed to the proper reducer. During the iterative process, the reducer's output is directly sent to the map function for the next round of the iteration. The process is repeated until the termination condition is met and the final output is obtained. However, each Map function needs to get the same data split during each iteration.

The different stages of computation are shown in Figure 5.4 and the description of each stage follows:

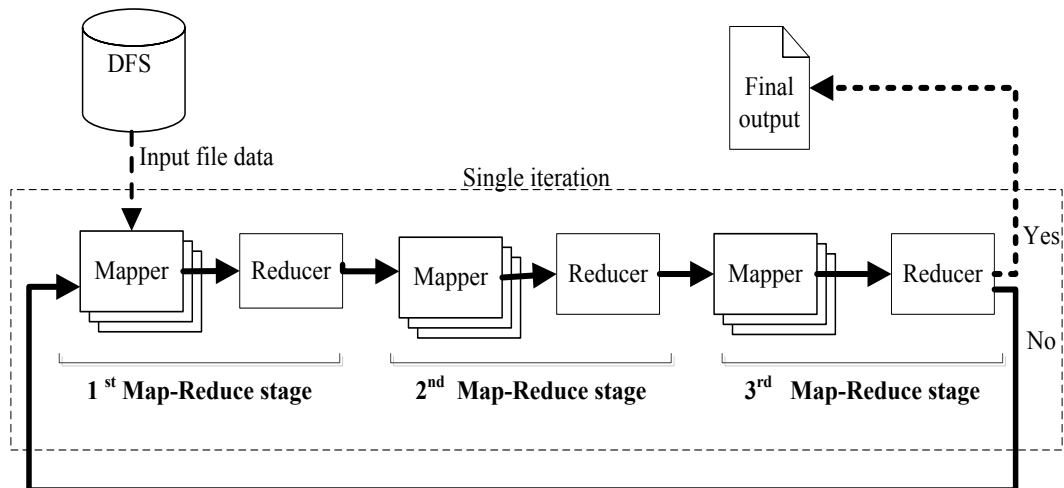


Figure 5.4 PDICCA workflow and architecture.

5.4.1 Description of Algorithm in MapReduce Model

- **Input**
 - Dataset (network) –Large
- **First Map Stage**
 - Step 1: Select one node at random (originator)
 - Step 2: apply the first phase of the DICCA approach to find the local community clusters
 - *Output* < node, cluster ID>
- **First Reduce Stage**
 - Step 3: Find overlapped node clustering
 - *Output*: <mismatched clustering nodes>
- **Second Map Stage**
 - Step 4: For each overlapped clustering node, re-compute the strength of answers
 - *Output*: <mismatched clustering nodes, strength>
- **Second Reduce Stage**
 - Step 5: Find the best answer for each mismatched node
 - *Output*: <mismatched clustering nodes, best answer>
- **Third Map Stage**
 - Step 6: Assign mismatched node to the best answer.
 - Step 7: Rebuild the network
 - *Output*: <Nodes, Cluster ID>
- **Third Reduce Stage**
 - Step 8: Compare the new discovered community and the old one (communities from previous iteration)
 - If similar → Stop
 - Else → Go to Step 1 to start another MapReduce Iteration
- **Use of Single Reducer**
 - The size of the dataset sent to the reducers is very small
 - Single reducer can tell whether any of the node is mismatched or not
 - Creates a single output file

It is worth mentioning that although the PDICCA approach is presented here using a MapReduce model, the approach can be implemented in a range of iterative MapReduce implementation frameworks such as Twister programming model that are built for iterative graph algorithms (Ekanayake et al, 2010).

5.5 Analysis of Results and Discussion

5.5.1 Environment Setup

The PDICCA approach is implemented in Matlab, a discrete event simulator for building P2P protocols. Using the LFR networks mentioned in chapter 3, several experiments have been conducted to evaluate the scalability and quality of the proposed algorithm. The experiments are performed on a system configured with 4® Core™ i7 6700K CPU 4.00GHz and 16 RAM available memory running windows. Because the approach initializes the originator randomly and in order to neglect the effect of randomness in our method each result is averaged over 100 runs.

5.5.2 Experimental Evaluation

5.5.2.1 Horizontal Scalability in Relation to the Number of Parallel Cores

To demonstrate how well the PDICCA approach handles datasets when more workers are available, the number of nodes in the network used in this evaluation is kept constant and the number of workers is varied from 1 to 4. Figure 5.5 shows the results of different cores when the number of nodes is constant, $n \in \{500, 1000\}$.

5.5.2.1.1 Quality

From Figure 5.5, the PDICCA shows a good scalability close to the optimal value, which is indicated by average modularity and NMI values. In addition, it is clear that using more than one worker to parallelise the algorithm does not adversely affect the accuracy of the result.

Consequently, the results prove that the algorithm is effective and able to achieve very high-quality results in a parallel manner. More especially, PDICCA is capable of exploiting multi-core architecture efficiently.

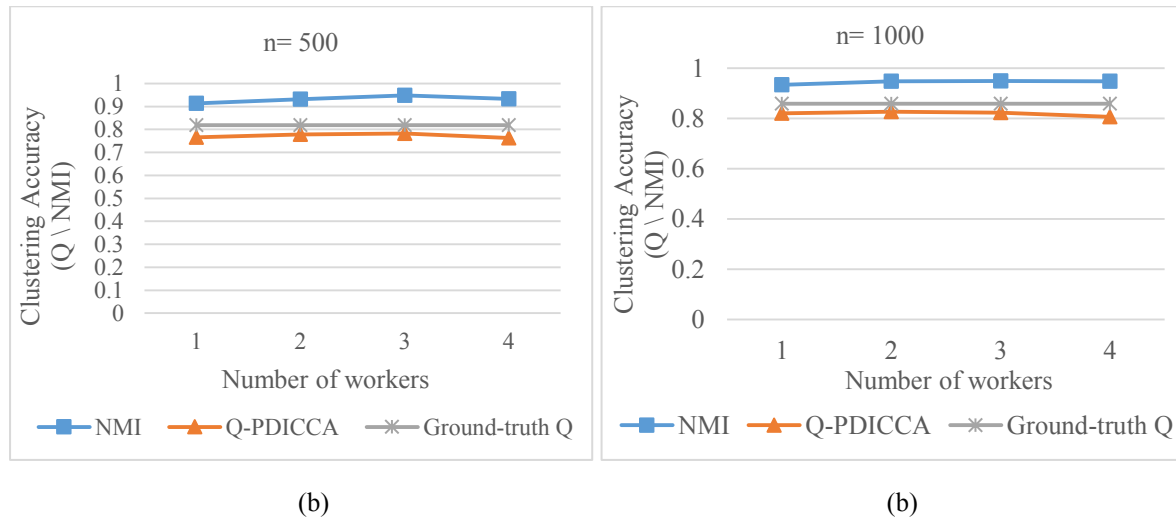


Figure 5.5 NMI, Q-PDICCS and Ground truth Q scores (y-axis) as number of workers (x-axis) changes number of nodes: (a) 500 (b) 1,000.

5.5.2.1.2 Message Complexity of the PDICCA Algorithm

Considering the number of exchanged messages for each worker, Figure 5.6 shows the percentage of exchanged messages at each iteration by each worker processor. As can be observed in each iteration, each worker generates almost the same number of messages, this can be clarified by the fact that the data has been partitioned equally among the workers so each worker has to process the same size of data. Hence, at each iteration, the master worker must wait until all workers have completed their processes. So, splitting the data equally over workers, can significantly reduce the expected time needed to wait until the slowest machine worker returned data.

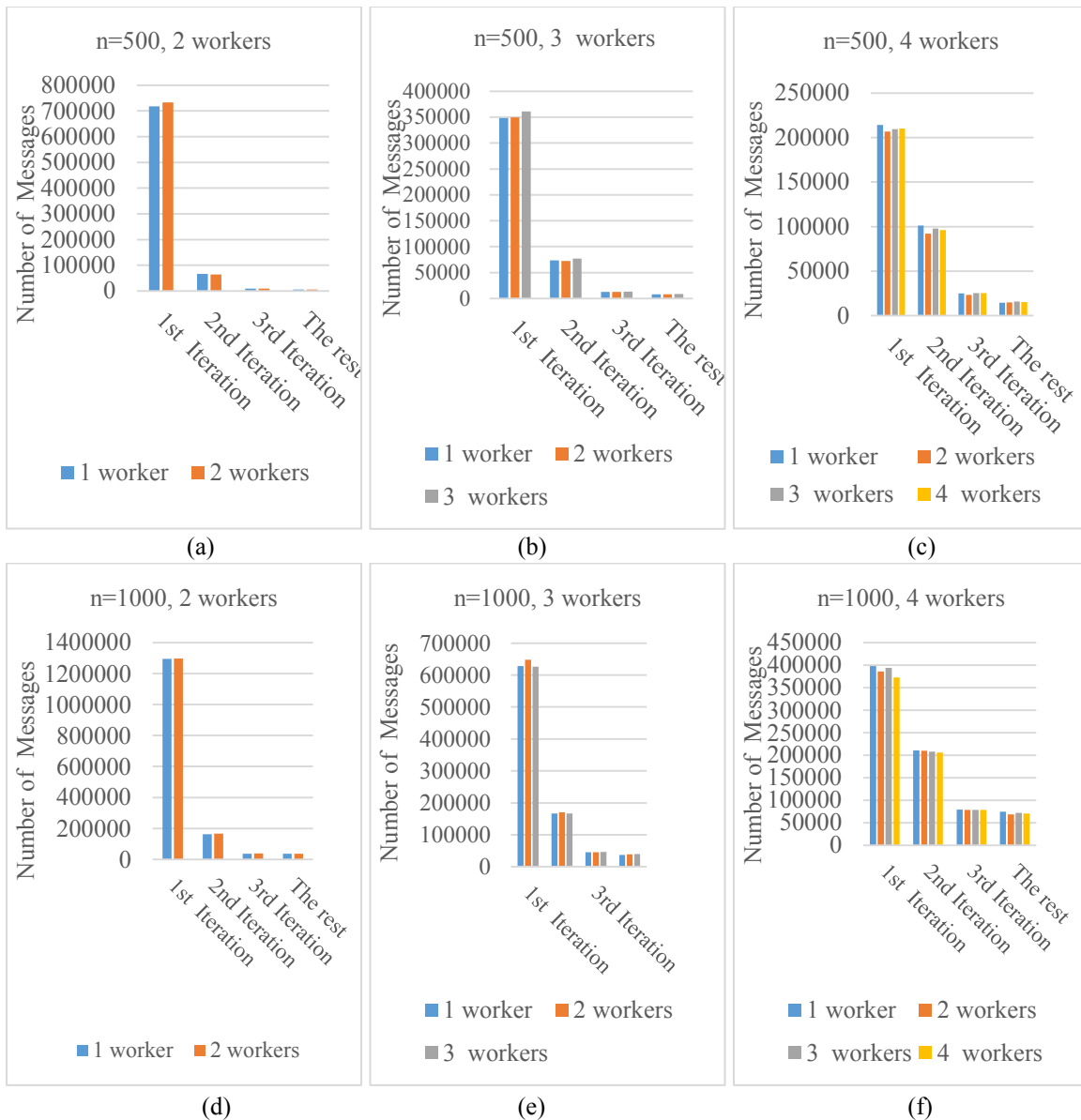


Figure 5.6 Number of Message exchanged in each iterations and for each worker with respect to the number of workers varied from 2 to 4 (a, b, c) for number of nodes 500 (d, e, f) for number of nodes 1,000.

For more in-depth analysis, Figure 5.7 shows the average percentage of exchanged messages in each iteration. It can be easily observed from the figure that data exchange for the algorithm is much greater at the first stage of iteration when each node is in its own cluster. Just after 2 to 3 initial iterations, most nodes have their cluster labels and the algorithm has merged the nodes belonging to the same cluster to be one node. It also becomes clear from the Table 5.2 that the percentage of exchanged messages between master and slaves, the communication cost,

is negligible. In comparison to the information exchanged locally in slaves which is very costly and constitutes the main body of the time consumption of the algorithm.

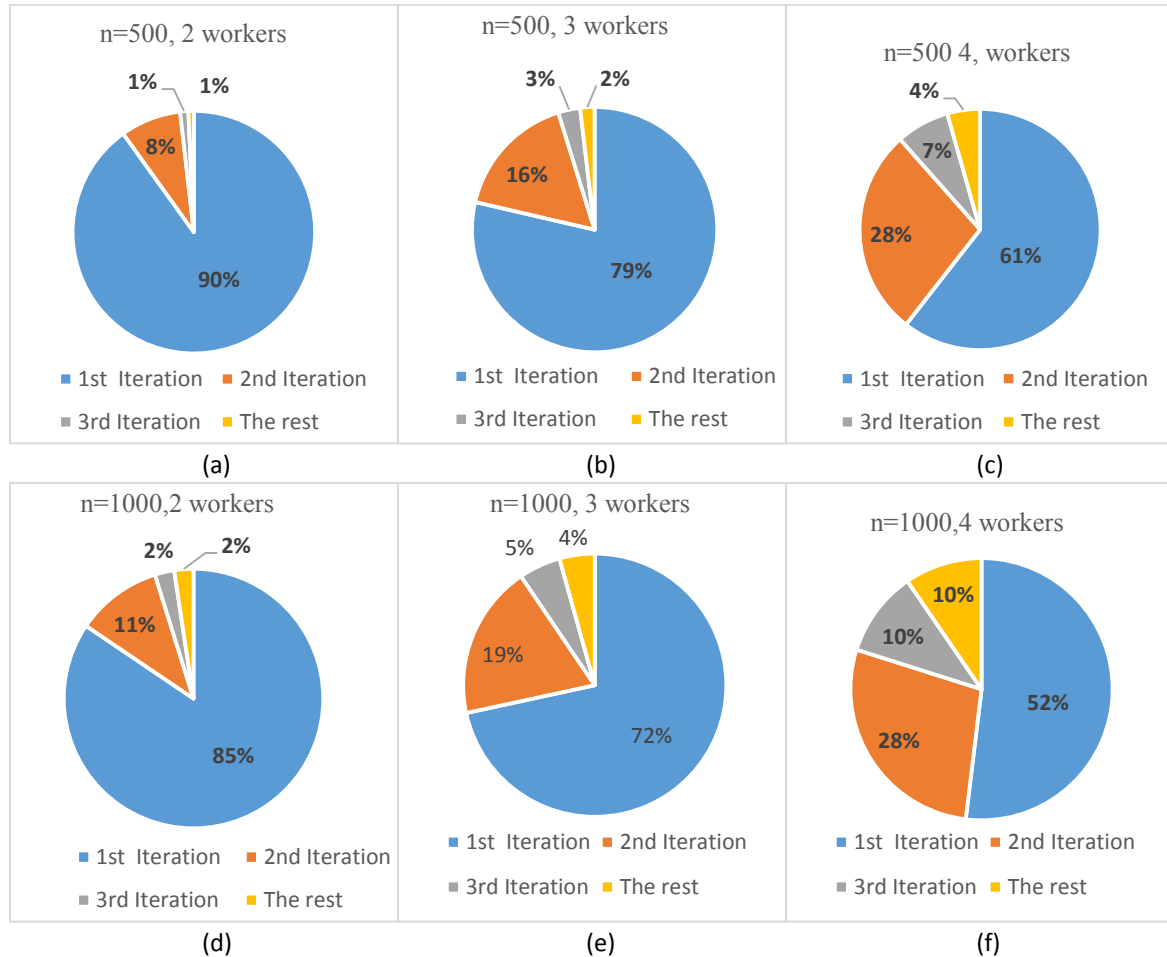


Figure 5.7 Average percentage of Message exchanged per each iteration with number of cores varied from 1 to 4 workers (a, b, c) network size 500 (d, e, f) network size 1,000.

Table 5.2 Comparison with message exchanged locally in hosts and messages exchanged between master and hosts

Number of nodes	500		1000	
No. of Workers	%Messages exchanged locally among slaves	% messages exchanged between master and slaves	%Messages exchanged locally among slaves	% messages exchanged between master and slaves
2	99.9767	0.0233	99.9760	0.0240
3	99.9636	0.0364	99.9631	0.0369
4	99.9599	0.0401	99.9629	0.0371

5.5.2.2 Clustering Results for Increasing Network Size

To demonstrate the performance influenced by scalability, the number of nodes is increased linearly from 500 to 5,000 and the number of workers is kept constant at 3. All other parameters and factors remain the same as previous evaluations.

5.5.2.2.1 Quality

The modularity values of the solutions obtained by the PDICCA approach are presented in Figure 5.8. It can be observed from the figure that the performance of the PDICCA is consistently good and close to the optimal value with NMI 0.96 and modularity 0.84 on average.

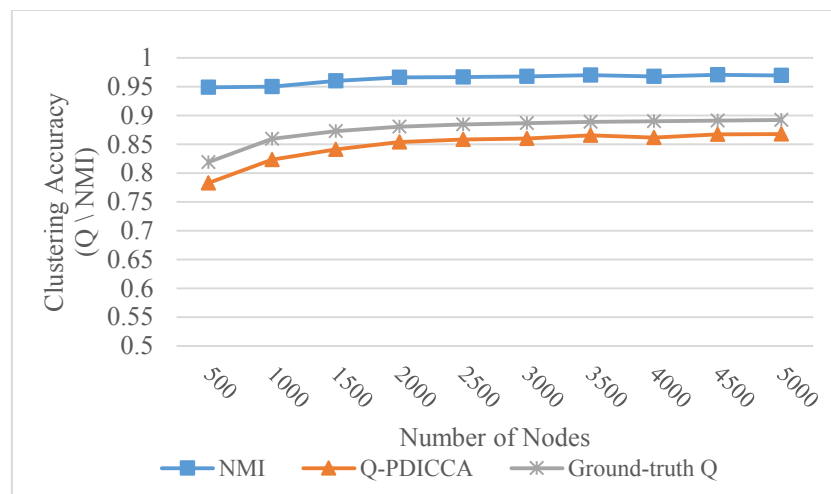


Figure 5.8 NMI, Q-DICCS and Ground truth Q scores (y-axis) as number of nodes (x-axis) changes.

5.5.2.2.2 Evaluating Repeatability of the Algorithm's Performance

To further investigate the ability of the PDICCA approach to produce consistent results across random starts across random data partitioning and initialisation, the standard deviation of the clustering results is measured where the algorithm is run 100 times each time with different random data partitioning and algorithm initialisation. The standard deviation value of both NMI and modularity for the data sets with different network size are displayed in Figure 5.9, which is relatively very small and in some cases around zero variation.

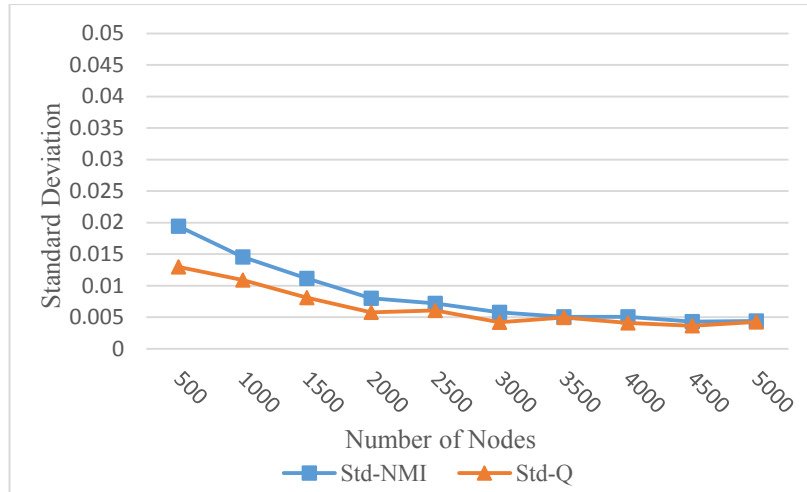
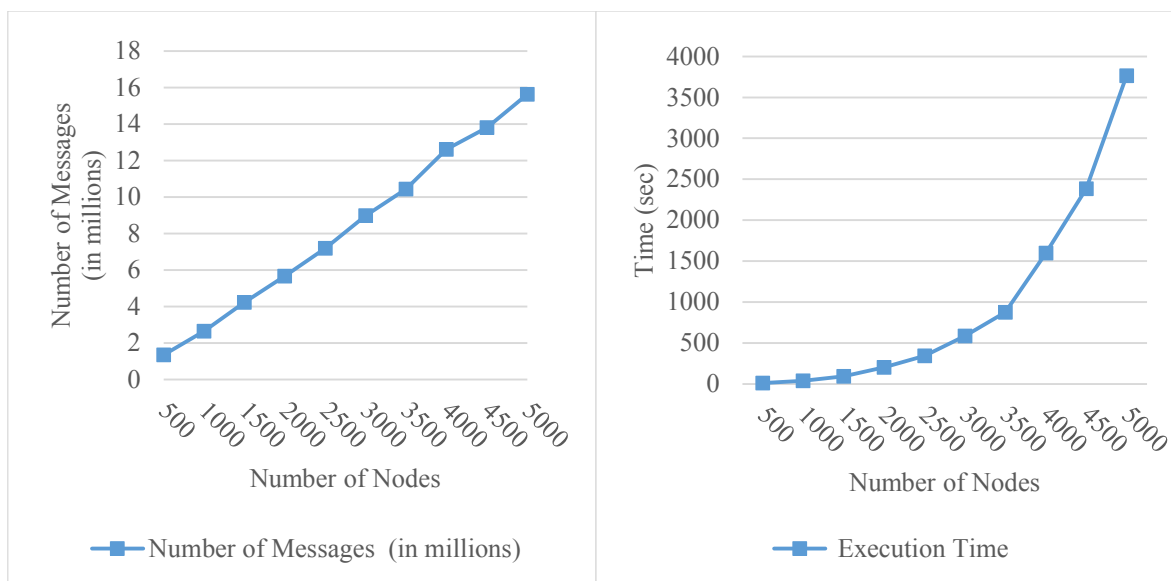


Figure 5.9 Standard deviation of final modularity/NMI with network sizes.

5.5.2.2.3 Evaluation of Complexity of the PDICCA Approach

To investigate the relationship between the number of nodes and complexity of approach, both the computing time and the total number of exchanged messages as a function of the network size are presented in Figure 5.10 (a and b). Since PDICCA requires a large number of exchanged messages between nodes, which is the most time consuming part during execution, the performance of PDICCA highly depended on the total number of exchanged messages. Therefore in this approach, the running time increases with the network size as a consequence of increasing the number of exchanged messages. For example, the computing time and total number of messages exchanged by PDICCA for $n \in \{500; 5,000\}$ are $\{8.6; 3,763\}$ and $\{1,344,282; 15,633,691\}$ respectively.

The average number of iterations and number of clustering solutions achieved are summarized in Table 5.3. As can be seen, the PDICCA usually tends to detect fewer communities than the ground truth solution. Another observation is that the number of iterations seems to depend more on the network structure than the size of network.



(b) (b)
Figure 5.10 (a) Total number of exchanged messages (y-axis) as number of nodes (x-axis) changes.
 (b).Running-time scalability of proposed algorithm in seconds.

Table 5.3 Experimental results of the PDICCA approach for increasing number of nodes in the network

Number of nodes	G.No. cluster	NMI	No. cluster	Iteration
500	16.4	0.9487652	14.9	4.65
1000	32	0.9497669	28.24	4.88
1500	51.4	0.9596985	45.79	5.01
2000	69	0.9660799	61.85	5.22
2500	87.6	0.9664238	77.84	5.11
3000	103.6	0.9675385	92.34	5.31
3500	122.6	0.9698933	108.89	5.29
4000	133.6	0.9674534	118.71	5.41
4500	154.8	0.9703297	137.01	5.35
5000	173	0.9695852	151.99	5.34

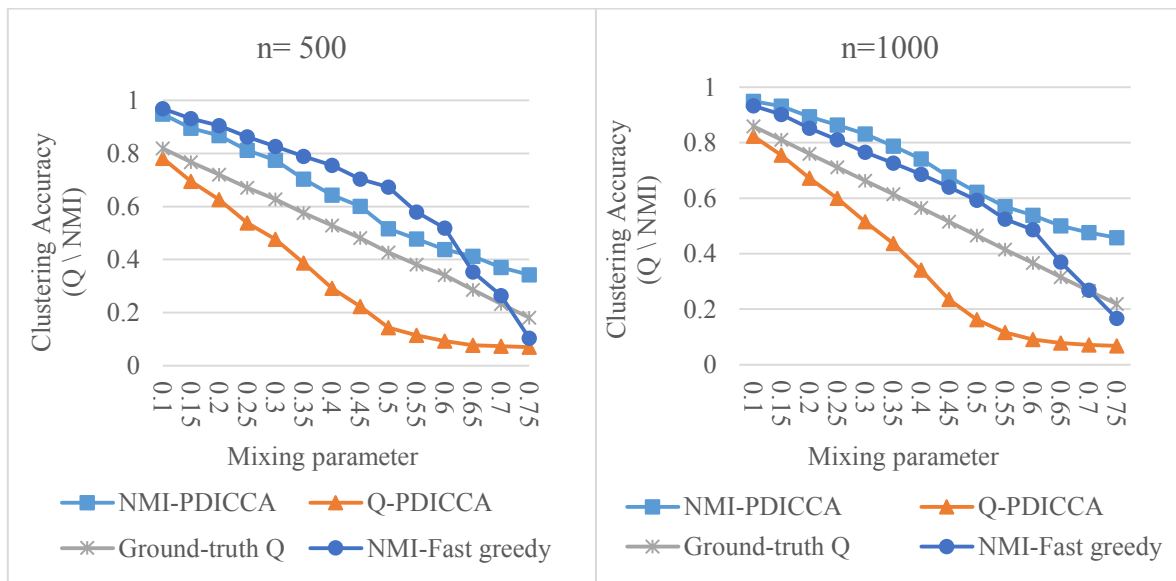
5.5.2.3 Evaluation of Clustering Performance Using Mixing Parameter

The PDICCA approach is evaluated with varying values of mixing parameter between 0.1 and 0.75, $\mu \in \{0.1, 0.15, \dots, 0.75\}$, and keeping the number of nodes constant, $n \in \{500, 1000\}$.

Figure 5.11 shows the results obtained for both modularity and NMI accuracy as a function of the mixing parameter using the PDICCA for network sizes 500 and 1,000 nodes. As can be clearly seen, the natural partitions of the network are always found (in principle) for the mixing parameter value of up to 0.5, after which the method starts to fail where the quality of PDICCA

was rather poor. However, fast greedy modularity optimisation algorithm does not have impressive performances either, and displays a similar pattern. Furthermore, the performance of PDICCA is expected to decrease as μ increases because higher values of μ indicates that the community clusters in the network are not well defined.

More results including the exact values of the Q and NMI performance measures along with ARI metric values can be found in the appendix A.2.



(a) (b)
Figure 5.11 Performance of the proposed algorithm using Mixing parameter μ . (a) Number of node in the network is 500, (b) Number of node in the network is 1,000.

5.6 Summary

In this chapter, the distributed-memory parallel version of the DICCA approach, named PDICCA, to extract an efficient community structure for large networks, is proposed. PDICCA builds around the idea of splitting data instances into blocks and then clusters each block independently and in parallel fashion across multiple cores/machines. The clusters extracted from blocks are then aggregated at the final stage using the re-clustering stage. The PDICCA approach provides several features simultaneously. Since it does not require a global knowledge of the network topology, it is effective to process massive datasets that are too large

to fit in memory. In addition, PDICCA addresses the computationally intensive issues and utilizes maximum hardware capabilities of modern multi-core systems for faster execution by processing multiple blocks in a parallel manner. Furthermore, when scalability issues occur as the data size grows beyond the processing power of a single machine, the proposed distributed approach based on the MapReduce computing platform will help address this. Finally, in this chapter the effectiveness and complexity of the PDICCA approach is tested and analysed using synthetic networks with ground truth communities. The experimental results of the PDICCA approach prove promising.

Since the nodes in the network contain a large amount of attribute information, this attribute information has important significance in completely presenting the community structure of the network. For example, in a social network, members of the same organisation are not only friends but also they are more likely to have common interests or common individual attributes. Therefore, in the following chapter, the approach which utilizes attribute information, shared neighbours' information and connectivity between nodes in the network to extract communities, is proposed.

CHAPTER 6

A PRE-PROCESSING APPROACH FOR ROBUST COMMUNITY CLUSTERING TECHNIQUES BASED ON COLLABORATIVE INFORMATION SOURCES

In this chapter, a pre-processing approach for improving the robustness of community detection in the existing weighted community detection algorithms, especially in networks with missing information is proposed. This is done through considering attribute information, shared neighbours' information and connectivity between nodes in the network, for the detection process. Empirical results demonstrate that the proposed approach is robust and can detect more meaningful community structures within incomplete information networks than the state-of-the-art methods that consider only topology information.

6.1 Introduction

In many real-world network structures such as social networks and the World Wide Web, in addition to the link information, nodes are accompanied with their attribute values referred to as attribute/content information. For example, in a social network, the nodes' properties could describe the roles of a person while the topological structure represents relationships among a group of people.

A fundamental property in network is the community structure. Another property of similar interest is transitivity or global coefficient clustering, which is defined as the tendency among two nodes to be connected if they share a mutual neighbour (Newman, 2001). In terms of network topology, recall from chapter 3 equation 3.4 transitivity defined as the presence of a

heightened number of sets of three vertices with edges between each pair of nodes (triangles) in the network.

Empirical studies have found that the concept of transitivity applies in about 70–80% of all cases across a variety of small group situations (Davis, 1970; Louch, 2000). Huijuan and Shixuan (2013) proposed a graph clustering algorithm called SNGC that considers both connectivity between nodes and shared neighbours. Their experimental results show that the proposed algorithm provides promising results and could be applied to the analysis of social networks, computer networks, bioinformatics, etc.

Another common occurrence in networks is that similar nodes associate with each other more often than with others (e.g. in social networks, people choose to be friends with people who share their beliefs). This property is known as Homophily (McPherson, Smith-Lovin and Cook, 2001). Traud et al (2011) show that a set of nodes' attributes can act as the primary organising principle of the communities. Several studies have been performed to investigate this phenomenon of Homophily, which is summarized in McPherson, Smith-Lovin and Cook (2001).

Most of the existing approaches found in the literature make use of either link information or attribute information analysis alone for community detection. However, in real-world networks neither piece of information on its own is sufficient in determining good clusters of the network. The link information is usually sparse and noisy. On the other hand, relying on the attribute information alone could mislead the process of community detection. For example, the process may not identify the strength of a node's relationship with its neighbours correctly. Consequently, by taking into account only one source of information, the algorithm may fail to detect accurately the entire community memberships. However, considering more than one source of information for community detection could produce meaningful clusters and improve the robustness of the network. For instance, in the case of attribute information, shared

neighbours and connectivity information are considered, then if either one source of information is noisy or missing, the other could make up for it. Therefore, the proposed approach will consider attribute information, shared neighbours and connectivity information aspects of the network for community detection. It should be noted that this work does not attempt to introduce a new community detection algorithm; rather proposes a pre-processing step to improve existing community detection algorithms and make them execute with better results in unreliable data network environments.

In this chapter, a network is represented as an undirected network $G = (V, E, A)$, where V is the set of nodes, E is set of edges between nodes. Each node $V_i \in V$ is associated with an attribute vector (A_i^1, \dots, A_i^d) . Where d is the attribute dimension and i represents the node ID.

The main goal of this work is to find K non-overlapping communities in the network where the community (C) is defined as a list of non-empty node subsets: $C = \{C_1, C_2, \dots, C_k\}$, $\cup_{i=1}^k C_i = V$ and $C_i \cap C_j = \emptyset$ for any $i \neq j$.

6.2 Related Literature and Contribution

During the past decade, the problem of community detection in networks has drawn a great deal of attention and several algorithms have been proposed. However, most of these existing methods use either link information or attribute information alone for detecting communities in the networks. Recently, there have been several studies (Dang and Viennet; Yang et al, 2009; Zhou, Cheng and Yu, 2009; Lin et al, 2012; Ruan, Fuhry and Parthasarathy, 2013; Salem and Ozcaglar, 2014) showing that the combination of attribute and link information to detect communities in a network can improve the clustering quality. Most of these studies propose new algorithms that aim to use both sources of information; however, their success relies on the completeness of the dataset. Moreover, most methods use all attributes the same way without considering which ones may influence the community structure more, and lack the

flexibility of balancing the information coming from network adjacency matrix and its node attributes. Additionally, none of the studies examines the quality and the number of community structures that could be identified in the network when some of the links are missing i.e. noisy network environment. So, to the best of our knowledge, this is the first study on the community structure that seeks to:

1. Design a unique pre-processing approach for the state of the art community detection algorithms by tightly integrating the attribute information, shared neighbours and connectivity information aspects of the network to produce a new matrix.
2. Study the correlation between communities and attributes in the network and introduce weight detection attribute model to learn the degree of contributions of different attributes based on the impact of attribute on the community structure.
3. Evaluate the performance of pre-processing approach within incomplete, noisy, networks.

6.3 Experimental Datasets

In order to investigate the correlations between attributes and community structure and to evaluate the proposed approach, anonymised Facebook datasets as introduced by Traud et al (Traud, Mucha and Porter, 2012) and (Traud et al, 2011) are used. The Facebook datasets are undirected and unweighted. The datasets were recorded on a particular day in September 2005 and contain Facebook networks from 100 different American university networks whose nodes represent users and the links represent friendships between users. Attribute information about each user is also provided. Each user has seven node attributes: a student/faculty status flag, gender, major, second major/minor (if applicable), dormitory (house), year and high school. In this work four networks from 100 Facebook datasets are used. In particular, the Caltech36, Reed98, Haverford76 and Vassar85 datasets, which contain 769, 962, 1,446 and 3,068 nodes

and 16,656, 18,812, 59,589 and 119,161 edges respectively are used. However, the proposed approach in this work is not limited to the social networks but can be applied to many kind of graph structures.

6.4 Correlation Analysis

6.4.1 Shared Neighbours

In order to measure how likely any two nodes with a common neighbour are themselves connected, the clustering coefficient of each node in the network is calculated.

Recall from chapter 3, the node clustering coefficient C_i , of a node i is defined as the ratio of the number of edges connecting the neighbours of i to the total possible number of such edges of i , K_i is the degree of node i .

$$C_i = \frac{2L_i}{K_i[K_i-1]} \quad (6.1)$$

Where, L_i is the number of edges between neighbours of node i (Costa et al, 2007).

The clustering coefficient for the whole network is the average of the local values C_i .

$$C = \frac{1}{n} \sum_{i=1}^n C_i \quad (6.2)$$

Where n is the number of nodes in the network (Costa et al, 2007).

Figure 6.1 shows the visualization results of the cluster coefficient for each node in the four datasets. In this figure, colours of nodes correspond to values of their corresponding clustering coefficients. As can be seen, there are some nodes that have high clustering coefficients, which indicates strong connectivity between each other. In the other words, they are more prone to be in the same cluster. Furthermore, the clustering coefficient for the considered networks are 0.4288, 0.3304, 0.3268 and 0.2487 for Caltech36, Reed98, Haverford76 and Vassar85 datasets respectively.

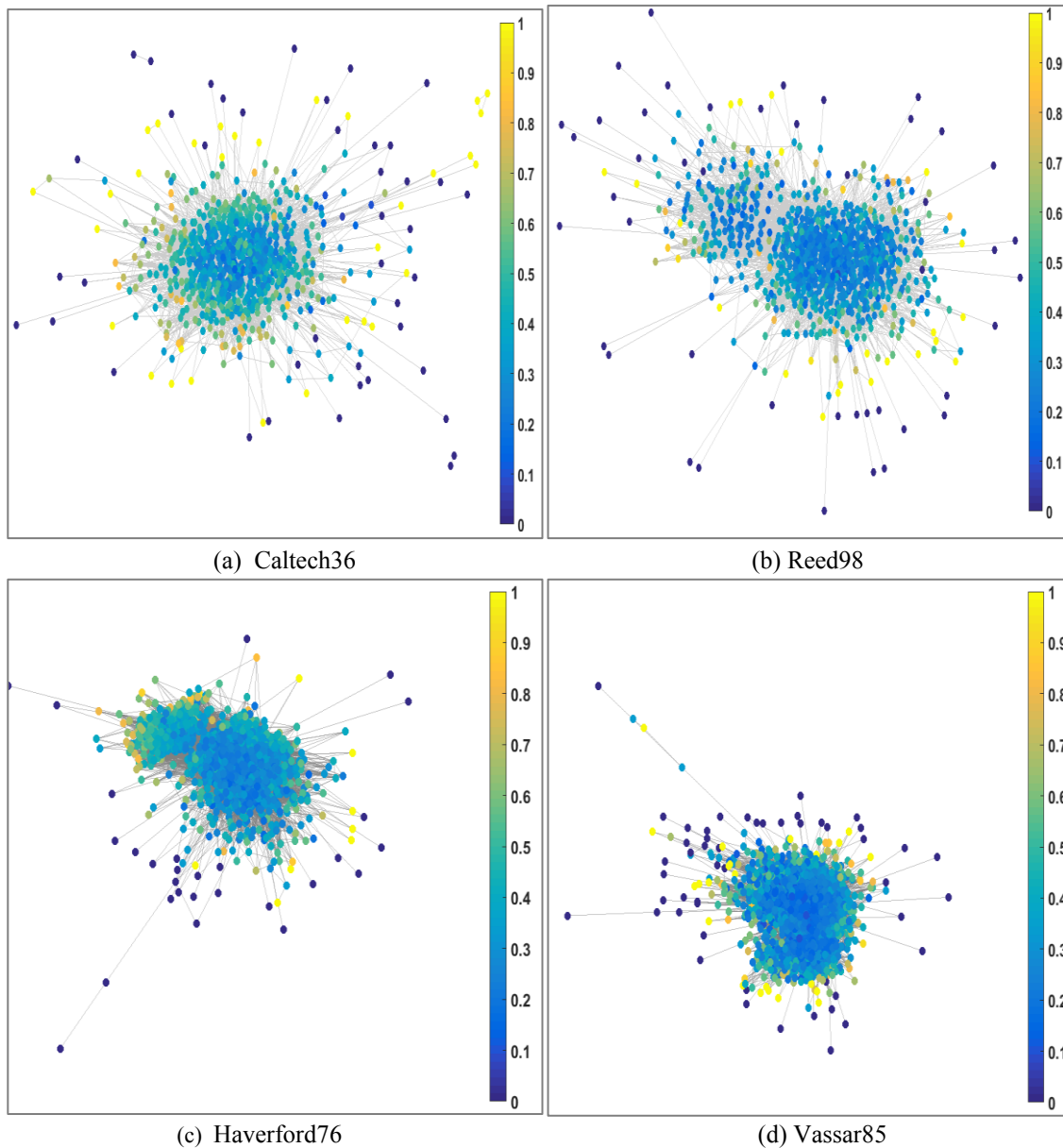


Figure 6.1 Visualization results of node clustering coefficient for subset of four datasets (should be viewed in colour).

Therefore it is clear from the above discussion that the shared neighbours' information can be used to describe the nature of connections between nodes in the network. This should motivate the use of shared neighbours' information in detecting community clusters in the network.

6.4.2 Correlation of Communities and Attributes

For the sake of computing the correlation between connectivity of nodes and their attributes, the nodes are clustered based on their attributes in which, the nodes whose attributes are similar are grouped together to form a cluster. Also, four different community clustering algorithms,

which are FastModularity (Clauset, Newman and Moore, 2004), Louvain (Blondel et al, 2008), leading eigenvector algorithm (Newman, 2006a) and WalkTrap (Pons and Latapy, 2005) are applied on the datasets to find the communities. Then the correlations between the resulting communities from these algorithms and the attributes are measured using Jaccard similarity index.

Figure 6.2 shows the correlations between attribute and communities clustering for Reed dataset. The visualization is done using R with the help of the Igraph package (Csardi and Nepusz, 2006). From this figure some of the correlations between attributes (colours) and the community structure can be observed.

Figure 6.3 presents the Jaccard similarity index for four different community detection algorithms with each attribute over the four networks in the Facebook dataset. It is interesting to notice that for the same dataset, the order of the correlation strength across different attributes is not the same and varies from one community clustering algorithm to another. For example in Reed98 dataset, if the agreement with the fast modularity algorithm is considered, the most agreement is observed with the attribute 'student faculty'. On the other hand, Louvain algorithm performs the best if the agreement with the 'year' is considered. This is due to the fact that each algorithm differs on how they treat the nodes and assign them to different communities with different size and number of communities.

Even though there exists a difference in attribute ranking across different algorithms and datasets, as an overview, the most agreements are observed with student faculty, gender, year and dormitory attributes. However, in computing the correlation between attributes and community structure, Traud et al (2011) reported that the order of correlation strength is significantly dependent on the agreement index used and not consistent across different indices.

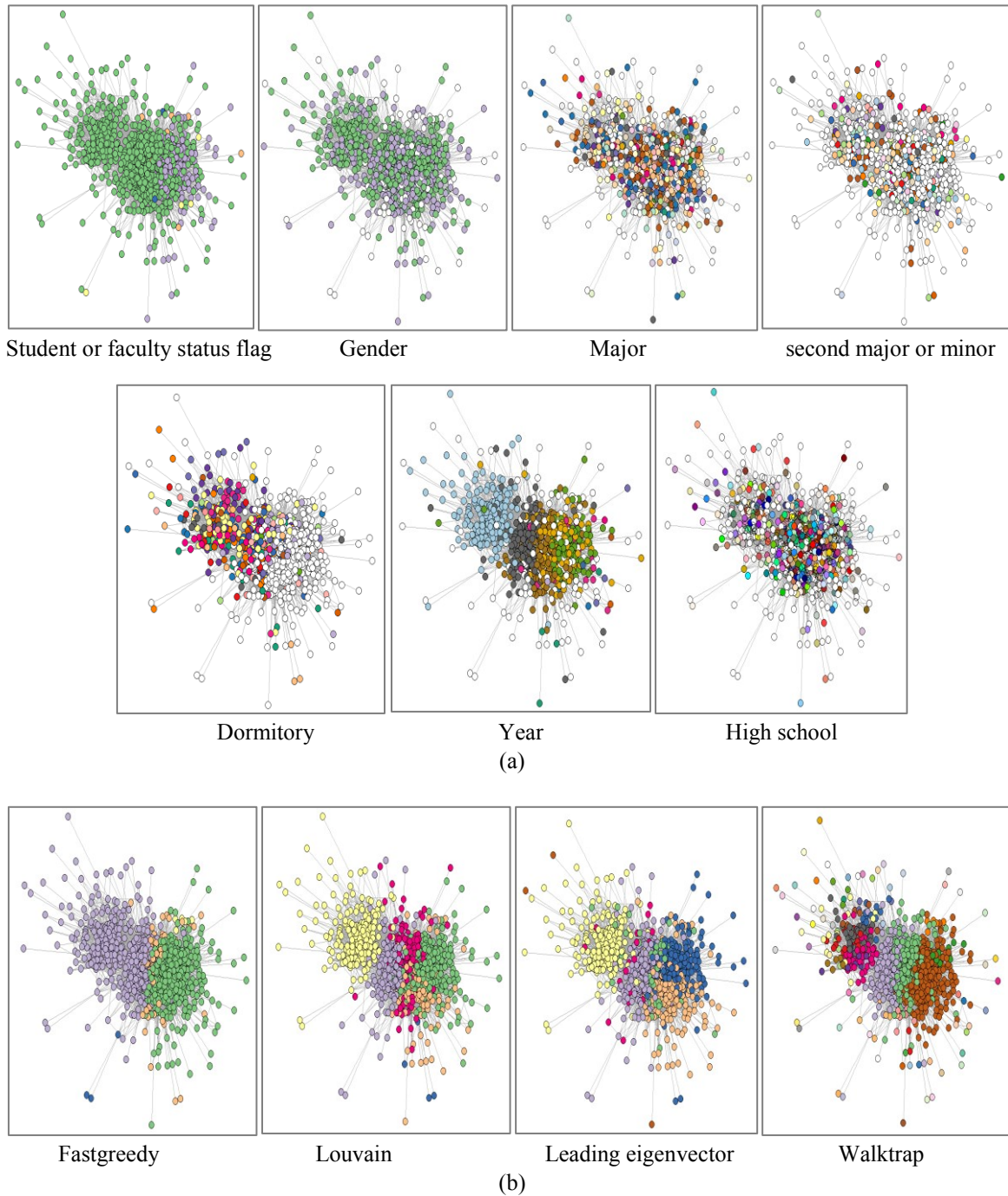


Figure 6.2 Visualization of correlations between attributes and communities for Reed dataset. (a) Communities based on attributes: nodes are coloured the same if they have the same value for the corresponding attribute; nodes with a missing value for an attribute are white. (b) Communities based on community clustering algorithm: nodes are coloured the same if they belong to the same community.

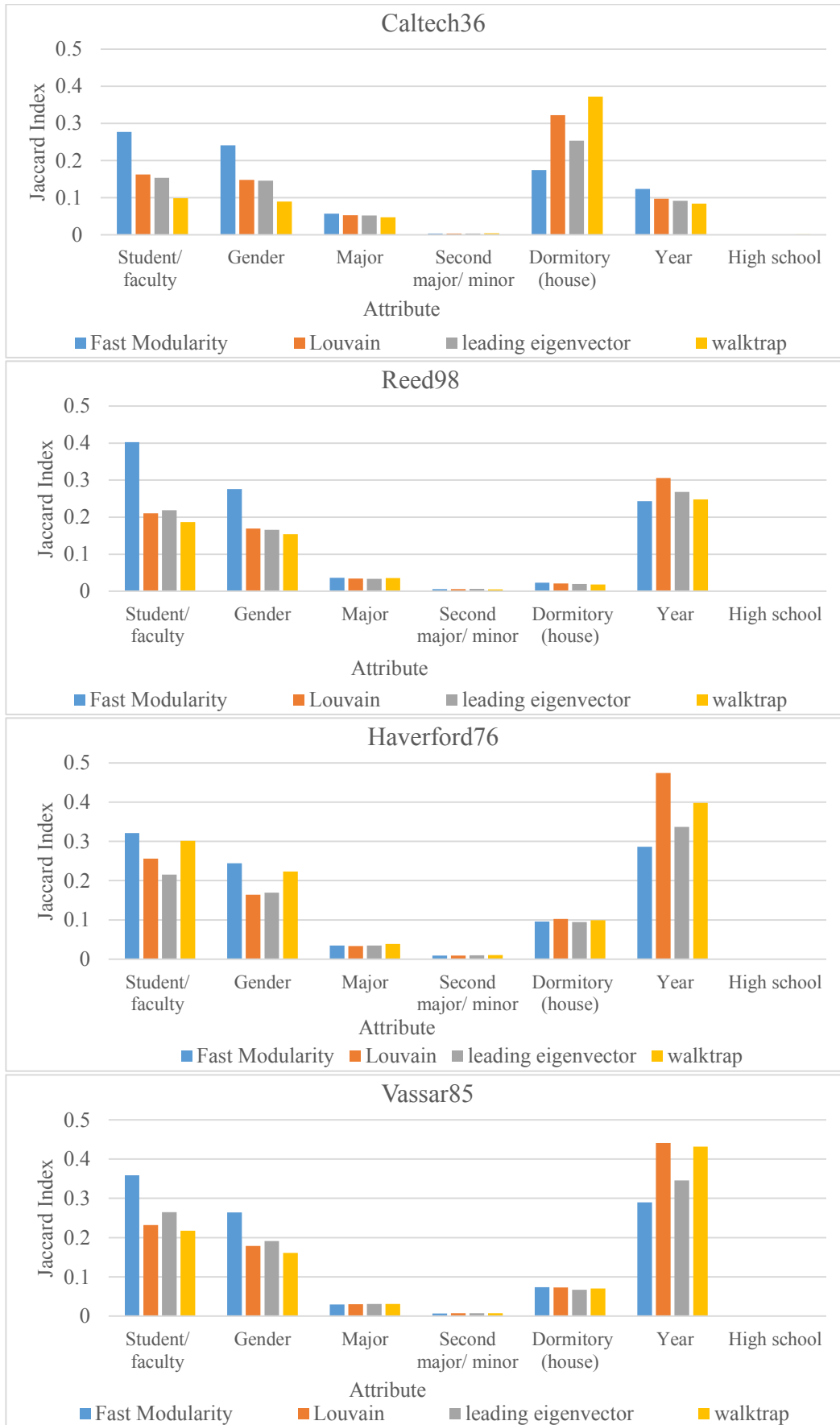


Figure 6.3 Agreement of different community detection algorithms with each attribute, for a subset of four datasets.

Observing a correlation between the attributes and the communities in the network, indicates the attribute information is a source of data that can be used to perform the community clustering task. Furthermore, based on the homophily property of a network as shown above it is clear that the linked nodes are more likely to share similar attributes. However, the attributes do not have the same influence as the community structure and some attributes weigh more than others in their influence. Thus the impact of different attributes on communities needs to be known and properly weighted according to their influence on the community structure. This will balance the role of network information and node attributes.

6.5 Description of the Proposed Approach

The proposed approach could be defined as a pre-processing phase for conventional community clustering algorithms, which takes a graph $G = (V, E, A)$, the weight of attributes (W) and two more weighting factors (α and β) as inputs. α is used to weight the contribution between connectivity information, and both attribute and shared neighbours' information. β is used to weight attribute information to the number of common neighbours. However, these weighting factors (W, α, β) can be either provided as part of the input if they are known a priori or calculated from the dataset.

The proposed approach returns a hybrid similarity matrix. The hybrid similarity matrix is a weighted combination of attribute information, shared neighbours' information and connectivity information between the nodes. Once the proposed approach constructs the hybrid similarity matrix, it can be supplied to any of the state-of-the-art clustering algorithms proposed for weighted graph (e.g. Newman fast Greedy algorithm, Louvain algorithm, Newman algorithm based on leading eigenvector of a modularity matrix or Walktrap algorithm) to extract community clusters.

The general architecture of proposed approach is shown in Figure 6.4. As can be seen in the figure, the approach has two phases named the parameter-learning phase and information aggregation phase. The first phase aims is to extract optimal parameters whereas the second one is used to build a hybrid similarity matrix.

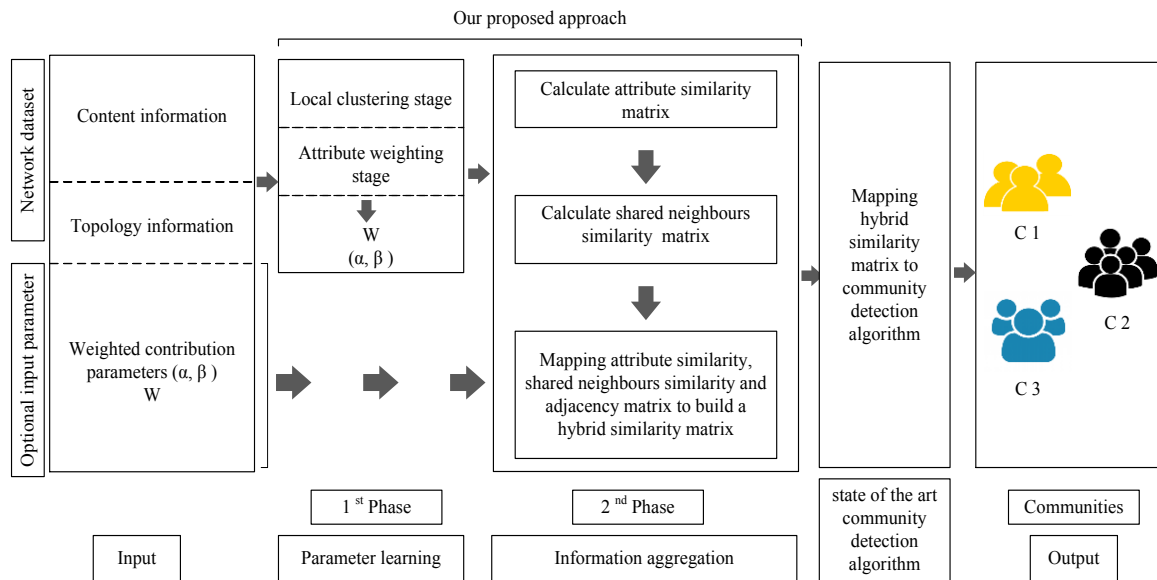


Figure 6.4 System architecture for the proposed approach.

We formally describe the generative process of hybrid similarity matrix as following:

$$H_{sim}(i, j) = \alpha \cdot A(i, j) + (1 - \alpha) [\beta \cdot Wa_{sim}(i, j) + (1 - \beta) \cdot SN_{sim}(i, j)] \quad (6.3)$$

$$Wa_{sim}(i, j) = W \cdot A_{sim}(i, j) \quad (6.4)$$

Where:

$H_{sim}(i, j)$: Hybrid similarity matrix

A: adjacency matrix

$Wa_{sim}(i, j)$: The weighted attribute similarity between a pair of nodes (i, j)

α : The weighting factor used for the contribution of connectivity information to the attribute information and shared neighbours information.

β : The weighting factor used for the contribution of attribute information to the number of common neighbours information.

$SN_{sim}(i, j)$: Shared neighbours similarity between nodes i and j .

$A_{sim}(i, j)$: The attribute similarity between a pair of nodes (i, j) in network $G = (V, E, A)$

W : A matrix containing the weights of each attribute of the node in the network.

Definition 6.1 Shared neighbours

Given a graph $G = (V, E)$, for a node $i \in V$, the neighbours of node i are nodes that directly connect to node i and is denoted by $\Gamma(i)$.

The shared neighbours of node i and j are the nodes that both directly connect to nodes i and j .

It is defined as:

$$SN(i, j) = \{\Gamma(i) \cap \Gamma(j)\} \quad (6.5)$$

The shared neighbours similarity between nodes i and j is calculated by dividing the number of shared neighbours between them by the maximum degree of i and j nodes. It is defined as:

$$SN_{sim}(i, j) = \frac{SN(i, j)}{\max[K_i, K_j]} \quad (6.6)$$

Where:

$SN(i, j)$: Shared neighbours between nodes i and j .

K_i : Degree of node i

In the hybrid similarity matrix, as it is defined in equation 6.3, the strength of relationship between nodes is determined by attribute information, connectivity information and shared neighbours and controlled by two weighting parameters (α and β). The α and β weighting parameters can be given as part of the input values by the human agent based on his knowledge

of the data structure and his perception of the importance of each attribute. However, choosing the right weighting values of attributes without a priori knowledge of the network is a challenging task. Furthermore, the proposed approach has attribute weighting factors (W), the values of which need to be set carefully. Thus, in the following sections, the two phases of the proposed approach (the parameter-learning phase and information aggregation phase) will be discussed in detail to provide guidelines on how to set these parameters.

6.5.1 The Parameter Learning Phase

Since the goal of utilizing details on attribute information, shared neighbours and connectivity information in this work, is to get the best community clusters for the network, the attributes of the nodes should be weighted in such a way that greater weight is given to the more influential attributes, and smaller weights for the less influential. Determining the influence and thus the weights of the attributes correctly, will enhance the community structure algorithm and improve the detection of communities in the networks. The main purpose of the proposed attribute weighting technique is to search for small groups of nodes (initial clusters) that contain more internal connections (links between nodes in the group) than external connections (between nodes of the group and nodes in other groups) and then find the attribute similarity between nodes in the same groups to get the influence factor for each attribute.

To accomplish this, the parameter-learning phase, as shown in Figure 6.4, is subdivided into two stages, local clustering stage and attribute weighting stage. Local clustering phase is to extract dense nodes from the network to form the initial clusters. These initial clusters are local small ones, far from being the optimal result and are only used in the second stage to weight the attributes of each node in the network as well as estimate the α and β parameter values.

In the local clustering phase, the initial clusters are obtained by applying the first phase of the DICCA approach proposed in chapter 4, named local clustering phase. The basic idea of the

local clustering phase in DICCA consists of picking up m nodes to be originators in which the m nodes should be spread out in all regions of the network and assigning each node to the closest originator to form a cluster.

The attribute weighting stage is then applied to find the strength of the weighting for each attribute based on the structures of current clustering results. During the attribute weighting stage, the set of attributes for each node are weighted according to its influence in the community in which the highly influential attributes are assigned with high strength weights; meanwhile the less influential attributes are assigned with low strength weights.

In more detail, to find the attribute weighting, it is necessary to measure the proximity between each pair of nodes in the initial clusters based on their attributes. To do so, the attribute similarity metric needs to be defined first.

6.5.1.1 Attribute Similarity Metric

The attribute similarity between nodes V_i and V_j within the same cluster is determined by examining each of d set of attributes on the two nodes and reflect on the strength of the relationship between them in terms of their attribute values.

Without loss of generality, regardless of the similarity metric considered to find the weight of attributes, first, the similarity between the attribute values of each pair of nodes belonging to the same local cluster is calculated as follows:

let $X_{N,d}^i$ be the similarity matrix for cluster i with N nodes each with d attributes, the local attribute weight for cluster i is obtained by adding the appropriate dimension attribute of each node in the cluster to form a vector of $1 \times d$ size and determined as:

$$LW_d^i = \frac{1}{N} \sum_{i=1}^d (X_{N,d}^i) \quad (6.7)$$

The weighting for the entire network is then calculated by adding the corresponding attribute of each local attribute weight (sum of the vectors) to form another vector in $1 \times d$ size. It is formally defined as:

$$W = \frac{1}{m} (\sum_{i=1}^m LW_d^i) \quad (6.8)$$

It is worth mentioning that the weights assigned to the attributes in the parameter learning phase $LW = \{LW_1, LW_2 \dots LW_m\}$ ranges between 0 and 1.

Whether or not a certain subset is optimal depends on the similarity metric employed. The question about what are the best similarity measures between nodes to choose for different types of attribute data is beyond the scope of this work. In this work, a Jaccard similarity coefficient is used to define the attribute similarity between nodes in the same cluster and to find the weight of attributes (W) during the parameter-learning phase. For an overview of the research work on determining the most meaningful similarity measures in various fields and for different types of data, see (Choi, Cha and Tappert, 2010; Arif and Basalamah, 2012).

Definition 6.2 Jaccard similarity. Given a network $G = (V, E, A)$, for any pair of nodes $V_i, V_j \in V$, the Jaccard similarity between nodes V_i and V_j with respect to attribute is indicated as $J(A_i, A_j)$ and is defined as the size of the intersection divided by the size union of the data sets, as given below:

$$J(A_i, A_j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|} \quad (6.9)$$

$J(A_i, A_j)$ returns a value between 0 and 1, with 0 denoting no similarity, and 1 denoting identical sets.

Furthermore, since in this work Jaccard similarity is used to measure attribute similarity between nodes, the $X_{N,d}^i$ could be defined as the Jaccard similarity matrix for cluster i and the weighted attribute similarity $Wa_{sim}(i, j)$, between any nodes i and j is defined as follows:

$$Wa_{sim}(i, j) = \frac{\sum_{L=1}^d (W_L * [Att_{i_L} \cap Att_{j_L}])}{\sum_{L=1}^d (W_L * [Att_{i_L} \cup Att_{j_L}])} \quad (6.10)$$

Where each node has d attributes and Att_i is the attribute vector of node i .

The pseudo code outlining the entire procedure with Jaccard similarity is listed in Algorithm 6.1.

Algorithm 6.1: The proposed approach

Input:

adj: adjacency matrix.

Att: An attribute nodes matrix.

Optional input parameter:

W: a matrix containing the weights of each attribute for each node in the network.

 α : The weighted Contribution of connectivity information to the attribute information
//and shared neighbours information. β : The weighted contribution of attribute information to the number of shared
//neighbour information.**Output:**

K: A set of communities in the network.

for each Node $i \in \text{adj}$ $A_{\text{sim}}(i, j) = \sum_{L=1}^d [\text{Att}_{i_L} \cap \text{Att}_{j_L}] / \sum_{L=1}^d [\text{Att}_{i_L} \cup \text{Att}_{j_L}]$ //get attribute
//similarity matrix between i & j where $i \neq j$ $\Gamma(i) \leftarrow$ get the neighbours of node (i) $k_i \leftarrow$ get the degree of node (i)**end** $\text{SN}(i, j) = \{ \Gamma(i) \cap \Gamma(j) \}$ //get the number of shared neighbours between each nodes $\text{SN}_{\text{sim}}(i, j) = \text{SN}(i, j) / \max[K_i, K_j]$ // shared neighbours similarity between nodes i
// and j where $i \neq j$ **C = local clustering phase** (adj) // run the first phase of DICCA algorithm**for** each cluster $lc \in C$ **For** each pairs of nodes $i, j \in lc$ $X_{N,d}^{lc} \leftarrow |\text{Att}_i \cap \text{Att}_j| / |\text{Att}_i \cup \text{Att}_j|$ // Jaccard similarity matrix for cluster lc **end** $N \leftarrow$ get number of nodes in lc $LW_d^{lc} = \frac{1}{N} \sum_{i=1}^d (X_{N,d}^i)$ **End** $m \leftarrow$ get number of initial clusters in c **if** (W not provided as an input parameter) $W = \frac{1}{m} (\sum_{i=1}^m LW_d^i)$ **end****if** (α not provided as an input parameter)) $\alpha = \text{avg}(W)$ **end****if** (β not provided as an input parameter)) $\beta = 0.5$ **end** $W_{\text{a}_{\text{sim}}}(i, j) = \sum_{L=1}^d (W_L * [\text{Att}_{i_L} \cap \text{Att}_{j_L}]) / \sum_{L=1}^d (W_L * [\text{Att}_{i_L} \cup \text{Att}_{j_L}])$ $H_{\text{sim}}(i, j) \leftarrow \alpha \cdot \text{Adj}(i, j) + (1 - \alpha) [\beta \cdot W_{\text{a}_{\text{sim}}}(i, j) + (1 - \beta) \cdot \text{SN}_{\text{sim}}(i, j)]$ K \leftarrow **community cluster** ($H_{\text{sim}}(i, j)$)**Return K** return the final division of adj.

6.5.1.2 Effect of α and β on the Quality of Community Structure

When considering the values to select for the two weighting factors (α and β), the type of emphasis on one of the network parameters needs to be considered. For example, emphasis on the connectivity information source means that the parameter α should be greater than 0.5. On the other hand, emphasis on attribute and shared neighbours information means that α should be less than 0.5. The same argument holds good for the parameter β , i.e., β greater than 0.5 indicates that attribute node information source has more contribution than the information related to the number of common neighbours. In the networks, the weighted combination of attribute information, shared neighbours and connectivity information are not the same and the values of α and β need to be selected carefully. However, in practice without any prior domain knowledge, it is quite difficult to scale the contribution of each source of information.

In order to determine the effects of varying α and β parameters on the quality of community clustering and thereby to determine the parameters' selection range, four different datasets are used to track how the community clustering changes when the values of α and β are varied from 0.1 to 1 with a step size of 0.1. Also, modularity index is used to evaluate the quality of community detection.

Figure 6.5 and 6.6 show how the two parameters influence the community clustering quality. The X-axis and Y-axis in the figures represent the values of α and β respectively, while the Z-axis represents the modularity score. As can be clearly seen from Figure 6.5 (a-d), the modularity is remarkably robust to the choice of parameter values. When $\alpha = \beta = 0$, the modularity of community detection is ≥ 0.25 for most of the algorithms for all the datasets. However, it is worth mentioning that $\alpha = \beta = 0$ indicates that the information used to find the community clustering is just based on the number of common neighbours $H_{sim}(i, j) = SN_{sim}(i, j)$.

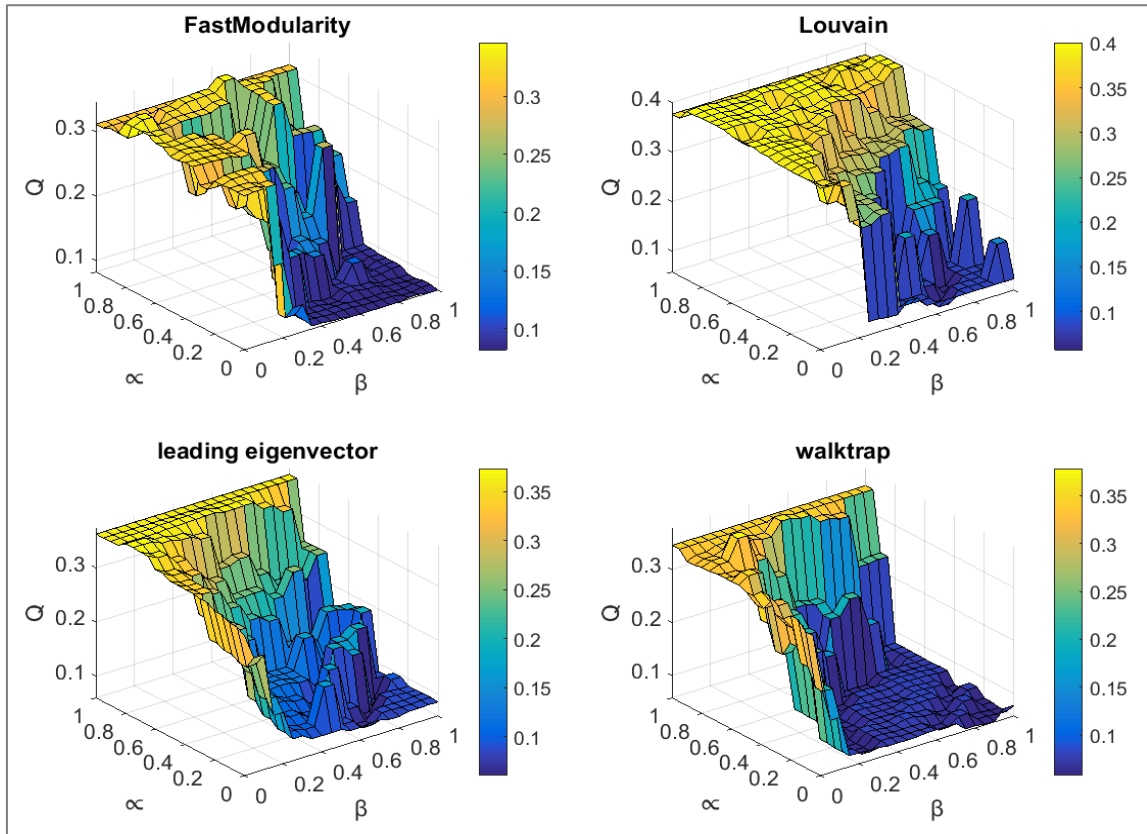
As an overview, with an increasing value of β , the quality of community clustering decreases for a constant value of α . On the contrary, with an increasing value of α , the quality of community clustering increases slightly for a constant β value. It is also noticed that, for values of $\alpha < 0.6$ the modularity is dramatically affected by varying the value of β . The modularity fluctuates between 0.01 and 0.4 and it becomes relatively stable when α value ranges between 0.6 and 0.7. However, the Modularity becomes almost stable for the vast majority of β values when $\alpha > 0.7$.

Experimental results also demonstrate that the connectivity information is more useful than the shared neighbours' information and attribute information. Therefore the value selected for α should be greater than or equal to 0.5. For the datasets considered in this work, high modularity values are obtained when $\alpha > 0.7$.

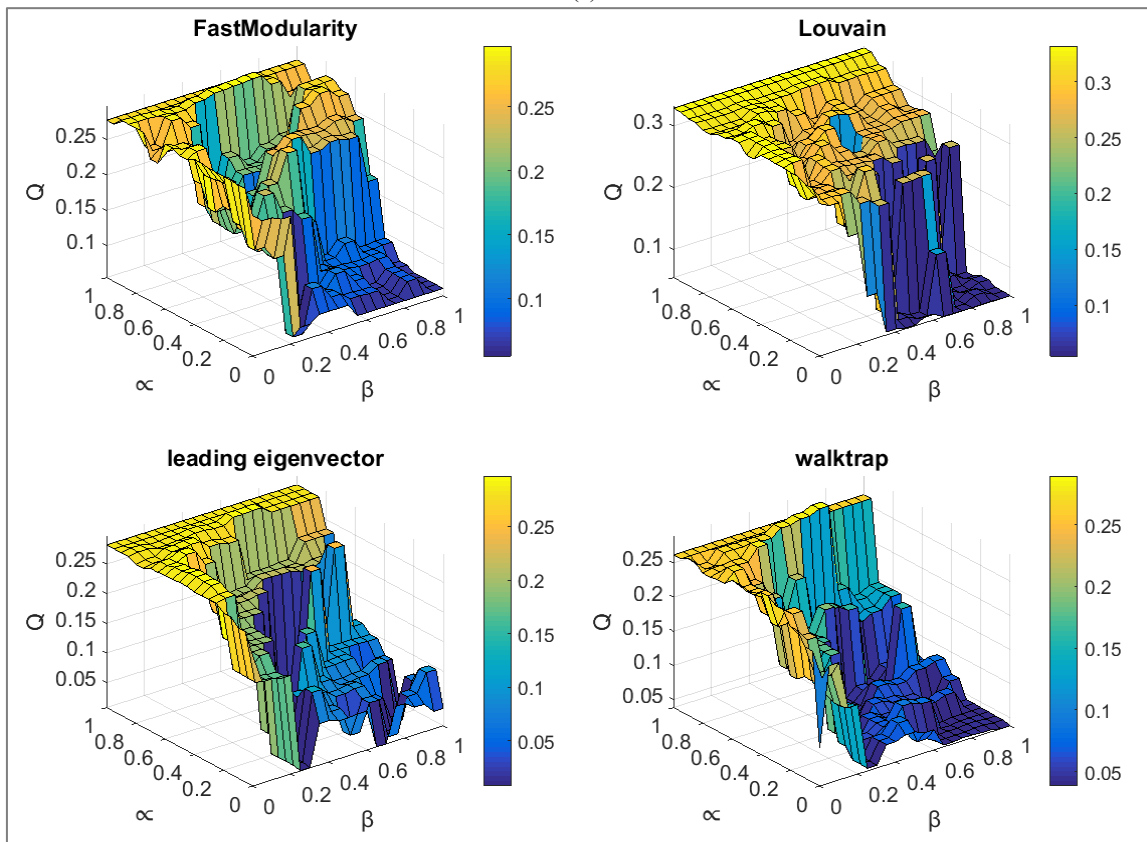
With regard to these two parameters α and β , there is no straightforward way to fit them to datasets and different datasets may require different parameter values. However, based on the above argument, in order to better exploit the sources of information and obtain optimum robustness in the detection of community clusters in the presence of noise, the value of α is set based on the weights of attributes (w) as follows:

$$\alpha = avg(w) \tag{6.11}$$

In this work, to avoid a cumbersome decision process, equal importance is given to shared neighbours and attribute information in which $\beta=0.5$ is set in all the following performed experimentations.

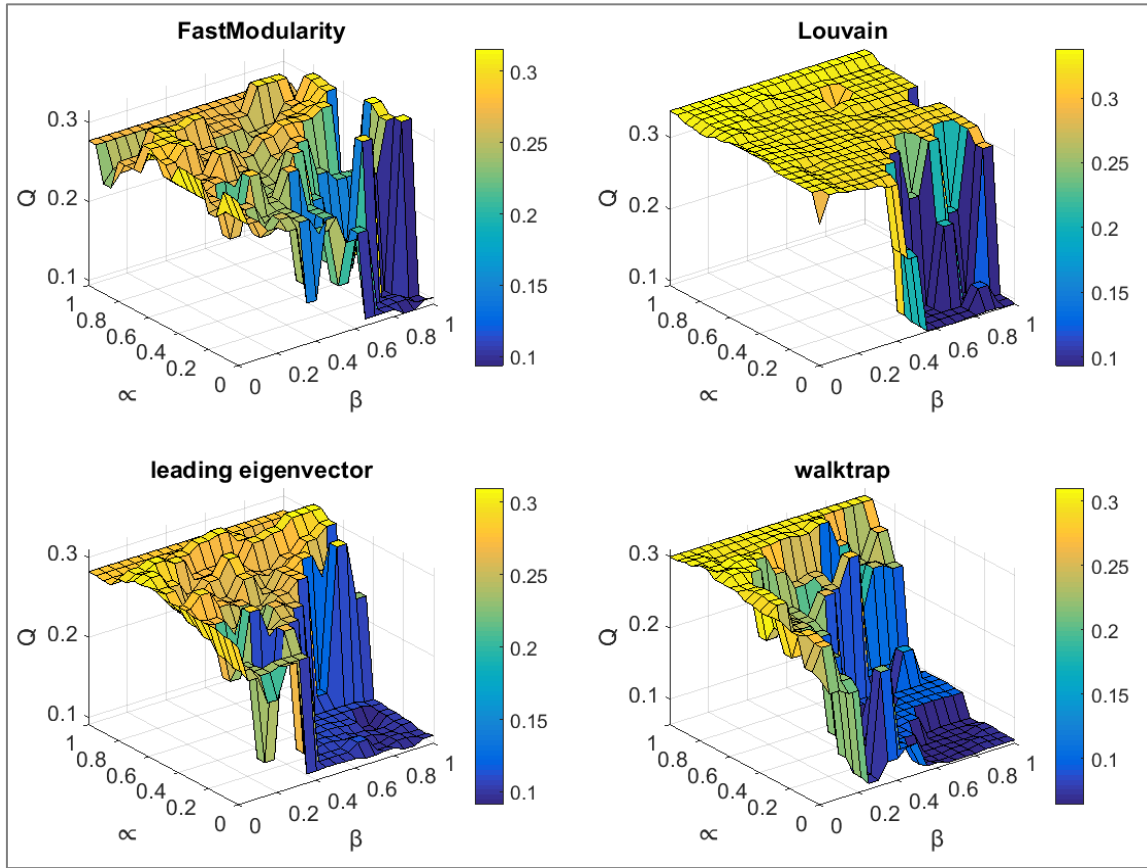


(a)

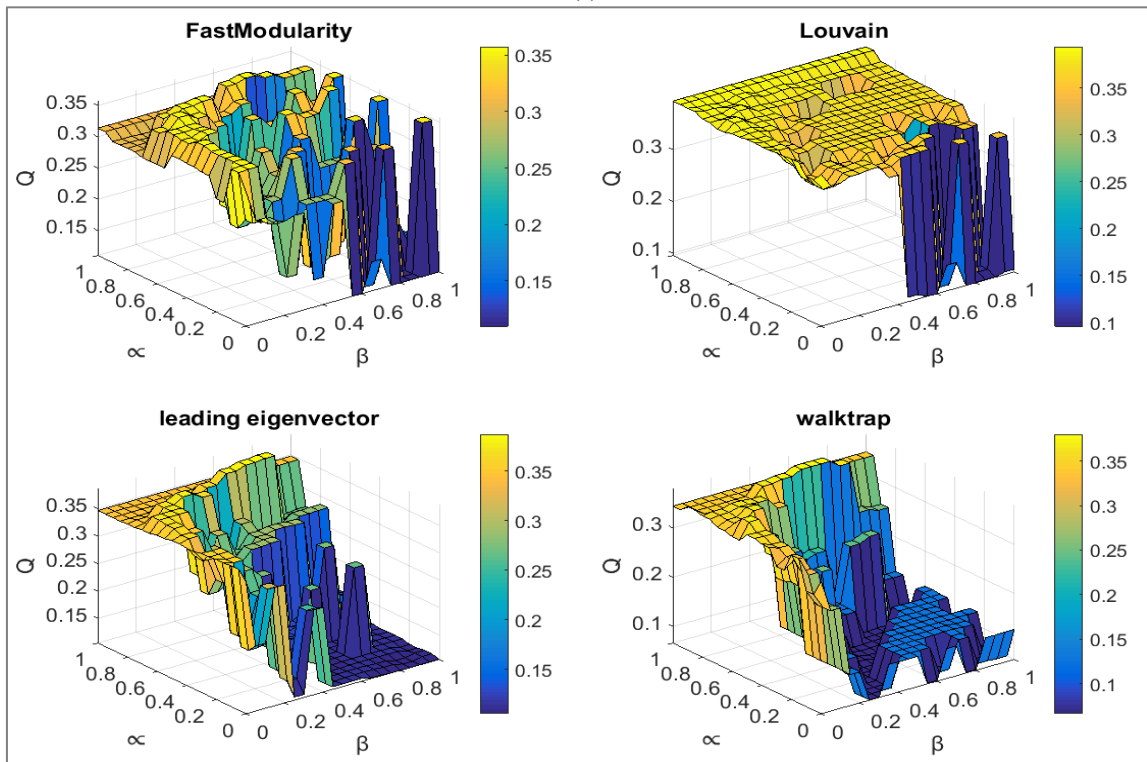


(b)

Figure 6.5 (a-b) Modularity value achieved by four community clustering algorithm dataset using different value of α and β on: (a) Caltech36 (b) Reed98 dataset.



(c)



(d)

Figure 6.6 (c-d) Modularity value achieved by four community clustering algorithm dataset using different value of α and β on: (c) Harvard76 (d) Vassar85 dataset.

6.5.2 Information Aggregation Phase

The information aggregation phase aims to build a weighted matrix, named hybrid matrix, based on the knowledge learned from the parameter learning phase. These weighted attributes w , α and β values are used to build a hybrid similarity matrix as defined in equation 6.3. In the hybrid matrix, the edges that link nodes do not have similar attributes or do not have shared neighbours, will be punished and assigned with low strength weights; while the edges connecting similar nodes or having shared neighbours will be assigned with high strength weights. Also, there are some edges which will be added between the nodes to represent the attribute and shared neighbour similarity.

6.6 Experimentation and Results

6.6.1 Experimental Setup

In order to assess the effectiveness of the proposed approach to detect communities under an unreliable network structure, an experimentation has been conducted using four different Facebook dataset networks when some edges are missing while the node attributes are fully available. Furthermore, for the sake of evaluation, edges are removed from the network at random and the number of removed links is increased from zero to half the number of edges in the network in steps of 5% of network edges.

In each experiment, the performance is computed using the results obtained by applying each of the four algorithms with and without applying the proposed approach as a pre-processing step. Each algorithm has been applied more than once on the data and the experimental results presented are the average of ten simulation runs.

To quantify the performance of the proposed approach, the quality of the obtained community structures is evaluated based on the modularity, number and size of detected communities.

Moreover, for simplification, in the following sections when the proposed approach is combined with Fast Modularity algorithm (FA) is referred to as Hybrid-FA; when combined with Louvain algorithm (LA) as Hybrid-LA; when combined with leading eigenvector (LE) as Hybrid-LE and Hybrid-WA when combined with Walktrap algorithm (WA). Additionally, to facilitate comparison of results in line charts, the results obtained using the proposed approach are denoted by dashed line style with “x” marker points.

6.6.2 Experimental Results and Discussion

In this subsection, the effectiveness and efficiency of the algorithm is assessed from two aspects. One is to evaluate the attribute weighted method proposed in this work along with the methodology used to set the parameter value. The other aspect is to integrate the proposed approach with well-known community clustering algorithms and make a comparison of the results achieved without the integration to show how the proposed approach can be used to improve the robustness and quality of well-known community clustering algorithms.

6.6.2.1 Evaluation of Attribute Weighting Method

As highlighted in section 6.4, different attributes have different significance for assessing the similarity between the nodes in the same community clusters, therefore the attribute weighting method is proposed. In this section, the performance of the proposed attribute weighting method is experimentally evaluated.

The evaluation is done by checking how well the weight of the attributes obtained by the weighting method match with the actual important attributes presented in Figure 6.3.

Figure 6.7 shows the attribute weights obtained by the weighting method for the four datasets under consideration. It is obvious that the attributes have different weight strengths and order of importance for different datasets. However, looking at the attribute weights of the four data sets, it is clear that four specific attributes (student, gender, dormitory and year attribute) have

the highest weighting values across all four data sets. Anyway, the remaining attributes (high school and major/minor attribute) do not have strong influence on the community structure, hence weighted with a very small value, if not dropped, in the attribute weighting stage.

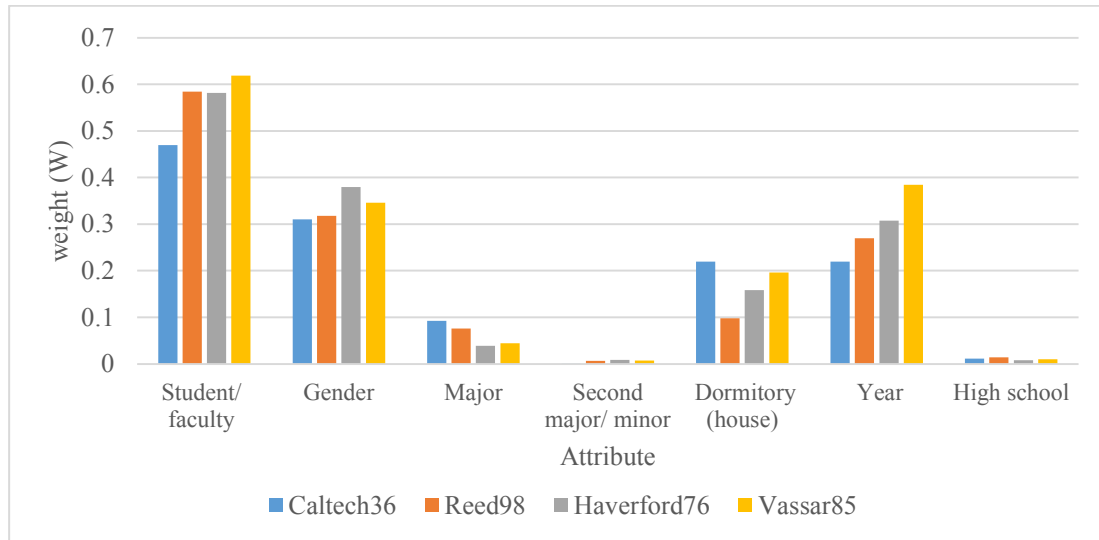


Figure 6.7 Attribute weights for four datasets.

Moreover, the comparison between Figure 6.3 and Figure 6.7 shows that the parameter learning phase achieves almost the same results in most cases. Whereas, the attribute importance order is either the same or only slightly different due to small differences in the attribute correlation. For example in Caltech36 dataset, the order of importance attributes are student, gender, year and house with attribute weight values 0.4695, 0.3102, 0.2195 and 0.2193 respectively. In comparison to Figure 6.3 and for the case of the fast modularity algorithm as an example, the order is changed to student, gender, house and year attribute, achieving Jaccard index values of 0.2772, 0.2412, 0.1746 and 0.1239 respectively.

Furthermore, to evaluate the performance of the proposed weighting method in handling noisy data, Figure 6.8 shows the values of attribute weight for the four largest weighted attributes obtained by the weighting method when the percentage of removed edges varied from 0 to 50%. From the figure, it is worth noting that the ordering of weights is remarkably stable and the

attribute weighting method shows an effective performance by getting rid of the noisy datasets and correctly weights attributes according to their importance.

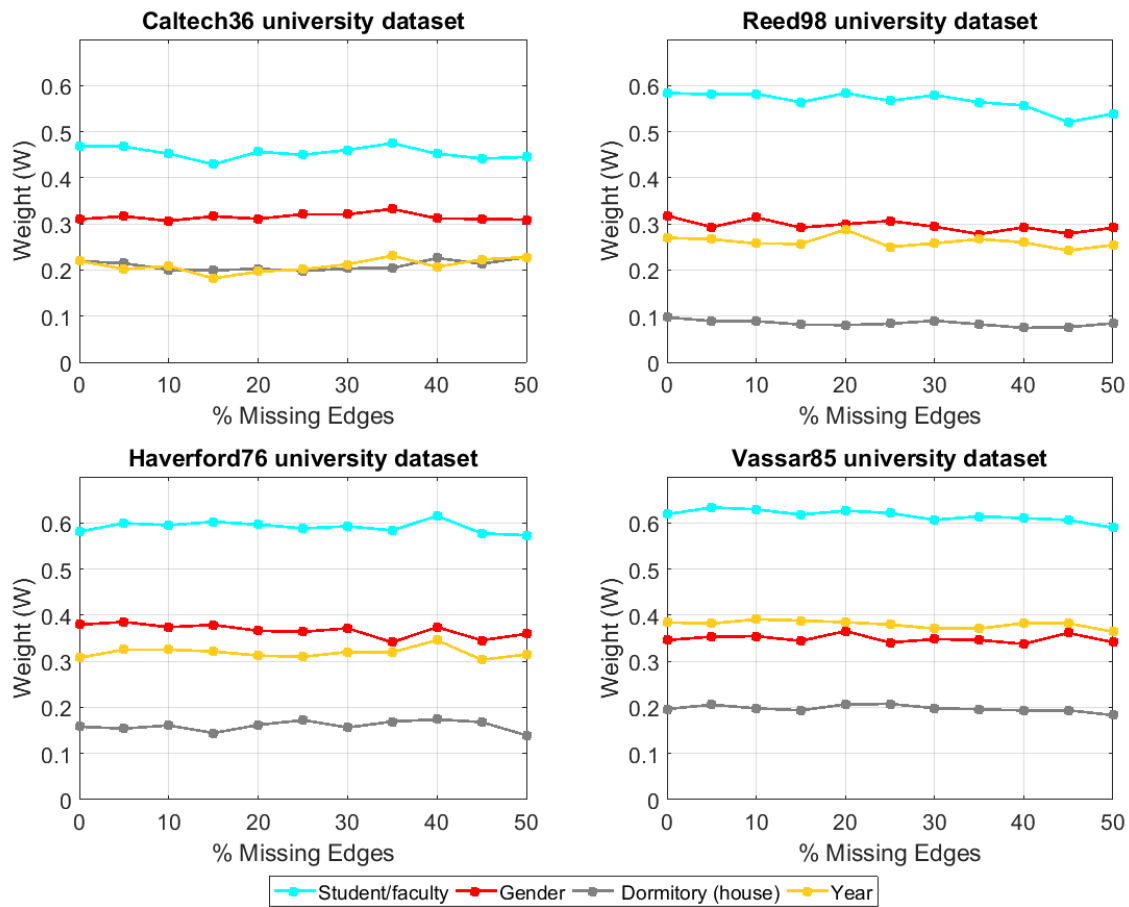


Figure 6.8 Robustness of weighting method to the edge removal.

To further assess the parameters analysis phase, the number of initial clusters identified at local clustering stage along with the value of α via percent of removed edges for four datasets are reported in Table 6.1.

The results in the Table 6.1 indicate that the noise has no significant influence on the value of α . In other words, the method used to define α value (see equation 6.11) is somewhat stable. In addition, it is clear that local clustering tends to partition data to a larger number of initial clusters. Considering Reed98 dataset for example, when the missing edges varied from 0% to 50%, the values of α and the number of obtained initial clusters were $\{0.8084, 382\}$ and $\{0.8231, 446\}$ respectively.

It is also worth noting from Table 6.1 that the value of α is not related to the number of initial clusters found by the local clustering stage. In some cases, higher value of α is obtained when more initial clusters are found. For others however, the value of α increases when fewer initial clusters are found. Considering Reed98 dataset, for instance, when the missing edges increased from 15% to 20%, both α value and the number of initial clusters increased from {0.8139, 399} to {0.8162, 405} respectively. On the other hand and for the same dataset, when the missing edges increased from 5% to 10% the value of α increased from 0.8123 to 0.8130 meanwhile the number of initial clusters decreased by 3. However, the value of α for the four considered datasets is always higher than 0.75. This value is in agreement with what was observed in section 6.5.1.2, where the connectivity information contains more useful information than the shared neighbours or attribute information ($\alpha \geq 0.5$) and to get high modularity the value of α should be higher than 0.7.

Overall, the results clearly demonstrate that the parameter learning method has the ability to extract essential and informative attributes and to weight them to reflect the relative importance of attribute in community clustering tasks.

Table 6.1 Results for four dataset

Dataset	Caltech36		Reed98		Haverford76		Vassar85	
	Number of initial clusters	α	Number of initial clusters	α	Number of initial clusters	α	Number of initial clusters	α
0	384	0.8127	382	0.8084	412	0.7792	824	0.7673
5	381	0.8156	392	0.8123	427	0.7811	835	0.7671
10	392	0.8177	389	0.8130	436	0.7822	844	0.7684
15	388	0.8161	399	0.8139	419	0.7823	873	0.7694
20	392	0.8161	405	0.8162	443	0.7827	898	0.7709
25	391	0.8159	397	0.8153	463	0.7827	921	0.7712
30	390	0.8156	409	0.8170	467	0.7843	927	0.7722
35	394	0.8168	402	0.8180	476	0.7834	948	0.7731
40	398	0.8152	418	0.8193	489	0.7861	953	0.7738
45	390	0.8171	432	0.8241	487	0.7879	1003	0.7763
50	387	0.8110	446	0.8231	514	0.7884	1036	0.7784

6.6.2.2 Model Performance

In this subsection, using the optimal parameters determined using the parameter-learning phase (as discussed in section 6.5.1), the performance of the pre-processing approach is evaluated.

6.6.2.2.1 Number of Community Clusters

Since the number of communities in the networks is unspecified, the algorithms try to automatically detect the most appropriate number of communities by maximizing the modularity.

The variation in number of community clusters when different numbers of edges are removed is given in Figure 6.9. It is observed that the conventional algorithms are adversely affected by noise so fail to account for appropriate community structures. Moreover, most cases result in an increasing number of communities with an increasing % of missing edges. The only exception is the LEA algorithm, which results in almost the same number of communities even without applying the pre-processing approach.

Considering Caltech36 dataset, for example, increasing proportions of edges are randomly removed from the network (from 0% to 50%), the number of communities detected by all conventional algorithms are changed from $\{10,10,12,72\}$ to $\{39,39,10,104\}$ for $\{FA, LA, LEA, WA\}$ algorithms respectively. Such behaviour can be explained by the fact that the conventional algorithms consider only topology information. On the other hand, the proposed approach considers attribute, shared neighbours and connectivity information. Since the nodes in the same community usually are not just highly connected but also have similar attributes and transitivity coefficient, the proposed approach uses attribute information to make up for the missing link information and to identify the community membership. Consequently, integrating the proposed approach with a conventional algorithm is more advantageous for

discovering the most appropriate number of community structures than using the conventional algorithm on its own.

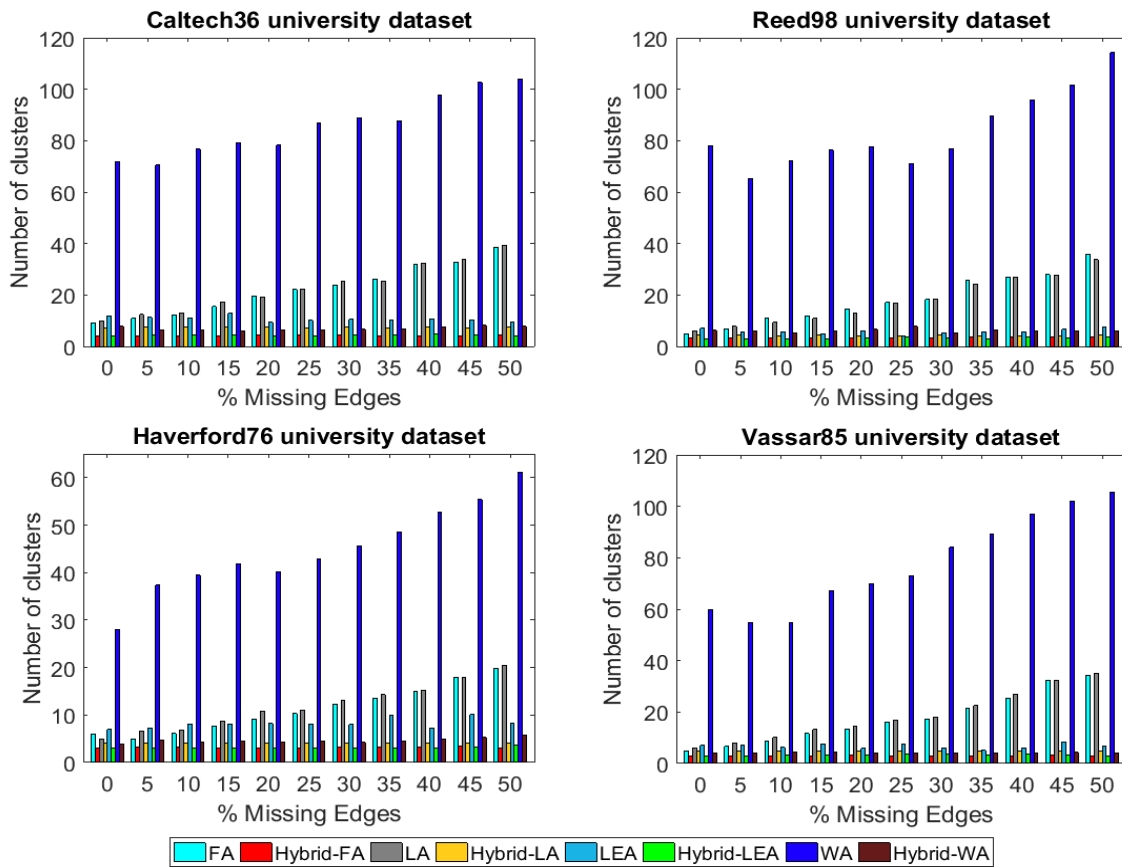


Figure 6.9 Number of community clusters for: (a) Caltech36 university dataset, (b) Reed98 university dataset (c) Haverford76 university dataset, (d) Vassar85 dataset.

Walktrap algorithm when run on the dataset on its own failed to detect the appropriate number of communities, and compared to the other algorithms the number of communities returned by Walktrap are extremely high for all considered datasets. However, applying the proposed approach as a pre-processing step to build the hybrid similarity matrix before applying the Walktrap community detection algorithm has significantly improved the performance to obtain just 8 clusters.

Furthermore, when the percentage of removed edges is increased from 0% to 50%, the number of clusters formed using the proposed approach is more similar to the original partition network when there is no noise applied. For example in the case of Caltech36 dataset when 50% of

edges are missing, the number of obtained communities are $\{8,8,4\}$ for $\{\text{Hybrid-FA}, \text{Hybrid-LA}, \text{Hybrid-LEA}, \text{Hybrid-WA}\}$ algorithms respectively. This demonstrates that the proposed approach has the capability to extract relevant information from highly noisy datasets and make these algorithms quite robust to edge removal. The complete tables showing the cluster performance for four datasets are included in appendix A.3.

To take a closer look at the sensitivity of obtained communities to the noise, the average size of the obtained communities, when percentage of removed edges is increased from 0% to 50%, is investigated and shown in Figure 6.10.

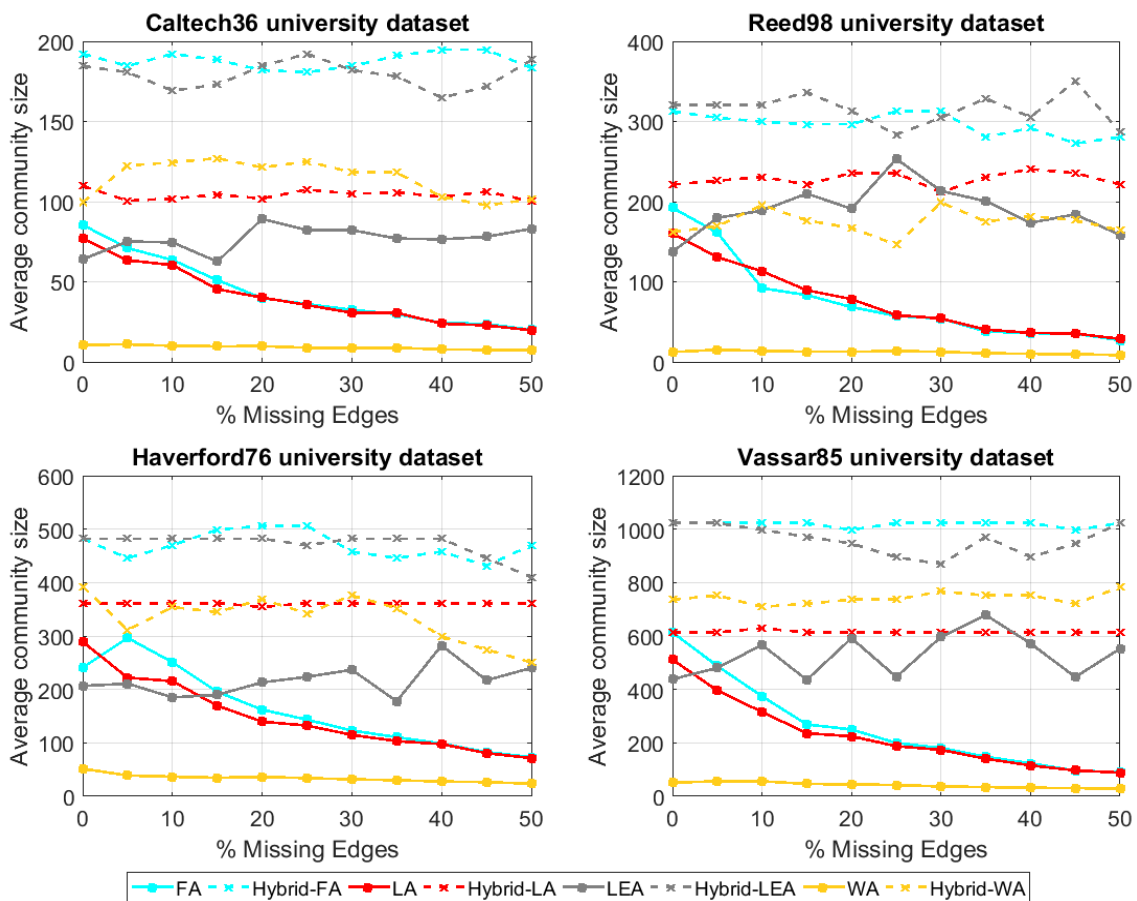


Figure 6.10 Average Community size for: (a) Caltech36 university dataset, (b) Reed98 university dataset (c) Haverford76 university dataset, (d) Vassar85 dataset.

Considering Vasser85 dataset, for example, increasing proportions of edges are randomly removed from network (from 0% to 50%), the average community size detected by all conventional algorithms dropped from $\{614, 511, 438, 51\}$ to $\{94, 95, 583, 28\}$ for $\{\text{FA},$

LA,LEA, WA} algorithms respectively. In contrast, combining the proposed pre-processing approach with the community-clustering algorithms considered in this work results in community clusters with almost constant average size. This effect comes from the fact that since the conventional community identification is based only on the adjacency matrix, the number of community clusters obtained are heavily dependent on the number of links in the network, so as the percentage of missing edges increases, the clustering algorithm becomes less stable and the clusters become smaller. In contrast, this is not the case for the hybrid similarity matrix, which is based on different considerations (attribute information, shared neighbours information and connectivity between nodes in the network).

6.6.2.2 Modularity

Regarding the quality of community clusters, the modularity metric is used as a scoring function to assess the quality of detected community clusters with and without applying the proposed pre-processing phase. Figure 6.11 shows the averaged Q values, plotted for each community detection algorithm. As shown in this figure, in most cases using the proposed pre-processing approach has resulted in a slightly lower modularity than the conventional community detection methods. However, the difference is negligible and the results suggest that the proposed approach is a promising and powerful tool to assist in the fine tuning of different sources of information in community clustering area.

Moreover, the comparison between Figure 6.9, Figure 6.10 and Figure 6.11 shows that while the approach achieves a good modularity quality that is comparable with the conventional methods, the approach is significantly more effective in terms of both number and size of communities detected where the network structure is found to have some unreliable or missing information.

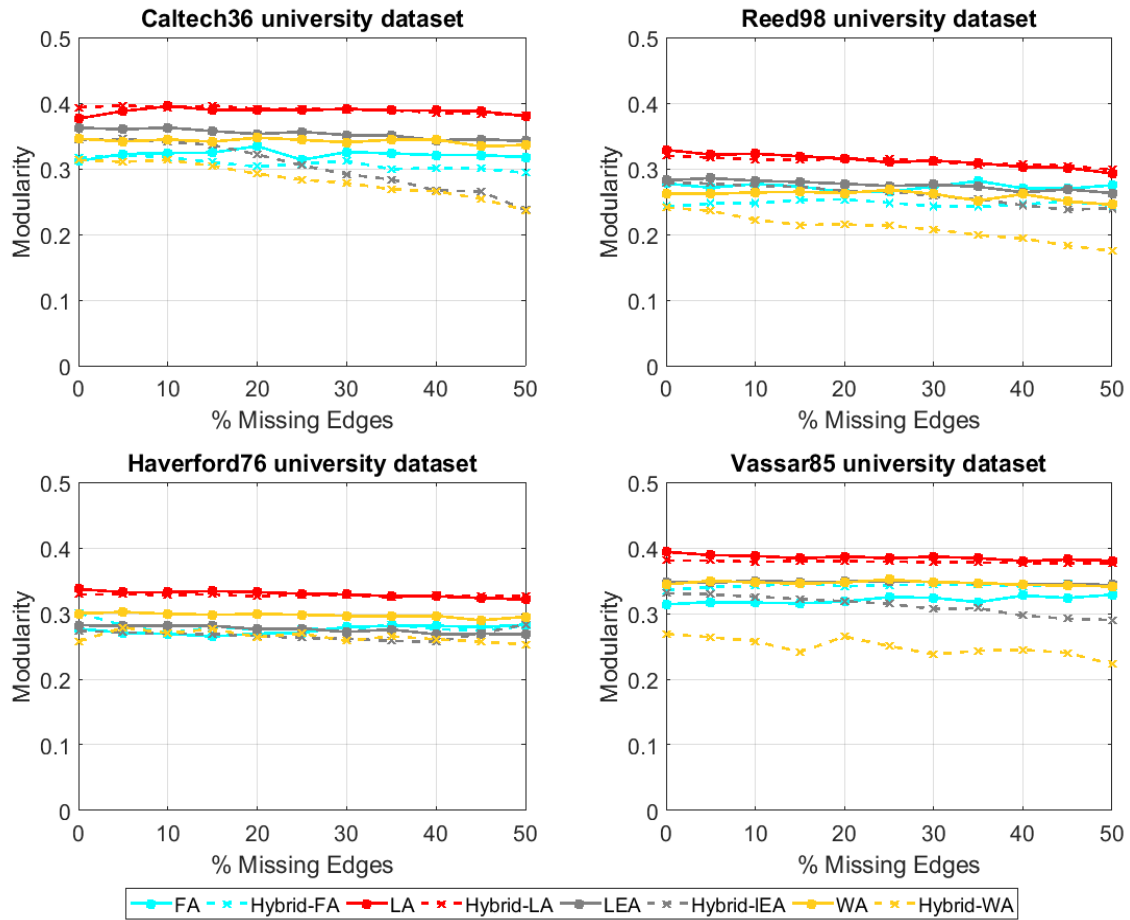


Figure 6.11 Modularity index vis missing edges for: (a) Caltech36 university dataset, (b) Reed98 university dataset (c) Haverford76 university dataset, (d) Vassar85 dataset.

It is worth noting that in the present context, using community clusters matching (e.g. NMI) to evaluate the quality of proposed approach might be particularly problematic, as the ground truth structures of four considered networks are not provided and both numbers and sizes of the obtained community clusters are not the same across the different community clustering algorithms. The exact values of results presented in this chapter are included as tables in appendix A.3.

6.7 Summary

In this chapter, a pre-processing approach that makes use of attribute information, shared neighbours and connectivity information aspects of the network to build a hybrid similarity matrix is proposed. Because the attributes in a network usually do not play equally important roles in clustering tasks, the proposed approach assigns a weighting value to each attribute

during the process of building Hybrid similarity matrix to reflect the relative importance of each attribute.

Besides the attribute weighting parameter, the approach required the specification of two more parameters α and β , these control the degree of contribution of connectivity information, attribute similarity and shared neighbours information for a good balance between them. The sensitivity of the pre-processing approach to α and β parameters is analysed. Also, a simple but effective model for determining attribute weighting value, α and β values of the approach to achieve an optimal result is provided.

In this work, a Jaccard similarity coefficient is used to denote attribute similarity between nodes. The proposed approach is tested in conjunction with four state-of-the-art algorithms (Fast Modularity algorithm, Louvain, leading eigenvector and Walktrap algorithm) popular in the literature by applying to four real-life Facebook data networks. The experimental results clearly demonstrate that the approach has the ability to incorporate attribute, structure and shared neighbours' information into meaningful information used to build a hybrid similarity matrix. Besides, the community clustering algorithms employed on the hybrid similarity matrix pre-processed by the proposed approach have shown a better effectiveness and robustness over noisy networks than the state-of-the-art algorithms without applying the pre-processing approach.

The approach proposed here could be used as well in conjunction with other community clustering algorithms and with other data sets.

CHAPTER 7

A CASE STUDY IN TELECOMMUNICATION

INDUSTRY OF SMARTPHONE USAGE.

In this chapter, a set of real-life android smartphone usage data has been skimmed and the different features of real-life Android smartphone usage are presented. With these results, community clustering and data mining techniques will be carried out as future work in order to develop a more profound understanding of the telecom network usage and users' characteristics. This chapter is published in the proceedings of the 17th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM.

7.1 Introduction

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

The material originally presented here (Chapter 7) cannot be made freely available via LJMU E-Theses Collection because of copyright. The material was published at 7th International Conference on Computer Systems and Technologies 2016, Palermo, Italy (pp. 81-88), ACM- available at: <https://dl.acm.org/citation.cfm?id=2983496>.

CHAPTER 8

CONCLUSION AND FUTURE WORK

This chapter concludes the research activities within this thesis. The first section summarises the original contribution and the main findings of the thesis. In the second section, the limitations of the work are discussed, and a number of further research directions that have been opened up by this thesis are presented.

8.1 Summary of Contributions

Many systems in the world can be represented as models of complex networks which are structures consisting of nodes or vertices connected by links or edges. Detecting and characterizing such community structures is one of the fundamental topics in network systems' analysis and it has many important applications in different branches of science including computer science, physics, mathematics and biology ranging from visualization, exploratory and data mining to building prediction models.

In this thesis, the major focus is given to the community analysis in networks which has been one of the active research topics for quite some time. However, based on a substantial background and literature review presented in chapter 2 and the properties of real-world networks presented in chapter 3, I argue that current community clustering techniques are no longer able to deal with the large real-world networks as the network size has increased beyond the capabilities of a single machine.

Hence, the focus in chapter 4 and 5 has been given to design the community clustering approaches to be able to handle massive datasets by efficiently utilizing the computing resources in a parallel processing topology. Following this, I propose an approach that uses both structural and attribute information to extract communities. Finally, I have studied the

real-world community structure of a large telecom dataset network. In the following, I summarize the contributions for each technical chapter (chapter 4, 5, 6 and 7) separately.

Chapter 4:

In chapter 4, a novel Decentralized Iterative Community Clustering Approach (DICCA) to extract an efficient community structure for large networks is presented. An important property of this approach is its ability to cluster the entire network without the global knowledge of the network topology. This ability means that the entire network does not need to be loaded into a single memory, and DICCA could be easily adapted to run in parallel on as many processors as available to find community clusters in big networks. This cannot be done in the majority of existing community detection algorithms that implicitly assume that the entire structure of the network is known and is available.

The DICCA approach is based on the random walk procedure and reachability of nodes in the network. The approach is run in an iterative fashion and uses two parameters, named threshold value and time to live (TTL). The question about what value of TTL to choose is discussed in this chapter along with the mathematical model to obtain optimal threshold value. Furthermore, the obtained results support the conclusion that the community clusters found by DICCA are meaningful and very close to the ground truth solution.

Chapter 5:

In chapter 5, a parallel decentralized iterative community clustering approach (PDICCA), which does not require any global knowledge of the graph topology is proposed. PDICCA is a distributed memory parallel processing approach that transforms the serial steps of the DICCA approach into parallelised tasks. It is scalable and will work with a range of computer architecture platforms (e.g. cluster of PCs, multi-core distributed memory servers, GPUs). The core idea of PDICCA is to split the data into blocks and cluster each block in a separate worker.

Then, the clusters extracted from blocks are aggregated at the final stage using re-clustering phase. PDICCA provides several features simultaneously; the PDICCA does not need to store the whole dataset in the one main memory so it is suitable for systems with limited memory and works well for massive datasets. Furthermore, PDICCA optimally utilizes the hardware capabilities of the parallel processors and minimizes the communication between workers during processing to reduce the bandwidth, memory and storage cost. Experimental results on a 4-core computer demonstrate that the proposed approach is quite effective, provides a consistent performance over time and has a great scaling characteristic without any noticeable loss in the performance.

Chapter 6:

Another problem in practical applications is that the network is usually noisy and imperfect with missing and false edges. These imperfections are often difficult to eliminate and highly affect the quality and accuracy of conventional methods that are used to find the community structure in the network. In this work, the pre-processing approach proposed in chapter 6 has the ability to incorporate attribute information, shared neighbours and connectivity information aspects of the network to build a hybrid similarity matrix. The matrix is built by assigning weights to the edges according to the strength of the connectivity, attribute similarity and number of shared neighbours. To accurately model, the proposed approach uses two weighting factors to identify the optimum trade-off between the information sources through a weighted matrix.

Extensive experiments with real Facebook data sets show that the results obtained by using the proposed approach in conjunction with the state-of-the-art community clustering algorithms have been demonstrated to be greatly improved. More specifically, while the approach achieves a good modularity quality that is comparable with the conventional methods, the approach is

significantly more effective in terms of both number and size of the communities detected where the network structure is found to have some unreliable or missing information.

Chapter 7:

Using a real-life android smartphone usage dataset, the different features of mobile phone usage is analysed in chapter 7. Furthermore, my plan was to apply the proposed community detection approaches to the smartphone usage dataset so that I can identify a community of users that often communicate with each other based on communication information between users along with other information present in the dataset. The community clustering might reveal interesting information about users, which then could be used by mobile server providers to design suitable marketing strategies for each group and thereby enhance business profitability. However, the fact that different phones pick a different hash for the same phone number, made it hard to detect the user communities. Thus, a data skimming technique is used to extract abstract information and trends from the given big dataset.

8.2 Recommendations and Future Works

Many lines of research remain open for future works, such as:

First, although the DICCA and PDICCA approaches for detecting community clusters in large networks (in chapter 4 and 5 respectively) have been extensively investigated and studied, there are still some issues that need further investigation. In particular, I intend to extend the studies and analysis on three specific points:

- Real-world networks often do not contain perfect communities where each node does not have only one possible clustering and nodes can belong to multiple communities at once. Identifying such overlapping communities (also known as fuzzy) is crucial for understanding the structure as well as the function of real-world networks. A further direction is to extend the DICCA approach to be able to detect such fuzzy communities.

- In this work, only the undirected networks have been taken into consideration. Therefore, I consider the directed network case as an interesting direction for further research.
- In PDICCA, in order to cluster networks in parallel, these networks need to be partitioned and distributed across different workers. How to generate and manage partitions is an important issue. Another interesting guideline for further work is to propose an effective method to partition the network into sub-networks to optimize the distribution of the network across a cluster so that clustering approaches can run with minimal communication effort and at the highest level of parallelism.

Secondly, considering the research line related to the novel pre-processing approach proposed in chapter 6, the approach has two aspects, which are worth investigating further:

- The proposed pre-processing approach utilizes a similarity function for comparing attributes. In a wide range of real-life applications, data contains a mixed type of attributes (e.g. numerical, categorical). Therefore, it is important to use appropriate similarity metrics to correctly measure the attribute proximity between two nodes in the network. However, the appropriate choice of the similarity measure depends on the attribute type of network to study. The natural extension of work in chapter 6 is to use a more sophisticated approach that supports datasets with mixed attribute types.
- Combining the proposed pre-processing approach with DDICA and PDDICA approaches (Algorithms proposed in chapter 4 and 5) for identifying more realistic communities.

Finally, for the smartphone usage dataset, although in chapter 7 of this thesis, data skimming type of analysis was carried out on real-life big dataset (Device Analyzer) from Cambridge Laboratories to understand the behavioural patterns of different mobile users, in the future, I intend to extend the analysis and studies to test the proposed community clustering approaches

DICCA/PDICCA on big telecom datasets to extract community clusters and find hidden trends and behavioural patterns. This could help CSPs improve profitability in many ways:

- Optimizing network routing and quality of service by analysing network traffic in real time.
- Improving security by analysing call data records in real time to identify fraudulent behaviour immediately.
- Enhancing customer experience by using insights into customer behaviour and usage to develop new products and services.

REFERENCES

Adamic, L.A. and Glance, N. (2005) The political blogosphere and the 2004 US election: divided they blog. *Proceedings of the 3rd international workshop on Link discovery of Conference*.

Aggarwal, C.C. and Wang, H. (2010) A survey of clustering algorithms for graph data. *Managing and mining graph data*, 275-301.

Aiello, W., Chung, F. and Lu, L. (2000) A random graph model for massive graphs. *Proceedings of the thirty-second annual ACM symposium on Theory of computing of Conference*.

Albert, R., Jeong, H. and Barabási, A.-L. (1999) The diameter of the world wide web. *arXiv preprint cond-mat/9907038*.

Almeida, H., Guedes, D., Meira, W. and Zaki, M.J. (2011) Is there a best quality metric for graph clusters? *Joint European Conference on Machine Learning and Knowledge Discovery in Databases of Conference*.

Amelio, A. and Pizzuti, C. (2014) Overlapping community discovery methods: a survey. In: (ed.) *Social Networks: Analysis and Case Studies*. Springer. pp. 105-125.

Amodio, S., D'Ambrosio, A., Iorio, C. and Siciliano, R. (2015) Adjusted Concordance Index, an extension of the Adjusted Rand index to fuzzy partitions. *arXiv preprint arXiv:1509.00803*.

Arias-Castro, E., Pelletier, B. and Pudlo, P. (2012) The normalized graph cut and Cheeger constant: from discrete to continuous. *Advances in Applied Probability*, 44 (4), 907-937.

Arif, M. and Basalamah, S. (2012) Similarity-dissimilarity plot for high dimensional data of different attribute types in biomedical datasets. *International Journal of Innovative Computing, Information and Control*, 8 (2), 1275-1297.

Bae, S.-H., Halperin, D., West, J.D., Rosvall, M. and Howe, B. (2017) Scalable and Efficient Flow-Based Community Detection for Large-Scale Graph Analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11 (3), 32.

Barkhuus, L. and Polichar, V.E. (2011) Empowerment through seamfulness: smart phones in everyday life. *Personal and Ubiquitous Computing*, 15 (6), 629-639.

Barnes, J. and Hut, P. (1986) A hierarchical $O(N \log N)$ force-calculation algorithm. *nature*, 324 (6096), 446.

Bedi, P. and Sharma, C. (2016) Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6 (3), 115-135.

Behrisch, M., Bach, B., Henry Riche, N., Schreck, T. and Fekete, J.D. (2016) Matrix reordering methods for table and network visualization. *Computer Graphics Forum of Conference*.

Blondel, V.D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. (2008) Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008 (10), P10008.

Borgatti, S.P., Everett, M.G. and Johnson, J.C. (2013) *Analyzing social networks*. SAGE Publications Limited.

Bu, Y., Howe, B., Balazinska, M. and Ernst, M.D. (2010) HaLoop: Efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment*, 3 (1-2), 285-296.

Cambridge, U.o. (2014) *Device Analyzer* [online]

Available at: <http://deviceanalyzer.cl.cam.ac.uk>

[Accessed: september]

Canright, G.S. and Engø-Monsen, K. (2008) Introducing network analysis. *Teletronikk. v1*.

Chen, J., Zaiane, O.R. and Goebel, R. (2009) Detecting communities in large networks by iterative local expansion. *Computational Aspects of Social Networks, 2009. CASON'09. International Conference on of Conference*.

Chen, P.-Y. and Hero, A.O. (2015) Deep community detection. *IEEE Transactions on Signal Processing*, 63 (21), 5706-5719.

Choi, S.-S., Cha, S.-H. and Tappert, C.C. (2010) A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8 (1), 43-48.

Clauset, A., Newman, M.E. and Moore, C. (2004) Finding community structure in very large networks. *Physical review E*, 70 (6), 066111.

Costa, L.d.F., Rodrigues, F.A., Travieso, G. and Villas Boas, P.R. (2007) Characterization of complex networks: A survey of measurements. *Advances in physics*, 56 (1), 167-242.

Csardi, G. and Nepusz, T. (2006) The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695 (5), 1-9.

Dang, T.A. and Viennet, E. Community detection based on structural and attribute similarities, 2012 of Conference.

Danon, L., Diaz-Guilera, A., Duch, J. and Arenas, A. (2005) Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005 (09), P09008.

Davis, J.A. (1970) Clustering and hierarchy in interpersonal relations: Testing two graph theoretical models on 742 sociomatrices. *American Sociological Review*, 843-851.

De, D. (2016) *Mobile cloud computing: architectures, algorithms and applications*. CRC Press.

Dean, J. and Ghemawat, S. (2008) MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51 (1), 107-113.

Derbeko, P., Dolev, S., Gudes, E. and Sharma, S. (2016) Security and Privacy aspects in MapReduce on clouds: A survey. *Computer Science Review*, 20, 1-28.

Do, T.M.T., Blom, J. and Gatica-Perez, D. (2011) Smartphone usage in the wild: a large-scale analysis of applications and context. *Proceedings of the 13th international conference on multimodal interfaces of Conference*.

Doulkeridis, C. and Nørnvåg, K. (2014) A survey of large-scale analytical query processing in MapReduce. *The VLDB Journal*, 23 (3), 355-380.

Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J. and Fox, G. (2010) Twister: a runtime for iterative mapreduce. *Proceedings of the 19th ACM international symposium on high performance distributed computing of Conference*.

eMarketer (November 23, 2016) eMarketer: Mobile Phone, Smartphone Usage Varies Globally. [online], eMarketer

Available at: <https://www.emarketer.com/Article/Mobile-Phone-Smartphone-Usage-Varies-Globally/1014738>]

Emmons, S., Kobourov, S., Gallant, M. and Börner, K. (2016) Analysis of network clustering algorithms and cluster quality metrics at scale. *PloS one*, 11 (7), e0159161.

Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R. and Estrin, D. (2010) Diversity in smartphone usage. *Proceedings of the 8th international conference on Mobile systems, applications, and services* of Conference.

Faloutsos, M., Faloutsos, P. and Faloutsos, C. (1999) On power-law relationships of the internet topology. *ACM SIGCOMM computer communication review* of Conference.

Fernández, A., del Río, S., López, V., Bawakid, A., del Jesus, M.J., Benítez, J.M. and Herrera, F. (2014) Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4 (5), 380-409.

Fortunato, S. *Benchmark graphs for testing community detection algorithms* [online]
Available at: www.santo.fortunato.googlepages.com/benchmark.tgz

Fortunato, S. (2010) Community detection in graphs. *Physics reports*, 486 (3), 75-174.

Fortunato, S. and Barthélemy, M. (2007) Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104 (1), 36-41.

Gehweiler, J. and Meyerhenke, H. (2010) A distributed diffusive heuristic for clustering a virtual P2P supercomputer. *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on* of Conference.

Ghoniem, M., Fekete, J.-D. and Castagliola, P. (2004) A comparison of the readability of graphs using node-link and matrix-based representations. *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on* of Conference.

Girvan, M. and Newman, M.E. (2002) Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99 (12), 7821-7826.

Grinter, R.E. and Eldridge, M.A. (2001) y do tngrs luv 2 txt msg? *ECSCW 2001* of Conference.

Hadoop, A. (2016) *Welcome to apache hadoop*.

Herman, I., Melançon, G. and Marshall, M.S. (2000) Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on visualization and computer graphics*, 6 (1), 24-43.

- Hu, H., Wen, Y., Chua, T.-S. and Li, X. (2014) Toward scalable systems for big data analytics: A technology tutorial. *IEEE access*, 2, 652-687.
- Hu, Y. (2005) Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10 (1), 37-71.
- Hubert, L. and Arabie, P. (1985) Comparing partitions. *Journal of classification*, 2 (1), 193-218.
- Huijuan, Z. and Shixuan, S. (2013) A Graph Clustering algorithm based on shared neighbors and connectivity. *Computer Science & Education (ICCSE), 2013 8th International Conference onof Conference*.
- Jackson, M.O. (2010) An overview of social networks and economic applications. *The handbook of social economics*, 1, 511-585.
- Jacomy, M., Venturini, T., Heymann, S. and Bastian, M. (2014) ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PloS one*, 9 (6), e98679.
- Jain, A.K., Murty, M.N. and Flynn, P.J. (1999) Data clustering: a review. *ACM computing surveys (CSUR)*, 31 (3), 264-323.
- Kajdanowicz, T., Kaziemko, P. and Indyk, W. (2014) Parallel processing of large graphs. *Future Generation Computer Systems*, 32, 324-337.
- Kang, U., Lee, J.-Y., Koutra, D. and Faloutsos, C. (2014) Net-ray: Visualizing and mining billion-scale graphs. *Pacific-Asia Conference on Knowledge Discovery and Data Mining of Conference*.
- Kannan, R., Vempala, S. and Vetta, A. (2004) On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51 (3), 497-515.
- Khatoon, M. and Banu, W.A. (2015) A survey on community detection methods in social networks. *International Journal of Education and Management Engineering (IJEME)*, 5 (1), 8.
- Labatut, V. (2015) Generalised measures for the evaluation of community detection methods. *International Journal of Social Network Mining*, 2 (1), 44-63.
- Lancichinetti, A., Fortunato, S. and Radicchi, F. (2008) Benchmark graphs for testing community detection algorithms. *Physical review E*, 78 (4), 046110.

Leskovec, J., Lang, K.J. and Mahoney, M. (2010) Empirical comparison of algorithms for network community detection. *Proceedings of the 19th international conference on World wide web* of Conference.

Li, R., Guo, W., Guo, K. and Qiu, Q. (2015) Parallel multi-label propagation for overlapping community detection in large-scale networks. *International Workshop on Multi-disciplinary Trends in Artificial Intelligence* of Conference.

Lin, W., Kong, X., Yu, P.S., Wu, Q., Jia, Y. and Li, C. (2012) Community detection in incomplete information networks. *Proceedings of the 21st international conference on World Wide Web* of Conference.

Liu, X. (2012) A survey on clustering routing protocols in wireless sensor networks. *sensors*, 12 (8), 11113-11153.

Louch, H. (2000) Personal network integration: transitivity and homophily in strong-tie relations. *Social networks*, 22 (1), 45-64.

Mahata, D. and Patra, C. (2016) Detecting and analyzing invariant groups in complex networks. In: (ed.) *Computational Intelligence in Data Mining—Volume 1*. Springer. pp. 85-93.

Malliaros, F.D. and Vazirgiannis, M. (2013) Clustering and community detection in directed networks: A survey. *Physics Reports*, 533 (4), 95-142.

Martin, S., Brown, W.M., Klavans, R. and Boyack, K.W. (2011) OpenOrd: an open-source toolbox for large graph layout. *Visualization and Data Analysis 2011* of Conference.

MATLAB (Release 2017a) Parallel Computing Toolbox, The MathWorks. Inc., Natick, Massachusetts, United State

[online],

Available at: https://uk.mathworks.com/products/parallel-computing.html?s_tid=srchtitle]

McPherson, M., Smith-Lovin, L. and Cook, J.M. (2001) Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27 (1), 415-444.

Moghaddam, S., Helmy, A., Ranka, S. and Somaiya, M. (2010) Data-driven co-clustering model of internet usage in large mobile societies. *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems* of Conference.

Mohebi, A., Aghabozorgi, S., Ying Wah, T., Herawan, T. and Yahyapour, R. (2016) Iterative big data clustering algorithms: a review. *Software: Practice and Experience*, 46 (1), 107-129.

Mutchler, L.A., Shim, J. and Ormond, D. (2011) Exploratory Study on Users' Behavior: Smartphone Usage. *AMCISof Conference*.

Newman, M. (2010) *Networks: an introduction*. Oxford university press.

Newman, M. (2016) Community detection in networks: Modularity optimization and maximum likelihood are equivalent. *arXiv preprint arXiv:1606.02319*.

Newman, M.E. (2001) Scientific collaboration networks. I. Network construction and fundamental results. *Physical review E*, 64 (1), 016131.

Newman, M.E. (2003) The structure and function of complex networks. *SIAM review*, 45 (2), 167-256.

Newman, M.E. (2004a) Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38 (2), 321-330.

Newman, M.E. (2004b) Fast algorithm for detecting community structure in networks. *Physical review E*, 69 (6), 066133.

Newman, M.E. (2006a) Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74 (3), 036104.

Newman, M.E. (2006b) Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103 (23), 8577-8582.

Newman, M.E. and Girvan, M. (2004) Finding and evaluating community structure in networks. *Physical review E*, 69 (2), 026113.

Orman, G.K., Labatut, V. and Cherifi, H. (2011) On accuracy of community structure discovery algorithms. *arXiv preprint arXiv:1112.4134*.

Orman, G.K., Labatut, V. and Cherifi, H. (2012) Comparative evaluation of community detection algorithms: a topological approach. *Journal of Statistical Mechanics: Theory and Experiment*, 2012 (08), P08001.

Parslow, R., Hepworth, S. and McKinney, P. (2003) Recall of past use of mobile phone handsets. *Radiation protection dosimetry*, 106 (3), 233-240.

Pons, P. and Latapy, M. (2005) Computing communities in large networks using random walks. *International Symposium on Computer and Information Sciencesof Conference*.

Pons, P. and Latapy, M. (2006) Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, 10 (2), 191-218.

Rahimian, F., Payberah, A.H., Girdzijauskas, S., Jelasily, M. and Haridi, S. (2013) Ja-be-ja: A distributed algorithm for balanced graph partitioning. *Self-Adaptive and Self-Organizing Systems (SASO), 2013 IEEE 7th International Conference onof Conference*.

Rahmati, A. and Zhong, L. (2013) Studying smartphone usage: lessons from a four-month field study. *Mobile Computing, IEEE Transactions on*, 12 (7), 1417-1427.

Ramaswamy, L., Gedik, B. and Liu, L. (2005) A distributed approach to node clustering in decentralized peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 16 (9), 814-829.

Rand, W.M. (1971) Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66 (336), 846-850.

Redner, S. (1998) How popular is your paper? An empirical study of the citation distribution. *The European Physical Journal B-Condensed Matter and Complex Systems*, 4 (2), 131-134.

Reid, D. and Reid, F. (2004) Insights into the social and psychological effects of SMS text messaging.

Rosvall, M. and Bergstrom, C.T. (2008) Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105 (4), 1118-1123.

Ruan, Y., Fuhry, D. and Parthasarathy, S. (2013) Efficient community detection in large networks using content and links. *Proceedings of the 22nd international conference on World Wide Webof Conference*.

Salem, S. and Ozcaglar, C. (2014) Hybrid coexpression link similarity graph clustering for mining biological modules from multiple gene expression datasets. *BioData mining*, 7 (1), 16.

Schaeffer, S.E. (2007) Graph clustering. *Computer science review*, 1 (1), 27-64.

Schank, T. and Wagner, D. Approximating Clustering-Coefficient and Transitivity.

Shah, D. and Zaman, T. (2010) Community detection in networks: The leader-follower algorithm. *arXiv preprint arXiv:1011.0774*.

Shye, A., Scholbrock, B., Memik, G. and Dinda, P.A. (2010) Characterizing and modeling user activity on smartphones: summary. *ACM SIGMETRICS Performance Evaluation Review of Conference*.

Silva, T.C. and Zhao, L. (2016) *Machine learning in complex networks*. Springer.

Sumathi, S. and Esakkirajan, S. (2007) *Fundamentals of relational database management systems*. Springer.

Tomassini, M. (2010) Introduction to graphs and networks. *Information Systems Department, HEC, University of Lausanne, Switzerland*.

Traud, A.L., Kelsic, E.D., Mucha, P.J. and Porter, M.A. (2011) Comparing community structure to characteristics in online collegiate social networks. *SIAM review*, 53 (3), 526-543.

Traud, A.L., Mucha, P.J. and Porter, M.A. (2012) Social structure of Facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391 (16), 4165-4180.

Travers, J. and Milgram, S. (1967) The small world problem. *Psychology Today*, 1, 61-67.

Travers, J. and Milgram, S. (1969) An experimental study of the small world problem. *Sociometry*, 425-443.

Verkasalo, H. and Hämmäinen, H. (2007) A handset-based platform for measuring mobile service usage. *info*, 9 (1), 80-96.

Vocaturro, E. and Veltri, P. (2017) On the use of Networks in Biomedicine. *Procedia Computer Science*, 110, 498-503.

Von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J.J., Fekete, J.D. and Fellner, D.W. (2011) Visual analysis of large graphs: state - of - the - art and future research challenges. *Computer graphics forum of Conference*.

Wagner, S. and Wagner, D. (2007) *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe.

Wang, M., Wang, C., Yu, J.X. and Zhang, J. (2015) Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework. *Proceedings of the VLDB Endowment*, 8 (10), 998-1009.

Warnke, S.D. (2016) *Partial information community detection in a multilayer network*. Naval Postgraduate School Monterey United States.

Watts, D.J. and Strogatz, S.H. (1998a) Collective dynamics of 'small-world' networks. *nature*, 393 (6684), 440.

Watts, D.J. and Strogatz, S.H. (1998b) Collective dynamics of 'small-world' networks. *nature*, 393 (6684), 440-442.

Weber, L.M. and Robinson, M.D. (2016) Comparison of clustering methods for high - dimensional single - cell flow and mass cytometry data. *Cytometry Part A*, 89 (12), 1084-1096.

Wehmeier, T. (2012) *Understanding today's smartphone user: Demystifying data usage trends on cellular & Wi-Fi networks*. Informa Telecoms and Media.

Xavier, F.H.Z., Silveira, L.M., Almeida, J.M.d., Ziviani, A., Malab, C.H.S. and Marques-Neto, H.T. (2012) Analyzing the workload dynamics of a mobile phone network in large scale events. *Proceedings of the first workshop on Urban networking of Conference*.

Xu, Q., Erman, J., Gerber, A., Mao, Z., Pang, J. and Venkataraman, S. (2011) Identifying diverse usage behaviors of smartphone apps. *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference of Conference*.

Yang, T., Jin, R., Chi, Y. and Zhu, S. (2009) Combining link and content for community detection: a discriminative approach. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining of Conference*.

Yang, Z., Algesheimer, R. and Tessone, C.J. (2016) A comparative analysis of community detection algorithms on artificial networks. *Scientific reports*, 6, 30750.

Yu, J.Y. and Chong, P.H.J. (2005) A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, 7 (1), 32-48.

Zachary, W.W. (1977) An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33 (4), 452-473.

Zhang, Y., Gao, Q., Gao, L. and Wang, C. (2012) imapreduce: A distributed computing framework for iterative computation. *Journal of Grid Computing*, 10 (1), 47-68.

Zhou, Y., Cheng, H. and Yu, J.X. (2009) Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2 (1), 718-729.

APPENDIX

Appendix A: Additional Results

A.1 Additional Results for DICCA described in chapter 4

Table A.1.1 Scalability of the proposed algorithm performance

size	Ground-truth Q	No. Of Ground-truth clusters	NMI	Modularity (Q)	Execution Time	No. Of Msg	No. of cluster	Iteration	ARI
500	0.819	16	0.914	0.765	3.355	1401045	13	5	0.751
1000	0.859	32	0.934	0.822	13.919	2681195	27	5	0.785
1500	0.873	51	0.937	0.839	33.846	4093201	41	6	0.758
2000	0.880	69	0.943	0.851	65.918	5484550	55	6	0.761
2500	0.884	88	0.948	0.857	109.672	6803586	70	7	0.769
3000	0.887	104	0.947	0.858	177.191	8404026	82	7	0.754
3500	0.889	123	0.950	0.861	254.517	9705058	98	7	0.758
4000	0.890	134	0.950	0.860	397.839	11814798	107	7	0.769
4500	0.891	155	0.953	0.864	524.625	13060131	124	7	0.766
5000	0.892	173	0.954	0.866	665.021	14664776	138	7	0.771

Table A.1.2 Summary of the performance of the proposed algorithm using Mixing parameter for n=500

Mixing parameter	GT Modularity	No. Of GT Cluster	NMI	Modularity	Time	No. Of Msg	No. Of Cluster	Iteration	ARI
0.1	0.819	16	0.914	0.765	3.355	1401045	13	5	0.751
0.15	0.768	16	0.791	0.624	3.364	1549821	11	7	0.557
0.2	0.721	16	0.742	0.551	3.435	1539140	10	7	0.498
0.25	0.670	17	0.708	0.493	3.699	2001578	10	8	0.468
0.3	0.628	17	0.692	0.451	3.960	2232687	11	8	0.455
0.35	0.576	16	0.645	0.383	3.935	2752697	11	8	0.454
0.4	0.528	16	0.591	0.321	4.035	3590013	12	9	0.403
0.45	0.481	18	0.555	0.266	3.988	4556827	15	10	0.361
0.5	0.427	16	0.540	0.216	4.322	6348157	24	8	0.333
0.55	0.382	17	0.484	0.159	4.489	8247020	34	7	0.232
0.6	0.341	16	0.436	0.123	4.999	10387158	43	7	0.157
0.65	0.286	17	0.388	0.095	5.534	14135123	52	7	0.087
0.7	0.233	17	0.348	0.083	6.199	16151336	56	7	0.048
0.75	0.181	17	0.313	0.079	5.931	16231281	57	7	0.028

Table A.1.3 Summary of the performance of the proposed algorithm using Mixing parameter for =1000

Mixing parameter	GT Modularity	No. Of GT Cluster	NMI	Modularity	Time	No. Of Msg	No. Of Cluster	Iteration	ARI
0.1	0.859	32	0.934	0.822	13.919	2681195	27	5	0.785
0.15	0.811	34	0.901	0.753	13.249	2973750	26	6	0.698
0.2	0.760	33	0.875	0.681	13.277	3370322	26	6	0.680
0.25	0.712	33	0.837	0.609	14.029	4271018	26	7	0.615
0.3	0.663	34	0.820	0.546	14.864	5350932	28	7	0.617
0.35	0.614	34	0.780	0.476	14.726	7064995	29	7	0.565
0.4	0.566	35	0.753	0.409	14.556	9379372	32	8	0.557
0.45	0.515	33	0.699	0.331	14.052	13770905	36	9	0.505
0.5	0.465	33	0.643	0.242	14.861	21328634	57	8	0.392
0.55	0.415	33	0.587	0.166	15.997	35480627	89	7	0.261
0.6	0.367	34	0.549	0.119	16.890	45995466	114	6	0.167
0.65	0.316	34	0.500	0.091	17.343	53388924	133	5	0.095
0.7	0.266	35	0.475	0.080	18.744	57209564	144	5	0.060
0.75	0.219	36	0.450	0.074	17.433	54101108	147	4	0.039

Table A.1.4 Performance of DICCA algorithm using different TTL values for n=500 without using Min_VALUE condition

TTL	GT Modularity	No. Of GT Cluster	NMI	Modularity	Time	No. Of Msg	No. Of Cluster	Iteration	ARI
1	0.819	16	0.661	0.583	0.296	4832	8	11	0.398
2	0.819	16	0.875	0.734	0.853	62990	12	6	0.669
3	0.819	16	0.918	0.764	24.873	1347024	13	5	0.763
4	0.819	16	0.922	0.765	10407.076	29680547	13	5	0.751

Table A.1.5 Performance of DICCA algorithm using different TTL values for n=500 when using Min_VALUE condition

TTL	GT Modularity	No. Of GT Cluster	NMI	Modularity	Time	No. Of Msg	No. Of Cluster	Iteration	ARI
1	0.819	16	0.689	0.608	0.289	4928	8	11	0.421
2	0.819	16	0.872	0.726	0.804	63881	12	6	0.648
3	0.819	16	0.914	0.765	3.355	1401045	13	5	0.751
4	0.819	16	0.915	0.766	5.039	3388457	13	5	0.754

Table A.1.6 Performance of DICCA algorithm using different TTL values for n=1000 without using Min_VALUE condition

TTL	GT Modularity	No. Of GT Cluster	NMI	Modularity	Time	No. Of Msg	No. Of Cluster	Iteration	ARI
1	0.859	32	0.769	0.695	0.890	9019	17	10	0.475
2	0.859	32	0.926	0.819	3.498	166525	26	6	0.760
3	0.859	32	0.946	0.831	98.738	3735475	27	5	0.810
4	0.859	32	0.956	0.838	34333.526	87794210	28	5	0.837

Table A.1.7 Performance of DICCA algorithm using different TTL values for n=1000 when using Min_VALUE condition

TTL	GT Modularity	No. Of GT Cluster	NMI	Modularity	Time	No. Of Msg	No. Of Cluster	Iteration	ARI
1	0.859	32	0.764	0.693	0.899	9031	17	10	0.477
2	0.859	32	0.930	0.820	3.078	162377	26	6	0.780
3	0.859	32	0.934	0.822	13.919	2681195	27	5	0.785
4	0.859	32	0.933	0.821	19.617	6963794	26	5	0.785

A.2 Additional Results for PDICCA described in chapter 5

Table A.2.1 Summary of the performance of the proposed algorithm using Mixing parameter for n=500

Mixing parameter	GT Modularity	No. Of GT Cluster	NMI-PDICC A	Q-PDICC A	Time	No. Of Msg	No. Of Cluster	Iteration	No. Of Swapped Msg	ARI
0.1	0.8189	16.4	0.9488	0.7824	8.5995	1344282	15	5	489	0.8708
0.15	0.7678	16.4	0.8953	0.6950	8.5076	1773968	14	5	504	0.7506
0.2	0.7207	16.4	0.8677	0.6265	7.9940	2222215	14	5	518	0.7089
0.25	0.6705	16.8	0.8118	0.5376	8.8348	3561679	14	6	527	0.6312
0.3	0.6278	17.4	0.7747	0.4771	8.8118	4511721	14	6	536	0.5878
0.35	0.5759	16.2	0.7038	0.3867	8.6034	7490749	16	7	552	0.5165
0.4	0.5282	16.2	0.6427	0.2924	9.1207	12045080	23	7	550	0.4387
0.45	0.4811	17.6	0.6004	0.2235	9.5374	18467793	35	6	544	0.3330
0.5	0.4267	16	0.5159	0.1429	10.4627	31859841	54	6	534	0.2034
0.55	0.3817	16.6	0.4777	0.1140	10.8221	36917671	59	6	530	0.1467
0.6	0.3414	16.4	0.4384	0.0928	11.1384	43860527	68	5	524	0.0976
0.65	0.2858	17.2	0.4129	0.0769	12.6901	54613811	81	5	517	0.0593
0.7	0.2332	17.4	0.3710	0.0729	12.5550	55755527	78	5	518	0.0369
0.75	0.1813	17.2	0.3426	0.0702	13.2440	59511312	80	6	519	0.0216

Table A.2.2 Summary of the performance of the proposed algorithm using Mixing parameter for n=1000

Mixing parameter	GT Modularity	No. Of GT Cluster	NMI-PDICC A	Q-PDICC A	Time	No. Of Msg	No. Of Cluster	Iteration	No. Of Swapped Msg	ARI
0.1	0.8592	32	0.9498	0.8231	37.1294	2657238	28	5	981	0.8413
0.15	0.8106	34.2	0.9315	0.7550	36.3261	3726253	30	5	1001	0.8051
0.2	0.7603	33.2	0.8936	0.6723	33.5093	5571812	28	5	1038	0.7383
0.25	0.7121	33.4	0.8644	0.5997	35.1898	8534686	29	6	1061	0.6925
0.3	0.6629	34.4	0.8312	0.5158	31.5680	13850086	32	6	1077	0.6506
0.35	0.6145	34.2	0.7879	0.4373	32.1823	21098526	34	7	1098	0.5882
0.4	0.5656	35	0.7413	0.3418	28.8428	36494619	48	7	1102	0.5041
0.45	0.5152	33.2	0.6776	0.2355	31.6455	70023087	77	6	1092	0.3749
0.5	0.4654	33.4	0.6222	0.1625	31.7654	115217719	111	6	1068	0.2484
0.55	0.4154	32.8	0.5696	0.1158	37.2374	155653645	142	5	1045	0.1529
0.6	0.3668	34	0.5388	0.0906	38.5380	189698826	164	5	1036	0.1021
0.65	0.3160	33.6	0.5010	0.0779	39.8445	200467611	179	5	1024	0.0657
0.7	0.2664	35	0.4765	0.0707	42.6053	215485993	191	5	1017	0.0421
0.75	0.2186	36	0.4576	0.0678	42.8690	222938148	196	5	1015	0.0291

Table A.2.3 Number of messages exchanged in each iterations for each worker when the number of workers is two for n=500 and 1000

Number of nodes	No. Of Exchanged Msg			
	500		1000	
	1st worker	2nd worker	1st worker	2nd worker
1st Iteration	717661	733163	1295619	1297506
2nd Iteration	66842	63418	163026	166334.5
3rd Iteration	8909	8651	36774.34	37814.39
The rest	5777	5873	36194	369267

Table A.2.4 Number of messages in each iterations when the number of workers is three for n=500 and 1000

Number of nodes	500			1000		
	1st worker	2nd worker	3rd worker	1st worker	2nd worker	3rd worker
1st Iteration	347991	349277	360717	628164	648158	626507
2nd Iteration	73781	72508	76838	166452	170013	166283
3rd Iteration	12626	12920	13475	44852	45586	46264
The rest	8411	8390	8673	36746	38581	39632

Table A.2.5 Number of messages exchanged in each iterations when the number of workers is four for n=500 and 1000

Number of nodes	500				1000			
	1st worker	2nd worker	3rd worker	4th worker	1st worker	2nd worker	3rd worker	4th worker
1st Iteration	213942	206759	209541	209940	398284	385951	394337	372525
2nd Iteration	101267	91984	97896	96235	210895	209678	208015	206202
3rd Iteration	25039	23514	25520	25239	79371	78650	78701	78636
The rest	14665	14991	15895	15376	74467	68562	71947	70707

A.3 Additional results for pre-processing approach described in chapter6

Table A.3.1 Agreement of different community detection algorithms with each attribute for Caltech36 and Reed9 datasets using Jaccard index similarity.

Data set	Caltech36				Reed98			
	Fast Modularity algorithm	Louvain	leading eigenvector	Walktrap	Fast Modularity algorithm	Louvain	leading eigenvector	Walktrap
student/ faculty	0.2772	0.1629	0.1539	0.0989	0.4023	0.2106	0.2189	0.1866
Gender	0.2412	0.1478	0.1461	0.0898	0.2761	0.1692	0.1660	0.1543
major	0.0573	0.0530	0.0519	0.0473	0.0364	0.0344	0.0333	0.0360
second major/ minor	0.0034	0.0036	0.0037	0.0042	0.0059	0.0056	0.0061	0.0054
dormitory	0.1746	0.3220	0.2537	0.3720	0.0231	0.0210	0.0199	0.0181
year	0.1239	0.0973	0.0917	0.0840	0.2432	0.3060	0.2683	0.2482
High school	0.0009	0.0010	0.0011	0.0012	0.0005	0.0005	0.0007	0.0005

Table A.3.2 Agreement of different community detection algorithms with each attribute for Haverford76 and Aassar85 datasets. Using Jaccard index similarity.

Data set	Haverford76				Aassar85			
Attribute	Fast Modularity algorithm	Louvain	leading eigenvector	Walktrap	Fast Modularity algorithm	Louvain	leading eigenvector	Walktrap
student/ faculty	0.3214	0.2559	0.2156	0.3012	0.3585	0.2317	0.2647	0.2177
Gender	0.2443	0.1644	0.1697	0.2235	0.2643	0.1788	0.1912	0.1614
major	0.0346	0.0334	0.0348	0.0388	0.0301	0.0306	0.0313	0.0313
second major/ minor	0.0091	0.0093	0.0096	0.0104	0.0072	0.0074	0.0076	0.0077
dormitory	0.0958	0.1024	0.0945	0.0992	0.0741	0.0732	0.0671	0.0703
year	0.2862	0.4739	0.3369	0.3979	0.2896	0.4409	0.3455	0.4315
High school	0.0008	0.0009	0.0009	0.0008	0.0008	0.0009	0.0008	0.0008

Table A.3.3 The influence of the parameters α and β on the quality of clustering solutions for Caltech36 and Reed98 datasets

Data set		Caltech36				Reed98			
α	β	Fast Modularity algorithm	Louvain	leading eigenvector	Walktrap	Fast Modularity algorithm	Louvain	leading eigenvector	Walktrap
0	0	0.3212	0.3837	0.3181	0.2600	0.2534	0.3011	0.2279	0.0945
0	0.1	0.3230	0.3825	0.2720	0.1563	0.2420	0.2932	0.1786	0.2103
0	0.2	0.1981	0.2972	0.1547	0.0787	0.2358	0.2330	0.1746	0.1399
0	0.3	0.1242	0.0840	0.1003	0.0770	0.0744	0.1266	0.0085	0.0421
0	0.4	0.0867	0.0806	0.0896	0.0744	0.0843	0.0593	0.0778	0.0681
0	0.5	0.0847	0.1023	0.0898	0.0746	0.0841	0.0557	0.0721	0.0735
0	0.6	0.0821	0.0804	0.0828	0.0771	0.0655	0.0636	0.0735	0.0678
0	0.7	0.0821	0.0803	0.0856	0.0768	0.0655	0.0550	0.0081	0.0411
0	0.8	0.0821	0.0805	0.0812	0.0617	0.0655	0.0547	0.0659	0.0427
0	0.9	0.0847	0.0806	0.0853	0.0592	0.0655	0.0552	0.0546	0.0414
0	1	0.0821	0.0806	0.0860	0.0778	0.0655	0.0547	0.0373	0.0384
0.1	0	0.3213	0.3872	0.3272	0.3144	0.2859	0.3200	0.2619	0.2304
0.1	0.1	0.3212	0.3743	0.2480	0.2034	0.1946	0.2965	0.1716	0.1864
0.1	0.2	0.3148	0.3603	0.1767	0.0787	0.1745	0.2945	0.1670	0.1357
0.1	0.3	0.1025	0.2655	0.1040	0.0625	0.0551	0.2705	0.0112	0.0429
0.1	0.4	0.0883	0.2789	0.0993	0.0768	0.1146	0.2566	0.1235	0.0681
0.1	0.5	0.0863	0.0822	0.1664	0.0746	0.0945	0.0625	0.0813	0.0849
0.1	0.6	0.0868	0.0819	0.0871	0.0746	0.0835	0.0550	0.0311	0.0490
0.1	0.7	0.0848	0.0576	0.0693	0.0632	0.0824	0.1485	0.0755	0.0407
0.1	0.8	0.0847	0.0805	0.0863	0.0778	0.0655	0.0545	0.0082	0.0411
0.1	0.9	0.0821	0.0805	0.0856	0.0617	0.0824	0.0630	0.0659	0.0426
0.1	1	0.0847	0.1484	0.0874	0.0592	0.0655	0.0555	0.0961	0.0412

0.2	0	0.3199	0.3868	0.3347	0.3301	0.2869	0.3263	0.2736	0.2486
0.2	0.1	0.3247	0.3931	0.2570	0.2357	0.2184	0.2832	0.1729	0.2307
0.2	0.2	0.3246	0.3097	0.2102	0.0806	0.1801	0.2871	0.1923	0.1407
0.2	0.3	0.2365	0.2922	0.1150	0.0626	0.1846	0.2511	0.0108	0.0576
0.2	0.4	0.1743	0.2844	0.1087	0.0780	0.0638	0.2991	0.1030	0.0585
0.2	0.5	0.0867	0.0828	0.0967	0.0749	0.0903	0.0610	0.0965	0.0681
0.2	0.6	0.0833	0.1927	0.1722	0.0800	0.1076	0.2610	0.0831	0.0897
0.2	0.7	0.0920	0.0822	0.0894	0.0746	0.0893	0.2628	0.0863	0.0842
0.2	0.8	0.0887	0.0589	0.0613	0.0632	0.0688	0.0631	0.0550	0.0393
0.2	0.9	0.0833	0.0806	0.0885	0.0800	0.0831	0.0553	0.0085	0.0414
0.2	1	0.0821	0.0806	0.0865	0.0619	0.0655	0.0617	0.0694	0.0419
0.3	0	0.3014	0.3876	0.3435	0.3169	0.2871	0.3275	0.2831	0.2679
0.3	0.1	0.3006	0.3936	0.2629	0.2932	0.2656	0.3307	0.2869	0.2571
0.3	0.2	0.2566	0.3915	0.2745	0.2337	0.1897	0.2815	0.1926	0.1507
0.3	0.3	0.2338	0.3186	0.1343	0.0820	0.2323	0.2978	0.1902	0.1782
0.3	0.4	0.1690	0.2851	0.1182	0.0637	0.2001	0.2720	0.0140	0.0441
0.3	0.5	0.1561	0.2665	0.1114	0.0752	0.1823	0.2699	0.1612	0.0678
0.3	0.6	0.1443	0.0825	0.0887	0.0745	0.1003	0.0655	0.0939	0.0596
0.3	0.7	0.0914	0.0851	0.1855	0.0800	0.0921	0.0629	0.0795	0.0563
0.3	0.8	0.1158	0.1586	0.1631	0.0800	0.0902	0.2642	0.0817	0.0619
0.3	0.9	0.0863	0.0804	0.0601	0.0632	0.0918	0.0659	0.0935	0.0678
0.3	1	0.0897	0.2050	0.0963	0.0757	0.0708	0.0550	0.0108	0.0424
0.4	0	0.3246	0.3918	0.3464	0.3471	0.2889	0.3285	0.2822	0.2760
0.4	0.1	0.3235	0.3966	0.3388	0.3213	0.2630	0.3299	0.2948	0.2272
0.4	0.2	0.3218	0.3947	0.2715	0.2322	0.2094	0.2671	0.1896	0.1416
0.4	0.3	0.3160	0.3203	0.2006	0.0806	0.1948	0.3008	0.1926	0.1797
0.4	0.4	0.2823	0.3160	0.1245	0.0629	0.2267	0.2799	0.0150	0.0445
0.4	0.5	0.1029	0.2992	0.1259	0.0780	0.2648	0.2757	0.0288	0.0430
0.4	0.6	0.1561	0.2926	0.1204	0.0750	0.1559	0.2726	0.1615	0.0693
0.4	0.7	0.1429	0.0861	0.0966	0.0734	0.0956	0.2583	0.0951	0.0634
0.4	0.8	0.0833	0.1945	0.1725	0.0800	0.1113	0.0695	0.0968	0.0573
0.4	0.9	0.0833	0.1947	0.1859	0.0778	0.0908	0.0550	0.0399	0.0404
0.4	1	0.0899	0.0822	0.0848	0.0578	0.0924	0.2548	0.0741	0.0678
0.5	0	0.3219	0.3952	0.3470	0.3311	0.2830	0.3161	0.2852	0.2661
0.5	0.1	0.3267	0.3950	0.3395	0.3017	0.2903	0.3319	0.2962	0.2590
0.5	0.2	0.3236	0.3899	0.2692	0.2034	0.1958	0.3170	0.2411	0.2201
0.5	0.3	0.3165	0.3659	0.2327	0.0816	0.1563	0.2840	0.2017	0.1539
0.5	0.4	0.3285	0.3764	0.2040	0.0808	0.2015	0.3019	0.2013	0.1641
0.5	0.5	0.2653	0.3189	0.1244	0.0622	0.2502	0.2823	0.0158	0.0574
0.5	0.6	0.2260	0.3146	0.1230	0.0780	0.2643	0.2648	0.0160	0.0457
0.5	0.7	0.1711	0.2966	0.1761	0.0775	0.2360	0.2800	0.2437	0.0748
0.5	0.8	0.2715	0.3041	0.1176	0.0807	0.2436	0.2610	0.0979	0.0691
0.5	0.9	0.0924	0.2190	0.0952	0.0800	0.2392	0.2589	0.0974	0.0805
0.5	1	0.0926	0.1957	0.1902	0.0778	0.1708	0.2276	0.0960	0.0854

0.6	0	0.3309	0.3950	0.3544	0.3366	0.2883	0.3299	0.2839	0.2708
0.6	0.1	0.3264	0.3764	0.3447	0.3487	0.2820	0.3319	0.2925	0.2560
0.6	0.2	0.3282	0.3953	0.3330	0.2472	0.2244	0.3143	0.2529	0.2115
0.6	0.3	0.3179	0.3914	0.2868	0.2081	0.1577	0.2691	0.1970	0.1246
0.6	0.4	0.2768	0.3754	0.2506	0.2035	0.1806	0.2917	0.2083	0.1605
0.6	0.5	0.2420	0.3862	0.2180	0.1496	0.1030	0.3053	0.2123	0.1931
0.6	0.6	0.2439	0.3338	0.2049	0.0634	0.2348	0.1405	0.0158	0.0447
0.6	0.7	0.1952	0.3082	0.1461	0.0627	0.2474	0.2770	0.0175	0.0581
0.6	0.8	0.1069	0.3002	0.1426	0.0778	0.2454	0.2752	0.0869	0.0450
0.6	0.9	0.1915	0.3115	0.1888	0.0777	0.2534	0.2808	0.2496	0.1593
0.6	1	0.1550	0.2983	0.1821	0.0768	0.2448	0.2802	0.1078	0.0678
0.7	0	0.3379	0.3999	0.3721	0.3362	0.2555	0.3246	0.2825	0.2602
0.7	0.1	0.3308	0.4005	0.3634	0.3312	0.2673	0.3232	0.2888	0.2733
0.7	0.2	0.2879	0.3976	0.3436	0.3234	0.2612	0.3295	0.2970	0.2434
0.7	0.3	0.2194	0.3909	0.2991	0.2254	0.1719	0.3193	0.2704	0.2147
0.7	0.4	0.2452	0.3950	0.2838	0.2236	0.1770	0.2755	0.1982	0.1511
0.7	0.5	0.2382	0.3807	0.2421	0.0814	0.2000	0.2869	0.2015	0.1471
0.7	0.6	0.2417	0.3870	0.2430	0.1955	0.1984	0.3051	0.2142	0.1729
0.7	0.7	0.3233	0.3804	0.2209	0.2220	0.1842	0.2987	0.2131	0.1635
0.7	0.8	0.2661	0.3424	0.2590	0.1617	0.2685	0.2752	0.2071	0.0456
0.7	0.9	0.2634	0.3101	0.0874	0.0641	0.2818	0.2865	0.2122	0.0610
0.7	1	0.2154	0.3029	0.1593	0.0777	0.2687	0.2747	0.0784	0.0717
0.8	0	0.3378	0.3994	0.3733	0.3336	0.2885	0.3185	0.2759	0.2581
0.8	0.1	0.3441	0.3996	0.3731	0.3403	0.2577	0.3176	0.2791	0.2611
0.8	0.2	0.3201	0.3960	0.3651	0.3782	0.2914	0.3169	0.2882	0.2629
0.8	0.3	0.3230	0.3991	0.3598	0.3246	0.2736	0.3269	0.2907	0.2797
0.8	0.4	0.3226	0.3726	0.3499	0.2820	0.1534	0.3199	0.2769	0.1659
0.8	0.5	0.3286	0.3930	0.2844	0.2144	0.2132	0.3039	0.2725	0.2135
0.8	0.6	0.2476	0.3969	0.2919	0.2080	0.1950	0.2811	0.2009	0.1269
0.8	0.7	0.3276	0.3830	0.2503	0.1529	0.2088	0.3008	0.2009	0.1269
0.8	0.8	0.2430	0.3636	0.2209	0.1501	0.2163	0.2797	0.2031	0.1227
0.8	0.9	0.2185	0.3702	0.2157	0.0783	0.2768	0.2941	0.2345	0.1304
0.8	1	0.1898	0.3915	0.2544	0.0783	0.2777	0.2944	0.2345	0.1455
0.9	0	0.3171	0.3976	0.3630	0.3304	0.2885	0.3244	0.2824	0.2683
0.9	0.1	0.3037	0.3998	0.3636	0.3367	0.2881	0.3229	0.2851	0.2540
0.9	0.2	0.3058	0.3962	0.3636	0.3436	0.2865	0.3225	0.2816	0.2694
0.9	0.3	0.3253	0.3943	0.3653	0.3451	0.2832	0.3249	0.2836	0.2688
0.9	0.4	0.3313	0.3998	0.3666	0.3394	0.2888	0.3217	0.2865	0.2686
0.9	0.5	0.3325	0.3958	0.3676	0.3629	0.2939	0.3219	0.2838	0.2788
0.9	0.6	0.3461	0.3986	0.3640	0.3569	0.2955	0.3247	0.2928	0.2831
0.9	0.7	0.3284	0.3734	0.3649	0.3396	0.2974	0.3246	0.2938	0.2893
0.9	0.8	0.3256	0.3957	0.3480	0.3328	0.2810	0.3157	0.2812	0.1607
0.9	0.9	0.2453	0.3834	0.3312	0.2314	0.2545	0.3163	0.2812	0.1374
0.9	1	0.2241	0.3915	0.2536	0.2340	0.2583	0.3149	0.2815	0.1451

1	0	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621
1	0.1	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621
1	0.2	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621
1	0.3	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621
1	0.4	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621
1	0.5	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621
1	0.6	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621
1	0.7	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621
1	0.8	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621
1	0.9	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621
1	1	0.3120	0.3764	0.3623	0.3459	0.2776	0.3288	0.2823	0.2621

Table A.3.4 The influence of the parameters α and β on the quality of clustering solutions for Haverford76 and Aassar85 datasets.

Data set		Haverford76				Vassar85			
α	β	Fast Modularity algorithm	Louvain	leading eigenvector	Walktrap	Fast Modularity algorithm	Louvain	leading eigenvector	Walktrap
0	0	0.3007	0.2911	0.2871	0.2403	0.3246	0.3891	0.3454	0.3351
0	0.1	0.2689	0.3274	0.2419	0.2154	0.2747	0.3617	0.2236	0.2700
0	0.2	0.2525	0.3230	0.2066	0.2071	0.2606	0.3693	0.3208	0.1291
0	0.3	0.2437	0.3185	0.2455	0.0724	0.2655	0.3719	0.1108	0.0672
0	0.4	0.1453	0.3122	0.1025	0.0943	0.1537	0.3459	0.2600	0.1277
0	0.5	0.2433	0.2032	0.1034	0.1077	0.2667	0.1040	0.1100	0.1277
0	0.6	0.2068	0.0943	0.1026	0.0714	0.1105	0.0957	0.1105	0.0679
0	0.7	0.1022	0.0934	0.0990	0.0670	0.1564	0.1453	0.1063	0.1270
0	0.8	0.1025	0.0948	0.1007	0.0670	0.1101	0.0957	0.1051	0.0678
0	0.9	0.0938	0.0956	0.0953	0.0689	0.1101	0.0962	0.1053	0.1268
0	1	0.1023	0.0938	0.1008	0.0689	0.1101	0.1008	0.1056	0.1276
0.1	0	0.3035	0.3298	0.2920	0.2448	0.3499	0.3716	0.3622	0.3010
0.1	0.1	0.2683	0.3289	0.2483	0.2749	0.3512	0.3477	0.2095	0.2685
0.1	0.2	0.2391	0.3222	0.2357	0.2190	0.2554	0.3690	0.3114	0.1303
0.1	0.3	0.2540	0.3202	0.2402	0.0756	0.1607	0.3717	0.1338	0.0669
0.1	0.4	0.2427	0.3205	0.2399	0.2143	0.2683	0.3556	0.2524	0.0674
0.1	0.5	0.2206	0.2073	0.1053	0.0974	0.2670	0.3670	0.1084	0.1277
0.1	0.6	0.1468	0.0954	0.1055	0.0677	0.3153	0.3684	0.1104	0.1277
0.1	0.7	0.2057	0.0961	0.1137	0.0648	0.1105	0.0957	0.1106	0.0721
0.1	0.8	0.1018	0.0960	0.0971	0.0671	0.3214	0.3615	0.1093	0.1270
0.1	0.9	0.1025	0.1224	0.1008	0.0704	0.1101	0.1046	0.1051	0.0679
0.1	1	0.0949	0.0956	0.0982	0.0689	0.3470	0.3523	0.1089	0.0665
0.2	0	0.2662	0.3305	0.2955	0.2858	0.2911	0.3913	0.3610	0.3600
0.2	0.1	0.2691	0.3293	0.2542	0.2799	0.3518	0.3506	0.3563	0.3208
0.2	0.2	0.2641	0.3237	0.2340	0.2168	0.2445	0.3669	0.2017	0.1296
0.2	0.3	0.2394	0.3238	0.1094	0.0914	0.2700	0.3559	0.2736	0.1281

0.2	0.4	0.2424	0.3213	0.2659	0.0744	0.3295	0.3722	0.1181	0.0683
0.2	0.5	0.1461	0.3184	0.1056	0.0943	0.1477	0.3452	0.2661	0.1277
0.2	0.6	0.2215	0.0994	0.1094	0.0943	0.1565	0.3588	0.1091	0.1277
0.2	0.7	0.1478	0.0954	0.1070	0.0657	0.3155	0.3686	0.1101	0.1276
0.2	0.8	0.3035	0.2061	0.1021	0.0660	0.1110	0.0972	0.1106	0.1289
0.2	0.9	0.0984	0.0960	0.0987	0.0659	0.1834	0.3659	0.1103	0.1271
0.2	1	0.3135	0.2981	0.1025	0.0704	0.1481	0.0972	0.1067	0.0679
0.3	0	0.3034	0.3305	0.2958	0.2920	0.3227	0.3898	0.3592	0.3221
0.3	0.1	0.2708	0.3296	0.2674	0.2375	0.3511	0.3661	0.3590	0.3364
0.3	0.2	0.2170	0.3218	0.2083	0.2368	0.2258	0.3190	0.1999	0.1294
0.3	0.3	0.2881	0.3247	0.2873	0.2191	0.2634	0.3716	0.2622	0.1281
0.3	0.4	0.2462	0.3232	0.2501	0.0688	0.2906	0.3729	0.1179	0.0682
0.3	0.5	0.2779	0.3216	0.2585	0.0743	0.3207	0.3731	0.1194	0.0681
0.3	0.6	0.2209	0.3134	0.1090	0.0943	0.1564	0.3710	0.1097	0.1277
0.3	0.7	0.1476	0.2418	0.1097	0.0975	0.1569	0.2092	0.1102	0.1272
0.3	0.8	0.1470	0.3137	0.1072	0.0642	0.3438	0.3703	0.2649	0.1277
0.3	0.9	0.2409	0.2047	0.0911	0.0664	0.2894	0.3616	0.1106	0.1271
0.3	1	0.3046	0.3145	0.1042	0.0659	0.1479	0.3458	0.1107	0.1271
0.4	0	0.3052	0.3380	0.2984	0.2900	0.3253	0.3876	0.3556	0.3534
0.4	0.1	0.2708	0.3308	0.2744	0.3002	0.3551	0.3823	0.3341	0.3471
0.4	0.2	0.2762	0.3290	0.2701	0.2445	0.2715	0.3647	0.2836	0.2598
0.4	0.3	0.2655	0.3256	0.2498	0.2161	0.2532	0.3731	0.2945	0.1294
0.4	0.4	0.2391	0.3234	0.2323	0.0914	0.3170	0.3720	0.2755	0.1281
0.4	0.5	0.2453	0.3242	0.2673	0.0721	0.1581	0.3720	0.1284	0.0710
0.4	0.6	0.2832	0.3216	0.2590	0.1219	0.3210	0.3731	0.1369	0.0676
0.4	0.7	0.2165	0.3200	0.1093	0.0974	0.2670	0.3469	0.1106	0.1272
0.4	0.8	0.1477	0.3140	0.1094	0.0974	0.2671	0.3705	0.1111	0.1272
0.4	0.9	0.1459	0.3178	0.1095	0.0974	0.1565	0.3480	0.1117	0.1277
0.4	1	0.3156	0.3163	0.1061	0.0962	0.3469	0.3715	0.1102	0.0703
0.5	0	0.2827	0.3379	0.2976	0.2937	0.3266	0.3884	0.3570	0.3577
0.5	0.1	0.2711	0.3313	0.2824	0.3030	0.3468	0.3845	0.3618	0.3567
0.5	0.2	0.2707	0.3293	0.2958	0.2431	0.3526	0.3651	0.2497	0.2548
0.5	0.3	0.2953	0.3253	0.2737	0.2387	0.2256	0.3326	0.2609	0.1294
0.5	0.4	0.2653	0.3242	0.2663	0.2190	0.2613	0.3748	0.2626	0.2515
0.5	0.5	0.2492	0.3238	0.2520	0.0880	0.2395	0.3730	0.1239	0.0710
0.5	0.6	0.2660	0.3200	0.2643	0.0667	0.3487	0.3728	0.1302	0.0692
0.5	0.7	0.2797	0.3243	0.2768	0.0729	0.3355	0.3731	0.1364	0.0707
0.5	0.8	0.2316	0.3198	0.1240	0.0981	0.2304	0.3713	0.2752	0.1280
0.5	0.9	0.2170	0.3202	0.3028	0.0981	0.1834	0.3717	0.1118	0.1272
0.5	1	0.2155	0.3191	0.2300	0.0981	0.1830	0.3688	0.1120	0.1272
0.6	0	0.3023	0.3375	0.3027	0.3014	0.3009	0.3856	0.3535	0.3396
0.6	0.1	0.2877	0.3296	0.2956	0.2999	0.3496	0.3877	0.3673	0.3710
0.6	0.2	0.3031	0.3283	0.2605	0.2198	0.3500	0.3813	0.3480	0.3274
0.6	0.3	0.2697	0.3296	0.2747	0.2333	0.3046	0.3647	0.3052	0.1301

0.6	0.4	0.2793	0.3263	0.2600	0.2266	0.2292	0.3723	0.2709	0.1294
0.6	0.5	0.2654	0.3255	0.2826	0.2565	0.3376	0.3755	0.2892	0.1294
0.6	0.6	0.2815	0.3236	0.2564	0.0885	0.3195	0.3737	0.2911	0.1282
0.6	0.7	0.2910	0.3200	0.2757	0.0874	0.2500	0.3734	0.1283	0.0700
0.6	0.8	0.2889	0.3198	0.2856	0.0755	0.3186	0.3735	0.1360	0.0716
0.6	0.9	0.2976	0.3201	0.2562	0.1428	0.3497	0.3735	0.1456	0.0703
0.6	1	0.1394	0.1036	0.1142	0.0974	0.1489	0.3726	0.1507	0.1289
0.7	0	0.2715	0.3334	0.3006	0.3015	0.3007	0.3810	0.3505	0.3444
0.7	0.1	0.2979	0.3376	0.3055	0.3091	0.3412	0.3941	0.3699	0.3540
0.7	0.2	0.2983	0.3307	0.2860	0.3045	0.3572	0.3825	0.3704	0.3685
0.7	0.3	0.2704	0.3277	0.2632	0.2173	0.3472	0.3801	0.2489	0.2649
0.7	0.4	0.2701	0.3297	0.2772	0.2290	0.3246	0.3655	0.3111	0.2598
0.7	0.5	0.2480	0.3265	0.2980	0.2009	0.2092	0.3193	0.2665	0.1302
0.7	0.6	0.2485	0.3222	0.2678	0.2438	0.2423	0.3764	0.2893	0.1294
0.7	0.7	0.2661	0.3268	0.2620	0.2812	0.2639	0.3770	0.2928	0.2409
0.7	0.8	0.2849	0.3250	0.2750	0.1062	0.3427	0.3748	0.2913	0.2503
0.7	0.9	0.2893	0.3244	0.2892	0.1058	0.3152	0.3739	0.1354	0.0684
0.7	1	0.3013	0.3249	0.3006	0.0921	0.3508	0.3732	0.1447	0.0694
0.8	0	0.2707	0.3280	0.2932	0.3101	0.3043	0.3866	0.3481	0.3532
0.8	0.1	0.2708	0.3360	0.3001	0.3025	0.3047	0.3866	0.3512	0.3563
0.8	0.2	0.2996	0.3362	0.3058	0.3079	0.3439	0.3933	0.3700	0.3621
0.8	0.3	0.2950	0.3289	0.2981	0.3024	0.3516	0.3826	0.3798	0.3600
0.8	0.4	0.3010	0.3286	0.2701	0.2731	0.3545	0.3806	0.3688	0.3374
0.8	0.5	0.2702	0.3285	0.2717	0.2454	0.3204	0.3795	0.2205	0.2162
0.8	0.6	0.2711	0.3274	0.2696	0.2511	0.3315	0.3825	0.2998	0.2234
0.8	0.7	0.2557	0.3050	0.2927	0.0991	0.1339	0.3167	0.2700	0.1294
0.8	0.8	0.2579	0.3270	0.2974	0.1838	0.1713	0.3421	0.2694	0.1305
0.8	0.9	0.2798	0.3266	0.3093	0.2384	0.2567	0.3753	0.2845	0.1305
0.8	1	0.3053	0.3266	0.3098	0.2293	0.2552	0.3759	0.2853	0.1302
0.9	0	0.2325	0.3311	0.2807	0.3031	0.3038	0.3940	0.3472	0.3575
0.9	0.1	0.2477	0.3377	0.2812	0.3031	0.3036	0.3901	0.3472	0.3445
0.9	0.2	0.2527	0.3376	0.2820	0.2957	0.3023	0.3869	0.3476	0.3445
0.9	0.3	0.2502	0.3322	0.2822	0.2972	0.3020	0.3929	0.3468	0.3453
0.9	0.4	0.2668	0.3376	0.2880	0.2993	0.3009	0.3865	0.3521	0.3672
0.9	0.5	0.2737	0.3306	0.2974	0.3013	0.3042	0.3882	0.3632	0.3706
0.9	0.6	0.2731	0.3292	0.3011	0.2980	0.3494	0.3828	0.3791	0.3799
0.9	0.7	0.2702	0.3303	0.2855	0.3059	0.3542	0.3822	0.3848	0.3796
0.9	0.8	0.3026	0.3279	0.2622	0.2664	0.3464	0.3818	0.3872	0.3652
0.9	0.9	0.3018	0.3277	0.2691	0.2308	0.3368	0.3797	0.2659	0.2594
0.9	1	0.2677	0.3276	0.2695	0.2683	0.3383	0.3788	0.2680	0.2525
1	0	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443
1	0.1	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443
1	0.2	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443
1	0.3	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443

1	0.4	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443
1	0.5	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443
1	0.6	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443
1	0.7	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443
1	0.8	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443
1	0.9	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443
1	1	0.2769	0.3373	0.2823	0.3000	0.3138	0.3940	0.3472	0.3443

Table A.3.5 Attribute weights vs. missing edges for Caltech36 dataset

% missing edges	student/faculty	Gender	major	second major/minor	dormitory	year	High school
0	0.4695	0.3102	0.0924	0.0002	0.2193	0.2195	0.0112
5	0.4677	0.3166	0.0864	0.0008	0.2146	0.2021	0.0059
10	0.4522	0.3065	0.0864	0.0020	0.2006	0.2085	0.0050
15	0.4293	0.3166	0.0846	0.0042	0.1994	0.1820	0.0105
20	0.4564	0.3110	0.0818	0.0016	0.2027	0.1966	0.0063
25	0.4499	0.3209	0.0766	0.0007	0.1978	0.2014	0.0097
30	0.4604	0.3208	0.0843	0.0040	0.2035	0.2122	0.0098
35	0.4750	0.3325	0.0744	0.0010	0.2040	0.2312	0.0050
40	0.4523	0.3119	0.0775	0.0022	0.2260	0.2066	0.0039
45	0.4418	0.3108	0.1031	0.0057	0.2134	0.2229	0.0074
50	0.4451	0.3093	0.0892	0.0007	0.2280	0.2279	0.0078

Table A.3.6 Attribute weights vs. missing edges for Reed98 dataset

% missing edges	student/faculty	Gender	major	second major/minor	dormitory	year	High school
0	0.5840	0.3180	0.0761	0.0064	0.0976	0.2698	0.0143
5	0.5808	0.2931	0.0567	0.0048	0.0894	0.2667	0.0100
10	0.5824	0.3141	0.0596	0.0061	0.0892	0.2575	0.0145
15	0.5638	0.2920	0.0619	0.0044	0.0818	0.2567	0.0142
20	0.5836	0.2997	0.0498	0.0049	0.0806	0.2875	0.0136
25	0.5670	0.3065	0.0554	0.0041	0.0836	0.2501	0.0099
30	0.5794	0.2940	0.0685	0.0028	0.0900	0.2580	0.0123
35	0.5638	0.2777	0.0615	0.0040	0.0823	0.2671	0.0045
40	0.5569	0.2928	0.0512	0.0059	0.0746	0.2600	0.0101
45	0.5208	0.2790	0.0514	0.0053	0.0761	0.2422	0.0138
50	0.5391	0.2917	0.0529	0.0062	0.0846	0.2543	0.0044

Table A.3.7 Attribute weights vs. missing edges for Haverford76 dataset

% missing edges	student/faculty	Gender	major	second major/minor	dormitory	year	High school
0	0.5815	0.3794	0.0387	0.0084	0.1582	0.3077	0.0079
5	0.5995	0.3854	0.0344	0.0107	0.1541	0.3254	0.0097
10	0.5950	0.3740	0.0323	0.0098	0.1610	0.3255	0.0101
15	0.6025	0.3791	0.0423	0.0065	0.1444	0.3213	0.0051
20	0.5966	0.3660	0.0355	0.0110	0.1619	0.3124	0.0049

25	0.5883	0.3640	0.0376	0.0127	0.1724	0.3096	0.0062
30	0.5927	0.3716	0.0362	0.0081	0.1563	0.3202	0.0047
35	0.5839	0.3411	0.0386	0.0093	0.1692	0.3195	0.0057
40	0.6154	0.3741	0.0461	0.0117	0.1740	0.3460	0.0066
45	0.5775	0.3455	0.0450	0.0136	0.1680	0.3031	0.0056
50	0.5732	0.3594	0.0444	0.0118	0.1395	0.3150	0.0063

Table A.3.8 Attribute weights vs. missing edges for Vassar85 dataset.

% missing edges	student/faculty	Gender	major	second major/minor	dormitory	year	High school
0	0.6188	0.3457	0.0442	0.0073	0.1964	0.3843	0.0102
5	0.6337	0.3534	0.0420	0.0090	0.2058	0.3818	0.0095
10	0.6293	0.3544	0.0438	0.0084	0.1979	0.3910	0.0094
15	0.6179	0.3441	0.0392	0.0069	0.1936	0.3882	0.0071
20	0.6264	0.3654	0.0444	0.0074	0.2066	0.3847	0.0105
25	0.6215	0.3406	0.0413	0.0093	0.2076	0.3796	0.0104
30	0.6066	0.3479	0.0433	0.0072	0.1983	0.3710	0.0090
35	0.6142	0.3463	0.0405	0.0072	0.1957	0.3709	0.0077
40	0.6105	0.3374	0.0456	0.0075	0.1934	0.3828	0.0076
45	0.6064	0.3614	0.0450	0.0082	0.1937	0.3823	0.0095
50	0.5894	0.3412	0.0408	0.0062	0.1831	0.3642	0.0089

Table A.3.9 Number of community clusters vs. Missing edges for Caltech36 and Reed98 datasets

Data set	Caltech36								Reed98							
	FA	Hybrid-FA	LA	Hybrid-LEA	LEA	Hybrid-LEA	WA	Hybrid-WA	FA	Hybrid-FA	LA	Hybrid-LEA	LEA	Hybrid-LEA	WA	Hybrid-WA
0	9	4	10	7	12	4	72	8	5	3	6	4	7	3	78	6
5	11	4	12	8	11	4	71	6	7	3	8	4	6	3	65	6
10	12	4	13	8	11	5	77	6	11	3	9	4	6	3	72	5
15	16	4	17	7	13	5	79	6	12	3	11	4	5	3	76	6
20	20	4	19	8	9	4	78	6	15	3	13	4	6	3	78	7
25	22	4	22	7	10	4	87	6	17	3	17	4	4	4	71	8
30	24	4	25	7	11	4	89	7	18	3	18	5	5	3	77	5
35	26	4	26	7	10	4	88	7	26	4	24	4	6	3	90	7
40	32	4	33	8	11	5	98	8	27	4	27	4	6	4	96	6
45	33	4	34	7	10	5	103	8	28	4	28	4	7	3	102	6
50	39	4	39	8	10	4	104	8	36	4	34	4	8	4	114	6

Table A.3.10 Number of community clusters vs. missing edges for Haverford76 and Vassar85 datasets

Data set	Haverford76								Vassar85							
	FA	Hybrid-FA	LA	Hybrid-LA	LEA	Hybrid-LEA	WA	Hybrid-WA	FA	Hybrid-FA	LA	Hybrid-LA	LEA	Hybrid-LEA	WA	Hybrid-WA
0	6	3	5	4	7	3	28	4	5	3	6	5	7	3	60	4
5	5	3	7	4	7	3	37	5	7	3	8	5	7	3	55	4
10	6	3	7	4	8	3	40	4	9	3	10	5	6	3	55	5
15	8	3	9	4	8	3	42	5	12	3	13	5	7	3	67	5
20	9	3	11	4	8	3	40	4	13	3	15	5	6	3	70	4
25	10	3	11	4	8	3	43	5	16	3	17	5	8	4	73	4
30	12	3	13	4	8	3	46	4	17	3	18	5	6	4	84	4
35	14	3	14	4	10	3	49	5	21	3	23	5	5	3	89	4
40	15	3	15	4	7	3	53	5	26	3	27	5	6	4	97	4
45	18	4	18	4	10	3	55	5	33	3	32	5	8	3	102	4
50	20	3	21	4	8	4	61	6	34	3	35	5	7	3	105	4

Table A.3.11 Community size vs. missing edges for Caltech36 and Reed98 datasets

Data set	Caltech36								Reed98							
	FA	Hybrid-FA	LA	Hybrid-LA	LEA	Hybrid-LEA	WA	Hybrid-WA	FA	Hybrid-FA	LA	Hybrid-LA	LEA	Hybrid-LEA	WA	Hybrid-WA
0	85	192	77	110	64	185	11	100	192	313	160	221	137	321	12	162
5	71	185	63	101	75	181	11	122	161	305	131	226	180	321	15	169
10	64	192	60	102	75	169	10	125	92	300	113	231	189	321	14	196
15	51	188	45	104	63	173	10	127	83	297	89	221	210	337	13	176
20	40	182	40	102	89	185	10	121	68	297	78	236	191	313	13	167
25	36	181	35	108	82	192	9	125	57	313	58	236	253	284	14	146
30	32	185	31	105	82	182	9	118	54	313	54	212	213	305	13	199
35	30	191	31	106	77	178	9	118	38	281	40	231	201	329	11	175
40	24	195	24	103	77	165	8	103	36	292	36	241	174	306	10	182
45	24	195	23	106	78	172	8	98	35	273	35	236	184	350	10	177
50	20	183	20	100	83	188	7	101	27	281	29	221	158	287	8	164

Table A.3.12 Community size vs. missing edges for Haverford76 and Vassar85 datasets

Data set	Haverford76								Vassar85							
	% missing edges	FA	Hybrid-FA	LA	Hybrid-LA	LEA	Hybrid-LEA	WA	Hybrid-WA	FA	Hybrid-FA	LA	Hybrid-LA	LEA	Hybrid-LEA	WA
0	241	482	289	362	207	482	52	393	614	1023	511	614	438	1023	51	736
5	296	446	222	362	211	482	39	311	488	1023	396	614	480	1023	56	752
10	251	470	215	362	185	482	37	354	374	1023	316	629	567	997	56	709
15	196	499	170	362	190	482	35	345	269	1023	236	614	436	972	47	721
20	162	506	140	354	213	482	36	369	250	997	224	614	592	946	44	736
25	143	506	132	362	224	470	34	342	200	1023	188	614	447	895	42	736
30	123	458	115	362	237	482	32	376	181	1023	174	614	597	869	37	767
35	111	446	103	362	178	482	30	352	148	1023	140	614	678	972	35	752
40	99	458	98	362	282	482	28	299	123	1023	116	614	572	895	32	752
45	83	431	81	362	217	446	27	275	96	997	97	614	446	946	30	721
50	74	470	72	362	240	410	24	251	90	1023	89	614	553	1023	30	782

Table A.3.13 Modularity index vs. missing edges for Caltech36 dataset

% missing edges	FA	Hybrid-FA	LA	Hybrid-LA	LEA	Hybrid-LEA	WA	Hybrid-WA
0	0.3120	0.3174	0.3764	0.3935	0.3623	0.3445	0.3459	0.3133
5	0.3224	0.3206	0.3877	0.3963	0.3602	0.3454	0.3414	0.3105
10	0.3238	0.3177	0.3952	0.3932	0.3627	0.3411	0.3446	0.3135
15	0.3246	0.3098	0.3897	0.3961	0.3573	0.3358	0.3412	0.3041
20	0.3344	0.3033	0.3900	0.3910	0.3529	0.3217	0.3473	0.2923
25	0.3134	0.3074	0.3891	0.3916	0.3562	0.3052	0.3440	0.2833
30	0.3255	0.3119	0.3912	0.3900	0.3513	0.2914	0.3403	0.2784
35	0.3233	0.2994	0.3890	0.3893	0.3507	0.2838	0.3443	0.2686
40	0.3208	0.3012	0.3889	0.3853	0.3433	0.2669	0.3445	0.2658
45	0.3207	0.3000	0.3873	0.3834	0.3451	0.2655	0.3341	0.2542
50	0.3177	0.2938	0.3805	0.3815	0.3420	0.2372	0.3362	0.2369

Table A.3.14 Modularity index vs. missing edges for Reed98 dataset

% missing edges	FA	Hybrid-FA	LA	Hybrid-LA	LEA	Hybrid-LEA	WA	Hybrid-WA
0	0.2776	0.2423	0.3288	0.3199	0.2823	0.2785	0.2621	0.2411
5	0.2711	0.2470	0.3214	0.3170	0.2858	0.2775	0.2617	0.2358
10	0.2768	0.2473	0.3229	0.3142	0.2815	0.2739	0.2640	0.2223
15	0.2731	0.2525	0.3190	0.3134	0.2800	0.2724	0.2649	0.2145
20	0.2641	0.2532	0.3157	0.3153	0.2771	0.2662	0.2629	0.2152
25	0.2649	0.2481	0.3099	0.3156	0.2737	0.2658	0.2678	0.2131
30	0.2729	0.2430	0.3122	0.3104	0.2758	0.2583	0.2615	0.2075
35	0.2814	0.2422	0.3086	0.3060	0.2726	0.2539	0.2515	0.1990
40	0.2702	0.2457	0.3027	0.3073	0.2641	0.2443	0.2615	0.1938
45	0.2696	0.2502	0.3014	0.3044	0.2686	0.2376	0.2504	0.1830
50	0.2749	0.2439	0.2928	0.2986	0.2629	0.2400	0.2453	0.1747

Table A.3.15 Modularity index vs. missing edges for Haverford76 dataset

% missing edges	FA	Hybrid-FA	LA	Hybrid-LA	LEA	Hybrid-LEA	WA	Hybrid-WA
0	0.2769	0.3010	0.3373	0.3293	0.2823	0.2736	0.3000	0.2573
5	0.2706	0.2818	0.3324	0.3291	0.2811	0.2714	0.3024	0.2786
10	0.2692	0.2706	0.3324	0.3285	0.2817	0.2699	0.2994	0.2701
15	0.2651	0.2785	0.3342	0.3291	0.2814	0.2685	0.2982	0.2764
20	0.2694	0.2757	0.3325	0.3259	0.2761	0.2657	0.2991	0.2641
25	0.2709	0.2773	0.3301	0.3283	0.2772	0.2630	0.2983	0.2694
30	0.2796	0.2753	0.3291	0.3278	0.2720	0.2615	0.2965	0.2584
35	0.2811	0.2835	0.3265	0.3249	0.2756	0.2584	0.2958	0.2653
40	0.2813	0.2761	0.3262	0.3275	0.2682	0.2570	0.2965	0.2607
45	0.2794	0.2740	0.3236	0.3272	0.2692	0.2696	0.2896	0.2572
50	0.2830	0.2809	0.3214	0.3261	0.2685	0.2836	0.2951	0.2531

Table A.3.16 Modularity index vs. missing edges for Vassar85 dataset

% missing edges	FA	Hybrid-FA	LA	Hybrid-LA	LEA	Hybrid-LEA	WA	Hybrid-WA
0	0.3138	0.3354	0.3940	0.3809	0.3472	0.3307	0.3443	0.2688
5	0.3176	0.3405	0.3889	0.3807	0.3470	0.3292	0.3498	0.2638
10	0.3166	0.3420	0.3878	0.3786	0.3499	0.3257	0.3474	0.2580
15	0.3156	0.3458	0.3841	0.3798	0.3478	0.3220	0.3457	0.2411
20	0.3182	0.3414	0.3869	0.3795	0.3487	0.3188	0.3474	0.2650
25	0.3250	0.3432	0.3843	0.3790	0.3492	0.3146	0.3518	0.2504
30	0.3240	0.3440	0.3865	0.3784	0.3480	0.3068	0.3475	0.2379
35	0.3172	0.3449	0.3844	0.3777	0.3463	0.3081	0.3467	0.2435
40	0.3274	0.3412	0.3799	0.3773	0.3442	0.2968	0.3437	0.2447
45	0.3237	0.3455	0.3823	0.3765	0.3442	0.2921	0.3429	0.2400
50	0.3286	0.3417	0.3805	0.3762	0.3437	0.2901	0.3412	0.2231

Appendix B: Permission to Reuse IEEE Material



RightsLink®

Home

Create Account

Help



Title: Decentralized Iterative Community Clustering Approach (DICCA)
Conference Proceedings: Personal, Indoor, and Mobile Radio Communications (PIMRC), 2017 IEEE 28th Annual International Symposium on
Author: Amhmed Bhih
Publisher: IEEE
Date: Oct. 2017
Copyright © 2017, IEEE

LOGIN
If you're a [copyright.com](#) user, you can login to RightsLink using your [copyright.com](#) credentials. Already a [RightsLink](#) user or want to [learn more?](#)

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW

Copyright © 2018 [Copyright Clearance Center, Inc.](#) All Rights Reserved. [Privacy statement.](#) [Terms and Conditions.](#) Comments? We would like to hear from you. E-mail us at customer@copyright.com