UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

A Two-Stage Particle Competition Model for Unbalanced Community Detection in Complex Networks

Luan Vinícius de Carvalho Martins

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Luan Vinícius de Carvalho Martins

A Two-Stage Particle Competition Model for Unbalanced Community Detection in Complex Networks

Dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Zhao Liang

USP – São Carlos
August 2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Martins, Luan
M386t A Two-Stage Particle Competition Model for UnbalancedCommunity Detection in Complex Networks / Luan Martins; orientador Liang Zhao. -- São Carlos, 2020.
75 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2020.

1. Complex Networks. 2. Community Detection. 3. Unbalanced Community. 4. Particle Competition. I. Zhao, Liang, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Luan Vinícius de Carvalho Martins

Competição de Partícula de Dois Passos para Detecção de Comunidades Desbalanceadas em Redes Complexas

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Zhao Liang

USP – São Carlos
Agosto de 2020

ACKNOWLEDGEMENTS

Firstly, I would like to thank my family, especially my parents, for their support and encouragement throughout this journey. Thanks also to my friends who shared this experience with me. Special thanks to Ana Caroline, not only for her companionship during this time but also for having endured all my nonsensical concepts for algorithms and research ideas.

I especially thank the professors Dr. Rafael Rossi and Dr. Ricardo Marcondes Marcacini, who have been with me since graduation at UFMS. They are examples of professionals and inspiration, not only for their knowledge and experience in academic research but also in humility and friendship. Even after my graduation, they continued to contribute with my academic journey, including during this master's, even though our only connection was our friendship. In fact, it was Dr. Marcacini who encourage me to pursue academic research as a field of work.

I also would like to express my deepest gratitude to my advisor, Dr. Liang Zhao, for his patience, help, and friendship during this journey. His experience and support were essential for the execution of this project. I hope to be able to further contribute to your work and ideas.

Finally, I would like to thank ICMC-USP: the teachers and staff. Special thanks to CNPq (National Council for Scientific and Technological Development) for the financial support, which allowed me to work full-time in this project. Although this particular contribution in this field is small, the experience obtained during this master's will be essential as a foundation to enable me to do further research. I hope one day to be able to fully repay the investment by continuously contributing to the development of Brazilian scientific research.

RESUMO

MARTINS, L. V. C. **Competição de Partícula de Dois Passos para Detecção de Comunidades Desbalanceadas em Redes Complexas**. 2020. 79 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

O uso de redes complexas provou ser uma excelente ferramenta para revelar informações de sistemas complexos devido à sua capacidade de descrever relações espaciais, funcionais e topológicas entre os dados. Uma característica inerente às redes complexas, que é uma excelente fonte de informações, é sua estrutura de comunidade–geralmente definida como um conjunto de nós mais densamente conectados entre si do que com outros nós da rede. Para extrair essa informação, diversas técnicas foram propostas. Uma técnica interessante é a Competição de Partícula, que é uma abordagem inpirada de fenômenos da natureza na qual um conjunto de partículas é inserido na rede e deve competir entre si para capturar o maior número possível de nós. A competição, aqui representada como um sistema dinâmico estocástico que controla as partículas, é um comportamento amplamente encontrado na natureza quando há escassez de recursos, como água, alimentos ou parceiros–os vértices do grafo são esses recursos escassos. No entanto, comunidades desbalanceadas são frequentes em redes complexas reais. Embora muitas técnicas de detecção da comunidade tenham sido desenvolvidas e algumas delas possuam um certo grau de tolerância ao diferentes tamanhos de comunidade, ainda falta um mecanismo explícito e eficiente para tratar esse problema. Neste documento, propomos um modelo de Competição de Partículas em Dois Passos para detectar comunidades desbalanceadas. No primeiro estágio, chamado Competição, as partículas competem entre si para ocupar o maior número possível de nós. No segundo estágio, um mecanismo de regularização do tipo difusão é introduzido para determinar o nível de dominância de cada partícula, baseado no grau de dominância da vizinhança de cada nó. As duas etapas executam alternativamente até o processo de regularização convergir. No modelo original da Competição de Partículas, todas as partículas têm o mesmo comportamento; portanto, não há como cada partícula ocupar corretamente as comunidades com diferentes tamanhos ou estruturas. No modelo proposto, o mecanismo de regularização faz com que cada partícula tenha um comportamento diferente de acordo com a estrutura da rede. Consequentemente, comunidades com diferentes tamanhos ou estruturas podem ser corretamente detectadas pelas partículas. Simulações de computador mostram resultados promissores do modelo proposto. Além disso, o mecanismo de regularização melhora a precisão e a velocidade computacional do método, pois menos iterações são necessárias até a convergência, quando comparado aos métodos anteriores de Competição de Partículas.

Palavras-chave: Rede Complexa, Detecção de Comunidade, Comunidades Desbalanceadas, Competição de Partícula.

ABSTRACT

MARTINS, L. V. C. **A Two-Stage Particle Competition Model for Unbalanced Community Detection in Complex Networks**. 2020. 79 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

The usage of Complex Networks has proved to be an excellent tool in reveling prevalent information from complex systems due to its ability to describe spatial, functional, and topological relations among the data. One inherent characteristic of Complex Networks, which is an excellent source of information, is its community structure–commonly defined as a set of nodes more densely connected than to other nodes of the networks. In order to extract this information, many techniques have been proposed. One interesting technique is the Particle Competition method, which is a bio-inspired approach in which a set of particles are inserted into the network and must compete with themselves to capture as many nodes as possible. Competition, here represented as a stochastic dynamic system that controls the particles, is a behavior widely encountered in nature when there is a shortage of resources, such as water, food, or mates—the nodes of the graph are the limited resources each particle must conquer. However, unbalanced communities commonly appear in real complex networks. Although many community detection techniques have been developed, and some of them possess a certain degree of tolerance to unbalance, there is still lacking an explicit and efficient mechanism to treat this problem. In this document, we proposed a Two-Stage Particle Competition model to detect unbalanced communities. At the first stage, named Competition, the particles compete with themselves to occupy as many nodes as possible. At the second stage, a diffusion-like regularization mechanism is introduced to determine the dominance level of each particle at a neighborhood of each node. The two stages work alternatively until the regularization process converges. In the original Particle Competition model, all particles have the same behavior; therefore, there is no way to correctly occupy the communities with different sizes or structures by the particles. In the proposed model, the regularization mechanism makes each particle to have a different behavior according to the network structure. Consequently, communities with different sizes or structures can be correctly detected by the particles. Computer simulations show promising results of the proposed model. Moreover, the regularization mechanism improves both the accuracy and computational speed of the method as fewer iterations are required until convergence when compared to previous Particle Competition methods.

Keywords: Complex Networks, Community Detection, Unbalanced Community, Particle Competition.

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

CONTENTS

CHAPTER

1

INTRODUCTION

Humans have a special ability and necessity to analyze, interpret information and, ultimately, build knowledge. Our search for knowledge has come a long way, and time has proved humankind's everlasting effort in pursuing it. Humanity went from rudimentary drawings on walls to advanced communication through speaking and written language and, in the last century or so, to the development of machines capable of making complex calculations in a matter of seconds, which no average person would be able to accomplish. In the last few decades, to be specific, it has shown a rush towards extracting information and learning knowledge through machines in the so-called machine learning field. The focus of research in machine learning is in creating and studying models for data representation and ever-improving methods to extract information from those models, through tasks such as classification [Wang *et al.* 2020, Silva and Zhao 2012], clustering, regression [Wan *et al.* 2015], pattern recognition.

Although humans and machines obtain, represent, and analyze data in different ways, that does not stop us from taking inspiration from real-life and nature-inspired behaviors when developing new models and techniques for computer intelligence. That is the case of graphs and Complex Networks: a data representation model capable of describing real-world systems and phenomenons effortlessly, such as social networks, the internet [Rowe *et al.* 2007, Simeonovski *et al.* 2017], co-authorship networks [Chuan *et al.* 2018], among others. Graph as a data representation object for machine learning is currently the focus of much research because of its ability to represent data understandably and intuitively to humans. Additionally, graphs have proved to generate great results in many fields and tasks, ranging from sentiment analysis [Wang *et al.* 2018] to 3D DNA folding [Cabreros, Abbe and Tsirigos 2016] – a comprehensive review of applications can is available in [Costa *et al.* 2011].

Graphs work in a very intuitive way: there is a set of nodes, and there is a set of links. Nodes (or vertices) are objects or agents that interact with each other in the system; On the other hand, the edges (or links) represent the connections or interactions between those nodes or agents. In a graph, the edges link the vertices in such a way relevant to the context represented.

The graph is used to extract information since real-world systems are likely present complex behaviors and unique properties such as a high number of nodes and links, presence of hubs, community structure [Boccaletti *et al.* 2006, Newman 2018]. Graphs can reveal the represented system's inherent behaviors and characteristics by its structure.

An essential property of a graph that can reveal relevant information is its' community structure. The concept of a community in graphs has long been the focus of discussion [Costa *et al.* 2007]. Nevertheless, it is wildly referred to as a sub-group of nodes more connected to each other than to other nodes of the graph [Girvan and Newman 2002]. The detection of the community structure has great potential in revealing vital information on the system represented by the Complex Network. Community detection is an NP-complete problem, and there are many community detection algorithms developed so far [Fortunato 2010], each one with their own set of benefits and drawbacks.

One interesting approach to community detection is the bio-inspired Particle Competition model, first introduced in [Quiles *et al.* 2008] and later improved on [Silva and Zhao 2012]. Competition is a natural behavior that occurs when there is a lack of resources such as food, water, or mates. This bio-inspired method works by randomly releasing particles in a network, which then must compete with each other to conquer as many nodes as possible, with the nodes of the network being the limited resource. The particles are random-walkers: in short, means they visit the network node by node making a trajectory.

The Particle Competition model works in the following way: each node has a domination-level that informs which community it belongs, and the particle has an energy level to guide its behavior. Each time a particle visits a node, its domination-level on the node increases, while the domination-level of its rivals decreases, simultaneously the particle's energy level is also altered. In essence, it increases when it visits a node it owns, but it decreases when it attacks a node owned by a rival. The particle with most domination-level is said to own the node. The particle's energy level guides the Competition by forcing the particle to stay in its community, defending it. In essence, if the particle attacked too many nodes belonging to rivals, it gets teleported back to its community, where it must recharge its' energy by visiting and defending its nodes. Finally, each particle occupies a set of nodes to form a community.

To improve the Competition process, each Particle Competition model proposes two ways for the particle to choose the neighboring node to visit. The first movement type is referred to as the Random Walk movement: each neighboring node has the same probability of being chosen. This movement type is essential for good clustering results, as it acts as a novelty finder and helps the particle explore the network and avoid traps. Additionally, each version of the Particle Competition model proposes a second movement type, in which the particle chooses the neighboring node based on the members of the particle's community. The idea is to assert that the particle will remain in its community to defend it from rivals. In the work of [Quiles *et al.* 2008], this biased movement type is known as the Deterministic Walk. In essence, each time the

particle is about to visit a new node, the movement type is chosen using the λ parameter. If the chosen movement type is the Deterministic Walk, the particle is only allowed to visit nodes that already belong to the particle's community. However, the work of [Silva and Zhao 2012] further improved the model by proposing a combination of the two walking types, rather than selecting and only using a movement each time. Such a combination, named Preferential Walk, allows the particle to choose the next node to navigate to freely, but giving preference (i.e., higher chance) to choose nodes it already visited more frequently in the past. The Preferential Walk further improved the model's ability to detect and provide accurate community results.

Both previous iterations of the Particle Competition model rely on knowledge build upon the frequency of visitation each node received up to that moment as a guide. The second movement type uses this guide to incentivize the particle to continue visiting the nodes most likely to belong to its' community. As studied in both papers, the combination of both movement types is essential to obtain good clustering results. However, over time it was found that the random walk alone has a few disadvantages, which can mainly be summarized as a "lack of vision". 1) because each particle behaves the same, the detection of communities of different sizes is nearly impossible; 2) to achieve good clustering results, it is necessary a high amount of iterations for the particle to find, own and defend all member of the community.

1.1 Motivation

The usage of graphs for machine learning tasks already proved to be quite useful [Wang *et al.* 2020, Chuan *et al.* 2018, Simeonovski *et al.* 2017, Javed *et al.* 2020], specially through the detection of communities. However, there is one salient aspect in community detection often overlooked when choosing or even when developing new community detection techniques: the detection of communities of different sizes or densities, i.e., unbalanced community detection. In reality, balanced classes or communities of the same sizes are not a reasonable expectation when representing a real-world system or data sets in graphs [Danon, Díaz-Guilera and Arenas 2006]. Entirely on the contrary, the works of [Guimera *et al.* 2003, Palla *et al.* 2005] found that in reality, real-world networks are likely to have community sizes that follow a power-law distribution.

However, not every community detection method can accurately identify community structures of different sizes. For example, the authors of [Fortunato and Barthelemy 2007] proved many modularity optimization-based techniques might fail to correctly identify communities that are too small when compared to others, related to network size. Such characteristic highlights how overlooked the field might be since the usage of modularity is one of the most famous approaches to community detection [Fortunato 2010]. Indeed, many great techniques (modularity-based or not), including the Particle Competition, are unable to provide accurate results when used in a network with such behavior. Still, unbalanced community detection is an important topic that has received increasing attention due to the complex nature of data in real-world systems.

In order to work around such predicament, researchers started to focus on the development of new methods capable of dealing with such expectations. However, up to now, there still lacks a general and efficient method to detect unbalanced communities in Complex Networks.

Furthermore, the proposed method works by proposing a novelty combination of two well-establish concepts: the sequential propagation of information with Random Walk, through the Competition and parallel propagation of information through the Regularization. Therefore, the proposed method can be extended or modified for other areas, leaving room for future work.

The Particle Competition is based on random walks in networks, where each walker randomly selects a node to visit at each step. Random walk is a stochastic process that has been extensively studied in various disciplines, such as in physics and chemistry, biology, financial economics, image processing, and vision science [W. and N. 1974, G. 1992, H. 1984, Grady 2006, Rucci and Victor 2015]. The approach is also employed in several machine learning methods [Silva and Zhao 2016]. Similarly, parallel propagation of information, also known as just propagation or diffusion, is another extensively studied topic, which can be used to model various natural and artificial processes [Landau and Lifshitz 1980, Pavliotis 2014]. It has also been employed in many machine learning techniques. Examples ranges from community detection [Fortunato 2010, Raghavan, Albert and Kumara 2007, Zhu and Xia 2018] to semi-supervised learning [Culp and Michailidis 2008, Zhou *et al.* 2004] and even deep learning [Wu *et al.* 2019].

The combination of both fields of study proposed in this work may contribute to the further development of graph-based techniques for machine learning.

1.2 Objective

The Particle Competition model can achieve excellent results in both community detection and data-clustering tasks at the same as it has a low computationally complexity. However, because every particle in the system shares the same behavior, the method is prone to fail to identify communities with different sizes correctly. This failure in detecting unbalanced communities happens because every particle in the system has the same ability to attack and defend its community from its peers. Therefore, a particle attempting to defend a larger community will not be able to defend its nodes at the same rate in which a particle from a smaller community can attack them. The project's primary goal is to propose a new mechanism for the Particle Competition model to improve its community detection results on networks with an unbalanced community structure.

1.3 Main Results

In summary, this project augments the Particle Competition model by proposing a new "guide" mechanism based on local observation of the graph, generated from a process we named Regularization. Previously, all Particle Competition models use the number of visits up to that point to bias the particle towards staying in its' community through the deterministic or preferential walking model. In this project, we propose a new Particle Competition model, augmented with a new diffusion-like regularization mechanism that creates a custom guide for each particle to follow. This guide will influence the particle to visit the nodes that are most likely to belong to its community, allowing for the detection of unbalanced communities.

The guide is based on the neighborhood of each node. In essence, if a given node has more neighbors that belong to a rival particle, then this rival particle will have a higher likelihood of visiting this given node. Thus, we create, for each particle, a individual guide with a custom and unique behavior tailored to the community structure the particle is dominating. This mechanism poses a massive functional difference in the proposed method in comparison to previous Particle Competition, as no longer the Preferential Walk can be misled by the particles' equal ability to attack and defend nodes. Rather, the Preferential Walk will adapt to the structure of the network.

Conceptually, we are proposing a model which combines two propagation models: sequential and parallel. The Particle Competition propagates the label information in sequence by visiting the nodes one by one in a detailed way and provides a chance to correct errors. In contrast, the parallel propagation can quickly share the label information to the nodes, but it loses the self-correcting ability. The proposed model combines both models to better understand the community structure of the network.

The usage of this guide, build from the particle's community and its neighborhood, provides some advantages to the proposed model when compared to both versions of the Particle Competition model [Quiles *et al.* 2008, Silva and Zhao 2012, Gao *et al.* 2019], mainly:

1. **Unbalanced Community Detection**: the diffusion-like guide builds upon the direct neighbors of each node and allows each particle to have different behavior, per the network's topology. Such behavior makes it possible for each particle to identify communities of different sizes, regardless of the level of unbalance in the communities of the network.

2. **Improved detection of nodes sharing links with other communities**: in the original Particle Competition model, the domination-level of a node (which is responsible for its' label) only changes when it gets visited by a particle. As a result, nodes sharing links to other communities frequently gets visits from rivals, which may affect the accuracy of the results in larger networks, due to the large number of iterations required to defend those nodes properly. The diffusion-like regularization model asserts to only guide the particle to visit nodes that it has the majority of its neighbors. Therefore, nodes whose

links are shared between different communities are more likely to be correctly clustered in the proposed model.

3. **Improved time efficiency of the model**: the Particle Competition model can only propagate labels node by node, which means that it needs to visit every node of network at least once to classify it. Moreover, due to the stochastic nature of the method, the model is prone to initially misclassify nodes during the initial iterations, until each particle roughly settles in a community. Therefore, the correct identification of the communities and eventual correction of misclassified nodes requires Competition to occur, which demands a large number of iterations, especially in larger networks. The proposed diffusion-like regularization function improves the method on both aspects. In essence, no longer the Particle Competition is required to run enough iterations to visit every node of the network. When the Regularization mechanism propagates the labels through the network, nodes that have not yet received visits are automatically labeled. Additionally, the proposed regularization mechanism thrives in fixing eventual misclassifications, which are certain to occur. Therefore, the proposed method can handle much larger networks with drastically fewer iterations. At the same time, it maintains the same computational complexity order of the previous Particle Competition models.

In this document, we propose the new Two-Stage Particle Competition method for the Particle Competition family of algorithms. The method has two steps. The first is the Competition itself, which is when the particles will roam the network, trying to conquer as many nodes as possible. The second step is the diffusion-like regularization mechanism, which will generate a custom guide for each particle, tailored for its community's topology. Both stages learn and regulate each other until convergence. The guide is built upon the topology of the particle's community, allowing each particle to have different behavior, making possible the detection of communities of different sizes.

1.4 Document organization

The organization of this document is as follows: firstly, a quick introduction of the main concepts, ideas, and results is given in this chapter. Next, in Chapter 2, we shall dive into details into the main topics of the document: *complex networks*, *community detection*, and the *Particle Competition* algorithm. Then, in Chapter 3, the proposed method for unbalanced community detection is described in detail. Chapter 4 contains an empirical analysis of the proposed model, which illustrates how it works and analyzes how it behaves under different circumstances and different parameters. Finally, in Chapter 5, the proposed method is applied to real-world and artificial community detection and data-clustering tasks. The last chapter, Chapter 6, concludes this document and discusses future works regarding the proposed method.

CHAPTER

2

FUNDAMENTAL CONCEPTS

In this section, essential concepts related to this project are introduced, such as the history, models, and techniques used on graph-based machine learning. Models for community detection related to this project are also revised.

2.1 Complex Networks

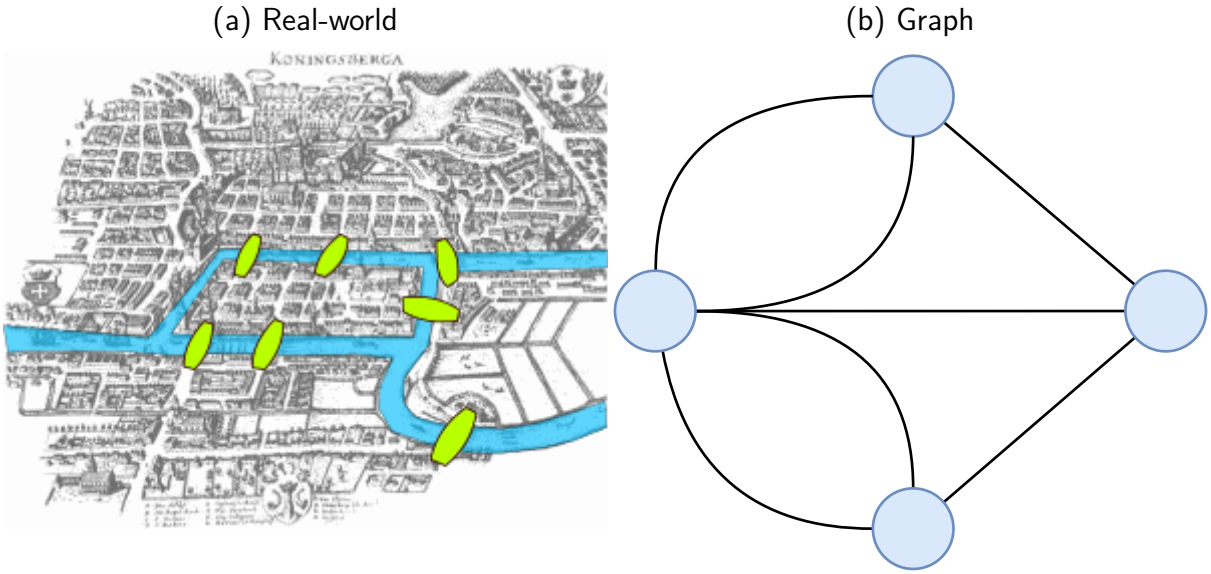

Figure 1 – Graph representation from real-world scenarios. (a) Visual representation of the Seven Bridges of Königsberg [1]. (b) Graph representation of the Seven Bridges of Königsberg problem.

Graph was first proposed as an analytical solution to the famous "Seven Bridges of Königsberg" problem. The problem was to determine whether or not it was possible to navigate

[1] By Bogdan Giuşcă - Public domain (PD), based on the image, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=112920>

through the seven bridges that existed back then in Königsberg (now Kaliningrad) and go back to the starting point without ever crossing a bridge more than once. It was Leonhard Euler in 1736 who proved such a task was not possible by representing such a system as a graph, with the locations as the vertices and the bridges as the edges, representing the bridges between them. This structure, which is the base of a graph object and is still in use today, marked the start of Graph Theory and the study of Topology [Shields 2012]. The following years showed an increase in the research and new developments for graphs in terms of theory and applications. Although research started in the mathematics field, it eventually expanded to other areas such as computer science, physics, linguistics, as indeed, there are many applications for graphs [Fortunato 2010].

Formally a graph is represented as tuple $G = (V, e)$, in which $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes or agents in the system and $e = \{e_1, e_2, ..., e_n\}$ is the edges or links that connect those agents. However, in order to allow for even more robust and direct representation of real-world phenomenons without losing information, new graphs structures were proposed that extend this basic definition. Some specific definition of graphs is as follows:

- **Directed Graph**: A directed graph can better represent a system in which a link between two nodes is not symmetric. A famous example is when representing social networks, in which a person may follow someone that does not follow them back. An intuitive definition is when the link between node v_i and node v_j is different from the link between node v_j to node v_i, i.e., $(v_i, v_j) \neq (v_j, v_i)$.
- **Weighted Graph**: Weighted graphs allows additional information in the form of weight in the links connecting nodes. The weight between the nodes can be used by machine learning algorithms to better extract information from the represented system. Formally, the edges of a weighted graph has the following structure: $e = \{(v_i, v_j, w_1)\}$.

There are different approaches to represent a graph in a computer. The most common one is by using a square adjacency matrix A, with dimension $\dim(A) = |V| \times |V|$. Each element A_{ij} is used to represent a possible edge in the graph connecting the nodes v_i and v_j, i.e., $A_{ij} > 0$ when such link exists, otherwise $A_{ij} = 0$. Therefore, the matrix A needed to constitute an undirected and unweighted graph is symmetrical, containing only 1s and 0s to represent the links. In the case of a weighted graph, A_{ij} can represent the weight of the link between nodes v_i and v_j. Many relevant properties of the graph may be extracted using such representation. For example: to obtain the degree of a node, one must sum the row which belongs to it.

The distinction between a graph and Complex Networks has also been the focus of discussion. Some authors define graphs as a mathematical representation of a network [Newman 2018]. Others define networks as a natural representation of a complex system and graphs as a human-made representation [Mihalcea and Radev 2011]. However, the most used definition is that networks are graphs with non-trivial topology, often representing systems with hubs and community structure [Mihalcea and Radev 2011, Newman 2018]. Networks frequently exhibit

a high number of nodes and edges with a diversity of connections, in the form of weights and directions.

In machine learning, the traditional model for data representation is vector-based, which represents data through instances and their attributes. Therefore, in order to take advantage of methods and machine learning approaches available for graphs, an important topic is how to translate a real-world data set, originally a vector-based representation, as a graph for machine learning tasks. There are many approaches to accomplish such a task, as this is a relevant topic that is receiving increasingly more attention. Some of the simplest approaches to such task are:

- **K-Nearest Neighbor Graph**: one of the most famous approaches is to generate a graph by connecting each item of the data set to their K most similar data point. Initially, each data point of the data set is a node in the graph. Then, using a similarity measure, all elements get connected to their first K most similar data point from the data set.
- **ϵ-Radius Graph**: another approach to generate to consider each data point as a node in the graph. Each node gets linked to its most similar neighbors, who are inside the radius of e. This approach can generate densely connected communities, at the risk of generating too many edges, as K is sensitive both to the applied similarity function, as well as the data set.
- **Mutual K-Nearest Neighbor Graph**: [Brito *et al.* 1997] proposes a modification to the K-Nearest Neighbor graph. This approach only connects nodes (data points) that are both in their k most similar neighbors.

2.2 Community Detection

One inherent aspect of a Complex Network is the presence of a community structure, which is an essential source of information explored in many works [Javed *et al.* 2020, Kawasaki and Ikeda 2020]. Indeed, in the context of a society, the members of a community can be defined as a group of invidious that shares common characteristics, interests, or beliefs, often because of religion, culture, or historical heritage. That is the type of information or insight a community can provide: which members of the system share the same behavior, function, or play similar roles in the overall system. This information can and have been exploited to understand many real-world scenarios, for example, in amazon.com data set, energy point of failure example, and molecular structure [Guimera and Amaral 2005]. Additionally, community detection through graphs proved to be an advantageous model for data-clustering tasks of real-world data sets, once the data get represented as a graph [Silva and Zhao 2012, Javed *et al.* 2020, Kawasaki and Ikeda 2020].

The definition of a community in a Complex Network is simple. In essence, a community is a sub-group of nodes more densely connected to themselves than to nodes of other communities

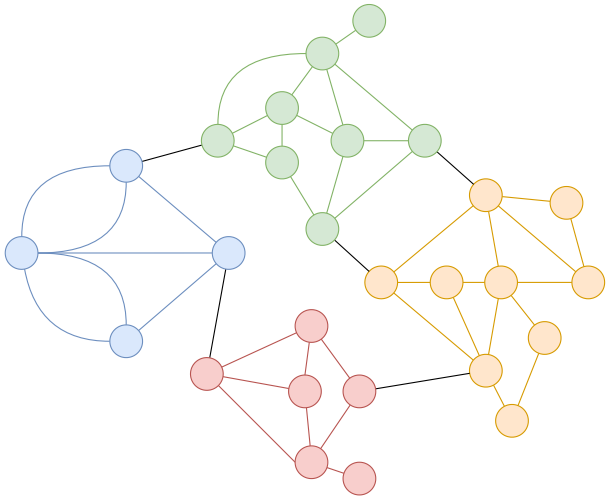

Figure 2 – Visual example of a community: a sub-set of nodes more densely connected with each other than to other nodes of the graph.

[Girvan and Newman 2002]. Due to the complex nature of real-world data, graphs may display very complex behavior in the form of their structure, including in its community. Two prevalent examples of complex behaviors found in the communities of a Complex Network are: they may be unbalanced, i.e., have strikingly different sizes or exhibit a hierarchical structure in which communities are composed of smaller sub-communities.

Community detection is somewhat similar to graph partition, which is an NP-Hard problem. Throughout the last two decades, however, many authors proposed different approximate and efficient heuristic methods to this task [Fortunato 2010].

2.3 Community Detection Algorithms

Many community detection methods have been developed [Fortunato 2010]. However, most of them do not provide an explicit mechanism to detect unbalanced communities, which is a phenomenon widely observed in real-world applications. In this section, we provide a quick review of some community detection approaches and methods.

2.3.1 Divisive Based Methods

Divisive methods are one of the most classic approaches to community detection. Those are methods that increasingly divide the members of a network by removing the edges which connect different communities following given criteria. The idea is that, following this criteria, one may identify what members belong in the same community.

One of the earliest proposals in this field is in [Girvan and Newman 2002, Newman and Girvan 2004], as it not only defined an excellent approach for community detection but also marked the beginning of a new era in the field by highlighting the importance of studying

the community structure in a network and introducing the field to physicists. The method introduces one of the most straightforward approaches to divisive methods. In essence, using the *edge centrality* measure, the authors propose the detection of the community structure in the network by removing the edges which connect different communities, highlighted by higher edge betweenness values. In practice, the *edge centrality* measures how important an edge is when flowing information through the network. Propose we need to exchange information between two communities of a given network. By sending the information node by node, the edges which connect different communities would be highly used, as there are not many choices of edges when passing the information outside the community in comparison to inside the community. The algorithm proposed in [Girvan and Newman 2002, Newman and Girvan 2004] consists of calculating the edge centrality measure for all links in the network. Then, the link with the highest value gets removed from the network. The process is restarted by calculating the centrality measure for all links again and repeating the process.

In [Tyler, Wilkinson and Huberman 2005], the authors improve the original GN method in terms of computational speed, allowing it to be employed in more massive graphs. Additionally, the model can detect overlapping communities, which is the identification of nodes that belong to more than one community at a time. The idea is only to calculate the *edge betweenness* measure to a set of a limited number of centers in the graph, which are picked at random following a Monte Carlo estimate. The lower number of calculations assures a faster computational speed. This approach also allows identifying the degree of overlapping between communities each node has. The method proposed in [Rattigan, Maier and Jensen 2007] is also a faster version of the GN method; this approach uses a *network structure index*, which is a set of node annotations combined with a distance measure [Rattigan, Maier and Jensen 2006]. This approach consists in approximating the *edge betweenness* by dividing the network into regions and computing the distances of every node from each region using the distance measure. Such an approach has a lower complexity in comparison to the original GN algorithm.

Another highly relevant method worth to be explained in detail is presented in the recent work of [Žalik and Žalik 2018], where the authors propose a model for detecting unbalanced communities in networks by combining a set of different network measures. The model's most interesting aspect is to identify the communities one by one, which supposedly assures the correct identification of every community regardless of its size or density. The detection of the local community uses a set of measures that analyses the topology of the network. Important measures used are:

$$S_{ij} = A_{ij}.A_i^T.A_j \tag{2.1}$$

$$simdeg_i = \sum_{j \in G} S_{ij} \tag{2.2}$$

$$simC_{i,l} = S_i^T.C_l \tag{2.3}$$

$$simC_l = \sum_{i \in C_l} simC_{i,l} \tag{2.4}$$

$$simC_i^{max} = \max(simC_{i,l'}), l' \in P \tag{2.5}$$

$$simC_{i,l} = simC_i^{max} \tag{2.6}$$

The first equation (Eq. 2.1) quantifies the *similarity* between two given nodes (i and j): their similarity increases proportionally to the number of common nodes they share connections to. Equation 2.2 ($simdeg_i$) is referred to as the *similarity degree* of node i, which is the sum of *similarities* (Eq. 2.1) of all direct neighbors of node i. The next set of measures compares nodes to communities: Eq. 2.3 ($simC_{i,l}$), for example, quantifies the *similarity* of the node i to the members of the community C_l. Equation 2.5 extends the previous equation for comparison to other communities detected ($l' \in P$, and P is the set of communities), this equation is important because it allows comparing how coherent it is for a given node i to be in part of the community l, which is what the Eq. 2.6 yields.

$$simdeg_{i,j} = \frac{S_{i,j}}{deg_i} \tag{2.7}$$

$$PV_l = \{i | simdeg_{i,j} > \alpha \text{ AND } simdeg_{j,i} > \beta, i \notin l, j \in l\} \tag{2.8}$$

The authors propose the following algorithm: nodes currently not in a community are in a set of *free nodes*. Each iteration, the node with the highest *similarity degree* (Eq. 2.2) is chosen from this set as the starting point for the local community identification. From this node, a new community l will be uncovered, starting by the node's neighbors. For that, we use the set PV_l of candidate nodes. It defines a threshold that limits what nodes may get added to the community currently in creation. Only nodes that are not yet part of the community l, but that shares a $simdeg_{i,j}$ (Eq. 2.7) higher than α to any node j of the community, may be added to the current community. This criterion assures only nodes in this threshold can be added to the community, therefore, allowing to detect only one community at each time. Additionally, by controlling α and β, the user may set the behavior in which communities are discovered.

The candidate nodes available in PV_l are added to the current community if they are strongly connected to the community as per Eq. 2.6, extended with all possible nodes of the community l (Eq. 2.3). At which point, the next community must be uncovered, and the process restarts again by selecting the next node with the highest *similarity degree* from the *free nodes* set. The process repeats until there are no more nodes in the *free nodes* set.

$$coupling_{l,u} = \sum_{i \in l, j \in u} S_{i,j} \tag{2.9}$$

$$coupling_l^{max} = \max(coupling_{i,u}), u \in P; \tag{2.10}$$

$$QM = \sum_{l=1}^{k} \frac{e_{ll} - 2 * coupling_l^{max}}{\alpha_l}; e_{ll} = simC_l; \alpha_l = \sum_{i \in l} simdeg_i \tag{2.11}$$

$$\Delta QM = \frac{e_{ll} - 2 * coupling_l^{max}}{\alpha_l} - (\frac{e_{ii} - 2 * coupling_i^{max}}{\alpha_i} - \frac{e_{jj} - 2 * coupling_j^{max}}{\alpha_j}) \tag{2.12}$$

The communities identified so far are called "preliminary communities", they can be unstable and as such, must be joined together to form a stable community. For this task, the authors define Eq. 2.12, which is a quality function of *stability*, used when joining two preliminary communities C_i and C_j to form a new stable community C_l. Equation 2.12 compares the input communities using measures already defined, such as $simC_l$ (Eq. 2.4), $simdeg_i$ (Eq. 2.2) and $coupling_{l,u}$ (Eq. 2.9), which quantifies how separable the communities are. Each community is merged together with its neighbor as to maximize and increase the quality function ΔQM (Eq. 2.12).

However, the selection of reasonable values for the parameters α and β adds to the complexity of the method, as it is unclear what values are appropriate for each network. Additionally, because of the selection of specific network measures, the ability of unbalanced community detection may be restricted.

2.3.2 Modularity-Based Methods

The modularity equation was first proposed by [Newman and Girvan 2004] with the simple goal of providing a measure of how modular the network is, and as such, to qualify the results obtained by community detection methods. The Eq. 2.13 presents the modularity function.

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \gamma \frac{k_i k_j}{2m}) \delta(g_j, g_j) \tag{2.13}$$

To better define the modularity function, consider the input Complex Network as an adjacency matrix A. The modularity function contains the following elements: firstly, A_{ij} are the elements of the graph matrix A, m is the sum of all weights in the network, k_i is the weight of the i vertex, as k_j is the weight of the j vertex. Finally, γ is a parameter to define the scale of the community detection: smaller values for γ generates a higher number of communities. The modularity function uses the concept of a null model network, which is a synthetic network

whose edges are created at random and, therefore, should display no community structure. Q exploits this concept by comparing the original network against the null model: given a network and a community structure, Q returns a close to 1 when the structure is highly modular, indicating the existence of a community structure. Otherwise, Q returns a value close to 0, indicating the given community structure is no more modular than that of the null model.

Later, the discovery of modularity leads to a new set of methods that exploits the value as a means to obtain accurate community detection results. By using the measure as an indicator of the quality of a community division, one can identify communities by identifying which set of communities or divisions of the graph maximizes this value. The challenge, however, becomes a matter of identifying the best way to select and generate such community division, as arranging V nodes into different sets of possible communities is very costly [Newman 2004], and it is indeed an NP-Complete problem [Brandes *et al.* 2007]. In light of this, in the same year, the work of [Newman 2004] proposed a new community detection technique based on modularity optimization through a standard greedy optimization algorithm, capable of providing fast and reliable results. Exploiting the same idea, [Blondel *et al.* 2008] proposes a method able to handle massive networks with modularity optimization. It proposes a new and faster way to compute the modularity, which makes it feasible to decide which community each node should be by change each node's community to increase its value.

Modularity was exploited in a wide range of ways for community identification. Besides the Greedy optimization techniques, some authors also proposed the optimization of modularity through simulated annealing [Kirkpatrick, Gelatt and Vecchi 1983] – an approach that manages to obtain highly accurate results at the cost of high computational complexity. The work of [Guimera, Sales-Pardo and Amaral 2004] proposes the usage of two moves for such an optimization: local moves, in which only one node gets changed from a cluster to another; global moves, which can split or join different communities. The two-movement types generate different community structures, which are validated and improved with the simulated annealing algorithm. However, although this approach can obtain accurate results, it has a low computational speed. In order to obtain good computational speed, some authors also proposed the usage of extremal optimization through a heuristic search [Duch and Arenas 2005]. As a fitness function to be optimized, the method proposes the usage of local modularity divided by the node's degree. The method starts by randomly dividing the graph into two communities. Then, on each iteration, the node with the lowest fitness is moved from its community to the other. This step gets repeated until the modularity value does not change anymore. In order to detect more than two communities in a network, the process gets repeated, considering each generated community as a graph itself. This approach manages to obtain accurate results and maintains good computational speed.

Most recently, researchers also studied the ability to detect communities using new approaches for optimization. Examples of such work are on [Yang *et al.* 2016], which can identify the best combination of communities by combining the optimization of modularity with

a stochastic block model using deep learning. The work of [Zhou *et al.* 2019] employs a new global optimization method named Discrete State Transition Algorithm. The idea is to find the state (a possible configuration of the community) to maximize the fitness function given by the modularity value. The authors define two operations to change such a state: the first is the Vertex Substitute Transformation, which changes the community of k selected nodes. The selection of the k nodes is made using the neighborhood of the node: if most neighbors belong to another community, this node has a higher chance to be selected. The other is the Community Substitute Transformation, which changes the label of all nodes of the same community. Once again, the criteria used is the neighborhood of the community. In essence, if the neighborhood is mostly composed of different labels, the community has a higher chance of being selected.

Focusing on the resolution problem modularity-based methods faces, [Zhang and Zhao 2012] proposed a new community detection technique that focuses on working around the problems in modularity-optimization, as to make it suitable for unbalanced community detection. The model focuses on community detection through the leading eigenvector model [Newman 2006], which is unable to correctly identify communities when they have drastically different degrees: a common occurrence in unbalanced networks. First, the authors propose the identification of the number of communities presented in the network. Such a task is accomplished through a modification of a graph partition technique the authors also propose. The first step of the method is to generate the $2A - D$ matrix, with A being the adjacency matrix representing an unweighted graph and D being a diagonal matrix containing the degree of each one of the vertex in the graph. Such a matrix exposes the idea of modularity: it is an approximation of which vertice's degree most differs from that of a network with no community structure. With the number of communities K of the network, the model proposes the identification of the largest K eigenvector of the $2A - D$ matrix. Such result is then used to generate a new matrix T, with dimension $dim(T) = |V| \times K$. The K-means clustering method is then applied to the newly created T matrix, identifying the communities of the network. By clustering the vertices using the combination of the K largest eigenvectors and the K-means method, the authors manage to bypass the resolution limit, by treating each community separately. The downside of the approach is that the method has a somewhat high complexity order. Additionally, the authors themselves commented that the leading eigenvector model has several limitations when dealing with larger networks with a high number of communities, and it is not clear either this approach improves on such limitations as well.

2.3.3 Label Propagation-Based Methods

Another approach to community detection is through a process named propagation. The work of [Bagrow and Bollt 2005] was one of the earliest approaches to community detection using this concept. The idea is to answer the question, "how would a person who just moved to a new city identify what community they belong to?". [Bagrow and Bollt 2005] answered by

proposing a new *l-shell* algorithm for local community detection, which propagates a label it received outwards, to their neighbors. The process repeats until the number of edges proceeding out of the community drops below a threshold value. In order to extend this concept to detect all communities in the network, the method requires the creation of a matrix with $|V| \times |V|$ dimension. In this matrix, the ijth element is equal to 1 if they belong to the same community and 0 otherwise. Then, each row of the community gets rearranged to be next to each other based on their similarity. The process gets repeated until the distance of a row to the next is higher than a given threshold, identifying, then, which elements belong to the same community. The process repeats until there are no more communities in the network, which makes it computationally expensive.

One of the most famous approaches to community detection using such a concept is introduced in [Raghavan, Albert and Kumara 2007]. The authors propose a fast method for community detection, able to handle large networks. The method works in two steps; the first step is to assign a new random label for every node of the network. The second step is the propagation procedure, in which every node gets assigned the label whose most neighbors share. During the propagation procedure, ties will happen when choosing what label will be propagated to the current node: the method deals with such a situation by breaking the ties randomly. One of the advantages to such an approach is that not only the method has a low complexity order, making it able to handle larger networks, but it is also able to detect unbalanced community structure to some degree.

In order to improve the stability of results and increase the accuracy, a set of techniques that attempts to improve on the label propagation algorithm were proposed. For example, [Zhu and Xia 2018] proposes the usage of an Adaptive H-index value to rank the nodes of the network in an attempt to set an optimal order for propagation. By propagation the neighborhood information in the order given by proposed measure, the method can provide more stable results and even increase accuracy in some cases.

CHAPTER

3

A TWO-STAGE PARTICLE COMPETITION MODEL

In this section, we introduce and describe in detail the proposed model for unbalanced community detection through the combination of two stages that learns and regulates each other. The model built upon the foundation of the Particle Competition family of algorithms. In all Particle Competition models, a set of particles are randomly inserted in the network to compete with each other to capture as many nodes as possible for its community. We start by revising the first Particle Competition method, used as the first stage of the proposed model almost unchanged. Next, we introduce the second step and describe how both stages interact with each other. Then, we address the initialization and termination criteria strategies for both stages. Finally, we present the full model as an algorithm and discuss its computational complexity and parameters.

3.1 Particle Competition

The Particle Competition model was first proposed by [Quiles *et al.* 2008], which presented a procedure of the method without a formal definition. Later, [Silva and Zhao 2012] improved the Particle Competition method by proposing a new paradigm for particle exploration and modeling the behavior of the particles as a stochastic dynamical system. The model presented in this work builds upon the work of [Silva and Zhao 2012]. Here we briefly review the Particle Competition model proposed initially in [Quiles *et al.* 2008, Silva and Zhao 2012, F.A.N. and L. 2018], which will contribute to the conceptual understanding and later, to develop the Two-Stage Particle Competition model for detecting unbalanced network communities.

Consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ where $\mathcal{V} = \{v_1, ..., v_V\}$ is a set of nodes (or vertices) and $\mathcal{E}$ is a set of links $\mathcal{E} = \{e_1, ..., e_L\}$. The graph object is represented by an adjacency matrix A, with $\dim(A) = V \times V$ and $a_{ij} > 0$ indicates a edge connecting node i to node j.

The Competition step of the proposed system uses a set of particles $\mathcal{K} = \{1, ..., K\}$,

which will explore the network and compete with each other to conquer as many the vertex of the graph as possible. Each particle carries both a flag to label the nodes and an energy level to guide the exploration process. When a particle visits a node, its domination level on that node strengthens while the domination level of its rival weakens. In order for a particle to label a node of the network, it must become its owner by having the highest domination level on it. When a particle visits a node it owns, its energy level increases, on the other hand, it decreases when the particle invades a node belonging to a rival particle.

The particle's energy level guides the Competition by discouraging the particle from wondering around rival communities for too long. When a particle is *exhausted*, i.e., its energy level is too low, the particle teleports back to a node of its community to recharge the energy and, consequently, defend its community from rival particles. Otherwise, when the particle is *active*, it is free to explore the network using a combination of two movements: Random Walk and Preferential Walk.

The Random Walking movement type, in which the particle will randomly choose a neighbor node to visit, is responsible for the discovery of new territories and its ability to attack and conquer new nodes. On the other hand, Preferential Walk guides the particle to choose the nodes that are most likely to belong to that particle. As a result, it is responsible for guiding the particles into settling in a community and defend them from rivals.

The particle competition is a stochastic dynamical system, which is given by

$$p^{(k)}(t+1) = j, j \sim \mathbb{P}^{(k)}_{\text{transition}}(t) \tag{3.1}$$

$$N_i^{(k)}(t+1) = N_i^{(k)}(t) + \mathbb{1}_{\left[p^{(k)}(t+1)=i\right]} \tag{3.2}$$

$$E^{(k)}(t+1) = \begin{cases} \min\left(w_{max}, E^{(k)}(t) + \Delta,\right. \\ \text{if owner(k, i)} \\ \max\left(w_{min}, E^{(k)}(t) - \Delta,\right. \\ \text{if } \neg \text{ owner(k, i)} \end{cases} \tag{3.3}$$

$$S^{(k)}(t) = \mathbb{1}_{\left[E^{(k)}(t)=w_{min}\right]} \tag{3.4}$$

$$\tag{3.5}$$

In Eq. 3.1, $p^{(k)}(t+1)$ represents the network position (node) of particle k at iteration $t+1$. This equation controls which particle is visiting which vertex of the graph. This is done using the time-varying transition matrix $\mathbb{P}^{(k)}_{\text{transition}}(t)$, explained in detail in Eq. 3.8. Equation 3.2 counts the amount of times that each particle k has visited each node i up to the time t, in other words, it updates the ownership of the particles on the nodes ($N_i^{(k)}(t+1)$). Equation 3.3 updates the energy level of each particle $E^{(k)}(t)$: it increases by Δ when visiting a node that the particle owns, otherwise it decreases by Δ if the particle is invading a rival's territory. The last equation

updates and keeps track of the state of each particle. When the energy level is decreased to the minimal level i.e., $S^{(k)}(t) = 1$, its status is *exhausted* and it should be randomly reset at another node of the network; otherwise, the particle is at *active* status and walks normally in the network.

$$\mathbb{P}^{(k)}_{\text{rean}}(i,j,t) = \frac{\mathbb{1}\left[\arg\max_{m \in K}(N_j^{(m)}(t)) = k\right]}{\sum_{u=1}^{V} \mathbb{1}\left[\arg\max_{m \in K} N_j^{(m)}(t) = k\right]} \tag{3.6}$$

$$\mathbb{P}^{(k)}_{\text{rand}}(i,j) = \frac{a_{ij}}{\sum_{u-1}^{V} a_{iu}} \tag{3.7}$$

Equations 3.6 and 3.7 controls two of three prevalent walking behaviors of the particles. Equation 3.6 is responsible for teleporting the particle directly to a node it owns. In essence, it works by creating a transition matrix with an equal chance of movement for each node in which the current particle (k) has the highest domination-level. The operation $\mathbb{1}_{[.]}$ returns 1 if the logical criteria in $[.]$ is met and 0 otherwise. Equation 3.7 controls the Random Walk policy of the particles. This equation is given by the weight or existence of a link between the current node and its neighborhood. In essence, $\mathbb{P}^{(k)}_{\text{rand}}$ results in a transition matrix that allows the particle to chose and visit a neighbor of the current node: the particle has a higher likelihood of choosing the neighbor with the highest weight on a weighted Complex Network.

Finally, the transition matrix $\mathbb{P}^{(k)}_{\text{transition}}(t)$, which governs the behavior of the particle, is presented:

$$\begin{aligned}\mathbb{P}^{(k)}_{\text{transition}}(t) = (1 - S^{(k)}(t))\left[\lambda \mathbb{P}^{(k)}_{\text{pref}} + (1-\lambda)\mathbb{P}^{(k)}_{\text{rand}}\right] \\ + S_{(k)}(t)\mathbb{P}^{(k)}_{\text{rean}}(t)\end{aligned} \tag{3.8}$$

In summary, Eq. 3.8 works in the following way: when a given particle's status is *exhausted*, i.e., $S^{(k)}(t) = 1$, the movement policy switches to a defensive strategy using the $\mathbb{P}^{(k)}_{\text{rean}}$ transition matrix, which will return the particle to a node it owns to recharge its' energy and defend its' community. Otherwise, when the particle is *active*, the movement policy will use a combination of random walking, using the transition matrix $\mathbb{P}^{(k)}_{\text{rand}}$, and preferential walking, given by the $\mathbb{P}^{(k)}_{\text{pref}}$ transition matrix, which we will introduce in the next section. The λ parameter, $0 \leq \lambda \leq 1$, indicates the emphasis on the preferential walking policy, which keeps the particle defending the nodes it is most likely to belong to the particle's community.

3.2 The Two-Stage Particle Competition Model

In the original Particle Competition models, the particle's only bias is to defend the nodes it already conquered. While this is a valid and desirable behavior as proved in [Quiles *et*

al. 2008, Silva and Zhao 2012], it does not take into consideration the topology of the network but rather, it uses its own previous performance and exploration of the network. Indeed, the usage of only its previous exploration of the network is the Particle's downfall when dealing with unbalanced communities, as the sequential nature of Random Walks is unable to extract such a complex information from network by itself. On the other hand, the Regularization stage employs a parallel mechanism to diffuse the label information in the network. Such a mechanism, can indeed better understand the topology of the community. Therefore, the combination of both mechanisms, sequential and parallel, can improve the community detection results and allow for the correct detection of unbalanced communities.

In order to help the Competition step with the preferential walk, the Regularization step uses the knowledge obtained from the Competition stage to generate a guide for each particle. This guide helps the particles to visit the nodes most likely to belong to its community on the next *epoch* (τ) by using the labels propagated to the neighbors of each node in the previous Competition step. In brief, if a particle owns most of the neighbors of a given node, this node will have a higher likelihood of belonging to that particle than to any other, which in turn will force the particle to defend it on the next *epoch*, i.e., the next iteration of both stages.

Both steps learn and regulate each other. The Competition guides the Regularization, which in turn influences the Competition. Once both steps runs, an *epoch* – denoted by τ – passes, and the process restarts. In a broad horizon of time, the preferences of the particle (given by the Regularization guide) no longer changes, uncovering the community structure in the network.

3.2.1 The Proposed Regularization Stage Mechanism

The Regularization step's main goal is to generate the matrix $B(\tau)$, which defines the preference level each particle upon every vertex of the graph. Moreover, this preference matrix will act as a guide for the next iteration of Competition and will ultimately contain the community structure of the network. For this purpose, we define the regularization function as:

$$R(i,k,M) = \frac{\sum_{u=1}^{V} a_{iu} * M_{uk}}{\sum_{j=1}^{K} \sum_{u=1}^{V} a_{iu} * M_{uj}} \tag{3.9}$$

Eq. 3.9 yields the likelihood of the node i to belong to the particle k by taking into account the preference levels on the input matrix M, with dim(M)$= V \times K$.

Notice that in Eq. 3.9, M has not been defined yet. That is because M is a placeholder for either the matrix $N(\tau)$ or $B(\tau)$: for example, when the Regularization step first starts, Eq. 3.9 is called with $M = N(\tau)$, therefore, it takes into consideration the domination levels from the Competition (which is given by $N(\tau)$). By multiplying the weights from all links from node i (A_{iu}) by the domination matrix (when $M = N(\tau)$), Eq. 3.9 produces a number close to 1 when most neighbors of i belongs to the community k (indicating a high likelihood of the node i

belonging to the particle k); otherwise, Eq. 3.9 returns a low preference if the particle k doesn't have high domination levels in node i's neighbors.

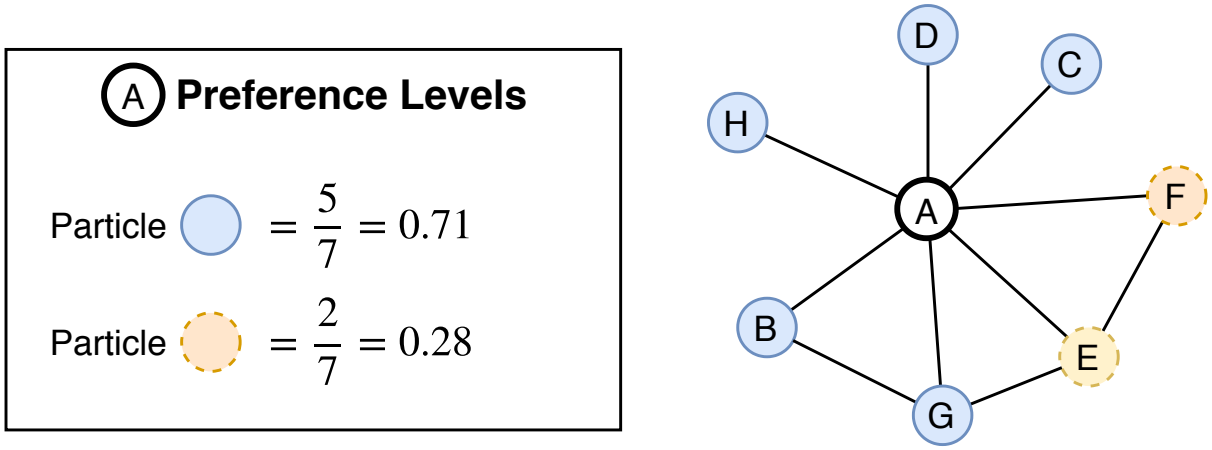

Figure 3 – Example of the effect of Eq. 3.9 on the preference levels of a node. Suppose the Competition step just ended and the Regularization step starts with $M = N(\tau)$: the preference level of A takes into consideration its direct neighbors: as most neighbors belong to the "blue particle", this particle will have a higher likelihood of defending it on the future.

With Eq. 3.9, the current step of the system holds the preference levels of each one of K particle in the system for the node i up to epoch τ using the vector:

$$B_i(\tau) = [B^{(1)}(\tau), B^{(2)}(\tau), ..., B^{(K)}(\tau)] \tag{3.10}$$

with dim($B_i(\tau)$)$= 1 \times K$. Finally, the B matrix is defined by extending the $B_i(\tau)$ vector to every node of the network, having dim($B(\tau)$) = $|V| \times K$:

$$B(\tau) = [B_1(\tau), B_2(\tau), ..., B_{|V|}(\tau)] \tag{3.11}$$

The Regularization stage is executed after the Competition ends, which can be after a fixed number of iterations α or by the termination criteria presented in [Silva and Zhao 2012]. The dynamics of the Regularization stage is modeled as follows:

1. When the Regularization step first starts, Eq. 3.9 is applied to each entry $B(t_B)$, i.e., $B_i^{(k)}(t_B) = R(i,k,N(\tau))$. Therefore, the Regularization step initially takes into consideration the domination level of the particles after the Competition step has ended.

2. Additionally, the incremental execution of Eq. 3.9 on the resulting preference matrix $B(t_B)$ can yield better results in networks with very dense and unbalanced communities. As such, it might be desirable to execute Eq. 3.9, μ more times ($\mu \geq 0$) upon the matrix $B(t_B)$, i.e., $B_i^{(k)}(t_B) = R(i,k,B(t_B-1))$. The user-defined parameter μ is denotes the total amount of incremental executions of Eq. 3.9 applied to $B(t_B)$. The behavior of the incremental executions is illustrated in Fig. 4.

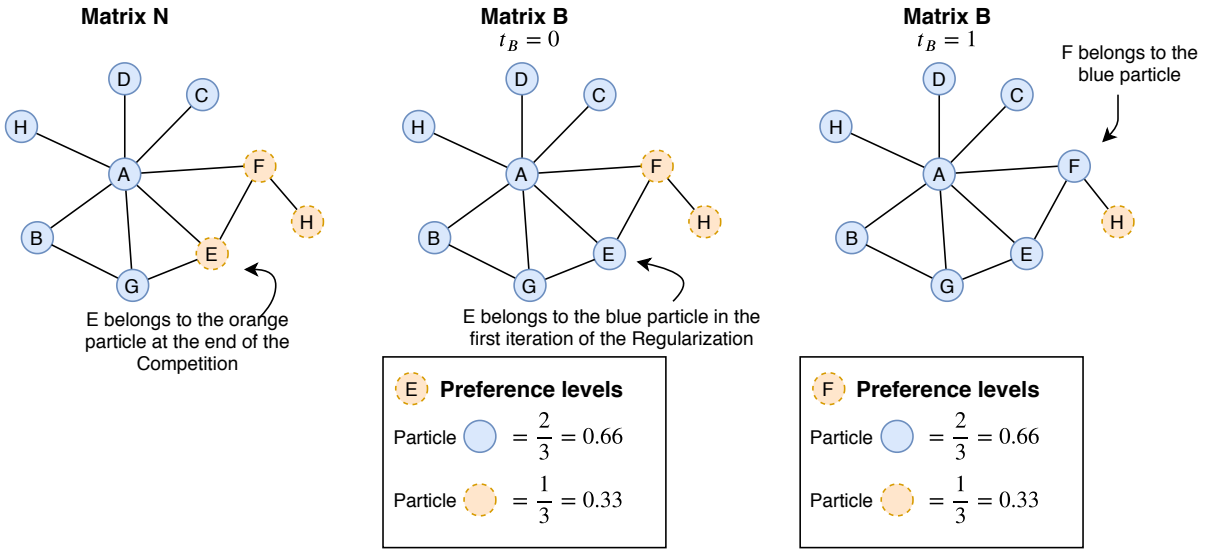

Figure 4 – Example of the incremental executions of the regularization function upon the previous execution: by applying Eq. 3.9 incrementally on the previous result $B(t_B - 1)$, the model is able to increasingly improve the results from the Competition stage, which may be beneficial to obtain fast and accurate community detection results with fewer iterations.

3.3 Transition Matrix for Network Exploration

The walking dynamics of the particles connect both steps of the system. They govern how the particle chooses the next vertex of the network to navigate, and, therefore, dictates how they compete for the nodes of the network.

In the Two-Step Particle Competition model, the random walk rule doesn't change, i.e., at each step, each particle randomly select a neighbor node to visit. However, the preferential rule has been changed considering the new guiding matrix $B(\tau)$, generated by the the Regularization function described in the previous section. The $\mathbb{P}^{(k)}_{\text{pref}}$ transition matrix guides the particle k to remain visiting the nodes it is more likely to belong to its community. Therefore, each k particle has its own preferential transition matrix on the time t. $\mathbb{P}^{(k)}_{\text{pref}}$ is then generated by applying the following equation on each one of $(i, j) \in V \times V$ elements:

$$\mathbb{P}^{(k)}_{\text{pref}}(i, j, t) = \frac{a_{ij} B_{jk}(t)}{\sum_{u=1}^{V} a_{iu} B_{uk}(t)} \tag{3.12}$$

Equation 3.12 denotes the likelihood of the particle k visiting the node j from the current node i by taking into consideration the guide $B(\tau)$ matrix, from the Regularization step. If node j has more neighbors belonging to particle k, this particle has a higher likelihood of visiting node j in the future.

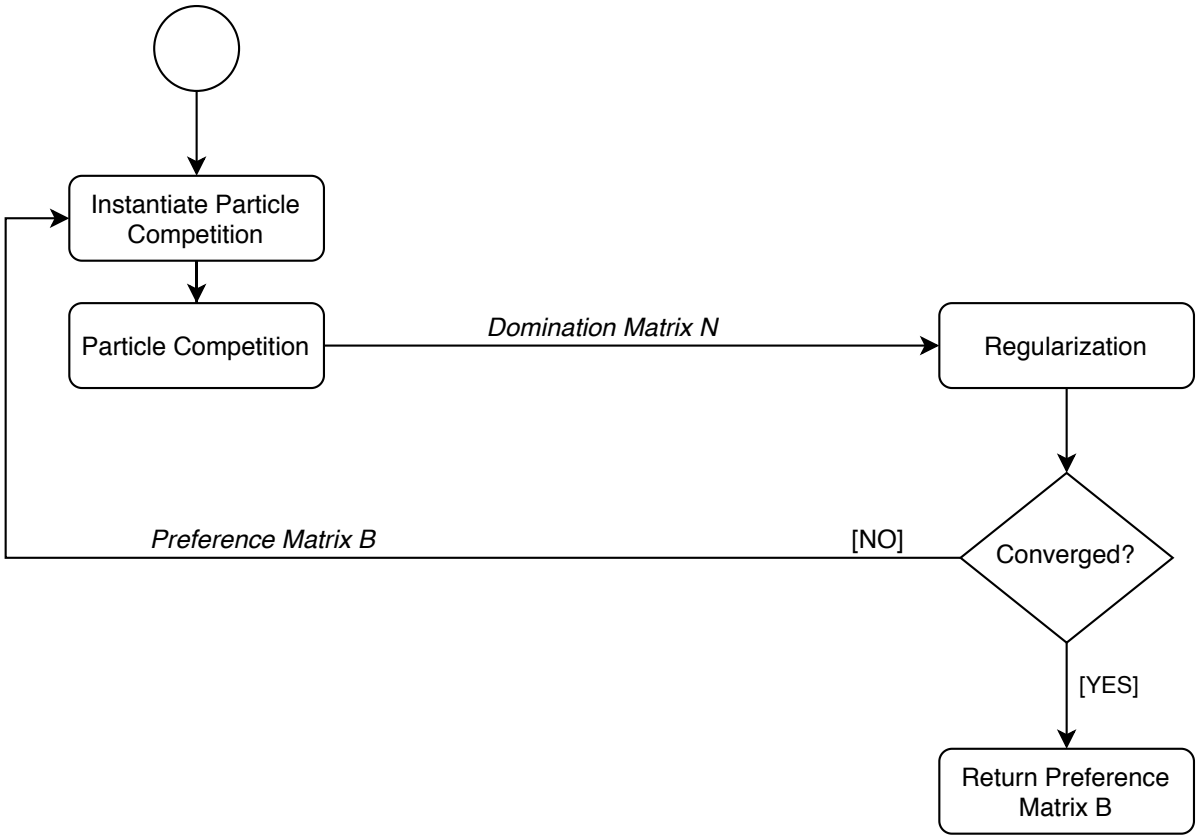

Figure 5 – Execution flow of the proposed method

3.4 The Proposed Algorithm

The Regularization step learns from the Competition using the ownership matrix the Competition generates. Competition is, in turn, influenced by the preference matrix generated from the Regularization step.

The process starts with the Competition step without preferential walking or bias. The dynamic system of the Competition stage runs until each particle has roughly settled in a community. As a result, the Competition stage yields the $N(\tau)$ matrix, containing the domination level of the particles over the vertex of the network. After which, the Regularization step will take resulting domination matrix $N(\tau)$ and use it to generate the guide matrix $B(\tau)$, as explained in detail in 3.2.1. Finally, the Competition step restarts and the label propagation occurs from the beginning, allowing for multi-samples of Competition results.

When both stages of the system run and a new preference matrix gets generated by the Regularization stage, we say a *epoch* (τ) passed. Therefore, τ counts how many times both stages were executed. The Competition and Regularization steps shall be executed alternately until the preference matrix $B(\tau)$ converges. On each new epoch, the Competition must be restarted clean, following the procedure detailed in Section 3.5.

3.5 Initial Condition of the System

The initial conditions to run the proposed method are now described. The proposed method starts with the Competition step. On its first execution, however, there is no data to generate the guide matrix $B(\tau)$. Therefore, each entry of $B(0)$, is initialized with

$$B_i^{(k)}(0) = \frac{1}{K} \tag{3.13}$$

, indicating no strong preference for the vertices of the graph by any of the particles.

The Competition step initialization procedure first selects a starting vertex for each particle in the system. The vertex selection uses the particle's preference over the nodes, given by the guiding matrix $B^{(k)}(\tau)$. When $\tau = 0$, this selection is naturally based on a uniform distribution, as initially, the particles have no strong preference over any of the vertex of the network, as stated in 3.13. However, should the guiding matrix $B^{(k)}(\tau)$ follow another distribution – which may happen after the Regularization step executes once – the starting vertex would be selected based on the preference of particle for it.

Additionally, the Competition must be restarted clean in the start of a new *epoch*, i.e., the only information it retains from the previous epoch is the new guide matrix, $B(\tau)$. Once the starting nodes are selected, the dominance-level matrix $N(0)$ is initialized by the expression

$$N_i^{(k)} = \begin{cases} 2, & \text{if vertex } i \text{ was selected for particle } k \\ 1, & \text{otherwise} \end{cases} \tag{3.14}$$

Lastly, the Competition initialization also requires to initialize the energy-level control vector $E(t)$ equally for each particle and the particle's state is set to *active*, as indeed the particles have enough energy to start to explore the network:

$$E(0) = \frac{1}{K} \tag{3.15}$$

$$S^{(k)}(0) = 0 \tag{3.16}$$

3.6 Termination criteria

With the model now defined, an important subject to be addressed is how to identify when it is the moment to stop iterating the Competition stage as well as the model itself. In this section, we will discuss possible termination criteria for both.

3.6.1 Competition stage termination criteria

The Competition must run enough times so that each particle may roughly dominate a community of the network. Notice that, as the task of clustering the nodes is delegated to the

Regularization stage, the Competition is no longer required to identify all the members of the community correctly – therefore, it is not required to run for many iterations. There are two approaches to decide when to stop the Competition stage:

- **Run a fixed amount of iterations**: the simplest approach is to set a fixed amount of iteration for the stage through the α user-defined parameter.

- **Check if the biggest change in domination-levels is below a certain threshold**: [Silva and Zhao 2012] proposes the usage of the infinity max-norm of the difference between the previous iteration $N(t-1)$ and the current one $N(t)$ to identify when is the best moment to stop the stage. The idea is that, if the highest level of change between iterations is below a certain threshold, then the model most likely has converged.

In the proposed two-stage model, we opted to use a fixed amount of iterations through the α parameter, as it provided better control over the behavior and speed of the Competition.

3.6.2 Model termination criteria

The following strategies can be employed as the termination criteria for the proposed method:

- **Fixed amount of iterations**: a basic approach would be to run both stages a fixed number of times and then return the preference matrix ($B(\tau)$), which contains what community each node belongs.

- **Check if the biggest change in preference-levels ($B(\tau)$) is below a threshold**: the equation $||B(\tau) - B(\tau-1)||_\infty$, which is the infinity max-norm of the difference between $B(\tau)$ and the $B(\tau-1)$ matrix, yields how much change has occurred from one *epoch* τ to another. If the change is smaller than a threshold (ϵ), then the model most likely has converged, and there is no need to continue the process.

- **Iterate until no class change between epochs**: another simple concept is to stop the model when no node change owners. At the end of each epoch, the newly preference-matrix $B(\tau)$ is compared to the previous $B(\tau-1)$ matrix. If none of the particles changes the majority of preference for the nodes, then it is an indicator that the model converged.

In this document we opted to check the difference of preference levels of epochs, using the equation $||B(\tau) - B(\tau-1)||_\infty < \epsilon$, where $||.||_\infty$ is the matrix max-norm of the difference between the preference matrix at *epoch* τ and the preference matrix at epoch $\tau-1$.

3.7 Algorithm and Parameters

The proposed method is now described in an algorithmic manner (Algorithm 1). It takes a total of six user-defined parameters; however, only K, Δ, λ and μ are essential parameters. The parameters of the model will be discussed in detail in the next section.

Algorithm 1 – Two-stage Particle Competition Algorithm

1: **procedure** PCR($K, A, \Delta, \lambda, \alpha, \mu, \epsilon$)
2: $B(0) \leftarrow$ initializeB() ▷ Use 3.13
3: $\tau \leftarrow 0$
4: **repeat**
5: $N(\tau) \leftarrow$ ParticleCompetition(K, A, $B(\tau)$, Δ, λ, α)
6: $\tau \leftarrow \tau + 1$
7: $B(\tau) \leftarrow$ Regularization(A, $N(\tau-1)$, μ)
8: **until** $||B(\tau) - B(\tau-1)||_\infty < \epsilon$
return $B(\tau)$
9: **end procedure**

Algorithm 2 – Particle Competition step

1: **procedure** PARTICLECOMPETITION($K, A, B, \Delta, \lambda, \alpha$)
2: $p \leftarrow$ assignRandomVertex(A, K)
3: $\mathbb{P}_{\text{rand}} \leftarrow$ generateRandomMovement(A) ▷ Use 3.7
4: $N(0) \leftarrow$ generateInitialN($p(0)$) ▷ Use 3.14
5: $E(0) \leftarrow$ generateInitialE(K) ▷ Use 3.15
6: $N(0) \leftarrow$ generateInitialS($p(0)$) ▷ Use 3.16
7: $t \leftarrow 0$
8: **while** $t < \alpha$ **do**
9: **for** $k = 1$ to K **do**
10: $\mathbb{P}^{(k)}_{\text{rand}}(t) \leftarrow$ calculate$P_{\text{pref}}(N(t), p(t))$ ▷ Use 3.12
11: $\mathbb{P}^{(k)}_{\text{rean}}(t) \leftarrow$ calculate$P_{\text{rean}}(N(t), p(t))$ ▷ Use 3.6
12: $\mathbb{P}^{(k)}_{\text{trans}}(t) \leftarrow$ calculate$P_{\text{trans}}(\mathbb{P}^{(k)}_{\text{rean}}(t), \mathbb{P}_{\text{rand}}, \lambda)$ ▷ Use 3.8
13: $p^{(k)}(t) \leftarrow$ chooseNextVertex($\mathbb{P}^{(k)}_{\text{trans}}(t)$, k)
14: **end for**
15: $N(t) =$ updateN($N(t-1)$)
16: $E(t) =$ updateE($N(t-1)$) ▷ Use 3.3
17: $S(t) =$ updateS($E(t-1)$) ▷ Use 3.4
18: $t \leftarrow t + 1$
19: **end while**
return $N(t)$
20: **end procedure**

The Competition stage is mostly similar to the one presented in [Silva and Zhao 2012], with the critical difference being line 18 of [Silva and Zhao 2012, Algorithm 1], in which, originally, the domination-level matrix $\bar{N}(t)$ gets calculated. The $\bar{N}(t)$ matrix is simply a normalized version of the $N(t)$ matrix, that the model uses to: 1) calculate the $\mathbb{P}_{\text{rand}}$ random

walking matrix; 2) identify the particle who owns each node of the network. Such operation is no longer necessary: it is delegated to the Regularization step.

The Regularization stage, presented as a standalone function in Algorithm 3, takes the input graph (A), the domination-level matrix $N(\tau)$ from the previous Competition step, and the user-defined parameter μ. The μ parameter ($\mu \geq 0$) allows additional Regularization to be executed upon the newly generated $B(\tau)$ preference matrix. Additional executions of the function may improve the regularization results with graphs with a very unbalanced community structure – Section 4.2.3 provides an empirical study of additional executions.

Algorithm 3 – Regularization step

1: **procedure** REGULARIZATION(A, $N(\tau), \mu$)
2: $B(0) \leftarrow$ calculateB(A, $N(\tau)$) ▷ Use 3.9
3: $t_B \leftarrow 1$
4: **for** i to μ **do**
5: $B(t_B) \leftarrow$ calculateB(A, $B(t_B - 1)$) ▷ Use 3.9
6: $t_B \leftarrow t_B + 1$
7: **end for**
return $B(t_B)$
8: **end procedure**

3.8 Model Complexity

We now discuss the computational complexity of the proposed method. The Particle Competition function, presented in 2, has the following complexity:

1. *Line 2*: Selecting a random node to start the particle has the complexity order of $O(K)$;
2. *Line 3*: The random movement transition matrix requires each edge to be visited, as such, it has the complexity order of $O(L)$;
3. *Line 4*: Each V vertex has a counter for every K particle in the system to be initialized; therefore, this step has the complexity order of $O(K|V|)$;
4. *Line 5* and *line 6*: For both lines, each particle has an energy level and a state to initialize, the complexity order is $O(K)$;
5. *Line 10*: The preferential movement transition matrix requires each link of the current node to be visited. Consider $\langle k \rangle$ to be the average degree of the graph: this step can be generalized to $O(\langle k \rangle)$;
6. *Line 11*: By using a hashable table to keep track of which nodes belong to each particle, this step can have the complexity order of $O(1)$.

7. *Line 12*: The transition matrix generation requires a scalar multiplication with a complexity order of $O(\langle k \rangle)$.

8. *Line 13*: The probability distribution given by line 12 is used to choose the next node to visit. This distribution takes into consideration the links connecting the current node to its neighbors. Therefore, the complexity order can be generalized to $O(\langle k \rangle)$, as $\langle k \rangle$ represents the average degree of the graph;

9. *Line 15*: To update the domination-matrix $N(t)$, it is necessary to increment K positions visited by each particle. It takes $O(K)$;

10. *Line 16* and *line 17*: Complexity order for both is $O(K)$, as it is required to update each one of the K particles in the system.

The lines 10 to 13 has the computational complexity of $O(\langle k \rangle)$ and is repeated for each particle for a total of K times, for a total complexity of $O(K\langle k \rangle)$. Inside the main loop still, lines 15 to 18 have the complexity of K, which brings the complexity of the main loop block to $O(K\langle k \rangle + K)$. Additionally, the Competition step must also run enough iterations so the communities may be defined. We propose two termination criteria for this loop. The simplest one is to iterate a fixed α amount of time. If such an approach is used, the Competition stage has the complexity order of $O(\alpha K\langle k \rangle + \alpha K)$.

The work of [Silva and Zhao 2012] makes an approximation of the total amount of iterations necessary for the main loop. For the sake completeness, let us estimate the required amount of iterations of the Competition stage (approximate α): propose we have a network with completely separated communities. In such a network, it only takes one single visit in every node to classify it correctly. Therefore, the main loop would be required to iterate $c_1|V|$ times ($O(|V|)$), where c_1 is the fraction of random movement the particles may perform (which dictates how likely the particle is to visit unknown nodes). Extending the idea to a connected network with a well-defined community structure, the number of iterations required is $c_2|V|$, with c_2 being a constant $c_2 > c_1$ (again, $O(|V|)$). Generalizing the same idea for any network, we find that the total amount of iterations required for all nodes to be dominated by the particles is $c|V|$, and c is a constant, which increases with the proportion of inter-community links. Therefore, we estimated that the main loop of the Competition stage repeats $c|V|$ times.

The Regularization stage complexity order is much simpler to calculate. Equation 3.9 gets executed for all V nodes of the network. For each node, we must identify the domination-levels of each K particle of the node's neighbor. Such a task carries the complexity order of $O(|V|\langle k \rangle K)$, as $\langle k \rangle$ represents the average degree of the network.

In summary, taking into consideration the main loop of the Competition runs $c|V|$ times, the Competition stage has the Complexity order of $O(\alpha K\langle k \rangle |V| + K|V|)$. The Regularization

stage, which repeats μ times, has the Complexity order of $O(\mu|V|\langle k\rangle K)$. In practice, there are a few situations to consider when dealing with real-world networks:

- **Sparse networks**: in sparse networks, when $\langle k\rangle \ll |V|$, the Competition stage runs closer to $O(K|V|)$, with K and V being the most impactful parameters.

- **Dense networks**: if the average degree $\langle k\rangle$ grows proportionally to the number of vertices $|V|$ in a network (and, therefore, the network is very dense), the Competition stage will run at $O(K|V|^2)$.

- **In general**: many real-world networks are sparse ($\langle k\rangle \ll |V|$) and display fewer communities than nodes ($K \ll |V|$). For such networks, the Competition stage should run near-linear time, with a complexity order of $O(|V|)$.

In practice, the complexity of the proposed model is dominated by the Competition stage, as the amount of iterations of the Regularization, μ, is much smaller than the number of iterations of the Competition: $\mu \ll \alpha$: as discussed in Section 3.9, an appropriate value for μ resides between 0 and 20, often staying on the smaller side of this scale. Therefore μ can be considered as a constant, i.e., $\mu = c = O(1)$.

3.9 Parameters overview

In order to elucidate the user-defined parameters used by the technique and presented in Alg. 1, we present them in table 1, with their description and a small discussion. In section 4.2, the main parameters of the model are further discussed, including how they impact the accuracy and performance of the model.

Parameter	Description and discussion
K	The number of communities to identify. Both [Silva and Zhao 2012] and [Gao *et al.* 2019] defines two different models for automatically identifying the appropriate K parameter, which can be used with the Competition stage of the proposed method.
Δ	Ranging from 0 to 1, Δ represents the amount of energy to recharge or to consume from a particle when it defends its community or attacks a node who belongs to another particle. In general, a good value for Δ is 0.2.
λ	Ranging from 0 to 1, λ defines the emphasis (or weight) given to the preferential walking model. A higher λ means the particle will have a higher preference in defending its community. A good value for λ is often 0.6.
μ	The total amount of additional executions of the Regularization function (Eq. 3.9) upon its previous execution. When $\mu = 0$, the Regularization stage only takes into consideration the domination-matrix $N(\tau)$ of the Competition. The behavior of a positive μ is discussed in Section 3.2.1 and highlighted in Figure 3: a higher value for μ may improve how fast the model is able to achieve good clustering results.
α	Implementation detail: α is used as a straightforward termination criteria for Competition step, i.e., when the competition iteration counter (t) reaches α, the Competition ends and the Regularization stage begins with the domination matrix $N(t)$.
ϵ	Implementation detail: ϵ is used as the termination criteria threshold of the model. Section 3.6 defines other criterias to identify when it's appropriate to stop the proposed model and provide the results. An appropriate value for ϵ, used throughout this project, is 0.05.

Table 1 – The list of parameters of the algorithm, containing their description and a short discussion.

CHAPTER

4

EMPIRICAL ANALYSIS OF THE MODEL

With the proposed model now fully described, this chapter aims to illustrate how the model works in practice while also providing guidelines for its usage in real-world scenarios. Additionally, it also highlights the aspects in which the Proposed model excels previous Particle Competition models on unsupervised learning tasks and acts as a motivator for its usage over the original Particle Competition models.

4.1 Empirical Model Analysis

This section demonstrates how both stages can work with each other to identify the communities of the network. We present four different analyses: firstly, a single execution of the model, with snapshots of the domination-level matrix N and the preference-level matrix B during the process, the point is to highlight how both stages learn with each other. in practice. Next, each stage is analyzed separately, starting with the behavior of the Competition stage, which shows that each particle is, indeed, capable of finding the whereabouts of the communities of the network. Then, we illustrate the behavior of the Regularization stage. The objective is to highlight the usefulness of the incremental executions of the Regularization function. Lastly, we apply the model on a larger network, which serves to motivate the usage of multiple *epochs* on the clustering process.

We apply the model on the GN Benchmark network [Danon *et al.* 2005] with $Z_{out}/\langle k \rangle = 0.2$. The network is generated with $M = 4$ communities with $V = 128$ nodes in total, making it 32 nodes per community.

4.1.1 Single-execution illustration

To illustrate how the proposed method behaves and to exemplify how both steps learn with each other, Fig. 6 shows the domination and regularization clustering results during four

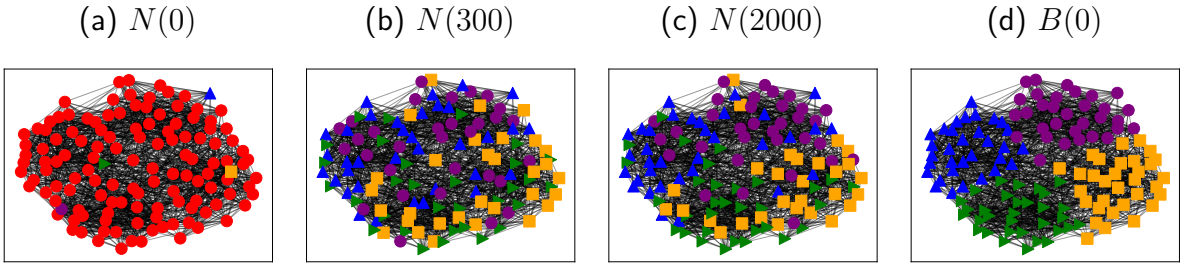

Figure 6 – Snapshots of a single execution of the proposed method on the GN network with $Z_{out}/\langle k \rangle = 0.3$. (a), (b), (c) shows the $N(t)$ domination-levels of the particles on time 0, 300, and 2000 respectively. It can be noted that at time 2000, each particles roughly settles in a community. (d) shows the first regularization result obtained upon $N(2000)$, which is indeed the expected community result.

distinct moments in the process. Fig. 5a reflects the domination matrix $N(0)$ on time $t = 0$, the first iteration of the Competition: each particle has been randomly placed in a node and automatically dominates it. Figure 5b and 5c, however, shows the Competition at later times ($t = 300$ and $t = 2000$, respectively), where each particle has roughly settled and dominated a community. Finally, Fig. 5d shows the first iteration of the Regularization, which in this particular network is enough to yield the correct community result. Next, the Competition would then restart from scratch, but this time each particle would take into preference visiting the nodes from the previous Regularization step, i.e., Fig. 5d.

Although not reported in Fig. 6, the process is then repeated until the current and previous classification results are similar enough, which, depending on the network, may take only one additional execution to converge depending on the criteria used (as specified in Section 3.6). It is also worth pointing out that, although possible with enough iterations, the Competition step is not required to identify all the members of the community correctly, but rather, only label its' whereabouts, as shown in Fig. 6.

4.1.2 Competition over time ($N(t)$)

The Competition stage must roughly label the communities of the network for the Regularization step. Therefore, each particle must settle in a community at the end of the Competition. In this section, we verify such a requirement by analyzing the domination-level matrix $N(t)$ throughout the first epoch, which will ultimately contain the node's owner.

Figure 7 shows the behavior of the domination matrix $N(t)$, normalized between 0 and 1, and averaged over all the nodes of each community for each particle. The analysis shows, for each one of the $K = 4$ particles in the system, the average domination level each particle possesses upon all nodes of $M = 4$ communities in the network. The particle with the highest domination level is the owner of the node; therefore, the expectation is that each particle to have a higher average domination level on the nodes of a single community only. Fig. 7 indeed shows

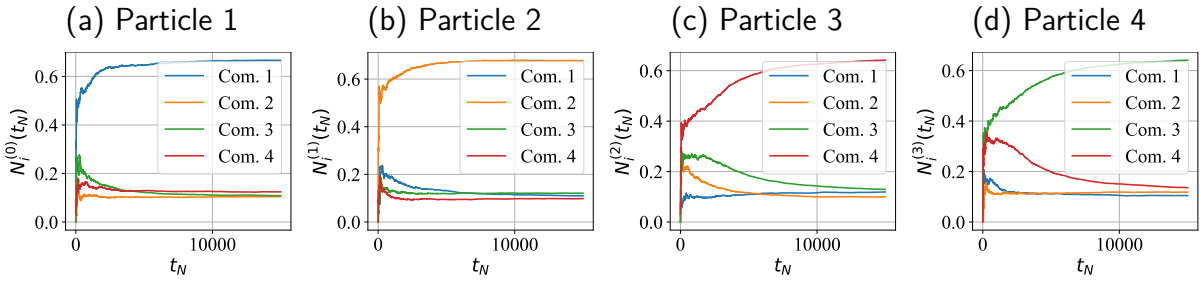

Figure 7 – Example of the average domination levels on the nodes of each $M = 4$ communities of the network, for the $K = 4$ particles in the system. The GN benchmark network was generated with $Z_{out}/\langle k \rangle = 0.2$. Each particle finds and dominates a community of the network.

that each one of the particles, during the Competition stage, can correctly find, dominate, and defend the members of a single community of the network. Furthermore, it is worth pointing out the particle's ability to expel or remove an enemy particle from its dominated community, such as in Fig. 6d, where particle four gets expelled from dominating the members of community 4, which is ultimately owned by particle 3.

4.1.3 Regularization analysis ($B(t_B)$)

The Regularization function will ultimately contain the community structure of the network by taking into account the neighbors of each node. Therefore, it is required to be executed at least once after the Competition stage. Still, additional executions of the Regularization function upon the first Regularization result might be desired. Incremental executions may help classify networks with very unbalanced or high-density communities.

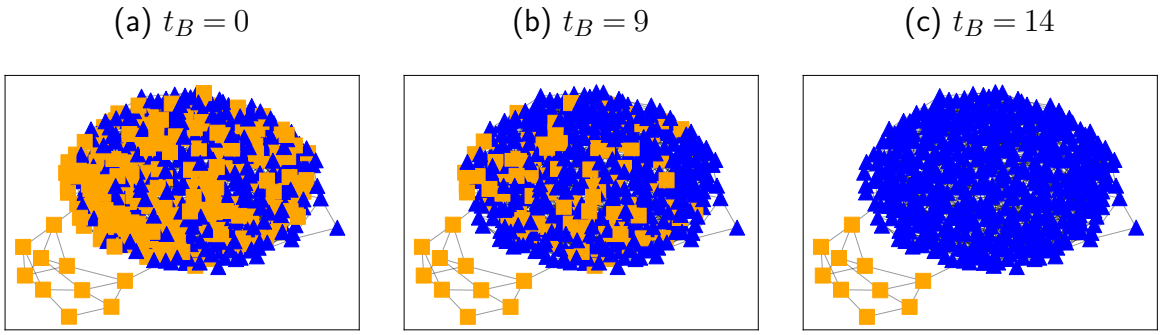

Figure 8 – Additional executions of the regularization function can be beneficial to correctly identify communities in networks with unbalanced or otherwise densely connected communities.

Figure 8 illustrates the behavior of incremental executions of the Regularization function in extremely unbalanced networks with two communities, one containing 10 nodes and the other community having 500: Fig. 7a shows the first execution of the Regularization, which is not enough to identify the community in such a degree of unbalance. The incremental executions of

the function, is beneficial to community detection, as shown in Fig. 7b and 7c. In Section 4.2.3, we also discuss and provide a guideline for choosing the value of μ.

4.1.4 Epoch analysis (τ)

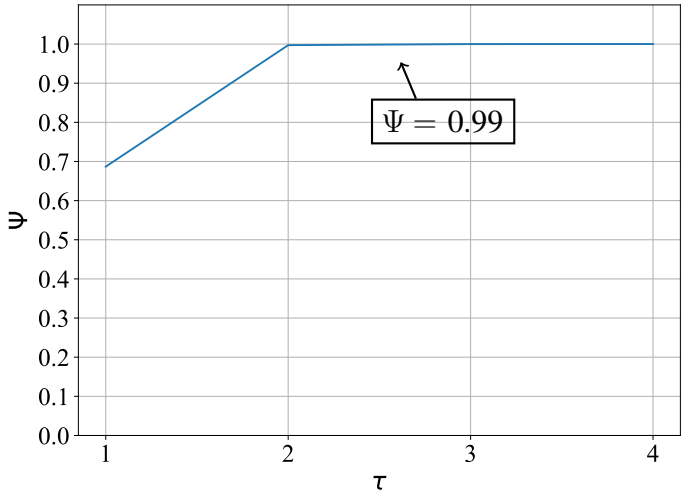

Figure 9 – Analysis of the accuracy of results through different epochs on a GN benchmark network with $Z_{out}/\langle k \rangle = 0.3$ and $V = 10000$ nodes, which shows that the back and forth between both stages may improve the community detection results.

Lastly, we investigate the behavior throughout different epochs by analyzing the community detection accuracy (Ψ) from the Regularization function, obtained at the end of each epoch. Figure 9 shows the result from such test, executed upon a GN benchmark network with $Z_{out}/\langle k \rangle = 0.3$ and $V = 10000$. As Fig. 9 shows, the back-and-forth of the Competition with the guide data from the Regularization is beneficial to obtain accurate community detection results. In essence, the first epoch's community detection rate (Ψ) is only 0.69; the second epoch improves the result to 0.99; finally, the community can be fully recovered by epoch 3 and confirmed by epoch 4.

4.2 Parameters analysis

In this section, the main parameters of the model are tested with the object to compare its behavior with the previous Particle Competition model. It also provides guidelines on how to choose appropriate values for each parameter, depending on the network. To empirically test the sensitivity of the proposed method, we will employ two artificial network models with community structure. We will employ those networks for testing purposes throughout the rest of the document; therefore, we shall describe them now.

The first network is the Girvan and Newman's benchmark network, first described in [Newman and Girvan 2004]. It quickly became the standard benchmark for community

detection [Danon *et al.* 2005] and has been widely used to test the accuracy and robustness of many community detection techniques. The method takes as input the number of vertices (V, usually kept at 128), the average degree of the nodes ($\langle k \rangle$, usually 16), the number of communities (M, usually 4) and $Z_{out}/\langle k \rangle$ which is the proportion of links made to nodes inside the community. The last parameter is often between the 0.0 to 0.5 range: by changing the proportion of links going to nodes outside of the community, one can specify how well defined the communities of networks are. The community-mixing parameter is especially useful to identify how well a community detection technique performs under different densities of communities. When $Z_{out}/\langle k \rangle$ is close to 0.0, there will be fewer links going to nodes of another community, and the community will be highly dense and easier to classify. On the other hand, as $Z_{out}/\langle k \rangle$ gets closer to 0.5, the model will generate communities highly mixed. When $Z_{out}/\langle k \rangle = 0.5$, for example, half of the links of each node will be going to nodes of other communities; therefore, the community will not be as well defined.

The second network employed in this document will focus on testing the model's ability to identify unbalanced communities correctly. We propose an artificial network, composed of two communities with varying sizes. The following parameters control the creation of the network: the first community size and degree, the second community size and degree, and finally, the number of bridge links, i.e., links connecting the two communities in a network. We now describe how to create the network: firstly, the nodes of both communities are generated and connected within themselves randomly until the target inner community degree of the community is reached. After generating both communities, links are created between nodes of both communities at random, until the target number of connecting links is reached. As a result, a network with two communities of different sizes and densities is generated for unbalanced community tests.

4.2.1 Impact of Δ

The Δ parameter controls how quickly the particle's energy gets recharged (when visiting a node it owns) or drained (when visiting node belonging to another particle). The previous version of the Particle Competition model was not susceptible to the Δ parameter when $0.05 \leq \Delta \leq 0.5$, therefore, the general rule is to choose a $\Delta \leq 0.5$ to assure good community detection accuracy. On the other hand, $\Delta \geq 0.5$ could be detrimental to the performance of the method and the reason is straight forward: higher values for Δ forces the particle to retreat to its community as soon as it attacks a node owned by another particle, limiting the occurrence of Competition.

The proposed method, however, is even more robust to the value of Δ. Figure 10 shows the impact of different values for Δ in the GN Benchmark network with $Z_{out}/\langle k \rangle = 0.5$ and $M = 4$ communities. Once again, the Proposed Model manages to obtain similar accuracy regardless of the Δ used. In terms of speed, Δ also does not impact the total amount of iterations required to converge.

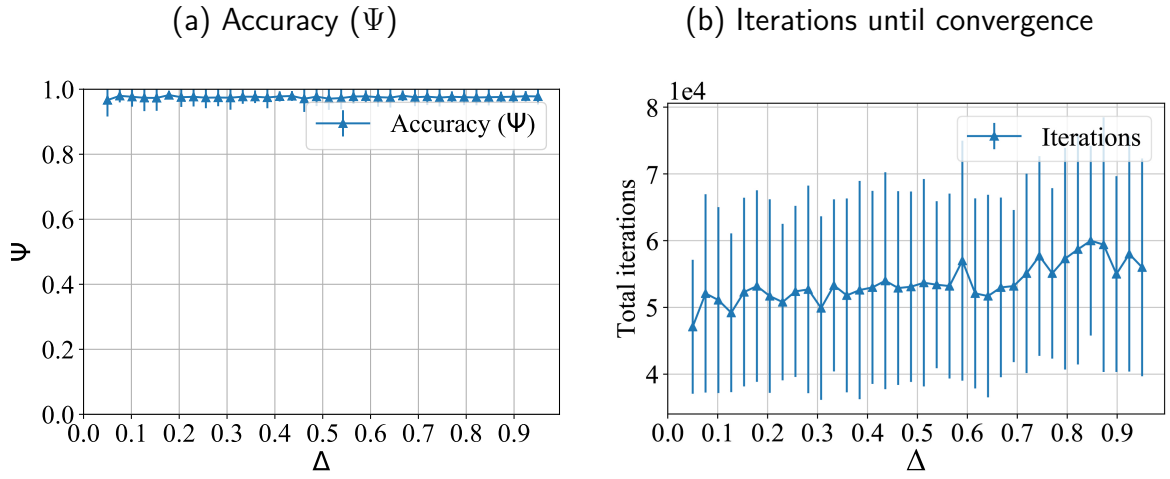

Figure 10 – Effect of Δ parameter on the GN Benchmark network with $Z_{out}/\langle k \rangle = 0.5$. Averaged over 100 executions

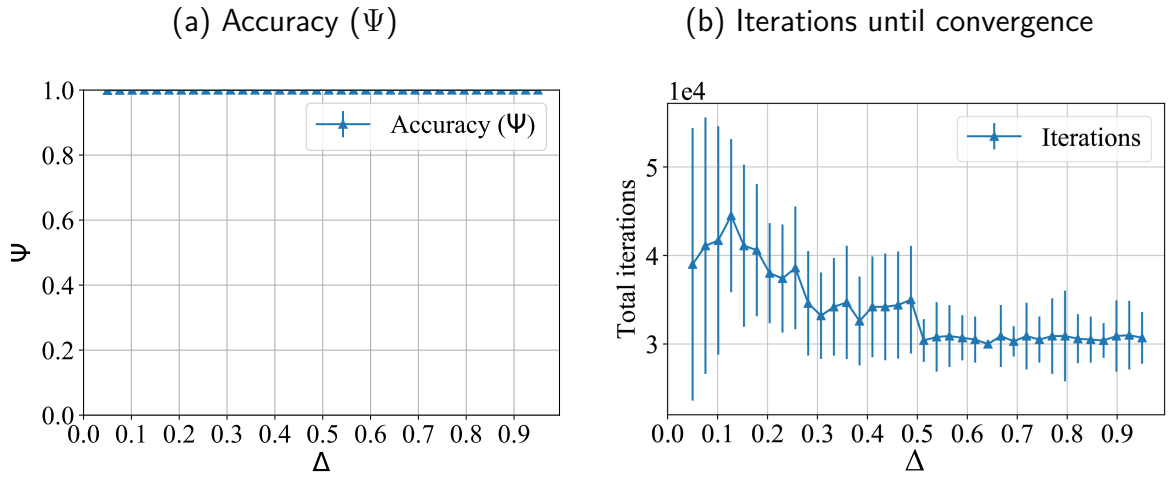

Figure 11 – Effect of Δ parameter on the unbalanced community network, containing two communities with 10 and 100 nodes. Averaged over 100 executions

Figure 11 shows the same test in a network with an unbalanced community structure. Although it gives the same results in terms of accuracy, it displays an unusual behavior in the total iterations before converging. Values for Δ above 0.5, which forces the particle to teleported back to its community faster, converge much quicker than when $\Delta \leq 0.5$, which allows the particle to wander around an enemy community for longer. This behavior may happen because the network has only two unbalanced communities. Therefore, the quick return of the invading particle may be beneficial to constrain it inside its community.

4.2.2 Impact of λ

The λ parameter, discussed in Section 3.9, controls the mixture of random and preferential behavior of the particle. Higher values for λ bias the particle towards choosing the nodes it already owns. Both previous iterations of the Particle Competition model [Quiles *et al.* 2008, Silva and Zhao 2012] proved that choosing the appropriate value for λ was essential to obtain good community detection results and indeed, the empirical tests showed that 0.6 was good value for λ, as it allowed for both movements to be combined [Quiles *et al.* 2008].

The GN benchmark network is also employed in the same test in [Quiles *et al.* 2008, Fig. 2], therefore the can be used as a comparison between the performance of both techniques. Generally, the proposed method can achieve good clustering results in a wider range of values for λ, usually at the cost of speed. The model was tested with the following parameters set: $\mu = 1$, $\Delta = 0.07$ and $\epsilon = 0.05$.

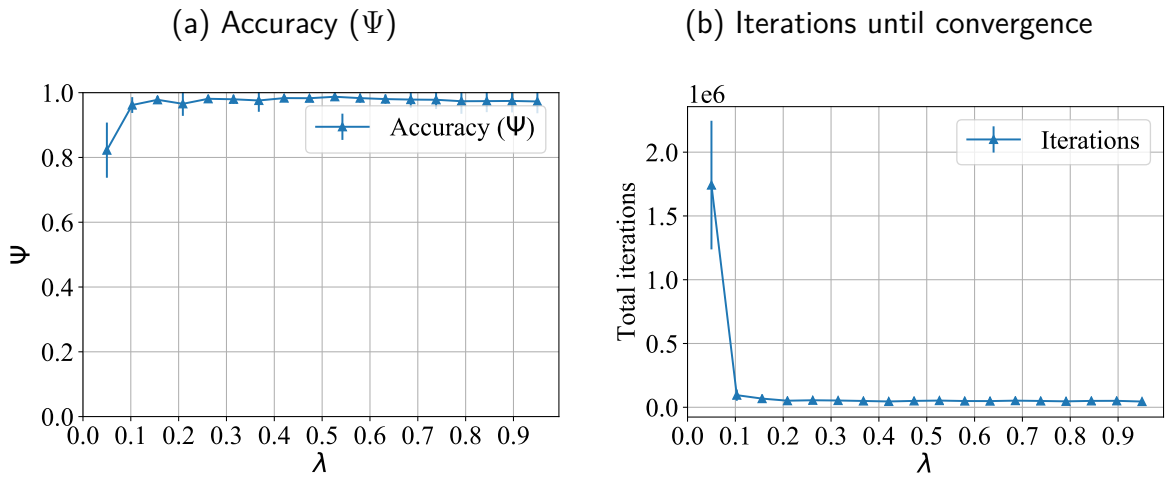

Figure 12 – The impact of different values for the λ parameter for the GN Benchmark network, with $Z_{out}/\langle k \rangle = 0.5$. Averaged over 30 executions.

Figure 12 show the results obtained when executing the proposed method with different values for λ, in terms of community detection accuracy (Fig. 12a) and total amount of iterations until convergence (Fig. 12b) for the GN Benchmark network. For such a network, the model can also achieve good clustering results in a wide range of values for λ. When comparing to [Quiles *et al.* 2008, Fig. 2], the proposed method manages to surpass its results in terms of accuracy regardless of the chosen value for λ. Additionally, once again, values close to 0.6 ensured the highest average of accuracy.

Figure 13 presents the results on the unbalanced network previously described. The model can correctly identify all members of the communities of the network when $\lambda \geq 0.25$: once again, allowing for a broader range of acceptable values for this parameter, but often failing when λ is too low. In essence, the most significant impact of an inappropriate value for λ

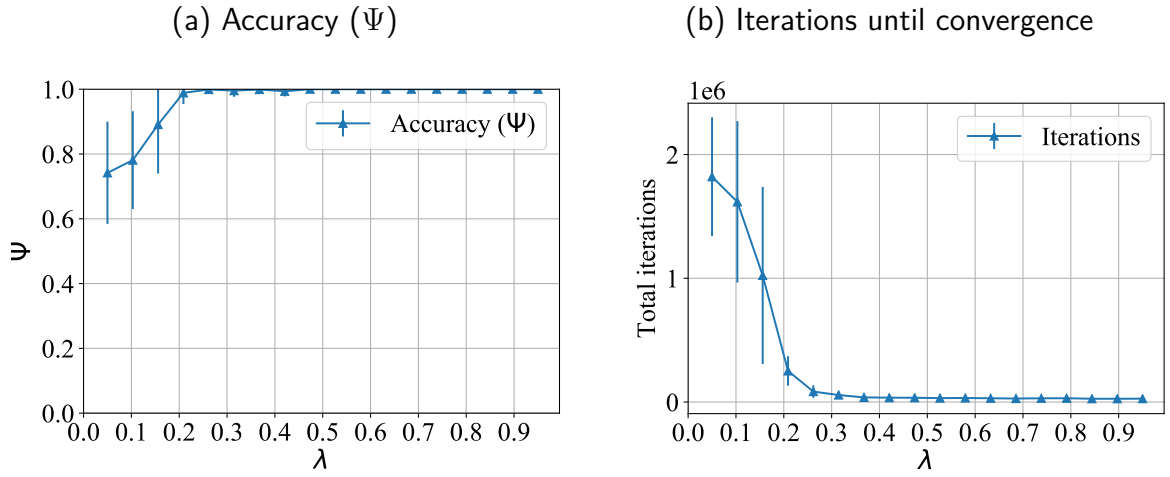

Figure 13 – Effects of the λ parameter in terms of accuracy and iterations until convergence on a network with unbalanced community structure containing two communities, one with 10 nodes and another with 100 nodes. Average of 30 executions.

also observed in Fig. 12, is the model's speed: the selection of values that do not allow for the appropriate mixture of movement forces the model to continue iterating until convergence.

The empirical tests demonstrate that the proposed model is much less susceptible to the value of λ. The model is more robust than previous Particle Competition techniques because the clustering results are always generated from the Regularization stage, which displays a high ability to correctly identify which members of the community belong with each other, often fixing the occasional mistakes of the Competition. Moreover, the back and forth between the two stages, combined with the used termination criteria, assures that the model will continue to work when uncertain of the results (through results too different between *epochs*). Such a combination of characteristics makes the proposed method more robust and resilient than the previous versions.

4.2.3 Impact of μ

Although the proposed model can achieve accurate results in a wide range of values for the parameters Δ and λ, the same can not be said to parameter μ, as the model displays a high sensibility to the accuracy of the results obtained.

In order to test the impact of the parameter, the proposed model is applied to two sets of tests with 5 different values for μ: 0, 5, 10, 15, 20. The first test checks the accuracy of the model when executed in a balanced GN benchmark network, with a varying mixture of communities ($Z_{out}/\langle k \rangle$). The second test compares how each μ behaves when faced with networks with an unbalanced community, whose generation was previously described. Once again, the first community contains 10 nodes, and the size of the second community varies. The tests reveal the model is very susceptible to the value of μ, which must be chosen according to the topology

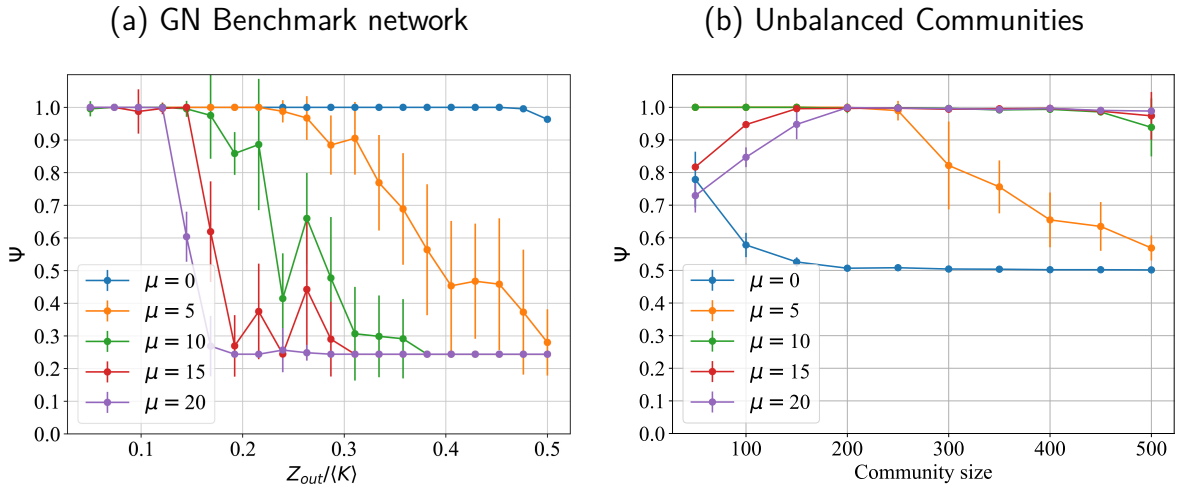

Figure 14 – Impact of different values for μ on different network structures. (a) shows the GN Benchmark network with varying $Z_{out}/\langle k \rangle$, allowing the comparison on different level of community mixture. (b) shows the impact on networks with varying degree of community sizes: the first community has 10 nodes and the second community varies. Averaged over 100 executions.

of the network, requiring some knowledge about it. Figure 14 shows the accuracy obtained on both tests: Fig. 13a is the GN Benchmark with a varying mixture of links and Fig. 13b is the unbalanced network with varying levels of unbalance.

As can be observed, μ displays different effects on different networks. Figure 13a, which shows the effects on the accuracy on communities with increasing levels of mixtures, it can be observed that higher values for μ are detrimental to the accuracy of the model: the higher μ is, the quicker it starts to fail in detecting the communities. This effect occurs because higher values for μ may spread the influence of the particles beyond their community, exceeding the boundaries of their community. Therefore, higher values for μ are only appropriate when the network exhibits *strongly defined communities*, which is when the proportion of links between communities is low.

Figure 13b, however, presents a very different behavior in comparison to Fig. 13a: smaller values are not able to produce accurate results. This effect happens because, if the community structure is too unbalanced, too few iterations of the Regularization are not enough to fix the wrongfully classified nodes and restrain the particles. This consequence occurs primarily in networks with highly unbalanced communities (see Fig. 3). However, if μ is too high, the Regularization might overstep the boundaries of each community and wrongfully generate a guide that joins two or more communities in one.

The value of μ must be chosen based on the structure of the network: higher values are appropriate only when the network displays highly unbalanced and sharply defined communities. Otherwise, lower values should be used, as they are appropriate for networks that display poorly defined communities. Still, it is essential to notice that, even when μ is 0, the model is still able to

provide more accurate and faster results in comparison to previous Particle Competition models, as seen in the next section.

CHAPTER

5

COMMUNITY DETECTION AND DATA-CLUSTERING SIMULATIONS

In this chapter, we apply the proposed method to a set of artificial networks and real-world networks for community detection. Additionally, the model is tested in graph-based classification tasks, using famous real-world data sets from UCI, where it is compared to the state-of-the-art methods for data classification.

5.1 Simulations on Artificial Data

In order to test the performance and accuracy of the proposed method, the Girvan and Newman's benchmark [Danon *et al.* 2005] is used. A robust community detection method should be able to correctly identify communities even when $Z_{out}/\langle k \rangle$ is high, and communities are highly mixed. Other parameters are kept fixed as $V = 128$ nodes, $\langle k \rangle = 16$ degree and $M = 4$ communities, thus the networks contains communities with 32 nodes each.

As can be observed in Fig. 15, which shows the community detection accuracy (Ψ) for different levels of community mixture ($Z_{out}/\langle k \rangle$), the proposed method is able to correctly identify the communities of the networks even when $Z_{out}/\langle k \rangle$ is close to 0.5, when the communities are barely defined. Not only that, but the method is also able to obtain good results when $Z_{out}/\langle k \rangle > 0.5$. Because there are $M = 4$ communities in the network, even though most of the links of a given node are going to nodes outside the community, the number of communities still makes the frequency of internal links higher than outside links (this would not be the case with $M = 2$). Therefore, the network still has a community structure to some degree.

Furthermore, by comparing the results obtained in Figure 15 with [Quiles *et al.* 2008, Fig .8] as well as [Silva and Zhao 2012, Fig. 8], it can be observed that the proposed method excels the results obtained in both previous versions of the Particle Competition algorithm. Additionally, by comparing Fig. 15 to [Danon *et al.* 2005, Fig. 2], it is possible to verify that the

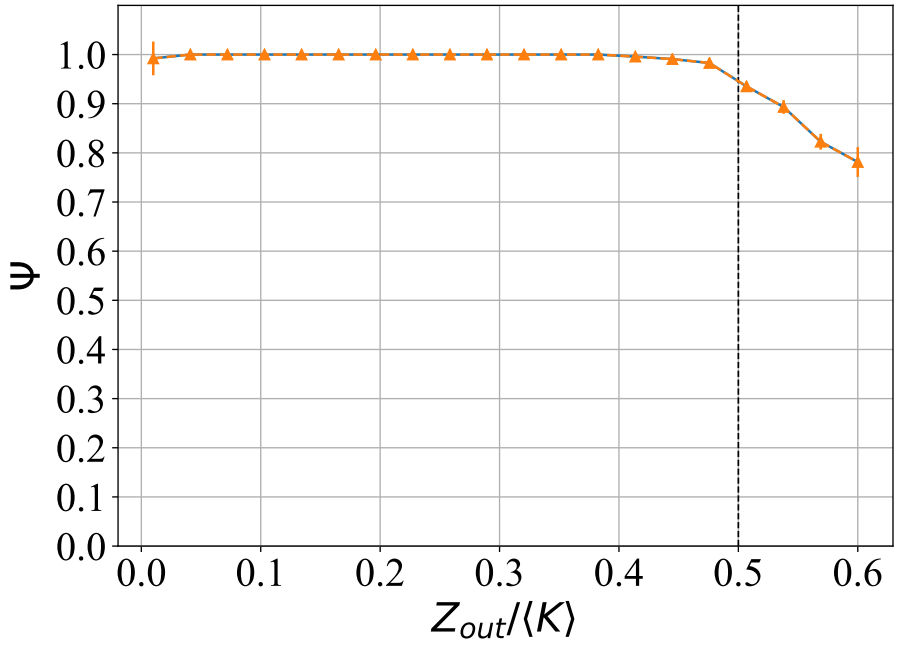

Figure 15 – Community detection accuracy of the proposed method on the GN benchmark network [Danon *et al.* 2005], averaged over 100 attempts. After $Z_{out}/\langle k \rangle = 0.5$, the communities are no longer strongly defined. However, because there are $M = 4$ communities, the proposed method is still able to identify the communities with certain accuracy, as the number of inner community links are higher than the amount of links to vertex of other communities.

proposed method also obtains excellent accuracy when compared to other robust methods for community detection.

Next, to evaluate the ability of the proposed method on unbalanced community detection, the Normalized Mutual Information value (NMI) is used to test the models' performance [Danon *et al.* 2005]. For this set of tests, besides the average NMI, we also plot the best and worst results obtained in the 100 attempts. Additionally, the proposed model is compared to a set of related community detection techniques. The first technique is [Gao *et al.* 2019], which employs the original Particle Competition mechanism proposed in [Silva and Zhao 2012] and that the proposed method uses as the first stage. We also test the original Label Propagation technique [Raghavan, Albert and Kumara 2007], which employs a propagation concept that also takes into consideration the neighborhood information of the nodes, but with a different propagation mechanism. Finally, we also test a more recent Label Propagation-based approach [Zhu and Xia 2018], which employs an Adaptive H-index rank to order the nodes of the network in an attempt to provide more stable results in comparison to the original LPA.

Figure 16 measures the performance of the models in terms of different sizes only. The network is built with two communities, the first containing 50 nodes, while the other varies between 50 and 5050 nodes. As can be observed, the proposed technique can consistently identify the correct community structure. The LPA technique can also recover the correct result most

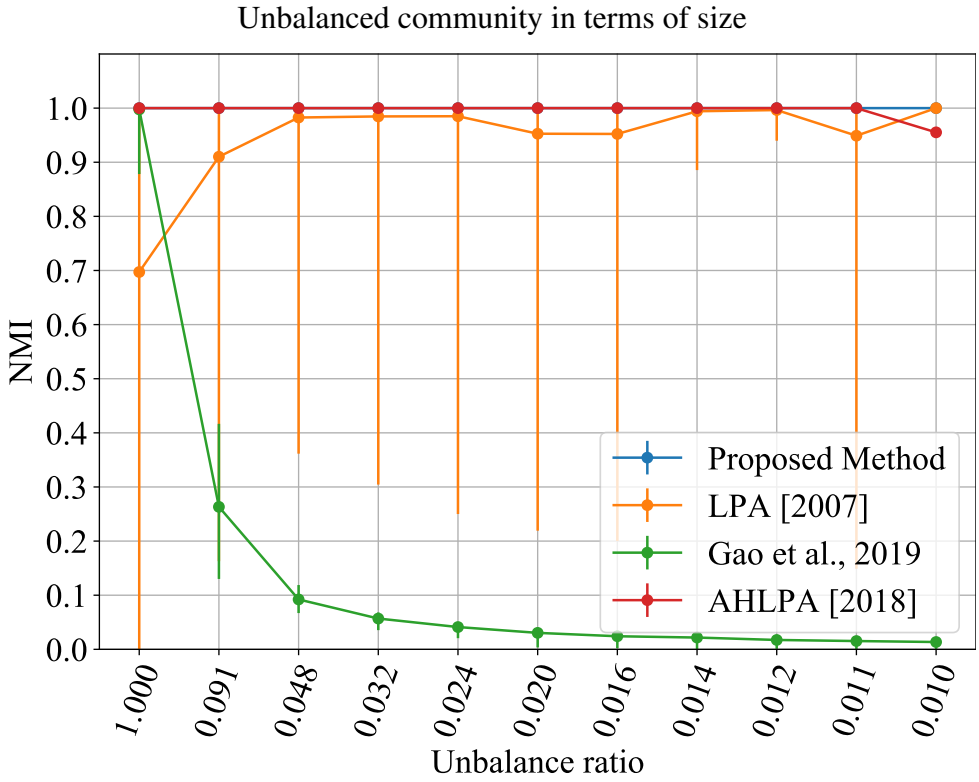

Figure 16 – Unbalanced community in terms of size by varying the size of the second community between 50 and 5050 nodes. The plotted line is the NMI, while the error bars shows the best and worst NMI obtained in 100 attempts.

of the time, however the model is sometimes prone to fail. The original Particle Competition is unable to identify communities with different sizes. The AHLPA technique can also consistently recover the correct partition.

Next, we test the models' ability to recover community structure with different densities. The networks are built with two 100 nodes communities. The first of which has an inner average degree of 6 and the second varies between 6 and 60. As can be observed in Figure 17, the proposed model, as well as the Particle Competition, can correctly identify the community structure of the networks. The LPA-based models, however, seem to struggle in producing correct results consistently.

Finally, Figure 18 combines density and size changes: the networks are generated with one community having 50 nodes with an average degree of 6. In contrast, the other community varies between 50 to 5050 nodes and an average degree between 6 and 60. As can be observed, the proposed method can consistently identify the correct community structure of the network. On the other hand, [Gao *et al.* 2019] is unable to identify communities with different sizes. The label propagation-based techniques, even though they are able sometimes to recover the correct community, are unable to provide consistent results.

In conclusion, the proposed model provides a robust approach to unbalanced community detection. Moreover, the proposed technique can achieve not only high unbalanced community

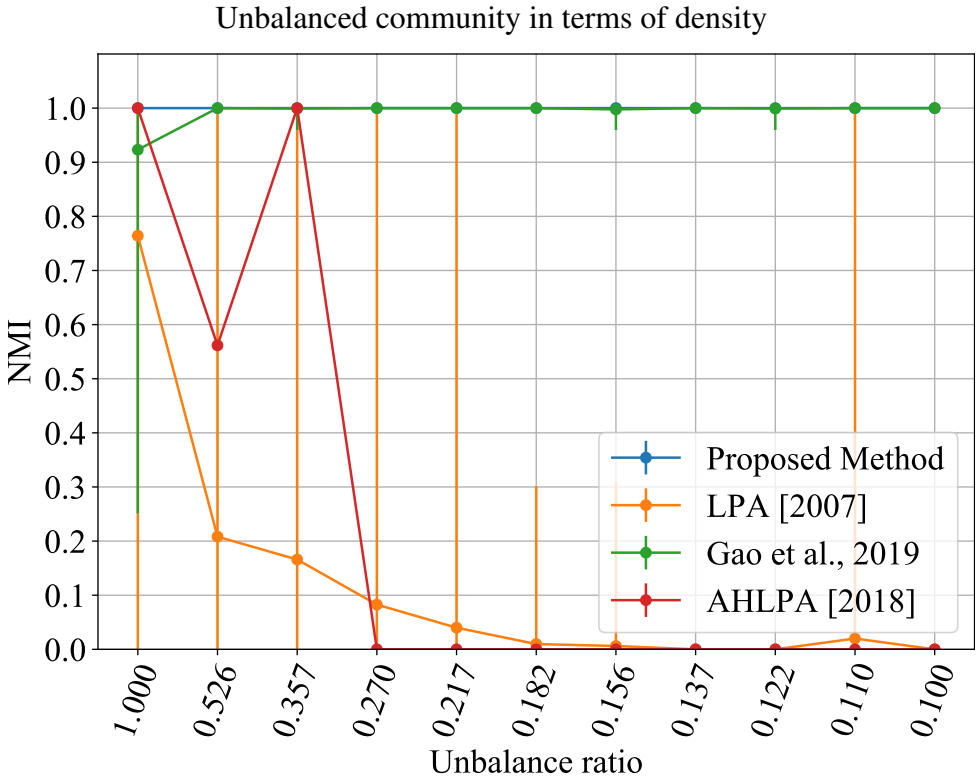

Figure 17 – Unbalanced community in terms of density by varying the inner degree of the first community. The plotted line is the NMI, while the error bars shows the best and worst NMI obtained in 100 attempts.

detection precision but also present high stability with minimal oscillation of detection results.

5.2 Simulations on Real World Data

Now the proposed method is applied to a set of real-world networks to study its' performance and accuracy. The first network used is the famous Zachery's Karate Club, first introduced in [Zachary 1977]. It models the friendship between students of a karate club in the early 1970s by using the social interactions between students inside and outside the club environment. During the data collection, an internal dispute between the teacher and the administration divided the students into two groups. Figure 19 shows the clustering result of the proposed method, which is indeed the correct community result for this network. None of the previous versions of this technique [Quiles *et al.* 2008, Silva and Zhao 2012] were able to identify the community of all nodes correctly.

In order to further test the proposed method as a data clustering technique, the proposed method is applied to 12 well known real-world data sets from the UCI machine learning repository [Asuncion and Newman 2007]. The data set metadata is available in table 2, which reports the number of instances, the data dimension, the number of classes and the size of the first 3 biggest classes in the data set. Although the class label of the data is available, its only used to verify each method's result (data clustering accuracy). The method is compared to 7

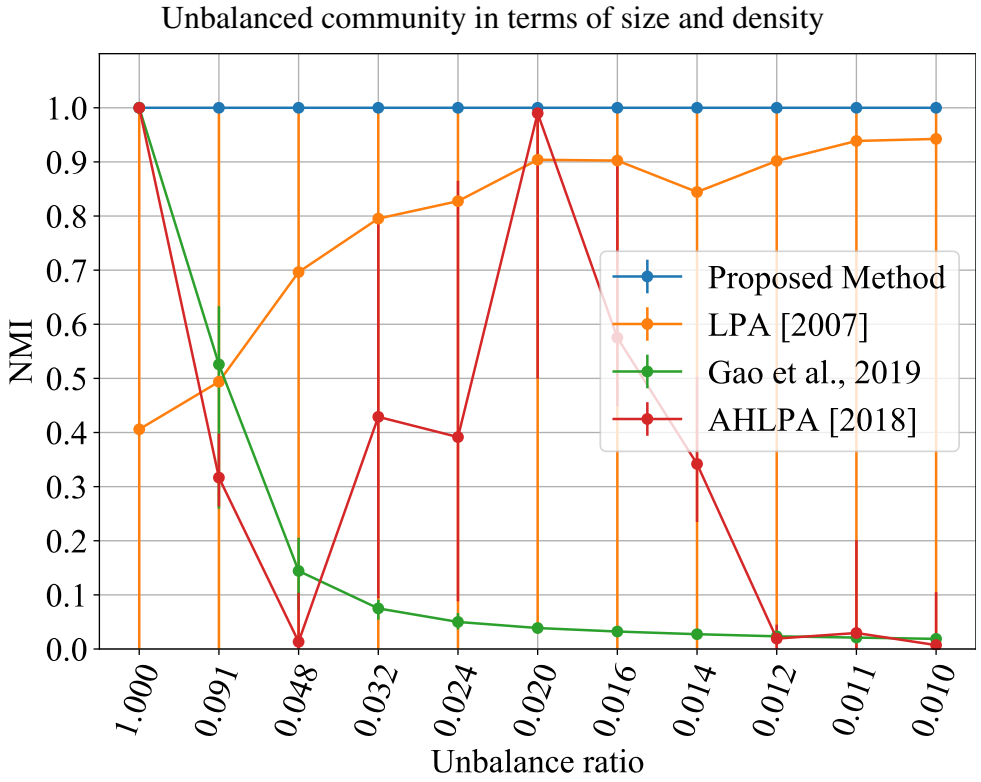

Figure 18 – Unbalanced community detection accuracy averaged over 100 executions with comparison to other three methods. All the methods are applied to a sequence of networks each containing two communities, the first of which containing 50 nodes and the second community varying from 50 to 5050 nodes. Since the first community has fixed size (50 nodes), the networks become more unbalance as the size of the second community increases (shown by x-axis). When $x = 0.010$, the second community is 100 times larger than the first one, i.e., 50 nodes vs. 5050 nodes.

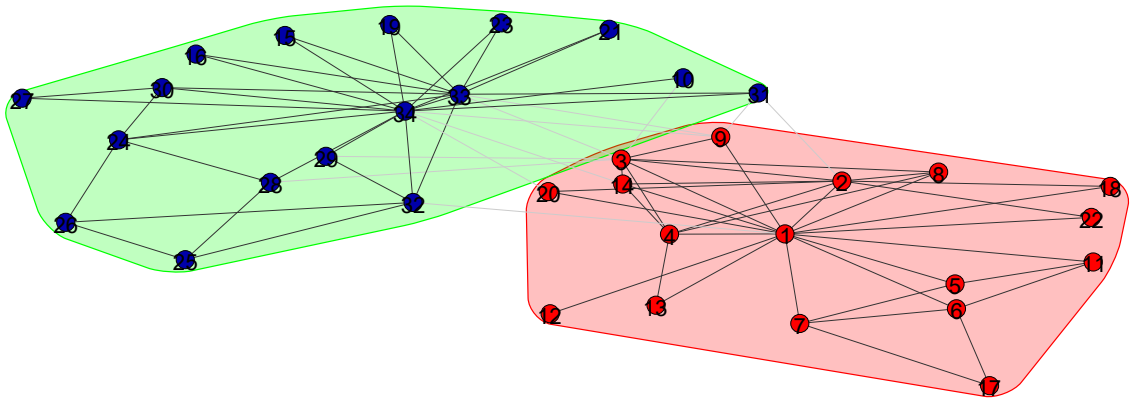

Figure 19 – Community detection result of the proposed method with $K = 2$ on the famous Zachery's karate club network. The proposed method is able to correctly identify the community structure of the network. Previously no Particle Competition model were able to correctly identify node 9.

well-know and established techniques: the Fast Greedy Modularity algorithm [Clauset, Newman and Moore 2004], Fuzzy C-Means [Bezdek 2013], expectation-maximization algorithm [Gupta, Chen *et al.* 2011], K-Means++ [Arthur and Vassilvitskii 2007], the Particle Competition [Silva and Zhao 2012] and the Label Propagation algorithm [Raghavan, Albert and Kumara 2007]. The Fast Greedy Modularity, Label Propagation, and Particle Competition algorithm [Silva and Zhao 2012] are all graph-based techniques and thus have been adapted to generate a graph from the given data set. For this task, the k-nearest neighbor graph formation technique was used with an optimized k for each data set.

Table 2 – UCI data set meta data

	Instances	**Dimension**	**Classes**	**Class ratio**
Iris	150	4	3	0.33, 0.33, 0.33
Breast Cancer	569	30	2	0.63, 0.37
Wine	178	13	3	0.40, 0.33, 0.27
Glass	214	10	6	0.36, 0.33, 0.14, ..
Ionosphere	351	34	2	0.64, 0.36
Vowel	990	13	11	0.09, 0.09, 0.09, ..
Yeast	1484	8	10	0.31, 0.29, 0.16, ..
Vertebral	310	6	3	0.48, 0.32, 0.19
Arrhythmia	452	279	13	0.54, 0.11, 0.10, ..
Parkinson	195	21	2	0.75, 0.25
Heart Disease	294	13	5	0.64, 0.13, 0.10, ..
E. Coli	336	7	8	0.43, 0.23, 0.15, ..

The proposed method parameters were optimized over the following parameters using a grid-search algorithm: $0 \leq \mu \leq 10$ and $0 \leq K \leq 30$ to identify the appropriate combination of parameters of each data set, resulting in the best accuracy. The results in terms of accuracy and standard deviation, are presented in Table 3, which are the average of 20 independent runs on each data set.

Additionally, the Table 3 also includes the average rank of each technique: this value is obtained by raking the technique average data clustering accuracy for each data set, i.e., the most accurate algorithm gets rank 1, the second most gets rank 2. The average rank is an indicator of the algorithm performance on the selected data sets. The proposed method has the highest rank, which indicates it performed better than others most of the time, and it is followed by the Silva & Zhao, 2012 [Silva and Zhao 2012].

To conclude this section and properly compare the techniques, we submit the empirical results reported on Table 3 to a statistical validation, in order to identify whether or not it is fair to state the proposed method performs better in terms of accuracy than the others. For this task of comparing multiple classifiers over several data sets, we employ the Friedman Test for hypothesis validation and later, Bonferroni-Dunn as a post-hoc test, with a fixed significance level of 5%, widely used in the literature [Demšar 2006]. However, as per [Demšar 2006], we also apply the

Table 3 – Comparison between data clusterization methods and the proposed algorithm in 20 independent runs

	Modularity	**Fuzzy C-Means**	**Exp-Max**	**K-Means++**	**Silva & Zhao, 2012**	**LP**	**Proposed Method**
Iris	85.92 ±0.00	87.97 ±0.00	77.58 ±0.13	83.62 ±0.60	83.38 ±0.46	86.95 ±8.97	90.13 ±7.60
Breast Cancer	60.26 ±0.00	75.04 ±0.00	54.02 ±0.00	82.91 ±0.27	89.04 ±0.33	84.72 ±5.00	88.13 ±0.00
Wine	81.08 ±0.00	71.38 ±0.00	69.83 ±0.00	96.20 ±0.00	94.20 ±1.22	81.05 ±2.96	96.17 ±0.15
Glass	77.61 ±0.00	84.33 ±0.00	73.35 ±0.00	74.81 ±3.28	79.01 ±2.33	76.51 ±5.94	79.37 ±2.03
Ionosphere	55.39 ±0.00	58.65 ±0.00	53.85 ±0.00	58.88 ±0.05	57.83 ±4.83	55.67 ±1.28	59.47 ±3.28
Vowel	88.37 ±0.00	83.94 ±0.12	77.73 ±3.14	84.65 ±0.37	90.31 ±0.16	90.86 ±0.07	90.32 ±0.08
Yeast	75.64 ±0.00	71.87 ±0.18	51.59 ±15.39	76.23 ±0.12	76.26 ±0.17	75.92 ±1.14	76.20 ±0.27
Vertebral	68.82 ±0.00	67.22 ±0.00	71.13 ±1.58	67.32 ±0.41	68.20 ±1.23	65.42 ±4.67	69.96 ±2.39
Arrhythmia	64.55 ±0.00	50.24 ±0.00	55.79 ±5.62	65.11 ±0.72	65.92 ±0.16	65.57 ±0.32	65.89 ±0.09
Parkinson	45.97 ±0.00	59.75 ±0.00	59.75 ±0.00	51.96 ±0.00	58.07 ±1.10	45.39 ±1.76	58.16 ±1.50
Heart Disease	56.93 ±0.00	53.72 ±0.00	57.38 ±6.09	57.32 ±0.60	57.78 ±1.67	56.01 ±0.79	57.98 ±1.67
E. Coli	79.90 ±0.00	79.12 ±0.19	57.60 ±20.75	80.52 ±1.14	80.31 ±2.09	81.36 ±3.76	85.98 ±0.24
Average Rank	4.75	4.66	5.58	3.83	2.75	4.5	1.83

approximation proposed on [Iman and Davenport 1980]. The Friedman Test's null-hypothesis is that all the techniques tested are equivalent and, therefore, their ranks should be the same. In order to verify such claim (null-hypothesis), we first find that we have $N = 11$ and $k = 12$, as we are testing 11 techniques over 12 data sets. The next step is to identify our critical value, which indicates the minimum value required on the F-distribution to reject the null hypothesis, with $k-1$ and $(k-1)(N-1)$ degrees of freedom. The critical value is $F(6,66) = 2.23$, with the parameters being the degrees of freedom previously described. Finally, by applying the results obtained, we find that $F_F = 5.54$. With $F_F > F(6,66)$, the null hypothesis is rejected with a 5% significance level. We may proceed to the post-hoc tests.

The post-hoc test compares how the proposed algorithm performs in comparison to others and allows us to identify whether or not the performance, in terms of accuracy, is significantly different. As previously stated, the Bonferroni-Dunn is chosen with the proposed method as the control technique, as it can correctly compare the performance of two or more methods [Demšar 2006]. In essence, the idea is to identify a critical difference (CD) in the ranking of methods. Should the distance between the rank of the proposed method and the rank of other techniques be higher than said CD, then we may say the proposed technique is statistically superior to such techniques. Otherwise, they do not present a significant difference, and the results presented at Table 3 is not enough to prove if the proposed method performs better than the techniques in comparison.

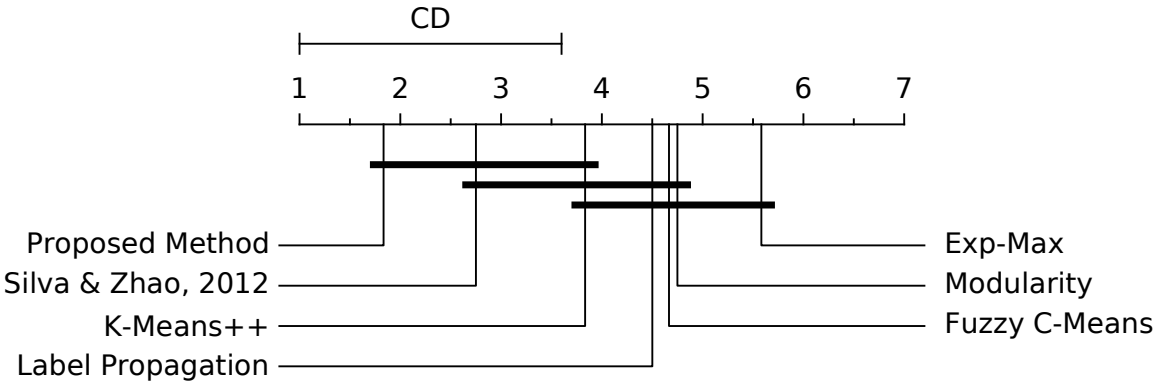

Figure 20 – Visualization of the results of the Bonferroni-Dunn test. The methods outside of the CD range from the proposed technique are said to be significantly different from the proposed method.

The results of the Bonferroni-Dunn are presented in Fig. 20. With CD$= 2.32$, it is possible to verify that the model performs better than Label Propagation, Fuzzy C-Means, Modularity, and Expectation Maximization at a statistically significant level. However, for Silva & Zhao, 2012, and K-Means, the differences are statistically insignificant.

Nevertheless, it should be noted that the technique presented in [Silva and Zhao 2012] still has a few disadvantages, mainly the inability to accurately cluster unbalanced communities. Moreover, as previously stated, the proposed method require fewer iterations in comparison to the original Particle Competition method, which improves its running times. The running times

Table 4 – Proposed Method and Silva & Zhao, 2012 average running time in seconds

	Proposed Method	**Silva & Zhao, 2012**
Iris	6.06s ±2.12s	23.40s ±2.43s
Breast Cancer	8.60s ±1.75s	63.00s ±5.16s
Wine	3.82s ±1.45s	21.05s ±1.36s
Glass	6.66s ±2.42s	37.03s ±3.57s
Ionosphere	7.15s ±1.74s	39.86s ±3.37s
Vowel	155.91s ±52.84s	637.16s ±18.25s
Yeast	78.62s ±24.62s	319.17s ±50.17s
Vertebral	23.68s ±8.76s	61.01s ±4.35s
Arrhythmia	71.26s ±22.64s	113.78s ±7.54s
Parkinson	2.68s ±0.55s	20.69s ±1.62s
Heart Disease	11.53s ±4.18s	51.36s ±4.80s
E. Coli	9.44s ±3.45s	57.27s ±1.93s

required to execute the set of tests that composes Table 3 is available in Table 4 – the tests were executed on a personal computer with a Core i7 8550U with 1.80GHz. The lower running time of the proposed method makes it more appropriate for larger networks.

CHAPTER

6

CONCLUSIONS

This document presents a new stochastic learning model for community detection and graph-based data-clustering consisting of two interacting learning steps: Competition and Regularization. Competition is a nature-inspired behavior which occurs when there is a lack of resources such as food or water. In a network, the particles compete with each other to try to dominate as many nodes as they possibly can. The Regularization step provides a guide for those particles: it introduces a parallel and diffusive mechanism into the model. This mechanism takes into account the domination-level presented in the neighborhood of each node to create a guide for each particle, tailored for the community it is currently trying to detect. The preference-level matrix, or guide, then helps the next execution of the Competition until a consensus is reached. The first step of the system is stochastic and thrives in exploring the unknown and trying new things – indeed, a necessary step for learning. The second step, however, builds upon the knowledge and experience obtained over time from the "unknown" stage to guide the learning process. The following *epoch* then combines both visions using the λ parameter, allowing for knowledge to be built.

Additionally, the proposed method employs a novelty combination of sequential and parallel diffusion of information. Such a mechanism is crucial to the obtain good community detection results in unbalanced networks and may be further improved or applied in other problems.

6.1 Conclusions

The computer simulations in this document show that the proposed method can achieve quite good results in data-clustering and community detection tasks. Furthermore, it shows that the proposed method improves previous Particle Competition models' results in terms of running time and, more importantly, in the ability to identify the community structure of networks with unbalanced communities. Indeed, the proposed method can obtain good community detection

accuracy when faced with networks with an unbalanced community structure, regardless of the unbalance ratio.

6.2 Limitations

In some cases, to correctly classify all the members of a community, it is not enough to execute the Regularization function only once. To correctly identify the "borders" of the community, the second stage must be repeatedly executed to propagate the labels and fix misclassifications. The proposed model uses a user-defined μ parameter in an attempt to satisfy such a necessity. However, the μ parameter has a significant impact on the model's performance, and it is usually hard to define. Although the proposed method provides excellent results and excels the original Particle Competition method in many tasks, its usage may be more difficult due to the selection of the appropriate value for the μ parameter.

6.3 Submissions during Master Period

The research project reported in this document resulted in the following submissions:

1. **Luan Martins and Liang Zhao (2020)**, "A Competitive and Diffusive Learning Model for Unbalanced Network Community Detection," *IEEE Transactions on Knowledge and Data Engineering*

2. **Luan Martins and Liang Zhao (2020)**, "Particle Competition for Unbalanced Community Detection in Complex Networks," *BRACIS 2020 (IX Brazilian Conference on Intelligent Systems)*

6.4 Future work

The usage of both stages has provided great results in community detection tasks and improved previous Particle Competition models both in terms of accuracy and running time. However, the proposed model may also be extended and modified so it can improve or work in the following set of problems:

- **Theoretical analysis**: although the proposed method has been empirically tested, a theoretical analysis of the model is still lacking.

- **Improved Regularization Equation**: the model's biggest limitation steams from the μ parameter, whose appropriate selection is crucial for the best clustering accuracy, especially in networks with very unbalanced community structure. One desirable improvement over the proposed technique is to replace the Regularization equation with a mechanism capable

of identifying when it is no longer necessary to continue iterating, thus, removing the μ parameter.

- **Semi-supervised classification**: the Particle Competition family of algorithms also has models for semi-supervised learning, in which a few of the node's labels (communities) are known before-hand and the model must use this information to identify which nodes share the same label. It would be interesting to see if the proposed model, working together with the semi-supervised version of the Particle Competition, can improve the results obtained by the Particle Competition alone, in terms of accuracy and computational speed.

- **One-class classification**: one can describe the stages of the proposed model as the Competition being the exploratory behavior of a particle in a network, which allows it to be creative and avoid pit-falls. In contrast, the Regularization stage is responsible for its decision making and constraint, which allows it to focus and defend a region. Perhaps, with some modifications, the combination of a set of cooperating particles and a modified Regularization mechanism (which would consider that there is only one type of particle cooperating) would accurately make graph-based one-class classifications by identifying only the community (label) of the given class.

BIBLIOGRAPHY

ARTHUR, D.; VASSILVITSKII, S. k-means++: The advantages of careful seeding. In: SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS. **Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms**. [S.l.], 2007. p. 1027–1035. Citation on page 66.

ASUNCION, A.; NEWMAN, D. **UCI machine learning repository**. 2007. Citation on page 64.

BAGROW, J. P.; BOLLT, E. M. Local method for detecting communities. **Physical Review E**, APS, v. 72, n. 4, p. 046108, 2005. Citation on page 35.

BEZDEK, J. C. **Pattern recognition with fuzzy objective function algorithms**. [S.l.]: Springer Science & Business Media, 2013. Citation on page 66.

BLONDEL, V. D.; GUILLAUME, J.-L.; LAMBIOTTE, R.; LEFEBVRE, E. Fast unfolding of communities in large networks. **Journal of statistical mechanics: theory and experiment**, IOP Publishing, v. 2008, n. 10, p. P10008, 2008. Citation on page 34.

BOCCALETTI, S.; LATORA, V.; MORENO, Y.; CHAVEZ, M.; HWANG, D.-U. Complex networks: Structure and dynamics. **Physics reports**, Elsevier, v. 424, n. 4-5, p. 175–308, 2006. Citation on page 22.

BRANDES, U.; DELLING, D.; GAERTLER, M.; GORKE, R.; HOEFER, M.; NIKOLOSKI, Z.; WAGNER, D. On modularity clustering. **IEEE transactions on knowledge and data engineering**, IEEE, v. 20, n. 2, p. 172–188, 2007. Citation on page 34.

BRITO, M.; CHAVEZ, E.; QUIROZ, A.; YUKICH, J. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. **Statistics & Probability Letters**, Elsevier, v. 35, n. 1, p. 33–42, 1997. Citation on page 29.

CABREROS, I.; ABBE, E.; TSIRIGOS, A. Detecting community structures in hi-c genomic data. In: IEEE. **2016 Annual Conference on Information Science and Systems (CISS)**. [S.l.], 2016. p. 584–589. Citation on page 21.

CHUAN, P. M.; ALI, M.; KHANG, T. D.; DEY, N. *et al.* Link prediction in co-authorship networks based on hybrid content similarity metric. **Applied Intelligence**, Springer, v. 48, n. 8, p. 2470–2486, 2018. Citations on pages 21 and 23.

CLAUSET, A.; NEWMAN, M. E.; MOORE, C. Finding community structure in very large networks. **Physical review E**, APS, v. 70, n. 6, p. 066111, 2004. Citation on page 66.

COSTA, L. d. F.; JR, O. N. O.; TRAVIESO, G.; RODRIGUES, F. A.; BOAS, P. R. V.; ANTIQUEIRA, L.; VIANA, M. P.; ROCHA, L. E. C. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. **Advances in Physics**, Taylor & Francis, v. 60, n. 3, p. 329–412, 2011. Citation on page 21.

COSTA, L. d. F.; RODRIGUES, F. A.; TRAVIESO, G.; BOAS, P. R. V. Characterization of complex networks: A survey of measurements. **Advances in physics**, Taylor & Francis, v. 56, n. 1, p. 167–242, 2007. Citation on page 22.

CULP, M.; MICHAILIDIS, G. Graph-based semisupervised learning. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 30, p. 174 – 179, 2008. Citation on page 24.

DANON, L.; DÍAZ-GUILERA, A.; ARENAS, A. The effect of size heterogeneity on community identification in complex networks. **Journal of Statistical Mechanics: Theory and Experiment**, IOP Publishing, v. 2006, n. 11, p. P11010, 2006. Citation on page 23.

DANON, L.; DIAZ-GUILERA, A.; DUCH, J.; ARENAS, A. Comparing community structure identification. **Journal of Statistical Mechanics: Theory and Experiment**, IOP Publishing, v. 2005, n. 09, p. P09008, 2005. Citations on pages 12, 51, 55, 61, and 62.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine learning research**, v. 7, n. Jan, p. 1–30, 2006. Citations on pages 66 and 68.

DUCH, J.; ARENAS, A. Community detection in complex networks using extremal optimization. **Physical review E**, APS, v. 72, n. 2, p. 027104, 2005. Citation on page 34.

F.A.N., U. P. V.; L., Z. Network unfolding map by vertex-edge dynamics modeling. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 29, n. 2, p. 405–418, 2018. Citation on page 37.

FORTUNATO, S. Community detection in graphs. **Physics reports**, Elsevier, v. 486, n. 3-5, p. 75–174, 2010. Citations on pages 22, 23, 24, 28, and 30.

FORTUNATO, S.; BARTHELEMY, M. Resolution limit in community detection. **Proceedings of the national academy of sciences**, National Acad Sciences, v. 104, n. 1, p. 36–41, 2007. Citation on page 23.

G., V. K. N. **Stochastic Processes in Physics and Chemistry, revised and enlarged edition**. [S.l.]: North-Holland, Amsterdam, 1992. Citation on page 24.

GAO, X.; ZHENG, Q.; VERRI, F. A.; RODRIGUES, R. D.; ZHAO, L. Particle competition for multilayer network community detection. In: **Proceedings of the 2019 11th International Conference on Machine Learning and Computing**. [S.l.: s.n.], 2019. p. 75–80. Citations on pages 25, 50, 62, and 63.

GIRVAN, M.; NEWMAN, M. E. Community structure in social and biological networks. **Proceedings of the national academy of sciences**, National Acad Sciences, v. 99, n. 12, p. 7821–7826, 2002. Citations on pages 22, 30, and 31.

GRADY, L. Random walks for image segmentation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 28, p. 1768 – 1783, 2006. Citation on page 24.

GUIMERA, R.; AMARAL, L. A. N. Functional cartography of complex metabolic networks. **nature**, Nature Publishing Group, v. 433, n. 7028, p. 895–900, 2005. Citation on page 29.

GUIMERA, R.; DANON, L.; DIAZ-GUILERA, A.; GIRALT, F.; ARENAS, A. Self-similar community structure in a network of human interactions. **Physical review E**, APS, v. 68, n. 6, p. 065103, 2003. Citation on page 23.

GUIMERA, R.; SALES-PARDO, M.; AMARAL, L. A. N. Modularity from fluctuations in random graphs and complex networks. **Physical Review E**, APS, v. 70, n. 2, p. 025101, 2004. Citation on page 34.

GUPTA, M. R.; CHEN, Y. *et al.* Theory and use of the em algorithm. **Foundations and Trends® in Signal Processing**, Now Publishers, Inc., v. 4, n. 3, p. 223–296, 2011. Citation on page 66.

H., R. **The Fokker–Planck Equation**. [S.l.]: Springer, 1984. Citation on page 24.

IMAN, R. L.; DAVENPORT, J. M. Approximations of the critical region of the fbietkan statistic. **Communications in Statistics-Theory and Methods**, Taylor & Francis, v. 9, n. 6, p. 571–595, 1980. Citation on page 68.

JAVED, S.; MAHMOOD, A.; FRAZ, M. M.; KOOHBANANI, N. A.; BENES, K.; TSANG, Y.-W.; HEWITT, K.; EPSTEIN, D.; SNEAD, D.; RAJPOOT, N. Cellular community detection for tissue phenotyping in colorectal cancer histology images. **Medical Image Analysis**, Elsevier, p. 101696, 2020. Citations on pages 23 and 29.

KAWASAKI, R. K.; IKEDA, Y. Network analysis of attitudes towards immigrants in asia. **arXiv**, p. arXiv–2004, 2020. Citation on page 29.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **science**, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983. Citation on page 34.

LANDAU, L.; LIFSHITZ, E. **Statistical Physics**. [S.l.]: Butterworth-Heinemann, 1980. Citation on page 24.

MIHALCEA, R.; RADEV, D. **Graph-based natural language processing and information retrieval**. [S.l.]: Cambridge university press, 2011. Citation on page 28.

NEWMAN, M. **Networks**. [S.l.]: Oxford university press, 2018. Citations on pages 22 and 28.

NEWMAN, M. E. Fast algorithm for detecting community structure in networks. **Physical review E**, APS, v. 69, n. 6, p. 066133, 2004. Citation on page 34.

____. Finding community structure in networks using the eigenvectors of matrices. **Physical review E**, APS, v. 74, n. 3, p. 036104, 2006. Citation on page 35.

NEWMAN, M. E.; GIRVAN, M. Finding and evaluating community structure in networks. **Physical review E**, APS, v. 69, n. 2, p. 026113, 2004. Citations on pages 30, 31, 33, and 54.

PALLA, G.; DERÉNYI, I.; FARKAS, I.; VICSEK, T. Uncovering the overlapping community structure of complex networks in nature and society. **nature**, Nature Publishing Group, v. 435, n. 7043, p. 814, 2005. Citation on page 23.

PAVLIOTIS, G. A. **Stochastic Processes and Applications Diffusion Processes, the Fokker-Planck and Langevin Equations**. [S.l.]: Springer-Verlag New York, 2014. Citation on page 24.

QUILES, M. G.; ZHAO, L.; ALONSO, R. L.; ROMERO, R. A. Particle competition for complex network community detection. **Chaos: An Interdisciplinary Journal of Nonlinear Science**, AIP, v. 18, n. 3, p. 033107, 2008. Citations on pages 22, 25, 37, 39, 57, 61, and 64.

RAGHAVAN, U. N.; ALBERT, R.; KUMARA, S. Near linear time algorithm to detect community structures in large-scale networks. **Physical review E**, APS, v. 76, n. 3, p. 036106, 2007. Citations on pages 24, 36, 62, and 66.

RATTIGAN, M. J.; MAIER, M.; JENSEN, D. Using structure indices for efficient approximation of network properties. In: **Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2006. p. 357–366. Citation on page 31.

____. Graph clustering with network structure indices. In: **Proceedings of the 24th international conference on Machine learning**. [S.l.: s.n.], 2007. p. 783–790. Citation on page 31.

ROWE, R.; CREAMER, G.; HERSHKOP, S.; STOLFO, S. J. Automated social hierarchy detection through email network analysis. In: **Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis**. [S.l.: s.n.], 2007. p. 109–117. Citation on page 21.

RUCCI, M.; VICTOR, J. D. Random walks for image segmentation. **Trends in Neurosciences**, v. 38, p. 195 – 206, 2015. Citation on page 24.

SHIELDS, R. Cultural topology: The seven bridges of königsburg, 1736. **Theory, Culture & Society**, Sage Publications Sage UK: London, England, v. 29, n. 4-5, p. 43–57, 2012. Citation on page 28.

SILVA, T. C.; ZHAO, L. Network-based high level data classification. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 23, n. 6, p. 954–970, 2012. Citation on page 21.

____. Stochastic competitive learning in complex networks. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 23, n. 3, p. 385–398, 2012. Citations on pages 22, 23, 25, 29, 37, 39, 41, 45, 46, 48, 50, 57, 61, 62, 64, 66, and 68.

____. **Machine Learning in Complex Networks**. [S.l.]: Springer, 2016. Citation on page 24.

SIMEONOVSKI, M.; PELLEGRINO, G.; ROSSOW, C.; BACKES, M. Who controls the internet? analyzing global threats using property graph traversals. In: **Proceedings of the 26th International Conference on World Wide Web**. [S.l.: s.n.], 2017. p. 647–656. Citations on pages 21 and 23.

TYLER, J. R.; WILKINSON, D. M.; HUBERMAN, B. A. E-mail as spectroscopy: Automated discovery of community structure within organizations. **The Information Society**, Taylor & Francis, v. 21, n. 2, p. 143–153, 2005. Citation on page 31.

W., G. N.; N., R.-D. **Stochastic Models in Biology**. [S.l.]: Academic Press, 1974. Citation on page 24.

WAN, M.; OUYANG, Y.; KAPLAN, L.; HAN, J. Graph regularized meta-path based transductive regression in heterogeneous information network. In: SIAM. **Proceedings of the 2015 SIAM International Conference on Data Mining**. [S.l.], 2015. p. 918–926. Citation on page 21.

WANG, H.; ZHANG, F.; HOU, M.; XIE, X.; GUO, M.; LIU, Q. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In: ACM. **Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining**. [S.l.], 2018. p. 592–600. Citation on page 21.

WANG, Y.; HU, M.; LI, Q.; ZHANG, X.-P.; ZHAI, G.; YAO, N. Abnormal respiratory patterns classifier may contribute to large-scale screening of people infected with covid-19 in an accurate and unobtrusive manner. **arXiv preprint arXiv:2002.05534**, 2020. Citations on pages 21 and 23.

WU, F.; ZHANG, T.; JR, A. H. d. S.; FIFTY, C.; YU, T.; WEINBERGER, K. Q. Simplifying graph convolutional networks. **arXiv preprint arXiv:1902.07153**, 2019. Citation on page 24.

YANG, L.; CAO, X.; HE, D.; WANG, C.; WANG, X.; ZHANG, W. Modularity based community detection with deep learning. In: **IJCAI**. [S.l.: s.n.], 2016. v. 16, p. 2252–2258. Citation on page 34.

ZACHARY, W. W. An information flow model for conflict and fission in small groups. **Journal of anthropological research**, University of New Mexico, v. 33, n. 4, p. 452–473, 1977. Citation on page 64.

ŽALIK, K. R.; ŽALIK, B. A framework for detecting communities of unbalanced sizes in networks. **Physica A: Statistical Mechanics and Its Applications**, Elsevier, v. 490, p. 24–37, 2018. Citation on page 31.

ZHANG, S.; ZHAO, H. Community identification in networks with unbalanced structure. **Physical Review E**, APS, v. 85, n. 6, p. 066114, 2012. Citation on page 35.

ZHOU, D.; BOUSQUET, O.; LAL, T. N.; WESTON, J.; SCHÖLKOPF, B. Learning with local and global consistency. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2004. p. 321–328. Citation on page 24.

ZHOU, X.; YANG, K.; XIE, Y.; YANG, C.; HUANG, T. A novel modularity-based discrete state transition algorithm for community detection in networks. **Neurocomputing**, Elsevier, v. 334, p. 89–99, 2019. Citation on page 35.

ZHU, X.; XIA, Z. Label propagation algorithm based on adaptive h index. In: SPRINGER. **International Conference on Data Mining and Big Data**. [S.l.], 2018. p. 53–64. Citations on pages 24, 36, and 62.


ICMC
USP
SÃO CARLOS