UNIVERSITY OF CALIFORNIA,
IRVINE

Potentially near-optimal community discovery via stochastic graphlet sampling

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Computer Science

by

Pablo Martin Redondo

Thesis Committee:
Associate Professor Wayne B Hayes, Chair
Chancellor's Professor David A. Eppstein
Professor Emeritus Amelia C. Regan

2023

DEDICATION

This work is the culmination of two years of dedicated effort during the final step of my education, pursuing my Master's degree. When I made the decision three years ago to move to the United States for higher education, I could never have anticipated the incredible experiences that awaited me. Throughout this journey, my family has provided unwavering support, and I dedicate this achievement to my dad, my mum, my sister, and my grandma—for being the best family one could ever have.

Gracias familia, os quiero con locura.

To my partner, who has been my rock, my guiding light, my muse, and the source of my energy to keep pushing forward. Your patience, love, and unconditional support make me the luckiest person on Earth. You have borne my relentless pursuit of completion, which often required me to be away for extended periods. I am immensely grateful to have such an incredible woman in my life.

Спасибо, любимАЯ.

To my friends, who bring joy to my life and are there for me when no one else can be. I know I can always count on you, and have no doubt that you can count on me as well. A heartfelt thank you to all my brothers in Ramuras, LA PSOE, La Plaza, and everywhere else: Carlos, Guti, Ivan, Joan, Alberto, Moha, Mateo, Luca, Jose, Delafu, David, Sergio. You guys rock.

Esto hay que celebrarlo.

By combining our love, support, and shared moments, we have made this achievement possible. It is with profound gratitude and an overflowing heart that I dedicate this thesis to my beloved family, partner, and friends.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

Page

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I extend my heartfelt gratitude to the University of California, Irvine and the esteemed Donald Bren School of Information and Computer Science for providing me with this invaluable opportunity. Their unwavering support and belief in my potential have been instrumental in shaping my academic journey. I am also deeply appreciative of the job opportunities they offered, allowing me to serve as a Teaching Assistant, which significantly aided in funding my studies and lightening the financial burden along the way.

A special acknowledgment goes to Microsoft, a renowned leader in the industry, for granting me the privilege to work with their exceptional professionals during the summer and beyond my graduation. This invaluable experience will undoubtedly propel me into a new chapter of my life, surrounded by a talented and accomplished community. I eagerly anticipate the knowledge and growth that await me in this exciting endeavor.

Finally, I would like to express my sincere appreciation to Dr. Wayne B Hayes, a remarkable researcher and an extraordinary mentor. I have been truly fortunate to have him as my guide throughout this journey. Dr. Hayes's unparalleled dedication and unwavering commitment have been instrumental in bringing this work to fruition. His invaluable contributions, including insightful ideas, meticulous code assistance, and the generous investment of time, have been invaluable to me. I am profoundly grateful for his exceptional support and guidance.

VITA

Pablo Martin Redondo

EDUCATION

Master of Science in Computer Science — **2023**
University of California, Irvine — *Irvine, California*

Bachelor of Engineering in Telecommunication Technologies and Services — **2020**
Universidad Politécnica de Madrid — *Madrid, Spain*

RESEARCH EXPERIENCE

Undergraduate Research Assistant — **2018–2019**
Universidad Politécnica de Madrid — *Madrid, Spain*

TEACHING EXPERIENCE

Teaching Assistant — **2022–2023**
University of California, Irvine — *Irvine, California*

INDUSTRY EXPERIENCE

Software Engineer Intern — **2022**
Microsoft — *Redmond, Washington*

Software Consultant Developer — **2020–2021**
Guidewire Software — *Madrid, Spain*

REFEREED JOURNAL PUBLICATIONS

Managing gestational diabetes mellitus using a smartphone application with artificial intelligence (SineDie) during the COVID-19 pandemic: Much more than just telemedicine — **2020**
Diabetes Research and Clinical Practice

ABSTRACT OF THE THESIS

Potentially near-optimal community discovery via stochastic graphlet sampling

By

Pablo Martin Redondo

Master of Science in Computer Science

University of California, Irvine, 2023

Associate Professor Wayne B Hayes, Chair

Graph theory has extensive applications across various fields, such as social science, biology, and physics. One prominent graph-related task is the search for subgraphs possessing significantly higher edge densities than the mean; such subgraphs are referred to as communities. Community detection is an NP-hard problem, and existing algorithms addressing this problem are categorized between those that try to find a partition of non-overlapping communities, and those that find overlapping communities.

In this work, we reimagine the definition of community and propose a novel algorithm for overlapping community detection based on graphlet sampling using BLANT, a subgraph sampling method developed at UCI by the Wayne Hayes's Lab. By adopting a definition that emphasizes uniformly dense subgraphs and allows for edges extending beyond the community boundaries, our algorithm offers an extensive collection of dense overlapping communities. Furthermore, it introduces a community overlap graph, providing users with insights into the degree of overlap and empowering them to identify relevant communities based on their specific use cases. Our algorithm, BLANT-clusters, demonstrates the ability to discover larger and denser communities compared to existing state-of-the-art methods. We discuss the strengths and weaknesses of our algorithm and provide a comparative analysis with current approaches. Finally, we conclude by outlining potential future work and applications

for subsequent iterations of this method, highlighting the potential capability of achieving near-optimal community discovery.

Chapter 1

Introduction

In graph theory, a **network** (or graph) is a mathematical representation of entities and their relationships. A graph G consists of a set of vertices (or nodes) V, representing the entities, and a set of edges E, representing the relationships between them. Let $G(V, E)$ be a graph G with a set V of n vertices and set E of m edges.

Networks can be classified as either directed or undirected, depending on whether the relationships are one-way or both-ways. In a **directed graph**, each edge has a direction indicating the relationship between the vertices it connects. For instance: in a graph in which the nodes represent interested parties in a sale and edges represents sells, the edges will be directed from sellers to buyers. On the other hand, in an **undirected graph**, the edges do not have a direction and simply represent a connection between two vertices. An example is network representing friendships in a social network.

Additionally, networks can be weighted or unweighted. Entities in **weighted graphs** have relationships with different levels of strength. The edges in weighted graphs have a numerical value w associated with them. On the other hand, the edges in an **unweighted graph** are binary, either there is a connection (1) or there is not (0).

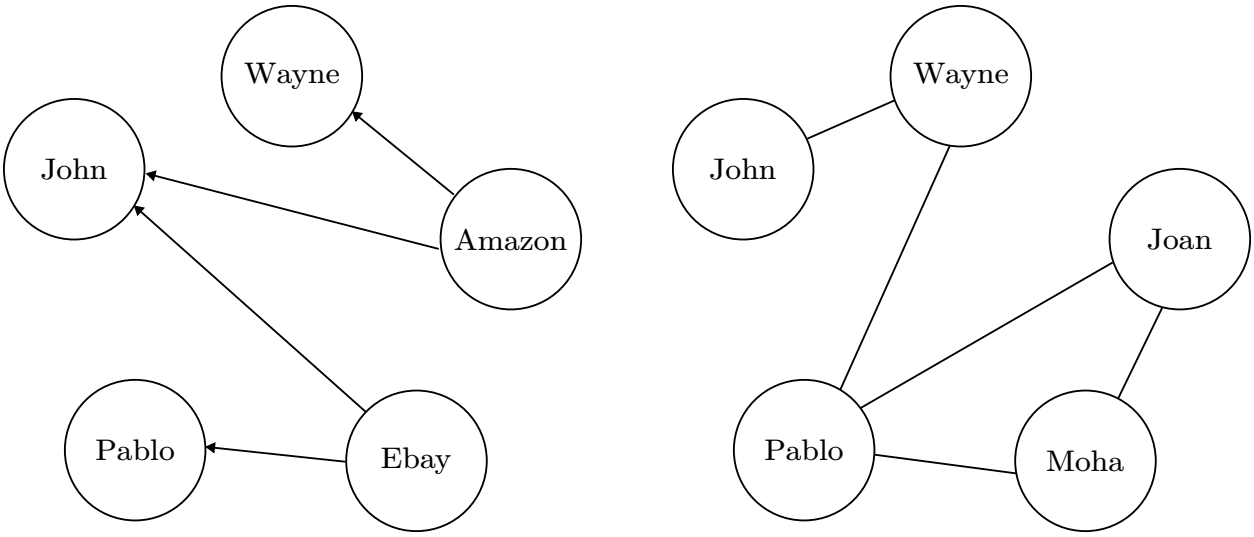


Figure 1.1: On the left, a directed graph of sales. On the right, an undirected graph of friendships.

In computer science, graphs are represented by **data structures**. The most common representations are the adjacency list L are the adjacency matrix A. L is a dictionary in which each key represents a node u and each value is a list of nodes that share an edge with u. If the graph is weighted, each member of the list is the pair of values (node, weight). A is a matrix of size $(n\ x\ n)$ - where n is the cardinality of V - in which a cell (u, v) represents a connection. If node u and node v share an edge, $A_{u,v} = A_{v,u} = 1$ in unweighted graphs and $A_{u,v} = A_{v,u} = w$ in weighted graphs.

Graphs can be used in a wide range of fields, including social science [1], biology [2], and physics[3]. For instance, in **social science**, graphs can be used to model and analyze social networks, where nodes represent people and edges represent their relationships. In **biology**, graphs could represent protein-protein interaction networks, where nodes represent proteins and two proteins are connected if they interact. In **physics**, graphs can be used to model the interactions between particles in a system. From a computer science approach, these networks are represented by a data structure and algorithms are developed in order to exploit them and obtain valuable information from it.

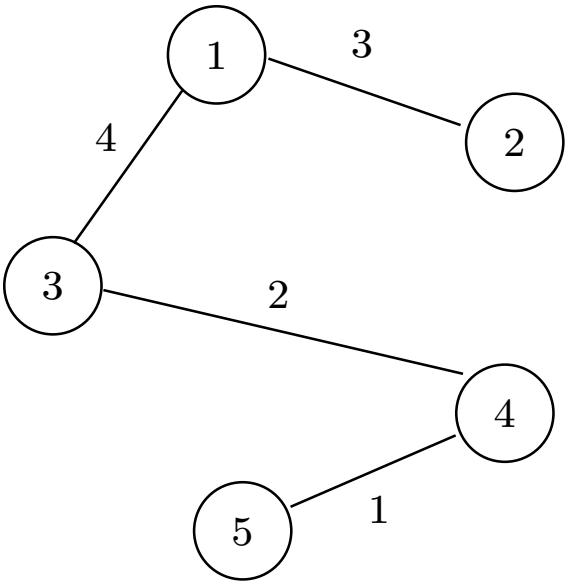


Figure 1.2: Weighted network representation.

In the example weighted network (1.2)

$$L = \{1 : [(2,3),(3,4)], 2 : [(1,3)], \\ 3 : [(1,4),(4,2)], 4 : [(3,2)], \\ 5 : [(4,1)]\}$$

and

$$A = \begin{pmatrix} 0 & 3 & 4 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Between the features that can be exploited, we have statistical and topological characteristics, including measures of connectivity, clustering, centrality or degree distribution. These techniques can provide insights into the structure and properties of the graph, as well as help to identify important vertices or induced **subgraphs**. An induced subgraph of G, $S(G)$, is a graph created using nodes and edges of G. Among all the graph study tasks, there is the search of subgraphs known as communities.

1.1 The community detection problem

A **community** is commonly defined in literature as a connected subgraph $G_c(V_c, E_c) \subseteq G(V, E)$ with n_c vertices and m_c edges, in which the number of connections within the nodes in the community is greater than the connections with nodes outside the community. Given the adjacency matrix A of a network, the most basic definition of a community implies that:

$$\sum_{\forall\, u,v\, \in V_c} A_{u,v} > \sum_{\forall\, u \in V_c, \forall\, v \notin V_c} A_{u,v} \tag{1.1}$$

The direct implication of this observation is that the likelihood of a node being connected to another node within the same community is higher compared to its likelihood of being connected to a node outside the community. It is important to note that in this work, we will not strictly adhere to this definition. The alternative definition will be presented and discussed in more detail in (1.3).

Communities in a graph can be categorized as either overlapping or non-overlapping. In the case of **non-overlapping communities**, each node is associated with at most one community. Conversely, in a graph with **overlapping communities**, nodes can belong to multiple communities simultaneously. While non-overlapping communities can be useful for certain applications, the following section (1.3) will discuss the realism of this community definition.

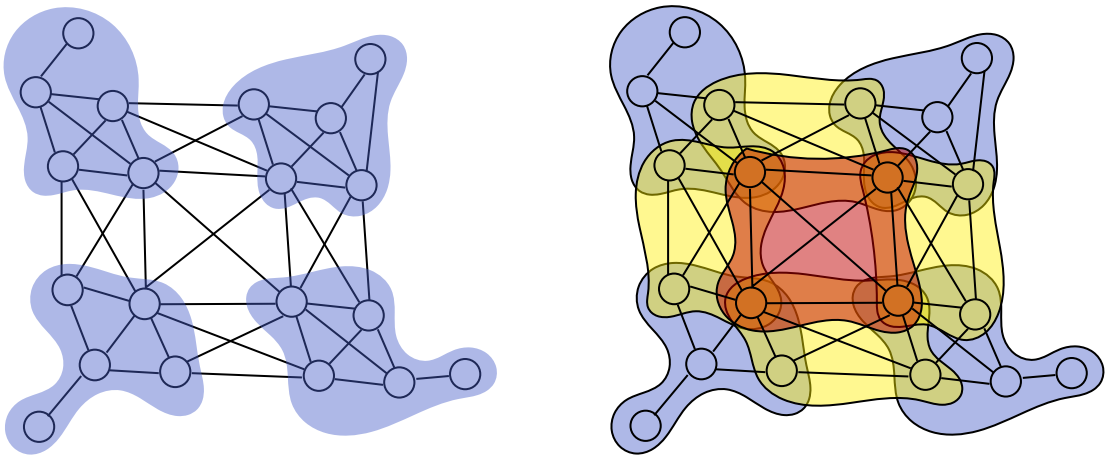


Figure 1.3: On the left, a separation of a graph in non-overlapping communities. On the right, a separation in overlapping communities.

The problem of community detection can be formulated as follows: given a network $G(V, E)$,

the objective is to identify and return a set of connected subgraphs that exhibit a community structure. It is worth noting that this task is not any easier than finding a maximum clique, which is already known to be NP-hard.

The concept of community is broad, encompassing various definitions and computational challenges. As a result, community detection has garnered significant attention in the literature, leading to a diverse body of research. Despite extensive study, community detection remains an open problem with active ongoing research.

1.2 Background

An open and unconstrained problem like community detection gives rise to a diverse range of solutions. One approach is to view **clustering** algorithms as non-overlapping community detection methods. However, even within clustering, there exist various techniques to obtain clusters. Some algorithms find clusters by identifying the minimum cut of a graph [4], while others employ spectral clustering algorithms [5] or graph partitioning algorithms [6]. Hierarchical clustering [7] discovers overlapping communities by initially identifying non-overlapping communities and subsequently aggregating them. Although each level in the hierarchy represents non-overlapping communities, the hierarchical structure can be interpreted as a division of overlapping communities.

Another category of non-overlapping community detection algorithms includes the **Girvan–Newman algorithm** [8] and its subsequent variations [9], [10]. This method utilizes betweenness centrality measures to identify and remove edges that bridge different communities. Other innovative approaches to non-overlapping community detection involve techniques such as **random walks** [11] and a variation of the Potts **spin glass** Hamiltonian [12].

However, the most extensively researched method for community detection is based on optimizing a **modularity** measure. Although the focus of this work is on overlapping community detection, it is essential to acknowledge the significance of this area of research. Modularity [13] quantifies the strength of the graph's division into communities, assuming that nodes belong to only one community. The measure evaluates the density of connections within a community and the sparsity of intercommunity connections.

Let $A_{u,v}$ denote the value of the adjacency matrix for the edge connecting vertex u to v, k_u represent the degree of node u (i.e., the number of edges it is connected to), K represent the number of communities, and m represent the number of edges.

The fraction of edges with one edge end lying in community c_i and other in community c_j is:

$$e_{ij} = \sum_{u \in c_i, v \in c_j} \frac{A_{u,v}}{2m} \tag{1.2}$$

The fraction of edge ends that are in vertices in c_i is:

$$a_i = \sum_{u \in c_i, v \in V} \frac{A_{u,v}}{2m} = \sum_{j=1}^{K} e_{ij} \tag{1.3}$$

With this defined, the modularity of a community partition of a graph is given by:

$$Q = \sum_{i=1}^{K} (e_{ii} - a_i^2) \tag{1.4}$$

The problem of finding the partition of $G(V, E)$ that maximizes the modularity Q is known to be NP-hard. As a result, the proposed algorithms for community detection rely on approximations and heuristics. The literature on these techniques is extensive and includes spectral optimization [14], simulated annealing [15], genetic algorithms [16], among others. However, the most widely used approach to maximize modularity is greedy optimization, which has been explored in works such as [17], [18], [19], and the renowned **Louvain** algorithm [20], known for its superior performance compared to other greedy algorithms.

Additionally, there exists a diverse range of methods for detecting overlapping communities.

These methods include label propagation fuzzy detection [21], **clique percolation** [22], local expansion and optimization [7], non-negative matrix factorization [23], statistical inference [24], and link partitioning [25]. Some of these methods, like label propagation and local expansion, rely on network diffusion principles to identify communities through the spread of information or influence within the network. Others, such as non-negative matrix factorization and statistical inference, employ mathematical and statistical approaches to identify communities based on structural and topological patterns in the network. These techniques are implemented in a variety of state-of-the-art algorithms, including Bigclam [26], CFinder [22], Copra [27] and Demon [28]. In the upcoming chapter (2), these algorithms will be compared and further described.

1.3 But, what is a community?

Let's address the elephant in the room. We have seen that the definition of community is very broad, but there is a bigger issue. The concept of single memberships is often too **restrictive** and does not adequately reflect the intricate relationships and associations nodes have in various contexts.

Consider a social graph G_s, where nodes represent people and edges represent connections. As an individual, I am part of multiple communities simultaneously. For example, I may belong to a complete clique representing my family, while also being a member of other cliques such as my friends in Madrid, my friends in Irvine, my roommates, and my soccer team. Moreover, as an MSc student in Computer Science at UC Irvine, I am connected to different communities, including my MSc friends, MSc classmates, students in my department, students in my school, and students at UC Irvine. These communities may have varying levels of edge density, but they all represent valid communities to which I belong.

The community of UC Irvine, despite having sparser connections, can still be highly resourceful and influential, sometimes even more so than my complete cliques. The number of edges connecting myself to these larger, sparser communities tends to be greater than the connections within my complete cliques.

This perspective extends beyond social networks. In biology, for instance, there are proteins that interact with multiple protein complexes [29]. Therefore, non-overlapping communities struggle to capture the diverse nature of real-world communities in most network contexts. Returning to the previous example, if we were to use a modularity-based algorithm, where would my node be assigned? The communities I mentioned would have many nodes extending beyond their boundaries, resulting in low modularity values. This limitation is known as the multiresolution limit of modularity [30]. Modularity, in this sense, becomes disconnected from reality. **Non-overlapping community detection algorithms fail to capture essential community information**, which creates a contradiction.

The notion of communities as non-overlapping likely originated as a simplification in early research on community detection. However, it is time to reconsider and redefine the term *community*. Dividing a graph into overlapping communities, even to the extent where some communities are fully contained within others, provides a more accurate representation of the complexity and interconnectedness of real-world communities.

Given a community c with n_c nodes and m_c edges, and the adjacency matrix A of the graph, the maximum number of possible edges is $\binom{n_c}{2}$. Then, its edge density is:

$$d_c = \frac{m_c}{\binom{n_c}{2}} = \frac{\sum\limits_{u,v \in c} A_{u,v}}{\binom{n_c}{2}} \tag{1.5}$$

This work presents a **re-imagined definition** of a community, considering it as a connected subgraph $G_c(V_c, E_c) \subseteq G(V, E)$ with $n_c \geq 3$ vertices and m_c edges, characterized by a reelatively high uniform edge density d_c. The term *uniform* refers to the property that any

subgraph within the community should not have a local edge density significantly lower than the mean edge density of the entire community. This definition includes the existing definitions as special cases, while simultaneously allowing significantly more flexibility. It serves as the foundation for BLANT-C[1], an algorithm specifically designed to identify subgraphs with an arbitrary user-defined d.

[1] At present, the algorithm used in this study does not possess a specific name. As a temporary placeholder, we can refer to it as BLANT-C (BLANT communities). However, suggestions for a suitable name are welcome and encouraged from the readers and researchers in the field.

Chapter 2

Overlapping community detection algorithms and quality measures

In this section, we will examine several prominent algorithms for community detection, including Bigclam, CFinder, Copra and Demon. These algorithms have been widely recognized as state-of-the-art and will serve as benchmarks for comparing BLANT-C.

To evaluate the algorithms' performance, various quality measures will be employed. These measures encompass overlapping modularity and a novel metric introduced within this thesis: edge density vs community size. Both overlapping modularity and edge density vs community size will be utilized to assess the algorithms' effectiveness in real network scenarios.

For benchmark LFR synthetic networks, the evaluation will employ the Normalized Mutual Information (NMI) as a quality measure. NMI quantifies the similarity between the detected community structure and the ground truth communities in the benchmark networks. It enables a comprehensive evaluation of the algorithms' ability to accurately identify overlapping communities in controlled synthetic environments.

While it would have been beneficial to compare algorithms based on diverse techniques, this study focuses on the current state-of-the-art approaches, resulting on having half of the algorithms be label propagation based. The selection of these algorithms ensures alignment with existing research and enables meaningful comparisons within the field of community detection.

2.1 Overlapping community detection algorithms

2.1.1 BigClam

BigClam [26], which stands for **Cluster Affiliation Model for Big Networks**, is a generative model designed for detecting overlapping communities. The algorithm employs a bipartite graph representation (nodes to communities) as a cluster affiliation model to capture the notion of overlapping communities. It combines this model with a variant of **non-negative matrix factorization** (NMF) to identify communities within the network.

The algorithm estimates the number of communities, denoted as K, and modifies the objective optimization function of NMF from the l_2 norm to log-likelihood. This adjustment enables scalability for big networks. In BigClam, a matrix F is used to represent the weighted bipartite graph, depicting the associations between each node and each community. The connections between vertices in the original graph are determined based on the weights of their affiliations to the communities. The probability of an edge between two vertices in the original graph is given by:

$$p(u, v) = 1 - e^{-\sum_K F_{uc} \cdot F_{vc}} = 1 - e^{-F_u \cdot F_v^T} \tag{2.1}$$

The probability of an edge increases as the number of communities shared by the vertices

increases. The matrix F is optimized using block coordinate gradient descent, taking the real network into account. Node membership within communities is determined based on the values of F_{uc}. Specifically, if $F_{uc} \geq \delta$ (where δ is a threshold), node u is considered a member of community c. The estimation of the optimal value for K is itself treated as an optimization problem within the BigClam algorithm.

2.1.2 CFinder

CFinder (**Community finder**) is a free software for detecting and drawing overlapping communities based on the **Clique Percolation Method** (CPM) [22]. This method operates by initially identifying all k-cliques (complete subgraphs) within the network, where k represents the number of nodes in the subgraph.

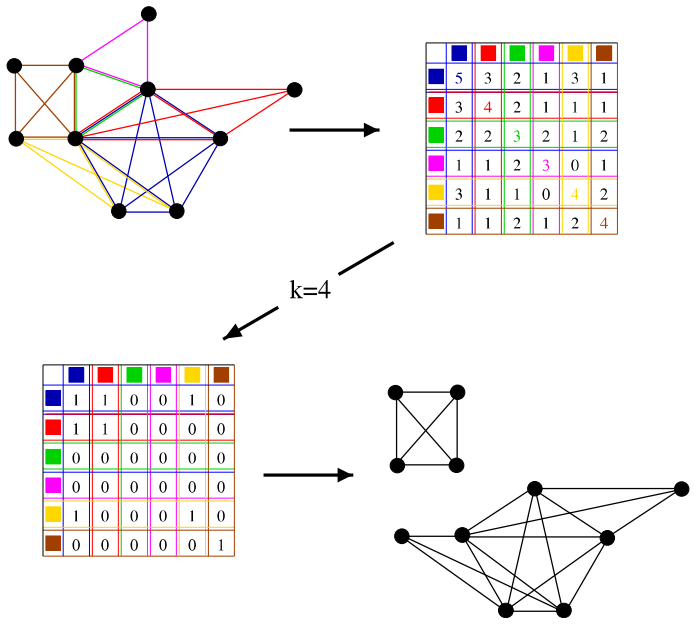


Figure 2.1: k-community detection example given the clique matrix in CPM.
Reproduced with permission from Springer Nature. [22]

The algorithm begins with the largest possible clique size, determined by the degree distribution, and subsequently reduces the clique size in each iteration. For each potential clique size, the algorithm samples the cliques by selecting an arbitrary node u, identifying the

cliques of the current size that include u, and removing the nodes and edges belonging to the discovered cliques from the graph. To mitigate the time complexity, complete subcliques of already found cliques are not considered. Nevertheless, the time complexity of this aspect of the algorithm remains exponential.

Subsequently, the algorithm constructs a matrix that depicts the cliques and the number of shared nodes among them. The communities are derived from this matrix. Clique cells on the diagonal with numbers greater than or equal to k, and greater than or equal to $k-1$ elsewhere in the matrix, represent cliques within a **k-community**.

2.1.3 Copra

The **Community Overlap Propagation Algorithm** (Copra) [27] extends the non-overlapping label propagation community detection algorithm [31] to enable the detection of overlapping communities. The original **label propagation** algorithm is notable for its near-linear complexity and can be summarized as follows:

Algorithm 1 Non-overlapping Label Propagation community detection algorithm

```
1: procedure LabelPropagation(G)
2:     Initialize every vertex v ∈ G with a unique label.
3:     while stopping condition is not reached do  ▷ Labels tend to converge to a solution
4:         for each vertex x ∈ G do
5:             N_x ← set of neighbors of x
6:             C_i ← count the number of times each label i appears in N_x
7:             l ← label with the maximum count in C
8:             if multiple labels have the same maximum count in C then
9:                 l ← randomly choose one of those labels
10:            end if
11:            Set the label of vertex x to l
12:        end for
13:    end while
14:    Create a community for each unique label in G
15:    Add all vertices with the same label to their respective communities
16: end procedure
```

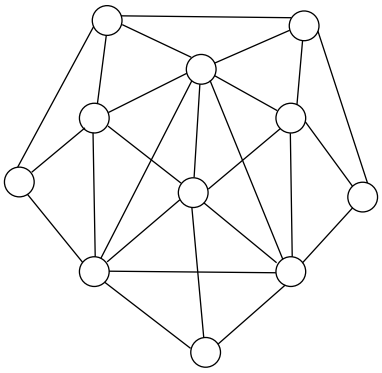
The **extension** introduced by Copra for overlapping community detection involves synchronous updates and the storage of a list of labels with corresponding node belonging coefficients.

These coefficients range from 0 to 1, and the sum of all coefficients for a given node must equal 1. If a coefficient falls below a threshold of $\frac{1}{\beta}$, it is removed from the node, and the remaining coefficients are updated to ensure they sum up to 1. If all coefficients are below the threshold, the largest coefficient is retained. In cases where all coefficients are equal, one is randomly chosen. The threshold value depends on β, which represents the maximum number of communities a node can belong to.

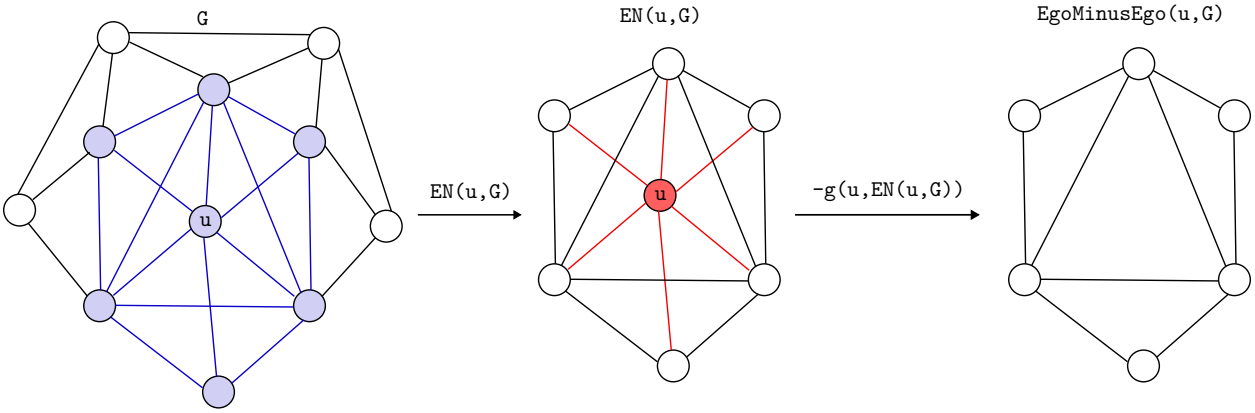
2.1.4 DEMON

The **Democratic Estimate of the Modular Organization of a Network** (DEMON) [28] is an overlapping community detection algorithm that also relies on label propagation. This variant of label propagation identifies local communities for each node and subsequently merges the maximal communities from all local communities to extract the global communities.

In DEMON, the $EgoMinusEgo$ subgraph is defined as $EgoMinusEgo(u, G) = -g(u, EN(u, G))$, where $EN(u, G)$ represents the subgraph obtained by extracting the ego network and $-g(u, G)$ denotes the graph-vertex difference operation. The ego network extraction operation $EN(u, G)$ involves constructing a graph consisting of node u, all its neighbors in G, and the connections among them. On the other hand, the graph-vertex difference $-g(u, G)$ operation entails removing node u and all edges connected to it from graph G.



(a) Example graph G



(b) Step by step $EgoMinusEgo(u, G)$ transformation

Figure 2.2: Example of a $EgoMinusEgo(u, G)$ subgraph in DEMON

To extract the local communities, DEMON performs **label propagation** on the $EgoMinusEgo(u, G)$ subgraph for each node u. Subsequently, node u is added to the communities identified through this label propagation step. Finally, the local communities of each node are merged if they exhibit sufficient similarity, determined by a threshold parameter ϵ, resulting in the extraction of global communities.

2.2 Edge Density and Overlapping Modularity

Benchmarking community detection algorithms poses a significant challenge due to several factors. Primarily, these algorithms are intended for real networks where the ground truth is often unavailable. Furthermore, the concept of community itself is broad and lacks a precise definition (1.3). Nonetheless, it is essential to employ quality measures to evaluate the results of BLANT-C against real networks and enable comparisons with state-of-the-art algorithms in our designated set of algorithms $\mathcal{A}$.

2.2.1 Overlapping modularity

The research community has shown significant interest in developing measures for quantifying overlapping communities. In line with these concerns, the concept of **overlapping modularity** [32] was introduced. This measure aims to capture some attributes of the detected communities that align with the ones of interest in this research, although it conflicts in others.

The first requirement of this measure is that a node within a community c should have more connections with nodes inside the community than outside of it[1]. This criterion can be expressed using the following equation:

$$\frac{\sum_{v \in c, v \neq u} A_{u,v} - \sum_{v \notin c} A_{u,v}}{k_u} \tag{2.2}$$

Here, $A_{u,v}$ represents the presence of an edge between nodes u and v, and k_u is the degree of node u. The second requirement is that communities should be dense, which is incorporated by including the edge density d_c in the equation.

[1]It has been argued that this requirement is unrealistic in (1.3). Nevertheless, this measure is commonly used and well-regarded in the research community. A formal definition of a measure that is considered to be an improvement will be proposed later.

Additionally, to prevent the occurrence of nearly identical overlapping communities being repeated, the number of communities a node is part of, denoted as s_u, is considered. The overlapping modularity of a community c is defined as:

$$M_c^{ov} = \frac{1}{n_c} \sum_{u \in c} \frac{\sum_{v \in c, v \neq u} A_{u,v} - \sum_{v \notin c} A_{u,v}}{k_u \cdot s_u} \cdot d_c \tag{2.3}$$

Finally, the overlapping modularity of each community is averaged over the K communities, yielding the overall overlapping modularity quality measure:

$$M^{ov} = \frac{1}{K} \sum_{c=1}^{K} \left[\frac{1}{n_c} \sum_{u \in c} \left(\frac{\sum_{v \in c, v \neq u} A_{u,v} - \sum_{v \notin c} A_{u,v}}{k_u \cdot s_u} \cdot d_c \right) \right] \tag{2.4}$$

It is important to note that according to the authors, every node must be assigned to at least one community to compute this measure. Nodes that do not belong to any community are typically added to a miscellaneous community. However, this approach has its limitations, as it may be the case that not every node is part of a community. Additionally, the miscellaneous community may be disconnected or have a low edge density, which deviates from the fundamental concept of a community. These considerations should be taken into account when interpreting and applying the overlapping modularity measure. In fact, for our experiments this requirement has been relaxed.

2.2.2 Edge density vs community size

Given the definition of community described in this work (1.3), a quality index to measure the **edge density vs the size of the communities** detected is required. For this matter, a score based on these parameters is proposed. The overlapping modularity measure is taken as a starting point. Nevertheless, the idea that a node in a community should have more edges going into the community than outwards it is against what was exposed in (1.3).

This is why, for this quality measure, what is taken into account is the edge density of the communities and the number of communities a node is part of.

Given the edge density d_c of community c defined in Equation (1.5), the number of nodes in a community n_c, the number of communities a node is part of s_u, and the number of communities K, let the edge density score $\mathcal{ED}_N$ of the communities detected by algorithm $\mathcal{A}_i \in \mathcal{A}$ in a real network be:

$$\mathcal{ED}_N(\mathcal{A}_i) = \frac{1}{K}\sum_{c=1}^{K}\left[\frac{1}{n_c}\cdot\sum_{\forall u\in c}\left(\frac{1}{s_u}\cdot d_c\right)\right] = \frac{1}{K}\sum_{c=1}^{K}\left[d_c\cdot\frac{1}{n_c}\cdot\sum_{\forall u\in c}\frac{1}{s_u}\right] \tag{2.5}$$

The score is an average of the score of each community. The score of each community is an average of the score that node has in that specific community. The score of a node directly depends on the density of the community - the higher the better - and inversely depends on the number of communities the node is member of - in order to avoid the repetitive communities that vary by a little number of nodes.

Note that if $s_u = 1$ for every member of the community c, $d_c \cdot \sum_{\forall u\in c}\frac{1}{s_u} = d_c \cdot n_c$, which represents a good compromise between taking into consideration the size of the communities and avoiding repetitive communities.

The measure under consideration ranges from a minimum value of 0 to a maximum value of 1. In addition to evaluating the measure, a direct comparison of the results based on community size versus edge density will be conducted to provide a more unfiltered and comprehensive assessment of the solution space.

This raw comparison will allow for a deeper understanding and analysis of the performance and characteristics of the different algorithms. By considering both the measure and the size-edge density comparison, a more nuanced and informative evaluation of the algorithms can be achieved.

2.3 LFR benchmark graphs and quality measures

The complexity of measuring the quality of a set of algorithms $\mathcal{A}$ in real networks arises from the inherent challenge of not knowing the ground truth communities. To overcome this limitation, researchers have sought to create synthetic networks that replicate real networks but possess known community divisions. In [33], the authors propose an algorithm for generating synthetic networks with non-overlapping community divisions.

Fortunately, the following year, the same authors extended this work to include overlapping communities [34]. These synthetic networks are commonly referred to as **LFR benchmark graphs**. The LFR benchmark graphs serve as valuable tools for evaluating and comparing the performance of algorithms in community detection tasks.

2.3.1 LFR benchmark graphs

It has been observed that the degree distribution of real networks can be approximated by a power-law distribution[2] [36]. Synthetic networks based on this principle are referred to as **scale-free networks**. In these networks, the fraction of nodes with degree k, denoted as $P(k)$, follows an approximate power-law distribution:

$$P(k) \approx k^{-\tau} \tag{2.6}$$

Furthermore, studies have shown that the size distribution of communities in real networks can also exhibit a similar power-law distribution [22]. This is why in the LFR benchmark graphs for both non-overlapping communities [33] and overlapping communities [34], the authors chose to generate networks with degree distribution and community size distribution that follow **power-law distributions**.

[2] Althought this model has recently been challenged, specifically for social networks. [35]

In this work, the official implementation algorithm from [34] is utilized. This algorithm generates a graph with an overlapping community division, allowing for the specification of various parameters. The number of nodes in the graph is controlled by the parameter N. The degree distribution is determined by the average degree k_{avg}, maximum degree k_{max}, and the power-law exponent τ_1. The number of edges that a node u shares with its communities, denoted as $k_u^{(in)}$, is controlled by the topological mixing parameter μ. Given the degree k_u of node u, the number of edges it shares with its communities is calculated as:

$$k_u^{(in)} = (1 - \mu) \cdot k_u \tag{2.7}$$

The community size distribution is determined by the exponent τ_2, as well as the minimum and maximum possible sizes of the communities, denoted as $\min(n_c)$ and $\max(n_c)$, respectively. Additionally, the number of overlapping nodes o_n and the number of memberships for the overlapping nodes o_m can also be specified.

The output of the algorithm includes the connections between nodes in the form of an edge list, as well as a list of communities with their corresponding node memberships. This information allows us to evaluate the ability of the set of algorithms $\mathcal{A}$ to accurately **detect the ground truth communities**.

2.3.2 Overlapping Normalized Mutual Information

After obtaining the results of the algorithms, they can be represented as a membership matrix M. In this matrix, each row represents a community, and each column represents a node in the graph. The value $M_{u,c}$ in cell (u, c) is set to 1 if node u is a member of community c, and 0 otherwise.

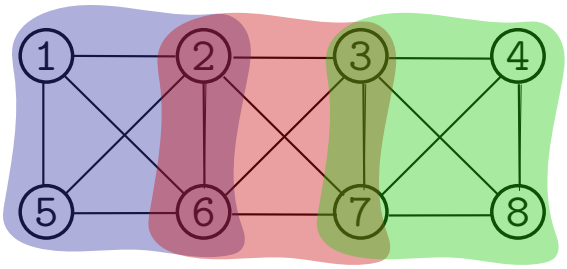


Figure 2.3: Example of a community division membership matrix

In the example community division (2.3) the membership matrix M would be defined as:

$$M = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

To compare the results of algorithm $\mathcal{A}_i$ with the ground truth, we aim to compare the membership matrix $\hat{M}$ obtained from algorithm $\mathcal{A}_i$ with the ground truth matrix M. To achieve this, **Normalized Mutual Information** (NMI) was proposed by [7]. However, it was later discussed in [37] that the normalization applied in the previous work was unconventional and led to an overestimation of the similarity between the results.

To address this issue, a different normalization method was proposed by the authors. In this approach, the mutual information between matrices M and $\hat{M}$ is defined as the average:

$$I(M : \hat{M}) := \frac{1}{2}\Big[H(M) - H(M|\hat{M}) + H(\hat{M}) - H(\hat{M}|M)\Big] \tag{2.8}$$

Here, $H(M)$ and $H(\hat{M})$ represent the entropy of matrices M and $\hat{M}$, respectively. The entropy of a matrix is approximated based on the conditional entropy of each vector. Subsequently, the overlapping normalized mutual information (NMI_{max}) between M and $\hat{M}$ is defined as follows:

$$NMI_{max} = \frac{I(M : \hat{M})}{\max(H(M), H(\hat{M}))} \tag{2.9}$$

Since communities may be listed in different orders between the matrices, NMI_{max} refers to the result of the best match between the matrices, accounting for different permutations of the communities.

Chapter 3

BLANT-C

The algorithm proposed for community detection employs a sampling of graphlets generated by BLANT as its starting point. This selection of subgraphs offers a highly advantageous starting point for uncovering communities. In this chapter, **BLANT** will be introduced and and it will be illustrated how the output of this algorithm is utilized for the proposed algorithm for network discovery.

3.1 Graphlets

The investigation of network topology is a broad domain of theoretical computer science research that holds considerable significance in bioinformatics. The fundamental approach involves modeling a graph G using synthetic networks in order to leverage its topological properties. This was the driving force behind the study presented in [38], which introduced the notion of **graphlets** and employed statistical analysis to demonstrate that protein-protein interaction networks were more accurately modeled using geometric random graphs as opposed to scale-free networks.

A k-graphlet g of a graph G is a connected induced subgraph of k nodes. The **automorphism orbits** of graphlet g are the nodes that are topologically identical to each other in g. An extension of k-graphlets are **k-graphettes**, which are induced subgraphs of k nodes that are not necessarily connected. These structures can be useful if we are doing statistical sampling of a network (selecting k nodes are random). The utilization of graphlets has been extensive in various applications such as network classification and comparison [39], local and global network alignment [2] and identification of the interplay between structure and function [40].

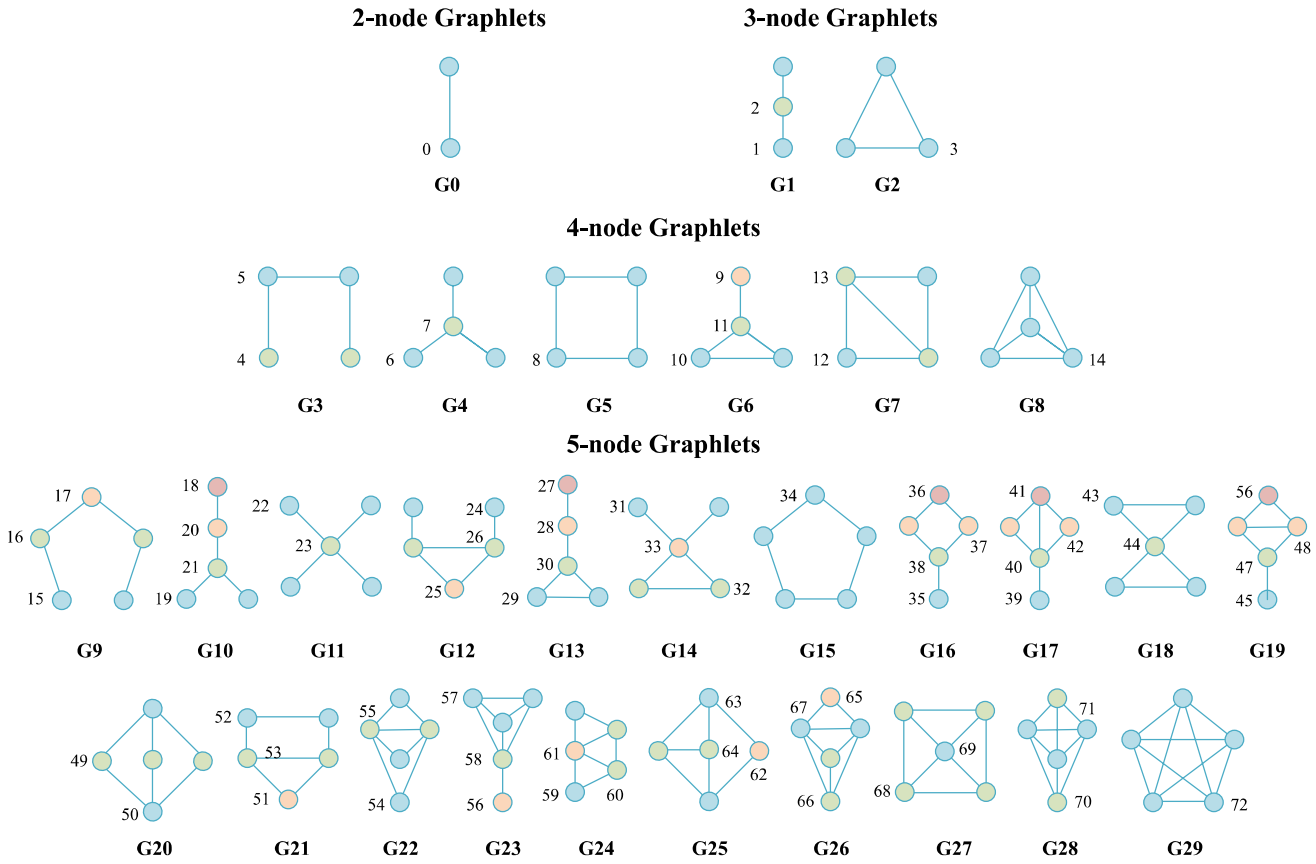


Figure 3.1: Graphlets up to $k = 5$.The orbit numbering is not necessary standard. *Figure extracted from [41]*

Sampling all the graphlets present in a network can quickly become a computationally intractable problem as the number of graphlets exponentially increase with both the graphlet size k and the number of nodes in the graph n. Consequently, exhaustive enumeration of

all graphlets is not feasible for large networks. This is where statistical methods, such as BLANT, come into play. By efficiently sampling subgraphs from the network, BLANT provides a valuable tool for approximating the full distribution of graphlets in a given network.

3.2 BLANT

The **Basic Local Alignment of Network Topology** (BLANT) [42] is a tool analogous to BLAST, specifically designed for performing local alignments of networks. BLAST, which stands for **Basic Local Alignment Search Tool**, is an algorithmic approach for rapidly identifying local alignments in proteomic and genomic sequences.

BLAST constructs a database of k-letter sequences, known as k-mers, which can subsequently be employed to initialize a local alignment between distant regions of two sequences. One of the key factors contributing to the speed and efficiency of the BLAST algorithm is its ability to access k-mers in constant time.

3.2.1 Graphette and orbit identification

BLANT is inspired by BLAST and employs a **pre-computed lookup table** that contains the lower triangular matrix of the adjacency matrix of all graphettes up to $k = 8$, represented as bit vectors. It also includes the pre-computed automorphism orbits of all canonical graphettes. A canonical graphette $K(g)$ is defined in the first paper published related to BLANT [43], which also describes how to compute the lookup tables.

In graph theory, an isomorphism is a mapping between two graphs that preserves their structure. Specifically, given two graphs G and H, an isomorphism is a bijection $f : V(G) \to V(H)$ between the vertex sets of G and H that preserves adjacency. That is, f maps adjacent

vertices in G to adjacent vertices in H, and non-adjacent vertices in G to non-adjacent vertices in H.

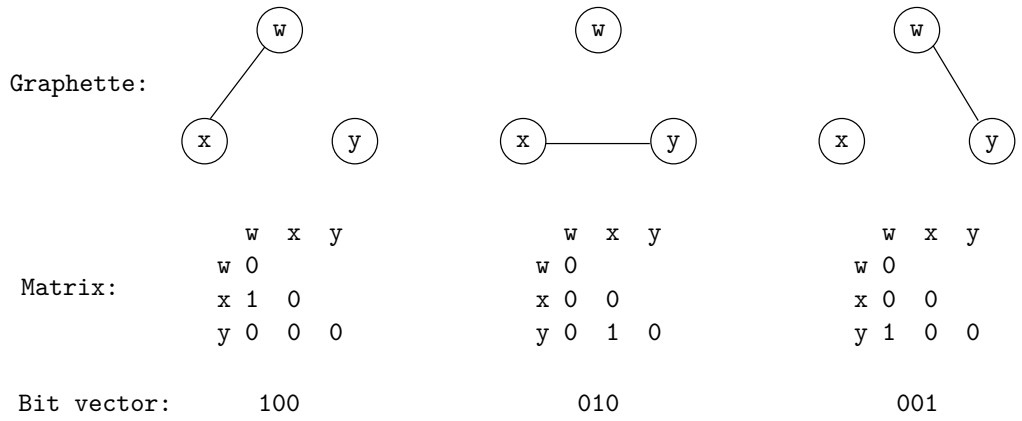


Figure 3.2: All the 3-graphettes of 1 edge. The canonical representation is the one one with the lowest bit vector (001). In this example, nodes that share an edge have the same orbit (i.e they are automorphic), while the disconnected node has a different orbit. Note that the three graphettes are isomorphic.

For a graphette g, the **canonical graphette** $\mathcal{K}(g)$ is a representative member of its isomorphism group. That is, all graphettes in the isomorphism group of g can be obtained from $\mathcal{K}(g)$ by applying a sequence of graph isomorphisms. By choosing a canonical representative for each isomorphism class of graphettes, BLANT is able to store $\mathcal{K}(g)$ and the permutations in lookup tables.

In runtime, with this information precomputed, given a graphette g of $k \leq 8$ nodes BLANT can compute in constant time the graphette identity and which orbit each of the k nodes belongs to.

3.2.2 Sampling and output

The BLANT software tool offers users the capability to extract a large number (up to millions per second) of sampled k-graphettes, where k ranges from 3 to 8. The tool provides flexibility in selecting various graphette sampling techniques and output formats. More-

over, the sampling computation process can be distributed across t threads for improved efficiency. BLANT offers five different sampling methods and five output formats to choose from, allowing users to tailor the tool to their specific needs and preferences.

Sampling methods

Among the available sampling methods in BLANT, the first one is **Node Based Expansion** (NBE). In NBE, each sample starts by selecting an edge uniformly at random from the input network. The two nodes at the endpoints of the selected edge are added to the graphette sample S_i, where $i \in [1, numSamples]$. Subsequently, the method incrementally chooses the remaining $k - |S_i|$ nodes in the graphette by uniformly selecting neighbors of the current set of vertices. If the number of neighbors of S_i is insufficient to complete the graphette, a new location is randomly chosen to add the remaining nodes. NBE allows for exploration of different regions of the network for a small value of $numSamples$, but it may introduce bias in the sampling process.

The **Edge Based Expansion** (EBE) method offers asymptotic speed improvement compared to NBE in very dense networks. Similar to NBE, EBE selects the first two vertices in the sample randomly. However, when gradually expanding S_i, it chooses adjacent nodes with a probability proportional to the number of edges connecting them to S_i. Although EBE is faster, it also introduces more bias into the sampling process.

The third method is **Reservoir Sampling** (RES), which reduces the bias of NBE but sacrifices computer efficiency . For each iteration, RES is initialized with an NBE sample $\hat{S}_i$ of k nodes. After that, for each sample, a random walk process takes place. For a number of steps, one node u is deleted from $\hat{S}_i^{t_j}$, and another adjacent node to $\hat{S}_i^{t_j} - \{u\}$ is selected at random. This is: $\forall j \in [1, s]$, where s is the number of steps, $\hat{S}_i^{t_{j+1}} = \hat{S}_i^{t_j} - \{u_{\text{random}}\} + \{v_{\text{random}}\}$. After these steps, the resulting $S_i = \hat{S}_i^{t_s}$.

The fourth method provided by BLANT is **Accept/Reject** (AR). AR selects k nodes uniformly at random and rejects the sample if the resulting subgraph is disconnected. AR is unbiased and asymptotically correct, but it can be exponentially slow as many samples in large networks will be rejected. This is why it is only provided for testing for testing purposes.

Finally, the **Markov Chain Monte Carlo** (MCMC) sampling method is available. This method finds and outputs the first sample S_1 in the same way NBE does. Nevertheless, to compute the next sample, it does not look for a new edge at random to start over. Instead, it begins a random walk with as many steps as there are samples left to output. For each step, it deletes at random one node u from S_i and selects - also randomly - a node v adjacent to $S_i - \{u\}$, outputting sample $S_{i+1} = S_i - \{u\} + \{v\}$.

This last method is the fastest per sample and, in the long run, is guaranteed to produce unbiased samples. The disadvantage is that on short timescales, its local random walk nature makes it fail to explore the whole network. Nevertheless, it can compute the bias on-the-fly and output asymptotically correct concentrations using the frequency output mode.

Output formats

Although BLANT includes an optimized output format for community detection algorithms, it's worth mentioning the five existing output formats available in the tool.

The most distinctive output format to BLANT is the **indexing Mode**. In this format, each line represents a graphlet sample and consists of $k + 1$ columns. The first column contains a canonical ID for the graphlet, while the following k columns represent the nodes that form the graphlet. The order of the nodes in this format enables local alignment with other graphlets sharing the same canonical ID.

A similar format is the **orbit indexing mode**, which shares similarities with the Indexing Mode. However, in this format, nodes belonging to the same automorphism orbit within a graphlet are separated by colons. Consequently, the number of columns in a line varies based on the number of orbits present in the graphlet. It's important to note that the Orbit Indexing Mode does not preserve the alignment property provided by the Indexing Mode.

For the purpose of frequency analysis, BLANT offers the **frequency mode** output format. This format produces a list with as many lines as there are canonical graphlets for a specific value of k. Each line consists of two columns: the canonical ID of the graphlet and the corresponding frequency or concentration of that particular graphlet. The frequency mode is particularly useful when using the MCMC sampling method, as it allows for on-the-fly bias computation and output of graphlet concentrations.

The **orbit degree vector** (ODV) output format closely resembles the output format of ORCA, an exhaustive sampling algorithm. Selecting this format in BLANT results in a number of lines equal to the number of nodes in the graph. Each line contains multiple columns representing the different orbits found in graphlets of size k. The frequency number within each column indicates how many times a node was sampled within that specific orbit of a graphlet. It's worth noting that due to the stochastic nature of BLANT, the output of ODV may differ from that of ORCA, which operates deterministically and exhaustively.

Lastly, the **graphlet degree vector** output format is similar to ODV, but instead of providing a list of orbit counts per node, it presents a list of graphlet counts per node. This format generates a number of lines equal to the number of nodes in the graph, and each line contains a number of columns corresponding to the canonical graphlets for a given value of k. The count in each column represents how many times a node was sampled within that specific graphlet.

3.3 BLANT-C

The proposed algorithm starts with the realization that high edge density connected graphlets highly resemble to small communities. In fact, given the definition of community proposed in (1.3) **high edge density graphlets are communities**. With this idea in mind, an algorithm that uses graphlets sampled by BLANT as the community discovery mechanism is proposed.

3.3.1 BLANT configuration

In BLANT-C, BLANT is executed for each value of k ranging from 3 to 7. Sampling 8-graphlets is excluded due to the significant time required to run for large networks. The number of samples is determined by the equation $\frac{M \cdot N}{k}$, where N represents the number of vertices in the graph and M represents the mean number of times each node u in the network should be sampled. We have found out that $M = 100$ usually suffices, thought higher values such as $M = 1,000$ and $M = 10,000$ can produce better results at the expense of more CPU.

The chosen sampling method is **Markov Chain Monte Carlo** (MCMC), which offers a balance between bias and efficiency (3.2.2). BLANT-C utilizes a **custom output format** called "mc." This format incorporates the on-the-fly bias computation using MCMC and incorporates it into the output. Instead of outputting indexes or frequencies, the community output format includes the following information for each line: a node u, a canonical ID ID of the graphlet, the number of times u was sampled in graphlet ID, and the remaining nodes in the ID graphlets where u appears.

By employing this configuration, BLANT-C aims to discover communities by leveraging the graphlets sampled by BLANT, focusing on high edge density graphlets as potential communities.

3.3.2 BLANT-C algorithm

Without further ado, the BLANT-C algorithm framework will be presented. Then, the different parts of the algorithm will be dissected.

Algorithm 2 BLANT-C algorithm

```
1: procedure BLANT-C(G,M,μ,E,Q)
2:     for d = 1/e, e ∈ [1, 2, ..., E] do          ▷ This for loop is run in parallel
3:         for k ∈ [3, 7] do                        ▷ This for loop is run in parallel
4:             tmpResults[d][k]=listCommunities(G,M,μ,k,d)
5:         end for
6:         results[d] ← filter(tmpResults[d],μ)
7:     end for
8:     communities ← filter(results,μ)
9:     overlapGraph ← createOverlapGraph(communities)
10:    communities ← optimize(Q, G, overlapGraph, communities, μ)
11:    return communities,overlapGraph
12: end procedure
```

In the context of the algorithm, G represents the network that needs to be divided into communities. As mentioned before, M denotes the sampling multiplier, while μ represents the overlapping threshold for the resulting communities. For example, if $\mu = 0.5$, it implies that two communities generated as output can share up to 50% of the nodes from the larger community. If the overlap exceeds this threshold, the smaller community is discarded.

Additionally, the parameter E guides BLANT-C in determining the number of edge densities to consider. These edge densities are evenly distributed along a linear scale. For instance, if BLANT-C is executed with $E = 5$, it will search for communities at edge densities of $[0.2, 0.4, 0.6, 0.8, 1]$.

During the execution of BLANT-C, **parallel** processing is employed to optimize computational performance. The number of samples generated by BLANT is determined based on the value of k and the sampling multiplier M, ensuring that a sufficient number of samples are taken using the MCMC sampling method to adequately explore the entire graph.

Algorithm 3 Find list of communities

```
1: procedure LISTCOMMUNITIES(G,M,μ,k,d)
2:     N ← numberOfNodes(G)
3:     n ← (M·N)/k
4:     samples ← runBLANT(G, k, n, MCMC, mc)
5:     tmpCommunities ← discover(G, samples, d, k)
6:     return tmpCommunities
7: end procedure
8: procedure NUMBEROFNODES(G)
9:     Returns the number of nodes in G
10: end procedure
11: procedure RUNBLANT(G, k, n, S, O)
12:     Runs BLANT in G to sample n k-graphettes using sampling method S
13:     and output format O
14: end procedure
15: procedure FILTER(communities, μ)
16:     communities ← join(communities)
17:     for community ∈ communities do
18:         if !overlapsAboveThreshold(filteredCommunities, community, μ) then
19:             filteredCommunities.add(community)
20:         end if
21:     end for
22:     return filteredCommunities
23: end procedure
24: procedure JOIN(communities)
25:     Joins a list of lists of communities in one list and sorts by size of the community
26: end procedure
27: procedure OVERLAP(c1,c2)
28:     return size(c1 ∩ c2)/MAX(size(c1), size(c2))
29: end procedure
30: procedure OVERLAPSABOVETHRESHOLD(filteredCommunities, community, μ)
31:     for fc ∈ filteredCommunities do
32:         if overlap(community, fc) > μ then
33:             return True
34:         end if
35:     end for
36:     return False
37: end procedure
```

For each specified edge density, community discovery is performed for every value of k, resulting in a set of communities. These communities, obtained for each pair (k, d), are then combined into a separate list for each edge density. Any communities that exceed the

specified overlap threshold μ are filtered out during this process. The combined communities for each edge density are then sorted based on the number of nodes in each community. In the case of a tie, the communities are further sorted by the smallest node ID.

Finally, the communities from each edge density are merged into a single list. Once again, any communities that surpass the overlap threshold are filtered out. The resulting list of communities is then sorted one more time.

Algorithm 4 Community discovery

```
1: procedure DISCOVER(G, samples, d, k)
2:     processedSamples ← process(samples, d)
3:     freqCount, nodes, neighbors ← extract(processedSamples)
4:     for u ∈ nodes do
5:         community ← explore(u, nodes, neighbors, freqCount, G , d)
6:         if size(community)> k then
7:             communities.add(community)
8:         end if
9:     end for
10:    return communities
11: end procedure
12: procedure EXTRACT(processedSamples)
13:    for sample ∈ processedSamples do
14:        nodes.add(sample.node); freqCount[node]=sample.freqCount
15:        L.add(node, sample.neighbors)                    ▷ Adjacency list
16:    end for
17:    return freqCount, nodes, L
18: end procedure
```

During the processing stage, the BLANT graphette samples are examined, and any samples that have a lower edge density than the specified requirement are removed. Next, a list is created, sorted by the **frequency of node appearances** in the remaining graphlets. Each value in the list includes the node, its frequency, and the nodes in the graphettes where it appears. This list serves as the basis for constructing an exploration graph.

The **exploration graph** is built using the sorted list. Each node in the graph represents a node that is guaranteed to be part of a graphette that meets the edge density requirements.

```
19: procedure PROCESS(samples, d)
20:     minEdges ← roundUp(d · (k choose 2))
21:     for sample ∈ samples do
22:         if canonEdgeCount(sample.canonID) < minEdges then
23:             continue
24:         end if
25:         freqCount[sample.node]+=sample.Frequency
26:         neighbors[sample.node].add(sample.neighbors)
27:     end for
28:     for node ∈ freqCount do
29:         sample ← (node, freqCount[node], neighbors[node])
30:         processedSamples.add(sample)
31:     end for
32:     return sort(processedSamples)                    ▷ Sort by freqCount
33: end procedure
34: procedure CANONEDGECOUNT(canonID)
35:     Constant time lookup of the number of edges a canonical graphette has.
36: end procedure
```

An edge is added between two nodes if they share membership in a graphette that meets the edge density requirements.

Additionally, an adjacency list is constructed to facilitate node exploration during the community discovery process. This adjacency list allows for efficient traversal of neighboring nodes in the exploration graph. Furthermore, a list of nodes is created, specifying the order in which exploration should be performed.

The logic behind the order is that nodes with higher graphlet counts, indicating their involvement in **more high edge density graphettes**, should be given **higher priority** during exploration. These nodes are referred to as origin nodes, and each of them serves as the starting point for a search to discover communities.

Local expansion in BLANT-C follows a **Breadth First Search** (BFS) approach. When a node is dequeued, it is added to a new community. However, if adding the node would result in a drop in the edge density below the desired threshold, it is not included in the community and expansion is halted for that node.

Algorithm 5 Node exploration

```
1: procedure EXPLORE(originNode, nodes, neighbors, freqCount, G , d)
2:     visited, community, misses ← set(), set(), 0
3:     FQ ← FIFOQueue()
4:     FQ.push(originNode); visited.add(originNode); community.add(originNode)
5:     while !FQ.empty() do
6:         u ← FQ.next()
7:         community.add(u)
8:         if edgeDensity(community) < d then
9:             community.remove(u);misses++
10:            if misses>MAX(size(community), N/100) then                ▷ Heuristic
11:                break
12:            end if
13:        else
14:            expand(u, originNode, nodes, neighbors, freqCount, FQ, visited)
15:        end if
16:    end while
17: end procedure
18: procedure EXPAND(u, originNode, nodes, neighbors, freqCount, FQ, visited)
19:    for v ∈ randomOrder(neighbors[u]) do
20:        if shouldExplore(u, v, originNode, nodes, freqCount) then
21:            FQ.push(v); visited.add(v)
22:        end if
23:    end for
24: end procedure
25: procedure SHOULDEXPLORE(u, v, originNode, nodes, freqCount)
26:    if v ∈ visited then
27:        return False
28:    end if
29:    if nodes.index(v) ≤ nodes.index(originNode) then                  ▷ Heuristic
30:        return False
31:    end if
32:    if freqCount[v] / freqCount[u] ≤ 0.5 then                         ▷ Heuristic
33:        return False
34:    end if
35:    return True
36: end procedure
```

A count is maintained to track the number of nodes that fail to be added. If this count exceeds a specified threshold, the exploration process is terminated. This heuristic is employed to optimize time complexity by preventing prolonged failures during expansion. We refer to this search strategy as **Graphlet First Search** (GFS).

In the expansion process, when a node is successfully incorporated into a community, its neighboring nodes are explored to further expand the community. However, certain conditions must be met for a neighbor to be expanded. First, the neighbor should not have been visited in the current exploration, adhering to the principles of classic BFS. Second, the neighbor must be an origin node that appears after the current origin node in the exploration order. Lastly, the frequency count of the neighbor should not experience a significant drop compared to the node from which it is being expanded.

These **heuristics** are employed to ensure that the search remains efficient and focused on identifying communities of maximum size. Through practical implementation, these heuristics have proven to be highly effective.

Algorithm 6 Create community overlap graph

```
1: procedure createOverlapGraph(communities)
2:     K← size(communities)
3:     for c_i ∈ [1, K] do
4:         for c_j ∈ [i + 1, K] do
5:             overlap ← size(communities[c_i] ∩ communities[c_j])
6:             if overlap>0 then
7:                 A[c_i][c_j]=A[c_j][c_i]=overlap          ▷ Adjacency matrix
8:             end if
9:         end for
10:    end for
11:    return A
12: end procedure
```

The creation of a community overlap graph opens up a multitude of possibilities for further analysis and exploration. With communities of varying edge densities, one can explore options such as hierarchical community divisions, overlapping community divisions within a specific range of edge densities, optimization of divisions for specific applications, or even identification of cliques within the graph. **The range of possibilities is vast** and offers flexibility for tailored analyses.

In the current implementation of BLANT-C, the algorithm focuses on optimizing a com-

munity division based on a given measure. However, it is important to note that this final part of the algorithm can be replaced or modified to output divisions that align with specific requirements or objectives. By adapting this aspect of the algorithm, users can tailor the output to match their specific needs and goals, expanding the utility and applicability of the tool.

Various optimization approaches were explored during the study. One particular approach, as described below, prioritizes runtime improvements over optimizing the score, albeit with a slight sacrifice in the score metric.

Algorithm 7 Optimize community division

```
1: procedure OPTIMIZE(Q,G, overlapGraph, communities, μ)
2:     PQ← PriorityQueue()
3:     for community ∈ communities do
4:         if alreadyExplored(commuity) then
5:             continue
6:         end if
7:         connectedSubgraph ← DFS(community, overlapGraph)
8:         randomStart ← getRandomNode(connectedSubgraph)
9:         PQ.push(score(Q, randomStart),randomStart)
10:        visited.add(randomStart)
11:    end for
12:    while !PQ.empty() do
13:        community ← PQ.pop()
14:        attemptAddition(community, division, Q)
15:        search(community, overlapGraph, division, Q, μ, PQ)
16:    end while
17:    return division
18: end procedure
19: procedure ALREADYEXPLORED(community)
20:    Returns true if a community has already been seen in a previous DFS exploration
21: end procedure
22: procedure DFS(community)
23:    Runs a Depth First Search node traversal to retrieve all the communities
24:    that are reachable from the given community in the community graph
25: end procedure
26: procedure GETRANDOMNODE(subgraph)
27:    Returns a node of the subgraph uniformly at random
28: end procedure
```

29: **procedure** SCORE(Q, community)
30: **if** $Q == M^{ov}$ **then**
31: **return** $\frac{1}{n_c} \sum_{u \in c} \frac{\sum_{v \in c, v \neq u} A_{u,v} - \sum_{v \notin c} A_{u,v}}{k_u \cdot s_u} \cdot d_c$
32:
33: **else if** $Q == \mathcal{ED}_N$ **then**
34:
35: **return** $\sum_{\forall u \in c} \left(\frac{1}{s_u} \cdot d_c \cdot \frac{1}{n_c} \right)$
36: **end if**
37: **end procedure**
38: **procedure** ATTEMPTADDITION(community, division,Q)
39: $P \leftarrow$ `potentialScore(community, division,` Q`)`
40: **if** `currentScore(division,` Q`)` $<$ `P` **then**
41: `division.add(community))`
42: **end if**
43: **end procedure**
44: **procedure** POTENTIALSCORE(community, division, Q)
45: Returns the score of the division if the community was added
46: **end procedure**
47: **procedure** CURRNTSCORE(division, Q)
48: Returns the current score of the division
49: **end procedure**
50: **procedure** SEARCH(community, overlapGraph, division, Q, μ, PQ)
51: **for** $c_2 \in$ `overlapGraph.neighbors(community)` **do**
52: **if** $c_2 \in$ `visited` **then**
53: **continue**
54: **end if**
55: $P \leftarrow$ `potentialScore(`c_2`, division,` Q`)`
56: $CS \leftarrow$ `currentScore(division,` Q`)`
57: **if** $CS < P$ `and overlap(community,` c_2`)` $\leq \mu$ **then**
58: `PQ.push(`P`,` c_2`)`
59: **end if**
60: `visited.add(`c_2`)`
61: **end for**
62: **end procedure**

To optimize the measures in BLANT-C, a **greedy approach with random start** is employed. Initially, the connected subgraphs of the community overlap graph are identified. For each subgraph, a random node is selected as the starting point for the greedy expansion process.

During the expansion, each community is visited once. If adding a community to the division

would not decrease the current score, it is added to a priority queue. The priority of a community in the queue is determined by the score it would have if it were added to the division during the exploration.

Since the division changes every time a node is added, the potential scores in the priority queue can become outdated. However, it is important to note that the contribution of a community c_1 to the total score **will never increase** with the addition of another community c_2 to the division. It can only decrease. As a result, when a node is removed from the queue, its potential score is recomputed. If the recomputed score does not improve the current score, the community is not added to the division.

To ensure a comprehensive analysis of the algorithm's performance and explore its limits, a pure $O(n^2)$ greedy algorithm is employed in the comparative analysis. This algorithm iteratively selects the best available community until the score no longer increases, disregarding any trade-offs in runtime efficiency. This approach allows for a rigorous examination of the algorithm's capabilities and enables a thorough understanding of its performance in different scenarios.

As mentioned earlier, the output of BLANT-C provides an extensive list of communities. The subsequent steps following the community detection phase will depend on the specific application and objectives of the study. By considering the application context, researchers can make informed decisions about how to utilize the discovered communities and extract meaningful insights from them.

To summarize, BLANT-C harnesses the power of **BLANT** to sample graphlets within a network, enabling an informed **Graphlet First Search** local expansion approach for community discovery. Through this process, a comprehensive catalog of **big, dense communities** is obtained. Additionally, BLANT-C generates a **community similarity overlap graph**, which highlights the extent of overlap between communities. This valuable infor-

mation can be leveraged in various use cases. Given the stochastic nature of the algorithm and its expansion method, we posit that this approach achieves **near-optimal solutions** for community discovery[1].

[1]More about this in (7)

Chapter 4

Experimental setup

In this chapter, we will present the real-world networks used for the experiments and define the parameters assigned to the algorithms. Additionally, we will provide details about the computational resources utilized for running the algorithms.

4.1 Machines used

The algorithms in this study were executed without any time limitations and were allocated sufficient computational resources. The computational power utilized for running the algorithms was provided by the openlab machines at the Donald Bren School of Information and Computer Science.

Machine	**CPU cores**	**RAM**
circinus[1-96]	24	96GB
hermod	32	192GB
odin	64	512GB
poison	40	95GB
tristram	32	96GB

Table 4.1: Computational resources.

This approach ensured that the algorithms had access to the necessary resources required for their execution. The school has equipped all students and faculty members with machines as listed in the table. These machines were extensively utilized to parallelize the work and conduct a sufficient number of experiments. By making the most of these resources, the research was able to run numerous experiments simultaneously, effectively utilizing the computational power available.

4.2 The data

In order to comprehensively evaluate each algorithm, a diverse dataset comprising various sizes and network topologies was selected. The choice of datasets was based on the commonly used datasets in the literature, as well as those available in the Stanford Large Network Dataset Collection of SNAP [44].

Graph	**Nodes**	**Edges**	**Size**
Zachary's karate club	34	78	405 B
American College football	115	613	3.728 KB
Protein Interaction	1,846	2,203	18.375 KB
Arxiv HEP-TH citations	27,769	352,285	3.705 MB
Amazon product network	334,863	925,872	12.065 MB
DBLP collaboration network	317,080	1,049,866	13.409 MB
Social circles from Twitter	81,306	1,342,296	15.135 MB
Youtube online social network	1,134,890	2,987,624	38.21 MB

Table 4.2: Real Networks dataset

While a few small graphs were included to verify the correct operation of the algorithms, larger graphs, such as the YouTube graph with 2,987,624 edges, were also incorporated to assess the scalability and performance of the algorithms under realistic conditions. By utilizing datasets of varying sizes and topologies, the study aimed to ensure a fair and comprehensive evaluation of each algorithm's effectiveness.

4.3 Algorithms' parameters

The following chapters compare BLANT-C against four alternative algorithms: BigClam (2.1.1), CFinder (2.1.2), COPRA (2.1.3), and Demon (2.1.4).

In each experiment, the algorithms were run using the parameters recommended in their respective papers. For BigClam, the number of communities to attempt was determined automatically, with a minimum of $m_c = 5$ and a maximum of $x_c = 100$. The number of trials was set to $n_c = 10$, and the number of threads used was $n_t = 4$. The backtracking search parameters were configured as $\alpha = 0.05$ and $\beta = 0.3$.

Regarding CFinder, the size of k-cliques was set to $k = 4$. In the case of COPRA, the maximum number of memberships allowed for a node was restricted to $v = 8$. Lastly, for Demon, the merging threshold was set to $\epsilon = 0.3$, and the minimum community size was specified as $m_c = 3$.

By employing these specific parameter configurations for each algorithm, the experiments aimed to ensure consistency and comparability across different community detection approaches.

Algorithm	**Parameters**
BigClam	$m_c = 5$, $x_c = 100$, $n_c = 10$, $n_t = 4$, $\alpha = 0.05$, $\beta = 0.3$
CFinder	$k = 4$
COPRA	$v = 8$
Demon	$\epsilon = 0.3$, $m_c = 3$
BLANT-C	$M = 100$, $k \in [3, 7]$, $E = 22$, $d_c \in [0.0056, 1]$, $\mu = 0.2$

Table 4.3: Parameters of the algorithms

For BLANT-C, it was executed with a sampling multiplier of $M = 100$. The community discovery process was performed for values of k ranging from 3 to 7, and for a total of $E = 22$ different edge densities evenly spaced within the range $dc \in [0.0056, 1]$. In order to discard candidate communities, an overlapping threshold of $\mu = 0.2$ was applied.

Chapter 5

Comparative analysis for real networks

Each algorithm was executed for every network in the dataset, enabling a thorough analysis of their performance. The results demonstrate that BLANT-C excels in identifying the largest and most densely connected communities compared to other algorithms. This achievement can be attributed to the utilization of Graphlet First Search expansion, which ensures a nearly uniform edge density within any induced subgraph.

In this chapter, the obtained results will be presented, focusing on these characteristics. A comparison of the uniformity of the found communities is exposed, as well as of the quality measures used to evaluate the effectiveness of the algorithms. The presentation of these findings will shed light on the superior performance of BLANT-C in terms of identifying large, dense communities, while also providing insights into its overall quality as compared to the other algorithms.

5.1 Denser, bigger communities

Upon evaluating the algorithms on the real networks described in (4.2), it is evident that BLANT-C surpasses other algorithms in terms of its ability to identify big and dense communities. The forthcoming plots depict the comparative coverage area wherein the algorithm identifies communities of specific edge densities and sizes.

To enhance the visualization, we have applied filters that eliminate local minima for the upper curve and local maxima for the lower curve. By employing this approach, we ensure that the results are neither overestimated nor underestimated, as opposed to methods like convex hulls or Gaussian noise filters. This filtering technique contributes to a smoother visualization of the comparative performance.

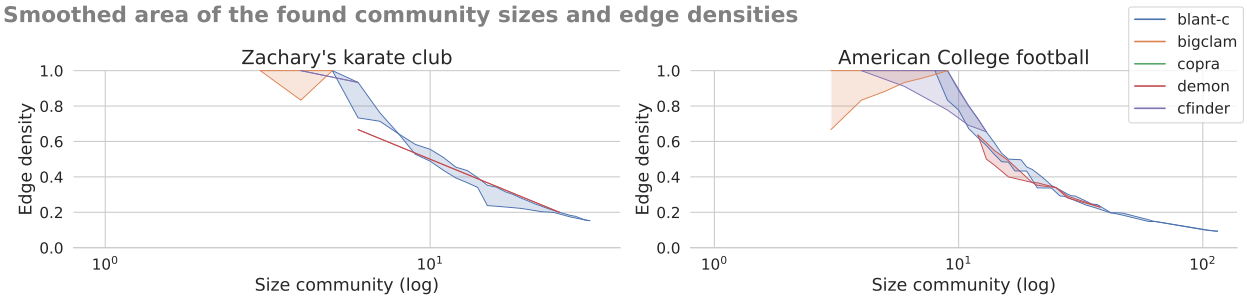


Figure 5.1: In the x-axis, the size of the community in a logarithmic scale. In the y-axis, the edge density of a community. The image shows the area occcupied by the solution given by each algorithm for the two smallest networks.

Although the differences may not be as pronounced in the case of the smallest networks, even in those instances, there are discernible advantages offered by BLANT-C. By examining the results obtained from Zachary's karate club network, it becomes evident that BLANT-C is superior in terms of both edge density and community size.

It is worth noting that the extent of improvement may vary depending on the specific network topology under consideration. However, across all cases, there is a consistent enhancement achieved by utilizing BLANT-C.

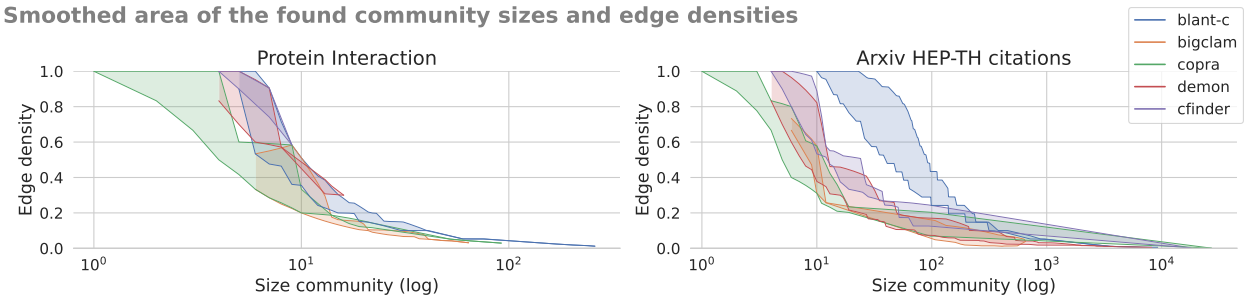


Figure 5.2: In the x-axis, the size of the community in a logarithmic scale. In the y-axis, the edge density of a community. The image shows the area occcupied by the solution given by each algorithm for the two networks that are between the smallest and the mid-size ones.

The Arxiv HEP-TH citations network demonstrates a remarkable improvement with BLANT-C. Its performance surpasses other algorithms, showcasing higher edge density and larger communities. Moreover, when examining the protein interaction network, BLANT-C achieves comparable results in terms of edge density and community size, while also displaying superior capability in identifying significant cliques within the network.

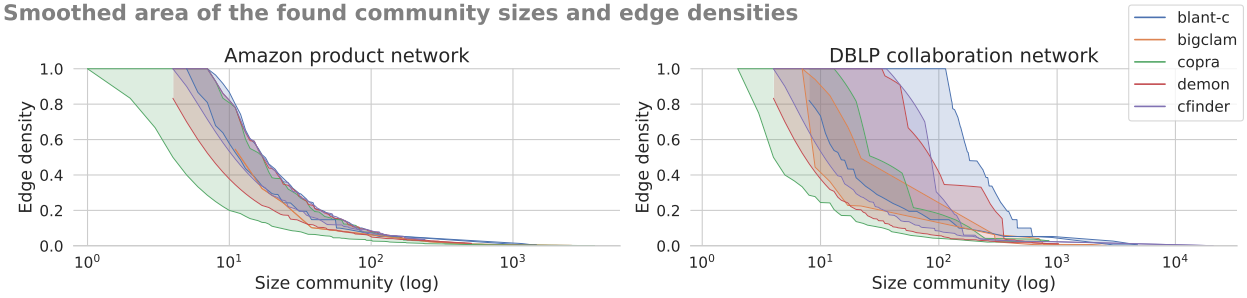


Figure 5.3: In the x-axis, the size of the community in a logarithmic scale. In the y-axis, the edge density of a community. The image shows the area occcupied by the solution given by each algorithm for the mid-size networks.

Upon evaluating networks of various sizes, it becomes evident that CFinder emerges as the second most competitive method for networks within that range. However, even in such cases, BLANT-C remains unrivaled in terms of performance, particularly evident in the DBLP collaboration network. BLANT-C demonstrates superior results in this network compared to all other algorithms.

When examining the Amazon product network, the improvement may not be as pronounced.

As acknowledged earlier, the degree of enhancement achieved by BLANT-C varies depending on the specific topology of the network. Nevertheless, despite the relatively less noticeable improvement, BLANT-C maintains its position as the most competitive algorithm for the Amazon product network.

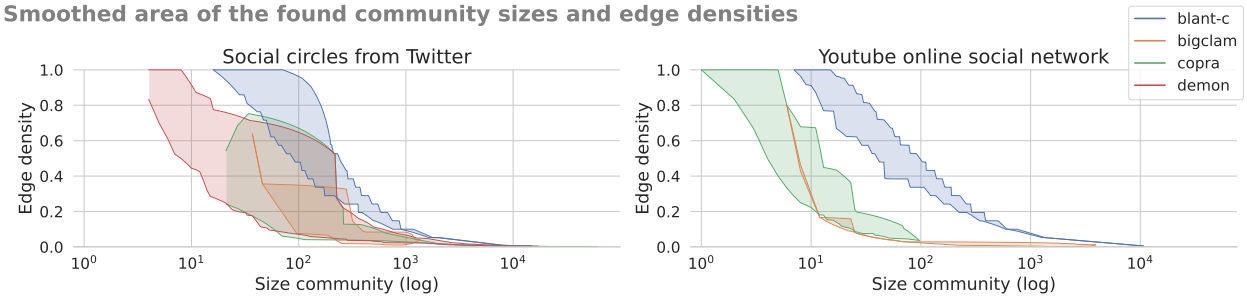


Figure 5.4: In the x-axis, the size of the community in a logarithmic scale. In the y-axis, the edge density of a community. The image shows the area occcupied by the solution given by each algorithm for the biggest networks.

In the case of the largest networks, CFinder proves to be ineffective in providing a solution, despite being allocated a substantial amount of RAM (up to 500GB with the Odin machine). This limitation indicates that CFinder faces challenges in handling such large-scale networks.

On the other hand, Demon manages to produce a solution for the Youtube online social network. However, the resulting solution consists of a single community comprising 44,514 nodes, with an edge density of only 0.0532843%. Consequently, there is no discernible area available for plotting this community due to its sparse nature.

In contrast, when running BLANT-C on these large graphs, there is a notable improvement in both community size and edge density. BLANT-C demonstrates its effectiveness in capturing meaningful community structures within these massive networks, highlighting its superiority over alternative algorithms.

The significant improvement observed in terms of size and edge density further emphasizes the strength of BLANT-C in handling large-scale networks and extracting meaningful communities from them.

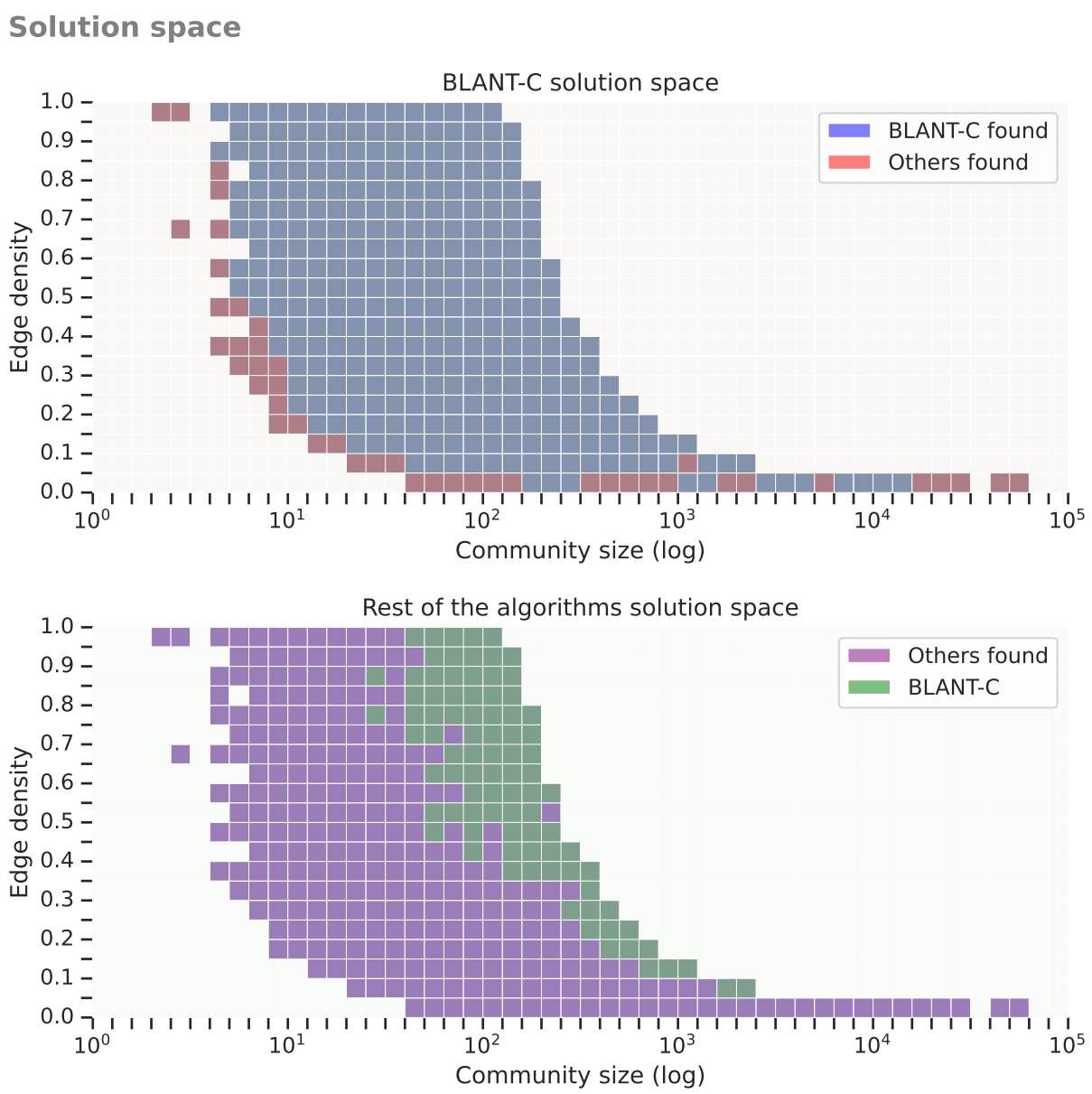


Figure 5.5: In the x-axis, the size of the community in a logarithmic scale. In the y-axis, the edge density. The plot shows the ranges in which BLANT-C and the rest of the algorithms found a community across every network. In blue, the solution space of BLANT-C. In purple, the union of the solution space of every other algorithm. The ranges in which BLANT-C found a solution but the rest of the algorithms did not is marked in green. The ranges in which the rest of the algorithms found a community but BLANT-C did not, are marked in red.

In conclusion, BLANT-C consistently outperforms other state-of-the-art algorithms in identifying larger and denser communities. This superiority is clearly demonstrated in Figure (5.5), where the solution space expands significantly, with an average 3.4x increase in community size for communities with an edge density $d_c \geq 0.55$. Notably, BLANT-C excels in finding cliques, with the ability to identify cliques of up to 126 nodes.

It is important to highlight that BLANT-C effectively identifies the majority of communities, with the exception of those characterized by extremely low edge density or communities with the same edge density but smaller size. However, even in such cases, BLANT-C seamlessly incorporates these smaller communities into larger ones with the same edge density, ensuring a comprehensive coverage of the network. These findings highlight the exceptional performance of BLANT-C in detecting significant communities within complex networks.

5.2 Edge density uniformity

As part of the re-imagined community definition (1.3), one key criterion is that a community with a specified edge density, denoted as d_c, should exhibit a uniform edge density across the entire community and its induced subgraphs. This criterion ensures that a community is not formed solely by a highly connected core with sparse connections in its surroundings.

One advantage of utilizing BLANT as a sampling tool is that it allows for the measurement of other interesting properties of the algorithms, such as the uniformity of edge density within the detected communities. Leveraging BLANT's capabilities, we can efficiently induce multiple subgraphs (graphlets) and compute their edge density in constant time.

To conduct the experiment, 100 graphlets were induced for each community within every solution of every network in the real networks dataset. This process was carried out for each k value within the range of 3 to 7. Subsequently, the edge density of each induced

graphlet was computed, enabling the evaluation of the uniformity of edge density across the communities.

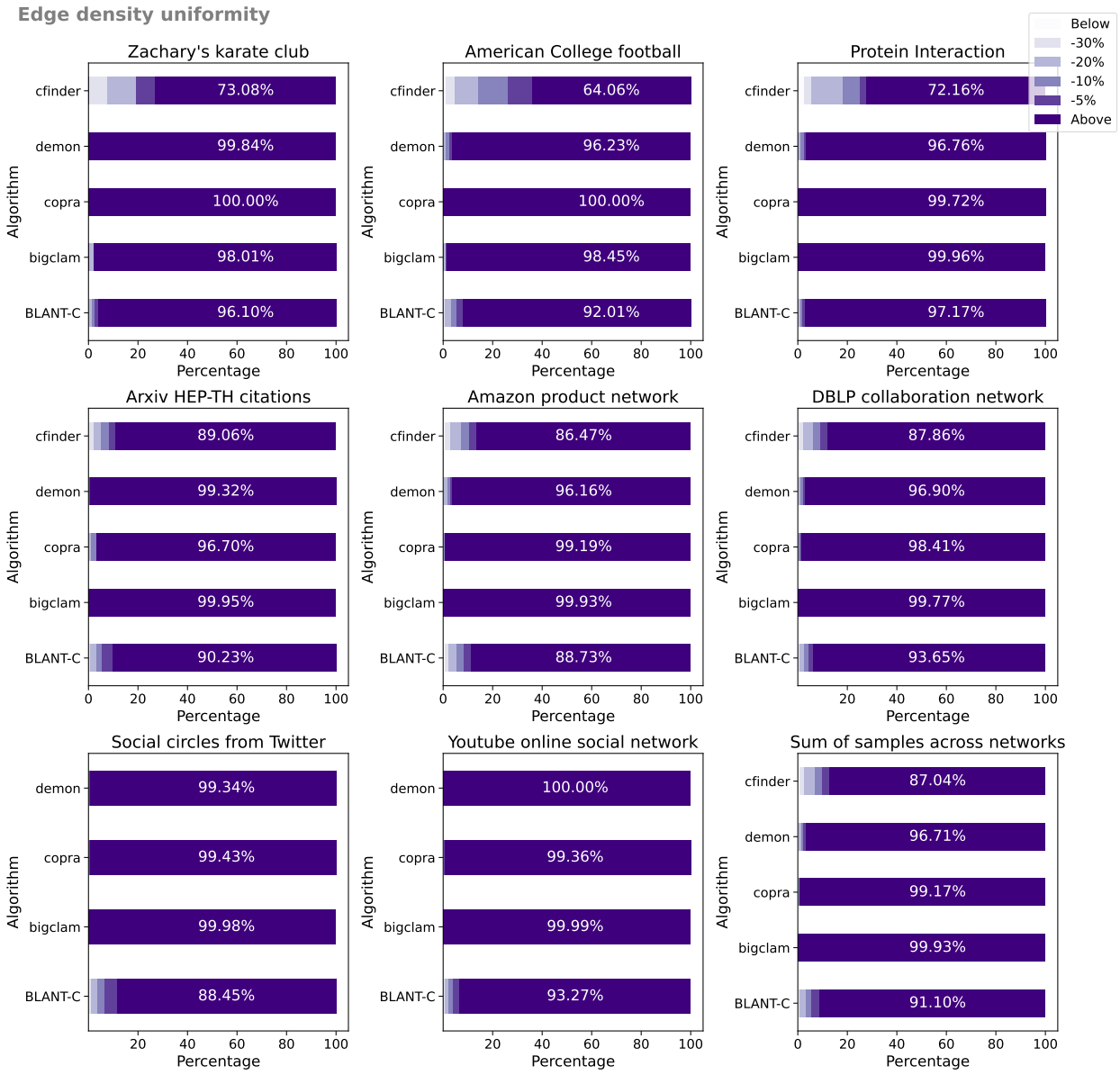


Figure 5.6: Uniformity of edge density for each network and for the sum of all Samples. The stacked bar graph illustrates the distribution of sampled subgraphs based on their edge density in relation to the edge density of the corresponding community. Subgraphs with an edge density equal to or higher than the community's edge density are represented in the 'Above' category. Subgraphs that exhibit a lower edge density are categorized based on the extent of the drop: less than 5%, more than 5% but less than 10%, more than 10% but less than 20%, more than 20% but less than 30%, and more than 30%.

The results indicate that algorithms utilizing label propagation and non-negative matrix

factorization demonstrate strong performance in detecting communities that adhere to the uniformity of edge density condition. In contrast, CFinder exhibits comparatively weaker results in this aspect. Notably, the proposed algorithm, BLANT-C, successfully identifies communities where at least 90% of all randomly sampled induced subgraphs maintains an edge density drop of no more than 5%. While these results are deemed satisfactory, there remains potential for further enhancements and improvements in future iterations of the algorithm.

5.3 Quality measures

Based on the solutions obtained by each algorithm for each network, the quality measures outlined in (2.2) have been computed. It is important to note that, as mentioned, the minimum size requirement for a subgraph to be considered a community is set at $n_c \geq 3$. Additionally, the condition for overlapping modularity, which necessitates every node to be a member of at least one community, has been relaxed.

Network	**BigClam**	**BLANT-C**	**CFinder**	**COPRA**	**Demon**
Zachary's karate club	0.1598	**0.8667**	0.8111	0.1390	0.4036
American College football	0.4585	**0.8646**	0.8106	0.0935	0.1938
Protein Interaction	0.1236	0.6320	**0.8814**	0.4186	0.7341
Arxiv HEP-TH citations	0.0431	**0.8419**	0.5665	0.5750	0.2049
Amazon product network	0.0149	**0.9363**	0.7064	0.2165	0.4397
DBLP collaboration network	0.0761	**0.8735**	0.6248	0.4921	0.4814
Social circles from Twitter	0.0426	**0.9141**	-	0.1584	0.0980
Youtube online social network	0.0563	**0.7981**	-	0.3876	0.0005

Table 5.1: $\mathcal{ED}_N$ score of the solution for each algorithm. The best score for each network is highlighted.

The rationale behind this relaxation was also discussed in (2.2). For BLANT-C, the results have been optimized for each measure using a parameter value of $\mu = 0.2$ and has been done after sorting the raw results by edge density instead of by size. This optimization process

aims to maximize the performance of BLANT-C with respect to each individual measure.

The table clearly demonstrates that BLANT-C consistently achieves the best results across all networks, with the exception of the Protein Interaction network. It is worth noting that CFinder, being a Clique Percolation Method, was expected to perform well in terms of the $\mathcal{ED}_N$ measure. However, as mentioned in 5.1, CFinder does not scale well for larger networks. This limitation becomes evident as CFinder's results are missing for the last two networks in the table.

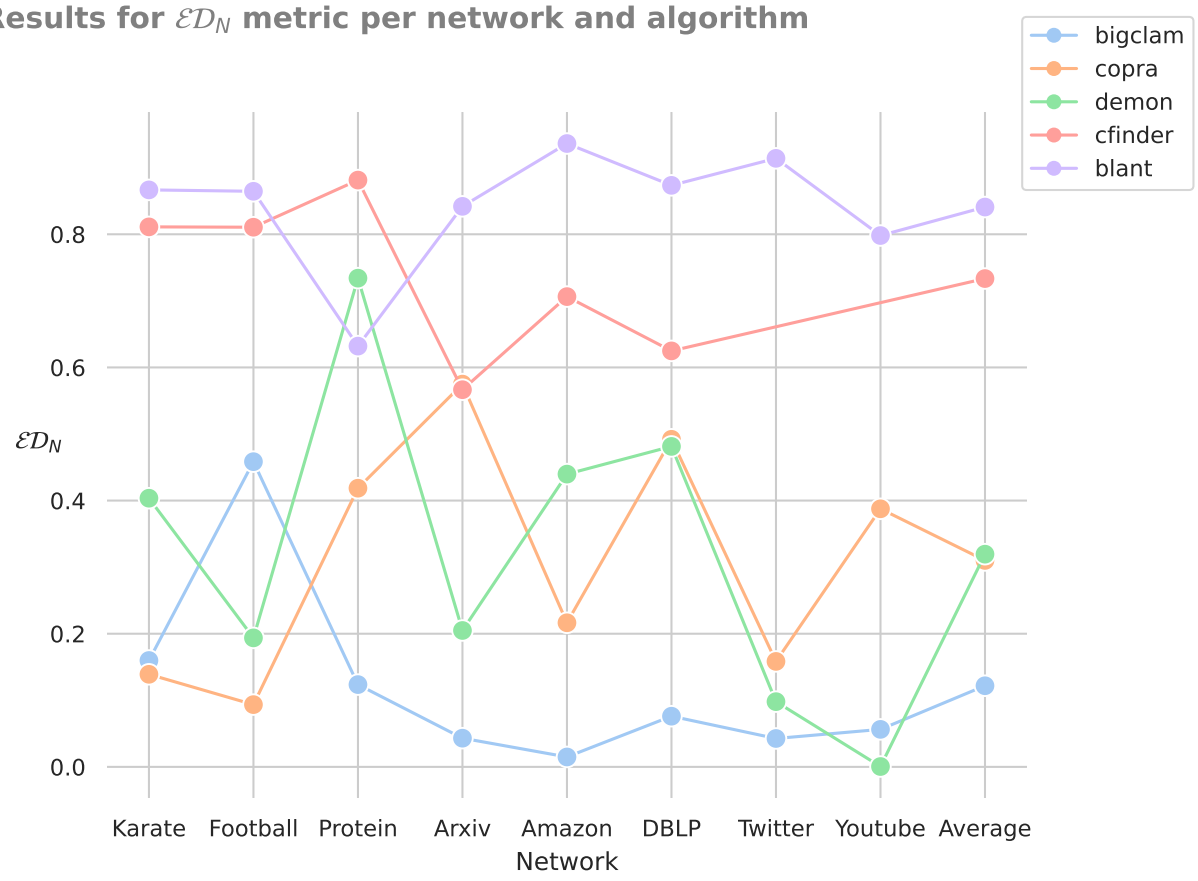


Figure 5.7: Comparison of $\mathcal{ED}_N$ scores across different networks and algorithms. Networks are arranged in ascending order of size from left to right. The final value represents the average score across all networks.

It is interesting to note that the Demon and COPRA algorithms compete in performance and frequently alternate positions depending on the network topology. This observation is

particularly evident in the DBLP collaboration network, where both algorithms achieve similar scores. On the other hand, BigClam consistently demonstrates the lowest performance among the evaluated algorithms in relation to this metric.

These findings highlight the nuanced nature of community detection algorithms and their performance characteristics, which can vary depending on the specific network characteristics and topology.

Although there have been previous objections regarding the suitability of the overlapping modularity measure and its alignment with the proposed community definition, it is worth noting that this measure is commonly employed in community detection research. Therefore, we have included the results for overlapping modularity as well.

Network	**BigClam**	**BLANT-C**	**CFinder**	**COPRA**	**Demon**
Zachary's karate club	0.0476	0.0819	0.1249	0.1390	**0.3531**
American College football	-0.1325	**0.3510**	0.2159	0.0935	0.0847
Protein Interaction	0.0991	0.1969	0.3045	0.2466	**0.3219**
Arxiv HEP-TH citations	0.0318	0.0657	-0.0615	**0.5543**	0.1046
Amazon product network	0.0129	**0.2793**	0.2789	0.1696	0.2505
DBLP collaboration network	0.0696	**0.4344**	0.3168	0.4136	0.3369
Social circles from Twitter	0.0314	**0.1549**	-	0.1471	0.0910
Youtube online social network	0.0520	**0.4216**	-	0.2364	0.0002

Table 5.2: Overlapping modularity score of the solution of each algorithm. The best score for each network is highlighted.

As observed in the results, BLANT-C surpasses the performance of the other algorithms for the four largest networks. This further reinforces the effectiveness and superiority of BLANT-C in detecting overlapping communities compared to the competing algorithms. Despite the concerns raised about the measure's compatibility with the community definition, BLANT-C's consistent outperformance demonstrates its ability to uncover meaningful and coherent communities even under the overlapping modularity framework.

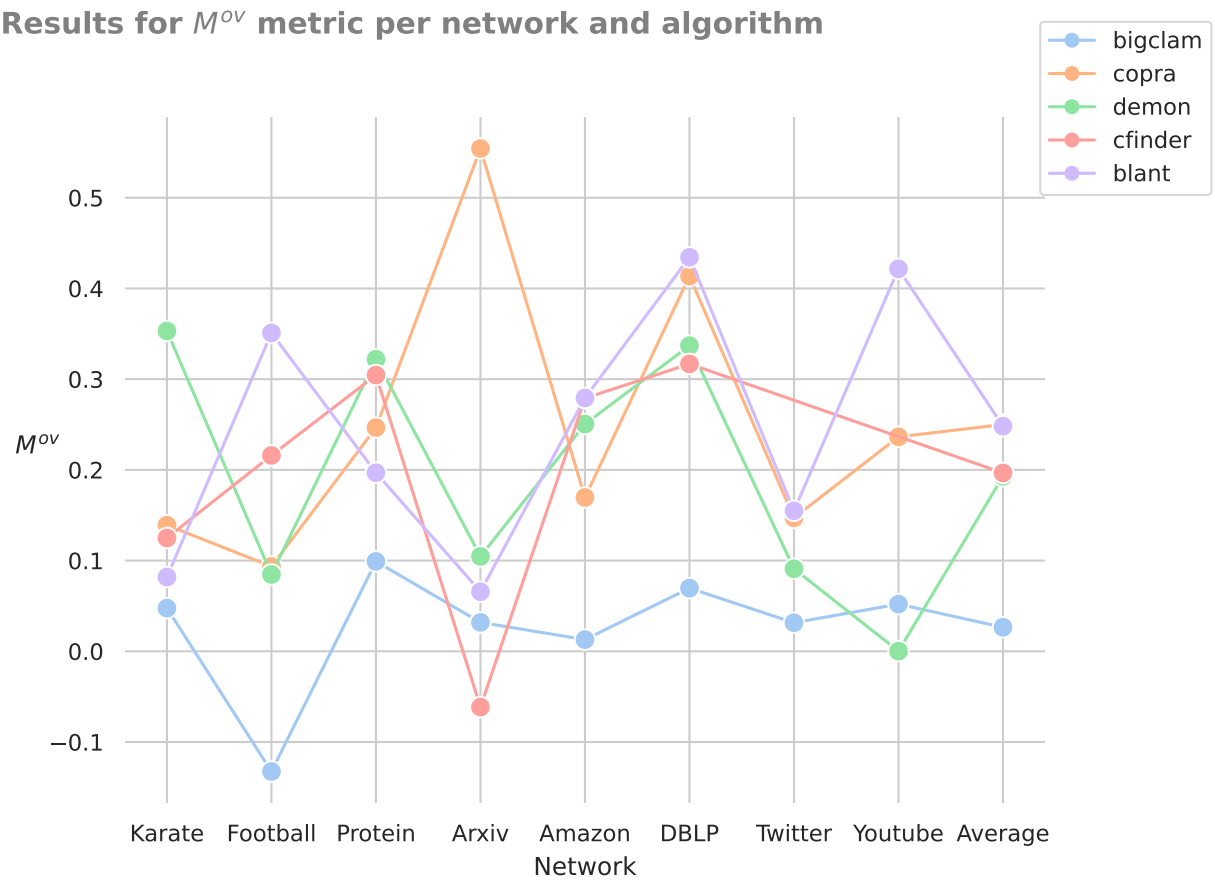


Figure 5.8: Comparison of M^{ov} scores across different networks and algorithms. Networks are arranged in ascending order of size from left to right. The final value represents the average score across all networks.

Indeed, the results for overlapping modularity indicate that the performance of different algorithms can vary depending on the specific network topology and size. BigClam consistently performs poorly in capturing overlapping community structures, suggesting limitations in its ability to handle complex network relationships.

On the other hand, BLANT-C, CFinder, and the label propagation algorithms demonstrate competitive performance, often vying for the top scores in this metric. It is worth noting that Demon's performance appears to decline as the network size increases, while BLANT-C consistently maintains its superiority in large graphs.

These observations highlight the importance of considering both the algorithm's performance and its scalability when dealing with networks of varying sizes. BLANT-C's ability to maintain its effectiveness and outperform other algorithms in larger graphs suggests its scalability and adaptability to handle complex community structures. Overall, these results underscore the significance of selecting an algorithm that not only performs well in capturing overlapping communities but also demonstrates scalability and robustness across different network sizes and topologies.

Chapter 6

Comparative analysis for synthetic networks

After analyzing the performance of BLANT-C on real networks, a comparative analysis was conducted using LFR benchmark graphs. The methodology closely follows the recent comparative analysis of [45], which served as a valuable reference.

In this analysis, the algorithms under consideration were executed on a dataset of LFR graphs. The obtained results were then compared with the ground truth communities using the overlapping normalized mutual information (NMI) measure, as defined in 2.3.1. Furthermore, the runtime of each algorithm was also analyzed, providing insights into their efficiency and computational performance.

Through this comparative analysis, certain weaknesses in BLANT-C were identified. Moreover, the analysis yielded important insights that can guide potential enhancements to improve the algorithm's performance.

6.1 LFR benchmark graphs generation

The primary objective of the comparative analysis was to assess the robustness of algorithms in the face of changes in community structure. This was achieved by modifying various parameters used in generating LFR benchmark graphs and evaluating how closely the algorithms approximated the ground truth established by the LFR algorithm.

For the generation of LFR benchmark graphs, default parameter values were used as follows: $N = 10000$ nodes, an average degree of $k_{avg} = 20$, and a maximum degree of $k_{max} = 50$. The topological mixing parameter was set to $\mu = 0.4$, while the degree distribution exponent and community size distribution exponent were $\tau_1 = 2$ and $\tau_2 = 1$ respectively. The community sizes ranged from $\min(n_c) = 20$ to $\max(n_c) = 100$. Aditionally, the number of overlapping nodes o_n is set to be 20% of N and the number of memberships of the overlapping nodes is set to $o_m = 2$.

While there was interest in evaluating how algorithms would react to changes in degree distribution and community size distribution, due to time constraints and convergence issues with the LFR generation algorithm, the focus of the analysis was primarily on evaluating the impact of μ, N, o_m, and o_n, on the algorithms' performance. To ensure clarity, only one parameter was modified at a time during the analysis.

LFR dataset	N	k_{avg}	k_{max}	μ	τ_1	τ_2	$\min(n_c)$	$\max(n_c)$	o_n	o_m
Study of μ-graphs	10,000	20	50	[0.1, 0.8]	2	1	20	100	$0.2 \cdot N$	2
Study of N-graphs	[10000, 100000]	20	50	0.4	2	1	20	100	$0.2 \cdot N$	2
Study of o_n-graphs	10,000	20	50	0.4	2	1	20	100	$[0.1 \cdot N, 0.6 \cdot N]$	2
Study of o_m-graphs	10,000	20	50	0.4	2	1	20	100	$0.2 \cdot N$	$[1, 9]$

Table 6.1: LFR benchmark graphs datasets.

For the mixing parameter μ, a range of values $[0.1, 0.8]$ was explored, with a step size of 0.1. The number of nodes N was tested within the range $[10000, 100000]$ nodes, with a step size of 10,000 nodes. It is worth noting that this was the only deviation from the referenced analysis [45], as we decided to generate larger graphs for evaluation purposes.

To assess the impact of the number of memberships of overlapping nodes o_m, a range of values $[1, 9]$ was considered, with a step size of 1. Finally, the number of overlapping nodes o_n was tested within the range of $[0.1 \cdot N, 0.6 \cdot N]$, with a step size of 0.1.

6.2 Algorithms configuration

The algorithms used for comparison with BLANT-C were executed with the same configuration, as specified in (4.3). In order to thoroughly explore the performance limits of BLANT-C, it was executed for values of $k \in [3, 7]$, and for 95 linearly spaced edge densities in the range of $d \in [0.05, 1]$. Each (k, d) pair was executed in parallel. To filter the results, an overlapping threshold of 1.0 was applied, and the community size was restricted to a range of 18 to 111.

For the evaluation of BLANT-C results, two measures were optimized: the overlapping modularity M^{ov} and a new measure referred to as $\mathcal{ED}_N$. Following this optimization, the overlapping normalized mutual information (NMI) was computed between the algorithms' results and the ground truth.

Notably, it was observed that depending on the graph, either M^{ov} or $\mathcal{ED}_N$ yielded higher NMI values. This suggests that neither measure is perfect, and there is still room for improvement in optimizing the results produced by BLANT-C.

6.3 Runtime

The runtime of BLANT-C is determined by calculating the average runtime of each parallel process using the GNU time tool, which captures the elapsed wall clock time. It is important to note that this runtime measurement excludes the optimization algorithm of BLANT-C and is applicable only to a distributed computing paradigm. If the experiment were to be conducted on a single machine, the runtime would need to be multiplied by the number of edge densities and values of k for which BLANT-C is executed.

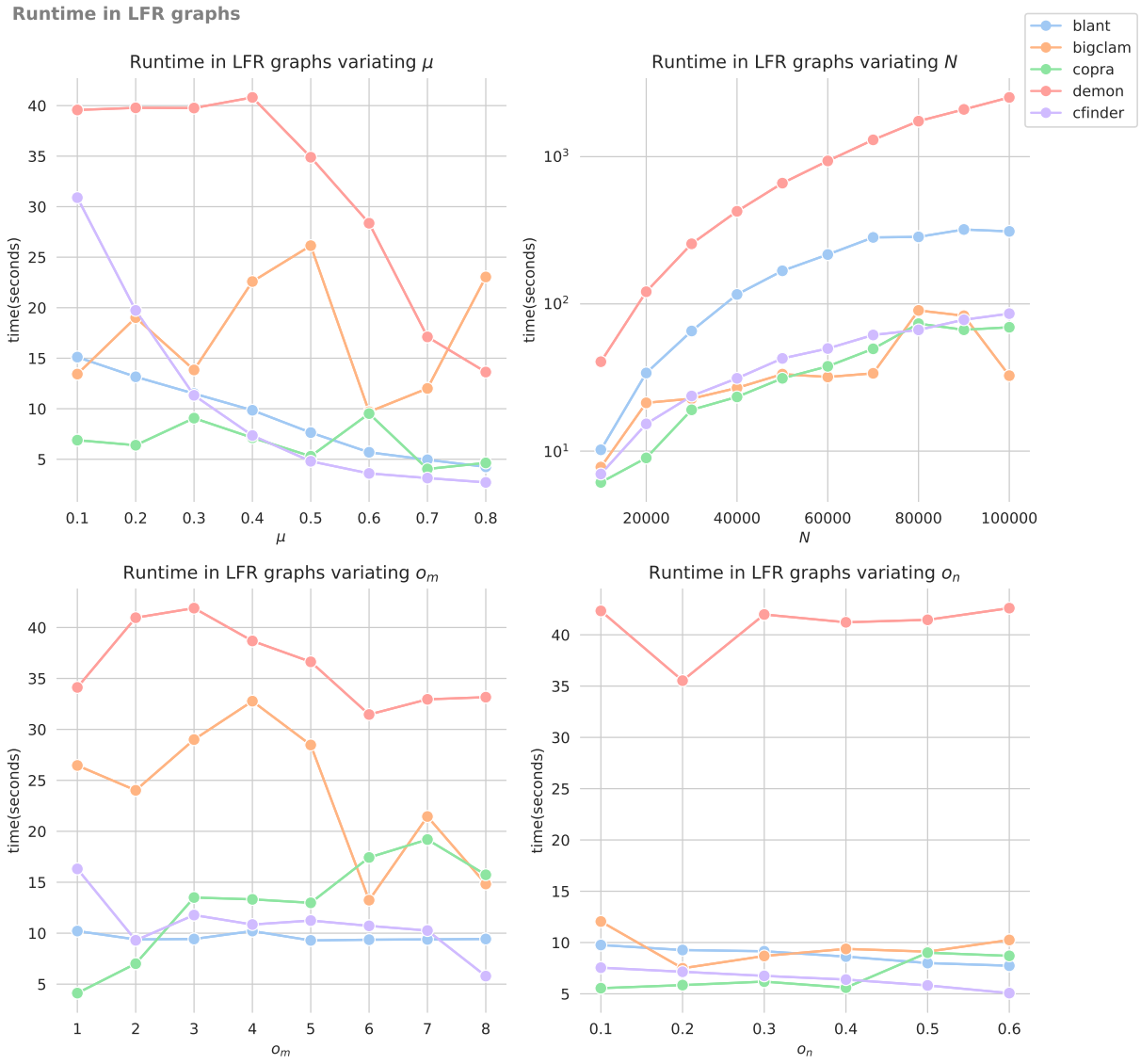


Figure 6.1: Runtime analysis in the LFR benchmark graphs datasets

The runtime of the algorithms demonstrates a direct relationship with the number of nodes in the graph, while remaining relatively constant for variations in other parameters. BLANT-C and Demon experience the most significant increase in runtime as the number of nodes increases.

Regarding CFinder, it is important to note that the reported runtime is not realistic, as CFinder's runtime is expected to exhibit exponential growth. However, in order to save time, a time limit per node was imposed during CFinder execution. Among the evaluated algorithms, COPRA and BigClam were found to be the fastest.

When the number of nodes grows, the runtime of BLANT-C is comparatively higher than that of other algorithms. This is primarily due to BLANT-C's comprehensive retrieval of communities, which can be up to 100 times larger. This information underscores the importance of exploring options in future iterations of the algorithm to enhance runtime efficiency. One potential approach is to allow users to selectively retrieve communities that align with their specific interests, based on predefined rules or criteria.

6.4 Normalized mutual information

Currently, BLANT-C lags behind label propagation methods in the task of detecting ground truth communities in LFR graphs. There are several reasons that contribute to this phenomenon. Firstly, the communities generated by the LFR algorithm adhere to a definition of community that differs from what was stated in (1.3). This disparity in community definitions presents a fundamental challenge for BLANT-C.

Additionally, BLANT-C detects a comprehensive list of communities (much longer than o_n) across a wide range of edge densities, whereas the communities generated by the LFR algorithm are concentrated within a narrower range of edge densities. While the first issue

cannot be resolved, it is worth noting that BLANT-C not only identifies communities that align with the ground truth but also identifies additional communities that were not defined by the LFR algorithm but still exhibit high density and size, which lowers the NMI.

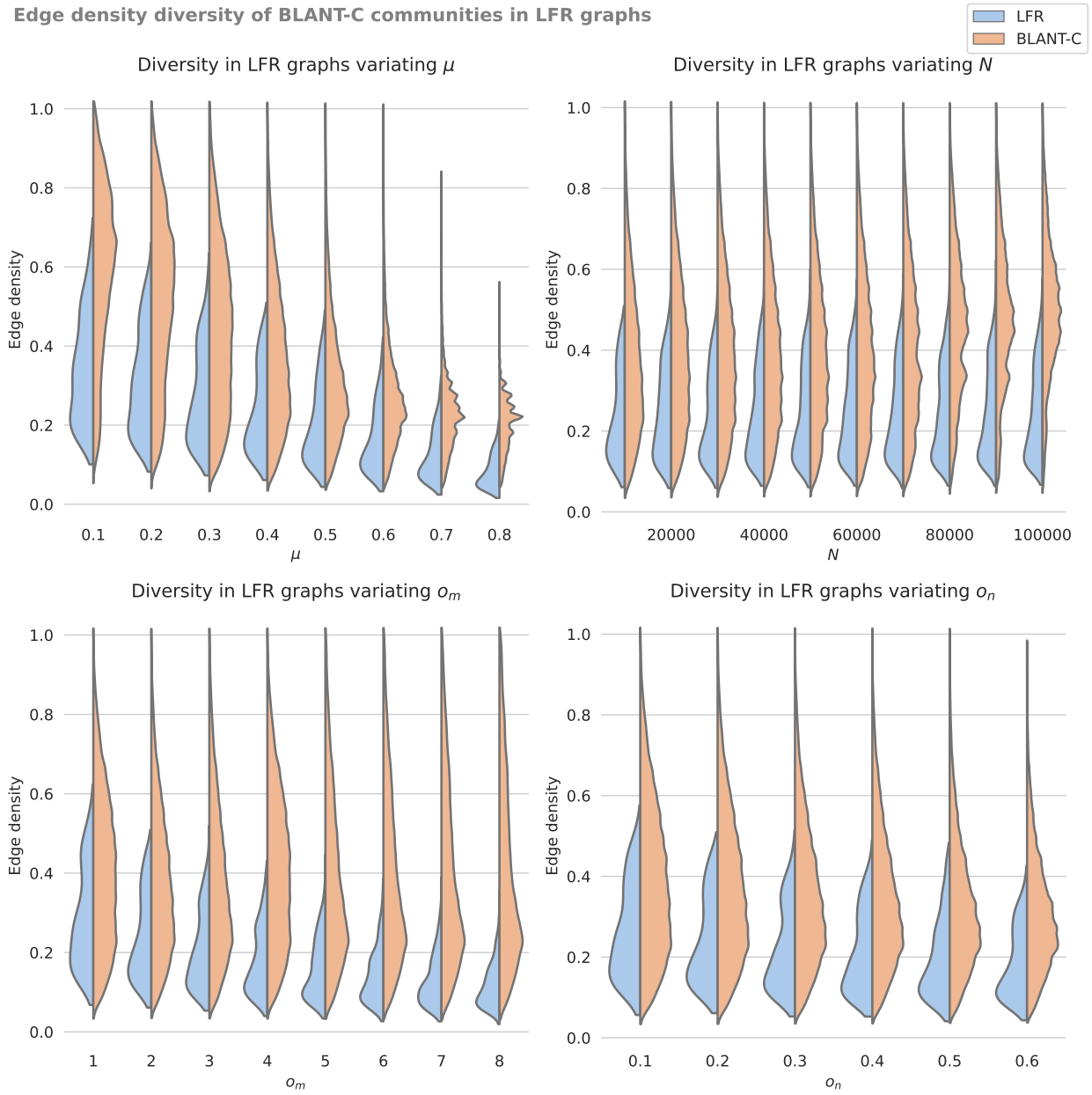


Figure 6.2: Distribution of the edge density of the communities defined by the LFR algorithm as the ground truth and the ones found by BLANT-C.

To address this, a comparison is presented that showcases both the maximum results achieved by BLANT-C after optimization and the results it would yield if only the communities match-

ing those defined by the LFR algorithm were selected from its extensive list of communities. This analysis allows for a comprehensive understanding of BLANT-C's performance, considering both its ability to detect ground truth communities and its capacity to identify additional dense and large communities beyond those defined by the LFR algorithm.

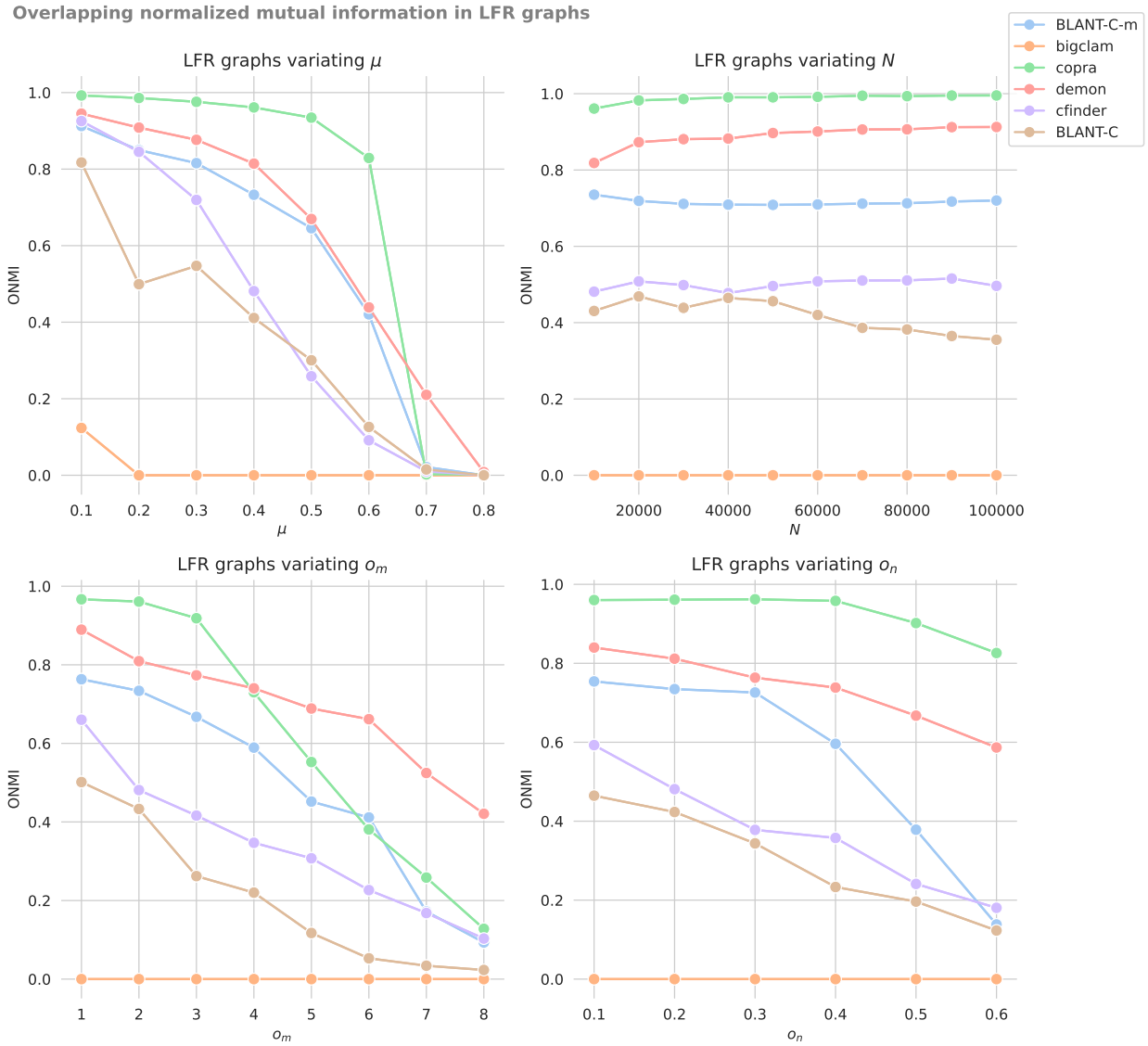


Figure 6.3: Normalized mutual information of each algorithm for each LFR graph in the dataset plus BLANT-C-m, that represents the results BLANT-C would yield if only the communities matching those defined by the LFR algorithm were selected

COPRA consistently achieves remarkable results across the datasets, with the exception of high values of o_m where Demon outperforms it. On the other hand, the results of BLANT-

C, even after optimization, are not as strong, with only BigClam performing worse. This is mainly due to the detection of numerous big and dense communities that are not included in the LFR ground truth, resulting in a lower score for BLANT-C.

However, it is worth noting that the results of BLANT-C do contain several solutions that match the ground truth communities of the LFR graphs. By selectively choosing only those communities that align with the ground truth, BLANT-C's performance is not significantly behind label propagation algorithms.

Furthermore, considering the stochastic nature of BLANT-C and its expansion method, it is believed that by increasing the sampling multiplier and exploring a wide range of edge densities, it is possible to identify all ground truth communities. This suggests that further improvements can be made to enhance the performance of BLANT-C in detecting ground truth communities, thereby closing the gap with label propagation algorithms.

Chapter 7

Conclusions and Next Steps

7.1 Conclusions

In this study, we have introduced BLANT-C, a novel algorithm for overlapping community detection in complex networks. Building upon the BLANT sampling method, BLANT-C incorporates Graphlet First Search expansion to identify large and dense communities. By redefining the notion of community, BLANT-C generates an extensive list of communities and constructs an overlapping graph, which can be further optimized for various applications.

The performance of BLANT-C has been rigorously evaluated and compared against state-of-the-art algorithms in the field. The experimental results demonstrate that BLANT-C surpasses these algorithms in terms of its ability to identify large and dense communities. Moreover, when applied to real-world networks, BLANT-C consistently outperforms competing algorithms in the newly introduced measure $\mathcal{ED}_N$ in overlapping modularity (M^{ov}) for big networks.

Through the comparative analysis conducted on synthetic networks, we have identified im-

portant limitations of BLANT-C. One such limitation pertains to the generation of a lengthy list of communities, which poses challenges in terms of post-processing the algorithm's output. Additionally, there is room for improvement in terms of achieving higher edge density uniformity within the communities. Lastly, enhancing the runtime efficiency of BLANT-C is crucial for its scalability to larger networks.

7.2 Next Steps

The current work lays the foundation for the future development and refinement of BLANT-C. A significant avenue for future research involves further exploring the potential optimality of BLANT-C's community discovery. It is worth noting that, similar to the convergence of simulated annealing towards a globally optimal solution with probability 1 [46], the stochastic nature of BLANT-C, coupled with the Graphlet First Search expansion method, holds the potential to converge towards near-optimal results. This hypothesis suggests that by iteratively refining the community detection process, BLANT-C can progressively approach an optimal solution, effectively capturing the underlying community structure of complex networks. Further research and experimentation will be conducted to validate and explore the convergence properties of BLANT-C.

In terms of algorithmic improvements, the next steps will involve addressing the computational efficiency of BLANT-C. Specifically, efforts will be directed towards refining the time complexity by devising more intelligent exploration strategies. Moreover, the post-processing of results or real-time processing during the algorithm's execution will be enhanced. This will have two main objectives. Firstly, it will facilitate more efficient and effective community identification. Secondly, the aim will be to improve the uniformity of the edge density across the detected communities.

Finally, an additional line of research that has already been initiated involves the creation of benchmark graphs that align with the redefined notion of community presented in this study. These benchmark graphs aim to encompass a diverse range of edge densities, significant overlapping possibilities, and nested communities.

By undertaking these initiatives, BLANT-C will solidify its position as a powerful and efficient algorithm for detecting overlapping communities in complex networks. The advancements achieved through future iterations will contribute to the broader field of community detection and provide researchers with valuable tools for analyzing and understanding the intricate structures present in complex real networks.

Bibliography

[1] Mark S Granovetter. The strength of weak ties. *American journal of sociology*, 78(6):1360–1380, 1973.

[2] Oleksii Kuchaiev, Tijana Milenković, Vesna Memišević, Wayne Hayes, and Nataša Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, 7(50):1341–1354, 2010.

[3] Ernesto Estrada. Graph and network theory in physics. *arXiv preprint arXiv:1302.4378*, 2013.

[4] Olivier Goldschmidt and Dorit S Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of operations research*, 19(1):24–37, 1994.

[5] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[6] Alex Pothen. Graph partitioning algorithms with applications to scientific computing. *ICASE LaRC Interdisciplinary Series in Science and Engineering*, 4:323–368, 1997.

[7] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New journal of physics*, 11(3):033015, 2009.

[8] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[9] Joshua R Tyler, Dennis M Wilkinson, and Bernardo A Huberman. Email as spectroscopy: Automated discovery of community structure within organizations. In *Communities and Technologies: Proceedings of the First International Conference on Communities and Technologies; C&T 2003*, pages 81–96. Springer, 2003.

[10] Matthew J Rattigan, Marc Maier, and David Jensen. Graph clustering with network structure indices. In *Proceedings of the 24th international conference on Machine learning*, pages 783–790, 2007.

[11] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005: 20th International*

Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20, pages 284–293. Springer, 2005.

[12] Jörg Reichardt and Stefan Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. *Physical review letters*, 93(21):218701, 2004.

[13] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004.

[14] Mingming Chen, Konstantin Kuzmin, and Boleslaw K Szymanski. Community detection via maximization of modularity and its variants. *IEEE Transactions on Computational Social Systems*, 1(1):46–65, 2014.

[15] Jian Liu and Tingzhan Liu. Detecting community structure in complex networks using simulated annealing with k-means algorithms. *Physica A: Statistical Mechanics and its Applications*, 389(11):2300–2309, 2010.

[16] Mursel Tasgin, Amac Herdagdelen, and Haluk Bingol. Community detection in complex networks using genetic algorithms. *arXiv preprint arXiv:0711.0491*, 2007.

[17] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[18] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[19] Leon Danon, Albert Diaz-Guilera, and Alex Arenas. The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(11):P11010, 2006.

[20] 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.

[21] Muhammad Aqib Javed, Muhammad Shahzad Younis, Siddique Latif, Junaid Qadir, and Adeel Baig. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, 108:87–111, 2018.

[22] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *nature*, 435(7043):814–818, 2005.

[23] Shawn Mankad and George Michailidis. Structural and functional discovery in dynamic networks with non-negative matrix factorization. *Physical Review E*, 88(4):042812, 2013.

[24] Mahsa Ghorbani, Hamid R Rabiee, and Ali Khodadadi. Bayesian overlapping community detection in dynamic networks. *arXiv preprint arXiv:1605.02288*, 2016.

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

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

[27] Steve Gregory. Finding overlapping communities in networks by label propagation. *New journal of Physics*, 12(10):103018, 2010.

[28] Michele Coscia, Giulio Rossetti, Fosca Giannotti, and Dino Pedreschi. Demon: a local-first discovery method for overlapping communities. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 615–623, 2012.

[29] Anne-Claude Gavin, Markus Bösche, Roland Krause, Paola Grandi, Martina Marzioch, Andreas Bauer, Jörg Schultz, Jens M Rick, Anne-Marie Michon, Cristina-Maria Cruciat, et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415(6868):141–147, 2002.

[30] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Phys. Rev. E*, 84:066122, Dec 2011.

[31] 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.

[32] Anna Lázár, Dániel Abel, and Tamás Vicsek. Modularity measure of networks with overlapping communities. *Europhysics Letters*, 90(1):18001, 2010.

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

[34] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.

[35] Anna D Broido and Aaron Clauset. Scale-free networks are rare. *Nature communications*, 10(1):1017, 2019.

[36] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[37] Aaron F McDaid, Derek Greene, and Neil Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515*, 2011.

[38] Natasa Pržulj, Derek G Corneil, and Igor Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

[39] Wayne Hayes, Kai Sun, and Nataša Pržulj. Graphlet-based measures are suitable for biological network comparison. *Bioinformatics*, 29(4):483–491, 2013.

[40] Darren Davis, Ömer Nebil Yaveroğlu, Noel Malod-Dognin, Aleksandar Stojmirovic, and Nataša Pržulj. Topology-function conservation in protein–protein interaction networks. *Bioinformatics*, 31(10):1632–1639, 2015.

[41] Rafael Espejo, Guillermo Mestre, Fernando Postigo, Sara Lumbreras, Andres Ramos, Tao Huang, and Ettore Bompard. Exploiting graphlet decomposition to explain the structure of complex networks: the ghust framework. *Scientific reports*, 10(1):12884, 2020.

[42] Sridevi Maharaj, Brennan Tracy, and Wayne B Hayes. Blant—fast graphlet sampling tool. *Bioinformatics*, 35(24):5363–5364, 2019.

[43] Adib Hasan, Po-Chien Chung, and Wayne Hayes. Graphettes: Constant-time determination of graphlet and orbit identity including (possibly disconnected) graphlets up to size 8. *PloS one*, 12(8):e0181570, 2017.

[44] Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1–20, 2016.

[45] Vinícius da Fonseca Vieira, Carolina Ribeiro Xavier, and Alexandre Gonçalves Evsukoff. A comparative study of overlapping community detection methods from the perspective of the structural properties. *Applied Network Science*, 5(1):1–42, 2020.

[46] Debasis Mitra, Fabio Romeo, and Alberto Sangiovanni-Vincentelli. Convergence and finite-time behavior of simulated annealing. *Advances in applied probability*, 18(3):747–771, 1986.