UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Scalable Community Detection for Social Networks

Arnau Prat Pérez

PhD Thesis in Computer Architecture
by the Universitat Politècnica de Catalunya

Advisor: Josep Lluís Larriba Pey
Co-Advisor: David Domínguez Sal

Barcelona, Catalonia 2015

Acknowledgements

I would like to specially thank to all the people that has supported me in achieving this enormous goal, specially my parents Toni and Elena for all the love, education and guidance they have given to me since I was born, with out them this achievement would have been impossible. To my brother Adrià, for all the good moments we have enjoyed together, and for always being a reference for me. To my girlfriend Raquel for bringing such amount of happiness and joy to my live. To my grandmother Montserrat, but also grandparents Josep and Joan, and grandmother Ramoneta, who are no longer with us. Special thanks to my uncle Ventura and Manel and my aunts Ester and Contxita, and my cousins Xell, Judit, Joan and Eva.

I would also like to thank all my friends, specially Joan, Xavi M., Xavi A., Fran, Bruno i Mireia, Víctor, David and Jessica, Andreas, to all my online friends with whom I have played an endless amount of hours, specially Javi, Adrián, David, Álvaro, Dario, Jesus and Ferran. Also thanks to all the present and prior members of DAMA-UPC and Sparsity, specially Mike, Xavi S., Cesc, Dámaris, Norbert, Víctor M., Ariel, Mathew, Jordi, Carlos, Ricard and Jordi Nin.

I would also like to thank my advisors Larri and David D. for their unvaluable guidance during all the thesis, Josep Maria Brunat for its contributions on the mathematical proofs, as well as Pedro, Andreas and Panayiotis for their warm welcome during my stay in Cyprus and their key contributions and support to my work about triangle counting. I would also thank all the people from the LDBC team, specially Alex, Andrey, Duc, Moritz, Peter and Orri, for all the things I can learn from them everyday.

Finally, I would also thank Generalitat de Catalunya for their SGR-890 grant as well as the Ministerio de Ciencia y Tecnologia for the various grants given to DAMA-UPC during my thesis, which were key for its development. Also to Oracle Labs for they financial support to many of the projects related to my work, as well as the European Comission for their funds to the LDBC, Tetracom and HiPEAC projects.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many real world systems and problems can be intuitively modeleded as graphs (or networks). A graph representation simplifies their analysis, allowing to better understand how the different entities involved in the system interact with each other. The list of systems that can be modeled as a graph is endless: social networks, where vertices represent persons and edges connect friends; the Internet, where vertices represent web sites and edges correspond to the url links connecting one page with another; router networks, where vertices represent routers and edges their physical connections; or protein-protein interaction networks, where vertices represent proteins and edges connect those with similar metabolic functions, just to cite a few of them.

One characteristic commonly observed in real graphs is that they are structurally organized into the so called communities or clusters [16, 31, 43]. Communities are groups of entities more densely connected among them than with other vertices not pertaining to the community, and emerge naturally as a consequence of the dynamics that drive the formation of the network. For instance, in a social network, a community might contain people with similar interests or people who work for the same company, as they have a higher probability to become friends than being connected with people not having anything in common.

The identification of communities based on the underlying structure of the network has become a hot research topic during the last decade because of the amount of applications. On the one hand, community detection is a tool that helps to better understand how complex networks are structured, which is

fundamental for the study of, for example, how epidemics spread and how to control them [59]. Being able to identify those persons connecting communities can be of high importance to control the spread of a disease. Also, persons within a community have a larger probability to be infected if one of them already is. Being able to effectively detect these persons and isolate them is also crucial to apply preventive measures. Similarly, in information networks communities help us to understand which members are more influential and can potentially control the information that others in the same community can access. The weak ties theory [18] states that the few links between members of different communities are more important for the spread of information than the internal links in a community. Entities connected by weak ties have a high control over what information is transmitted, and the presence of such links may indicate a weak structural point in a system such as a computer network.

On the other hand, detecting communities allows inferring hidden information from the network's structure. Missing edges between subjects belonging to the same community indicate potentially similar entities willing to be connected in the near future (link prediction). These have applications in recommendation systems (e.g. in social networks or product co-purchasing networks) [62], for targeting marketing campaigns, for finding proteins with similar metabolic functions in a protein-protein interaction network [9] or to identify spam in email and web networks [8, 34]. Finally, communities can be used to ease the visualization of large networks in data exploration tools, by summarizing the structure of the network and reducing the complexity of its representation [12].

The amount of literature about community detection is huge. Most of the existing work aims at finding what are known as disjoint communities, that is, communities which do not overlap (do not share vertices). More concretely, the most widely accepted metric to detect disjoint communities is *Modularity* [37]. Modularity measures how relevant is the internal edge density of a set of vertices compared to that observed in a random graph with the same degree sequence (the null model). Due to its popularity, many algorithms based on its maximization have been proposed following different strategies, like greedy techniques [7], spectral theory [38], simulated annealing [19] or extremal optimization [13].

However, it has been shown that modularity has several problems. First, modularity has a resolution limit, meaning that it is not able to discern

communities smaller than a certain size, a size that depends on the total size of the network. This means that the larger the graph, the less effective modularity is [16]. Second, modularity has a paradoxical behavior, in the sense that it is easier for it to detect communities which are worsely defined than those that are better defined. Finally, maximizing modularity can lead to sets of vertices without an appreciable community structure [6]. Therefore, even though modularity works well under some circumstances, under other circumstances it provides undesirable results.

Besides those methods based on modularity maximization, we find in the literature many other approaches. Some of them are those based on performing random walks. The rationale is that when performing a random walk, the probability of moving to a member of the same community is larger than moving to a different community. *Walktrap* [45] and *Infomap* [58] are examples of algorithms that fall into this category.

Finally, another popular family of algorithms, with an increasing amount of popularity due to their simplicity and scalability, are those based on label propagation [54]. In label propagation, a unique label is initially assigned to each vertex of the graph. By means of an iterative process, each vertex acquires the label most seen in its neighbors, until the process converges where no vertex changes its label. The problem with these approaches is that they are affected by label epidemics, meaning that some labels plague the network leading to very large and meaningless communities.

In general, existing community detection algorithms suffer from several problems: First, they are typically tailored after the informal definition of communities (sets of vertices more internally connected than externally), taking edges as sets but ignoring the structures that form the community either internally or externally. These algorithms and metrics aim at being generic, but ignore what characterize the communities of a given domain. As a consequence, existing algorithms fail to correctly capture the notion of a community under certain circumstances (for instance, when the graph is large for the case of modularity), and end up finding groups of vertices with a meaningless community structure. Second, existing work is not designed with parallelism in mind to perform well on current multi-core architectures, hindering their possibilities to scale to larger graphs.

In this thesis, we propose a new methodology for community detection design called *domain specific community detection*. This consists of defining a set of *structural* and *behavioral* properties any community detection metric or algorithm for a given domain should fulfill. On the one hand, structural properties are those that define how the the structural characteristics those communities found by an algorithm or those best ranked by a metric, should look like. That is, what internal and external structures communities should contain and form in order to be considered communities. On the other hand, behavioral properties are those properties that specify how a metric or algorithm should behave under different circumstances. For instance, how the metric/algorithm deals with graphs of different sizes, or whether it limits or not the growth of the communities.

In our particular case, we focus on community detection for the specific domain of social networks. From an analysis of the problems and drawbacks of existing metrics and algorithms when it comes to properly detect the communities in a social network, we identify a set of three structural properties we think a community detection metric for social networks should fulfill:

- **Internal Structure Sensitive**: The quality of a community should not depend only on the amount of internal edges but how these are internally structured, forming structures relevant for the application domain.
- **Bridge Resistant**: A good community should never contain a bridge, as this is clearly a weak point in the community structure.
- **Cut Vertex Resistant**: Similarly, a community detection metric for social networks should be resistant to containing vertex cuts.

Similarly, we identify three behavioral properties we think any community detection metric for social networks should fulfill:

- **Scale Independent**: A community detection metric should be scale independent, that is, it should be robust and accurate regardless of the size of the graph.
- **Linear Community Cohesion**: The amount of connections between a vertex in a community and the other vertices of the same community should scale linearly with the size of the community.

- **Adaptive**: A community detection metric for social networks should adapt to the heterogeneous nature of the graph, and be able to identify relevant regions with an appreciable community structure compared to their surroundings. Thus, it should consider both the internal and the external connectivity of the community.

Based on the aforementioned properties, we design a new community metric called *Weighted Community Clustering*(WCC) aimed at finding communities in social network like graphs. WCC is based on taking the triangle as the basic motif that indicates the presence of an appreciable structure within vertices in the graph. Triangles emerge naturally in social networks due to the dynamics of link formation, which are influenced by the Homophily principle [33], which states that similar people have a higher probability to be connected. We formally prove that maximizing WCC fulfills the properties desired for a community detection metric for social networks. Moreover, as a side outcome of our formal analysis, we theoretically find the detectability threshold (the point at which WCC is able to discern the communities of a graph), giving more insight to the actual behavior of WCC, and we analytically show that WCC does not suffer from the so called community detection paradox. Finally, we experimentally show that communities with good WCC exhibit good characteristics using a set of statistical indicators in an extensive experimental analysis using real graphs.

We also propose *Scalable Community Detection*(SCD), a community detection algorithm based on WCC maximization, designed to scale on SMP machines. We show that SCD achieves the best quality among all the existing algorithms in the state of the art, when it comes to find communities in real social network like graphs with ground truth communities created from the network's metadata. Moreover, we also show that SCD achieves close to linear scalability on SMP machines, becoming the one of the fastest community detection algorithm developed so far, and being the fastest one when using all the cores in a SMP machine. In practice, we are able to process the Friendster graph, with almost 2 billion edges and more than 100 million vertices, in about 7 seven hours with an unprecedented quality using a commodity server with 8 cores.

In order to better understand the characteristics of ground truth communities, and how these compare to the traditional community definition, we performed

a study of the distributions of the ground truth communities using several structural indicators. We observe that ground truth communities are far from the traditional community concept, not being isolated from the rest of the graph and not showing a so prominent internal edge density. However, we also observe that structures such as the triangle are considerably present in those ground truth communities. These results suggest that more complex structures such as the triangle, are important to discern the strong relations than the weak ones. We finally compare the distributions observed in ground truth communities to those output by synthetic graph generators, showing that some of the most widely used synthetic graph generators for community detection benchmarking generate communities significantly different from those observed in reality.

Finally, we have studied the problem of triangle counting in a modern architecture such as the Intel Single-chip Cloud Computer [32]. The interconnection network is one of the bottlenecks that limits the scalability of many-core processors. As such, computer architects are thinking of new architectures with simpler cache hierarchies with incoherent caches that simplify the design of such processors. As a downside, this leads to more complex programs and more responsibility to the programmer when creating correct software. In our study we have revealed that the Producer-Consumer programming model is a suitable model for architectures with non-coherent caches and on-chip memory buffers, able to dynamically adapt to the load of the application and to get the most out of the available resources of the machine. We have also shown that modern social network applications such as triangle counting, which are memory bound and exhibit very non-local memory accesses, can greatly benefit from such architectures when implemented using the Producer-Consumer programming model.

The rest of this document is structured as follows. In Chapter 2, we introduce the background and related work in community detection. In Chapter 3, we describe the design of WCC and in Chapter 4, we introduce SCD. In Chapter 5, we show the experimental results of our evaluation and in Chapter 6 we introduce a study of the real community distributions. In Chapter 7, we show our study on the implementation of triangle counting on an experimental many-core architecture and finally, in Chapter 8, we conclude the work and give some guidelines for future research.

1.1 Contributions

In this thesis we make the following contributions:

- We have proposed a new *domain specific community detection* design methodology.
- We have proposed a novel community detection metric (WCC) based on triangle counting (CIKM 2012, TKDD).
- We have introduced a set of community detection metric properties we think should be fulfilled by any community detection metric for social networks (CIKM 2012, TKDD).
- We have experimentally evaluated WCC and compared different state of the art algorithms based on it. We show that WCC is able to correctly rank communities robustly, something not done by existing metrics like Modularity or Conductance (CIKM 2012, TKDD).
- We have introduced the concepts of structural isolation and structural intraconnectivity and overall, presented a novel methodology to design community detection metrics based on the notion of structure (TKDD).
- We have performed an extensive Detectability analysis of WCC, giving a better insight of how does WCC behaves, and also showing that it is not affected by the so called community detection paradox (TKDD).
- We have proposed Scalable Community Detection (SCD) for SMP machines, which is an algorithm based on WCC. The algorithm has been designed with parallelism in mind, to take advantage of the resources of current architectures (WWW 2014, TKDD).
- We have evaluated the performance of SCD and compared it to the most popular state of the art algorithms, showing that SCD is able to scale to billion edge graphs, and outperforms current state of the art methods both in terms of quality and execution time (WWW 2014, TKDD).
- We have analyzed the structure of ground truth communities found in several real graphs and shown that they are not so well defined and

isolated as one would expect, but at the same time have a large number of triangles with a small number of bridges (GRADES 2014).

- We have compared the community structure of synthetically generated graphs that output a community structure, the LDBC Datagen and LFR graph generators, to real graphs with ground truth communities. We have shown that the LDBC Datagen graph generator is able to produce graphs that better reproduce the characteristics of those communities found in real graphs. (GRADES 2014).
- We have studied the viability of the Producer-Consumer model on future many-core architectures without coherent caches, using the problem of triangle counting (ARCS 2013).

Chapter 2

Related Work

Community detection is a broad topic that has attracted the interest of many researchers over the last decade because of its applications in diverse domains such as biology, epidemics, social network analysis, marketing, recommendation systems etc., to just cite a few of them. Due to its complexity and the lack of a strict formalism of what a community is, the topic is subject to many interpretations and subtleties. As such, we start this section by clarifying some concepts to better delimit the spectrum of the problem covered by this thesis. Then, we make a review of the most widely used metrics and algorithms for community detection, to give the reader a broad view of the dimension of the topic. For a more comprehensive and detailed state of the art review, please refer to the following surveys [16, 31, 43].

Communities are informally defined as subsets of vertices of a graph with more internal connections than connections with vertices from other communities, and this is the definition most existing metrics are based on. Most of the work on community detection has traditionally focused on the *disjoint* community detection problem. The problem consists in classifying the vertices of a graph into disjoint cohesive sets, that is, each vertex can belong to just one community. Figure 2.1 shows a graph with three disjoint communities. Disjoint community detection is similar to the more traditional graph partitioning problem, but differs from the later in the fact that the number of communities and their size are driven by the structure of the network itself instead of being passed as a parameter by the user. Figure 2.2 shows the same graph of Figure 2.1 partitioned into two groups of size ten that minimize the cut, as in typical graph partition algorithms

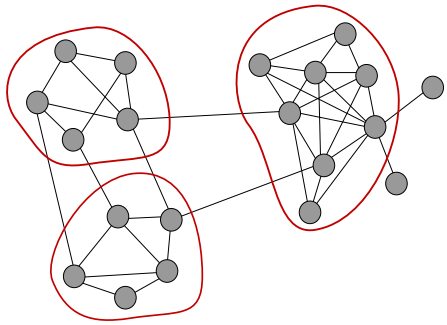


Figure 2.1: A classification of the vertices of a graph into communities

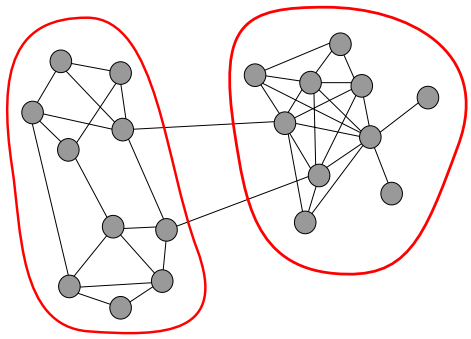


Figure 2.2: A graph partitioned into two groups of size ten.

Application wise, graph partitioning is mostly motivated by the necessity of distributing data and load across the computing nodes of a distributed system [22], while community detection is used in data mining [63], visualization [12, 57] and knowledge discovery [20]. However, community detection has also been used to obtain smarter ways to layout data in memory [46].

Besides disjoint communities, other types of communities exist that have recently attracted the interest of the research community. One of these types are *overlapping* communities, which are observed in many types of real networks and applications where it is natural to think of entities that can belong to more than one community. Figure 2 shows an example with three communities that overlap on a set of vertices. In general, detecting overlapping communities is a more complex problem than detecting disjoint communities. First, it is

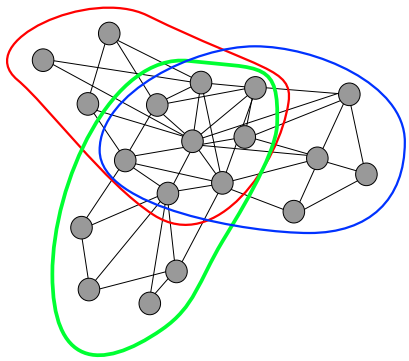


Figure 2.3: Three overlapping communities that overlap in a region.

not clear what characterizes the overlap between a set of communities, and which is the point when a set of overlapping communities ought to be merged into a single and larger one. Some work about trying to better understand what characterize the overlaps has been done by [65]. Second, the number of possible *covers* (classifications of vertices into subsets in such a way that can overlap) in a graph is much larger than the number of possible partitions, thus making the problem more computationally expensive. Nevertheless, the simpler disjoint community detection problem is still far from being solved and existing solutions still suffer from many issues in order to securely move to the more complex overlapping counterpart. Finally, there also exist other types of communities such as 2-mode communities, that is, communities that form bipartite subgraphs of two types of different entities [68].

Typically, most of the work on community detection is restricted to undirected graphs without weights on the edges. However, for some problems it is reasonable to think of links not to be equally strong, which is typically represented as a weight between zero and one representing the strength of the link (unweighted graphs can be seen as graphs where all edges have a weight of one). As such, some approaches have included this notion of strength [15]. Similarly, some researchers have worked on what they call fuzzy communities [35]. In this context, the notion of belonging to a community or not is not binary but a real value (weight) or probability between zero and one. Finally, some researchers have worked on detecting communities in directed networks [15].

In general, community detection methods can be can classified into two different categories independently of the type of communities they find: those based on finding graph partitions that maximize objective functions or metrics that tell how good the partition into communities is, and those based on detecting the communities as a result of running an algorithmic process. The problem with algorithmic based community detection is that, in general, it is more difficult to theoretically model the behavior of the algorithm and as a consequence, to provide formal guarantees of the expected result. The work presented in this thesis falls into the first category, and aims at detecting disjoint communities in undirected graphs by maximizing the proposed metric called WCC.

2.1 Metric based community detection

Most of the existing work in community detection falls into this category. Formally, given a graph $G(V, E)$, the goal is to find a partition or classification of vertices into subgroups $\mathcal{P} = \{C_1, \ldots, C_n\}$ such that for all pairs $< i, j >$, $C_i \cap C_j = 0$. Then, given an objective function $f(\mathcal{P})$ which measures the quality of a partition of a graph into communities, the goal is to find:

$$\arg\max_{\mathcal{P}} f(\mathcal{P}),$$

that is, that partition into communities with the best quality. The amount of variants of f found in the literature is vast, thus we cover the most relevant approaches. Since most of the existing work in community detection defines metrics at a community level instead of a partition level, to compute the quality of a partition the average quality of all communities using the given metric is typically reported. Therefore, unless contrarily stated, the default way to combine in this thesis such community defined metrics will be using the average. Finally, before introducing the different metrics, we start with the definition of some terms:

- Let $n = |V|$ be the number of vertices in a graph.
- Let $m = |E|$ be the number of edges in a graph.
- Let C_i be the community of vertex i.

- Let k_i be the degree of vertex i.
- Let k_i^{in} be the degree of vertex i pointing to vertices in C_i.
- Let k_i^{out} be the degree of vertex i pointing to vertices outside C_i.
- Let K_s be the degree of community s.
- Let K_s^{in} be the internal degree of community s.
- Let K_s^{out} be the external degree of community s.
- Let E_{st} be the edges between communities s and t.
- Let $\delta(C_i, C_j)$ be the Kronecker delta function (1 if and only if $C_i = C_j$, 0 otherwise)
- Let A be the adjacency matrix of the graph.

2.1.1 Modularity

The most popular family of disjoint community detection algorithms is composed by those based on maximizing the *modularity* metric [37]. Given a partition, modularity measures how relevant is the internal edge density of the subsets that composed the partition, compared to that observed in a random graph with the same degree sequence (the null model). Modularity is defined as follows:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \delta(C_i, C_j)$$

Due to its popularity, most of the existing community detection methods are guided by modularity. One of the most widely used modularity maximization algorithm is the Louvain method [7], which is based on constructing a dendogram guided by modularity maximization. The algorithm starts with a partition only containing communities with a single vertex and subsequently merging them by generating a new graph made out of vertices representing the communities from the previous step. Louvain does not only obtain good results (i.e. is able to find partitions with a large modularity score), but it

is also very fast, being able to run on very large graphs. Modularity can be also optimized by means of spectral based methods, more concretely, by using the eigenvalues and eigenvectors of an special matrix called the modularity matrix [38]. Using this matrix, one can find the bisection of the graph with the best modularity. Then, by subsequently applying the procedure on each of the bisections, one can find the partition with an optimal modularity. Other approaches to maximize modularity are those based on Simulated Annealing [19] or Extremal Optimization [13], and in general many algorithmic methods use modularity as their finishing criteria.

However, in [16], the authors show that modularity has a resolution limit. Basically, this limit means that modularity cannot detect communities if these are smaller than a certain threshold, which depends on the total size of the network. Then,the larger the network, the larger the threshold and thus the loss of resolution. If S and T are two communities inside a much larger network of size M, then S and T will be detected if and only if:

$$K_s^{in} K_t^{in} > 2mE_{st}. \tag{2.1}$$

We see that condition 2.1 depends on the total number of edges of the network. This means that as M increases, the larger the difference between internal degrees of the S and T and the amount of edges connecting them needs to be, reaching a point where even if S and T are two complete subgraphs with a single edge connecting them, it is not a sufficient condition for them to be detectable.

In order to overcome this limitation, several multi-resolution methods have been proposed. These methods are based on tuning the importance or contribution of the null model into the modularity formula, following different strategies [17,55]. However, these methods suffer from the fact that in order to detect the small communities, the big ones are shattered into smaller subcommunities, even if these are complete subgraphs (cliques). This means that these methods do not completely solve the problem, as the community definition still depends on the size of the network [64]. Altogether makes modularity a metric which is not *scale independent*, that is, his behavior depends on the scale of the graph. The scale independent property is discussed in more detail in Chapter 3.

Also, modularity is affected by the so called paradoxical behavior [52]. This means that for modularity it is easier to identify communities when these are not well defined than when these are very well defined. Finally, in [6] the author shows that structures with a very low community like structure such as trees, can have a very large modularity. These can make modularity a metric that finds non structurally relevant communities for some applications. With all these issues, even though modularity might work well in practice, it is not a robust metric and under some circumstances it fails to identify capture the communities.

2.1.2 Other community detection metrics

Besides modularity, we can find other metrics, which are classified depending on whether they look at the internal connectivity of the community, the external connectivity of the community or both.

Internal Connectivity: This category is formed by those metrics based on looking at how the vertices of a community are connected internally. Here we find metrics such as the *average degree*:

$$AverageDegree(S) = \frac{1}{|S|} \cdot \sum_{i \in S} k_i,$$

the *internal edge density*, which is the ratio of the internal number of edges divided by the total possible edges [53]:

$$EdgeDensity(S) = \frac{K_s}{|S| \cdot (|S| - 1)},$$

and the *triangle participation ratio* (TPR), which is the fraction of vertices in the community that closes at least one triangle with two other vertices in the community [65].

External Connectivity: This category is formed by metrics that only look at the external connectivity of the community. Here, we find the *cut ratio*,

which is the the ratio between the actual number of edges pointing outside the cluster and the total possible number of edges pointing outside the cluster [15]:

$$CutRatio(S) = \frac{K_s^{out}}{|S| \cdot (|V| - |S|)},$$

and the *expansion*, which is the ratio between the number of edges pointing outside the cluster and the size of the cluster (the lower, the better) [53]:

$$Expansion(S) = \frac{K_s^{out}}{|S|}.$$

Internal/External connectivity: This type of metrics encompass those functions that look both at the internal and the external connectivity of the community. A very popular metric that falls into this category is *Conductance* [21]. Conductance measures how isolated from the rest of the graph is a set of vertices, by counting the number of edges going outside a set over the total edges of the set (the lower, the better). Given a community S, then the conductance is measured as:

$$Conductance(S) = \frac{K_s^{out}}{K_s}.$$

Similarly, *Flake ODF*, which stands for *Flake Out Degree Fraction*, is the average fraction of vertices in the community that have fewer edges pointing inside than outside of the community (the lower, the better) [15]:

$$FlakeODF(S) = \frac{|\{x \in S : k_x^{in}) < k_x^{out})\}|}{|S|}.$$

In general, looking at only the internal or the external connectivity of a community is not a good idea, as only a partial view of the community is taken into account. Finding communities is about finding structurally relevant regions in a graph, and therefore, both internal and external connectivity should be taken into account. Moreover, some of these metrics, like the expansion, conductance or FlakeODF, suffers from a maximization problem.

In general, since these metrics have the form of a fraction where the numerator measures the cut (number of edges going outside the community), all the graph in a single community gives the maximum value. In a previous study [67], the authors conclude that conductance and TPR are the two metrics that better capture the notion of a real community. However, these metrics still lack of formal guarantees when it comes to detect the communities in a social network, as we will discuss in Chapter 3.

The WCC metric (proposed in this thesis), falls into the category of metrics that take into account both for the internal and the external connectivity of the vertices in a community, but it does it in such a way that overcomes the observed limitations of existing proposals.

2.2 Algorithmic community detection

Algorithmic community detection consists of all those methods that detect communities as a result of an algorithmic process. Some of the most representative algorithms that fall into this category are those belonging to the random walks family. The rationale of random walks is that since communities are in general groups of vertices with a larger internal density than external, when one performs a walk over a graph randomly, the probability of traversing edges that connect vertices of the same community is larger than moving to another community. One of the main exponents of this family of algorithms is Walktrap [45]. Walktrap relies on the definition of a distance measure between vertices based on performing random walks, in such a way that the larger the number of random walks two vertices share, the closer the distance between them. Then, vertices are clustered based on this distance measure. In the original paper, the author construct a dendogram based on the distance measured, and the partition with the larger modularity is output. However, as the original authors suggest, this method can use any other community detection metric to construct the final partition, depending on the actual application. For this reason, we have decided to put Walktrap into the algorithmic category.

Another very popular random walk-based algorithm is Infomap [58], which is based on finding the most compressed codification to represent random walks in memory. The codification, is based on a partition of the graph into communities, in such a way that the more accurate this partition is, the smaller the memory footprint of the codification of a set of random walks. Infomap

is currently one of the most popular algorithms in the literature, and it has been shown that works very well using synthetically generated graphs of a few hundreds of thousands of edges. However, no formal guarantees have been given proving that it scales well to large graphs of millions of vertices, and as we will see in Chapter 5, its performance and quality decays for large graphs.

The Label Propagation Algorithm (LPA) [54] stands as another algorithm (and actually, family of algorithms nowadays) that has recently gained a lot of popularity due to their simplicity and scalability. In label propagation, vertices are initially assigned with a unique label. Then, by means of an iterative process, each vertex changes its label to that most popular label out of those own by its neighbors. The process continues until the process converges where no vertex changes its label. One of the interesting characteristics of this method that makes it different from other existing proposals, and therefore making it worth to mention, is its performance. LPA suits very well the vertex centric programming model of many scalable graph programming frameworks like Pregel [30] or Giraph [3], and also graphic processing units (GPUs). However, label propagation is not exempted to problems but suffers from well-known issues such as "label epidemic", where labels can "plague" the network leading to very large and meaningless communities. This makes label propagation unreliable as it does not guarantee a minimum quality of the communities found.

2.2.1 Algorithms for overlapping communities

Although this is not the aim of this work, we think it is worth to mention some of the most popular algorithms for detecting overlapping communities. One of the most widely used and older methods to detect this type of communities is the clique percolation method [11]. The clique percolation defines communities as k-clique chains. A clique chain is defined as a group of k-cliques that overlap on k-1 of their vertices. Since a clique can overlap with a clique chain on less than k-1 vertices, overlaps between communities emerge naturally. Clique percolation is the base method that form the community detection framework used in biological networks CFinder [1].

Link clustering [2] is another popular overlapping community detection algorithm, which instead of trying to cluster vertices, it tries to cluster edges. Given two edges that share a vertex, the Jaccard coefficient of the adjacency lists

of the non shared vertices is computed. Given this measure, a dendogram of edge relations is computed, placing those edges with a larger Jaccard measure closer to the leaves. Then, the dendogram is cut at that level that maximize a metric called partition density, and those edges under the same branch in the dendogram are clustered together, forming a community. Since this methods cluster edges, overlapping communities emerge naturally as a vertex can be contained in multiple edges, which can belong to different communities.

In [41], the authors propose a novel community detection metric for overlapping communities, based on a new operator they call the *directed Laplacian*. This operator operates on a graph representation of the community solution space (those possible communities that can exist), that is, a graph where each vertex is a community and two vertices are connected if and only if their size differ in just a single vertex (and the rest of vertices remain the same). Over this graph they define an *euclidean distance* based metric, which is then optimized.

Finally, a recent work [66] has modeled communities as tiles in an adjacency matrix. The novel observation is that these tiles, tend to be denser in the parts that they overlap, in contrast with the common believe of overlapping communities denser in the non overlapping part. Based on this observation, the authors propose a graph generation model based on overlapping communities called the Affiliation Graph Model (AGM). Based on this model, they propose BigClam, an algorithm based on non-negative matrix factorization that looks for the parameters of the model that explain an observed graph structure. Since this method is build on top of well known parallel matrix computation algorithms, they are able to scale up to graph of 100 millions of vertices without a problem.

2.3 Parallel and distributed community detection

There is some work on parallel community detection for SMP machines, mostly focused on parallel versions of known sequential community detection algorithms. In [29], the authors propose a parallel version of the Louvain algorithm, achieving an speedup of 16x on a machine with 32 threads. Similarly, in [56], the authors propose an agglomerative modularity optimization algorithm for the Cray XMT and Intel based ma- chines. The authors report that their solution is able to analyze billion edge graphs (100 Million vertices, 3.3 Billion edges) in 500 seconds. Finally, in [4] the authors propose RelaxMap, a parallel

version of Infomap based on relaxing con currency assumptions of the original method, achieving a parallel efficiency of about 70%. The same authors proposed in [5] a distributed version of Infomap. Finally, the label propagation method described above fits very well the vertex centric computing model, therefore it can be easily distributed using a graph programming distributed framework.

Chapter 3

Weighted Community Clustering

In this chapter, we describe the *Weighted Community Clustering* (WCC) [48, 49], a novel community detection metric for social networks. We start by discussing the limitations of existing metrics and algorithms when it comes to discover the communities in social graphs, and argument that new metrics need to be more domain specific to better exploit the particular characteristics of each type of network. Out of the discussion, we derive a set of *behavioral* and *structural* properties we think any socially oriented community detection metric should fulfill. These properties define how the metric should behave, as well as what type of structures the communities resulting from its maximization should contain or avoid.

Next, we formalize the problem of community detection and reduce it to the task of defining a function that measures the level of cohesion between a vertex x and a set of vertices S. Based on this formalization, we define WCC and back it up with an extensive formal analysis showing that it behaves as expected when optimized, not suffering from the limitations observed in existing approaches but fulfilling the desired properties.

3.1 Domain specific community detection

Traditionally, existing community detection metrics and algorithms have been designed to be generic, built around the informal community definition that defines communities as sets of vertices more densely connected internally than externally. In other words, they assume that communities are an ubiquitous

concept whose characteristics do not depend on the domain of the network analyzed. However, when we run existing community detection algorithms in social networks, we observe limitations that affect the quality of the produced results. For instance, *modularity* suffers from the well known resolution limit, which makes communities undetectable as the size of the graph increases. In the context of a social network, this is counter-intuitive, as the existence of a community should be independent of the size of the graph, and just depend on its local characteristics. In other words, a community metric for social networks should be *scale independent*. Similarly, we see that existing community detection metrics (e.g. modularity, conductance, etc.), take edges as a set instead of looking for more complex yet domain specific structures. As a consequence, they end up finding tree-like communities in social networks, where people are loosely connected without a significantly relevant structure among them. However, in a computer network, communities with a tree-like structure make sense as this kind of network is structured in a more hierarchical way. Therefore, a community detection metric ought to be sensitive to the *internal structure* of the community instead of just counting edges, and the particular notion of structure ought to be dependent on the actual domain of the network.

From these observations, we see that the informal community definition is too lax, leading to metrics with an undesired behavior. Thus, we think that we need to extend the community definition and link it to the actual domain of the applications where it is going to be used. With this objective in mind, we define a set of generic community detection properties that any community detection metric designed for social networks should fulfill. We classify them into two main categories: *structural* and *behavioral* properties. On the one hand, structural properties are those that refer to how structurally should or should not the communities found by maximizing a given metric be. In this category, we introduce, the *internal structure*, the *bridges resistant* and the *cut vertex resistant* properties. On the other hand, behavioral properties are those properties that specify how a metric should conceptually behave regardless of the structure of the community. In this category we have *scale independent*, *linear community cohesion* and *adaptive* properties. Properly defining these properties is very useful for two main reasons:

1. They set up a framework to design a robust community detection metric for social networks.

2. Provide the user with an insight into how the metric behaves, thus having enough information to decide whether the metric is suited or not for its application.

In the following sections we introduce these properties and argument why they are required for a socially oriented community detection metric. For other domains, these properties might be completely different.

3.1.1 Structural properties

3.1.1.1 Internal structure sensitive

One of the main characteristics of social networks and their communities is the significant presence of transitive relationships (triangles) among three vertices of the graph. The relevance of triangles in social networks has been confirmed in previous studies [36, 39, 60, 61] and models describing the growth of social networks give triangle closing as a key factor of network evolution [27]. In social networks, triangles emerge naturally as a consequence of the Homophily principle [33], which states that similar people are more likely to be connected, forming transitive relations among them, which also translates into networks with a significantly larger clustering coefficient than that expected in a random graph [36, 39].

Therefore, how the edges in a community are structurally connecting their vertices is important to determine whether these form a good community or not. The particular notion of structure then, depends on the actual domain of the network, which in the case of social networks one of such structures is the triangle. Therefore, we define the property as follows:

> *"A community detection metric is sensitive to the internal structure, if and only if it does not only take into account the number of internal/external edges, but also to how these connect vertices inside/outside the community, forming relevant structures for the specific domain"*

Typically, existing community metrics consider all the internal edges of a community equally important without taking into account whether they form structures or not. The reason is that many of the existing metrics are too tightly adhered to the informal community definition, which attempts to be generic and domain agnostic. This is the case of some of the most popular metrics such as modularity or conductance.

3.1.1.2 Bridge resistant

The connections in real graphs are known not to be local, they can connect distant vertices [28]. In a social network, people are typically connected to people with similar characteristics (e.g. their work mates, people they know from where they studied, family, people with similar interests, etc.), but also and less frequently, with people who are dissimilar and that they have met due to more sporadic interactions (i.e. random). A bridge is an edge that if it is removed from the induced subgraph of the community, it creates two separate connected components. A bridge is a very weak relation between two sets of vertices that are unrelated, because it only affects one member of each subset of vertices, and therefore, in a social network it clearly reveals two dissimilar subsets connected by one of such random links. Therefore,

> *"A comunity detection metric is bridge resistant if and only if resulting communities from its maximization never contain a bridge"*

Most of the existing metrics like modularity or conductance are not bridge resistant as they rank as good a community sets of vertices with a tree-like structure, thus containing bridges. Actually, as a consequence of its resolution limit, modularity is not bridge resistant, as pairs of two cliques in a chain of cliques connected by a single edge (bridge) are identified as a community.

3.1.1.3 Cut vertex resistant

In a social network, it is common to find situations where a person belongs to two or more different communities, such as that one formed by its football friends and the other formed by its family. In this particular situation, there is an overlap of two communities on that person. In the context of disjoint communities, however, if the two communities are not related enough, then

they should be split and the person placed into one of them (the more important one or relevant from the perspective of the person, as in the disjoint community detection problem a vertex can only belong to a single community). Note that it is not the aim of this work to consider the problem of overlapping communities.

In such a situation, the person within the induced subgraph formed by the two communities it belongs to, is what is known in graph theory a cut vertex. His removal from the induced subgraph separates the two induced communities into two (or more) connected components, in a similar way as bridges do. Cut vertices are weak links in a community, because there are no edges between the connected components the cut vertex is connecting. If the connected components have a strong internal structure, then it is more natural to split the induced subgraph into several communities. Therefore,

> *"A community detection metric is cut vertex resistant if and only if resulting communities do contain a cut vertex"*

3.1.2 Behavioral properties

3.1.2.1 Scale Independent

In a social network, the existence of a community (e.g. those persons interested in a given topic) is independent whether the network is formed by one or ten million people. Similarly, in a protein-protein interaction networks, the proteins in a community of protein have a large probability to have similar metabolic functions. Whether this is true or not, does not depend on whether there are one or ten million different proteins in the world. Therefore, a community detection metric or algorithm for social networks should work well at any scale and should not degrade as the size of the analyzed network increases. This means that the community definition should not depend on the size of the network, but only on local information. Otherwise, clearly defined communities would become undetectable as the size of the graph changes. Therefore, we define the property as follows:

> *"A community detection metric is scale independent if and only if it does not depend on the total size of the graph"*

Most of the existing community detection work does not formally guarantee to be scale independent. Actually, all those algorithms based on modularity maximization, which form the bulk of the literature, suffer from the so called resolution limit as explained in Chapter 2. This is because modularity computation depends on the total number of edges in the graph, and different alternatives aimed at solving this issue do not fix the problem entirely [64].

3.1.2.2 Linear community cohesion

In social networks, communities are formed by persons with a significant level of cohesion among them. This means that, as long as the size of the community increases, the number of connections between a person and its community has to increase proportionally to maintain the level of cohesion of the community. This simple restriction limits the community growth if there is not a significant cohesion among its members. For this reason, we define the following property:

> *"A community detection metric has a linear community cohesion if and only if the number of connections between a vertex and a community grows linearly with the size of the community"*

If it grew sublinearly, the larger the community, the easier it would be for a person to join that community with respect to the size of that community. On the other hand, if it grew faster than linear, the communities would have a maximum possible size, since after a certain point, the number of necessary links between a person and the rest of the community would be larger than the possible number of links.

Existing metrics such as Conductance, Flake ODF, Cut ratio or Expansion, do not fulfill this property. In these metrics, as long as the number of edges between communities is reduced, the value of the metric improves, thus they do not place any restriction on a minimum number of connections (just greater than zero) between a vertex and a community. More concretely, in such metrics a "hair" (a vertex connected to a community with a single edge) will always be included in that community regardless of its size.

3.1.2.3 Adaptive

In a social network, different communities might have different degrees of cohesion or structure, depending on the how tight is the relation between the members of the community compared with other people not belonging to it. This relation between internal and external connectivity is what determines whether a set of vertices is a community or not. Finding communities is about finding relevant regions in a graph, that is, sets of vertices whose structure deviates significantly from what it is expected in a given part of the network. This is what indicates that some relevant structure is there, and thus it is worth to be analyzed in more detail. Thus, we define the property as follows:

> *"A community detection metric is adaptive if and only if it establishes a relation between internal and external connectivity, and thus, adapts to the context of the community in the graph"*

Many existing community detection metrics (e.g. Edge Density, Cut Ratio and Expansion) are defined in such a way that they just look at either the internal or the external connectivity of a community, but not to both. In the first case, metrics end up missing sets of vertices that although they do not fulfill the internal criteria, they are so isolated from the rest of the graph that should be detected as a community. On the second case, these metrics end up missing groups of vertices that although they are not isolated enough to meet the criteria, their internal community structure is so strong that they should be considered a community as well.

3.2 Problem Definition

Given a graph $G = (V, E)$, the problem of disjoint community detection consists in classifying the $|V| = n$ vertices of the graph into q[1] non-empty pairwise disjoint cohesive sets, S_i for $1 \leq i \leq q$. We call those q sets a partition of V, i.e. $\mathcal{P} = \{S_1, \ldots, S_q\}$, in such a way that $S_1 \cup \cdots \cup S_q = V$.

The criterion to measure the degree of cohesion of each set is formally obtained by defining a metric, that is, a function f_s, that assigns a real number to each

[1] We assume that q is a value determined by the nature of the graph rather than an arbitrary value determined by the user.

subset S_i of V such that $0 \leq f_s(S_i) \leq 1$. A good/bad *community* is a set of vertices S with a value of f_s close to 1/0.

We define the cohesion $f_s(S)$ of a community S, as the average of $f_v(x, S)$ for each vertex $x \in S$, which measures the level of cohesion of vertex x with respect to the set S:

$$f_s(S) = \frac{1}{|S|} \sum_{x \in S} f_v(x, S). \tag{3.1}$$

That is, the quality of a community, depends on how cohesive are the individual vertices in the community with the community.

Similarly, we define the quality of a partition $\mathcal{P}$ by taking the weighted average of the value of the function on the sets S_i of the partition:

$$f(\mathcal{P}) = \frac{1}{n} \sum_{i=1}^{q} \left(|S_i| \cdot f_s(S_i) \right). \tag{3.2}$$

For a given graph and a given metric f in G, the goal is to obtain an *optimal partition*, that is, a partition $\mathcal{P}$ such that $f(\mathcal{P})$ takes a maximum value. We call the communities in an optimal partition the *optimal communities* of the graph. Therefore, following this formalization, the problem resides in properly defining $f_v(x, S)$ in such a way we encompass the properties introduced in Section 3.1.

3.3 The Weighted Community Clustering

The *Weighted Community Clustering* (WCC) is a community detection metric aimed at detecting communities in social networks, and designed to fulfill the properties discussed in Section 3.1. As such, WCC takes the triangle as the basic indicator of the presence of structure in a graph, since triangles are an structural indicator of the presence of a strong relation between three persons, as explained in Section **??**. Therefore, we consider two vertices of a graph to be structurally connected if they share at least one triangle. In other words, we take the presence of triangles to define the cohesion of x with respect to S, $WCC_v(x, S)$, as our particular implementation of $f_v(x, S)$ (Equation 3.1).

Let $t(x, S)$ be the number of triangles that vertex x closes with the vertices in a set S and by $vt(x, S)$ the number of vertices of S that form at least one triangle with x. Then, :

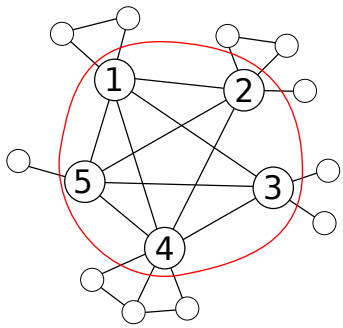
$$WCC_v(x,S) = \begin{cases} 0 & \text{if } t(x,V) = 0 \\ \underbrace{\frac{t(x,S)}{t(x,V)}}_{isolation} \cdot \underbrace{\frac{vt(x,V)}{vt(x,V) + |S \setminus \{x\}| - vt(x,S)}}_{intraconnectivity} & \text{if } t(x,V) \neq 0. \end{cases} \tag{3.3}$$

Note that $vt(x, V) + |S \setminus \{x\}| - vt(x, S) = 0$ implies that $S = \{x\}$ and $vt(x, V) = 0$. Then, condition $vt(x, V) + |S \setminus \{x\}| - vt(x, S) = 0$ is included in condition $t(x, V) = 0$.

Isolation: The left factor of $WCC_v(x, S)$ is the ratio of triangles that vertex x closes with set S, as opposed to the number of triangles that x closes with the whole graph. The left factor is maximized for a vertex x when S includes *all* the vertices that form triangles with x. Note that since a pair of vertices can build many triangles, the left term rewards the inclusion of the vertices that build more triangles with x. This left factor represents what we call as *structural isolation*, which measures how structurally isolated is a vertex from the rest of the graph if put in community S.

Intraconnectivity: The right factor is the ratio between the number of vertices in V that close at least one triangle with x, and the number of vertices in V that close at least one triangle with x plus the number of those in $S \setminus \{x\}$ not closing any triangle with x and another vertex $u \in S \setminus \{x\}$. The right term is maximized for x when S contains *only* vertices that do form at least one triangle with x and a third vertex $u \in S$. Similarly, the right factor represents what we call as *structural intraconnectivity*, which measures how structurally connected is a vertex x with the rest of vertices in S.

The two factors of $WCC_v(x, S)$ are finally combined with a multiplication because we want to maximize both structural isolation and intraconnectivity. If any of the two terms is zero, then the cohesion of the vertex with respect to

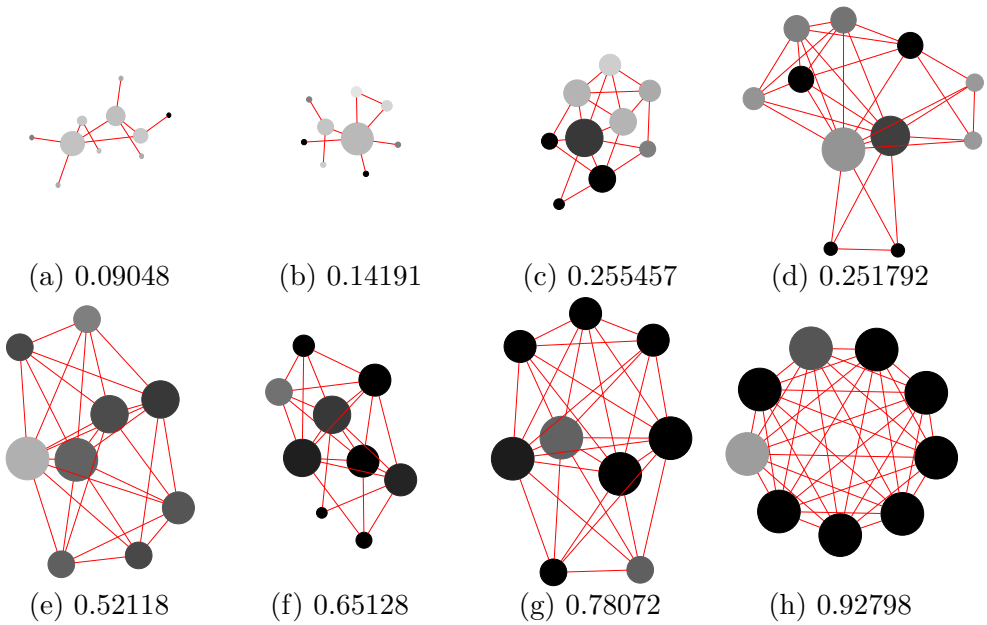


Vertex	$t(x,V)$	$t(x,S)$	$vt(x,V)$	$vt(x,S)$	$WCC_v(x,S)$
1	6	5	6	4	0.833
2	4	3	5	3	0.625
3	4	3	3	3	0.750
4	7	5	7	4	0.714
5	5	5	4	4	1.000

Figure 3.1: A community formed by five vertices, and the statistics of each vertex.

the set is zero. If we combined them with a sum, then a vertex x closing a single triangle with a set S, and being completely isolated from the rest of the graph, could have a fairly large level of cohesion with the set, regardless of the size of the set S, and therefore regardless of the structural itraconnection of that vertex with the set. This would make the metric not to have a *linear community cohesion*, and would be prone to bridges and cut vertexes. Also note that neither the left nor the right factors depend on the size of the graph. This is necessary for having a *scale independent* metric. Finally, analogously to $f_s(S)$ in Equation 3.1, we denote the quality of a community as $WCC_s(S)$.

Figure 3.1 shows an example consisting of a community within a small graph. The vertices of the community have been labeled with numbers from one to five. Next to the graph, we show a table with the statistics of each vertex in the community, including their $WCC_v(x,S)$. We see that vertex five has a WCC_v of one, since all the triangles it participates in are contained in S, and it closes at least one triangle with each of the members of the community. In other words, all those and only those vertices with whom vertex five is structurally connected, are in S, meaning it is fully isolated from the rest of the graph and intraconnected. On the other hand, vertex two is that with the smallest WCC_v out of all the vertices, as 25% of all of its triangles (one out of four) are outside the community, thus reducing its isolation. Moreover, at the same time, it does not close any triangle with vertex three, thus reducing his intraconnectivity with S.



(a) 0.09048 (b) 0.14191 (c) 0.255457 (d) 0.251792

(e) 0.52118 (f) 0.65128 (g) 0.78072 (h) 0.92798

Figure 3.2: Examples of communities from real graphs, sorted by WCC_s.

Finally, Figure 3.2 shows some examples of communities with different values of WCC_s, showing different levels of cohesion. These communities are extracted at random from the set of communities found in the real graphs by algorithms used in Chapter 5. The color of the vertices represents the percentage of neighbors belonging to the community. The darker the vertex, the larger the percentage of neighbors of the vertex that belong to the community, that is, the larger the *isolation* of the vertex. On the other hand, the size of the vertices represents the percentage of vertices of the community that are actual neighbors of that vertex. The larger the size of the vertex, the more connected the vertex is with the other vertices of the community, that is, the larger the *intraconnectivity*. In other words, the color represents the isolation, while the size represents the intraconnections. Thus, the better the community is, the larger and darker are its vertices. We see then, that there is a correlation between high WCC_s values and sets with a more appreciable community structure.

3.4 Formal analysis of WCC

In this section, we perform an extensive formal analysis of WCC, aiming at giving the user a broader insight of its behavior, as well as proving that WCC fulfills the properties introduced at the beginning of this chapter. We start with an analysis of the basic behavior of WCC, followed by the structural properties. Then, we turn our focus on proving that WCC fulfills the different behavioral properties, starting with the *lineal community cohesion*, and then performing a detectability analysis that allows us to show that WCC is *scale independent* and *adaptive*. Altogether, we show that WCC is a robust and reliable metric for social networks, from a theoretical perspective [2].

3.4.1 Basic behavior

We formally summarize the basic behavior of $WCC_v(x, S)$ as follows:

Proposition 1 *Let $G = (V, E)$ be a graph and $\emptyset \neq S \subseteq V$. Then,*

(i) $0 \leq WCC_v(x, S) \leq 1$ *for all* $x \in V$.

(ii) $WCC_v(x, S) = 0$ *if and only if* $t(x, S) = 0$.

(iii) $WCC_v(x, S) = 1$ *if and only if* $vt(x, V) = vt(x, S) = |S \setminus \{x\}| \geq 2$.

The value of $WCC_v(x, S)$ indicates the cohesion of vertex x with respect to S. This value is a real number between 0 and 1 (Proposition 1 (i)). These two extreme values are only observed in particular situations (Proposition 1 (ii-iii)). On the one hand, for a given vertex x, in order to have some degree of cohesion with a subset S, the vertex must at least form one triangle with two other vertices in set S. If a vertex builds no triangle with the vertices in S, then the cohesion of the vertex with respect to the set is zero. On the other hand, value one is reached if and only if S includes exactly and only all the vertices that close triangles with x. Furthermore, from the point of view of WCC_v,

[2] All the proofs for the propositions and theorems of this section can be found in the Appendix.

only those edges in E closing at least one triangle are relevant and influence the cohesion of a vertex.

We infer three characteristics on $WCC_s(S)$ from Proposition 1 as follows.

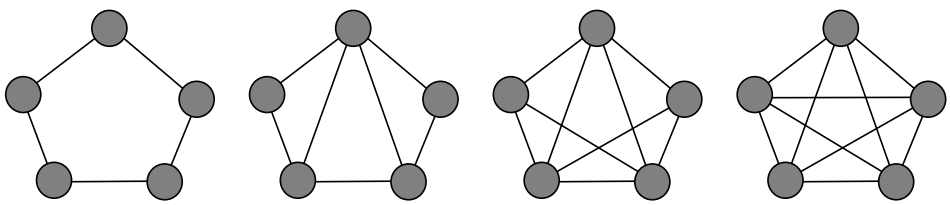
Proposition 2 *Let $G = (V, E)$ be a graph and $\emptyset \neq S \subseteq V$. Then,*

(i) $0 \leq WCC_s(S) \leq 1$.

(ii) $WCC_s(S) = 0$ *if and only if S has no triangles.*

(iii) $WCC_s(S) = 1$ *if and only if S is a clique with $vt(x, V) = vt(x, S)$ for all $x \in S$.*

The clique is the subgraph structure that best resembles the perfect community, and thus, WCC_s rates it with the largest value. On the other hand, if the community has no triangles, its quality is the minimum possible. In Figure 3.3(a-d), we show a community of five vertices with an increasing number of internal triangles. The larger the density of triangles, the larger the WCC_s value for the community.



(a) 0 (b) 0.7 (c) 0.9 (d) **1**

Figure 3.3: Example of the sensitivity of WCC against triangles

Finally, according to Equations 3.1 and 3.2, an optimal partition is such that, for all vertices of the graph, function $WCC_v(x, S)$ is optimized.

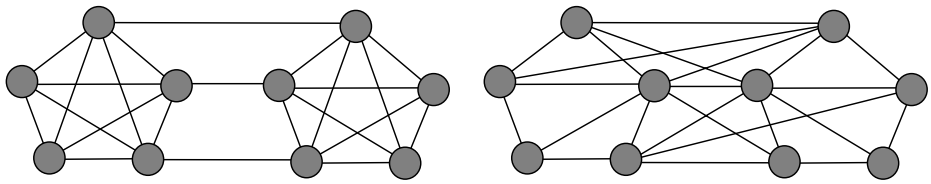
3.4.2 Structural Properties of WCC

3.4.2.1 Internal structure sensitive

WCC is crafted to be sensitive to triangles, and as a consequence, to be sensitive to the internal structure of the community. We verify this property

for WCC: the left factor in Equation 3.3 is the ratio between the number of triangles that vertex x closes with the vertices in S and the number of triangles that vertex x closes with the whole graph. Hence, this left factor is affected by the number of triangles inside the community. On the other hand, the right factor depends on the number of vertices that form triangles with vertex x. Therefore, the distribution of triangles inside the community affects the right factor.

Figure 3.4 shows two examples of communities with the same number of vertices and edges, but distributed differently. While in Figure 3.4(a) we see two cliques with only three edges connecting them, in Figure 3.4(b) we see a more uniformly structured community closing more triangles. We see that WCC_s scores Figure 3.4(b) higher, since the community is more structurally intraconnected, even though there are two cliques in Figure 3.4(a).



(a) 0.444 (b) **0.511**

Figure 3.4: Consequences of the internal structure on the WCC

3.4.2.2 Bridge resistant

Optimal communities found by maximizing WCC *never* contain bridges. We show this based on the following observation:

Theorem 1 *Let S_1 and S_2 be two communities in a partition of graph $G = (V, E)$ such that:*

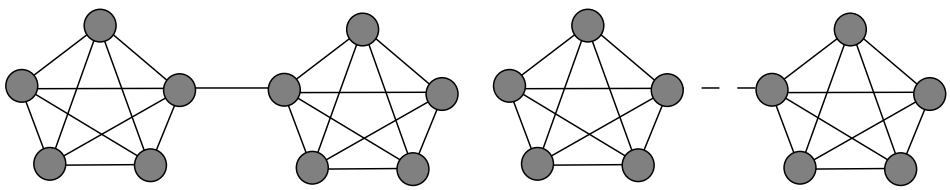
(i) *S_1 and S_2 are the set of vertices of two different connected components.*

(ii) *$WCC_s(S_1) > 0$.*

Then, the following inequality holds:

$$WCC(\{S_1, S_2\}) > WCC(\{S_1 \cup S_2\}).$$

When an edge does not close any triangle, it does not affect the computation of WCC. A bridge does not close any triangle, hence, a bridge is never accounted by WCC. Thus, sets of vertices connected by bridges are not merged into a community because of Theorem 1. In Figure 3.5 (a-b), we show an example of the application of Theorem 1. We see that having the two cliques separated is better than considering a single community with a bridge, in terms of WCC.



(a) 0.444 (b) **1**

Figure 3.5: Example of the behavior of WCC against bridges

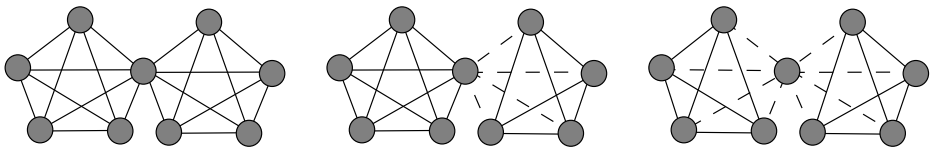
3.4.2.3 Cut Vertex resistant

In Figure 3.6(a-c), we show two cliques (note that the clique is the highest density graph structure) of size five sharing a vertex. Here, WCC is able to separate the communities for this particular case because the left and right sets of vertices have enough structural intraconnectivity and isolation to become separate communities, assigning the cut vertex to one of them. We prove this property for WCC for the case where communities have the highest possible density, which is the clique:

Theorem 2 *Let $G = (V, E)$ be a graph of order n which consists of two cliques K_r and K_s of orders r and s, respectively, that intersect in a vertex t. Assume $r \geq s \geq 4$.*

(i) *If $\mathcal{P}_1 = \{K_r \cup K_s\}$, then*

$$n \cdot WCC(\mathcal{P}_1) = \frac{(r-1)(r-1)}{r+s-2} + \frac{1}{r+s-2} + \frac{(s-1)(s-1)}{r+s-2}; \quad (3.4)$$



(a) 0.556 (b) **0.722** (c) 0.444

Figure 3.6: Example of WCC against vertex cuts

(ii) *if* $\mathcal{P}_2 = \{K_r,\ K_s \setminus \{t\}\}$, *then*

$$n \cdot WCC(\mathcal{P}_2) = (r-1) + \frac{(r-1)(r-2)}{(r-1)(r-2)+(s-1)(s-2)} \tag{3.5}$$

$$+ \frac{(s-1)(s-2)(s-3)}{(s-1)(s-2)}; \tag{3.6}$$

(iii) *if* $\mathcal{P}_3 = \{K_r \setminus \{t\}, \{t\},\ K_s \setminus \{t\}$, *then*

$$n \cdot WCC(\mathcal{P}_3) = \frac{(r-1)(r-2)(r-3)}{(r-1)(r-2)} + \frac{(s-1)(s-2)(s-3)}{(s-1)(s-2)}; \tag{3.7}$$

(iv) $WCC(\mathcal{P}_3) \leq WCC(\mathcal{P}_2)$.

(v) $\max\{WCC(\mathcal{P}_1), WCC(\mathcal{P}_2), WCC(\mathcal{P}_3)\} = WCC(\mathcal{P}_2)$.

This theorem illustrates the fact that WCC avoids merging two very well defined communities (such as two cliques) because of a single vertex. The reason is that WCC is a metric that not only takes into account the vertices that are connected and forms triangles, but also the vertices that do not. Thus, if the triangles inside the community are not distributed evenly among all the vertices then the quality of the community is penalized.

3.4.3 Behavioral Properties

3.4.3.1 Linear Community Cohesion

WCC is crafted to have a linear community cohesion by means of the following theorem:

Theorem 3 *Let $G = (V, E)$ be a random graph of order r in which each edge occurs independently with probability p and closes at least one triangle. Let $v \notin V$ be a vertex connected to and forming at least one triangle with $d \geq 2$ vertices of V. Consider the two partitions $\mathcal{P}_1 = \{V \cup \{v\}\}$ and $\mathcal{P}_2 = \{V, \{v\}\}$. Then,*

(i) $(r+1)WCC(\mathcal{P}_1) = (r-1)p + 2dr^{-1}$.

(ii) $(r+1)WCC(\mathcal{P}_2) = (r-d)p + \frac{d}{r} \cdot \frac{((r-1)p+1)(r-1)(r-2)p^2}{(r-1)(r-2)p^2 + 2(d-1)}$.

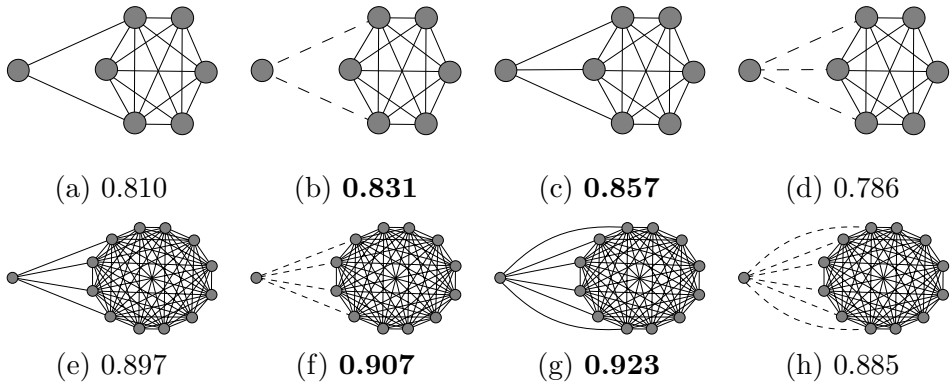
(iii) *For r large enough, $WCC(\mathcal{P}_1) > WCC(\mathcal{P}_2)$ if and only if* $d > rp\left(\sqrt{p^2+2p+9} - (1+p)\right)/4$.

For example, in the particular case of the clique (where $p = 1$), it is necessary to connect to roughly more than one third of the vertices to become a member of the community.

Corollary 1 *Let S be a clique of order r. Given a vertex v, there must exist at least $0.37 \cdot r$ edges between v and S to hold $WCC(\{S \cup \{v\}\}) > WCC(\{S, \{v\}\})$.*

In Figure 3.7 we show an example of Theorem 3, where we represent four groups of examples: a-b, c-d, e-g and g-h. The left graph of each group (a,c,e and g), represents a partition with one single community, while the right graph assumes a partition with two distinct communities formed by a single vertex and a clique. We see that group a-b has a better WCC score for distinct communities while group c-d for one community. Note that WCC gives a better score for vertices connected to more than 0.37 vertices of a clique. The same happens in examples e-f (four connections are less than 0.37 vertices), and g-h (6 connections are more than 0.37 vertices). This example illustrates the linear community cohesion of WCC, where the number of connections required by a vertex to become part of a community, scales with its size.

We empirically validated the linearity property of WCC. We generated instances of graphs formed by a vertex v and a community C, with a probability



(a) 0.810 (b) **0.831** (c) **0.857** (d) 0.786

(e) 0.897 (f) **0.907** (g) **0.923** (h) 0.885

Figure 3.7: Examples showing the linear community cohesion of WCC

p_{in} for an edge to exist between two vertices of C, and a probability p_{out} for an edge to exist between v and a vertex of C. We generated two types of graphs: one with a community of size 100, and another one with a community of size 200. For each possible configuration of p_{in} and p_{out} (in steps of 0.01), we generated 100 instances of each type. Using the WCC based algorithm proposed in Chapter 4, we tested for which configurations the resulting partition was formed by a single community containing both the original community and the vertex, and which did not. Figure 3.8(a) and (b) show the theoretical threshold line of Theorem 3 for both sizes in dim white, and in grey scale the results obtained by the algorithm. White means that the algorithm opted to merge the communities into a single one for all the instances of that particular configuration, while black means that a non merging configuration was always returned. From the figure we can observe that the theoretical threshold is empirically observed, becoming sharper as the size of the graph increases. The black region at the bottom of Figure 3.8(b) becomes smaller as the graph grows (as opposed to Figure 3.8(a)), and as predicted in Theorem 3, it disappears for arbitrarily large graphs. For small graphs and small p_{in}, the probability for a vertex to exist without closing any triangle is large, and therefore the community is shattered into smaller sub communities.

3.4.3.2 Detectability Analysis of WCC

Recent studies have revealed the difficulties of existing community detection metrics, such as modularity, to detect communities if they are not well de-

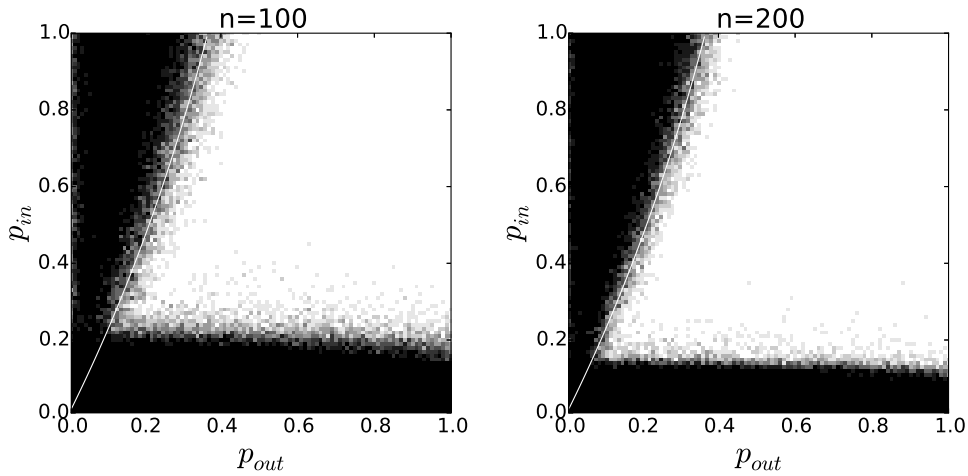


Figure 3.8: Empirical evaluation of Property 2.

fined [10]. Typically, these studies use simplified graph models, being the *Stochastic Block Model* one of the most widely used. This model assumes a graph with q communities, where there is an edge between two vertices with probability p_{in} if both belong to the same community, and probability p_{out} if they belong to different communities. The question is whether a given community detection metric is able to detect the communities of the model for a given set of configuration parameters (p_{in}, p_{out}, n, and q). In this section we analyze the level of detectability of WCC using the stochastic block model. This analysis will serve us to show that WCC is both *context aware* and *scale independent*, at least for the configurations embraced by the model. For the sake of simplicity, in our study we will stick to the case where we have a graph with n vertices, consisting of $q = 2$ communities of size $\frac{n}{2}$.

Given a stochastic block model graph G of size n and two communities of size $\frac{n}{2}$, we want to find the *detectability threshold* of WCC, that is, the point in the relation between p_{in} and p_{out} where maximizing WCC obtains the expected communities of the model. Actually, this can be also seen in terms of *intraconnectivity* and *isolation*: the value of p_{in} models the *intraconnectivity* of the communities of the model (the larger p_{in}, the better the *intraconnectivity*) while p_{out} models the *isolation* of the communities (the lower p_{out} is, the better

the *isolation*). The *detectability threshold* of WCC is defined in Theorem 4, whose proof can be found in Appendix 9.6.

Theorem 4 *Let* G *be a an arbitrarily large graph with n vertices with two communities A and B of size* $\frac{n}{2}$ *each. Let* $C(x)$ *be the community where vertex x is assigned. Two vertices x and y are connected with probability* p_{in} *if* $C(x) = C(y)$*, and with probability* p_{out} *if* $C(x) \neq C(y)$*. Let* $\mathcal{P}$ *be any possible partition of the graph, being* $\mathcal{P}_1 = \{A, B\}$ *and* $\mathcal{P}_2 = \{A \cup B\}$ *particular instances of that partition. Then:*

(i)

$$WCC(\mathcal{P}_1) = \frac{(\frac{n}{2}-1)(\frac{n}{2}-2)\cdot p_{in}^3 \cdot ((\frac{n}{2}-1)\cdot p_{in} + \frac{n}{2}\cdot p_{out})}{((\frac{n}{2}-1)(\frac{n}{2}-2)\cdot p_{in}^3 + (\frac{n}{2}-1)n\cdot p_{in}\cdot p_{out}^2)(\frac{n}{2}\cdot p_{out} + \frac{n}{2} - 1)}$$

(ii)

$$WCC(\mathcal{P}_2) = \frac{((\frac{n}{2}-1)\cdot p_{in} + \frac{n}{2}\cdot p_{out})}{n-1}$$

(iii) $\arg\max_{\mathcal{P}} WCC(\mathcal{P}) \in \{\mathcal{P}_1, \mathcal{P}_2\}$ *if and only if* $p_{in} > p_{out}$*;*

(iv) $WCC(\mathcal{P}_1) > WCC(\mathcal{P}_2)$ *if and only if*

$$p_{in} > \frac{\sqrt{(2 - 2\cdot p_{out})(p_{out}+1)}\cdot p_{out}}{1 - p_{out}} \tag{3.8}$$

Theorem 4 states that the partition with optimal WCC for these particular graphs is either that containing the original communities ($\{A, B\}$) or that taking the graph as a single community ($\{A \cup B\}$), and the transition point between the former and the later is that expressed by Equation 3.8. In other words, WCC is able to recover the original communities if and only if the condition in Equation 3.8 (the *detectability threshold*) holds.

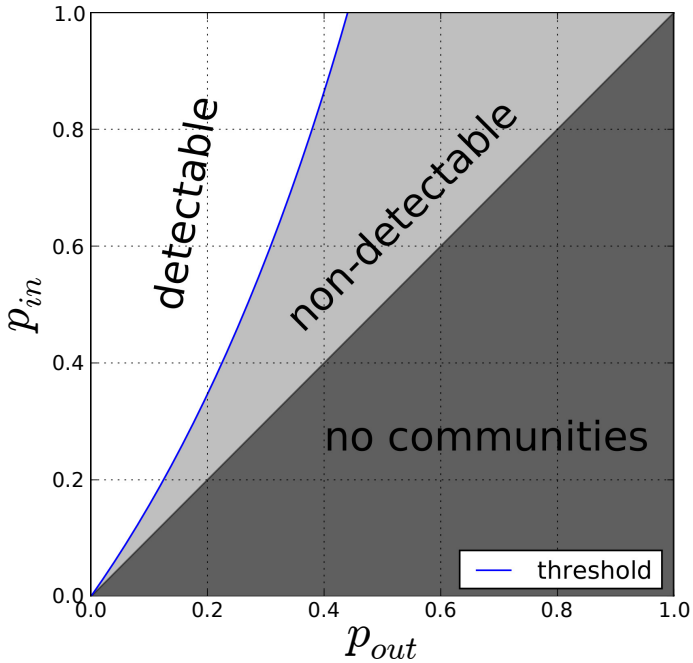


Figure 3.9: Transition point for WCC and different values of p_{in} and p_{out}

3.4.3.3 WCC's detectability discussion

The question is whether the detectability threshold of WCC is good or not. Intuitively, according to the informal community definition, one would expect a community detection metric to detect communities whenever $p_{in} > p_{out}$. However, this definition is incomplete and, in practice, $p_{in} > p_{out}$ is not a sufficient condition to define a community. As an example, suppose a clique of size n. Just removing an edge of the clique would imply a configuration with two communities of size $\frac{n}{2}$, with $p_{in} = 1 > p_{out} = \frac{n^2-4}{n^2}$, if we strictly adhere to the informal community definition. Therefore, a community metric with a detectability threshold of $p_{in} > p_{out}$ would potentially only detect communities with a perfectly uniform edge distribution. Otherwise, these would break up into smaller subcommunities with a uniform density. Clearly, this is not practical in a real application, as real graphs are typically not uniform and situations such as that described above appear frequently. Therefore, the

actual transition point between detectable and non-detectable configurations for a given metric is a direct consequence of the community definition of that metric, and whether this is good or bad is determined by the application in use. In the case of WCC, we evaluate it using several graphs from social networks, proving that it outperforms existing methods in the literature, in Chapter 5.

This analysis allows us to show that WCC is both *adaptive* and *scale independent*. First, we see that the balance between *intraconnectivity* and *isolation* is independent of the size of the network (Equation 3.8), which makes the metric *scale independent*. This is not the case, for instance, for modularity, whose detectability threshold is $c_{in} - c_{out} \geq 2\sqrt{c_{in} + c_{out}}$, where $c_{in} = \frac{n}{2} \cdot p_{in}$ and $c_{out} = \frac{n}{2} \cdot p_{out}$. In this case, the smaller the graph, the larger c_{in} needs to be compared to c_{out} to be able to correctly identify the communities. This issue is related to the known resolution problems of modularity, which is unable to detect small and well defined communities once the size of the graph increases.

Second, we see that WCC does not impose either a minimum level of *intraconnectivity* or *isolation* to a set of vertices to be detected as a community. This is better observed in Figure 3.9, where we plot three well defined regions, corresponding to the different configuration spaces where WCC is able to detect the communities, where it is not able to detect them and where communities do not exist ($p_{in} \leq p_{out}$). We also show the detectability threshold of WCC, which delimits the detectable and non-detectable regions. We see that the threshold establishes a relation between p_{in} (*intraconnectivity*) and p_{out} (*isolation*), in such a way that the more intraconnected the communities are (the larger p_{in} is), the less isolated (the larger p_{out}) these can be and vice versa. This means that WCC is a metric that is *adaptive* and locally identifies relevant sets of vertices, thus adapting to the heterogeneous nature of real graphs. Finally, we see that WCC does not detect the original communities whenever they do not exist – i.e. when $p_{in} \leq p_{out}$ –, which is desirable in a community detection metric, thus, not to detect false positives.

In Figure 3.10 we show a numerical validation of the detectability threshold of WCC. We test different configurations of stochastic block model graphs (p_{in} and p_{out} from 0 to 1 in steps of 0.01, for different values of n) with two communities using the WCC maximization algorithm proposed in Chapter 4. We generated 100 graphs for each tested configuration, executed the algorithm, and compared the resulting partition with that expected from the model

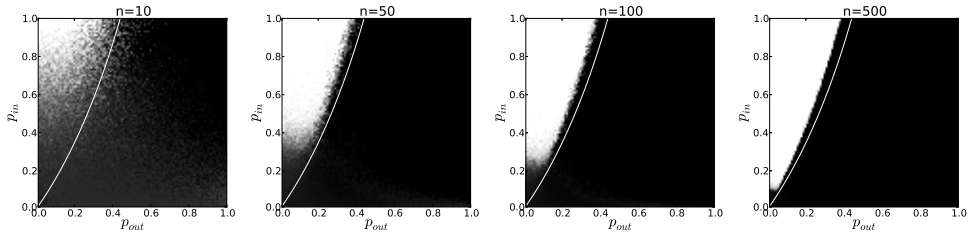


Figure 3.10: Detectability of the proposed algorithm for the stochastic block model Graphs of different sizes with different configuration parameters (p_{in} and p_{out}). The closer to white is, the better the NMI between the detected partition and that expected by the model. The closer to black, the more different.

using the *Normalized Mutual Information* (NMI) [15]. Each point in the graph corresponds to the average NMI of those 10 executions. The whiter the color, the closer to one the average NMI is (the algorithm finds the expected communities), and the darker the color, the closer to zero the NMI is (the communities found by the algorithm are very different from those expected from the model). We also draw the detectability threshold.

We see the detectability threshold line fits very well with the empirical results obtained, with a very well defined transition point between the detectable and non-detectable regions. The larger the size of the graph, the better this fitness, because the larger the graph, and as a consequence, the larger the communities, the more uniform the internal edge density of these is. Furthermore, we also empirically confirm that when communities do not exist ($p_{in} \leq p_{out}$), WCC does not detect them. These empirical validation also suggests that the algorithm proposed in Chapter 4 is able to produce results close to the optimal, even though it does not formally guarantee to produce an optimal solution.

3.4.3.4 The community detection paradox and WCC

Another issue that affects modularity maximization based algorithms is the so called community detection paradox [52]. Counter-intuitively, the paradox states that the worse defined the communities are, the easier it is for the algorithms to detect them, while the better defined they are, the harder it is. In order to test whether WCC is affected by this issue or not, we first need

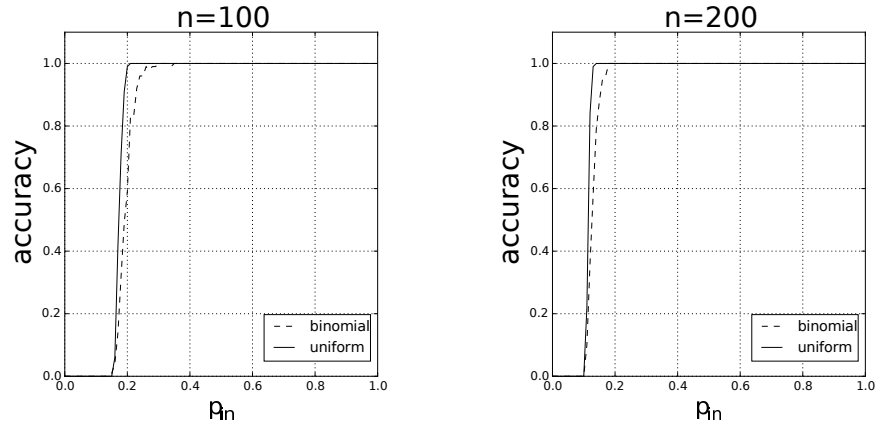


Figure 3.11: Accuracy of SCD to detect a single community ($p_{out} = 0$) with a uniform vs binomial degree distribution, for different expected values of p_{in}.

to define what is a good and a bad community in terms of WCC, and then, test whether in situations where we have bad communities, they are harder or easier to detect. More concretely, for WCC, a well defined community shows both a good isolation and intraconnectivity. Also, a bad community is not well isolated nor intraconnected[3].

We first analyze the case where a community is perfectly isolated, that is, it does not have external edges connecting its vertices to other communities. The *detectability threshold* shows that when the internal degree of the vertices of the community is uniform, the larger the p_{in} is, the larger the p_{out} can be and the community can still be detectable up to a certain point. In the case that the degree of the vertices is not uniform, some vertices might be well intraconnected, while others might not close enough triangles to be part of the community. If this happens, the community structure is not so well defined and thus, we expect WCC to fail at identifying the community.

Figure 3.11 shows the accuracy of SCD when the degrees of the vertices follow a *Binomial* distribution compared to when the degrees of the vertices are uniform, for different values of p_{in} and communities of size $n = 100$ and

[3]Note that in the case of modularity and other state of the art algorithms these definitions change, and a community is usually considered to be well defined when its vertices have more internal edges than external edges.

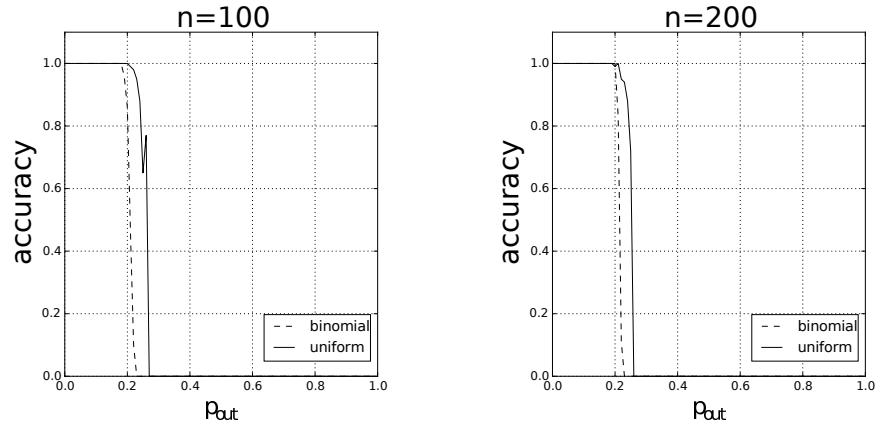


Figure 3.12: Accuracy of SCD to detect two communities with a uniform degree distribution with density $p_{in} = 0.5$, and a uniform vs binomial out degree distribution, for different expected values of p_{out}.

$n = 200$ vertices. In this case the graph consists of a single community, thus p_{out} is zero. For each configuration, we have randomly generated 100 graphs, executed the algorithm and averaged the results. An accuracy of one means that the algorithm is able to fully recover the communities for all of the 100 generated instances, while an accuracy of zero means that it was not able to recover the communities in some of them. We see that as long as the value of p_{in} increases, there is a transition point for both distributions where the algorithm starts to correctly detect the community for all the instances of the graph. This transition point is seen earlier for the uniform graph than for the binomial. In the case of WCC, one would expect this transition point not to exist for perfectly uniform graphs and always detect the communities, as predicted in Theorem 3 where the number of edges required between a vertex and a community tends to zero as p_{in} approaches zero. However, this is the case for arbitrarily large graphs, but not for small graphs where having perfectly uniformly distributed edges is more difficult as these either exist or not. Therefore, the smaller p_{in} is, the larger the probability a vertex not having enough edges with the community or even not closing any triangle with it exists. This probability is larger for the case of the binomial distribution, where degrees are less homogeneous.

We also tested the situation where we have two well intraconnected and uniform communities, but the out degree distribution connecting both communities is not uniform but follows a Binomial distribution. In this case, we expect that for the Binomial distribution the communities will be harder to detect, as some observed vertices will have a larger out degree than that expected, making them less isolated from the rest of the graph, even sometimes being more intraconnected with vertices of the other community. Figure 3.12 shows the accuracy of the community detection algorithm based on WCC optimization, when the two communities have a uniform distributed internal degree with p_{in} of 0.5, and the out degree follows a Binomial and a uniform degree distribution, for different values of p_{out}. In this case, we see that when the out degree follows a Binomial distribution, the transition point between detectable configurations and non-detectable configurations is seen earlier (for smaller values of p_{out}).

In conclusion, we see that WCC is sensitive to how the edges are internally and externally distributed. The more uniformly distributed these are, the easiest is for WCC to detect the communities. If some vertices do not have enough intraconnection or isolation, then WCC will not classify them in the correct community, as expected.

Chapter 4

Scalable Community Detection

When looking at existing community detection algorithms, it is possible to realize they are not designed to tackle the current trends in computer architecture and data mining. Besides the problems arising from maximizing metrics with known issues, existing algorithms not prepared to run on real sized graphs. Many of the existing proposals are based on complex metrics or procedures, that can run on small graphs of thousands of vertices, they are not practical for datasets containing millions or even billions of vertices and edges. Furthermore, existing methods are not designed to exploit the architectural characteristics of modern hardware –i.e multi-cores–, and therefore they are inefficient in terms of resources usage.

In order to overcome these issues, we present the **Scalable Community Detection** (SCD) [48, 50], a novel community detection algorithm based on WCC and designed to scale on SMP machines. Thanks to WCC maximization, SCD is able to find high quality communities and to exploit the parallel nature of triangle counting to scale on shared memory machines. Altogether, SCD is a high quality algorithm with formal guarantees able to run on billion edge graphs in a few hours in commodity hardware. The algorithm is divided into three phases: *graph cleanup*, *initial partition* and *partition refinement*, which are now summarized.

Graph Cleanup: After loading the graph into memory, we perform a cleanup process aimed at removing the unnecessary edges and computing a set of statistics that will be helpful during the next phases. The process consists of:

1. computing the number of triangles closes by each edge in the graph closes.

2. Then, removing those edges that do not close any triangle from the graph, as these are irrelevant from the point of view of WCC and do not have any effect during the computation of WCC.

By removing these edges, we reduce the memory consumption and improve the performance of SCD. Furthermore, we can also simplify the heuristic proposed in Section 4.1 (which we use to improve the performance of the *partition refinement* step) since we can assume that each edge closes at least one triangle. Finally, we take the opportunity to store the number of triangles each edge closes, as those will be used in the subsequent steps.

Initial Partition: The goal of this phase is to create an initial partition which we can later refine. This initial partition is computed by a fast heuristic process described in Algorithm 1. We first sort the vertices of the graph by their clustering coefficient decreasingly (which was computed during the graph cleanup step). For those vertices with equal clustering coefficient, we use the degree as a second sorting criterion (Line 2). Then, the vertices are iterated and, for each vertex v not previously visited, we create a new community C that contains v and all the neighbors of v that were not also visited previously (Line 6 to 12). Finally, community C is added to partition P (Line 13) and all the vertices of the community are marked as visited. The process finishes when all the vertices in the graph have been visited.

This heuristic is built on top of the following intuition: the larger the clustering coefficient of a vertex, the larger the number of triangles the vertex closes with its neighbors, and the larger the probability that its neighbors form triangles among them. Hence, considering Equation 3.3, the larger the clustering coefficient of a vertex, the larger is the probability that the WCC of its neighbors is large if we include them in the same community.

Partition Refinement: Algorithm 2 describes the partition refinement step. It takes the initial partition computed in the "Initial partition" step and refines it by following a hill climbing strategy, that is, in each iteration, a new partition is computed from the previous one by performing a set of modifications

ALGORITHM 1: Phase 1, initial partition.

```
   Data: Given a graph G(V,E)
   Result: Computes a partition of G
 1 Let P be a set of sets of vertices;
 2 S ← sortByCC( V );
 3 foreach v in S do
 4 |   if not visited(v) then
 5 |   |   markAsVisited(v);
 6 |   |   C ← v;
 7 |   |   foreach u in neighbors(v) do
 8 |   |   |   if not visited(u) then
 9 |   |   |   |   markAsVisited(u);
10 |   |   |   |   C.add(u);
11 |   |   |   end
12 |   |   end
13 |   |   S.add(C);
14 |   end
15 end
16 return P;
```

(movements of vertices between communities) aimed at improving the WCC of the new partition. The algorithm repeats the process until the WCC of the new partition does not percentually improve over the best WCC observed so far more than a given threshold, and a set of *lookahead* iterations have been performed. These *lookahead* iterations are used to make the algorithm more robust against local maxima. In our tests, setting the threshold to 1% and the *lookahead* to five iterations provided a good tradeoff between performance and quality.

In each iteration, for each vertex v of the graph, we use the `bestMovement` function to compute the movement of v that improves the WCC of the partition the most (Line 8). There are four types of possible movements:

- **NO_ACTION**: leave the vertex in the community where it currently is.

- **INSERT**: insert a singleton community[1] into an existing community. Remove the empty community resulting from this movement.
- **REMOVE**: remove the vertex from its current community and create a new singleton community containing the vertex.
- **TRANSFER**: remove the vertex from its current community (source) and insert it into another one (destination).

Note that `bestMovement` does not modify the current partition, and that the best movement of each vertex is computed independently from the others. This allows computing in parallel the best movements for all the vertices. Once we compute the best movement of all the vertices of the graph, we apply all of them *simultaneously* (`applyMovements` Line 10). Finally, we update the WCC of the new partition (Line 11) and check whether it improved compared to the last iteration.

Before describing function `bestMovement` in detail, we first introduce some auxiliary functions that are used in it. The proofs of the theorems introduced in this section can be found in the Appendix.

- $WCC_I(v, C, P)$ computes the improvement of the WCC of a partition P when vertex v (which belongs to a singleton community of P) is inserted into community C of P.

  Theorem 5 *Let $P = \{C_1, C_2, \ldots, C_k, \{v\}\}$ and $P' = \{C'_1, C_2, \ldots, C_k\}$ be partitions of a graph $G = (V, E)$ where $C'_1 = C_1 \cup \{v\}$. Then,*

  $$WCC(P') - WCC(P) = WCC_I(v, C_1, P) = \\ = \frac{1}{|V|} \cdot \sum\nolimits_{x \in C_1} \left[WCC(x, C'_1) - WCC(x, C_1)\right] + \frac{1}{|V|} \cdot WCC(v, C'_1).$$

- $WCC_R(v, C, P)$ computes the improvement of the WCC of a partition P when vertex v is removed from community C of P and placed as a singleton community.

[1] A singleton community is a community composed by a single vertex

ALGORITHM 2: Phase 2, refinement.

```
Data: Given a graph G(V,E) and a partition P
Result: A refined partition P'
1  bestP ← P;
2  bestWCC ← computeWCC(P);
3  triesRemaining ← lookAhead;
4  repeat
5  |  triesRemaining - -;
6  |  M ← ∅;
7  |  foreach v in V do
8  |  |  M.add( bestMovement(v,P) );
9  |  end
10 |  P ← applyMovements(M,P);
11 |  newWCC ← computeWCC(P);
12 |  if (newWCC − bestWCC)/bestWCC ≥ t then
13 |  |  bestP ← P;
14 |  |  bestWCC ← newWCC;
15 |  |  triesRemaining ← lookAhead;
16 |  end
17 until triesRemaining > 0;
18 return bestP;
```

Theorem 6 *Let partitions* $P = \{C_1, C_2, \ldots, C_k\}$ *and* $P' = \{C'_1, C_2, \ldots, C_k, \{v\}\}$ *of a graph* $G = (V, E)$ *where* $C_1 = C'_1 \cup \{v\}$. *Then,*

$$WCC(P') - WCC(P) = WCC_R(v, C_1, P) = -WCC_I(v, C'_1, P').$$

- $WCC_T(v, C_1, C_2, P)$ computes the improvement of the WCC of a partition when vertex v is transferred from community C_1 and to C_2.

  Theorem 7 *Let* $P = \{C_1, C_2, \ldots, C_{k-1}, C_k\}$, $P' = \{C'_1, C_2, \ldots, C_{k-1}, C_k, \{v\}\}$ *and* $P'' = \{C'_1, C_2, \ldots, C_{k-1}, C'_k\}$ *be partitions of a graph* $G = (V, E)$ *where* $C_1 = C'_1 \cup \{v\}$ *and* $C'_k = C_k \cup \{v\}$. *Then,*

$$\begin{aligned} WCC(P'') - WCC(P) &= WCC_T(v, C_1, C_k, P) \\ &= WCC_R(v, C_1, P) + WCC_I(v, C_k, P). \\ &= -WCC_I(v, C'_1, P') + WCC_I(v, C_k, P). \end{aligned}$$

From Theorem 5, we conclude that computing the improvement of WCC resulting from inserting a vertex v (i.e. a singleton community) into a community C, we only need to recompute the WCC of vertex v and those vertices in C. Therefore, when computing $WCC_I()$ for a vertex and a community, only a very local portion of the graph needs to be accessed, and the number of computations performed is small compared to computing the WCC of the whole partition. Furthermore, Theorems 6 and 7 show that we can express movements INSERT, REMOVE and TRANSFER, in terms of function $WCC_I()$, which in turn simplifies the implementation of the algorithm.

Algorithm 3 describes the `bestMovement` function. First, we compute the improvement of removing vertex v from its current community (Line 3). Then, we obtain the set of candidate communities, formed by those communities containing the neighbors of v (Line 6). After that, we calculate which is the candidate community where inserting or transferring vertex v (depending whether the v forms a singleton community or not) improve the WCC most (Lines 7 to 17). Finally, we select whether the best improvement is obtained from removing the vertex from its current community (REMOVE) or inserting/transferring it into a new community (INSERT/TRANSFER) (Lines 18 to 26). If neither of the two movements improves the WCC of the partition, we keep the vertex in the current community (NO_ACTION) (Line 1).

4.1 Heuristic

We have seen that we can express any movement by means of WCC_I. Computing $WCC_I(v, C, P)$ requires computing the triangles that v and those vertices in C close with the other vertices in C and v. For a vertex, this operation is bound by the number of neighbors that have (d), which has a complexity of $O(d^2)$ (for each neighbor in $C \cup \{v\}$ we have to test against all of its other neighbors in $C \cup \{v\}$). Also, since real graphs typically have power law distributions, this cost is large for the highest degree vertices in the graph. Finally, considering that the number of times WCC_I is called is bounded by the number of edges m of the graph, it quickly becomes the most time consuming part of the algorithm. In this section, we propose a model to

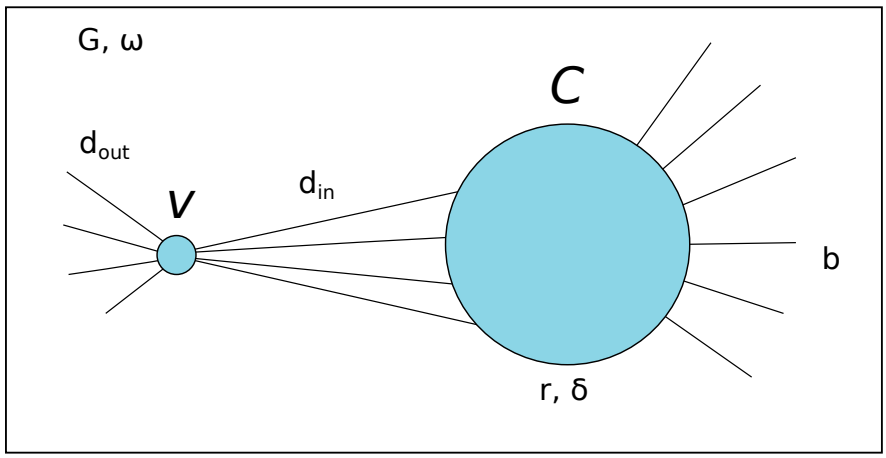


Figure 4.1: Model used for estimating the WCC_I.

estimate $WCC_I()$ with a constant time complexity function (given some easy to compute statistics) that we call $WCC'_I()$.

$WCC'_I()$ stands the approximated increment of WCC when vertex v is inserted into a community C. In Figure 4.1, we depict the simplified model on top of which $WCC'_I()$ is built. For a given vertex v, we only record the number of edges that connect it to community C. For each community C, we keep the following statistics: the size of the community r; the edge density of the community δ; and the number of edges b that are in the boundary of the community. We also use the clustering coefficient of the graph ω, which is constant along all the community detection process and has been computed during the *graph cleanup* step. The clustering coefficient of the graph is equivalent as the observed probability that two given edges that share a vertex close a triangle. These statistics homogenize the community members and allow the computation of $WCC'_I()$ as follows:

Theorem 8 *Consider the situation depicted in Figure 4.1, with the following assumptions:*

- *Every edge in the graph closes at least one triangle.*
- *The edge density inside community C is homogeneous and equal to δ .*

- *The clustering coefficient of the whole graph equals to ω.*

Then,

$$\begin{aligned} WCC(P') - WCC(P) &\approx WCC'_I(v, C) \\ &= \frac{1}{V} \cdot (d_{in} \cdot \Theta_1 + (r - d_{in}) \cdot \Theta_2 + \Theta_3), \end{aligned} \tag{4.1}$$

where,

$$\Theta_1 = \frac{(r-1)\delta+1+q}{(r+q)\cdot((r-1)(r-2)\delta^3+(d_{in}-1)\delta+q(q-1)\delta\omega+q(q-1)\omega+d_{out}\omega)} \cdot (d_{in}-1)\delta;$$

$$\Theta_2 = -\frac{(r-1)(r-2)\delta^3}{(r-1)(r-2)\delta^3+q(q-1)\omega+q(r-1)\delta\omega} \cdot \frac{(r-1)\delta+q}{(r+q)(r-1+q)};$$

$$\Theta_3 = \frac{d_{in}(d_{in}-1)\delta}{d_{in}(d_{in}-1)\delta+d_{out}(d_{out}-1)\omega+d_{out}d_{in}\omega} \cdot \frac{d_{in}+d_{out}}{r+d_{cout}};$$

and $q = (b - d_{in})/r$.

Conceptually, Θ_1, Θ_2 and Θ_3 are the WCC_v improvements of those vertices in C connected to v, those vertices in C not connected to v, and vertex v respectively, when v is added to community C. The evaluation of Equation 4.1 is $O(1)$ given all the statistics. Also, the update of all statistics is only performed when all communities are updated, with a cost $O(m)$ for the whole graph. Note that we use aggregated statistics to estimate the number of triangles, and thus we are *not* computing the triangles when we compute $WCC'_I()$.

4.1.1 Complexity of the Algorithm

Let n be the number of vertices and m the number of edges in the graph. We assume that the average degree of the graph is $d = m/n$ and that real graphs have a quasi-linear relation between vertices and edges $O(m) = O(n \cdot \log n)$. Then, the complexity of each of the steps of the algorithm is the following:

Graph Cleanup: In the graph cleanup phase, for each edge in the graph, we compute the triangles that each edge participates in. The triangles are found by intersecting the adjacency lists of the two connected vertices. Since we assume sorted adjacency lists, the complexity of computing the intersection is $O(d)$. Finally, we compute the local clustering coefficient of each vertex and the number of triangles each vertex closes, which has a cost of $O(m)$ once we have the triangles each edge participates in. Since the average degree is m/n, we have that the cost of the first phase is $O(m \cdot d + m) = O(m \cdot \log n + m)$.

Initial Partition: The cost of this step is the cost of sorting the vertices of the graph based on the local clustering coefficients computed in the previous phase, which is $O(n \cdot log(n))$.

Partition Refinement: Let α be the number of iterations required to find the best partition P', which in our experiments is between 3 and 7. In each iteration, for each vertex v of the graph, we compute, in the worst case, $d + 1$ movements of type $WCC'(I)$ that have a cost $O(1)$. Then, the computation of the best movement for all vertices in the graph in an iteration is $O(n \cdot (d + 1)) = O(m)$. The application of the all the movements is linear with respect to the number of vertices $O(n)$. We also need to update, for each iteration of the second phase, the statistics δ, c_{out}, d_{in} and d_{out} for each vertex and community, which has a cost of $O(m)$. Finally, the computation of the WCC for the current partition is performed by computing for each edge the triangles, which is $O(m \cdot \log n)$ as already stated. Hence, the cost of the refinement phase becomes $O(\alpha \cdot (m + n + m + m \cdot \log n))$, which after simplification, becomes $O(m \cdot \log n)$ assuming α as constant.

Finally, The final cost of the algorithm is the sum of the three phases: $O(m \cdot \log n + n \cdot log(n) + m \cdot \log n) = O(m \cdot \log n)$.

ALGORITHM 3: `bestMovement`.

```
Data: Given a graph G(V,E) a partition P and a vertex v
Result: Computes the best movement of v.
1  m ← [NO_ACTION];
2  sourceC ← GetCommunity(v,P);
3  wcc_r ← WCC_R(v,sourceC,P);
4  wcc_t ← 0.0;
5  bestC ← ∅;
6  Candidates ← candidateCommunities(v,P);
7  for c in Candidates do
8  |  if size(sourceC) > 1 then
9  |  |  aux ← WCC_T(v,sourceC,c,P) ;
10 |  else
11 |  |  aux ← WCC_I(v,c,P) ;
12 |  end
13 |  if aux > wcc_t then
14 |  |  wcc_t ← aux;
15 |  |  bestC ← c;
16 |  end
17 end
18 if wcc_r > wcc_t and wcc_r > 0.0 then
19 |  m ← [REMOVE];
20 else if wcc_t > 0.0 then
21 |  if size(sourceC) > 1 then
22 |  |  m ← [TRANSFER , bestC];
23 |  else
24 |  |  m ← [INSERT , bestC];
25 |  end
26 end
27 return m;
```

Chapter 5

Experiments

In this Chapter, we conducted a set of experiments, aimed at empirically proving (i) that WCC is able to correctly measure whether a community is good or not, (ii) that communities found by maximizing WCC are better than those found by other existing methods, and (iii) that SCD is able to scale and process large graphs.

With these goals in mind, the following algorithms of the state of the art were selected as baselines: *Infomap* [58], as a representative of the random-walk based family of algorithms and for being one of the best algorithms for detecting disjoint communities [23]; *Louvain* [7], as it is the fastest and one of the highest quality modularity maximization algorithms; and *Label Propagation Method (LPM)* [54], because it has become very popular due to its excellent scalability. We suggest reading the reference paper for each algorithm to understand them in detail. Our selection covers algorithms following diverse strategies to test the validity of WCC but it does not intend to be an evaluation survey of all community methods. Besides, other popular approaches in the literature, such as [2, 24, 41, 42] among others, aim at overlapping communities which are also out of the scope of this thesis. The implementation of all the algorithms has been taken from their author's web site.

For the experimentation, we used six real networks covering different aspects of real world data, mostly social networks[1]. All chosen networks have ground truth communities associated with them. The first is a network representing

[1]Downloaded from SNAP (`http://snap.stanford.edu`). We cleaned the original graphs by removing the self loops.

Table 5.1: Real-world graphs with ground truth data.

	Vertices	Edges	Communities
Amazon	334,863	925,872	151,037
Dblp	317,080	1,049,866	13,477
Youtube	1,134,890	2,987,624	8,385
LiveJournal	3,997,962	34,681,189	287,512
Orkut	3,072,441	117,185,083	6,288,363
Friendster	65,608,366	1,806,067,135	957,154

which products from Amazon have been copurchased by clients. In this dataset, ground truth communities match the different categories of products. The second is a graph of the DBLP network representing coautorship relations between authors, where ground truth communities correspond to authors that have published in the same journals and conferences. The third graph is a graph of Youtube, where ground truth communities correspond to the groups of users in youtube. The fourth , fifth and sixth datasets are graphs of the Livejournal, Orkut and Friendster social networks, where ground truth communities correspond to the groups created by the users. The characteristics of these graphs are summarized in Table 5. For more information about how these graphs are created and their communities are generated please refer to [67].

We used a machine with the following characteristics: 2xIntel Xeon E5-2609 @ 2.40GHz, with 4 cores each making a total of 8 cores, 128 GB ram and Linux 2.6.32-5-amd64. The used disks are regular 1TB spinning disks at 7200 rpm.

5.1 WCC_s quality

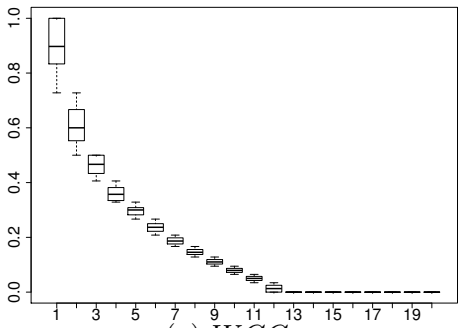
The goal of the experiment is to show the correlation between communities with good WCC_s and good statistical indicators including existing community detection metrics, structural and other characteristics such as: the *average edge density*, the *triangle density*, the *triangle participation ratio* , the *expansion*, the *average inverse edge cut*, the *Flake ODF*, the *conductance*, the *modularity*, the *bridge ratio*, the *normalized diameter* and the *size* of the communities. For formal definitions of these indicators please refer to [48].

We created a pool of communities by running *Infomap*, *Louvain* and *LPM* on the first four real world networks described above[2]. We sorted all the communities in the pool by their WCC_s value decreasingly although they have not been found with such metric. Then, we divided the communities into 20 groups of five percentiles according to their WCC_s and plotted for these 20 groups their corresponding statistical indicators in a boxplot, showing the minimum, the first, the second and the third quartiles, and the maximum for each of the groups. These results are shown in Figure 5.1. In all the charts, the x axis represents the group identifier (e.g. the leftmost group is always the 95 percentile that contains the top 5% communities in terms of their WCC_s) while the y axis shows the corresponding statistical value. The communities of size one and two, are omitted since their WCC_s value is always zero. As shown in Figure 5.1(a), the leftmost communities have high WCC_s values, and the rightmost communities have the lowest WCC_s values. Since these communities were not computed with WCC_s, we analyze both good and bad communities.

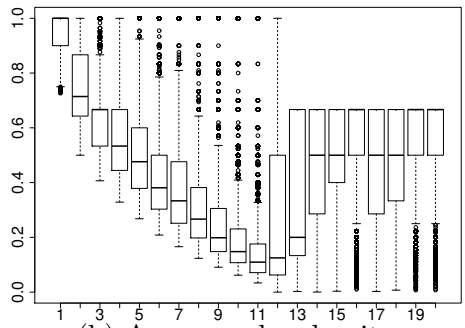
Broadly speaking, we observe two sections in each plot of Figure 5.1: from groups 1 to 11, the trend for all statistical indicators show that communities with higher WCC_s have better properties; from groups 12 to 20 this trend apparently changes in some statistical indicators. We focus first on groups 1-11 and we analyze groups 12-20 later.

Groups 1-11: In Figures 5.1(b) and (c) we see that the larger the WCC_s of a community, the larger the average edge density and the triangle density. The transitive relations between the vertices (Property 1) indicate the presence of communities with a defined homophilic structure. Note that these communities have been found with metrics that do not search for triangles and yet, they contain more such structures. Similarly, Figure 5.1(d) shows that the larger the WCC_s, the larger the TPR. However, we see that TPR has precision limitations, since it scores as good communities (with scores close to one), some communities that might not be that good according to other metrics such as the average edge density (Figure 5.1(b)), the expansion (Figure 5.1(e)), the average inverse edge cut (Figure 5.1(f)) and the conductance (Figure 5.1(g)). Finally, in Figures 5.1(e), (f) and (g), we see that the larger the WCC_s, the smaller the expansion, the larger the average inverse edge cut, and the

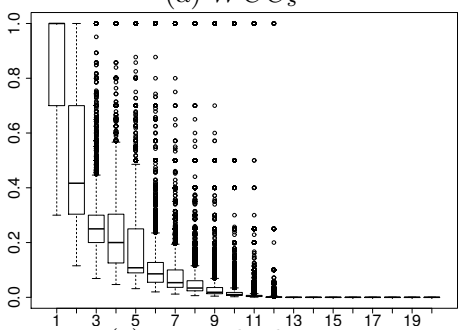
[2]We used the first four as they where the only ones where all the algorithms succeeded to execute.



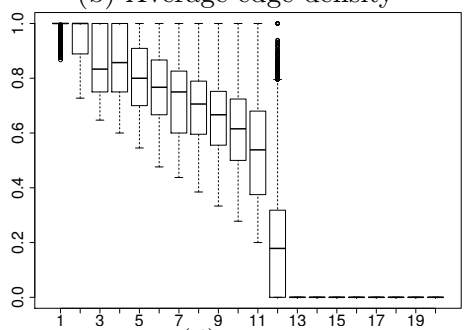
(a) WCC_s



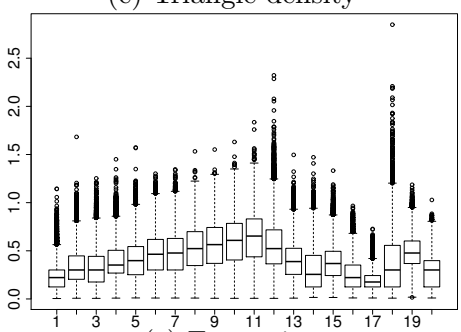
(b) Average edge density



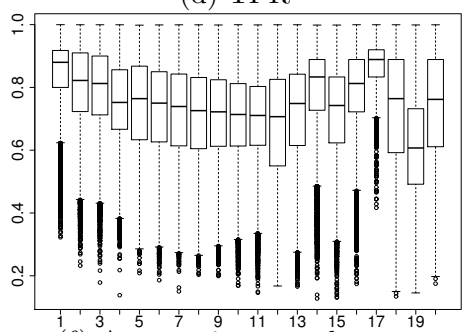
(c) Triangle density



(d) TPR



(e) Expansion



(f) Average inverse edge cut

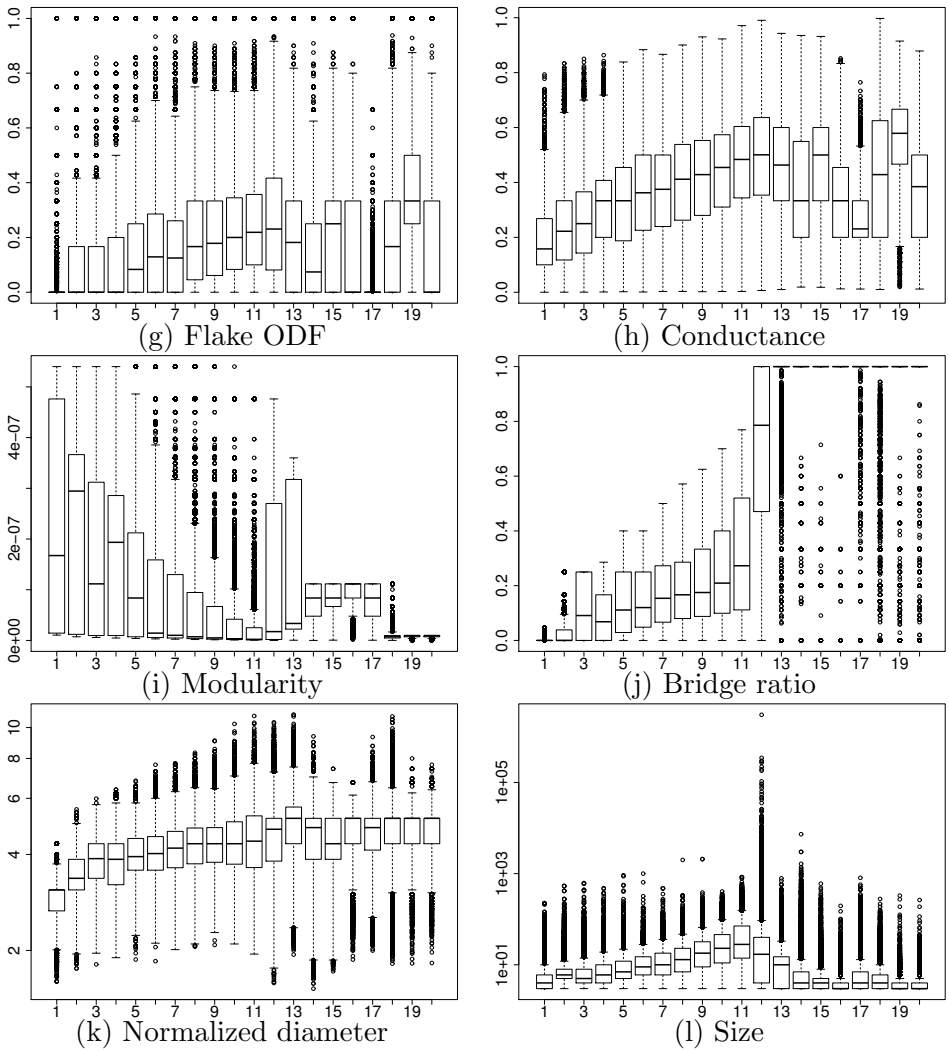


Figure 5.1: Statistics of communities from real world networks in 20 groups sorted by WCC_s. The x-axis represents the 5% percentile groups showing that with the largest WCC on the left and that with the smallest WCC on the right. The y-axis represents the value achieved for each of the metrics shown in the plots.

smaller the conductance, which means that the number of external connections decreases for the first, and that the communities are denser internally than externally for the last two. However, while having a large internal density and few external connections is a good starting point to identify a good community, it does not imply an internal structure as we will show when discussing groups 12-20.

In Figure 5.1(h-i) we compare WCC_s with the most used metrics in the state of the art: conductance and modularity. We see that for these groups, there is a correlation between communities with good WCC_s values, modularity and conductance (note that for conductance, the lower, the better).

Figure 5.1(j) shows that bridges penalize the WCC_s score. A large bridge ratio is a symptom of the presence of whiskers or treelike structures, which are inherently sparse and hence do not have the type of internal structure that one would expect from a community. A small diameter is another feature that any good community should have. In Figure 5.1(k) we see that large WCC_s implies smaller diameters for the communities. This means that any vertex in the community is close to any other vertex, which translates to denser communities. Finally, in Figure 5.1(l), we show the sizes of the communities.

Groups 12-20: We see that there is a trend change in some statistical indicators for those groups that have WCC_s close to 0. This behavior can be explained by Figures 5.1(c), (d) and (j). These figures reveal that the communities after group 15 are treelike: communities hardly contain triangles and almost all the edges in the community are bridges. Such structures cannot be accepted as good communities. Although some communities in groups 15-20 are isolated, we note that this is not a sufficient condition for them to be good communities. For example, most communities in groups 13-20 are trees with three vertices which have a good conductance. WCC_s is able to score these communities as bad communities while conductance does not. A similar behavior is seen for modularity. In Figure 5.1(i), we see that the communities in groups 13-20, which are tree-like, have a larger modularity than other sets with a more community-like structure than those. As described in [6], tree like networks can have high modularity and hence, algorithms maximizing it can lead to misleading results.

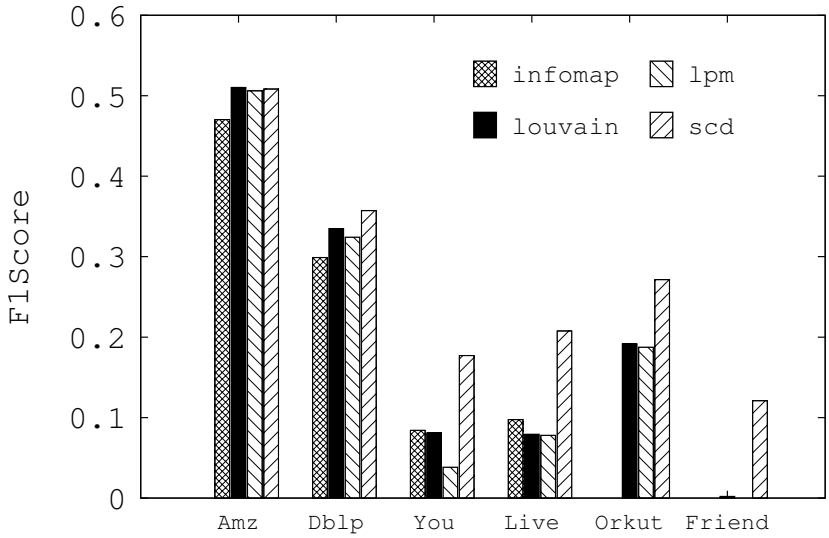


Figure 5.2: F1Score.

To sum up, while state of the art metrics fail to correctly rank communities under specific circumstances, WCC_s shows to be robust, that is, it is able to globally capture all the desirable characteristics a community should contain.

5.2 SCD results quality

We run the baseline algorithms and SCD on the graphs of Table 5 to compare the quality of their results. To measure the quality, we used two metrics: the *Average F1Score* and the *Normalized Mutual Information*(NMI). Given two sets A and B, the F1Score (F_1) between the two sets is computed as follows:

$$precision(A,B) = \frac{|A \cap B|}{|A|},\ recall(A,B) = \frac{|A \cap B|}{|B|}.$$
$$H(a,b) = \frac{2 \cdot a \cdot b}{a+b}$$
$$F_1(A,B) = H(precision(A,B), recall(A,B))$$

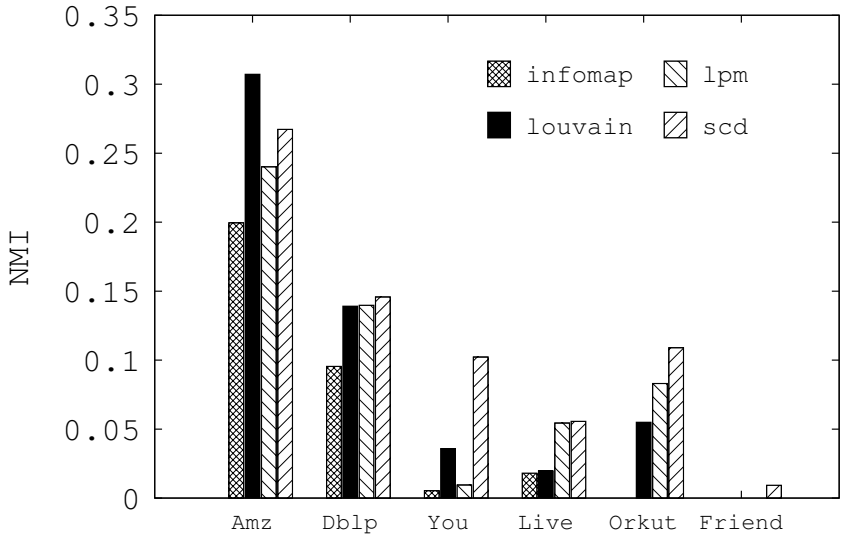


Figure 5.3: NMI.

Then, the average F1Score of two sets of communities C_1 and C_2 (which in our case are the partition and the ground truth communities respectively) is given by:

$$F_1(A, C) = \arg\max_i F_1(A, C_i),\ c_i \in C = \{C_1, \cdots, C_n\}$$

$$\bar{F_1}(C_1, C_2) = \frac{1}{2|C|} \sum_{c_i \in C} F_1(c_i, C') + \frac{1}{2|C'|} \sum_{c_i \in C'} F_1(c_i, C)$$

We also compare the quality of the results obtained using the *Normalized Mutual Information* (NMI), which is widely used in the community detection literature [15].

Figures 5.2, 5.3 and 5.4 show the Average F1Score, NMI and WCC of the partition obtained by the different algorithms respectively, in the tested graphs. Those missing bars are from executions that were not able to finish within a week or consumed too much memory. We observe that there is a strong correlation between WCC, and the F1Score and NMI obtained, that is, in

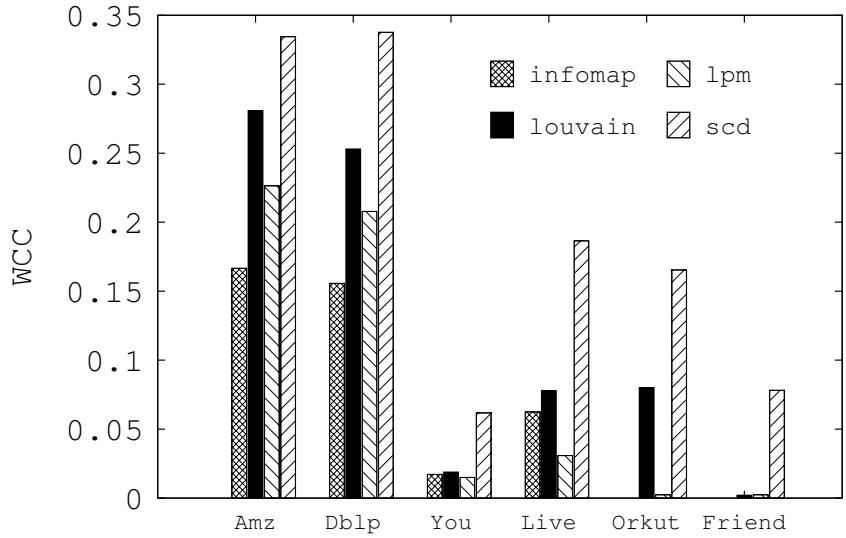


Figure 5.4: WCC.

general, the larger the WCC, the better the F1Score and NMI obtained. In general, SCD is the algorithm with the best quality using both metrics, except for the Amazon graph using NMI. We also see that the larger the graph, the larger the difference between SCD and the second, which is related to the fact that SCD is *scale independent* while existing algorithms are not.

In order to quantify the correlation between F1Score, NMI and WCC, we computed the Pearson Coefficient of variation that resulted **0.91** and **0.83** for F1Score and NMI respectively. This indicates a very strong agreement between both metrics and WCC since it is close to 1, which is the maximum value. Therefore, WCC proves to be a solid metric for evaluating the quality of community detection algorithms.

5.3 Performance, scalability and memory consumption of SCD

In Figure 5.5 we show the execution times of the different algorithms, for the different graphs. We see that SCD is the fastest algorithm for the smaller

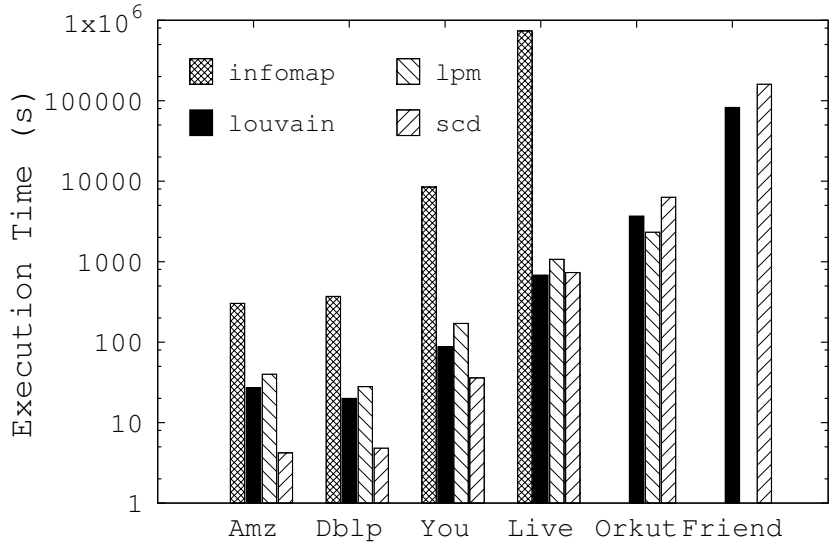


Figure 5.5: Execution times of the different algorithms single threaded.

graphs, and the fastest after LPM and Louvain when these become larger. However, the execution times are still competitive compared to the implementations of the state of the art algorithms used. Again, those missing bars belong to those executions that were unable to finish in less than a week or due to memory consumption.

We parallelized SCD in order to exploit the resources of current multi-core and many-core processors. More concretely, we parallelized the two most time consuming parts of the algorithm: the computation of the global and local clustering coefficient of the vertices during the graph clean up phase and the whole refinement phase. In the former, we parallelized the loop that computes, for each edge, the number of triangles that the edge closes. In the later, we parallelized both the loop in Line 7 of Algorithm 2, which calls the function `bestMovement` for each of the vertices in the graph and the computation of WCC for the partition at the end of the iteration (which can be parallelized for each vertex). Since all the parallel code is in the form of loops, we used OpenMP with dynamic scheduling, using a chunk size of 32. Figure 5.6 shows the normalized execution times of SCD with different number of threads. In

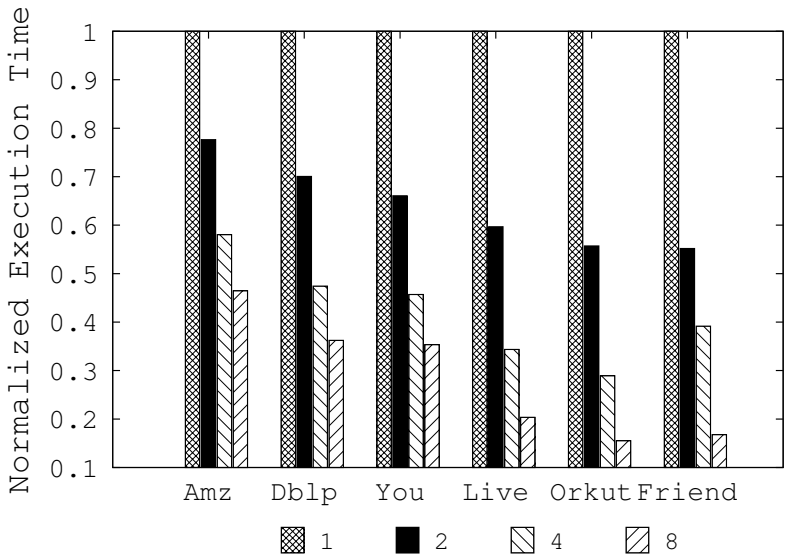


Figure 5.6: SCD normalized execution time with different number of threads.

this experiment, we have excluded the time spent in I/O, which includes reading the graph file and printing the results.

Broadly speaking, we see that with a simple OpenMP based parallelization, SCD is able to achieve very good scalability, specially for the larger graphs which are also those with a larger average degree. The larger the average degree of the graph, the larger the cost of those parts that have been parallelized: the larger the cost of computing WCC and the larger the number of movements to test between vertices and communities). These two parts quickly become dominant over the sequential ones. This means a better scalability due to a direct application of Ahmdal's Law.

We see that with a simple OpenMP based parallelization, SCD is able to achieve very good scalability, specially for the larger graphs which are also those with a larger average degree. The larger the average degree of the graph, the larger the cost of those parts that have been parallelized: the larger the cost of computing WCC and the larger the number of movements to test between vertices and communities). These two parts quickly become dominant over the sequential ones. This means a better scalability due to a direct application

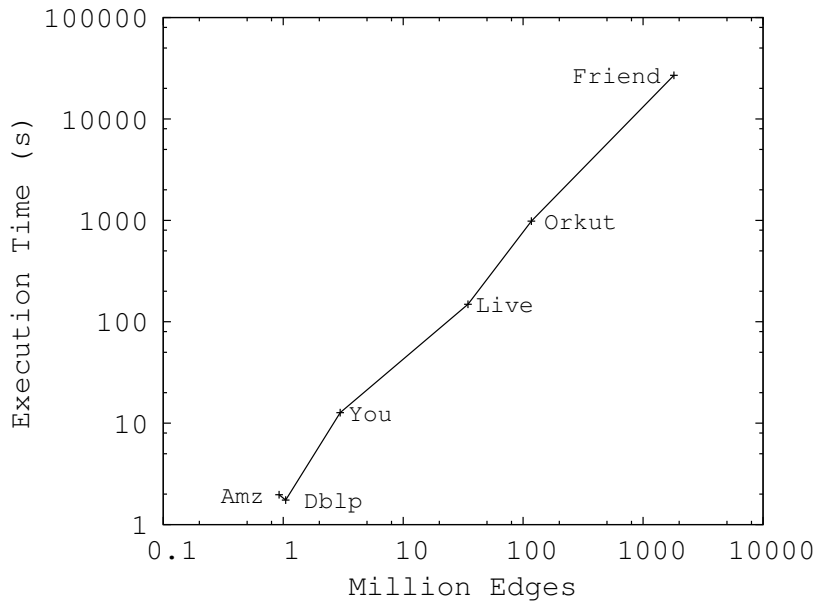


Figure 5.7: Execution time with eight threads vs number of edges.

of Ahmdal's Law. Finally, Figure 5.7 shows the execution time of SCD with respect to the number of edges of the graph. Each point represents the time spent by the eight thread version of SCD for the different graphs. We see that SCD shows a quasi linear scalability in practice, and that is able to process the Friendster graph, which contains almost two billions of edges, in about seven hours.

In Table 5.3, we show the memory consumption in MB of SCD for each of the graphs divided into three categories: Graph, Storage of Triangles and Partitions.

In general, the amount of memory consumed by SCD is dominated by the graph's representation which mostly depends on the number of edges of the graph. On the other hand, the other data structures (array of triangles and partitions) built by SCD scale linearly with the number of vertices of the graph, and not with the number of edges. We see that the amount of memory consumed for the Friendster graph is roughly 18GB, showing that much larger

Table 5.2: SCD Memory consumption in MB.

	Graph	Triangles	Partitions	Total
Amazon	11.4	1.3	16.0	28.7
Dblp	12.2	1.3	14.9	28.4
Youtube	37.5	4.5	68.9	110.9
Livejournal	325.4	16.0	197.7	539.1
Orkut	974.3	12.3	124.4	1111.0
Friendster	15235.8	262.4	3317.6	18815.8

graphs could be processed with a commodity server with 128 GB of memory. This supports the idea of proposing an algorithm for in memory machines, as these can perfectly manage the structural information of massive graphs of the order of billions of edges.

Chapter 6

Ground-truth vs Synthetic Communities

Most of the existing work in community detection is evaluated using synthetic graph generators, specially the LFR benchmark, which generates graphs with a power law degree distribution, and outputs communities based on the informal community definition (more internal edges than external) [25]. However, evaluating communities using LFR or other simple graph generators, has a very clear shortcoming. These benchmarks impose a community definition, which is usually very simplistic and it is not clear whether it corresponds or not to those structures found in real data. The result is that algorithms designed to work well with these graphs sink when they try to recover the communities observed in real datasets such as those with ground truth communities. Communities created based on network's metadata (aka ground truth communities, meta-communities or user defined communities), are not necessarily characterized by the same features or structures as those assumed in structural communities. But on the other hand, it is clear that having synthetic data generators capable of reproducing the characteristics of real datasets is of high importance, not only for benchmarking community detection algorithms, but also for benchmarking database systems [14] or to model complex systems more accurately, as they provide a flexible way to obtain real data when this cannot be easily obtained from real sources.

Therefore, there is a need to better understand which are the features that characterize the meta-communities found in real data, and whether existing community detection benchmarks are able to reproduce these characteristics or

not. For this reason, in [47] we analyze a set of real graphs with ground truth communities (Amazon, Dblp, Youtube and Livejournal) (see Chapter 5 for their particular characteristics), by means of their community distributions using several structural indicators (clustering coefficient, conductance, bridge ratio, tpr, size and diameter) (See Chapter 2 for their corresponding definitions), with the goal of understanding how the meta-communities in those datasets look like.

Then, we compare those communities observed in real graphs to those generated by two synthetic graph generators with a community structure: the LFR benchmark and the LDBC Social Network Benchmark data generator (LDBC-DG). LFR was designed as a benchmark for evaluating community detection algorithms [26]. Compared to other graph generators, its principal characteristic is that the building procedure is based on creating a graph that connects communities. LFR starts by generating a set of communities of different sizes following a power law distribution. Then, the edges between the vertices in the graph are created in such a way, that they follow power law distributions and for each vertex the mixing factor is fulfilled. The mixing factor is a parameter indicating the percentages of neighbors of every vertex that belong to a different community than that the vertex belongs to. A recent study indicated that communities are too well defined, and do not capture the noise found in real data [40]. We downloaded the generator from the author's web site.

The LDBC-DG is a fork of the S3G2 graph generator [44], which is customized to build social network datasets, which is used in the LDBC benchmarking initiative [14]. The LDBC aims at designing realistic and meaningful benchmarks for linked database systems, namely RDF and graph databases. The LDBC generator generates complex synthetic social-networks with many attributes related to the users and its activities in the network. The resulting schema is similar to the contents available in Facebook. For example, users have attributes that indicate their personal description (name, born place, school/university, etc); the friends of a user; posts and photographs created by a user; groups created by users indicating interests... It starts by generating a set of users with attributes following distributions found in the real world. Then, they sort the users in successive Hadoop jobs by different correlated attributes (i.e. user interests, user universities, etc...) and create friendships between users using a sliding window procedure, where users close in the

window have a higher probability to be friends. Following this schema, we create the communities using a similar procedure to the one described in the ground truth, by setting as a community each connected component of users (using the friends relation) that belong to a group. We downloaded the latest available version of LDBC-DG from the Github repository of LDBC on 30th March 2014.

6.1 Experimental Setup

Synthetic graph generators have several parameters that can be tuned to produce graphs of different characteristics. For both LFR and LDBC-DG, we generate a network with 150K users. In the case of LFR, we set the average and maximum degree to 10 and 400 respectively, and the minimum and maximum community size to 10 and 10000, respectively. One third of the vertices are set as overlapping vertices, and belong to three different communities instead of one. All these parameters have been set up as to mimic the characteristics found in the ground truth data. Finally, we have set the mixing factor of LFR from 0.1 to 0.5, therefore generating five networks named LFR1, LFR2, LFR3, LFR4 and LFR5, which in this range is expected to generate networks with communities [26]. For the LDBC-DG, we generate a single network using the default LDBC-DG parameters, which are fit to real data. The rest of the parameters for both generators are set to the default values, which are reported to generate realistic social network distributions [14, 26].

For each community, we compute all six structural indicators. Then, we analyze each indicator individually. We take each community as a sample and draw a histogram distribution. Then, we study the correlation between all pairs of histograms, by computing the Spearman correlation rank for each pair of graphs. The Spearman correlation rank test is a non parametric test that quantifies if two variables are monotonically related.

6.2 Results Discussion

Figure 6.1 shows the distributions of the statistical indicators for the Livejournal graph. We take the Livejournal graph as a representative of the rest of the graphs, which are reported in the Appendix. For the rest of the

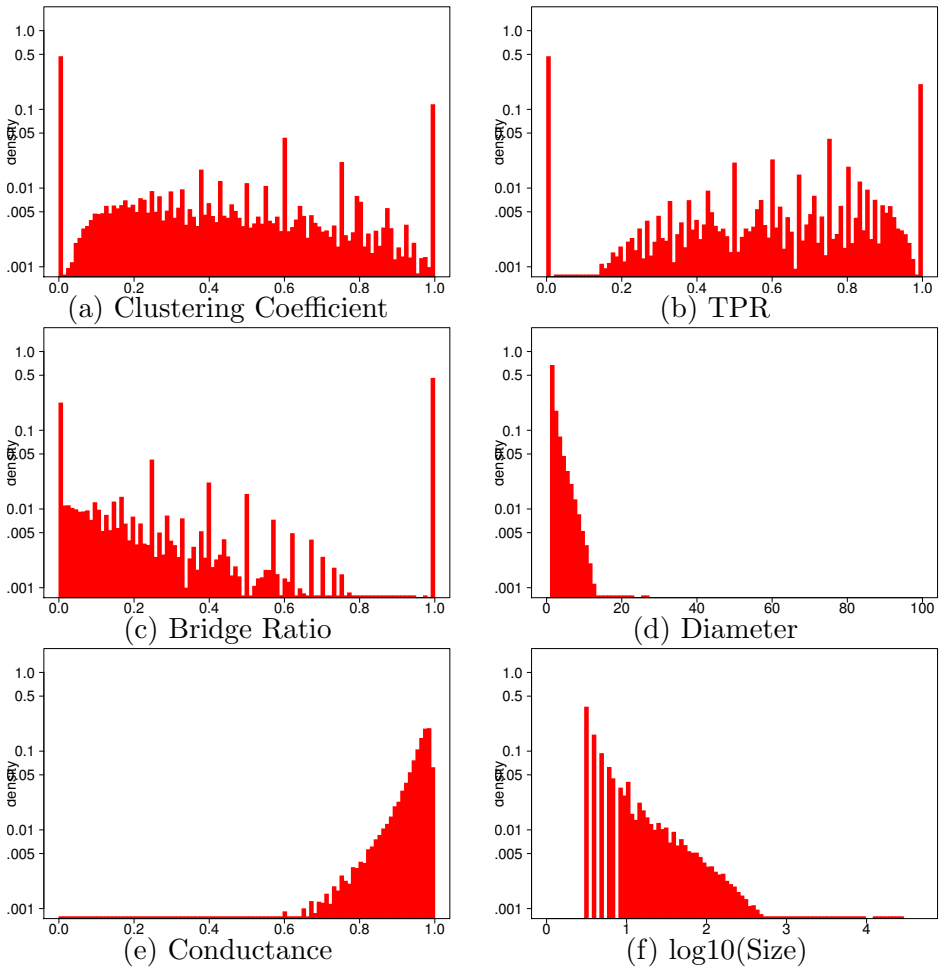
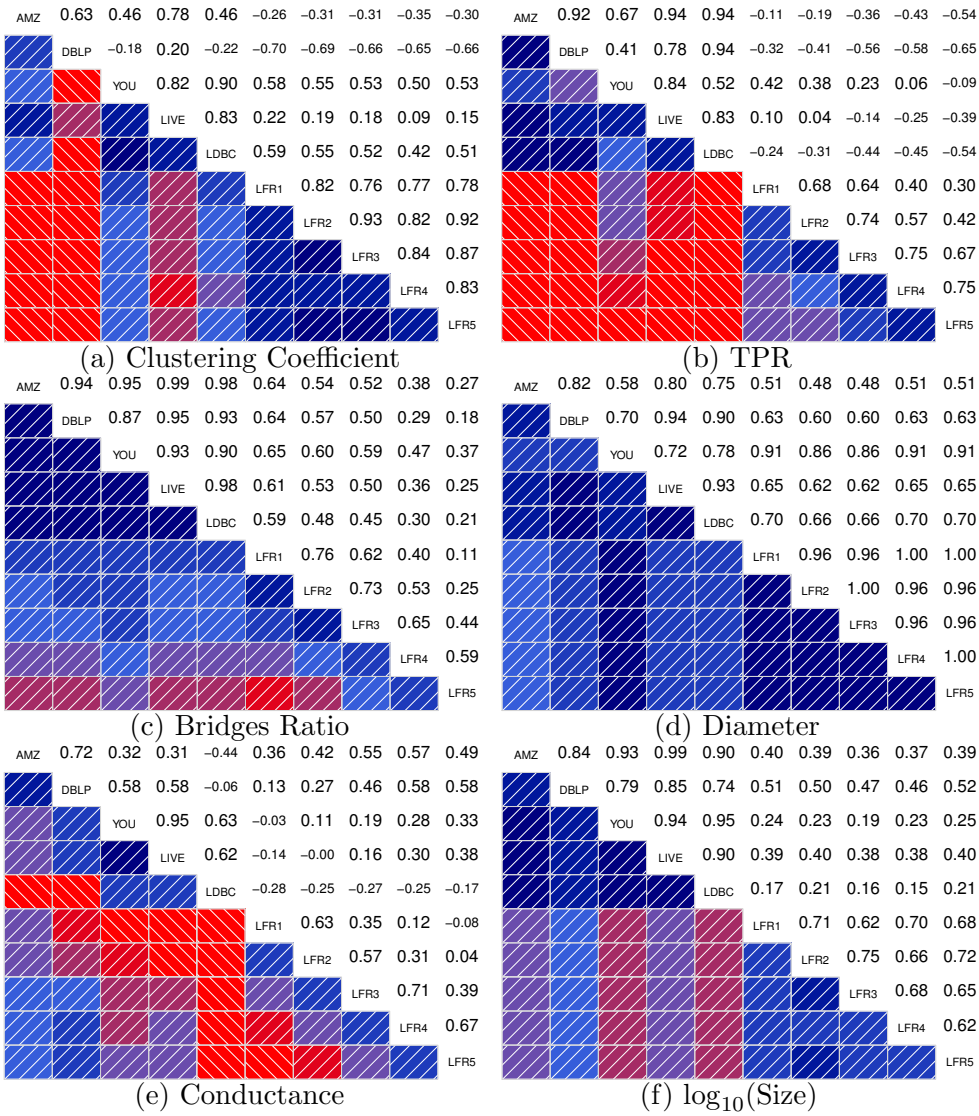


Figure 6.1: Distribution of the statistical indicators for the Livejournal graph.

real graphs, the distributions show similar characteristics as shown by the Spearman correlation tests of Figure 6.2.

Real Graphs: We start by analyzing the internal structure of Livejournal communities, hence we focus our attention on the clustering coefficient, the bridge ratio, the TPR and the diameter. Figure 6.1(a) shows a multimodal distribution. The largest peak contains 44% of the ground truth communities,



(a) Clustering Coefficient (b) TPR

(c) Bridges Ratio (d) Diameter

(e) Conductance (f) $\log_{10}$(Size)

Figure 6.2: Spearman rank correlation coefficient of the distributions between the different communities and structural indicators.

with a clustering coefficient between 0 and 0.01. This indicates that many communities have a small percentage of closed triangles. But, when we looked into detail, we found that many of those communities without triangles were very small and lots had only three vertices (59% of them). The second largest peak are communities with a clustering coefficient between 0.99 and 1, which are quasi-cliques or cliques and contain 11% of the communities. The rest of the communities fall into intermediate ranges. A similar multimodal result is seen for the TPR and the bridge ratio (Figures 6.1(b) and (c) respectively) with the two peaks at the extremes and with a trend towards participating in triangles and not having bridges in the central modal group. This multimodal distribution suggests that communities are not an homogeneous entity that can be described with a single model.

In Figure 6.1(d), we see that the bulk of the communities has a small diameter: 84% have a diameter smaller than five. This is because ground truth communities are well connected and small in many cases. We observe that conductance tends to be high and thus communities are not very well isolated as depicted in Figure 6.1(e). If we look at Figure 6.1(f), most of communities (about 74%) have a size smaller than 10. In the last three subfigures we observe that the largest fraction of the communities is small, have very small diameters, and are not very well isolated. For the three indicators, we observe a power law decay towards communities that depart from the typical community.

Figure 6.2 shows the correlograms of the Spearman rank correlation coefficient between the distribution of the different structural indicators for each pair of graphs. The upper half of the matrix shows the numerical score given a pair of variables. On the other hand, the lower half shows a color gradient, where two variables are correlated if they approach dark blue, while they are not correlated (or inversely correlated if negative) if they approach red.

The first four entries correspond to the real graphs. Broadly speaking, we observe that all four graphs show similar patterns for the six indicators. The correlation is specially strong for the bridge ratio, where the rank is over 0.9 for most of pairs of real graphs. The diameter, size and TPR distributions also show important correlations.

We observe that the less correlated distributions are for clustering coefficient and conductance, although correlation is still present. The correlations shown in Figure 6.2(a) indicate that there are differences between the clustering

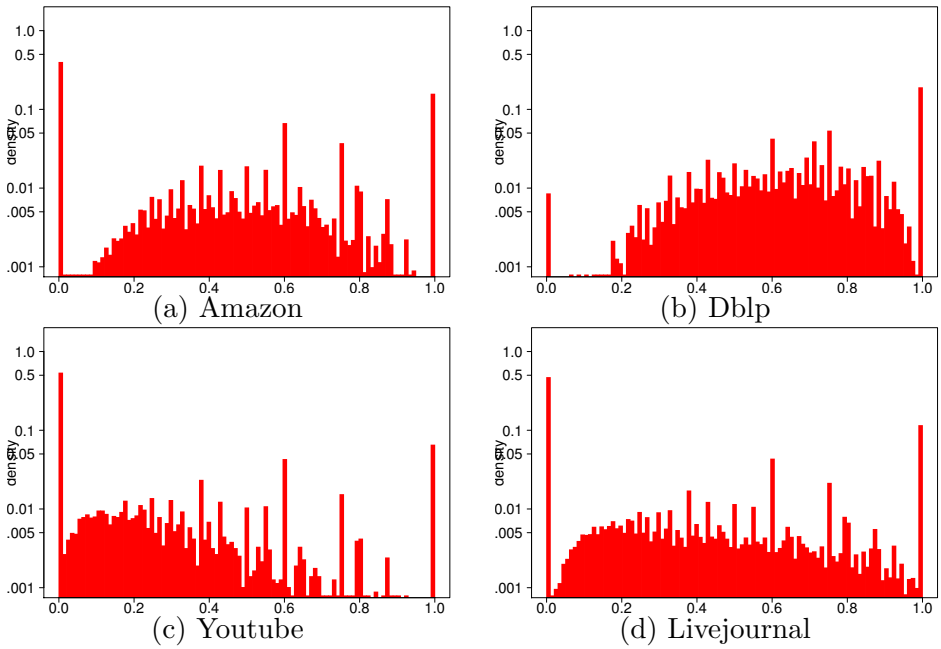


Figure 6.3: Clustering coefficient distribution of real graphs.

coefficient distributions for the real graphs, which can be visually compared in Figure 6.3. First, Youtube and Livejournal have a similar distribution, slightly biased to the left, and having similar peaks at their extremes. Second, Dblp is the graph with a distribution more biased to highly clustered communities. Furthermore, the peak extremes of the Dblp distribution are inverted compared to the rest. This explains why Dblp is not correlated with Livejournal and Youtube. Finally, Amazon lies between Dblp and Livejournal with a more centered distribution.

We see in Figure 6.2(e) that graphs have two types of conductance distributions. Figure 6.4 depicts that the conductance distribution of Amazon and Dblp is more diverse (for conductance, the smaller the better). Specifically, 63% and 73% out of the total number of communities for the Amazon and Dblp graphs respectively, have a conductance larger than 0.5. For Youtube and Livejournal, the distribution is more skewed towards the right of the chart and 98% and 99% of the communities have a conductance larger than 0.5.

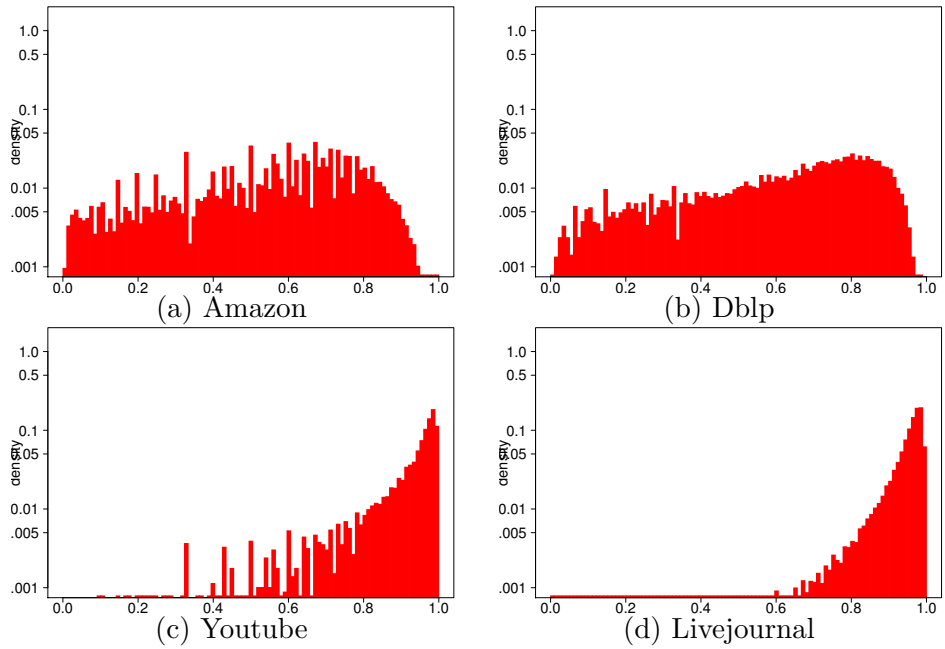


Figure 6.4: Conductance distribution of real graphs.

LDBC-DG Graph: Figure 6.5 shows the distributions of the structural indicators for the LDBC-DG graph and the fifth row of each correlogram in Figure 6.2 shows the correlation of each LDBC-DG plot with the real datasets. We observe that for most indicators the synthetic distributions are considerably similar to those for the real graphs, specially for Youtube and Livejounal.

First of all, the LDBC-DG reproduces the multimodal distributions of the clustering coefficient, the TPR and bridge ratio (Figures 6.5(a-c)). The multimodal clustering coefficient distribution of LDBC-DG shows a central part biased towards communities with a small clustering coefficient. This is similar to what we see for Youtube and Livejournal graphs. We find that the generator distributes evenly the triangles among the members of the communities, as shown by TPR in Figure 6.5(b). More specifically, 63% of communities have a TPR larger equal or larger than 0.5. Figure 6.5(d) shows the diameter distribution of the LDBC-DG communities. Compared to those found in real graphs, LDBC-DG communities have a slightly larger diameter, with 71% out of the total number of communities with a diameter less than 6.

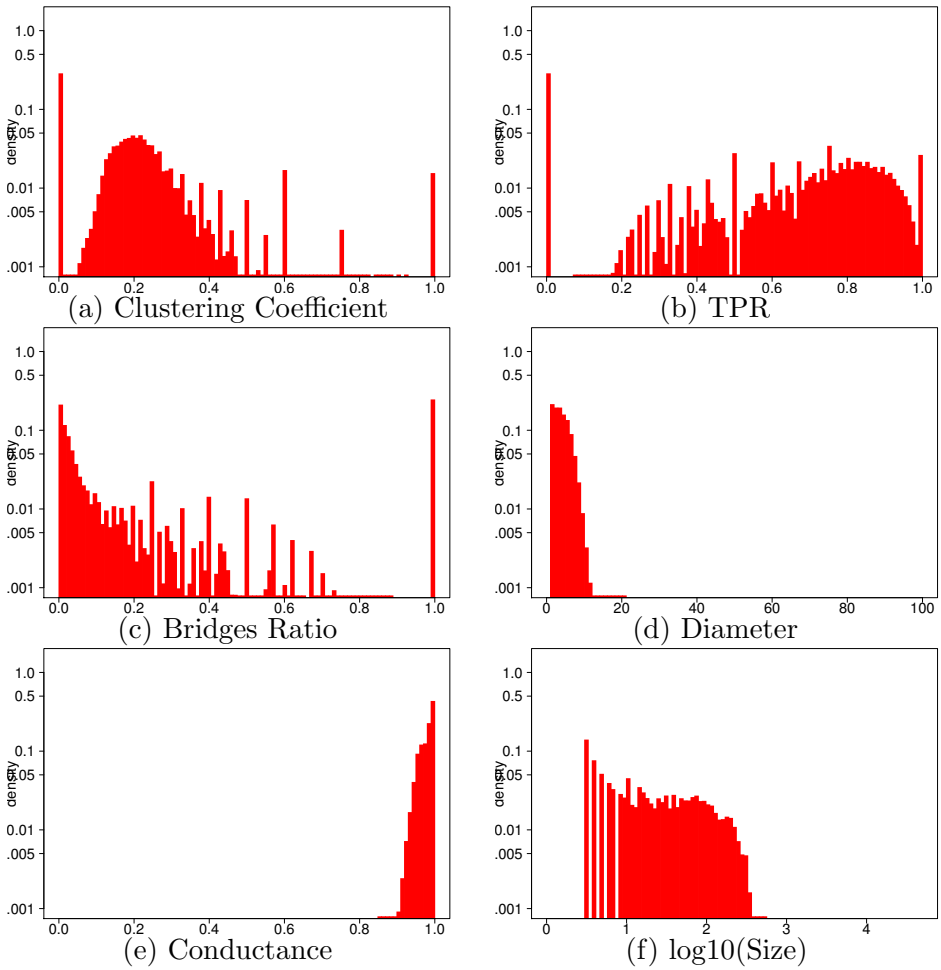


Figure 6.5: Distribution of the indicators for the LDBC-DG graph.

When we turn to analyze the conductance, as shown in Figure 6.5(e), we see that as with the real graphs, the LDBC-DG communities tend to have a large conductance, similarly to those found for Youtube and Livejournal. However, we note that the distribution is significantly more skewed to the right Thus, LDBC-DG communities are less well isolated than those in the real datasets.

In general, we see that the LDBC-DG reproduces many of the characteristics found in real graphs, specially those found in Youtube and Livejournal. Since

LDBC-DG models an online social network data, it seems natural that the communities generated resemble more the datasets from online social networks than the product and coworker network.

LFR Graphs: For the LFR graph, only the diameter distribution shows a strong correlation to those found in real graphs as shown in the last five rows of Figure 6.2. For the rest of the indicators, the degree of correlation is moderate or weak, though it varies depending on the mixing factor configuration. In order to better understand the characteristics of the community structure of the graphs output by the LFR generator, we show the distributions for the mixing factor 0.3 configuration in Figure 6.6.

In contrast to LDBC-DG, LFR does not produce the multimodal distribution for clustering coefficient (Figure 6.6(a)) observed in real graphs. LFR does not produce communities with a large clustering coefficient. According to Figure 6.6(b), the TPR distribution also lacks a peak for large participation ratios, and in contrast to LDBC-DG it also lacks the peak for the low TPR modality found in real graphs.

The bridge ratio (Figure 6.6(c)) distribution of LFR has moderate correlation to the real data, but the peak on the left extreme is missing and the peak on the right is smaller than the real ones. The diameter distribution is quite similar to that found for the real data, but with some more large diameter communities.

Conductance has a poor match with the real datasets. LFR produces a distribution centered in a certain value of conductance, as shown in Figure 6.6(e). This peak depends on the mixing factor (see Appendixes for more details), and goes towards the left when the mixing factor is large. Then, configurations of LFR with larger mixing factors produce more realistic conductance plots because they have larger conductances. However, these larger mixing factors, such as LFR5, are much worse in terms TPR, bridge ratio and size as seen in Figure 6.2.

We have observed that the main weakness of LFR is the regularity of the communities created. Since all the communities follow a single model, LFR is not able to create the multimodal distributions present for some indicators.

Broadly speaking, two main results can be derived from the analysis. The first, is that ground-truth communities are more diverse and complex than the idealistic vision of traditional structural communities. The distributions of the different structural indicators show that ground truth communities can be either internally well defined (large clustering coefficient) or loosely defined (small clustering coefficient), but that in general, they have many edges pointing outside (high conductance) the cluster. However, we also see that the triangle plays a significant role in defining the internal structure of the communities, as these are observed frequently.

This suggests that understanding the more complex structures that define these communities is key to better design effective algorithms for their detection. Actually, not properly understanding these structures is one of the reasons why traditional community detection algorithms and metrics based on just edge counting fail at their detection.

The second result is that synthetic graph generators based on the traditional view of structural communities (such as LFR) generate graphs with a community structure very different from that observed in ground truth communities. On the other hand, the LDBC-DG graph generator is able to reproduce many of the features observed, thus producing a community structure more similar to that observed in real graphs.

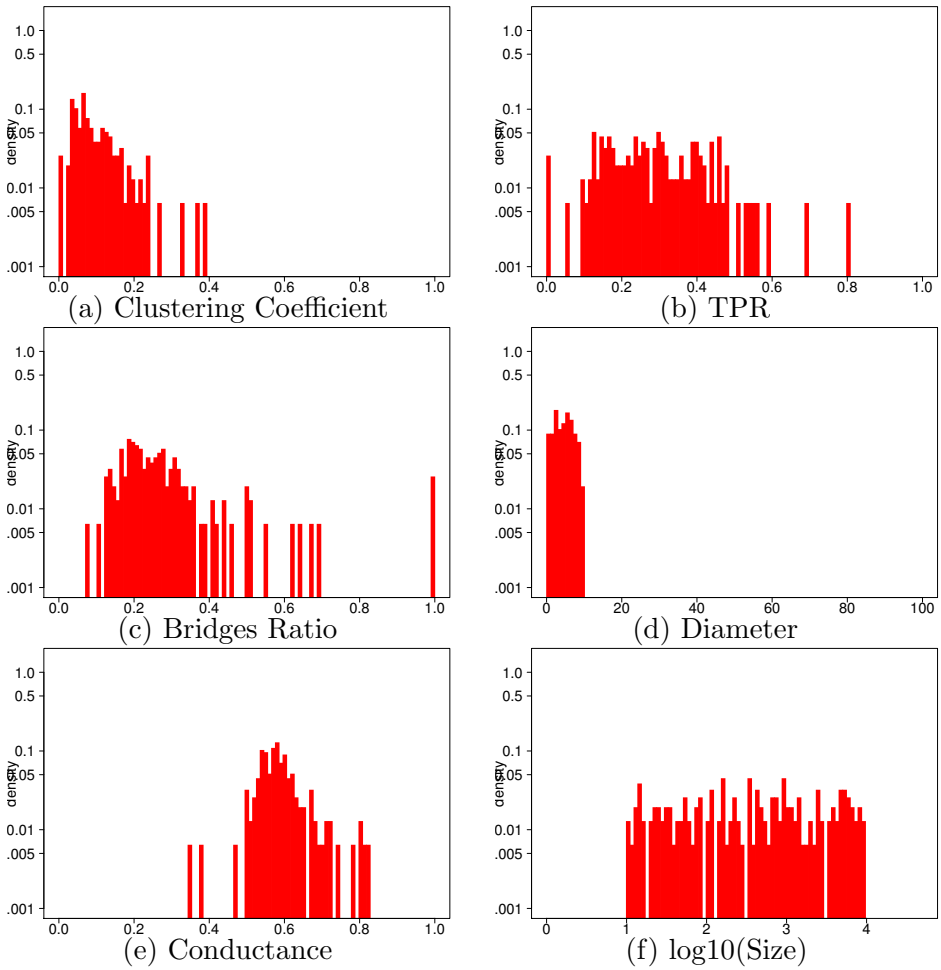


Figure 6.6: Distribution of the indicators for the LFR3 graph.

Chapter 7

Triangle counting in future many-core micro-architectures

As manufacturing processes evolve and more cores are added to a chip, the higher the pressure to the interconnection network and integrated memory controllers, higher synchronization costs and fewer shared resources. For this reason, computer architects are thinking of alternative architectures with simplified and non-coherent cache hierarchies, which can scale better (to tens or hundreds of cores), and at the same time save die area and power. As a downside, this comes at the cost of more complex programming models as the coherency of the caches have to be managed by the programmer.

One of such architectures is the Intel SCC (Single-chip Cloud Computer) [32]. The Intel SCC is a 48-core chip with shared memory, but non-coherent caches. Since flushing caches to achieve data coherence between cores is prohibitively costly, the Intel SCC incorporates a set of message passing buffers that can be used as communication channels between any pair of cores or as on-chip local memories for the most accessed data. In this piece of work [51], we explored the suitability of the Producer-Consumer programming model for the Intel SCC, by means of the triangle counting problem.

Counting triangles is a very important operation not only for the computation of WCC, but also for social networks and other types of network analysis in general. Furthermore, as many graph algorithms, triangle counting is a memory bound operation that also exhibits a very non-local memory access pattern. This makes triangle counting a good representative of a modern

application to test the suitability of the Producer-Consumer model on the Intel SCC.

We implemented triangle counting for the Intel SCC, and showed that the Producer-Consumer model is able to exploit both data and task parallelism, maximizing the usage of resources and achieving high scalability and performance (running up to 9 times faster than a baseline working on main memory).

7.1 The Intel SCC

The Intel SCC processor is a 48-core vehicle created by Intel Labs as a platform for many-core software research. This is a clustered architecture composed of P54C cores, grouped in tiles containing two cores each. Each tile (pair of two cores) has a router, forming a communication mesh within all the chip to access the four DDR3 memory controllers.

The Intel SCC has an aggregated address space of 64 GB of memory. However, each P54C core is able to address 4 GB. In our experiments, each core has a private region of memory of total size divided by 48 assigned. What makes the Intel SCC special is its non-coherent memory hierarchy. Each core has a non coherent L1 and L2 caches, of 16KB and 256KB respectively, with a cache line size of 32 bytes. Since the caches are non-coherent, it is the programmer who is responsible for maintaining the coherency of the caches manually. For this reason, the Intel SCC provides a fast core-to-core communication, consisting on 384KB of on-chip memory (also called the Message Passing Buffer(MPB)). Each tile is assigned 16KB of the buffer (8KB for each core), which is addressable by any core. We will call each 8KB section assigned to each core, an MPB section. Finally, in order to synchronize the access to the MPB and the memory by all the cores, the system provides 48 globally accessible test-and-set registers.

7.2 Producer consumer implementation of triangle counting

In the Producer-Consumer(P/C) model, the program is divided into tasks which adopt the role of a Producer, a Consumer or both. The Producers are

tasks that operate on the input data and produce the results, which are then sent to the Consumers. The Consumers are the tasks that receive the data from the Producers, operate on the data and then produce the results, which can be either stored in main memory or forwarded to the next Consumer in the P/C chain (and hence performing the Producer's role too).

The Producers and the Consumers can be executed independently and concurrently as long as they have available data to consume and available resources to store the results they produce. Hence, the task parallelism is exploited. Furthermore, the input data of the tasks, can be split and distributed among multiple instances of both Producers and Consumers, allowing the partitioning of the data and the execution of multiple tasks in parallel. This way, we are also exploiting the data parallelism.

Given a graph $G(V, E)$ (where V is the set of vertices and E is the set of edges), the Producers compute, for each edge e in E, the number of triangles the edge e belongs to. Given an edge e connecting two vertices a and b, the number of triangles of this edges corresponds to the size of the intersection between the adjacency lists of vertices a and b. In order to exploit data parallelism, different Producer instances are created, and each of them is assigned a subset of the edges to process. Then, the Consumers read the results produced by the Producers and accumulate, for each vertex, the number of triangles where the edges pointing to the vertex participate in. Several instances of Consumers are created, and each of them accumulates the result of a subset of the vertices of the graph. All the Producers and the Consumers are executed concurrently, as long as the data is produced and consumed.

We implemented three strategies to exploit data and task parallelism using the P/C model.

DataParallelMsg: It follows the data parallel paradigm and uses main memory to exchange the data between the tasks. In this strategy only one task is executed at a time, which means that all the cores execute the Producer step first, and when it is completed, all the cores execute the Consumer step. Each core is responsible for $1/P$ of the computational work, and the results are stored in main memory. Once all the cores have finished executing the Producer step, the results are broadcast to the other cores, by storing their results into their MPB sections. The other cores read the results and store

them again into their addressable main memory. Once all the cores have all the data, the Consumer step starts.

DataParallelMsgBlk: This implementation, follows the data parallel programming model like the previous implementation presented, but in this case, the results of the Producer step are iteratively produced in blocks of the size of the MPB section, instead of producing all the results and then broadcast them. Once all the Producers have finished to produce their chunk of data, the cores consume the data produced by the rest directly from the MPBs of the others, and execute the Consumers step. The process iterates until all the computations from the Producer step have been performed. The goal of this approach is to benefit from keeping all the data produced in the Producers' step inside the chip, instead of copying it into the main memory and hence achieving better performance.

Task&DataParallel: The goal of this implementation, is to fully benefit from the presence of the MPB, by implementing a version based on the task and data parallel paradigm. In this approach, we have k cores as Producers, and $P-k$ cores as Consumers and all cores execute their task at the same time. The Producers produce the data and store them directly into their MPB section, while the Consumers consume these data to perform their computations. A Producer computes and produces data as long as there is space in its MPB section. Once the buffer is filled, it waits until the data is consumed by all the consumers and the buffer is freed. Once the Producer has performed all the computations, it finishes. On the other hand, a Consumer waits for the data to be in the buffer. Once the data is available, it consumes them and tells the corresponding Producer that the data has been consumed. In order to reduce the contention on the buffer between Producers and Consumers, a double buffering scheme is used in every MPB section.

7.3 Experiments

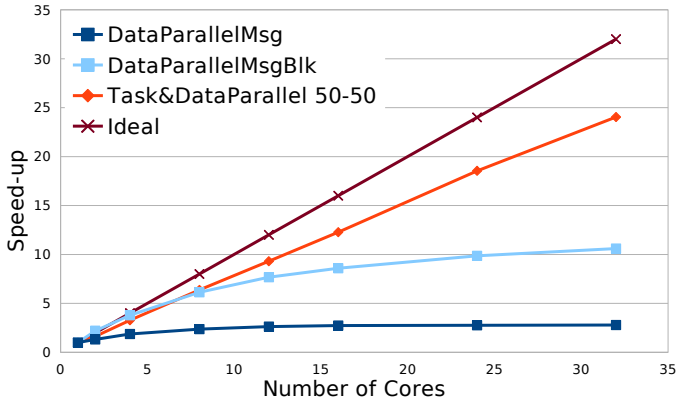
We used the Intel Single Chip Cloud Computing (Intel SCC) experimental processor, RockyLake version configured with 32GB of main memory. The frequencies of the tiles, mesh and memory are 533, 800 and 800 Mhz respectively. The operating system used for the Intel SCC cores is the Linux kernel provided by the RCCE SCC Kit 1.3.0.

To test the performance of the different implementations proposed, for the P/C model, we have implemented the local triangle count problem described in as described above. The input graphs, are built with the LFR graph generator [25], which creates graphs with social network characteristics. We generate two graphs: one with 100K vertices and 1M edge (*small*), and another one with 1M vertices and 10M edges (*big*). Each measure is obtained by averaging five executions.

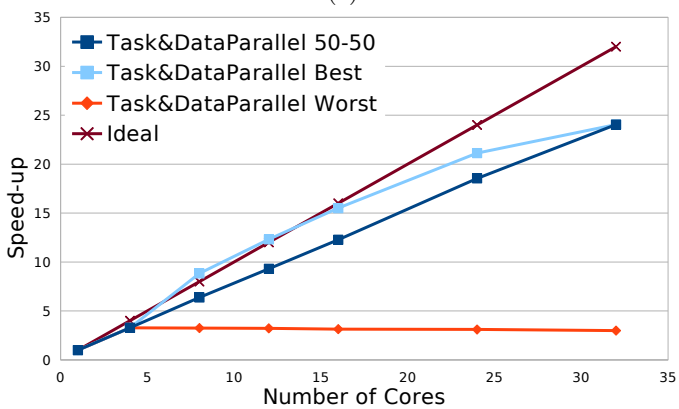
We show the results for the big graph. For a complete description of the results, please refer to [51]. In Figure 7.1(a), we show the obtained speed-up using the different strategies for different number of cores. For the Task&DataParallel strategy we set the number of Producers and Consumers to 50% and 50% out of the total respectively. As a baseline, we use the DataParallelMsg strategy with one core. We see that the data parallel version does not scale well, with a speed-up limited to 2.6x regardless of the number of cores used. This limit is due to the time to communicate the producer's results to the consumers, which become dominant as long as the number of cores increases.

For the optimized version of the data parallel approach, the *DataParallelMsgBlk* implementation, it scales a bit better but it is still limited to 10x for the larger number of cores used. The main difference between this implementation and the *DataParallelMsg* is that instead of the Producers writing the results into their MPB sections and the Consumers copying them to main memory, the Consumers make the computations by directly reading from the MPB's of the Producers. This does not change the communication pattern but does have an impact on the communication latency as the the main memory accesses are avoided. Consequently, we can observe that the speedup for this optimized version follows a similar trend to the *DataParallelMsg* implementation but the speedup achieved is higher as the communication fixed cost is smaller.

Finally, while both data parallel implementations showed a limit in their speedup, the *Task&DataParallel* implementation shows a near linear speedup even for large number of cores (about 24x of speedup for 32 cores). The reason is that Producers and Consumers are executing at the same time in a pipelining way, thus, we are able to hide the communication cost with the Producer and Consumer computation. The reason for the speedup to be slightly below the ideal has to do with another fact which is related to load



(a)



(b)

Figure 7.1: (a) Scalability using the different P/C strategies. (b) Scalability of the Task&DataParallel strategy for different assignments of cores to tasks.

balancing and contention on the double buffer space used to exchange the results between Producers and Consumers.

We also tested different assignment of cores to tasks for the Task&DataParallel strategy, in such a way that we executed the application on different configurations ranging from 2 up to 30 Producers and 2 up to 30 Consumers, with a maximum of 32 cores in total. Figure 7.1-(b) shows, for each fixed number of cores, the speedup for the worst, the best, the *50-50* and the ideal configurations. We observe that the *50-50* strategy performs well, close to the

Best and *Ideal*. It is also relevant to notice that the gap between the *Best* and *Worst* is quite large. This experiment illustrates the fact that with a smart dynamic scheduling for task assignment, we can potentially obtain a very good performance using the P/C model.

In conclusion, we see that the Producer Consumer model is a potentially good model for future many-core architectures with on-chip memories and non-coherent caches. Moreover, we see that triangle counting is a good candidate to benefit from such architectures due to its data bound computation and non-local memory accesses.

Chapter 8

Conclusions and Future Work

In this thesis, we have explored one of the hottest research topics in the past decade: the community detection problem. We have studied the existing approaches for community detection, and have detected the different shortcomings affecting them when it comes to detect communities in social networks. The problem is that existing methods try to be generic but ignore the specific characteristics that characterize the communities of each domain. As an outcome of this analysis, we have proposed a *domain specific community detection* design methodology, which consists of defining

1. a set of structural properties describing how the communities detected by a metric must be.
2. a set of behavioral properties describing how the metric must behave at different circumstances (e.g. at different scales).

Both structural and behavioral properties must be tailored for a given application domain.

In our case, we have focused on the particular case of social networks. As such, we have proposed three structural properties:

1. **Internal structure sensitive**: a community metric for social networks must be sensitive to the internal community structure.
2. **Bridge resistant**: communities do not contain bridges.

3. **Cut Vertex resistant**: communities do not contain cut vertices.

Similarly, we have proposed three behavioral properties,

1. **Scale Independent**: a community metric must not depend on the total size of the graph and be robust at any scale.

2. **Linear Community Cohesion**: the amount of links between community members must grow linearly with respect to the size of the community.

3. **Adaptive**: a community metric must look both into the internal and external community connectivity in order to adapt to the inhomogeneities of the graph.

Based on the domain specific community detection methodology, we have proposed a new community detection method called *Weighted Community Clustering* (WCC). WCC takes the triangle as the basic motif indicating the presence of structure between two vertices of the graph. The adoption of the triangle has been done after the observation that transitive relations are very common in social networks as analyzed in the existing literature.

We have mathematically proven that WCC is able to fulfill the behavioral and structural properties defined and as a side outcome of our analysis, we have also found the detectability threshold of WCC by means of the stochastic block model. Finally, we have also shown experimentally that, where others fail at correctly ranking the quality of a community, WCC is a robust and reliable community detection metric.

Based on WCC, we have designed a novel community detection algorithm called *Scalable Community Detection*(SCD). SCD has been crafted to scale to large graphs on Shared Memory Machines (SMP). SCD has been proven to be reliable and perform better than the most popular community detection algorithms of the state of the art, specially on large graphs, using real datasets with ground truth communities. SCD is able to find the communities on a graph of 1.8 Billion edges in about 7 hours, and exploit all the resources of current multi-core processors by showing an almost linear speed-up.

We have also analyzed the community structure of several real datasets with ground truth communities and revealed that these significantly depart from the typical informal community definition. We have observed that unlike the common belief that communities are highly isolated sets of nodes with a large internal edge density, ground truth communities exhibit a very large conductance and a quite small degree of clustering. However, we also observe that triangles are a very common structure inside these communities. Overall, the observed structure suggests that we need more complex ways to discern weak than strong links than just counting the number of edges, possibly tailored after specific domains of the data.

Finally, we have also studied the problem of triangle counting in a modern architecture such as the Intel Single-chip Cloud Computer. Our study has revealed that the Producer-Consumer programming model is a suitable model for architectures with non-coherent caches and on-chip memory buffers, and have shown that applications such as triangle counting, which are memory bound and exhibit very non-local memory accesses, can greatly benefit from such architectures.

8.1 Future Work

Community detection is a recent research topic and as such, many questions still remain to be answered. In this thesis we have designed a metric (WCC) and a community detection algorithm (SCD) following a novel methodology based on looking at the particular structures and characteristics that define the communities on a given domain, which in our case are social networks. Therefore, future work will include exploring the application of these ideas to other domains by investigating which are the particular characteristics and structures that characterize their communities.

WCC is a metric that works at a specific level of resolution. In other words, it explicitly establishes a fixed relation between structural isolation and structural intraconnectivity. This relation directly affects the detectability threshold of the metric. Whether a given threshold is good or not might depend on the needs of a particular application. Future work will include extending WCC to incorporate mechanisms to allow tuning this threshold to better meet the needs of the end user. We have already done some progress in this sense, by adding an extra parameter to WCC that allows tuning the degree

of importance of structural isolation and structure intraconnectivity in the community definition. In a similar way, understanding whether the important structures that characterize a community change depending on the desired resolution level is also important.

In this work, we have focused on simple undirected and unweighted graphs. In some applications (e.g. Twitter like networks), edges are directed and/or contain weights in the edges (e.g. road networks). Future work will also include extending WCC to other types of graphs with directed and/or unweighted edges.

Another line of research is that focused on designing metrics and algorithms for other types of communities, such as overlapping communities (where nodes can belong to more than one community) or egomunities (where communities are built around a specific set of vertices, possibly overlapping as well). In these cases, it is important to understand what characterizes the overlap between communities and how this compares to the non-overlapping zone.

Finally, real data is not static but changes rapidly. Therefore, there is a need for approaches that are able to detect time-evolving communities, how they are born, live and die. Even though there has been a great progress and lot of contributions during the last years, the problem of community detection is still far from being solved.

Chapter 9

Appendix

9.1 Proof of Proposition 1

Proof. (i) This is a consequence of the inequalities $t(x,S) \leq t(x,V)$ and $vt(x,S) \leq vt(x,V)$, $vt(x,V) \leq vt(x,V) + |S \setminus \{x\}| - vt(x,S)$

(ii) If $WCC_v(x,S) = 0$, then at least one of the following three identities holds: $t(x,V) = 0$, $vt(x,V) = 0$, and $t(x,S) = 0$. Now, each one of these conditions implies $t(x,S) = 0$. Reciprocally, by definition, if $t(x,S) = 0$, then $WCC_v(x,S) = 0$.

(iii) Assume $WCC_v(x,S) = 1$. By (ii), $t(x,S) \neq 0$. Hence, there exists an edge $\{y,z\}$ such that $y \in S \setminus \{x\}$ and $z \in S \setminus \{x\}$ forming triangle with x. Then $|S \setminus \{x\}| \geq 2$. As the two fractions defining $WCC_v(x,S)$ are ≤ 1, the condition $WCC_v(x,S) = 1$ implies that both fractions are 1. Left fraction is 1 if and only if $t(x,V) = t(x,S)$, which implies that $vt(x,V) \leq |S \setminus \{x\}|$, which turns into an equality (and therefore right fractions becomes 1) if and only if $vt(x,S) = |S \setminus \{x\}|$.

Reciprocally, the condition $vt(x,V) = |S \setminus \{x\}| = vt(x,S) \geq 2$ implies $t(x,S) = t(x,V)$. As $vt(x,V) = vt(x,S) = |S \setminus \{x\} \geq 2$, we have that both fractions in the definition of $WCC_v(x,S)$ have denominator $\neq 0$ and both fractions are 1. Therefore, $WCC_v(x,S) = 1$.

9.2 Proof of Proposition 2

PROOF. The proofs are a consequence of Proposition 1. (i) Since $0 \leq WCC_v(x,S) \leq 1$ for all $x \in S$, then $0 \leq WCC_s(S) \leq 1$.

(ii) $WCC_s(S) = 0$ implies that for all $x \in S$ $WCC_v(x,S) = 0$. Since the condition for $WCC_v(x,S) = 0$ is that $t(x,S) = 0$, then $WCC_s(S) = 0$ implies that S has no triangles.

(iii) $WCC_s(S) = 1$ implies that $WCC_v(x,S) = 1$ for all $x \in S$. This implies that a vertex $x \in S$ such that $vt(x,V) \neq vt(x,S)$ and $vt(x,S) = |S \setminus \{x\}|$ does not exist. Thus, all the vertices $x \in S$ form triangles only and with exactly all the vertices in S, which implies having an edge with all the vertices in S, and hence forming a clique.

9.3 Proof of Theorem 1

PROOF. Let $S = S_1 \cup S_2$. For $x \in S_i, i \in \{1,2\}$ we have $t(x,S_i) = t(x,S)$, $vt(x, V \setminus S_i) = vt(x, V \setminus S)$ and $|S_i \setminus \{x\}| < |S \setminus \{x\}|$. Then,

$$\begin{aligned} WCC_v(x,S) &= \frac{t(x,S)}{t(x,V)} \cdot \frac{vt(x,V)}{vt(x,V) + |S \setminus \{x\}| - vt(x,S)} \\ &< \frac{t(x,S_i)}{t(x,V)} \cdot \frac{vt(x,V)}{vt(x,V) + |S_i \setminus \{x\}| - vt(x,S_i)} = WCC_v(x,S_i). \end{aligned}$$

Therefore,

$$\begin{aligned} |S| \cdot WCC(\{S_1,S_2\}) &= |S_1| \cdot WCC_s(S_1) + |S_2| \cdot WCC_s(S_2) \\ &= \sum_{x \in S_1} WCC_v(x,S_1) + \sum_{x \in S_2} WCC_v(x,S_2) \\ &> \sum_{x \in S} WCC_v(x,S) \end{aligned}$$

implies

$$WCC(\{S_1,S_2\}) > \frac{1}{|S|} \sum_{x \in S} WCC_v(x,S) = WCC_s(S) = WCC_s(S_1 \cup S_2).$$

9.4 Proof of Theorem 2

PROOF. (i) For the $r-1$ vertices $x \in K_r \setminus \{t\}$, we have $WCC_v(x,V) = vt(x,V)/(n-1) = (r-1)/(n-1)$. For the vertex t, we have $WCC_v(v,V) = 1$. Finally, for the $s-1$ vertices $x \in K_s \setminus \{t\}$, we have $WCC_v(x,V) = (s-1)/(n-1)$. As $n-1 = r+s-2$, we obtain the formula (3.4).

(ii) For the $r-1$ vertices $x \in K_r$ we have $WCC_v(x,K_r) = 1$. For the vertex t, we have

$$\begin{aligned} WCC_v(x,K_r) =& \frac{\binom{r-1}{2}}{\binom{r-1}{2}+\binom{s-1}{2}} \cdot \frac{n-1}{r-1+s-1} \\ =& \frac{(r-1)(r-2)}{(r-1)(r-2)+(s-1)(s-2)}. \end{aligned}$$

For the $s-1$ vertices $x \in K_s \setminus \{t\}$, we have

$$WCC_v(x, K_s \setminus \{t\}) = \frac{\binom{s-2}{2}}{\binom{s-1}{2}} \cdot \frac{s-1}{s-1} = \frac{(s-2)(s-3)}{(s-1)(s-2)}.$$

This gives the formula (3.6).

(iii) For $x \in K_r \setminus \{t\}$,

$$WCC_v(x, K_r \setminus \{t\}) = \frac{\binom{r-2}{2}}{\binom{r-1}{2}} \cdot \frac{r-1}{r-1} = \frac{(r-2)(r-3)}{(r-1)(r-2)};$$

for vertex t,

$$\begin{aligned} WCC_v(x,\{t\}) =& \frac{\binom{0}{2}}{\binom{n-1}{2}} \cdot \frac{n-1}{0+r-1+s-1} \\ =& \frac{(1-1)(1-2)}{(r+s-2)(r+s-3)} = 0; \end{aligned}$$

for $x \in K_s \setminus \{t\}$,

$$WCC_v(x, K_s \setminus \{t\}) = \frac{(s-2)(s-3)}{(s-1)(s-2)};$$

This implies (3.7).

(iv) Define $f_1(r,s) = n \cdot WCC(\mathcal{P}_1)$, $f_2(r,s) = n \cdot WCC(\mathcal{P}_2)$, and $f_3(r,s) = n \cdot WCC(\mathcal{P}_3)$. The expression of these functions are those in (3.4), (3.6) and (3.7), respectively. The goal is to show that for all integers values r, s with $r \geq s \geq 4$ the inequality $f_3(r,s) \leq f_2(r,s)$ holds. Clearly, the first summand of $f_3(r,s)$ is smaller than the first summand of $f_2(r,s)$, and the last summands are equal. As $f_2(r,s)$ has the second summand ≥ 0, we have $f_3(r,s) \leq f_2(r,s)$.

(v) We shall prove $f_2(r,s) - f_1(r,s) \geq 0$ for $n \geq 7$ and $4 \leq r \leq n-3$. We have $s = n - r + 1$ and

$$\begin{aligned} f_2(r,s) - f_1(r,s) >& n - 4 - \frac{(r-1)^2 + (n-r)^2}{n-1} \\ =& \frac{-2r^2 + (2+2n)r - 5n + 3}{n-1}. \end{aligned}$$

The sign of $f_2(r,s) - f_1(r,s)$ is the sign of the polynomial function $-2r^2 + 2(n+1)r - 5n + 3$, which is a convex function on r with roots:

$$\begin{aligned} r_1 =& \frac{1}{2}(n+1-\sqrt{n^2-8n+7}); \\ r_2 =& \frac{1}{2}(n+1+\sqrt{n^2-8n+7}). \end{aligned}$$

Now, for $n \geq 7$, we have $r_1 \leq 4$ and $r_2 \geq n-3$. Therefore, for each $r \in \{4, \ldots, n-3\}$ we have $f_2(r,s) - f_1(r,s) \geq 0$.

9.5 Proof of Theorem 3

Proof. Let N be the set of neighbors of v.

(i) For $x \in V$, we have $WCC_v(x,V) = vt(x,V)/r$. Now,

$$vt(x,V) = \begin{cases} (r-1)p & \text{if } x \in V \setminus N; \\ (r-1)p+1 & \text{if } x \in N; \\ d & \text{if } x \in \{v\}. \end{cases}$$

Then

$$\begin{aligned}(r+1)WCC(\mathcal{P}_1) =& (r-d)\frac{(r-1)p}{r} + d\frac{(r-1)p+1}{r} + \frac{d}{r}\\ =& (r-1)p + 2\frac{d}{r}.\end{aligned}$$

(ii) For $x \in V \setminus N$,

$$WCC_v(x,V) = \frac{vt(x,V)}{r-1} = \frac{(r-1)p}{r-1} = p.$$

(iii) For $x \in N$, we have

$$\begin{aligned} t(x,V) &= \binom{r-1}{2}p^3;\\ t(x,V\cup\{v\}) &= \binom{r-1}{2}p^3 + (d-1)p;\\ vt(x,V\cup\{v\}) &= (r-1)p+1;\\ |V\setminus\{x\}| - vt(x,V) &= (r-1)-(r-1)p;\end{aligned}$$

$$WCC_v(x,V) = \frac{((r-1)p+1)(r-1)(r-2)p^2}{((r-1)(r-2)p^2+2(d-1))\cdot r}.$$

Moreover, $WCC_v(v,\{v\}) = 0$. Then,

$$(r+1)WCC(\mathcal{P}_2) = (r-d)p + \frac{d}{r}\cdot\frac{((r-1)p+1)(r-1)(r-2)p^2}{(r-1)(r-2)p^2+2(d-1)}.$$

(iv) We have,

$$\begin{aligned}(r+1)\left(WCC(\mathcal{P}_1) - WCC(\mathcal{P}_2)\right) =& p(d-1)\\ &+ 2\frac{d}{r} - \frac{d}{r}\frac{((r-1)p+1)(r-1)(r-2)p^2}{(r-1)(r-2)p^2+2(d-1)},\end{aligned}$$

and the condition $WCC(\mathcal{P}_1) - WCC(\mathcal{P}_2) > 0$ is equivalent to the condition

$$ad^2 + bd + c > 0, \tag{9.1}$$

where

$$\begin{aligned} a =& 2(2+pr), \\ b =& p^2(p+1)r^2 - p(3p^2+3p+4)r + 2p^3 + 2p^2 - 4, \\ c =& -p^3r^3 + 3p^3r^2 + 2p(1-p^2)r. \end{aligned}$$

For short, let we denote by $O(r^n)$ a polynomial expression of degree at most n. Then, the greatest solution of (9.1) is,

$$d_2 = \frac{-p^2(1+p)r^2 + O(r) + \sqrt{p^4(p^2+2p+9)r^4 + O(r^3)}}{4(2+pr)}$$

and we get

$$\begin{aligned} \lim_{r\to+\infty} \frac{d_2}{r} &= \frac{-p^2(1+p) + p^2\sqrt{p^2+2p+9}}{4p} \\ &= p\frac{\sqrt{p^2+2p+9} - (1+p)}{4}. \end{aligned}$$

Thus, for a large enough r, the condition

$$d > rp\left(\sqrt{p^2+2p+9} - (1+p)\right)/4,$$

is equivalent to $WCC(\mathcal{P}_1) > WCC(\mathcal{P}_2)$.

Note that function $p \mapsto p\left(\sqrt{p^2+2p+9} - (1+p)\right)/4$ is increasing in p. A large value of p means more edges in G, and then a large value of d/r is needed for $WCC(\mathcal{P}_1)$ being greater than $WCC(\mathcal{P}_2)$.

In the case of Corollary 1, $p = 1$, thus $d > \sqrt{3} - 1/2 = 0.37$.

9.6 Proof of Theorem 4

PROOF. Let x be any vertex of the graph $G(V, E)$, and S the community of vertex x. Let's assume that all the edges of the graph close at least one triangle.

(i) For $\mathcal{P}_1$

$$\begin{aligned}
t(x,S) &= \binom{\frac{n}{2}-1}{2} p_{in}^3; \\
t(x,V) &= \binom{\frac{n}{2}-1}{2} p_{in}^3 + \binom{\frac{n}{2}}{2} p_{in} \cdot p_{out}^2 + \binom{\frac{n}{2}-1}{1}\binom{\frac{n}{2}}{1} p_{in} \cdot p_{out}^2; \\
vt(x,V) &= (\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out}; \\
vt(x,V) + |S \setminus \{x\}| - vt(x,S) &= (\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out}\frac{n}{2} - 1 - (\frac{n}{2}-1)p_{in}.
\end{aligned}$$

Then,

$$WCC(\mathcal{P}_1) = \frac{(\frac{n}{2}-1)(\frac{n}{2}-2)p_{in}^3((\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out})}{((\frac{n}{2}-1)(\frac{n}{2}-2)p_{in}^3 + (\frac{n}{2}-1)n \cdot p_{in} \cdot p_{out}^2)(\frac{n}{2}p_{out} + \frac{n}{2} - 1)}.$$

(ii) For $\mathcal{P}_2$

$$\begin{aligned}
t(x,S) &= \binom{\frac{n}{2}-1}{2} p_{in}^3 + \binom{\frac{n}{2}}{2} p_{in} \cdot p_{out}^2 + \binom{\frac{n}{2}-1}{1}\binom{\frac{n}{2}}{1} p_{in} \cdot p_{out}^2; \\
t(x,V) &= \binom{\frac{n}{2}-1}{2} p_{in}^3 + \binom{\frac{n}{2}}{2} p_{in} \cdot p_{out}^2 + \binom{\frac{n}{2}-1}{1}\binom{\frac{n}{2}}{1} p_{in} \cdot p_{out}^2; \\
vt(x,V) &= (\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out}; \\
vt(x,V) + |S \setminus \{x\}| - vt(x,S) &= n-1.
\end{aligned}$$

Then,

$$WCC(\mathcal{P}_2) = \frac{(\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out}}{n-1}.$$

(iii) We numerically proof this statement. First, we need to compute the WCC of $\mathcal{P}_s$, which consist of those partitions where s vertices of each of the two communities have been correctly placed. More formally, let $\mathcal{P}_s = \{A', B'\}$ be any partition of the graph with two communities A' and B' of size $\frac{n}{2}$, where $|A \cap A'| = s$ and $|B \cap B'| = s$. Let $x_a \in \{A \cap A'\}$, $x_b \in \{B \cap B'\}$, $v_a \in \{A' \setminus A\}$ and $v_b \in \{B' \setminus B\}$. That is, any partition with two communities of size $\frac{n}{2}$

where $2s$ vertices have been well placed.

$$\begin{aligned}
t(x_a, A') &= \binom{s-1}{2} p_{in}^3 + \binom{s-1}{1}\binom{\frac{n}{2}-s}{1} p_{in} \cdot pout^2 \\
&+ \binom{\frac{n}{2}-s}{2} p_{in} \cdot pout^2; \\
t(x_b, B') &= t(x_a, A'); \\
t(x_a, V) &= \binom{\frac{n}{2}-1}{2} p_{in}^3 + \binom{\frac{n}{2}}{2} p_{in} \cdot p_{out}^2 \\
&+ \binom{\frac{n}{2}-1}{1}\binom{\frac{n}{2}}{1} p_{in} \cdot p_{out}^2; \\
t(x_b, V) &= t(x_a, V); \\
vt(x_a, V) &= (\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out}; \\
vt(x_b, V) &= vt(x_a, V); \\
vt(x_b, V) + |A' \setminus \{x_a\}| - vt(x_a, A') &= (\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out} \\
&+ (\frac{n}{2} - 1 - (s-1)p_{in} - (\frac{n}{2}-s)p_{out}); \\
vt(x_b, V) + |B' \setminus \{x_b\}| - vt(x_b, B') &= vt(x_b, V) + |A' \setminus \{x_a\}| - vt(x_a, A').
\end{aligned}$$

$$\begin{aligned}
t(v_a, A') &= \binom{\frac{n}{2}-s-1}{2} p_{in}^3 + \binom{\frac{n}{2}-s-1}{1}\binom{s}{1} p_{in} \cdot pout^2 \\
&+ \binom{s}{2} p_{in} \cdot pout^2; \\
t(v_b, B') &= t(v_a, A'); \\
t(v_a, V) &= \binom{\frac{n}{2}-1}{2} p_{in}^3 + \binom{\frac{n}{2}}{2} p_{in} \cdot p_{out}^2 \\
&+ \binom{\frac{n}{2}-1}{1}\binom{\frac{n}{2}}{1} p_{in} \cdot p_{out}^2; \\
t(v_b, V) &= t(v_a, V); \\
vt(v_a, V) &= (\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out}; \\
vt(v_b, V) &= vt(v_a, V); \\
vt(v_b, V) + |A' \setminus \{v_a\}| - vt(v_a, A') &= (\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out} + (\frac{n}{2} - 1 - (s-1)p_{in} - (\frac{n}{2}-s)p_{out}); \\
vt(v_b, V) + |B' \setminus \{v_b\}| - vt(v_b, B') &= vt(v_b, V) + |A' \setminus \{v_a\}| - vt(v_a, A').
\end{aligned}$$

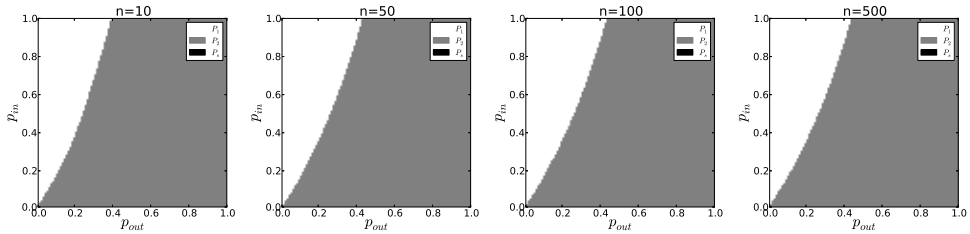


Figure 9.1: The best partition found for different configurations of p_{in}, p_{out} and n. All possible configurations of $\mathcal{P}_s$ have been tested

Then,

$$WCC(\mathcal{P}_s) = \frac{1}{n} \cdot 2 \cdot s \cdot \frac{(s-1)(s-2)p_{in}^3 + (s-1)(\frac{n}{2}-s)p_{in} \cdot p_{out}^2 + (\frac{n}{2}-s)(\frac{n}{2}-s-1)p_{out}^2 \cdot p_{in}}{((\frac{n}{2}-1)(\frac{n}{2}-2)p_{in}^3 + (\frac{n}{2}-1)n \cdot p_{in} \cdot p_{out}^2)} \cdot \frac{(\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out}}{(\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out} + (\frac{n}{2}-1-(s-1)p_{in} - (\frac{n}{2}-s)p_{out})}$$

$$+$$

$$\frac{1}{n} \cdot 2 \cdot (\frac{n}{2}-s) \frac{((\frac{n}{2}-s)-1)((\frac{n}{2}-s)-2)p_{in}^3 + ((\frac{n}{2}-s)-1)s \cdot p_{in} \cdot p_{out}^2 + s(s-1)p_{out}^2 \cdot p_{in}}{((\frac{n}{2}-1)(\frac{n}{2}-2)p_{in}^3 + (\frac{n}{2}-1)n \cdot p_{in} \cdot p_{out}^2)} \cdot \frac{(\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out}}{(\frac{n}{2}-1)p_{in} + \frac{n}{2}p_{out} + (\frac{n}{2}-1-(\frac{n}{2}-s-1)p_{in} - s \cdot p_{out})}.$$

Figure 9.1 shows, for each configuration of p_{in} and p_{out}, for different values of n, which partition $\mathcal{P} \in \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_s\}$ is that with a maximum WCC. In the case of $\mathcal{P}_s$, we tested for all possible values of s. We see that the statement is true, regardless of the the value of n.

(iv) We are interested in the transition point between $\mathcal{P}_1$ and $\mathcal{P}_2$, that is, we want to know for which values of p_{in} and p_{out}, $WCC(\mathcal{P}_1) - WCC(\mathcal{P}_2) = 0$. Since we are interested in arbitrarily large graphs, we compute the difference when n tends to infinitely large values :

$$\mathcal{L} = \lim_{n \to \infty} WCC(\mathcal{P}_1) - WCC(\mathcal{P}_2) = -\frac{1}{2} \frac{(p_{in}^2 \cdot p_{out} + 2 \cdot p_{out}^3 - p_{in}^2 + 2 \cdot p_{out}^2)(p_{in} + p_{out})}{(p_{out}+1)(p_{in}^2 + 2 \cdot p_{out}^2)}$$

If we solve $\mathcal{L} = 0$ for p_{in}, we obtain the following solutions:

$$-p_{out}, -\frac{\sqrt{(2-2\cdot p_{out})(p_{out}+1)}p_{out}}{1-p_{out}}, \frac{\sqrt{(2-2\cdot p_{out})(p_{out}+1)}p_{out}}{1-p_{out}}$$

, being the third solution the only positive and valid one.

9.7 Proof of Theorem 5

Proof.

$$\begin{aligned}
&WCC(P') - WCC(P) = \\
=&{}^1\!/\!_{|V|}\left(|C_1 \cup \{v\}| \cdot WCC(C_1 \cup \{v\}) + \sum_{i=2}^{k} |C_i| \cdot WCC(C_i)\right) - \\
&{}^1\!/\!_{|V|}\left(|C_1| \cdot WCC(C_1) + \sum_{i=2}^{k} |C_i| \cdot WCC(C_i) + WCC(\{v\})\right) \\
=&{}^1\!/\!_{|V|}(|C_1'| \cdot WCC(C_1')) - {}^1\!/\!_{|V|}(|C_1| \cdot WCC(C_1) + 0)
\end{aligned}$$

$$\begin{aligned}
=&{}^1\!/\!_{|V|}\left(\sum_{x\in C_1'} WCC(x, C_1') + \sum_{x\in C_1} WCC(x, C_1)\right) \\
=&{}^1\!/\!_{|V|}\left(\sum_{x\in C_1} WCC(x, C_1') + WCC(v, C_1') - \right. \\
&\left.\sum_{x\in C_1} WCC(x, C_1)\right)
\end{aligned}$$

9.8 Proof of Theorem 6

Proof. As stated in the theorem assumptions, the partition P' is build by removing v from C_1. Alternatively, the partition P can be build by removing vertex v to C_1' in P'. Then, the two following equalities hold:

$$\begin{aligned}
WCC(P) + WCC_R(v, C_1) &= WCC(P'), \\
WCC(P) &= WCC(P') + WCC_I(v, C_1') \\
\text{and thus: } WCC_R(v, C_1) &= -WCC_I(v, C_1')
\end{aligned}$$

9.9 Proof of Theorem 7

PROOF. Since WCC is a state function, all paths from P to P' have the same differential. Then, we express the transfer operation as a combination of remove and insert:

$$WCC(P) + WCC_T(v, C_1, C_k) = WCC(P')$$
$$WCC(P) + WCC_R(v, C_1) + WCC_I(v, C_k) = WCC(P')$$
$$WCC(P') - WCC(P) = -WCC_I(v, C_1') + WCC_I(v, C_k)$$

9.10 Proof of Theorem 8

PROOF. Consider the situation depicted in Figure 4.1. Let $N(x)$ be the set of neighbors of x. Given that, we define sets $F = N(v) \cap C$ which contains those vertices in C that are actual neighbors of v, and $G = (C \setminus N(x))$, which contains those vertices in C that are not neighbors of v. Therefore, from Theorem 5 we have:

$$\begin{aligned} WCC_I(v, C) = & \\ &= {}^1\!/_{|V|} \sum\nolimits_{x \in C} (WCC(x, C \cup \{v\}) - WCC(x, C)) + \\ &\quad {}^1\!/_{|V|} WCC(v, C \cup \{v\}) \\ &= {}^1\!/_{|V|} \sum\nolimits_{x \in F} (WCC(x, C \cup \{v\}) - WCC(x, C)) + \\ &\quad {}^1\!/_{|V|} \sum\nolimits_{x \in G} (WCC(x, C \cup \{v\}) - WCC(x, C)) + \\ &\quad {}^1\!/_{|V|} WCC(v, C \cup \{v\}) \end{aligned}$$

We know that $|F| = d_{in}$ and $|G| = r - d_{in}$, then we can define $WCC_I'(v, C)$ with respect to three variables Θ_1, Θ_2 and Θ_3, which represent the WCC improvement of a vertex of F, a vertex of G and v respectively. Then,

$$WCC_I'(v, C) = {}^1\!/_{|V|}(|F| \cdot \Theta_1 + |G| \cdot \Theta_2 + \Theta_3).$$

We define $q = (b - d_{in})/r$ as the number of edges connecting each vertex in C with the rest of the graph excluding v. Then,

(i) If $x \in F$, we have

$$\begin{aligned} t(x,C) =& (r-1)(r-2)\delta^3; \\ t(x,C\cup\{v\}) =& (r-1)(r-2)\delta^3 + (d_{in}-1)\delta; \\ t(x,V) =& (r-1)(r-2)\delta^3 + (d_{in}-1)\delta + q(r-1)\delta\omega + \\ & q(q-1)\omega + d_{out}\omega; \\ vt(x,V) =& (r-1)\delta + 1 + q; \end{aligned}$$

$$\begin{aligned} vt(x,V) + |C\cup\{v\}\setminus\{x\}| - vt(x,\{C\cup\{v\}\}) = r+q; \\ vt(x,V) + |C\setminus\{x\}| - vt(x,C) = r-1+q+1 = r+q; \end{aligned}$$

In $t(x,C)$, we account for those triangles that x closes with two other vertices in C. Similarly, in $t(x,C\cup\{v\})$ we account for those triangles that x closes with two other vertices in C, and those triangles that x closes with v and another vertex in C. $t(x,V)$ accounts for all triangles that vertex x closes with the graph, which are: $t(x,C\cup\{v\})$ plus those triangles that vertex x closes with another vertex of C and a vertex of $V\setminus C$, plus those triangles that vertex x closes with two other vertices in $V\setminus C$, plus those triangles vertex x closes with v and another vertex of $V\setminus C$. Since we assume that every edge in the graph closes at least one triangle, $vt(x,V)$ accounts for the number of vertices in C that are actual neighbors of x plus 1 (for vertex v) and q vertices that are connected to x. Finally, we have that the union of vertices in C and those vertices in V with whom x closes at least one triangle is $r+q$. Therefore,

$$\begin{aligned} \Theta_1 = & \quad WCC(x,C\cup\{v\}) - WCC(x,C) \\ = & \quad \frac{t(x,C\cup\{v\})}{t(x,V)} \cdot \frac{vt(x,V)}{|C\cup\{v\}\setminus\{x\}| + vt(x,V\setminus\{C\cup\{v\}\})} - \\ & \quad \frac{t(x,C)}{t(x,V)} \cdot \frac{vt(x,V)}{|C\setminus\{x\}| + vt(x,V\setminus C)} \\ = & \quad \frac{vt(x,V)}{(r+q)\cdot t(x,V)} \cdot (t(x,C\cup\{v\}) - t(x,C)) \\ = & \quad \frac{(r-1)\delta+1+q}{(r+q)\cdot((r-1)(r-2)\delta^3 + (d_{in}-1)\delta + q(r-1)\delta\omega + q(q-1)\omega + d_{out}\omega)} \cdot \\ & \quad (d_{in}-1)\delta. \end{aligned}$$

(ii) If $x \in B$, we have

$$\begin{aligned} t(x, C) =& (r-1)(r-2)\delta^3; \\ t(x, C \cup \{v\}) =& (r-1)(r-2)\delta^3; \\ t(x, V) =& (r-1)(r-2)\delta^3 + q(q-1)\omega + q(r-1)\delta\omega; \\ vt(x, V) =& (r-1)\delta + q; \end{aligned}$$

$$\begin{aligned} &vt(x, V) + |C \cup \{v\} \setminus \{x\}| - vt(x, \{C \cup \{v\}\}) = r + q; \\ &vt(x, V) + |C \setminus \{x\}| - vt(x, C) = r - 1 + q; \end{aligned}$$

$t(x, C)$ accounts for those triangles that x closes with two other vertices in C. Since, x is not connected to v, we have that $t(x, C) = t(x, C \cup \{v\})$. $t(x, V)$ accounts for the number of triangles that x closes with the rest of vertices in V. These are $t(x, C)$ plus those triangles that vertex x closes with another vertex of C and a vertex of $V \setminus C$, plus those triangles that vertex x closes with two other vertices in $V \setminus C$. $vt(x, V)$ accounts for the number of vertices in V with whom x closes at least one triangle, which are the neighbors of x in C and those t vertices with whom x is connected. Finally, we have that the union of vertices in $C \cup \{v\}$ and vertices in V with whom x closes at least one triangle is $r + q$, and the union of vertices in C and vertices in V with whom x closes at least one triangle is $r + q - 1$. Therefore,

$$\begin{aligned} \Theta_2 = & \quad WCC(x, C \cup \{v\}) - WCC(x, C) \\ = & \quad \frac{t(x,C\cup\{v\})}{t(x,V)} \cdot \frac{vt(x,V)}{|C\cup\{v\}\setminus\{x\}|+vt(x,V\setminus\{C\cup\{v\}\})} - \\ & \quad \frac{t(x,C)}{t(x,V)} \cdot \frac{vt(x,V)}{|C\setminus\{x\}|+vt(x,V\setminus C)} = \\ = & \quad -\frac{(r-1)(r-2)\delta^3}{(r-1)(r-2)\delta^3+q(q-1)\omega+q(r-1)\delta\omega} \cdot \frac{(r-1)\delta+q}{(r+q)(r-1+q)}. \end{aligned}$$

(iii) If $x = v$ we have

$$\begin{aligned} &t(x, C \cup \{v\}) = d_{in}(d_{in} - 1)\delta; \\ &t(x, V) = d_{in}(d_{in} - 1)\delta + d_{out}(d_{out} - 1)\omega + d_{out}d_{in}\omega; \\ &vt(x, V) = d_{in} + d_{out}; \\ &vt(x, V) + |C \setminus \{x\}| - vt(x, C) = r + d_{out}; \end{aligned}$$

In this case, $t(x, C \cup \{v\})$ accounts for those triangles that x closes with C, with whom it is connected to d_{in} vertices. $t(x, V)$ are those vertices vertex x closes with V, which are those x closes with C plus those x closes with other two vertices in $V \setminus C$. $vt(x, V)$ accounts for the number of vertices in V with whom x closes at least one triangle, which are d_{in} plus d_{out} since we assume that every edge closes at least one triangle. Finally, the union between the vertices in C and those vertices in V with whom x closes at least one triangle is $r + d_{out}$. Therefore,

$$\begin{aligned}
\Theta_3 = & \quad WCC(v, C \cup \{v\}) \\
= & \quad \frac{t(x,C\cup\{v\})}{t(x,V)} \cdot \frac{vt(x,V)}{|C|+vt(x,V\setminus C)} = \\
= & \quad \frac{d_{in}(d_{in}-1)\delta}{d_{in}(d_{in}-1)\delta+d_{out}(d_{out}-1)\omega+d_{out}d_{in}\omega} \cdot \frac{d_{in}+d_{out}}{r+d_{cout}}.
\end{aligned}$$

9.11 Distributions of Statistical Indicators

9.11.1 Amazon

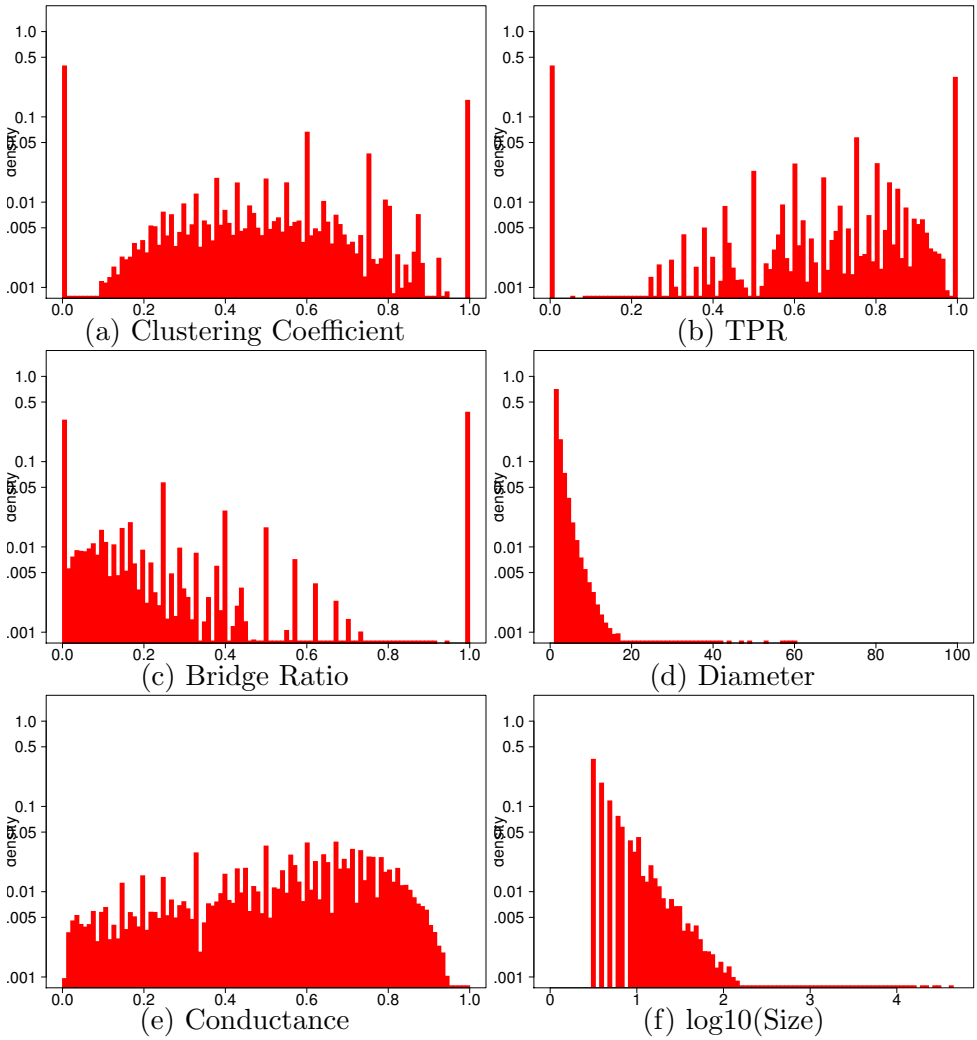


Figure 9.2: Distribution of the statistical indicators for the Amazon graph.

9.11.2 Dblp

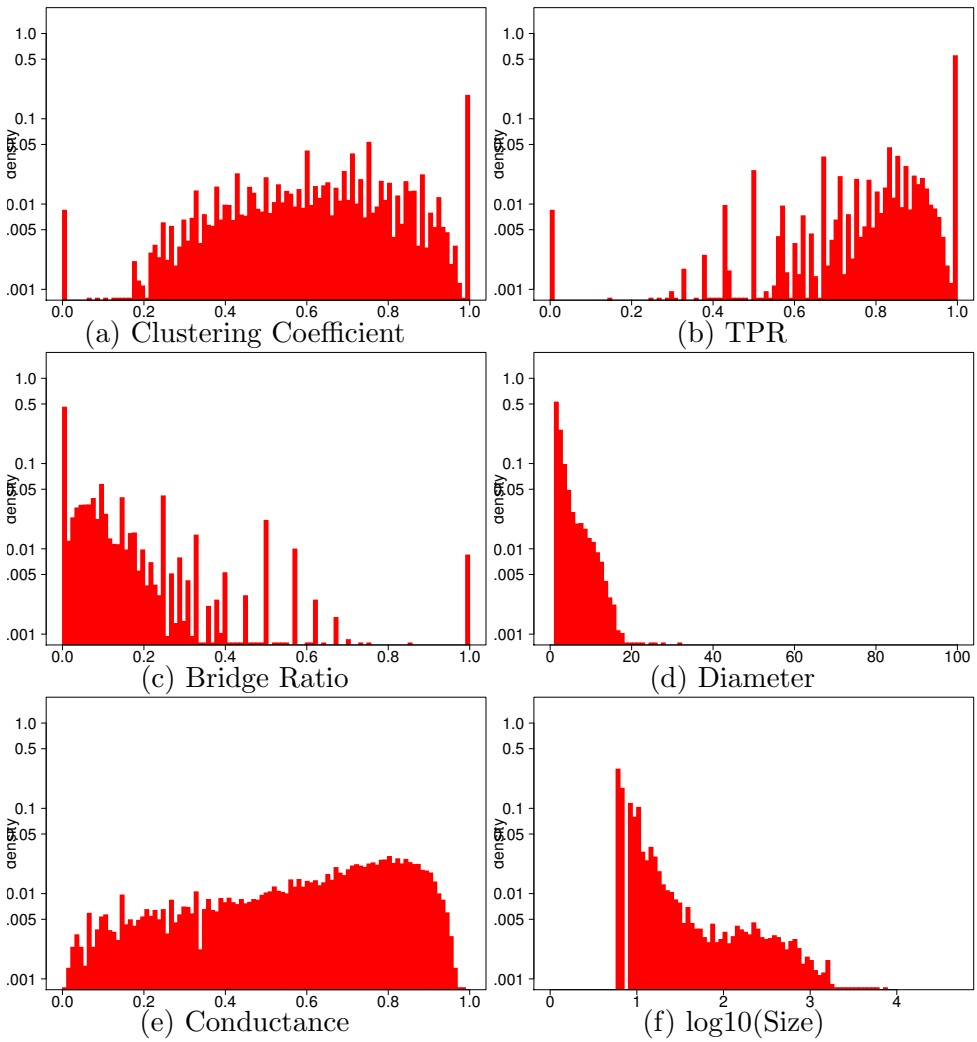


Figure 9.3: Distribution of the statistical indicators for the Dblp graph.

9.11.3 Youtube

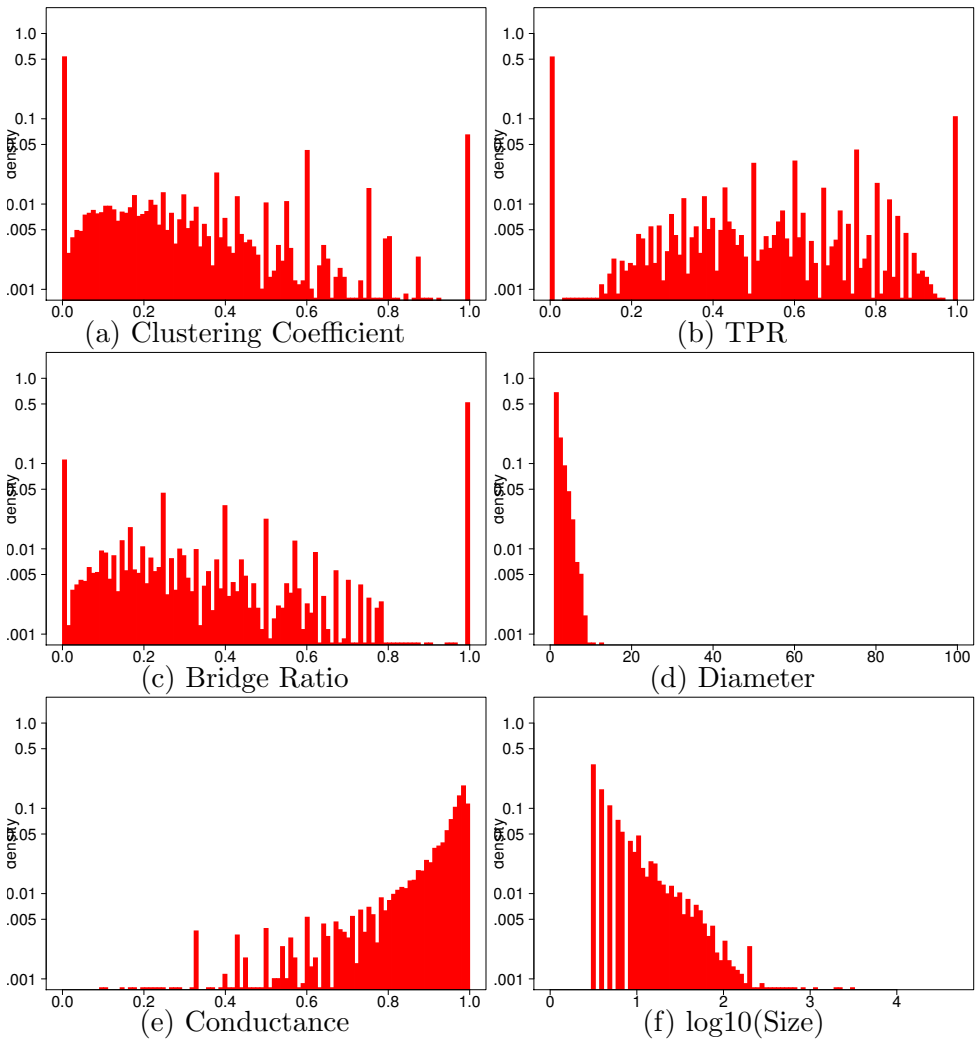


Figure 9.4: Distribution of the statistical indicators for the Youtube graph.

9.11.4 LFR 0.1

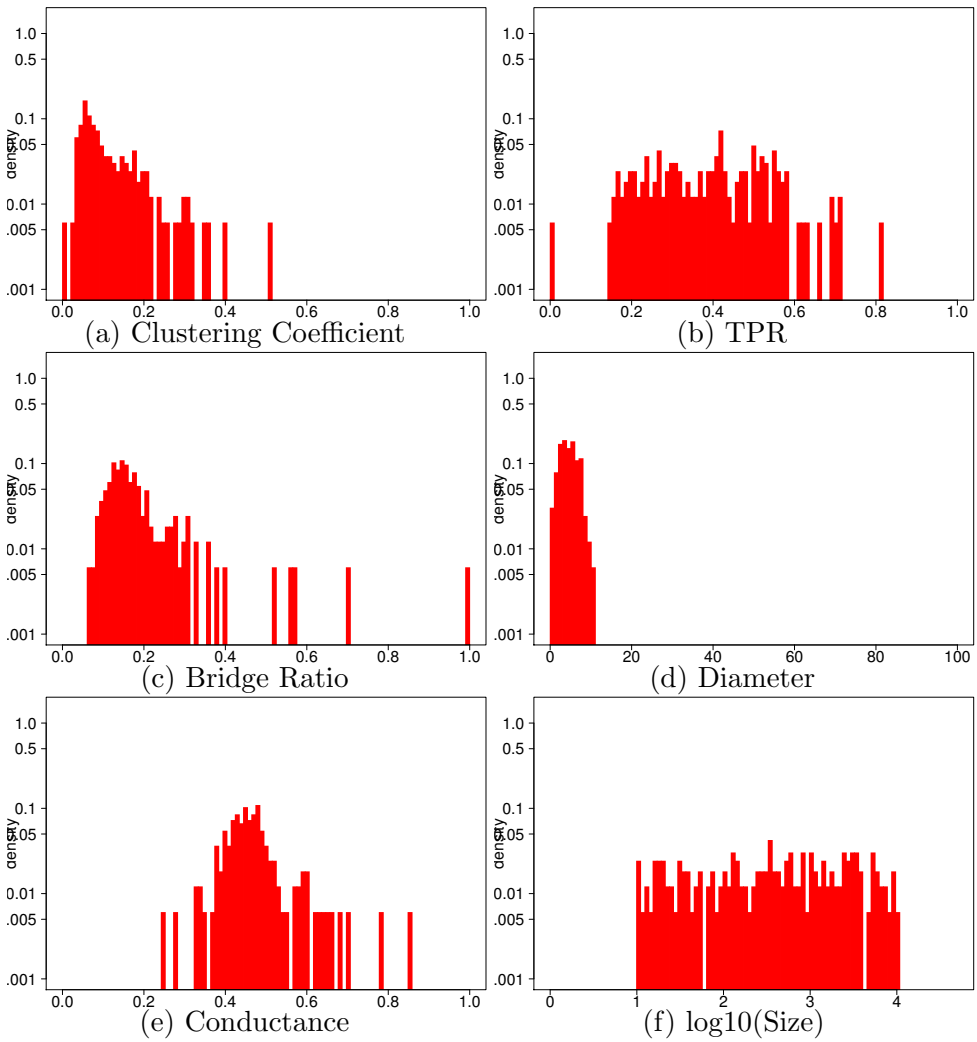


Figure 9.5: Distribution of the statistical indicators for the LFR1 graph.

9.11.5 LFR 0.2

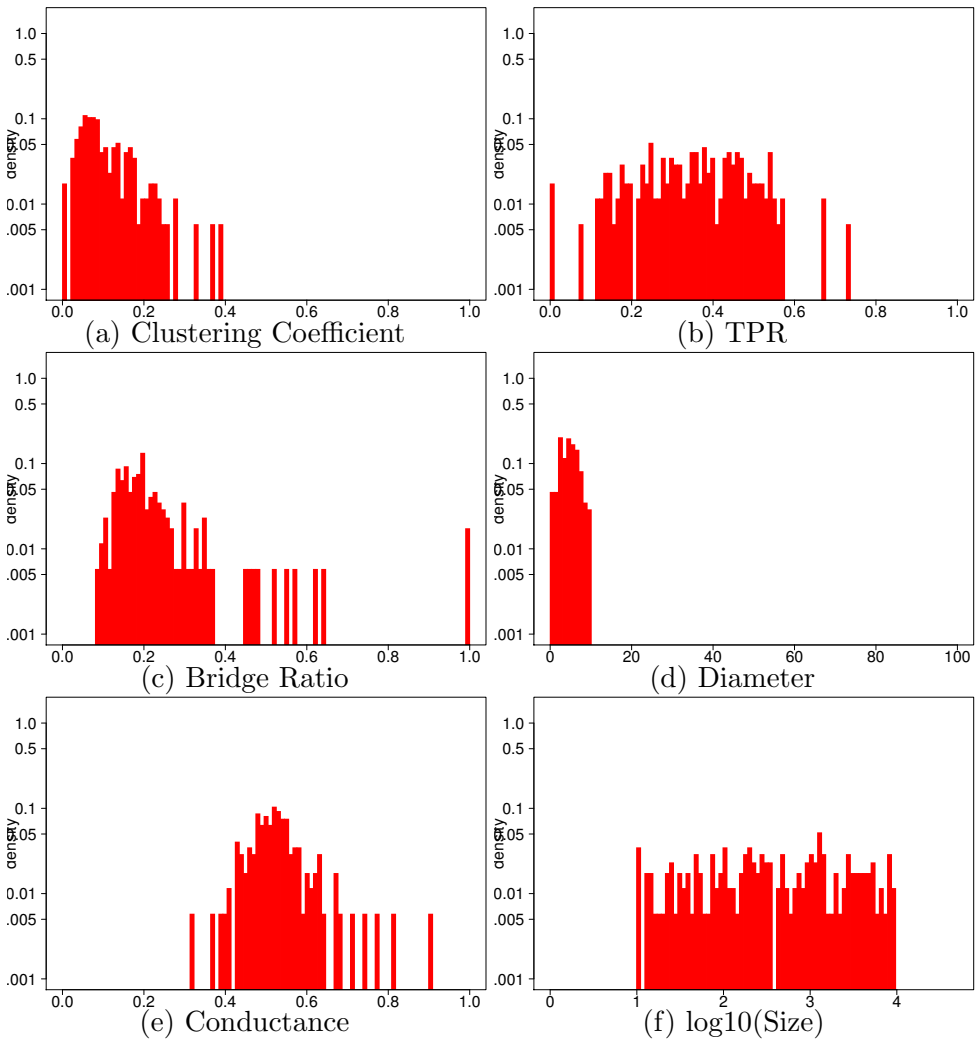


Figure 9.6: Distribution of the statistical indicators for the LFR2 graph.

9.11.6 LFR 0.4

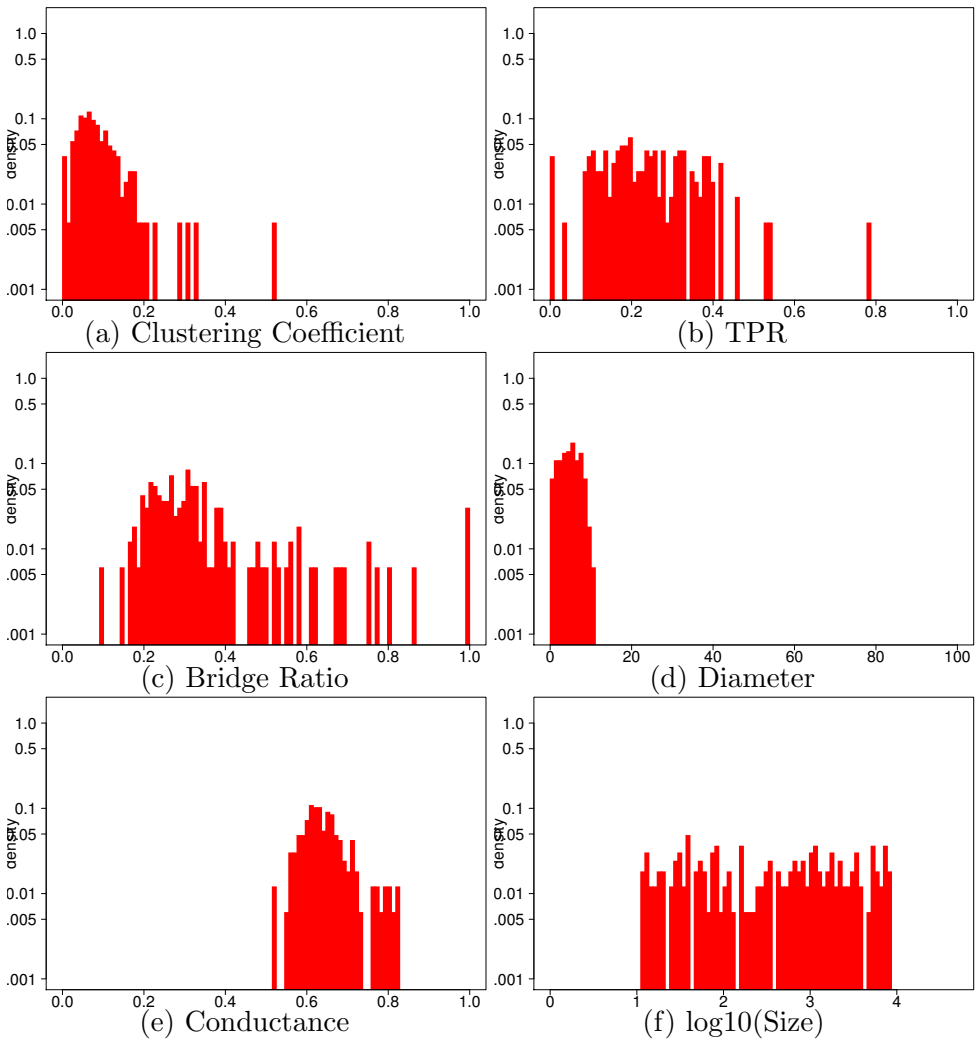


Figure 9.7: Distribution of the statistical indicators for the LFR4 graph.

9.11.7 LFR 0.5

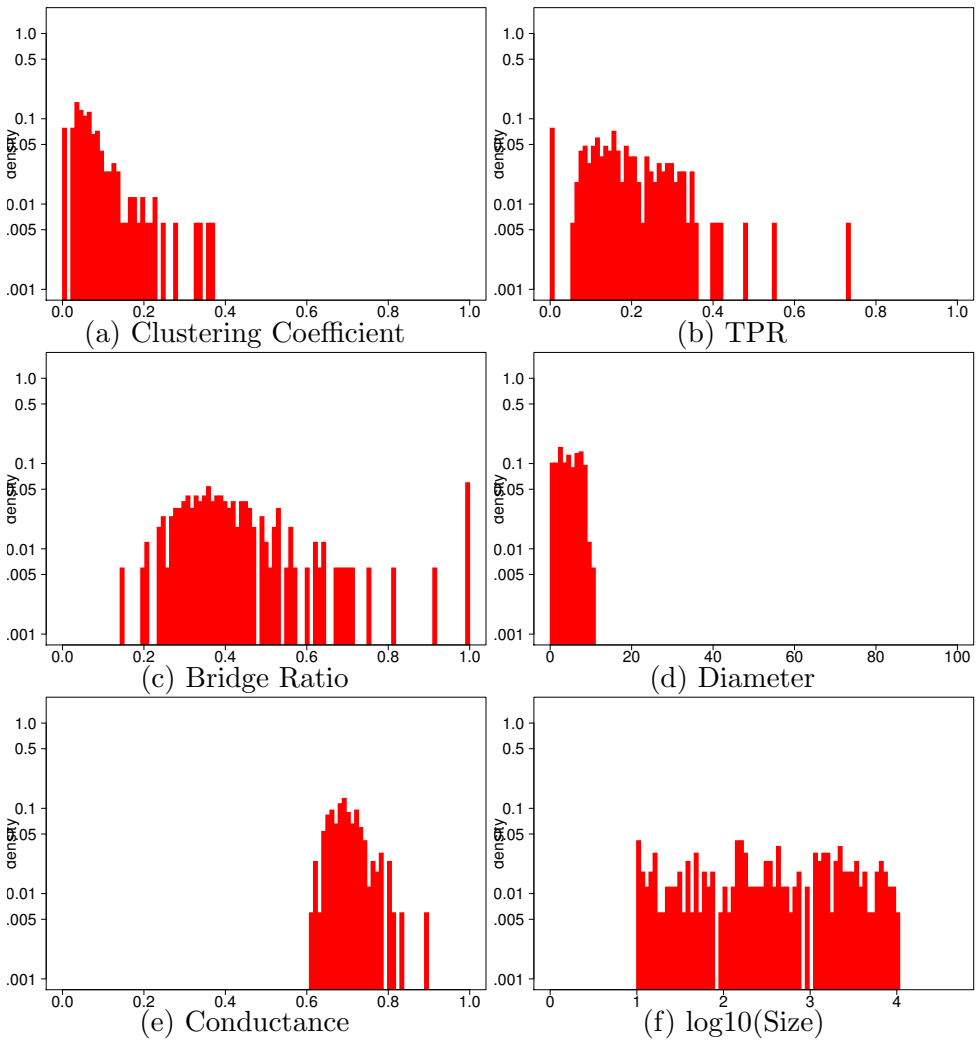


Figure 9.8: Distribution of the statistical indicators for the LFR5 graph.

Bibliography

[1] Balázs Adamcsek, Gergely Palla, Illés J Farkas, Imre Derényi, and Tamás Vicsek. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006. 18

[2] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010. 18, 57

[3] Ching Avery. Giraph: Large-scale graph processing infrastructure on hadoop. *Proceedings of the Hadoop Summit. Santa Clara*, 2011. 18

[4] Seung-Hee Bae, Dan Halperin, Jevin West, Martin Rosvall, and Brandon Howe. Scalable flow-based community detection for large-scale network analysis. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 303–310. IEEE, 2013. 19

[5] Seung-Hee Bae and Bill Howe. Gossipmap: a distributed community detection algorithm for billion-edge directed graphs. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 27. ACM, 2015. 20

[6] James P Bagrow. Communities and bottlenecks: Trees and treelike networks have high modularity. *Physical Review E*, 85(6):066118, 2012. 3, 15, 62

[7] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. 2, 13, 57

[8] Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: Web spam detection using the web topology. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 423–430. ACM, 2007. 2

[9] Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics*, 22(18):2283–2290, 2006. 2

[10] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Inference and phase transitions in the detection of modules in sparse networks. *Physical Review Letters*, 107(6):065701, 2011. 39

[11] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Physical review letters*, 94(16):160202, 2005. 18

[12] Emilio Di Giacomo, Walter Didimo, Luca Grilli, and Giuseppe Liotta. Graph visualization techniques for web clustering engines. *Visualization and Computer Graphics, IEEE Transactions on*, 13(2):294–304, 2007. 2, 10

[13] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical review E*, 72(2):027104, 2005. 2, 14

[14] Renzo et al. The linked data benchmark council: a graph and rdf industry benchmarking effort. In *To appear in SIGMOD Record*. ACM. 71, 72, 73

[15] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010. 11, 16, 43, 64

[16] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007. 1, 3, 9, 14

[17] Clara Granell, Sergio Gomez, and Alex Arenas. Hierarchical multiresolution method to overcome the resolution limit in complex networks. *International Journal of Bifurcation and Chaos*, 22(07):1250171, 2012. 14

[18] Mark Granovetter. The strength of weak ties: A network theory revisited. *Sociological theory*, 1(1):201–233, 1983. 2

[19] Roger Guimera, Marta Sales-Pardo, and Luís A Nunes Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2):025101, 2004. 2, 14

[20] John Hopcroft, Omar Khan, Brian Kulis, and Bart Selman. Natural communities in large linked networks. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 541–546. ACM, 2003. 10

[21] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004. 16

[22] George Karypis and Vipin Kumar. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995. 10

[23] A. Lancichinetti. Community detection algorithms: a comparative analysis. *Phy. Rev. E*, 80(5):056117, 2009. 57

[24] A. Lancichinetti, F. Radicchi, J.J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961, 2011. 57

[25] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008. 71, 87

[26] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Phy. Rev. E*, 78(4 Pt 2):6, 2008. 72, 73

[27] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *SIGKDD*, pages 462–470. ACM, 2008. 23

[28] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *ASIST*, 58(7):1019–1031, 2007. 24

[29] Hao Lu, Mahantesh Halappanavar, and Ananth Kalyanaraman. Parallel heuristics for scalable community detection. *Parallel Computing*, 2015. 19

[30] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for

large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010. 18

[31] Fragkiskos D Malliaros and Michalis Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics Reports*, 533(4):95–142, 2013. 1, 9

[32] T.G. Mattson, M. Riepen, T. Lehnig, P. Brett, W. Haas, P. Kennedy, J. Howard, S. Vangal, N. Borkar, G. Ruhl, and S Dighe. he 48-core SCC processor: the Programmer's view. In *SC*, pages 1–11. IEEE Computer Society, 2010. 6, 83

[33] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001. 5, 23

[34] Farnaz Moradi, Tomas Olovsson, and Philippas Tsigas. An evaluation of community detection algorithms on large-scale email traffic. In *Experimental Algorithms*, pages 283–294. Springer, 2012. 2

[35] Tamás Nepusz, Andrea Petróczi, László Négyessy, and Fülöp Bazsó. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E*, 77(1):016107, 2008. 11

[36] Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001. 23

[37] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006. 2, 13

[38] Mark EJ Newman. Spectral methods for community detection and graph partitioning. *Physical Review E*, 88(4):042822, 2013. 2, 14

[39] Mark EJ Newman and Juyong Park. Why social networks are different from other types of networks. *Physical Review E*, 68(3):036122, 2003. 23

[40] Günce Keziban Orman and Vincent Labatut. A comparison of community detection algorithms on artificial networks. In *Discovery Science*, pages 242–256. Springer, 2009. 72

[41] A. Padrol-Sureda, G. Perarnau-Llobet, J. Pfeifle, and V. Muntés-Mulero. Overlapping community search for social networks. In *ICDE*, pages 992–995, 2010. 19, 57

[42] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005. 57

[43] Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. Community detection in social media. *Data Mining and Knowledge Discovery*, 24(3):515–554, 2012. 1, 9

[44] Minh-Duc Pham, Peter Boncz, and Orri Erling. S3g2: A scalable structure-correlated social graph generator. In *TPCTC*, pages 156–172. Springer, 2012. 72

[45] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, 10(2):191–218, 2006. 3, 17

[46] A. Prat-Pérez, D. Dominguez-Sal, and J.L. Larriba-Pey. Social Based Layouts for the Increase of Locality in Graph Operations. In *DASFAA*, pages 558–569, 2011. 10

[47] Arnau Prat-Pérez and David Dominguez-Sal. How community-like is the structure of synthetically generated graphs? In *Proceedings of Workshop on GRAph Data management Experiences and Systems*, pages 1–9. ACM, 2014. 72

[48] Arnau Prat-Pérez, David Dominguez-Sal, Josep M Brunat, and Josep-Lluis Larriba-Pey. Put three and three together: triangle driven community detection. In *To be published in TKDD*. ACM. 21, 47, 58

[49] Arnau Prat-Pérez, David Dominguez-Sal, Josep M Brunat, and Josep-Lluis Larriba-Pey. Shaping communities out of triangles. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1677–1681. ACM, 2012. 21

[50] Arnau Prat-Pérez, David Dominguez-Sal, and Josep-LLuis Larriba-Pey. High quality, scalable and parallel community detection for large real

graphs. In *Proceedings of the 23rd international conference on World wide web*, pages 225–236. ACM, 2014. 47

[51] Arnau Prat-Pérez, David Dominguez-Sal, Josep-Lluis Larriba-Pey, and Pedro Trancoso. Producer-consumer: the programming model for future many-core processors. In *Architecture of Computing Systems–ARCS 2013*, pages 110–121. Springer, 2013. 83, 87

[52] Filippo Radicchi. A paradox in community detection. *EPL (Europhysics Letters)*, 106(3):38001, 2014. 15, 43

[53] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004. 15, 16

[54] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007. 3, 18, 57

[55] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006. 14

[56] Jason Riedy, David Bader, Henning Meyerhenke, et al. Scalable multi-threaded community detection in social networks. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1619–1628. IEEE, 2012. 19

[57] José F Rodrigues Jr, Hanghang Tong, Agma JM Traina, Christos Faloutsos, and Jure Leskovec. Gmine: a system for scalable, interactive graph visualization and mining. In *Proceedings of the 32nd international conference on Very large data bases*, pages 1195–1198. VLDB Endowment, 2006. 10

[58] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008. 3, 17, 57

[59] Marcel Salathé and James H Jones. Dynamics and control of diseases in networks with community structure. *PLoS Comput Biol*, 6(4):e1000736, 2010. 2

[60] V. Satuluri, S. Parthasarathy, and Y. Ruan. Local graph sparsification for scalable clustering. In *SIGMOD*, pages 721–732. ACM, 2011. 23

[61] X. Shi, L.A. Adamic, and M.J. Strauss. Networks of strong ties. *Physica A: Statistical Mechanics and its Applications*, 378(1):33–47, 2007. 23

[62] Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 939–948. ACM, 2010. 2

[63] Lei Tang and Huan Liu. Community detection and mining in social media. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2(1):1–137, 2010. 10

[64] Ju Xiang and Ke Hu. Limitation of multi-resolution methods in community detection. *Physica A: Statistical Mechanics and its Applications*, 391(20):4995–5003, 2012. 14, 26

[65] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, pages 745–754, 2012. 11, 15

[66] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013. 19

[67] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015. 17, 58

[68] Jaewon Yang, Julian McAuley, and Jure Leskovec. Detecting cohesive and 2-mode communities indirected and undirected networks. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 323–332. ACM, 2014. 11