Community Structure in Complex Networks

by

Shiva Zamani Gharaghooshi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

There is no shortage of community mining algorithms for discovering structure in complex information networks; most with unique advantages, however, all with drawbacks, including efficiency, correctness, resolution limit, and field of view limit. We introduce a novel efficient approach for community detection based on the notion of edge strength and a formal definition of the notion of community. We consider that there are two types of links in a graph: links that run between communities whose removal may divide the graph into disconnected subgraphs that we refer to as weak links, and links that are inside communities whose removal would not change graph connectivity, that we refer to as strong links. We put forward a new objective function, called SIWO (for Strong In, Weak Out), which encourages adding strong links to the communities while avoiding weak links. Optimizing this function allows us to discover dense subgraphs from which qualified communities that comply with the definition are extracted. This process allows us to effectively discover communities in social networks without the resolution and field of view limit problems some popular approaches, considered the state of the art, suffer from. Moreover, time complexity of our method, which piggybacks on the optimization of Louvain, known to be very efficient, is linear in the number of edges. We experimentally demonstrate the effectiveness of our approach on various real and artificial datasets with large and small communities to show the resilience to the resolution and field of view limits, including large size networks having million edges.

Preface

This thesis is an original work by Shiva Zamani Gharaghooshi. Part of this thesis were presented as a poster at The ACM Canadian Celebration of Women in Computing 2017 (Montreal, Canada).

Dedicated to

My parents,

Shahab, my brother,

And

Alireza, my love

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Osmar Zaïane and Christine Largeron for their invaluable insights, encouragement and patience. They helped me tremendously through my research, gave me the opportunity to express my ideas freely, and guided me to find the right direction.

I am further thankful to my collaborator Chang Liu. Many thanks go to my teammates from the Meerkat project: Abhimanyu Panwar, Sankalp Prabhakar, Talat Iqbal Syed, and Chi Zhang.

Last but not the least, I would like to thank my parents and my brother for their love and support. And finally, I would like to thank my love, Alireza Noamani, for his love and endurance, none of this could have been possible without his support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Complex Networks

Complex networks are present in a variety of application domains, for example co-authorship networks, protein-protein interaction networks, social networks, contact networks, hyperlink networks of web pages and phone call networks. It has been shown that these real networks deviate from the random networks and share common properties such as power law degree distribution [4], high clustering coefficient [48], assortative mixing [33] and presence of community structure [34], which are described in the following.

Power law degree distribution:

Node degrees in complex networks often follow a power law distribution; the vast majority of nodes have a low degree, whereas few have an extremely high degree. Figure 1.1 presents the degree distribution of three real networks (polblogs [1], eurosis [49] and email [52][29]).

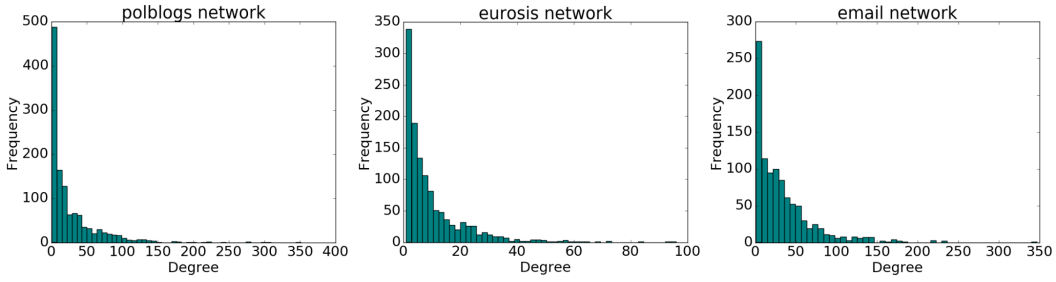

Figure 1.1: Degree distribution of three real networks

High clustering coefficient:

Clustering coefficient measures the fraction of connected neighbours per each node. A complex network tends to have a higher average clustering coefficient compared to random networks. The average clustering coefficients of some of the real networks are presented in Table 1.1.

Assortative mixing:

Assortative mixing is the tendency for high-degree nodes to connect with high-degree nodes. Newman [33] proposed an assortativity coefficient r based on standard Pearson correlation coefficient:

$$r = \frac{\sum_{xy} xy(l_{xy} - s_x e_y)}{\sigma_s \sigma_e} \tag{1.1}$$

where l_{xy} is the fraction of edges that connect nodes of degree x to nodes of degree y. s_x and e_y are the fraction of edges that start and end at nodes with degrees x and y respectively. σ_s and σ_e are the standard deviations of s and e. This measure lies in the range of [-1,1], with $r = 1$ denoting complete assortativity and $r = -1$ denoting complete disassortativity. Newman [33] showed that social networks tend to have positive assortativity coefficients while biological networks tend to have negative assortativity coefficients. The assortativity coefficients of some of the real networks are presented in Table 1.1.

Community structure:

A community is often defined as a dense subgraph that is sparsely connected to the rest of the graph. The members of a community are strongly tied with each other for a reason; they share the same characteristics (social networks), they peruse the same interests (co-authorship and criminal networks), they participate in the same functional modules (biological networks) etc. While it is not possible to study each node individually in the large networks, by detecting the communities we can split the network into smaller groups of similar nodes and each group can be studied separately and therefore it is important to design community detection algorithms. Moreover, due to the homophily [31] and social influence effect [2][12], the members of the same community affect and are affected by each other and detecting the communities

Table 1.1: Properties of real networks

	Karate [53]	citeseer [46]	email [52][29]	eurosis [49]	football [19]	polblogs [1]	polbooks [1]
nodes	34	3312	1005	1218	115	1222	105
edges	78	4660	16706	5999	613	16717	441
communities	2	-	-	-	12	2	3
diameter	5	-	-	10	4	8	7
average shortest path	2.41	-	-	4.04	2.51	2.74	3.08
degree assortativity	-0.48	0.05	-0.01	-0.02	0.16	-0.22	-0.13
avg clustering coef	0.57	0.14	0.4	0.33	0.4	0.32	0.49

enables us to predict the behaviour of the nodes. For instance, customers of an online shopping service who belong to the same community (have the same purchasing behaviour) are likely to buy the same products that other members of their community are interested in.

Many algorithms have been proposed to find community structures in networks. We can mention methods which aim to partition the graph into subgroups of nodes such as the spectral bisection method [6] or Kernighan-Lin algorithm [24]. These algorithms require prior information about the number/size of communities and cannot be applied to networks where we do not have such information in advance. There are also divisive hierarchical algorithms which provide a sequence of partitions by repeatedly removing the edges that run between communities such as the divisive algorithm based on edge betweenness of Newman and Girvan [35]. The computational cost of this algorithm grows exponentially with the number of edges in the network and cannot be applied to large networks. Another class of community detection algorithms are those which aim at maximizing an objective function such as Q-modularity [35]. Q-modularity is a very well-known objective function for community detection tasks. However, Fortunato and Barthélemy [17] showed that it suffers from the resolution limit which means that by optimizing Q-modularity, communities that are smaller than a scale cannot be detected.

[1] http://www.orgnet.com

The map equation [43] is another objective function which is based on the intuition that when a random walker enters a community, it will stay within the community for a long time before leaving it. Schaub *et al.* [45] showed that the map equation is highly sensitive to the field of view limit which means that communities with large diameter cannot be detected. We can also cite seed-centric approaches which first identify representatives of communities and then detect local communities around the representatives such as Top-leaders [25] and LICOD [50]. These algorithms require the number of communities or rely on some other input parameters that need to be tuned. On the other hand, propagation-based algorithms do not rely on any parameter and are very fast. However, these algorithms are non-deterministic and highly sensitive to the order in which nodes are processed and might result in undesirable solutions such as placing all nodes in a single community.

1.2 Thesis Statements

This thesis starts with identifying weak links (links that run between true communities) and strong links (links that are inside true communities). The thesis hypothesis here is that:

Thesis Statement 1. We can differentiate between weak links and strong links by considering the strength of connections to neighbours.

There have been some works [35][40] toward identifying the weak links in networks. However, to the best of our knowledge, there have been no studies done on creating an objective function for community detection tasks based on the notion of weak and strong links. Hence the second thesis hypothesis is:

Thesis Statement 2. An objective function can be created for detecting communities based on the notion of weak and strong links which is not sensitive to the resolution problems some popular objective functions suffer from.

Thesis Statement 3. A general community detection framework can be created such that it can be used to optimize any objective function and it is able to ensure that all the detected communities comply with the definition of community.

1.3 Thesis Objectives

The goal of this study is to develop an accurate, efficient community detection algorithm that does not rely on any parameter tuning and also does not suffer from the resolution limit and the field of view limit (the limits that some state of the art algorithms suffer from). To address this challenge, we propose an objective function based on strength of connections to neighbours and also a formal definition of the notion of community. We optimize this function in a general framework which can be also applied to other objective functions. After optimizing the objective function, we use the definition of community to lead the process of community detection. This step ensures that the detected communities comply with the definition of community. This framework also allows to change the definition of community without changing the objective function (there is no globally accepted definition for a community).

1.4 Thesis Outline

Chapter 2 reviews different classes of community detection algorithms and a set of internal and external measures for evaluation of community detection algorithms. Chapter 3 introduces the notion of weak and strong links and proposes a novel objective function **S**trong **I**n, **W**eak **O**ut (SIWO) for detecting communities; a formal definition of community; and a general framework in which our objective function is optimized and the quality of the detected communities is verified. Chapter 4 presents the comparison of SIWO and its contenders on real and synthetic networks with both large and small communities. Chapter 5 summarizes the key contributions, and describes future perspectives.

Chapter 2

Background and Related Work

2.1 Overview of Community Mining Algorithms

Many algorithms have been proposed to find community structures in networks. In this study, we focus on algorithms that aim to detect disjoint communities in undirected networks. These algorithms can be broadly classified into five categories: graph partitioning, divisive hierarchical clustering, agglomerative hierarchical clustering, seed-centric approaches and propagation-based approaches, which are described in the following.

2.1.1 Graph Partitioning

This approach aims to partition a given network into k disjoint clusters with predefined sizes. We can mention methods such as the Kernighan-Lin algorithm [24] or spectral bisection method [6]. The Kernighan-Lin algorithm is inspired by the problem of placing electronic components on circuit boards such that the connections between components in different boards is minimal. This algorithm tries to maximize the objective function Q which is the difference between the number of edges inside the modules and the number of edges that run between the modules. It starts by dividing the network into two modules (A and B) of predefined sizes. Afterwards, it swaps two subsets of elements between A and B that results in the maximum gain in Q. This procedure is repeated until there is no more gain in Q. The same partitioning method can be applied on any of the modules in the network until the network is partitioned into k disjoint clusters.

The spectral bisection method aims at minimizing the number of edges running between the communities (cut size) [16]. Suppose that we have a partition of graph G that splits the graph into two parts C_1 and C_2. We can present this partition as a vector P that assigns each node i to one partition or the other:

$$P_i = \begin{cases} +1 & \text{if } i \in C_1 \\ -1 & \text{otherwise} \end{cases} \tag{2.1}$$

Then the cut size can be written as:

$$\frac{1}{4} P^T L(G) P \tag{2.2}$$

where L is the Laplacian matrix. Therefore, in order to minimize the cut size, we need to find the partition vector P such that the quantity given in Equation 2.2 is minimized. It can be shown that the vector y that minimizes this quantity is actually the eigenvector v that corresponds to the second smallest eigenvalue of L. Finally, the partition can be determined using the signs of the components of v.

The drawback of these algorithms is that we need to know the number of communities or their sizes before we do the community mining which is not always the case so we can use these algorithms for a limited number of applications such as placing the electronic components on circuit boards.

2.1.2 Divisive Hierarchical Clustering

Divisive clustering algorithms start by placing all the nodes in the same community. Then they repeatedly detect and remove the edges than lie between the true communities. At some point, the graph will be divided into two parts and again they remove edges to get smaller and smaller communities. Finally, we have a hierarchy of communities and they use a quality measure to choose the best partitioning between these levels; Figure 2.1 gives a visualized example.

The most well-known divisive algorithm is proposed by Newman and Girvan [35]. They introduced a measure, edge betweenness, which is the number of shortest path that pass through a particular edge e_{ij} and has a large value

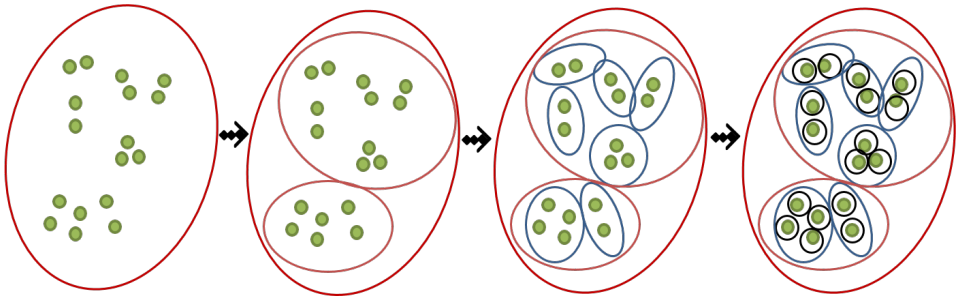

Figure 2.1: Communities resulting from a divisive hierarchical clustering algorithm

for the edges that lie between the true communities. Their algorithm starts by putting all the nodes in a single community and then by iteratively removing the edges with the largest betweenness, they split the network into smaller and smaller communities. This algorithm runs in $O(nm^2)$ where n and m are the number of nodes and edges respectively, which makes it not applicable to large networks. Later, Radicchi *et al.* [40] proposed another measure, edge clustering coefficient, which is the ratio of the number of triangles built on a particular edge e_{ij} over the maximum possible such triangles and can be given as follows:

$$ECC(e_{ij}) = \frac{|\{h : h \in V, e_{ih} \in E, e_{jh} \in E\}|}{\min(d_i - 1, d_j - 1)} \tag{2.3}$$

where V is the set of the nodes in the network, E is the set of edges and d_i is the degree of node i. This measure has a small value for the edges that lie between the communities. Their algorithm follows the same divisive algorithm by Girvan and Newman [19] except that they remove the edges with the minimum edge clustering coefficient in each step and it runs in $O(m^2)$.

The drawback of these algorithms is that they need to recalculate a measure (edge betweenness or edge clustering coefficient) in every iteration to identify the links that lie between the true communities which is very time consuming.

2.1.3 Agglomerative Hierarchical Clustering

This approach starts by placing each node in its own community. Then it repeatedly merges the communities which result in maximum gain in the objective function. Finally, they stop merging the communities when the objective

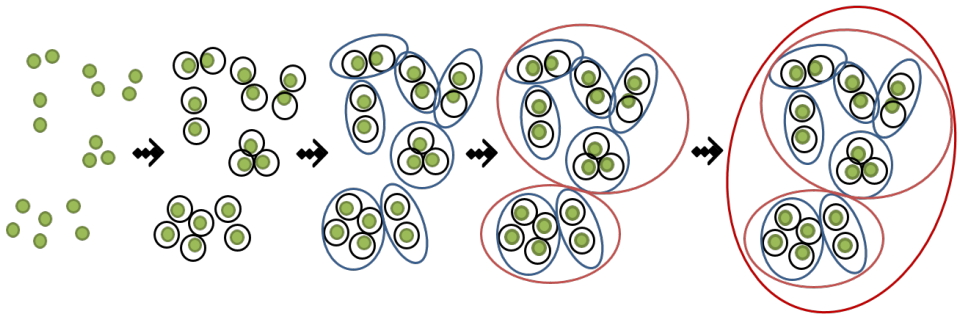

Figure 2.2: Communities resulting from an agglomerative hierarchical clustering algorithm

function has reached its local maxima; Figure 2.2 gives a visualized example.

The most popular objective function is the Q-modularity [35] which can be given as follows:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta(c_i, c_j) \tag{2.4}$$

where A denotes the adjacency matrix, m is the number of edges, d_i and c_i are respectively the degree and the community of node i and $\delta(x, y)$ is the Kronecker function equal to 1 if $x = y$ and 0 otherwise. This criterion compares the number of edges inside communities with the expected number of edges that would reside in the communities under the null hypothesis *i.e.* if edges were randomly placed in the network under the condition that the degree of every node remains unchanged.

Newman [36] proposed a greedy agglomerative algorithm to maximize Q-modularity. At the beginning of this algorithm, each node is placed in its own community and then they repeatedly merge the two communities which result in the maximum gain in Q-modularity. This algorithm runs in $O(n(n+m))$. Later, Clauset *et al.* [11] improved Newman's algorithm by utilizing efficient data structures and reduced the running time of the algorithm to $O(n \log^2 n)$ for sparse networks. Blondel *et al.* [7] proposed the fastest algorithm (linear time complexity) to optimize Q-modularity named Louvain. Their method is an agglomerative clustering algorithm with two main phases that are repeated iteratively until no further improvement in Q-modularity can be achieved. In the first phase, each node is assigned to a unique community. Afterwards,

they are placed in a random sequential order and then at each time one node is moved to one of its neighbouring communities that results in the maximum gain in Q-modularity. If no gain can be achieved by moving a particular node into its neighbouring communities, then the node stays in its own community. In the second phase, a new weighted graph is created in which each node corresponds to a community detected in the first phase. Each edge in the new graph is assigned a weight equal to the sum of the weights of edges between the nodes in the corresponding communities. These two phases are iteratively repeated until no further improvement in Q-modularity can be achieved.

Rosvall and Bergstrom [44] proposed another objective function for community detection tasks named the map equation. They build this function based on the intuition that when a random walker enters a community, it will stay within the community for a long time before leaving it. They use a two-level description method to describe random routes. In this method, modules are assigned with unique bitstreams. Nodes are also assigned with bitstreams such that the nodes within the same module have different bitstreams while nodes in different modules can have the same bitstreams. The description code for each route is constructed as follows: once the random walker enters a module, the bitstream of the that module is added to the description code of the route. It is then followed by the bitstreams of the next nodes in the walk that are in the current module. When the random walker leaves the current module, a special bitstream named the exit code, is added to the description. Using this method, each route in the network can be described with a unique description code. The more the random walker switches between the modules, the longer the description code is. However, if the modules correspond to the real communities in the network, the random walker would not switch between modules very often and the description code length would be relatively small. The map equation is the average description length of a single step. Therefore, minimizing the map equation is equivalent to minimizing the expected description length of a random walk. Rosvall and Bergstrom[44] proposed an extension of Louvain algorithm to minimize the map equation named Infomap. This algorithm runs in $O(m)$.

Pons and Latapy [38] also used random walks to build a distance measure between a node and its community. The more a particular node is structurally similar to the rest of the nodes in its community, the smaller the distance between the node and its community. This measure is used in a hierarchical agglomerative clustering algorithm to detect communities. Initially, each node is placed in its own community. Subsequently, in each step two communities will be chosen based on the distance measure to be merged together. After $n-1$ steps, all the nodes will be placed in a single community. Finally, they will use a quality measure such as Q-modularity to choose the best partition among different partitions which were created during the execution of the algorithm. This algorithm runs in time $O(n^2 \log n)$.

The drawback of these algorithms is that they may not be able to detect communities that are smaller than a certain scale. This problem is known as the resolution limit [17]. Another problem is that they may not be able to detect communities with a large diameter which is known as the field of view limit [45]. Q-modularity suffers from the resolution limit and the map equation suffers from the field of view limit.

2.1.4 Seed-centric Approaches

The algorithms that fall under this class have three main steps [22]: 1- detect seeds/leaders; 2- detect local community around each seed; 3- partition the network into communities based on the local communities detected in Step 2. Here, we detail two of the algorithms that follow this approach: Top-leaders [25] and LICOD [50].

The core of Top-leaders follows K-means clustering algorithm. Initially, a predefined number (k) of nodes are selected as representative of communities (leaders of communities). In the second step, each node x that is not a leader, is assigned to the leader that has the most common neighbours with x considering a neighbourhood of depth d. The neighbourhood depth starts at 1. If two or more leaders tie for the largest number of common neighbours, the depth of the neighbourhood is increased by 1. In case that the node is not assigned to any leader after reaching the neighbourhood depth threshold, that node is

labeled as an outlier. They repeat this process for all the nodes in the network. In Step 3, for each community c they select the node with the largest degree centrality within community c as the new leader of that community. The degree centrality of node i can be given as follows:

$$dc_i = \frac{N_i^{c_i}}{|c_i| - 1} \tag{2.5}$$

where c_i is the community of node i and $N_i^{c_i}$ is the set of neighbours of node i that are also in community c_i. Once a new leader is selected for each community, the second and the third steps are repeated iteratively until there is no change in the set of leaders.

In contrast to Top-leaders, LICOD does not require the number of communities as input. This algorithm starts by computing the node centralities. Afterwards, each node that has a centrality greater than or equal to γ percent of its neighbours centralities, is labeled as a leader. Then, each leader is assigned to a unique community. If the ratio of the common neighbours to the total number of neighbours between two leaders l_1 and l_2 exceeds a given threshold, they are assigned to the same community. Subsequently, a list of community membership preference is created for each node based on membership degree. The membership degree of node x to a community c is the inverse of the minimal shortest path between x and one of the leaders of community c and can be given as follows:

$$membership_x(c) = \frac{1}{\min\limits_{l \in leaders(c)} ShortestPath(x, l) + 1} \tag{2.6}$$

Then, the membership preference list of each node is adjusted by aggregating its initial list with the initial membership preference list of its direct neighbours. Finally, each node is assigned to the top-ranked communities in its preference list.

The drawback of these algorithms is that they either require the number of leaders or a threshold to identify the leaders and we may not have these information before we detect the communities.

2.1.5 Propagation-based Approaches

Raghavan *et al.* [41] proposed an efficient algorithm based on label propagation (LPA). This algorithm starts by assigning a unique label to each node in the network. Afterwards, nodes will be placed in a random order and their labels will be updated one node at a time. Each node takes the label of the majority of its neighbours. This step is iteratively repeated until approaching a fixed label for each node. Finally, the nodes with the same label are grouped together and placed into communities. This algorithm does not require any external parameter setting and runs in time $O(m + n)$.

One of the drawbacks of LPA is that it can result in undesirable solutions such as placing all the nodes in the same community [5]. Barber and Clark [5] proposed an extension of LPA named LPAm that can avoid these undesirable solutions. They formulate the LPA approach as maximizing the number of edges that connect nodes with the same labels (H) which can be given as follows:

$$H = \frac{1}{2} \sum_{i,j} A_{ij} \delta(L_i, L_j) \tag{2.7}$$

where A denotes the adjacency matrix, L_i is the label of node i and $\delta(x, y)$ is the Kronecker function equal to 1 if $x = y$ and 0 otherwise. They add a penalty term (P) to this objective function in order to avoid the undesirable solutions and they create a new objective function H':

$$H' = H - \lambda P \tag{2.8}$$

where λ is a weight that controls the penalty term. They define P such that maximizing H' results in partitioning nodes into communities with similar total degree. This penalty function can be given as follows:

$$P = \frac{1}{2} \sum_{L \in \{labels\}} d_L^2 \tag{2.9}$$

where d_L is the total degree of nodes with label L. P has its minimum value when each node is in its own community and reaches its maximum when all the nodes are placed in one giant community. They select λ to be $\frac{1}{2m}$ where

m is the number of edges in the network. Finally, they rewrite Equation 2.8 as:

$$\begin{aligned} H' &= \frac{1}{2}\Big[\sum_{i,j} A_{ij}\delta(L_i, L_j) - \frac{1}{2m}\sum_{L\in\{labels\}} d_L^2\Big] \\ &= \frac{1}{2}\Big[\sum_{i,j} A_{ij}\delta(L_i, L_j) - \frac{1}{2m}\sum_{i,j} d_i d_j \delta(L_i, L_j)\Big] \\ &= \frac{1}{2}\Big[\sum_{i,j}\Big[A_{ij} - \frac{d_i d_j}{2m}\Big]\delta(L_i, L_j)\Big] \\ &= mQ \end{aligned}$$

where Q denotes Q-modularity given in Equation 2.4. Therefore, Q-modularity can be locally maximized using the LPAm algorithm. The computational cost of LPAm increases linearly with the number of edges in the network.

One of the drawback of these algorithm is that they might give undesirable solutions such as placing all the nodes in the same community. They are also very sensitive to the order in which nodes are processed.

2.1.6 Embedding Approaches

The algorithms that fall under this class aim at encoding graph structure into low-dimensional embeddings. Here we detail one of the algorithms that follow this approach: Node2vec.

Node2vec maps nodes to a d-dimensional feature space such that nodes that belong to the same community have similar representations. Basically, they map each node v to a feature representation which is predictive of the neighbourhood of v. To find such representations, they build an objective function as follows:

$$\max_f \sum_{v\in V} \log P(N(v)|f(v)) \tag{2.10}$$

where V is the set of nodes, f is the mapping function between nodes to feature representations and $N(v)$ is the set of nodes in the neighbourhood of node v. To make this function tractable, they assume that observing a node in the neighbourhood of node v is independent of observing any other node in this

neighbourhood given f:

$$P(N(v)|f(v)) = \prod_{n_i \in N(v)} P(n_i|f(v)) \tag{2.11}$$

Node2vec models $P(n_i|f(v))$ by using the angle between the feature representations of n_i and v:

$$P(n_i|f(v)) = \frac{e^{f(n_i).f(v)}}{\sum_{u \in V} e^{f(u).f(v)}} \tag{2.12}$$

By replacing Equations 2.11 and 2.12 in Equation 2.10, the objective function can be given as:

$$\max_f \sum_{v \in V} \Big[\sum_{n_i \in N(v)} f(n_i).f(v) - \log \big(\sum_{u \in V} exp(f(u).f(v)) \big) \Big] \tag{2.13}$$

They propose a neighbourhood sampling procedure based on random walks to find the neighbourhood of each node. They use two parameters p and q to guide the random walks. Setting p to a small value ensures that the random walk stays close to the starting node and setting q to a small value ensures that the random walk visits the node that are far away form the staring point. After finding the neighbourhood of each node in the network, they optimize the objective function (Equation 2.13) using stochastic gradient descent [9]. By using this approach, nodes can be mapped to feature representations and finally these representations can be clustered using clustering algorithms such as K-means.

The drawback of these algorithm is that we need to specify the number of features to map nodes to feature vectors which is not obvious. Another drawback is that they rely on clustering algorithms to find the communities.

2.2 Overview of Evaluation Approaches

There exist two categories of measures that can be used to evaluate the results of community mining algorithms: external measures and internal measures; which are described in the following.

2.2.1 External Measures

These measures compare the detected communities against the ground-truth communities in networks. Ground-truth communities are available in a limited number of real networks [1]. In addition to real networks, one can use synthetic benchmarks such as LFR [27], FARZ [15] and Dancer [28] to generate networks with built-in communities. Here, we detail two of the most well-known external measures: Normalized Mutual Information (NMI) [47] and Adjusted Rand Index (ARI) [21]. Both NMI and ARI have the maximum value of 1 when the two partitions have a perfect one-to-one correspondence. Let $MI(P,Q)$ be the mutual information between partitions P and Q and $H(P)$ be the entropy of P, then $NMI(P,Q)$ can be given as:

$$NMI(P,Q) = \frac{MI(P,Q)}{\sqrt{H(P)H(Q)}} \tag{2.14}$$

NMI has been widely used to evaluate the performance of community mining algorithms. However, this measure is sensitive to the number of detected communities k. A larger value of k might lead to a larger NMI regardless of the true number of communities in the network [18][54].

Hubert and Arabie [21] proposed adjusting the Rand Index for chance. Let $P = \{p_1, p_2, \ldots, p_r\}$ and $Q = \{q_1, q_2, \ldots, q_s\}$ be two partitions of a set of nodes V. Rand Index (RI) can be given as follows:

$$RI(P,Q) = \frac{a+d}{\binom{|V|}{2}} \tag{2.15}$$

where a is the number of node pairs belonging to the same communities in both P and Q and d is the number node pairs belonging to different communities in P and different communities in Q. This measure lies in the range of [0,1] and it does not have a constant expected value for random partitions. Let $E(M)$ be the expected value of a measure M. The general form of M with a constant expected value can be given as:

$$\frac{M - E(M)}{max(M) - E(M)} \tag{2.16}$$

[1] www-personal.umich.edu/∼mejn/netdata

Let n_{ij} be the number of nodes in common between communities p_i and q_j. It can be shown that the numerator of equation 2.15 (RI) is a linear transformation of $\sum_{ij} \binom{n_{ij}}{2}$. Hubert and Arabie [21] adjusted RI by replacing M in Equation 2.16 by the term $\sum_{ij} \binom{n_{ij}}{2}$:

$$ARI(P,Q) = \frac{\sum_{ij} \binom{n_{ij}}{2} - E(\sum_{ij} \binom{n_{ij}}{2})}{\max \sum_{ij} \binom{n_{ij}}{2} - E(\sum_{ij} \binom{n_{ij}}{2})} \tag{2.17}$$

They found the expected value of $\sum_{ij} \binom{n_{ij}}{2}$ assuming the generalized hypergeometric distribution as the model of randomness, i.e., partitions P and Q are chosen at random such that the size of the communities in P and Q are fixed, is:

$$E(\sum_{ij} \binom{n_{ij}}{2}) = \frac{\sum_i \binom{|p_i|}{2} \sum_j \binom{|q_j|}{2}}{\binom{|V|}{2}} \tag{2.18}$$

By replacing Equation 2.18 in Equation 2.17, ARI can be written as:

$$ARI(P,Q) = \frac{\sum_{ij} \binom{n_{ij}}{2} - \sum_i \binom{|p_i|}{2} \sum_j \binom{|q_j|}{2} / \binom{|V|}{2}}{\frac{1}{2}[\sum_i \binom{|p_i|}{2} + \sum_j \binom{|q_j|}{2}] - \sum_i \binom{|p_i|}{2} \sum_j \binom{|q_j|}{2} / \binom{|V|}{2}} \tag{2.19}$$

ARI is expected to have a value of 0 for two random partitions. However, it does not have a lower bound and might take negative values for some partitions [32]. For instance, consider a network with four nodes (v_1,v_2,v_3 and v_4) and two partitions $P = [(v_0, v_2), (v_1, v_3)]$ and $Q = [(v_0, v_1), (v_2, v_3)]$ (each partition consists of two communities). In this example ARI(P,Q) is equal to -0.5.

2.2.2 Internal Measures

These measures evaluate the results of community mining algorithms by considering how dense the communities are and how well they are separated. Here, we detail three of the internal measures: Conductance [8], C-Index [39] and Q-modularity [35]. Conductance is based on the number of edges running between the communities (cut size) and it helps measuring the quality of a single community in a network. Let $R(c, \overline{c})$ be the number of edges connecting nodes in community c to the nodes outside of c and d_i be the degree of node i. Conductance can be given as:

$$\varphi(c) = \frac{R(c, \overline{c})}{\min(\sum_{i \in c} d_i, \sum_{j \in \overline{c}} d_j)} \tag{2.20}$$

Conductance of a community can be generalized to the conductance of a partition P with s communities ($P = \{c_1, c_2, \ldots, c_s\}$) by taking the average conductance of the communities in the partition:

$$\phi(P) = \frac{1}{s} \sum_{c \in P} \varphi(c) \tag{2.21}$$

Lower conductance indicates a better quality of communities.

C-Index is originally a clustering validity criterion and is generalized by Rabbany *et al.* [39] to measure the quality of communities in graph data. Let θ be the sum over the shortest distances between every two nodes that are in the same community:

$$\theta = \frac{1}{2} \sum_{l=1}^{k} \sum_{i,j \in c_l} d(i,j) \tag{2.22}$$

C-Index can be given as:

$$\frac{\theta - min\ \theta}{max\ \theta - min\ \theta} \tag{2.23}$$

where $min\ \theta / max\ \theta$ is the sum over m smallest/largest distances between every two nodes in the graph where $m = \sum_{l=1}^{k} \binom{|c_l|}{2}$ with C_l denoting the set of nodes in community l.

C-Index lies between 0 (when the within-community distances are the shortest distances) and 1 (when the within-community distances are the largest distances).

Q-modularity (given in Equation 2.4) is another well-known quality measure. It measures the difference between the number of edges that lie within communities in a given partition P and the expected such number if edges were placed at random while maintaining the degree distribution. This measure lies in the range of [-1,1] and higher Q-modularity indicates a better quality of communities.

Chapter 3

SIWO Approach

3.1 Motivations

Q-modularity [35] has been widely used as an objective function in community mining tasks. Newman proposed a greedy agglomerative algorithm to maximize Q-modularity [36]. This algorithm starts by placing each node in its own community and then repeatedly merges the two communities which result in a maximum gain in Q-modularity. This algorithm runs in $O(n(n+m))$. Later, Clauset *et al.* [11] improved Newman's algorithm by utilizing efficient data structures and reduced the running time of the algorithm to $O(n \log^2 n)$ for sparse networks. Blondel *et al.* [7] proposed the fastest algorithm (linear time complexity) to optimize Q-modularity named Louvain. Their method repeats two main steps iteratively until no further improvement in Q-modularity can be achieved.

Although Q-modularity has been widely used, Fortunato and Barthélemy [17] showed that Q-Modularity suffers from the resolution limit which means that by optimizing Q-modularity, communities that are smaller than a scale can not be resolved. Kawamoto and Rosvall [23] investigated the resolution limit of the map equation [44] (an objective function based on flow of information) and they showed that maximizing the map equation can discover a wider range of community sizes than Q-modularity can. The field of view limit [45] is in contrast to the resolution limit which results in overpartitioning the communities with large diameter. Schaub *et al.* [45] showed that both Q-modularity and the map equation are affected by the field of view limit.

To overcome the resolution limit of Q-modularity, several propositions have been made, notably by [3], [30], [42], who introduced variants of this criterion allowing the detection of community structures at different level of granularity. However, these revised criteria make the method time-consuming, since they require to tune a parameter. Moreover, they do not solve efficiently the resolution limit which results from the comparison to a null model underlying the criterion according to [26]. Therefore, in this work, we retain the greedy approach of Louvain for its efficiency and ability to handle very large networks but we introduce a new objective function, named **S**trong **I**n, **W**eak **O**ut (SIWO) because it relies on the notions of weak and strong links defined in Section 3.2.1 and, which is not based on a comparison with the null model. As Louvain remains one of the most efficient algorithms, we use the same process to optimize our proposed objective function.

Moreover, we consider, as is usually the case, that a community corresponds to a subgraph sparsely connected to the rest of the graph, but, contrary to the majority of community detection methods, we also formally define, in Section 3.2.4, the conditions that a subgraph should meet in order to be considered as a community. Then, in Section 3.3, we present the generic community detection algorithm that we introduce. In addition to the optimization step, this algorithm includes steps to ensure the respect of conditions defining a community. Note that this general process can be applied regardless of the objective function and consequently used to improve other community detection methods as our experiences show.

Finally extensive experiments, described in Chapter 4, confirm that our objective function is less sensitive to the resolution limit and the field of view limit compared to the objective functions mentioned earlier, and that our algorithm has consistently good performance regardless of the size of communities in the network and is efficient on large size networks having up to million edges.

3.2 Notations and definitions

3.2.1 Weak and strong links

A community is oftentimes defined as a subgraph in which nodes are densely connected to each other while sparsely connected to the rest of the network. To find such communities in a network, one simple way is to divide the network into subgraphs so that the number of links that lie within the subgraphs is maximized. However, this approach fails to find the true communities where we have no prior information about the number of communities or their sizes since by placing all the nodes in a single community, the number of within-community links is maximized. To avoid putting all the nodes in a single community, one may penalize the missing links within the communities. Let us take Q-modularity [35] as an example. Q-modularity is a well-known criterion for evaluating community mining tasks and also an objective function to detect communities in networks. It is defined by:

$$Q = \frac{1}{2m} \sum_{i,j} \Big[A_{ij} - \frac{d_i d_j}{2m} \Big] \delta(c_i, c_j) \tag{3.1}$$

where A denotes the adjacency matrix, d_i and c_i are respectively the degree and the community of node i and $\delta(x, y)$ is the Kronecker function equal to 1 if $x = y$ and 0 otherwise. This criterion compares the number of edges inside communities with the expected number of edges that would reside in the communities under the null hypothesis *i.e.* if edges were randomly placed in the network under the condition that the degree of every node remains unchanged. According to Equation 3.1, Q-modularity is the summation of $A_{ij} - \frac{d_i d_j}{2m}$ for each pair of nodes i and j belonging to the same community. If nodes i and j are not connected ($A_{ij} = 0$) then the term $A_{ij} - \frac{d_i d_j}{2m}$ would take a value less than 0 and otherwise it would be in the range of [0,1) (assuming $d_i, d_j << 2m$). Therefore, Q-modularity gains from the links inside the communities while it penalizes the missing links in the communities. Let us consider the graph illustrated in Figure 3.1. This graph contains 3 communities: A, B and C with N_A, N_B and N_C nodes respectively and a single node x connected to community C with h links. By using this approach (penalizing

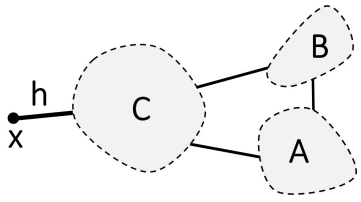

Figure 3.1: A simple network with three communities: A,B & C and one node (x) connected only to community C with h links

the missing links), if we add node x to the community C, we will add h links to within-community links while adding $(N_C - h)$ links to the missing links in community C and if $N_C >> h$ the quality of community C will be decreased. However, node x is not connected to any other community but community C so it definitely belongs to community C. Therefore, we do not follow this approach to build our criterion and we prefer to distinguish different kinds of links that we respectively call weak and strong links.

Basically, there are two types of links in a natural division of a graph into communities: links that run between communities (we refer to this type as **weak** links) and links that are inside communities (we refer to this type as **strong** links). Instead of penalizing the missing links, we build our criterion so that it encourages adding strong links to the communities while avoiding weak links. Now the question is, how can we differentiate between strong and weak links? Different links play different roles in graph connectivity; removing any random link within a community (a strong link) would not notably change graph connectivity while removing a link that runs between two communities (a weak link), may divide the graph into disconnected subgraphs. Figure 3.2 illustrates a network including two communities (each community is a clique of size 5). Let us focus on the link between nodes i and j (e_{ij}) and also the link between nodes j and k (e_{jk}) in this network. By removing e_{ij}, the graph will be divided into two disconnected subgraphs while removing e_{jk} does not change the graph connectivity remarkably which indicates that e_{ij} is more likely to be a between-community link than e_{jk}. Yet, what makes these links different from one another? Node j is connected to all the neighbours of node k (except node j itself). However, node i and j do not have any shared

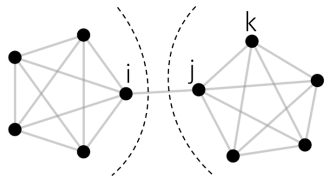

Figure 3.2: A simple network with two communities; each community is a clique of size 5

neighbours. Generally, nodes that are in the same community are more likely to have shared neighbours than nodes in different communities. In the next section, we use this property to build an edge weighting scheme to differentiate between strong and weak links.

3.2.2 Edge Strength

Given a graph $G = (V, E)$ where V is the set of nodes and E the set of edges with $|V| = n$ and $|E| = m$, we propose to assign a weight in the range of $(-1, 1)$ to each edge; larger weight indicates that the corresponding edge is more likely to be a strong link. As mentioned before, nodes in the same community tend to have more shared neighbours compared to nodes in different communities so if $S_{xy} > S_{xy'}$ then e_{xy} is more likely to be a strong link compared to $e_{xy'}$ if S_{xy} denotes the number of shared neighbours between nodes x and y defined by:

$$S_{xy} = |\{k \in V : (x, k) \in E, (y, k) \in E\}| \tag{3.2}$$

Note that two links are comparable according to S only if they have one shared endpoint otherwise comparing them based on S is not valid since they do not take their values from the same interval. To be able to compare the links according to S, we scale the S values down to $(-1, 1)$. Suppose that S_{xy} has the maximum value of S_x^{max} ($S_x^{max} = \max_{y:(x,y)\in E} S_{xy}$) for a particular node x. We divide the range $[-1, 1]$ into $S_x^{max} + 1$ equal-length segments. Each S value in the range of $[0, S_x^{max}]$ is then mapped to the center of $(n + 1)^{th}$ segment. The equation of this scaling can be given as:

$$w_{xy}^x = S_{xy} \frac{2}{S_x^{max} + 1} + \frac{1}{S_x^{max} + 1} - 1 \tag{3.3}$$

where w_{xy}^x is the scaled value of S_{xy} from the viewpoint of node x. As we can also scale S_{xy} from the viewpoint of node y: $w_{xy}^y = S_{xy}\frac{2}{S_y^{max}+1} + \frac{1}{S_y^{max}+1} - 1$ where $S_y^{max} = \max_{x:(y,x)\in E} S_{xy}$, the question is: which of the these weights (w_{xy}^x or w_{xy}^y) better represents the strength of e_{xy}? To decide whether we should trust x or y, we need to look at the role/importance of each one in the network. Is x in a very dense area in its community (community core) or is it sparsely connected to other communities? Local clustering coefficient (CC) [48] is one of the measures that reflects the role/importance of nodes which can be given as follows:

$$CC(x) = \frac{|\{e_{ij} : i, j \in N_x, e_{ij} \in E\}|}{\binom{d_x}{2}} \tag{3.4}$$

where d_x and N_x are respectively the degree and the set of neighbours of node x and e_{ij} is identical to e_{ji}. CC is in the range of [0,1] with 1 for nodes whose neighbours form cliques, and 0 for nodes whose neighbours are not connected to each other directly. Here, we scale each edge from the viewpoint of the endpoint that is more likely to be in a dense neighbourhood characterized by a large CC:

$$w_{xy} = \begin{cases} w_{xy}^x, & \text{if } CC(x) \geq CC(y) \\ w_{xy}^y, & \text{otherwise} \end{cases} \tag{3.5}$$

For instance, in Figure 3.2, $w_{jk} = 0.75$ and $w_{ij} = -0.75$. Using this method we expect strong links to have positive weights and weak links to have negative weights.

The notions of strong and weak links being defined, we assume that a good division of a graph into communities is a division where the weak links lie between the communities and strong links reside in the communities and, we introduce an objective function based on these notions.

3.2.3 SIWO Measure

The new measure, that we propose, encourages adding strong links into the communities while keeping the weak links outside of the communities (**S**trong

In, **W**eak **O**ut). This measure is defined as follows:

$$\sum_{i,j \in V} \frac{w_{ij}\delta(c_i, c_j)}{2} \tag{3.6}$$

where c_i is the community of node i and $\delta(x, y)$ is 1 if $x = y$ and 0 otherwise. SIWO is basically the sum of edge strength weights of edges which reside within the communities. This measure reaches its maximum when all the links with positive weights are inside the communities and all the links with negative weights lie between the communities. As mentioned in Section 3.2.2, strong links are expected to have positive weights while weak links are more likely to be assigned with negative weights using our edge weighting method. Therefore, maximizing this measure leads to placing strong links within communities and weak links between communities. This objective function provides a way to partition the set of nodes but it does not specify the conditions required by a subset of nodes to be a community. These conditions are defined in the following.

3.2.4 Community Definition

Following [40] we consider that a subgraph C is a community in a weak sense if the following condition is satisfied:

$$\sum_{v \in C} |N_v^C| > \sum_{v \in C} |N_v - N_v^C| \tag{3.7}$$

where N_v is the set of the neighbours of node v and N_v^C is the set of the neighbours of node v that are also in community C. This condition means that the collective of the nodes in a community have more neighbours within the community than outside. In this work, we expand this definition by adding one more condition. Given a partition $p = \{C_1, C_2, ..., C_t\}$ of a network, subgraph C_i is considered as a **qualified community** if it satisfies the following conditions:

1. C_i is a community in a weak sense (Equation 3.7).

2. The number of links within C_i exceeds the number of links towards any other subgraph C_j $(j \neq i)$ in the partition p taken separately, which can

be given as:

$$\frac{1}{2}\sum_{v\in C_i}|N_v^{C_i}| > \sum_{v\in C_i}|N_v^{C_j}| \tag{3.8}$$

In the next section, we introduce an original community detection algorithm which ensures that the detected communities meet the above conditions.

3.3 The SIWO Method

This algorithm starts with a pre-processing to calculate the edge strength weights based on the method proposed in Section 3.2.2. In order to reduce the computational time, we also remove the groups of nodes that satisfy the above-mentioned conditions (Equations 3.7 and 3.8) and for which the assignment to a community is obvious. In the second step, the SIWO measure introduced in Section 3.2.3 is optimized with the greedy procedure of Louvain which results in partitioning the network into disjoint subsets. Step 3 investigates whether these subsets satisfy the conditions defined in Equations 3.7 and 3.8 and sub-groups which do not comply with these conditions (unqualified communities) are identified. Afterwards, each unqualified community c is merged with one of its neighbouring communities (qualified or not), until no more unqualified community can be found in the network, as explained following. Finally, all the nodes that were temporarily removed in the previous steps, are assigned to appropriate communities in Step 4 (Post-processing). The details of the above-mentioned steps are discussed in order below.

Step 1. Pre-processing

This step aims to calculate the edge strength weights (w_{ij}) used in step 2, during the SIWO measure optimization. Moreover, in order to reduce the computational time, dangling nodes are also temporally removed. Node x is a dangling node if there exists node y such that by removing the link between x and y, the network would be divided into two disconnected parts with $part_x$ (the part containing node x) being a tree. Since $part_x$ has a tree structure and has the minimum possible density among connected graphs, it cannot form

Figure 3.3: A network with two communities and four dangling nodes (nodes 1,2,3 and 4)

a community on its own and, consequently its assignment to a community is obvious. So all the nodes in $part_x$ belong to the same community as node y (or they are outliers). For example, in Figure 3.3, nodes 1,2,3 and 4 are dangling nodes and they belong to the same community as node 5 unless we consider them outliers. Note that such tree-structured subgraphs that are attached to networks satisfy the conditions that we defined for qualified communities (Equation 3.7 and 3.8). However, they are very sparse and cannot be considered as communities. Therefore, they can be removed at the start in order to accelerate the process. The method to remove dangling nodes has two phases: 1- identify all singletons (nodes with degree of 1) in the network; 2- remove these nodes from the network (these nodes and the links incident to them are stored for later processing). These two phases are repeated until no more singletons can be found in the network. Note that we need to iterate over all the nodes in the network in the first iteration of the procedure to identify the singletons. However, for the rest of the iterations, we only need to iterate over the list of the neighbours of the nodes that are removed in the previous iteration. Algorithm 1 presents the method that we use for removing the dangling nodes.

Algorithm 1 Remove dangling nodes

Input: An undirected graph G
Output: Graph G with no dangling nodes

```
1: candidate_list ← list of nodes in G
2: while candidate_list is not empty do
3:     new_candidates ← empty list
4:     for all node ∈ candidate_list do
5:         if G.degree(node) = 1 then
6:             remember node and the link between node
7:             and its neighbour
8:             add node's neighbour to new_candidates
9:         end if
10:    end for
11:    for all node ∈ remembered nodes do
12:        remove node from G
13:    end for
14:    candidate_list ← new_candidates
15: end while
```

Step 2. Optimizing SIWO

Louvain's optimization process is retained in order to maximize SIWO since it is linear in the order of the number of edges in the graph. This greedy optimization process has two main phases that are iteratively executed until a local maximum of the objective function is reached (in this work we use SIWO measure as the objective function instead of the Q-modularity). The first phase starts by placing each node in graph G in its own community. Subsequently, each node is moved to the neighbour community which results in the maximum gain in the value of SIWO. If no gain can be achieved by moving a particular node into its neighbouring communities, then the node stays in its own community. In the second phase, a new weighted graph G' is created in which each node corresponds to a community detected in G. Two nodes in G' are connected if there exists at least one edge lying between their corresponding communities in G. Finally each edge e_{xy} in G' is assigned a weight equal to the sum of the weights of edges between the communities that match with x and y (if no weight is assigned to the edges, then it assumes that the weight of each edge is equal to 1). These two phases are iteratively repeated until no

further improvement in the SIWO objective function can be achieved.

Step 3. Qualified community identification

Step 3 aims to determine qualified communities complying with the conditions of our definition (Equations 3.7 and 3.8) from the dense subgraphs discovered in the previous step. Some of the previously-discovered communities might have only one member: they are called Lone communities and they correspond to nodes that are weakly connected to all of their neighbours ($S_x^{max} = 0$) and have links with non-positive weights incident to them. Therefore, they are left alone during the optimization of SIWO. Note that these nodes do not refer to dangling nodes which were removed in the first step. Since the decision about the communities of such nodes can not be made on edge strength, we let the majority of their neighbours decide about their communities. To reduce the computational time, like for dangling nodes, we prefer temporarily removing these nodes in this step and assign them to the community of the majority of their neighbours in the final step.

After removing the Lone communities, we identify the communities that do not satisfy the conditions given in Equations 3.7 or 3.8 (unqualified community); if no such community exists, we proceed to the next step (Post-processing). Otherwise, we keep merging each unqualified community with one of its neighbouring communities (qualified or not) until no more unqualified community exists. First, we remove the weights that we assigned to the edges in Step 1 and we assign a weight equal to 1 to each edge. Then, we create a new graph G^* in which each node corresponds to a community detected in Step 2 after the removal of the Lone communities. We add an edge between nodes x and y in G^* if their corresponding communities are connected. Each edge e_{xy} in G^* is assigned a weight equal to the sum of the weights of edges between the communities that correspond to x and y. We also add a self-loop to each node that has a weight equal to the sum of the weights of the edges that reside in its corresponding community. Then, we follow two phases of Louvain. In Phase 1, we put each node of G^* in its own community. Then we visit all the nodes in G^* sequentially. If a node x has a self-loop with a weight

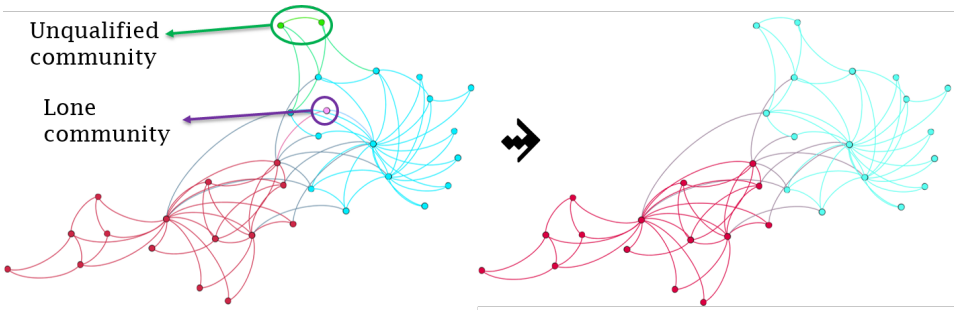

Figure 3.4: Detected communities by SIWO before and after step 3; communities are shown by different colors

that is larger than:

1. half of the sum of the weights of edges incident to it
2. weight of any edge connecting x to another node in G^*

which means that the community corresponding to x satisfies both the conditions in Equations 3.7 and 3.8, we let x stay in its community and proceed to the next node. Otherwise, we move node x to the neighbouring community that results in the maximum decrease in the sum of the weights of the edges that lie between communities in G^*. Once all the nodes in G^* are visited, the second phase starts. We create a new network similar to the way we created G^* with the difference that nodes correspond to the communities detected in the previous phase. Then we repeat the same 2-phase procedure for the new graph until no further changes can be made to the communities. Note that, if one disregards the removal of the Lone communities done to decrease the computational time, the time complexity of this step is of the same order as Louvain since it is based on the same greedy process. Figure 3.4 shows the detected communities by SIWO before and after step 3 on the Karate network [53].

Step 4. Post-processing

Finally, each Lone community that was temporarily removed from the network is sequentially added back to the network and is merged with the community in which it has the most neighbours. If two or more communities tie and they have more than one connection to the node, then one is chosen at random. If

the highest number of connections is 1, we choose the community of the most important neighbour, characterized by the largest degree of centrality within its community (number of neighbours in the same community over the size of community). Now, let us assume that nodes x and y are both in Lone communities and there is an edge between them and node x is added to the network before node y. At the time that we add node x to the network, we assign it to the neighbour community c which has the majority votes without considering node y. However, after adding node y, which may go to another community, community c might not be the most appropriate for x. In order to resolve this issue, once all Lone communities are added to the network, we repeat moving each one of them to the community of the majority of its neighbours until no further movement can be made.

Dangling nodes are also added to the communities in this step. Remember that their assignment is obvious. These nodes are added in the reverse order that they were removed from the network. Once they are added to the network, they are assigned to the community of their unique neighbour.

3.4 The Resolution Limit of SIWO

Fortunato and Barthélemy [17] used two sample networks to demonstrate how Q-modularity is affected by the resolution limits. These two networks are illustrated in Figure 3.5. The first example is a ring of cliques where each clique is connected to its adjacent cliques through a single link. The second example is a network containing 4 cliques: 2 of size k and 2 of size p. They showed that in the first network if the number of cliques is larger than about $\sqrt{m}$ with m being the total number of edges in the network, then optimizing Q-modularity results in merging the adjacent cliques into groups of two or more. However, each clique corresponds to a community in this network and they should not be merged together. They also show that in the second network, if $k >> p$ then by optimizing Q-modularity, the cliques of size p will be merged together and Q-modularity fails to find the correct communities.

Here, we analyze whether SIWO is affected by the resolution limit in these

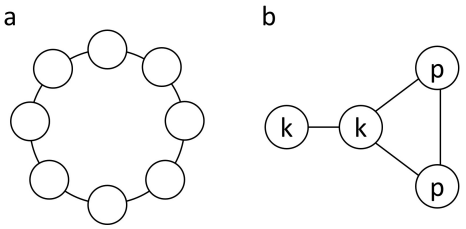

Figure 3.5: Schematic examples used to demonstrate the resolution limit a) a ring of cliques; adjacent cliques are connected through a single link b) a network with 2 cliques of size k and 2 cliques of size p; each line corresponds to a single link.

networks. As mentioned before, SIWO is the sum of the edge strength weights of the edges that reside in the communities. Therefore, if the sum of the weights of the edges between two communities c_1 and c_2 is larger than 0, then merging c_1 and c_2 results in increasing the value of SIWO, otherwise the SIWO value decreases. Now, let us consider the edge (e_{xy}) between two adjacent cliques in the first network. Since the two endpoints of this edge (x and y) do not have any shared neighbours ($S_{xy} = 0$), the edge between them has a non-positive weight from the viewpoint of x and y according to Equation 3.3 ($w_{xy}^x = \frac{1}{S_x^{max}+1} - 1 \leq 0$, $w_{xy}^y = \frac{1}{S_y^{max}+1} - 1 \leq 0$) and according to Equation 3.5 it has a non-positive total weight ($w_{xy} \leq 0$). Therefore, by maximizing SIWO measure in our algorithm, the adjacent cliques will not be merged together.

Now, let us consider the edge (e_{xy}) between the cliques of size p in the second network. Since the two endpoints of this edge (x and y) have at most 1 shared neighbours ($S_{xy} \leq 1$), the edge between them has a non-positive weight from the viewpoint of x and y according to Equation 3.3 (assuming $p \geq 4$: $S_x^{max}, S_y^{max} \geq 2$, $w_{xy}^x = \frac{2S_{xy}+1}{S_x^{max}+1} - 1 \leq \frac{2+1}{2+1} - 1 = 0$, $w_{xy}^y = \frac{2S_{xy}+1}{S_y^{max}+1} - 1 \leq \frac{2+1}{2+1} - 1 = 0$) and according to Equation 3.5 it has a non-positive total weight ($w_{xy} \leq 0$). Therefore, by maximizing SIWO measure in our algorithm, the cliques in the second network will not be merged either.

Fortunato and Barthélemy [17] also provided a general proof for the resolution limit of Q-modularity. However, the same proof is not possible here since to compute the SIWO value, we need to know the exact structure of the

network and therefore it is not possible to theoretically prove whether SIWO is affected by the resolution limit on any network. In the next section, we show experimentally how SIWO is able to detect communities of different sizes, thus resistant to both resolution limit and field of view limit.

3.5 Extension of SIWO for Weighted Networks

Edge weights in weighted networks often correspond to the strength of the connection between the two end nodes. In this case, edge weights can reveal useful information about the underlying community structure of the networks. Here, we generalize SIWO algorithm in order to take edge weights into account. SIWO has four steps: pre-processing, optimizing SIWO, qualified community identification and post-processing. In this extension, we only adjust the pre-processing step and the rest of steps are the same as the original algorithm. Let us remind that in the pre-processing step, SIWO assigns an edge strength weight (sw_{xy}) to each edge; a positive/negative weight indicates that the corresponding edge is probably a strong/weak link. In a weighted graph, each edge is associated with two weights:

1. the weight that is given in the input graph (gw_{xy})
2. the weight that SIWO assigns to it (sw_{xy})

We first scale gw down to [0,1]. Suppose that gw_{xy} has the maximum value of gw_x^{max} ($gw_x^{max} = \max_{y:(x,y)\in E} gw_{xy}$) and the minimum value of gw_x^{min} ($gw_x^{min} = \min_{y:(x,y)\in E} gw_{xy}$) for a particular node x. Each gw value can be scaled as follows:

$$\hat{gw}_{xy}^{x} = \begin{cases} \frac{gw_{xy} - gw_x^{min}}{gw_x^{max} - gw_x^{min}}, & \text{if } gw_x^{min} \neq gw_x^{max} \\ \frac{1}{deg(x)}, & \text{otherwise} \end{cases} \tag{3.9}$$

$\hat{gw}_{xy}^{x}$ is the scaled value of gw_{xy} from the viewpoint of node x; we can also scale gw_{xy} from the viewpoint of node y:

$$\hat{gw}_{xy}^{y} = \begin{cases} \frac{gw_{xy} - gw_y^{min}}{gw_y^{max} - gw_y^{min}}, & \text{if } gw_y^{min} \neq gw_y^{max} \\ \frac{1}{deg(y)}, & \text{otherwise} \end{cases} \tag{3.10}$$

where $gw_y^{max} = \max_{x:(y,x)\in E} gw_{xy}$ and $gw_y^{min} = \min_{x:(y,x)\in E} gw_{xy}$. Same as before, we scale each edge from the viewpoint of the endpoint that has a larger clustering coefficient (CC):

$$\hat{gw}_{xy} = \begin{cases} \hat{gw}_{xy}^{x}, & \text{if } CC(x) \geq CC(y) \\ \hat{gw}_{xy}^{y}, & \text{otherwise} \end{cases} \tag{3.11}$$

After scaling the gw values, we combine $\hat{gw}$ and sw values. We consider three cases:

1. Both $\hat{gw}$ and sw have large values (close to 1) for a particular edge: in this case both the input weight and our edge strength weighting method indicate that the corresponding edge is very strong and therefore the combination of $\hat{gw}$ and sw should take a large value to reflect the strength of this edge.

2. Both $\hat{gw}$ and sw have small values ($\hat{gw}$ close to 0 and sw close to -1) for a particular edge: in this case both the input weight and our edge strength weighting method indicate that the corresponding edge is very weak and therefore the combination of $\hat{gw}$ and sw should take a small value to reflect the weakness of this edge.

3. One of these weights ($\hat{gw}$ or sw) has a large value (close to 1) while the other one has a small value (close to 0 for $\hat{gw}$ and close to -1 for sw) for a particular edge: in this case there is a disagreement between the input weight and our edge strength weighting method and it is not clear whether the corresponding edge is strong or weak. Therefore, the combination of $\hat{gw}$ and sw should take a value close to 0 to reflect that the corresponding edge is neither strong nor weak.

Here, we combine gw and sw such that the above-mentioned cases are satisfied:

$$w_{xy} = \frac{sw_{xy} + 1}{2} + \hat{gw}_{xy} - 1 \tag{3.12}$$

w_{xy} is in the range of [-1,1]. Let us look at some examples to check whether the cases mentioned earlier are satisfied or not:

1. $\hat{gw}_{xy} = 0.9, sw_{xy} = 0.9 \rightarrow w_{xy} = 0.85$
2. $\hat{gw}_{xy} = 0.1, sw_{xy} = -0.9 \rightarrow w_{xy} = -0.85$
3. $\hat{gw}_{xy} = 0.9, sw_{xy} = -0.9 \rightarrow w_{xy} = -0.05$
4. $\hat{gw}_{xy} = 0.1, sw_{xy} = 0.9 \rightarrow w_{xy} = 0.05$

After combining sw and $\hat{gw}$ weights, we compute the SIWO measure based on the combined weight (w_{xy}) and we follow Steps 2, 3 and 4 of the original SIWO algorithm.

Chapter 4

Experimental results

4.1 Evaluation of SIWO Edge Weighting Method

In this section, we analyze how our edge weighting method differentiates between strong and weak links. As mentioned earlier in Section 3.2.2, we expect that our edge weighting method to assign positive weights to edges that lie inside communities and negative weights to edges that run between communities. We use four real networks with ground-truth communities:

1. Karate [53]: this network contains 34 nodes and 78 edges with nodes presenting members of a karate club and links presenting the friendship relationship between members. The club was divided into two communities after a conflict between the club's administrator and the instructor.

2. football [19]: this network contains 115 nodes and 613 edges with nodes presenting football teams. Two teams are connected if they played with each other. Each team is either an independent team or it belongs to one of 11 conferences. We consider all independents as one community, overall 12 communities.

3. polblogs [1]: we consider the largest connected component of this network with 1222 nodes and 16717 edges. Nodes present weblogs on US politics and edges correspond to the hyperlinks between weblogs. Weblogs are divided into two communities (liberal and conservative).

4. polbooks [1]: this network contains 105 nodes and 441 edges with nodes presenting books about US politics and links presenting frequent co-purchasing of books by the same customers. Each node is manually assigned to one of three communities (liberal, neutral and conservative).

Figure 4.1 presents the distribution of edge strength weights for within-community edges and between-community for each real network. Our edge weighting method is able to clearly differentiate between strong and weak links in Karate and football networks. It is also able to assign negative weights to the majority of between-community edges in polblogs and is able to assign positive weights to the majority of within-community weights in polbooks. Here, we also find the correlation between edge betweenness and the weights that our edge weighting method assigns to edges. The links that run between the true communities (weak links) have high edge betweenness. Therefore, we expect a negative correlation between the edge weights and their edge betweenness. The Pearson correlation between these two measures are as follows: Karate network: -0.23; football network: -0.77; polblogs network: -0.04; polbooks network: -0.4. In the next section, we test and compare SIWO and other state of the art community detection algorithms on both real and synthetic networks.

4.2 Comparison Methodology and Results

We compared the performance of our method with the most widely used and efficient algorithms, as pointed out in several recent state of art studies [14], [51], on both real and synthetic networks. The algorithms are: 1- Fastgreedy [11] available in igraph package [13]; 2- Infomap; 3- Infomap+ which is Infomap to which we added the third step of our algorithm; we added this step to make it less sensitive to the field of view limit and to demonstrate that our framework can be applied to improve other algorithms; 4- label propagation [41] available in igraph package; 5- Louvain[2] [7]; 6- Walktrap[3] [38].

[1] http://www.orgnet.com
[2] https://github.com/taynaud/python-louvain
[3] https://www-complexnetworks.lip6.fr/ latapy/PP/walktrap.html

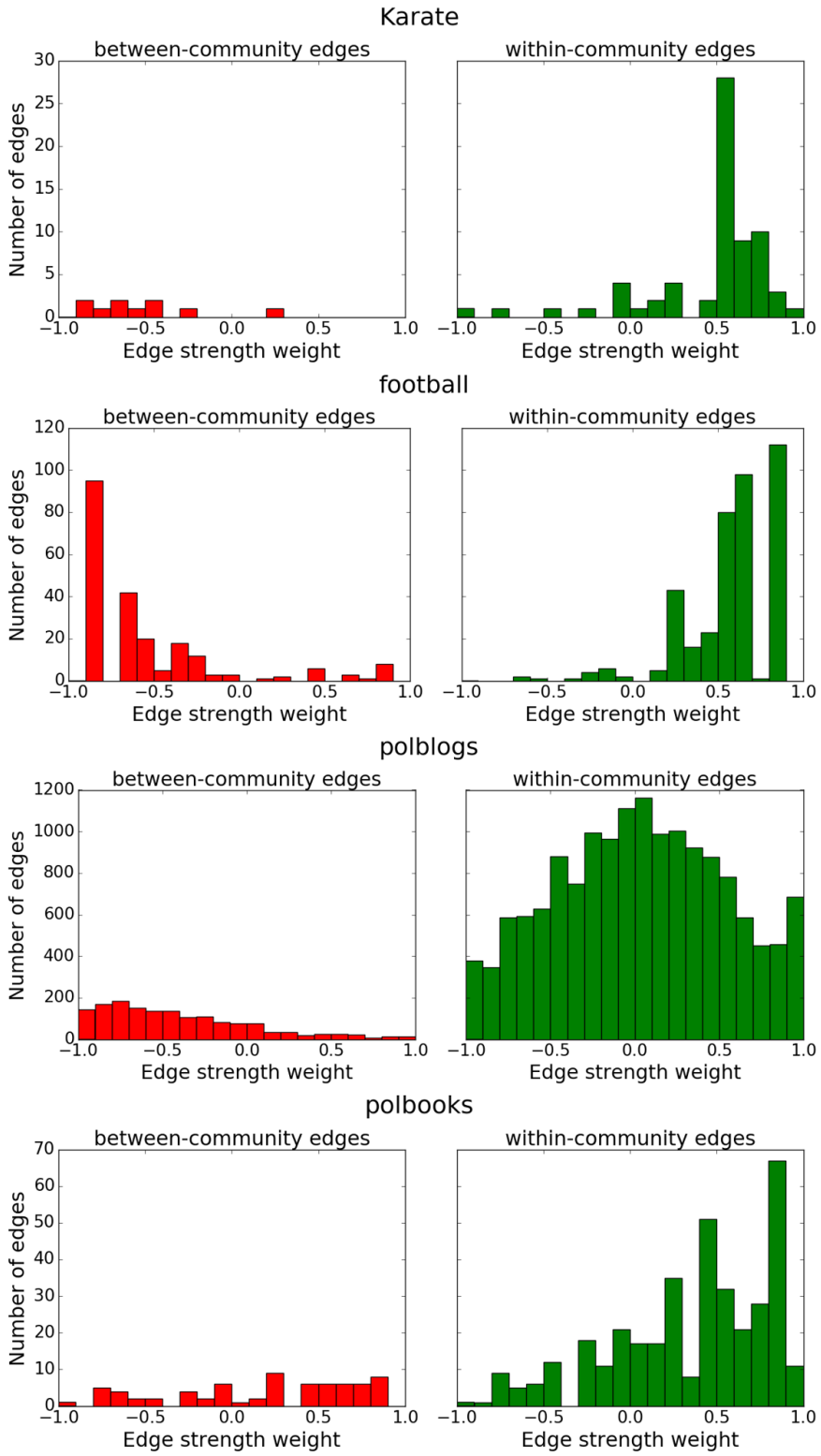

Figure 4.1: Evaluation of SIWO edge weighting method; the distribution of edge strength weights are presented separately for within-community edges and between-community edges for each network

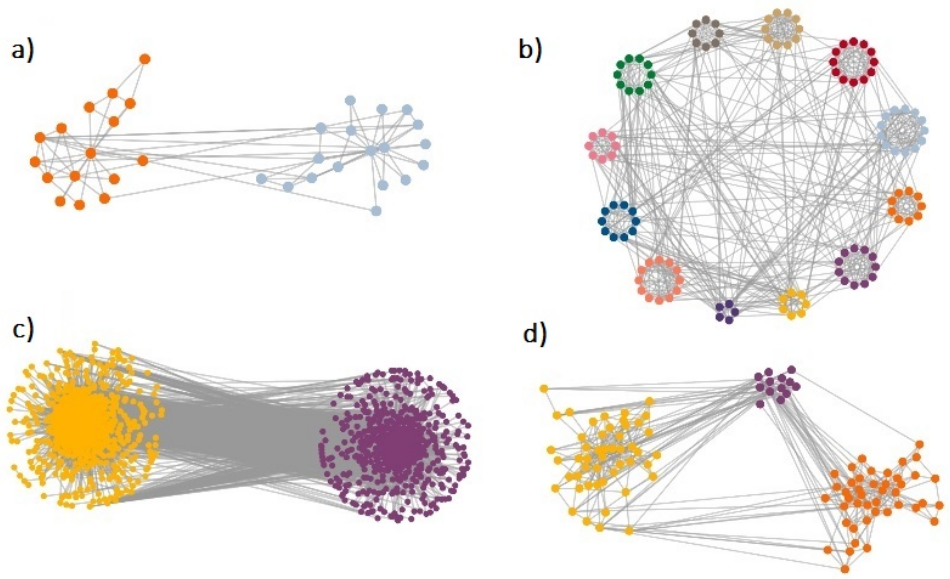

Figure 4.2: Visualization of real networks: a) Karate b) football c) polblogs d) polbooks

The results are evaluated according to the Normalized Mutual Information (NMI) [47] and ARI [21]. Both NMI and ARI have the maximum value of 1 when the two partitions have a perfect one-to-one correspondence. We also compared the results of different algorithms according to the ratio of the number of detected communities over the true number of communities in the ground-truth to observe how each of these algorithms is affected by the resolution and the field of view limits. All of the experiments are performed on a commodity laptop with an i7 processor and 8GB ram.

4.2.1 Unweighted Networks

Real networks

In this section, we use the four real networks introduced in Section 4.1. Figure 4.2 shows a visualization of real networks using Meerkat [10]; different colors refer to different ground-truth communities. The small size of these networks allows us to verify the quality of ground-truth communities. For large networks, it is not possible to manually assign nodes to communities so metadata is commonly used as the ground-truth communities. However,

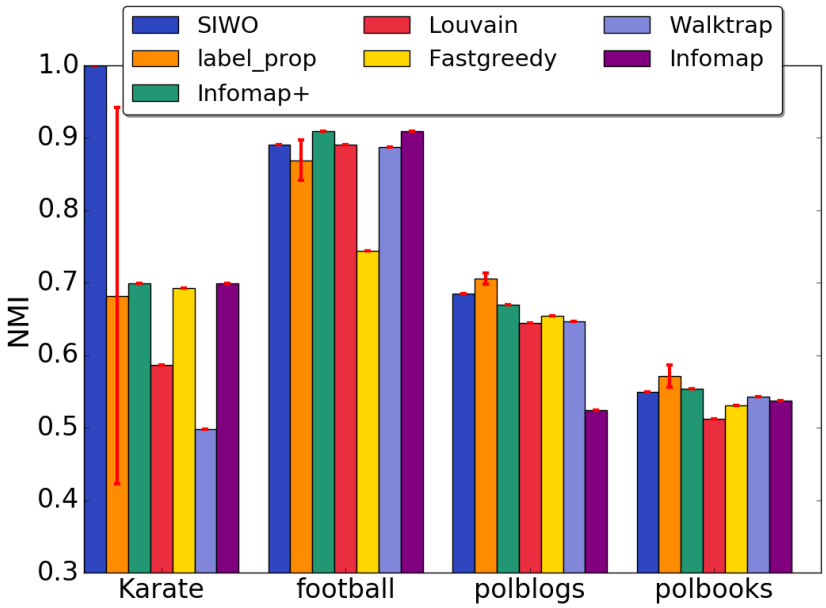

Figure 4.3: Evaluation of 7 algorithms according to NMI on real networks. We performed each algorithm 10 times on each network; the error bars correspond to the standard deviation of NMI.

recent studies [20], [37] have shown that metadata should not be used in evaluation of community detection algorithms. Thus, we use generators to create large networks with built-in communities.

Figures 4.3 and 4.4 present the comparison with respect to NMI and the ratio of the number of detected communities over the true number of communities in the ground-truth ($\overline{C}/C_{real}$). The horizontal red line corresponds to the case when $\overline{C}/C_{real}$ equals to 1 implying that the correct number of communities is detected (Figure 4.4). SIWO is among the top performers in all the networks both in terms of NMI and the number of communities. SIWO outperforms the rest of the algorithms on Karate network (it detects the exact communities as the ground-truth). All algorithms except SIWO, tend to split the 2 communities of Karate network into smaller communities. All the algorithms can detect the correct number of communities in football network except Fastgreedy, which underestimates this number. Infomap detects a considerably larger number of communities in polblogs network which indicates that this algorithm is sensitive to the field of view limit [45]. However, In-

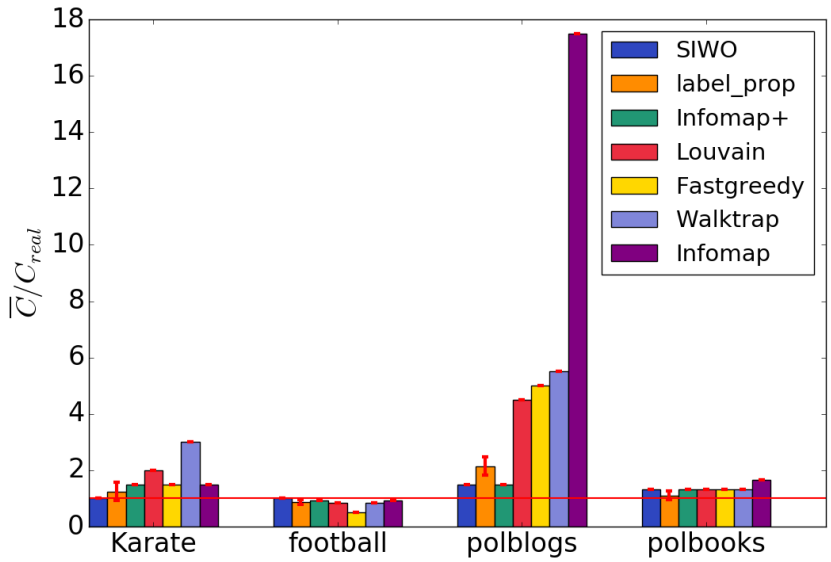

Figure 4.4: Evaluation of 7 algorithms according to the ratio of the number of detected communities over the true number of communities in the ground-truth on real networks. We performed each algorithm 10 times on each network; the error bars correspond to the standard deviation of $\overline{C}/C_{real}$).

fomap+ is much less sensitive to this limit which implies that our third step of SIWO algorithm, added to Infomap+, is very effective in resolving the field of view limit. All of the algorithms have similar results on polbooks networks.

We also compare SIWO and Louvain on Eurosis network [49]. This network presents scientific web pages and the hyper-links between them. These web pages belong either to one of 12 European countries or are considered as international. We remove the international web pages and consider its largest connected component with 1218 nodes and 5999 undirected links. The ground-truth communities are not known. However, since each of these European countries has its own language, web pages in different countries are sparsely connected to each other. In addition, as reported in [49], some of the countries can be divided into smaller components e.g. Montenegro network includes three components: 1- Telecom and Engineering, 2- Faculties and 3- High Schools. Louvain detects 13 communities; almost each community corresponds to a country except for two communities that belong to the same country. SIWO detects 16 communities in this network. Figure 4.5 presents

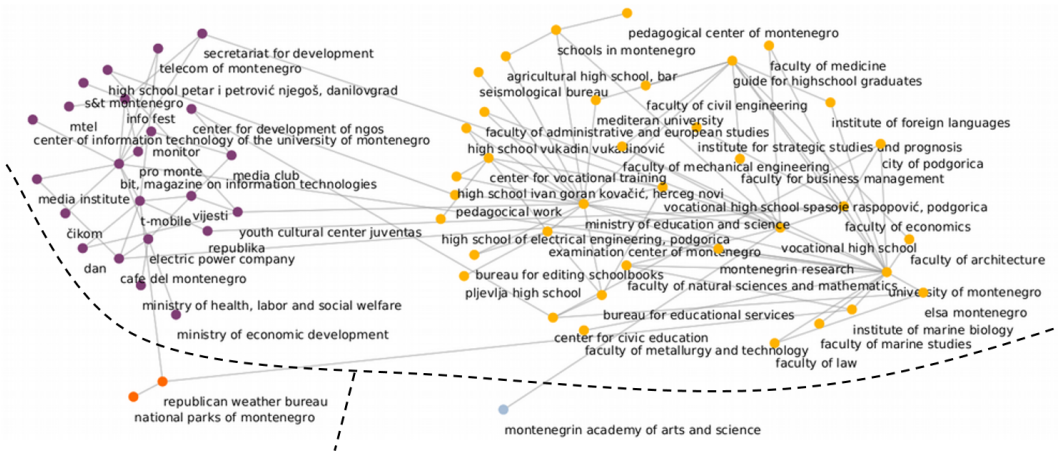

Figure 4.5: Detected communities in the subgraph corresponding to Montenegro in Eurosis network; the communities detected by SIWO are shown by different colors and the communities detected by Louvain are separated with dashed lines. Louvain merges the main components of this subgraph.

Table 4.1: Input parameters of LFR benchmark used for comparing the performance of SIWO with its contenders: Set 1 contains networks with large communities and Set 2 contains networks with small communities. For each combination of parameters we generated 10 networks

	Set 1	Set 2
number of nodes (N)	$[1, 10, 50, 100] \times 10^3$	$[1, 10, 50, 100] \times 10^3$
average degree	20	20
max degree	$N/10$	$\sqrt{N}$
mixing parameter	$[1, \ldots, 7] \times 0.1$	$[1, \ldots, 7] \times 0.1$
min community size	$N/20$	default
max community size	$N/10$	by default $\sqrt{N}$

Montenegro network. Louvain assigns all nodes (except for three) to one giant community. However, SIWO puts Faculties and High Schools in one community (shown in orange) and Telecom and Engineering web pages in another community (shown in purple). These two communities are connected to each other with only 7 links. However, Louvain cannot separate them due to its resolution limit.

Synthetic Networks

To analyze the effect of the resolution limit/the field of view limit, it is important to test how community detection algorithms perform on networks with

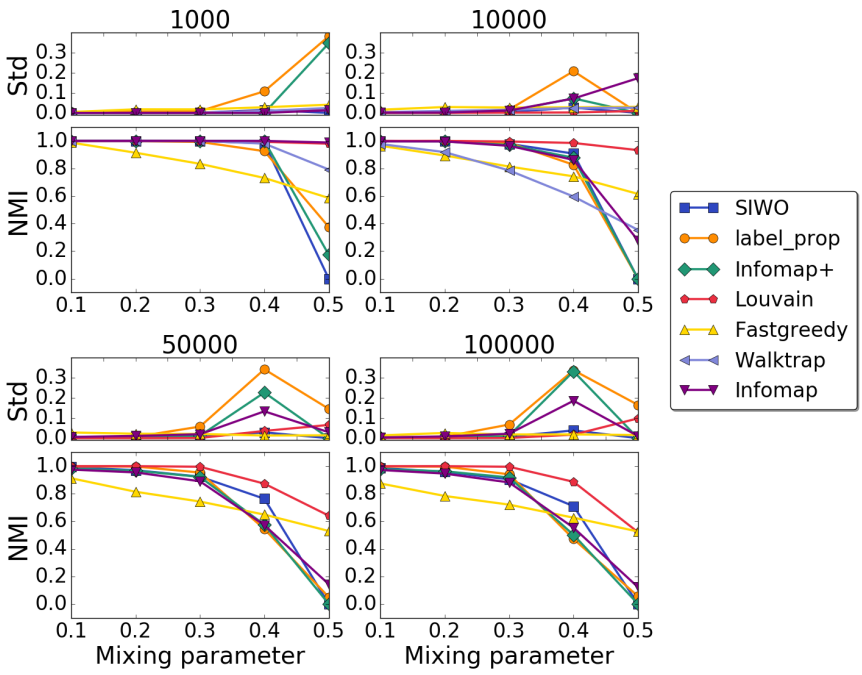

Figure 4.6: Evaluation of 7 different algorithms according to NMI on networks with large communities generated with LFR. Each panel corresponds to networks with a specific number of nodes (1000, 10000, 50000, 100000) and is divided into two parts; the lower/upper part illustrates the average NMI/ standard deviation of NMI over 10 graph realizations as a function of the mixing parameter. Walktrap fails for more than 10000 for lack of memory.

small/large communities. LFR benchmark [27] allows us to control the size of communities in the synthetic networks. Therefore, in this work we generate two sets of networks using LFR: one with large communities and one with small communities. The first set is in favour of algorithms that suffer from resolution limit such as Louvain and the second set is in favour of algorithms with field of view limit such as Infomap. Each set includes networks with a varying number of nodes and mixing parameter (the average fraction of the neighbours of a node belonging to a community other than the community of the node). We do not consider networks with mixing parameter > 0.5 since beyond this point and including 0.5, the communities in the ground-truth no longer satisfy the definition of community. Table 4.1 summarizes the range of input parameters used to generate these two sets. Figure 4.6 presents the evaluation of SIWO along with the baseline algorithms according to NMI on

networks with **large communities**.

Walktrap performs well on the smallest network in the set. However, there is a noticeable performance hit when applying this algorithm to the networks of size 10000. Unfortunately, we were not able to apply Walktrap on networks with sizes 50000 and 100000 due to memory constrains. Although, both Fastgreedy and Louvain try to detect communities by optimizing the same objective function (Q-modularity), Louvain has considerably better results compared to Fastgreedy. The performance of Fastgreedy monotonically decreases as the mixing parameter increases. However, Louvain correctly detects the communities when the mixing parameter is less than or equal to 0.3 (NMI $\simeq$ 1) regardless of the size of the network. The NMI remains almost steady up to when the mixing parameter reaches 0.4 and it drops when the mixing parameter reaches 0.5 as expected. Label propagation, Infomap and Infomap+ perform well up to when the mixing parameter reaches 0.3. However, a larger mixing parameter causes a rapid decrease in the NMI value when applying these algorithms to the two largest networks in the set. These three algorithms have a large standard deviation and their outputs are not stable on networks with large communities. SIWO has the best performance after Louvain (but remember that this set is in favour of algorithms with resolution limit such as Louvain); it has large NMI values up to when the mixing parameter reaches 0.5 and it suddenly drops which is expected since when the mixing parameter is equal to 0.5, the condition defined in Equation 3.7 is not satisfied and our algorithm merges all or most of the communities together. SIWO also has a very small standard deviation over all values of the mixing parameter which indicates that it has stable outputs.

Figures 4.7 and 4.8 illustrate the results according to $\overline{C}/C_{real}$ on networks with large communities. The best results corresponds to the case when $\overline{C} = C_{real}$ (*i.e.* 1 on the second axis). The results for Infomap and Walktrap are plotted separately in Figure 4.8 since they have a very different scale compared to others. Figure 4.7 shows that SIWO, Louvain and Fastgreedy are the best performers in terms of the number of communities and they also have a very small standard deviation whereas, Infomap+ and label propagation have a

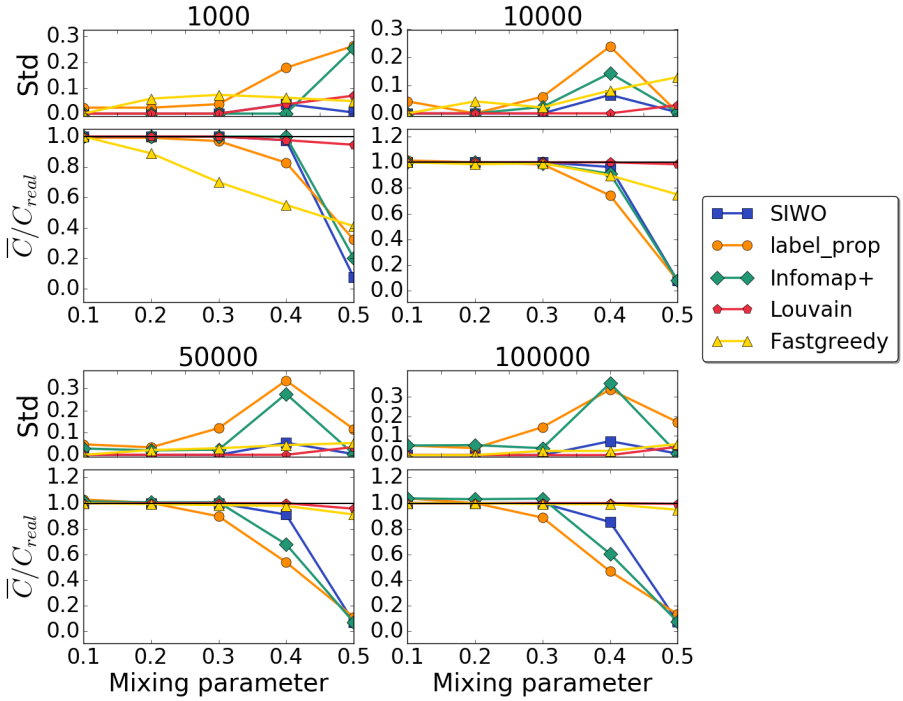

Figure 4.7: Evaluation of SIWO, label propagation, Infomap+, Louvain and Fastgreedy according to the ratio of the number of detected communities over the true number of communities ($\overline{C}/C_{real}$) on networks with large communities generated with LFR. Each panel corresponds to networks with a specific number of nodes (1000, 10000, 50000, 100000) and is divided into two parts; the lower/upper part illustrates the average $\overline{C}/C_{real}$/ standard deviation of $\overline{C}/C_{real}$ over 10 graph realization as a function of the mixing parameter.

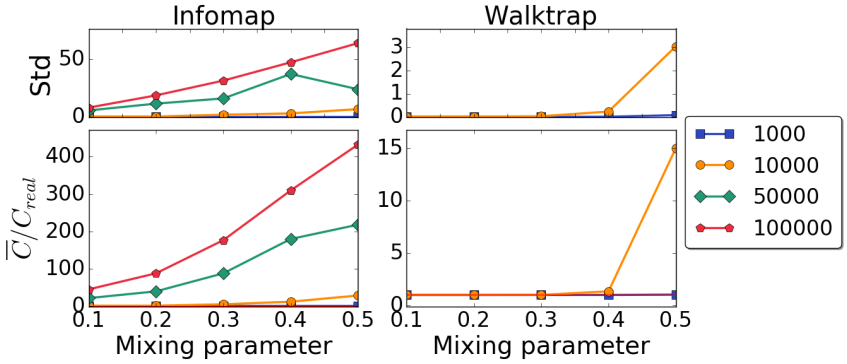

Figure 4.8: Evaluation of Infomap and Walktrap according to the ratio of the number of detected communities over the true number of communities ($\overline{C}/C_{real}$) on networks with large communities generated with LFR. Each panel corresponds to an algorithm and is divided into two parts; the lower/upper part illustrates the average $\overline{C}/C_{real}$/ standard deviation of $\overline{C}/C_{real}$ over 10 graph realization as a function of the mixing parameter.

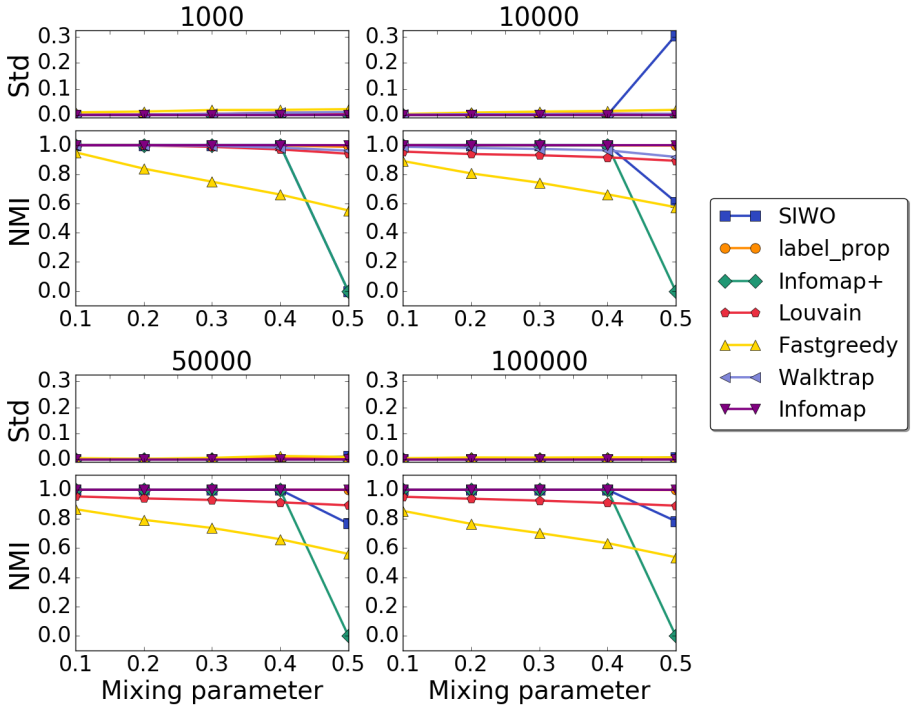

Figure 4.9: Evaluation of 7 different algorithms according to NMI on networks with small communities generated with LFR. Each panel corresponds to networks with a specific number of nodes (1000, 10000, 50000, 100000) and is divided into two parts; the lower/upper part illustrates the average NMI/ standard deviation of NMI over 10 graph realization as a function of the mixing parameter. Walktrap fails for more than 10000 for lack of memory.

large standard deviation and fail to find the correct number of communities when the mixing parameter exceeds 0.3.

Figure 4.8 shows that Infomap finds a significantly larger number of communities compared to the ground-truth, which indicates that this algorithm is very sensitive to the field of view limit. However, as said previously, this is not the case for Infomap+ notably when the mixing parameter is less or equal to 0.3 which implies that our third step of SIWO is very effective in resolving the field of view limit. Walktrap detects almost the correct number of communities when the mixing parameter is below 0.5 but it fails on networks with more than 10000 nodes for lack of memory.

Figure 4.9 shows the results according to NMI on the networks with **small communities** (remember that these networks are in favour of algorithms with

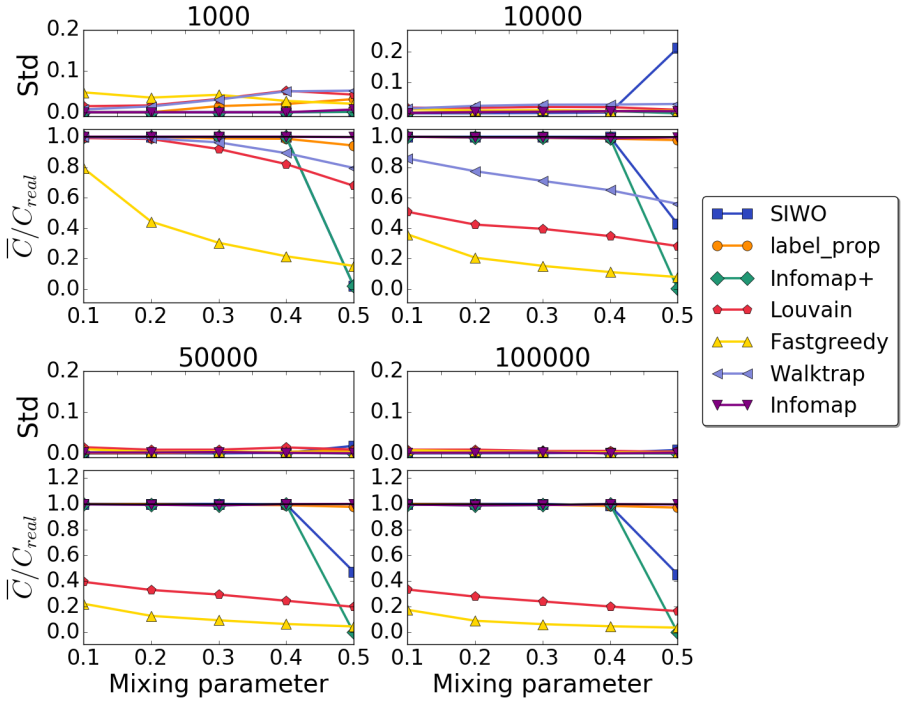

Figure 4.10: Evaluation of 7 different algorithms according to the ratio of the number of detected communities over the true number of communities ($\overline{C}/C_{real}$) on networks with small communities generated with LFR. Each panel corresponds to networks with a specific number of nodes (1000, 10000, 50000, 100000) and is divided into two parts; the lower/upper part illustrates the average $\overline{C}/C_{real}$/ standard deviation of $\overline{C}/C_{real}$ over 10 graph realization as a function of the mixing parameter. Walktrap fails for more than 10000 for lack of memory.

the field of view limit such as Infomap). All the algorithms (except for Fastgreedy) perform really well according to NMI on all networks regardless of their sizes (NMI $\simeq$ 1) for all values of the mixing parameter below 0.5. However, Figure 4.10 clearly shows that Louvain and Fastgreedy underestimate the number of communities. Although NMI is widely used in evaluating the community detection algorithms, it does not seem to be the right criterion for evaluation when we are dealing with large networks with many small communities. Therefore, we also provide the comparison according to ARI [21] (Figure 4.11). This figure clearly shows the resolution limit of Louvain, while SIWO detects almost the correct communities in the network.

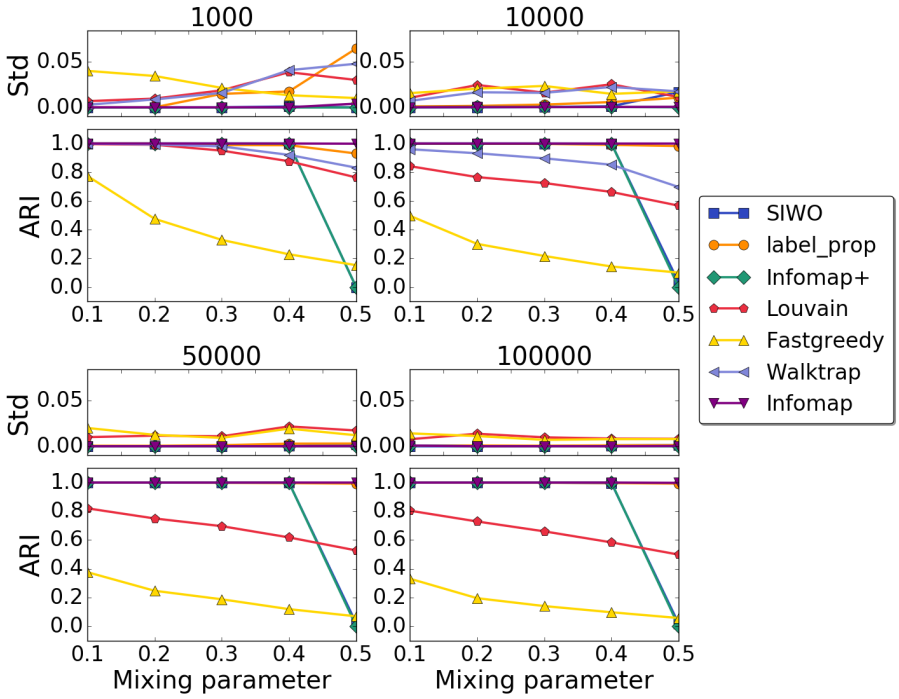

Figure 4.11: Evaluation of 7 different algorithms according to ARI on networks with small communities generated with LFR. Each panel corresponds to networks with a specific number of nodes (1000, 10000, 50000, 100000) and is divided into two parts; the lower/upper part illustrates the average ARI/ standard deviation of ARI over 10 graph realization as a function of the mixing parameter. Walktrap fails for more than 10000 for lack of memory.

4.2.2 Weighted Networks

We used LFR benchmark to generate weighted networks with built-in communities. LFR assigns weights to edges as follows:

1. They assign a strength s_i to each node i (they claim that there is a power law relation between the strength and the the degree of a node)

2. They specify a parameter μ_w to assign the internal strength (s_i^{in}) and the external strength (s_i^{ext}). The internal strength is the sum of the edge weights that connect i with another node in the same community as i and the external strength is the sum of the edge weights that connect i with another node outside of community of node i.

3. They assign weights such that for each node i:

$$s_i^{in} = (1 - \mu_w)s_i$$
$$s_i^{ext} = (\mu_w)s_i$$

By using this method, the average weight of the edges that connect node i with other nodes in the same community is $\frac{s_i^{in}}{k_i^{in}}$ (k_i^{in} is the number of neighbours of i that are in the same community as i). The average weight of the edges that connect node i with other nodes outside of community of node i is $\frac{s_i^{ext}}{k_i^{ext}}$ (k_i^{ext} is the number of neighbours of i that are not in the same community as i).

They also use another parameter named mixing parameter (μ_t) to assign the internal degree (k_i^{in}) and the external degree (k_i^{ext}). They assign degrees such that for each node i:

$$k_i^{in} = (1 - \mu_t)k_i$$
$$k_i^{ext} = (\mu_t)k_i$$

where k_i is the total degree of node i.

By default, μ_t is equal to μ_w. In this case, the average weight of within-community edges is equal to the average weight of between-community edges:

$$\frac{s_i^{in}}{k_i^{in}} = \frac{(1 - \mu_w)s_i}{(1 - \mu_t)k_i} = \frac{s_i}{k_i}$$
$$\frac{s_i^{ext}}{k_i^{ext}} = \frac{(\mu_w)s_i}{(\mu_t)k_i} = \frac{s_i}{k_i}$$

However, if we assume that the edge weights correspond to the strength of the connection between the two end nodes, then we should expect the within-community edges to have a higher average weight compared to between-community edges. Therefore, we set μ_w to be smaller than μ_t. We generate two sets of weighted networks: one set with 1000 nodes and another one with 10000 nodes. The values of the rest of the LFR parameters are shown in Table 4.2.

For each set of parameters we created 10 weighted networks and we tested each algorithm twice on each network: once considering the edge weights and

Table 4.2: Input parameters of LFR benchmark for weighted networks

average degree	20
max degree	$N/10$
μ_t	0.4
μ_w	0.1
min community size	$N/20$
max community size	$N/10$

once ignoring the edge weights. The results are given in Table 4.3; the values are the average over 10 networks and the standard deviation is given in parenthesis. Table 4.3 shows that the weighted version of the investigated algorithms from literature reflect better performance according to ARI, NMI, and $\overline{C}/C_{real}$ compared to their non-weighted versions. The SIWO contenders (except infomap and infomap+) show better performance according to $\overline{C}/C_{real}$ when they ignore the edge weights on networks with 10000 nodes. However, the edge weights are beneficial to the weighted SIWO and it is able to detect the correct number of communities on networks with 10000 nodes. Although the edge weights seem to be helpful in resolving the field of view limit of infomap on networks with 10000 nodes, SIWO tends to have a more accurate performance according to the number of communities.

4.3 SIWO, Deterministic or Non-deterministic?

The greedy approach of Louvain in optimizing the Q-modularity is known to be non-deterministic and sensitive on the order in which nodes are processed which means that if we change the node processing order we will get different results (partitioning) for the same network. SIWO also uses the same greedy approach of Louvain and therefore, is non-deterministic. In this section, we use Louvain and SIWO to detect communities in the four real networks that we introduced in Section 4.1. However, we change the node processing order in both Louvain and SIWO each time we execute them and we test each one 10 times on each network. Figure 4.12 shows the ground-truth communities and the frequency matrices of pairwise community membership obtained by SIWO and Louvain. The red blocks in the matrices on the left column correspond

Table 4.3: Evaluation of 7 algorithms on weighted networks; the values are average over 10 networks and the standard deviation is given in parenthesis

	networks with 1000 nodes			networks with 10000 nodes		
algorithm	ARI	NMI	$\overline{C}/C_{real}$	ARI	NMI	$\overline{C}/C_{real}$
SIWO	0.99(0.01)	0.99(0)	1(0)	0.90(0.05)	0.91(0.02)	0.98(0.03)
weighted SIWO	1(0)	1(0)	1(0)	1(0)	1(0)	1(0)
fastgreedy	0.54(0.04)	0.73(0.02)	0.55(0.05)	0.72(0.03)	0.75(0.02)	0.92(0.06)
weighted fastgreedy	1(0)	1(0)	1(0)	0.99(0.02)	0.99(0.02)	1.21(0.17)
infomap	1(0)	1(0)	1(0)	0.80(0.13)	0.85(0.05)	13.95(4.25)
weighted infomap	1(0)	1(0)	1(0)	0.97(0.05)	0.96(0.06)	7.08(7.88)
infomap+	1(0)	1(0)	1(0)	0.79(0.15)	0.87(0.05)	0.90(0.12)
weighted infomap+	1(0)	1(0)	1(0)	0.97(0.05)	0.97(0.06)	1(0)
label_prop	0.84(0.18)	0.96(0.05)	0.89(0.12)	0.75(0.25)	0.89(0.10)	0.82(0.15)
weighted label_prop	1(0)	1(0)	1(0)	0.98(0.02)	0.98(0.02)	2(0.62)
louvain	1(0)	1(0)	1(0)	0.99(0)	0.98(0)	1(0)
weighted louvain	1(0)	1(0)	1(0)	0.99(0.02)	0.99(0.02)	1.21(0.17)
walktrap	0.98(0.01)	0.98(0.01)	1(0)	0.57(0.03)	0.59(0.03)	1.45(0.29)
weighted walktrap	1(0)	1(0)	1(0)	0.95(0.09)	0.95(0.09)	1.96(0.48)

to ground-truth communities. The middle and the right columns present the frequency matrices of pairwise community membership obtained by SIWO and Louvain respectively. SIWO is not affected by the node processing order when applied to Karate and football networks and slightly sensitive to this order when applied to polblogs and polbooks. However, Louvain seems to be very sensitive to this order on Karate football and polbooks networks. Overall, SIWO is less sensitive to node processing order compared to Louvain.

4.4 Scalability

In this section, we analyze how the computational cost of SIWO varies with the size of the network. The pre-processing step has two phases: removing dangling nodes which requires a time of the order of n, and calculating the weights of edges which requires a time of the order of $nd^2 = 2md$ where d is the average degree (in many real networks d is much smaller than n and it

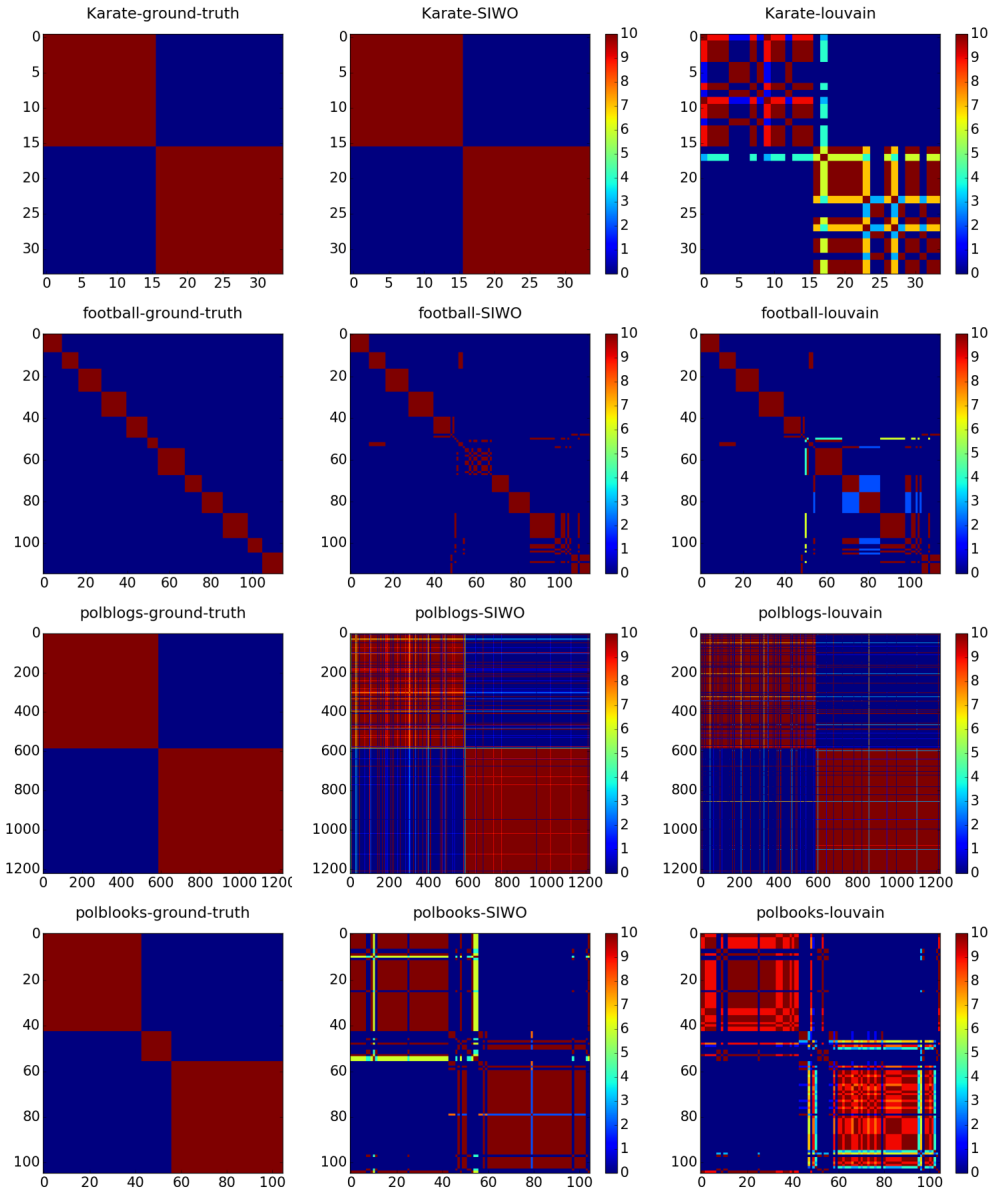

Figure 4.12: Ground-truth communities and frequency matrices of pair-wise community membership obtained by SIWO and Louvain. The red blocks in the matrices on the left column correspond to true communities.

Table 4.4: Input parameters of LFR benchmark used for scalability testing. For each combination of parameters we generated 10 networks.

number of nodes (N)	$[1, 10, 50, 100] \times 10^3$
average degree	20
max degree	$\sqrt{N}$
mixing parameter	0.3
min community size	default
max community size	by default $\sqrt{N}$

does not grow with n [16]). The second step (optimizing SIWO) and the third step (quality community identification), follow the same greedy process as Louvain does, so they require a time of the order of m (same as Louvain). The time complexity of the last step (Post-processing) depends on the number of Lone communities and the worst case (all the nodes are in Lone communities) requires a time of the order of nd^2. Overall, the time complexity of SIWO is $O(n + md)$, which is essentially similar to Louvain due to the fact that d is small and $n = 2m/d$.

Figure 4.13 shows the execution time of SIWO along with the baseline algorithms as a function of the number of nodes. We use a subset of synthetic networks used in Section 4.2.1 for comparing the performance of SIWO and the state of the arts algorithms. The input parameters of LFR benchmark used for creating the networks in this subset is given in Table 4.4.

The computational cost of label propagation, Infomap, Infomap+, Louvain and SIWO increases linearly with the size of the network and label propagation is the fastest among them. Note that the choice of the programming language can affect the execution time and these algorithms are implemented in different programming languages. The current implementation of SIWO is in Python. and it is able to detect communities in a networks with 100000 nodes (1 million edges) in about 1 minute on a commodity laptop with an i7 processor.

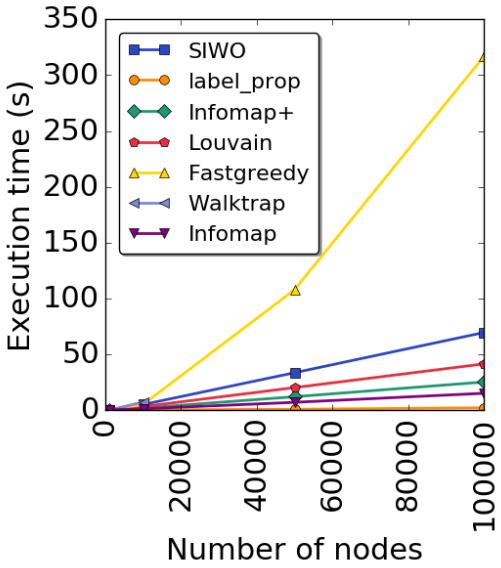

Figure 4.13: Execution time (average over 10 graph realizations) of SIWO and contenders on networks with varying sizes and the mixing parameter equal to 0.3. Walktrap fails for more than 10000 for lack of memory.

Chapter 5

Conclusion

5.1 Summary of Contributions

The main objective of this dissertation is to develop a community detection algorithm that has consistent good performance on all complex networks regardless of the size of communities. To this end, our proposed algorithm is less sensitive to the resolution limit and the field of limit compared to other state of the art algorithms while it is essentially as fast as the most efficient algorithms. The main results and contributions of this dissertation are summarized in the following.

1. We proposed an edge weighting scheme to differentiate between the links that run between the true communities (weak links) and the links that reside within the communities (strong links).

2. We proposed a novel approach in creating objective functions for community detection tasks which encourages adding strong links to the communities while avoiding weak links. We created an objective function called SIWO based on this approach.

3. We proposed a formal definition of the notion of community. A community is loosely defined as a dense subgraph that is sparsely connected to the rest of the graph. However, there is no globally accepted formal definition of community.

4. We proposed a general framework for detecting communities; this frame-

work can be applied to optimize not only our objective function (SIWO) but also other objective functions. After optimizing the objective function, our framework uses the formal definition of community to lead the process of community detection. This framework ensures that all the detected communities comply with the definition of community. One can also change the definition of community without changing the objective function within this framework. Since there is no globally accepted formal definition of community, it is not possible to integrate the objective function with the definition of community. Basically, by optimizing the objective function, dense groups are detected in the graph and these groups do not necessarily comply with the definition of community. Therefore, we developed this framework in a general way such that one can change either the objective function or the definition of community or both of them.

5. Our extensive experiments using both small and large networks (up to 1 million edges) confirm that our algorithm is consistent, effective and scalable for networks with either large or small communities demonstrating less sensitivity to the resolution limit and field of view limit that most community mining algorithms suffer from. Our experiments show that the application of the third step of our framework (the step that ensures that the communities comply with the definition of community) over one of the state of the arts algorithms improves the performance of the algorithm.

5.2 Limitations and Future Perspectives

An interesting direction is to generalize SIWO to attributed network. In some real networks there are two sources of information that one can use to find the communities: 1- the topological structure (the connections between the nodes) and 2- the attributes associated with the nodes. One interesting direction is to generalize SIWO approach such that it considers both topological information and node attributes to find communities. The current approach only

considers the connections between nodes. Another generalization that could be considered for the SIWO algorithm is to adapt it for directed networks and overlapping communities. The current approach ignores the direction of links and partitions the network into disjoint communities.

References

[1] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 U.S. election," in *Proceedings of the 3rd International Workshop on Link Discovery*, 2005, pp. 36–43. 1, 3, 36

[2] A. Anagnostopoulos, R. Kumar, and M. Mahdian, "Influence and correlation in social networks," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 7–15. 2

[3] A. Arenas, A. Fernandez, and S. Gomez, "Analysis of the structure of complex networks at different resolution levels," *New Journal of Physics*, vol. 10, no. 5, p. 053 039, 2008. 20

[4] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999. 1

[5] M. J. Barber and J. W. Clark, "Detecting network communities by propagating labels under constraints," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 80, no. 2, 2009. 13

[6] E. R. Barnes, "An algorithm for partitioning the nodes of a graph," *SIAM Journal on Algebraic Discrete Methods*, vol. 3, no. 4, pp. 541–550, 1982. 3, 6

[7] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, P10008, 2008. 9, 19, 37

[8] B. Bollobás, *Modern graph theory*, ser. Graduate texts in mathematics. Springer, 1998. 17

[9] L. Bottou, "Stochastic gradient learning in neural networks," in *In Proceedings of Neuro-Nîmes. EC2*, 1991. 15

[10] J. Chen, J. Fagnan, R. Goebel, R. Rabbany, F. Sangi, M. Takaffoli, E. Verbeek, and O. Zaïane, "Meerkat: Community mining with dynamic social networks," in *IEEE International Conference on Data Mining Workshops*, 2010, pp. 1377–1380. 39

[11] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 066 111, Dec. 2004. 9, 19, 37

[12] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri, "Feedback Effects between Similarity and Social Influence in Online Communities," in *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 160–168. 2

[13] N. T. Csardi G, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006. 37

[14] S. Emmons, S. Kobourov, M. Gallant, and K. Börner, "Analysis of network clustering algorithms and cluster quality metrics at scale," *PLOS ONE*, vol. 11, no. 7, pp. 1–18, 2016. 37

[15] J. Fagnan, A. Abnar, R. Rabbany, and O. R. Zaïane, "Modular Networks for Validating Community Detection Algorithms," *ArXiv e-prints*, Jan. 2018. arXiv: 1801.01229. 16

[16] S. Fortunato, "Community detection in graphs," *Phy. Rep.*, vol. 486, no. 3-5, pp. 75–174, 2010. 7, 53

[17] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007. 3, 11, 19, 31, 32

[18] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics Reports*, vol. 659, pp. 1–44, 2016. 16

[19] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002. 3, 8, 36

[20] D. Hric, R. K. Darst, and S. Fortunato, "Community detection in networks: Structural communities versus ground truth," *Physical Review E*, vol. 90, no. 6, p. 062 805, 2014. 40

[21] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, Dec. 1985. 16, 17, 39, 47

[22] R. Kanawati, "Seed-centric approaches for community detection in complex networks," in *Proceedings of the 6th International Conference on Social Computing and Social Media*, 2014, pp. 197–208. 11

[23] T. Kawamoto and M. Rosvall, "Estimating the resolution limit of the map equation in community detection," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 91, no. 1, p. 012 809, 2015. 19

[24] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970. 3, 6

[25] R. R. Khorasgani, J. Chen, and O. R. Zaïane, "Top Leaders Community Detection Approach in Information Networks," *Proceedings of the 4th Workshop on Social Network Mining and Analysis*, pp. 1–9, Jul. 2010. 4, 11

[26] J. M. Kumpula, J. Saramäki, K. Kaski, and J. Kertész, "Limited resolution in complex network community detection with potts model approach," *The European Physical Journal B*, vol. 56, no. 1, pp. 41–45, 2007. 20

[27] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 78, no. 4, pp. 1–5, 2008. 16, 43

[28] C. Largeron, P. N. Mougel, O. Benyahia, and O. R. Zaïane, "Dancer: Dynamic attributed networks with community structure generation," *Knowledge and Information Systems*, vol. 53, no. 1, pp. 109–151, Oct. 2017. 16

[29] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, pp. 1–40, Mar. 2007. 1, 3

[30] Z. Li, S. Zhang, R.-S. Wang, X.-S. Zhang, and L. Chen, "Quantitative function for community detection," *Physical Review E*, vol. 77, no. 3, p. 36 109, 2008. 20

[31] M. McPherson, L. Smith-Lovin, and J. M. Cook, *Birds of a Feather: Homophily in Social Networks*, 2001. 2

[32] M. Meilă, "Comparing clusterings — an information based distance," *Journal of Multivariate Analysis*, vol. 98, no. 5, pp. 873–895, 2007. 17

[33] M. Newman, "Assortative mixing in networks," *Physical Review Letters*, vol. 89, no. 20, p. 208 701, Oct. 2002. 1, 2

[34] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006. 1

[35] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026 113, Feb. 2004. 3, 4, 7, 9, 17, 19, 21

[36] M. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 69, no. 6, p. 066 133, Jun. 2004. 9, 19

[37] L. Peel, D. B. Larremore, and A. Clauset, "The ground truth about metadata and community detection in networks," *Science Advances*, vol. 3, no. 5, e1602548, 2017. 40

[38] P. Pons and M. Latapy, "Computing Communities in Large Networks Using Random Walks," in *Computer and Information Sciences*, 2005, pp. 284–293. 11, 37

[39] R. Rabbany, M. Takaffoli, J. Fagnan, O. R. Zaïane, and R. J. G. B. Campello, "Relative validity criteria for community mining algorithms," in *Encyclopedia of Social Network Analysis and Mining*, 2014, pp. 1562–1576. 17, 18

[40] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences*, vol. 101, no. 9, pp. 2658–63, Mar. 2004. 4, 8, 25

[41] N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 76, p. 036 106, 2007. 13, 37

[42] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Physical Review E*, vol. 74, no. 1, p. 16 110, 2006. 20

[43] M. Rosvall, D. Axelsson, and C. T. Bergstrom, "The map equation," *European Physical Journal: Special Topics*, vol. 178, no. 1, pp. 13–23, 2009. 4

[44] M. Rosvall and C. Bergstrom, "Maps of Random Walks on Complex Network Reveal Community Structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008. 10, 19

[45] M. T. Schaub, J. C. Delvenne, S. N. Yaliraki, and M. Barahona, "Markov dynamics as a zooming lens for multiscale community detection: Non clique-like communities and the field-of-view limit," *PLOS ONE*, vol. 7, no. 2, e32210, 2012. 4, 11, 19, 40

[46] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008. 3

[47] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, Mar. 2003. 16, 39

[48] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998. 1, 24

[49] *Webmapping of science and society actors in europe, final report*, `www.eurosfaire.prd.fr/7pc/documents/1274371553_finalreporteurosis3_1.doc` (2018-06-01). 1, 3, 41

[50] Z. Yakoubi and R. Kanawati, "Licod: A leader-driven algorithm for community detection in complex networks," *Vietnam Journal of Computer Science*, vol. 1, no. 4, pp. 241–256, Nov. 2014. 4, 11

[51] Z. Yang, R. Algesheimer, and C. J. Tessone, "A comparative analysis of community detection algorithms on artificial networks," *Scientific Reports*, vol. 6, no. 30750, 2016. 37

[52] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 555–564. 1, 3

[53] W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, pp. 452–473, 1977. 3, 30, 36

[54] P. Zhang, "Evaluating accuracy of community detection using the relative normalized mutual information," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2015, no. 11, P11006, 2015. 16