Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Adaptive Model to Community Detection in Dynamic Social Networks

Model Adaptativo para Detecção de Comunidades em Redes Sociais Dinâmicas

Aurélio Ribeiro Costa

Tese apresentada como requisito parcial
para conclusão do Doutorado em Informática

Orientadora
Prof.ª Dr.ª Célia Ghedini Ralha

Brasília
2023

CIP — Catalogação Internacional na Publicação

Ribeiro Costa, Aurélio.

Adaptive Model to Community Detection in Dynamic Social Networks

Model Adaptativo para Detecção de Comunidades em Redes Sociais Dinâmicas / Aurélio Ribeiro Costa. Brasília : UnB, 2023.
94 p. : il. ; 29,5 cm.

Tese (Doutorado) — Universidade de Brasília, Brasília, 2023.

1.

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Adaptive Model to Community Detection in Dynamic Social Networks

Model Adaptativo para Detecção de Comunidades em Redes Sociais Dinâmicas

Aurélio Ribeiro Costa

Tese apresentada como requisito parcial
para conclusão do Doutorado em Informática

Brasília, 16 de junho de 2023

Abstract

A vital problem tackled in network analysis is community structure identification. However, the current use of network analysis techniques concentrates on analyzing static community structures, which generates a research gap not considering the dynamic aspects of these structures. Some solutions for the community detection problem adapted to the dynamicity of the networks present limitations on the resulting performance, and others do not fit such contexts. This situation aggravates when considering the demand to analyze constantly growing social networks. This research aims to fill this gap by focusing on the topology change over time. We propose an adaptive model with an actor-critic reinforcement learning-based architecture to maximize the local modularity density of a community structure using a graph neural network to cope with changing aspects of large social networks. Extensive experiments conducted using the Actor–Critic for Community Detection (AC2CD) with real-world dynamic social network datasets show better accuracy when compared to the state-of-the-art solutions. Further investigation concluded that the architecture copes well with real-world social networks, even considering networks with unbalancing community sizes.

Keywords: Network Analysis, Community Detection, Reinforcement Learning

Resumo Extendido

Um problema crucial abordado na análise de redes é a identificação da estrutura da comunidade. Essa estrutura representa a associação dos vértices de uma rede a conjuntos ou comunidades. No entanto, as técnicas atuais de análise de redes se concentram na análise de estruturas estáticas de comunidades, o que cria uma lacuna de pesquisa que não leva em consideração os aspectos dinâmicos dessas estruturas. Algumas soluções para o problema de detecção de comunidades adaptadas à dinamicidade das redes apresentam limitações de desempenho, enquanto outras não se enquadram nesses contextos. Essa situação se agrava à medida que a demanda de análise de redes sociais continua crescendo constantemente.

No que diz respeito às classes de solução para o problema de Community Detection (CD), podemos separá-las em duas abordagens: as clássicas, que incluem otimização da modularidade, *Random Walk*, propagação, entre outros, e as não clássicas, como aquelas baseadas *graph embedding*, modelagem estatística e aprendizado de máquina.

A metodologia utilizada nesta pesquisa inclui uma revisão da literatura, definição do problema de pesquisa, delimitação do escopo do trabalho, desenvolvimento arquitetural do modelo de solução, implementação do modelo, validação por meio de experimentos utilizando redes sociais online (Online Social Network (OSN)) e redação de artigos científicos em conferências e periódicos da área de Ciência da Computação. A revisão da literatura seguiu o protocolo definido por Kitchenham (2004), compreendendo três fases: planejamento (*planning*), condução (*conducting*) e relato da revisão (*reporting the review*). A ferramenta Parsifal (Freitas, 2014) foi utilizada para realização da revisão de literatura.

Os repositórios da *ACM Digital Library*, *IEEE Xplore* e *Springer lInk* foram utilizados, considerando o período de 2015 a 2020 e atualização até 2023. As publicações selecionadas incluíram trabalhos em conferências e periódicos da área de Ciência da Computação. Foram identificados inicialmente 49 trabalhos, dos quais 29 foram excluídos por não atenderem aos critérios de inclusão, 3 estavam duplicados e 17 foram aceitos com base nas características definidas pela PICO (*Population, Intervention, Comparison, Outcome*). Durante a atualização, dois trabalhos de 2023 foram incluídos, totalizando 19 trabalhos como resultado da revisão de literatura.

É importante salientar que alguns trabalhos não utilizam técnicas de Inteligência Artificial, mas houve um aumento significativo de interesse a partir de 2019, principalmente no uso de aprendizado de máquina e aprendizado de máquina profundo. No entanto, mesmo nos trabalhos a partir de 2019, não foram encontradas abordagens de CD para lidar com OSN ou redes dinâmicas. Com base nos resultados da revisão da literatura, identificou-se uma lacuna na área de encontrar comunidades em redes dinâmicas desbalanceadas.

Esta pesquisa visa preencher essa lacuna, com foco na mudança de topologia ao longo do tempo, aplicando aprendizado por reforço profundo ao problema de detecção de comunidades em redes sociais dinâmicas. Propomos um modelo adaptativo com uma arquitetura de ator-crítico (*actor-critic*) baseada em aprendizado por reforço. A proposta visa maximizar a densidade de modularidade local de uma estrutura de comunidade, utilizando uma rede neural de grafos para lidar com aspectos mutáveis de grandes redes sociais. Extensos experimentos conduzidos com a arquitetura denominada *Actor-Critic to Community Detection* (AC2CD), utilizando conjuntos de dados dinâmicos de redes sociais do mundo real, mostraram maior precisão em comparação com as soluções apresentadas na literatura. Uma investigação mais aprofundada concluiu que a arquitetura lida bem com as redes sociais do mundo real, mesmo considerando redes com tamanhos de comunidade desbalanceados.

Nossa proposta consiste na combinação de aprendizado por reforço profundo (Deep Reinforcement Learning (DRL)) com rede de atenção em grafo (Graph Attention Network (GAT)) para realizar a descoberta de comunidades em redes dinâmicas. Foi implementada uma arquitetura baseada em *actor-critic*, que foi validada por meio de experimentos usando conjunto de dados (*datasets*) que representam redes reais. A arquitetura implementada é composta por dois componentes: o ator *actor* e o crítico *critic*. Em cada componente foram utilizadas duas camadas de atenção em grafos como camadas ocultas, uma camada de *dropout* na entrada e uma camada de *softmax* na saída.

Os experimentos realizados para validar o modelo utilizaram cinco *datasets* representando redes reais: *High School, BlogCatalog3, Email-EU-Core, Flicker e Youtube2*. A avaliação dos resultados foi feita com base em medidas de *F-measure*, como *macro-F1*, *micro-F1*, além da informação mútua normalizada (Normalized Mutual Information (NMI)). Os resultados obtidos confirmaram a hipótese, demonstrando que o modelo superou as soluções da literatura, como GraphGAN, ComE, SDNE, CLARE e CNN, especialmente em termos de estabilidade diante da complexidade de conjuntos de dados como *Flickr* ou *Youtube*.

Também foi conduzido um experimento para avaliar o desempenho ao utilizar duas implementações de redes neurais em grafo (Graph Neural Network (GNN)), GAT e rede convolucional em grafo (Graph Convolution Network (GCN)). Foi observado que ambas

implementações são equivalentes para o problema de CD, com uma variação de no máximo 0.08 na métrica de NMI no conjunto de dados *High School*. Além disso, o desempenho alcançado é em grande parte atribuído ao uso de modelo de aprendizado por reforço com arquitetura *actor-critic*.

Palavras-chave: Análise de rede, detecção de comunidade, aprendizado por reforço

Dedication

I dedicate this entire research to my two daughters, Ana Isadora, and Ana Júlia because they were the ones who, even without knowing it, encouraged me to complete it; my wife, Andressa, an inseparable companion even when I needed to be absent to complete this work. I also dedicate this work to my mother, Sônia, who founded the search for knowledge back then, and to my father, Álvaro.

Acknowledgements

First, I am very grateful to my advisor Prof. Celia Ghedini Ralha, for the perfect balance between charge and support. I also thank you for believing that our efforts would bring results one day.

I thank the participants of the InfoKnown group, especially Natan, Lucas, and Vanessa, for the partnership and idea even before this research began.

Thanks to Flávio Henrique for his comments, suggestions, and the idea of taking a break from work to complete the last round of experiments and finish writing this document.

Thanks also to Prof. Thiago for kindly providing the infrastructure for our case studies. This infrastructure is part of the KnEDLe project[1] funded by the Fundação de Apoio à Pesquisa do Distrito Federal (FAPDF), in partnership with the University of Brasília and the Fundação de Empreendimentos Científicos e Tecnológicos (FINATEC).

[1]KnEDLe project website `https://unb-knedle.github.io/`

List of Figures

List of Tables

Acronyms

AC2CD Actor–Critic for Community Detection. iv, xi, xii, 6–8, 39, 40, 44–48, 53, 58, 60, 62, 64–66

AI Artificial Intelligence. 19, 22, 32

CD Community Detection. v–vii, 1–13, 15, 17, 19, 28–30, 32, 33, 39, 47, 53–55, 62, 65–67

CLARE Community Locator And community REwriter. 7, 53, 56–58, 60, 64–66

CNN Convolutional Neural Network. 18–20

COO Coordinate Format. 41–43

CTDG Continuous-Time Dynamic Graph. 10

DGNN Dynamic Graph Neural Network. 33

DRL Deep Reinforcement Learning. vi, 5–8, 27, 40, 56, 57, 67, 68

DSCPCD Dual Structural Consistency Preserving CD. 32

DTDG Discrete-Time Dynamic Graph. 10, 11, 41

GAE Generalized Advantage Estimation. 40

GAN Generative Adversarial Network. 33, 53

GAT Graph Attention Network. vi, 5, 6, 19, 20, 39, 40, 45, 65–67

GCN Graph Convolution Network. vi, 5, 19, 20, 33, 39, 40, 53, 57, 58, 60, 62, 65–67

GMM Gaussian Mixture Model. 55

GNN Graph Neural Network. vi, 5–7, 15, 19, 20, 33, 39, 40, 45, 64–67

LSTM Long Short-Term Memory. 36

MC Monte Carlo. 24

MDP Markov Decision Process. 2, 22–24, 39

ML Machine Learning. 3, 5, 6, 11, 15, 16, 23, 32, 54, 64, 67

MRF Markov Random Fields. 19, 68

NA Network Analysis. 1, 4, 9, 32, 33

NMI Normalized Mutual Information. vi, vii, 6, 7, 22, 53, 64, 65

NN Neural Network. 16, 18

OSN Online Social Network. v, vi, 3, 6, 7, 19, 29, 39, 67, 68

PPO Proximal Policy Optimization. 24, 25, 27, 39, 40, 45

QAC Quality Assessment Checklist. 30

RL Reinforcement Learning. 5, 6, 8, 9, 15, 22, 23, 26, 28–30, 33, 39, 44, 45, 67

RLVECN Representation Learning Via knowledge-graph Embeddings and ConvNet. 32, 33

RNN Recurrent Neural Network. 21, 36

SBM Stochastic Block Model. 15, 17

SEAL Seed Expansion with generative Adversarial Learning. 33

SGD Stochastic Gradient Descent. 17

TD Temporal Difference. 24, 26, 27, 44

TRPO Trust Region Policy Optimization. 24, 25

Contents

Chapter 1

Introduction

One can view a network as a group of tied entities [1]. In this way, many interactions of the natural world can be modeled as a network, such as relations between people [2–4], proteins interactions [5], fraud detection [6] and supply chains [7]. People can classify these networks by many criteria, dynamic or static, regular, complex, or random, among other measures. Network Analysis (NA) is a discipline that aims to evaluate the target network in different aspects like node classification, link prediction, or Community Detection (CD). Each of these aspects might be analyzed alone or in an integrated way.

The real-world networks exhibit significant irregularities in terms of the degree of nodes and the distribution of edges, bringing out a high level of organization of the network. The distribution inhomogeneity of the edges connecting nodes results from a high edge density within special groups of nodes and low tightness between the nodes across different special groups. These special groups or subgraphs are called communities or clusters within the network. The nodes of the same community expect to have common interests or similarities. Disclosing these communities reveals the intercourse between the network's structure and functionality [8].

1.1 Problem

The CD problem has become one of the main pillars of network science research and has no canonical solution. In the big-data era, complex networks are an essential field of study [9]. Newman and Girvan [1] formulate the CD problem as finding groups of nodes densely connected inside these groups and sparsely connected among groups. According to [10], a valuable area in the study of complex networks is CD. CD views networks as graphs and tries to find nodes more firmly attached than the others.

Schaeffer [11] defines graph clustering as finding sets of *related* vertices in graphs and notes that in some of the clustering literature, a cluster in a graph is called a community

[12, 13]. Formally, given a graph, clustering aims to divide it into sets such that the elements assigned to a particular group are similar or connected in some predefined sense.

A clique is a completely connected subgraph. Considering the connectivity aspect, the loosest possible definition of a graph cluster is a connected component (i.e., a subgraph with a path between any of its nodes). The strictest definition is that each cluster should be a maximal clique (i.e., a subgraph into which one could add no vertex without losing the clique property). In most occasions, the semantically sound clusters lie between these two extremes. We can compute connected components in $\mathcal{O}(n+m)$-time with a breadth-first search, whereas clique detection is NP-complete [14]. The formalization of CD leads to NP-complete problems, which constrains us to heuristic solutions. When the input graph is large, relying on algorithms with exponential running time is highly infeasible, as even linear time computation gets tedious.

In a nutshell, the CD process of a given network typically begins with a scoring function (e.g., modularity density) that quantifies the intuition that communities correspond to densely linked sets of nodes. Then one applies a procedure to find groups of nodes with a high value of the scoring function. Identifying such communities in networks has proven to be a challenging task when considering the following aspects:

- The conceptual differences between distinct perspectives on CD emerge enormous technical details of different algorithmic implementations, approaches, methods, and solutions.
- There exist multiple structural definitions of network communities (e.g., modularity-based or similarity-based communities) [15].
- The growing size of the networks we want to unveil their communities. This aspect worsens when considering dynamic networks where one must store contextual information for temporal analysis.
- The lack of reliable ground truth evaluates a CD solution extremely difficult [16].

A probabilistic way to model the CD problem is through a Markov Decision Process (MDP). We can define an MDP by states and actions for transitioning between states. Given a network, one optimal community structure as a state of an MDP, and a quality function (e.g., modularity density) to evaluate each state, one can assign a node in this network to a community as an action and calculate the quality of this new state, repeating this process for all nodes. By Bellman's, the quality function value is optimized as each node assignment to improve the community structure. In this way, a terminal state can be a set of terms between nodes and communities that maximize the modularity density and may coincide with the optimal community structure initially proposed.

A large interdisciplinary community of scientists has been working on the CD problem proposing many methods for distinct complex networks, including complex dynamic networks. However, the proposed solutions present some constraints as the type of edges (directed or not) and heterogeneity of groups in the network. In the case of dynamic networks, the more common approach is to take snapshots of the network and apply techniques for static networks. There have been surveys of CD in graphs and networks from 2005 until recently [17–20].

The authors in [21] provide a focused review of the motivations that underpin CD. In this research, we will focus on the dynamic aspects of networks when trying to detect communities. The following section describes the reasons for conducting this research and provides a brief view of the literature's open questions.

1.2 Motivation

Although the research of CD solutions may seem mature, in recent years, we perceive a growing volume of publications seeking to improve research on CD solutions performance towards the usage of high volume datasets [3] or Machine Learning (ML) techniques to enhance the quality of scoring response [22–24]. However, as [25] indicates, the aggregating topological and content information can enable a more informative CD, in which cues from different sources integrate into more powerful models to generate more insights about the network behavior.

Recent advances in network science have brought out the importance of complex networks in many different domains, such as sociology (acquaintance networks, collaboration networks), biology (metabolic networks, gene networks), fraud detection (networks of communications toward crime eradication), supply chain management, and computer science (internet topology, Web graph, P2P networks). In general, the associated networks are globally sparse but locally dense, i.e., there are groups of nodes called communities, highly connected but with few links to other nodes. This kind of structure brings out much information about the network. For example, in a metabolic network, the communities correspond to the biological functions of the cell. In the Web graph, the communities correspond to topics of interest [26].

The problem of CD has a long history since researchers tried to understand people relations using mathematical approaches [27]. But sociology has paid much attention to Online Social Network (OSN) use. One instance of CD applied to sociology is the problem of identifying a relevant population of actors in a study of how information or new ideas diffuse through a community [2].

Biology is another research field that uses CD when dealing with protein interactions. The authors in [28] assert that densely connected proteins form the mass of biological processes. The protein-protein interaction network contains the communications among the protein groups that communicate closely, being used to predict the complexity or function of regular proteins. The structures of protein-protein interaction networks can reflect some principles of cellular organization. Computational identification of particular protein molecules is vital in understanding protein function. The community structure can reflect the community's topological relations directly. The real-world communities, such as protein-protein interaction networks in biology and World Wide Web networks in sociology, tend to follow the heavy-tailed power law that only a tiny amount of the nodes' degrees is higher than the rest. Therefore, we can apply CD to various research fields, including biology.

Another CD application emerges from the fraud detection domain. For instance, [29] describe NA applied to networks of organizational communications (e.g., Enron company dataset). Analysis of the frequency and direction of formal/informal email communication can reveal communication patterns among employees and managers. These patterns can help identify people engaged in fraudulent activities, promoting the adoption of more efficient forms of action toward crime eradication.

Resources management, mainly supply chain management, is another field in that CD can lead to a positive outcome. The authors in [7] advocate that a community within a supply chain is a set of firms clustered around similar interests or functions. That is, communities are bound together in clusters predominantly connected by horizontal relationships amongst firms with similar interests and processes. However, that is not to say that all firms within a community are entirely cooperative. The presence of horizontal connections provides the essence of a community. More efficient supply chains will possess communities that allow for improved horizontal information flow and innovation diffusion. Thus, vertical relationships between communities must form and maintain for supply chain systems to function efficiently from initial suppliers to final consumers. However, the transaction costs of inter-community connections vertically arranged are likely to be considerably higher than those observed with intra-community (i.e., predominantly horizontal relationships) because of each community's differing interests and functions. In this way, CD can model firms' interactions giving a better insight into inter-community communication.

Understanding the structure of Al-Qaeda is critical in fighting the war on terror and could help prevent future events such as another September 11 attack. Possessing an ecological map of a food chain will help keep environments stable. Because of limited resources, understanding the shipping merchant marine vessels traverse as they conduct international trade is vital to protecting ports of call. Understanding how a network of satellites connects to various world locations is critical for a global company's bottom line. A financial network such as the Enron fraud to destroy the entire company and make a lifetime's retirement fund disappear in a day is vital [30].

The different research scenarios described demonstrate the power of the CD application to solve real-life problems. Moreover, when considering social network data, as this research, one can take an aggregated insight into different areas, for example, fraud detection in the supply chain.

In this context, this research proposes an adaptive model for the problem of identifying these groups of highly connected nodes. The proposed model uses an architecture based on Reinforcement Learning (RL) and Graph Neural Network (GNN). Moreover, we consider the CD problem in the context of dynamic networks, a complex network with topology changing over time. Thus, we believe a solution to the CD problem that contemplates an adaptive model to consider this integrated scenario is relevant. Besides, if this adaptive model grasps the network dynamics, it can track the evolving relationships. Thus, the result will resemble the reality resulting in a more accurate community identification.

1.3 Hypothesis

The hypothesis held in this work is the application of the DRL approach to continually improve the modularity density of a community structure dealing with dynamic social networks. Compared to the state-of-the-art CD solutions, RL seems adequate to cope with high-dimensional networks through its iterative improvement of the accumulated reward.

The presented hypothesis leads to the research question: Can Deep Reinforcement Learning (DRL) improve the accuracy of CD in dynamic social networks considering the state of the art of classical and ML-based solutions?

1.4 Objectives

The main objective of this research is to explore the power of GNN-based networks (Graph Attention Network (GAT) and Graph Convolution Network (GCN)) and propose a model for CD using an Actor-Critic RL-based architecture to maximize the local modularity density of a community structure in the context of dynamic online networks.

As secondary objectives of this research, we can point out the following:

- The implementation of an Actor-Critic RL-based architecture using GNN as its function approximator and its availability in a public repository.

- The validation of the implemented architecture using real-world static and dynamic social network datasets.
- The application of modularity density metric as a component of the loss function to quantify community structure.
- The use of macro-F1, micro-F1, and Normalized Mutual Information (NMI) scores to evaluate results compared to the state-of-the-art classical and ML-based CD solutions available in the literature.

1.5 Contributions

The main contribution of this research is an adaptive model with an Actor-Critic RL-based method to maximize the local modularity density of a community structure using a GNN to cope with changing aspects of large social networks represented as static or dynamic datasets. The adaptive feature of the model focuses on the continual improvement of the modularity density and its flexibility to use fixed or dynamic networks as input.

The model takes the adjacency matrix of a graph as input and translates it to a low-dimension space as an input to the RL model. This translation to a low-dimension vector space is known as graph embedding. A challenge with this approach is to find a graph embedding model that preserves the community structure of the original network. We currently use the Node2Vec algorithm to perform this task [31], though we can use any other network embedding method that preserves the community structure.

The graph embedding and the community structure are the RL's environment components. The agent component has two blocks, the actor and the critic, each implementing a GNN. In this way, we prospected the RL power of continuously improving the objective function to cope with the dynamic aspect of OSN, especially in the case of unbalanced networks, which results in the problem of the resolution limit of modularity optimization methods as described by [32].

The Actor-Critic RL-based method forms the basis for the proposed architecture called *Actor–Critic for Community Detection* (AC2CD). The AC2CD was presented in [33] and validated with real-world dynamic network datasets (Email-Eu-core, BlogCatalog3, Flickr, Youtube2). The AC2CD was compared to state-of-art solutions presenting better results: GraphGAN [34], ComE [35], SDNE [36], and CNN [37]. The results prove the hypothesis that the DRL application improves the modularity density for CD in OSN. The GAT has experimented as an adequate core component in the action space of an RL approach. The AC2CD results highlight the flexibility of RL as building blocks for solutions to constrained problems to more general scenarios. Section 5.2 presents experiments with a discussion related to this article.

- Aurélio Ribeiro Costa, Célia Ghedini Ralha, AC2CD: An actor-critic architecture for community detection in dynamic social networks, Knowledge-Based Systems, Volume 261, 2023, 110202, ISSN 0950-7051, `https://doi.org/10.1016/j.knosys.2022.110202`.

Apart from the state-of-the-art solutions presented in the previous article, we compare AC2CD implemented architecture to Community Locator And community REwriter (CLARE) [38]). The experiments using real-world OSN datasets (Email-Eu-core, BlogCatalog3, Flickr) with micro-F1 and macro-F1, and NMI scores (Email-Eu-core, BlogCatalog3) demonstrate that GNNs and DRL approaches are better suited for the CD task than others solutions based on probabilistic or shallow networks. The comparative study indicates that AC2CD presents superior accuracy than other GNN-based methods (GraphGAN, ComE, and CLARE). Section 5.3 presents experiments with a discussion of these results.

We executed the experiments in a computer named Thorin composed of a CPU Intel® Xeon Gold 5220R with 48 cores, 187GB of RAM, and two GPU NVIDIA® V100S. The operating system used was Ubuntu, with all external libraries provided by the Conda project.[1] Thorin was purchased within the Project Knedle of the University of Brasília (UnB), funded by FAP-DF and Finatec and made available to run all the experiments of this research.[2] However, a desktop without GPU but with 32 GB of RAM and a 6-core CPU can reproduce the experiments using a small dataset like Email-Eu-core (taking approximately 80 hours to execute). Thus, we could conduct additional experiments with larger datasets with the NMI score, but we faced infrastructure limitations. For example, running the Flickr dataset occurs CUDA out-of-memory error in the Thorin when running experiments for the BRACIS article.

1.6 Limitations

There are approaches to CD using dynamic attributed networks based on social concepts, but this work does not consider this kind of network. The accuracy could be improved using such networks. However, our literature research did not focus on these networks. As a result, the proposal and validation did not focus on such networks.

It would be interesting to conduct more experiments with the AC2CD implementation using dynamic datasets as available in `https://icon.colorado.edu/`. However, we were

[1]Conda Project available at `https://docs.conda.io/en/latest/`

[2]`https://unb-knedle.github.io/`

able to execute the High School dataset that is social and dynamic but the smallest one. Thus, an experimental limitation of the implemented AC2CD architecture is the computational demand of the DRL method to validate the results using large networks considering unbalanced communities. However, we believe the proposed model copes well with finding communities in large networks using available computational resources as presented in the experiments of Chapter 5.

A limitation of the proposed model is the number of communities i directly related to the output of the Actor component of AC2CD. The AC2CD experimented with datasets where $i \ll |V|$. Thus, a significant i increase the memory necessary to run the model. Given a network $G =< V, E >$, where V is a set of nodes and E a set of edges, and a community structure $C = \{c_i\}$, we might explore other relations among the community structure and Actor component output to overcome this constraint.

1.7 Document Outline

The remaining sections of this document include in Chapter 2 some foundation theory about CD and RL. Chapter 3 presents the literature review. Chapter 4 details AC2CD. Chapter 5 describes experiments to validate the proposed architecture. Moreover, Chapter 6 presents conclusions with future research directions.

Chapter 2

Theoretical Aspects

This Section presents the theoretical aspects related to the CD problem. The definition of NA is in Section 2.1. Section 2.2 includes CD classical and non-classical approaches. Finally, Section 2.3 presents the relevant aspects of RL, focusing on the Actor-Critic method.

2.1 Network Analysis

NA is a set of techniques derived from network theory, which has evolved from computer science to demonstrate the power of social network influences. Using NA in domain analysis can add another layer of methodological triangulation by providing a different way to read and interpret the same data. The use of NA in knowledge organization domain analysis is recent and evolving. The visualization technique involves mapping relationships among entities based on the symmetry or asymmetry of their relative proximity [39].

NA is conducted by collecting relational data organized in matrix form. Suppose actors are nodes, and their relations are lines between pairs of nodes. In this way, the concept of social networks changes from being a metaphor to an operative analytical tool that utilizes the mathematical language of graph theory and matrix and relational algebra. Although deterministic approaches usually emphasize that NA enables studying how the social structure of relationships around a person, group, or organization affects behaviors and attitudes, structurally bounded purposive actions may affect the social structure and vice versa. NA is a set of techniques with a shared methodological perspective rather than a new paradigm in the social sciences. NA techniques allow researchers to specify empirical indicators and to control field hypotheses through the definition and measurement of traditional catch-all concepts like social structure and cohesion [40].

2.2 Community Detection

We need to characterize graphs and networks to define CD. According to [41], a graph consists of a set of vertices or nodes and established edges that connect them. A graph can become a network when we assign numbers $c_1, ..., c_m$ to the edges, and the number c_i can be the length or weight of edge i. In [42], the author defines a *network* generically as an abstraction that allows us to encode some relationship among pair of objects. This way, we can consider networks built of any elements, such as a set of persons, web pages, neurons, or computers. We can use sets of objects to encode some relationship that depends on the set's features. For instance, considering a group of persons, we can encode the friendship relationship between two persons. So, if two persons are friends, there is a relationship of friendship between them. On the contrary, there is no such relationship. Therefore between two objects, there is or is not a considered relationship.

Formally, the components of a network are the set of nodes denoted by V and the set of edges denoted by $E = \{e = (i,j) | i,j \in V\}$. Each edge represents a relationship between i and j. We can associate a weight to an edge e defining a function $\omega(e)$. In an oriented, i.e., directed, network, $(i,j) \neq (j,i)$. A convenient way to represent the network is using its *adjacency matrix* $A_{i,j}$. The adjacency matrix of a network contains the encoding of its edges, and in case an undirected network A is a symmetrical matrix. Equation 2.1 defines the matrix A for an unweighted network. In this type of network, all edges e have the same value for the weight function $\omega(e)$.

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

Another matrix that characterizes networks is the *degree matrix*, D. The matrix D is a diagonal matrix whose elements are obtained by the Equation 2.2, where function $\mathcal{N}(\rangle)$ returns the neighbors of the node i. Using the number of nodes and edges, we can define the density of a network, ρ as $\rho = \frac{m}{n(n-1)/2}$, where n is the number of nodes and m the number of edges of the network.

$$D_{i,i} = deg(i) = \sum_{j \in \mathcal{N}(i)} \omega((i,j)) \tag{2.2}$$

According to [43], there are two configurations to represent a dynamic network. One is the Continuous-Time Dynamic Graph (CTDG), where the temporal aspect forms an edge attribute. The other is the Discrete-Time Dynamic Graph (DTDG), defined by a network $\mathcal{G} = \{\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_t\}$ as a sequence of configurations changing along the time.

In the second method, the CD is processed at each snapshot $\mathcal{G}_i$ considering the network discretization at DTDG.

Typically, one can define a *community* as a bunch of densely connected vertices that connect sparsely with the other vertices, a static or dynamic graph. Finding communities in a graph helps unveil its internal organization. Also, we can use it to characterize the entities that compose it (e.g., groups of people with shared interests, products with similar properties) [44]. Figure 2.1 shows a network highlighting three disjoint communities (i.e., non-overlapping communities).

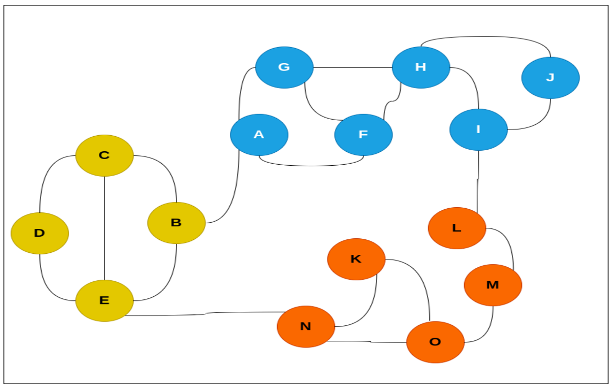


Figure 2.1: A network with three disjoint communities (yellow, blue, and red).

The problem of finding community in a network is not novel. However, even the precise definition of community in a network has yet to be discovered. The authors in [21] argue that we should not consider CD a well-defined problem but an umbrella term with many facets. These facets emerge from different goals and motivations of the network that we want to understand. They can also lead to different perspectives on formulating the CD problem. When selecting and comparing CD methods, it is crucial to consider these underlying motivations. The classes of CD methods may vary from classical approaches like statistical methods [10, 45] and optimization [46], to ML-based methods like [47–50].

One can confuse the CD problem with clustering at the first sign. On the one hand, considering the clustering problem, we have instances represented in a vector space as nodes and usually a distance function computed based on the node attributes. The distance function tells us how far the nodes are so we can group the nearest nodes in the same group. The CD is a learning task, similar classes of vertices from the network's topology using attributes of nodes and edges. In this way, CD is sometimes referenced as spectral clustering or a kind of CD solution.

2.2.1 Classical Approaches

Classical approaches represent a category of CD solutions that employ some analytical method. Figure 2.2 presents a taxonomy of CD approaches and algorithms inspired by the terminology used by the survey of [51]. We can group the main approaches to solving the CD problem by modularity optimization, random walk, propagation, and other approaches. In the sequence, we describe the approaches with respective algorithms for CD.

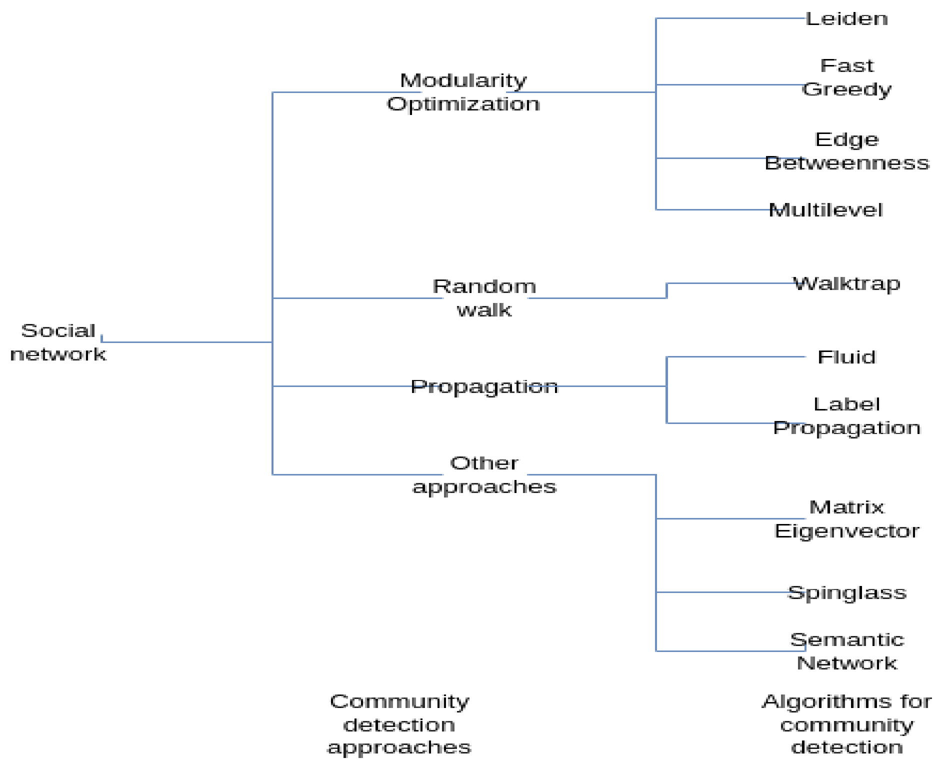


Figure 2.2: Taxonomy of CD approaches and algorithms. Source: [51].

Modularity Optimization

According to [52], the most popular method to detect communities in graphs consists of quality function optimization, the modularity introduced by [1] and [53]. Modularity quantifies the deviation of the internal link density of the clusters from the density one expects to find within the same groups of vertices in random graphs and the same predicted degree sequence of the network at study. The idea is that vertices linked randomly to each other should not form communities since no high values of link density exist. Consequently, modularity high values are supposed to indicate “suspiciously” high values of internal link densities for the subgraphs. Those are distinct from groups of randomly linked vertices regarded as natural communities.

The *Newman-Girvan modularity* is arguably one of the most common clustering measures used in the literature. This method was initially proposed from the clustering perspective [1, 53]. It is a global quality function and aims to find the network community structure as a whole. Given a partition $C = \{V_1, ..., V_k\}$ of a network into k groups, the modularity of C can be written as:

$$Q = \frac{1}{2m} \sum_{uv} \left[A_{uv} - \frac{d_u d_v}{2m} \right] \delta(c_u, c_v), \tag{2.3}$$

where $d_u = \sum_v A_{uv}$ is the degree of node u, $2m = \sum_u d_u$ is the total weight of all edges in the network, and $\delta(c_u, c_v)$ is the Kronecker delta whose value is one whether $c_u = c_v$ (i.e., both nodes are in the same community) and zero otherwise. By optimizing the modularity measure over the space of all partitions, one aims to identify groups of nodes that are more densely connected than one would expect from a statistical null model of the network. This statistical null model is commonly chosen to be the configuration model with preserved degree sequence [21].

The modularity optimization approach is a broader class of CD proposals. This approach tries to identify a community structure that maximizes internal and minimizes external modularities of a community. An example of a modularity optimization algorithm is the *Fast Greedy*. It merges individual nodes into communities in a way that greedily maximizes the modularity score of the network. If no merge can increase the current modularity score, the algorithm can stop since there is no further increase. This algorithm is said to run almost in linear time on sparse graphs.

Another example of a modularity optimization algorithm is the *Edge Betweenness*. The main idea is that the betweenness score of the edges connecting two communities is typically high, as many of the shortest paths between nodes in separate communities go through them. So, it gradually removes the edges with the highest betweenness score and recalculates the score after every removal. This way, later or soon, the network falls into separate components.

The *Multilevel* algorithm is bottom-up. In its operation, every vertex initially belongs to a separate community, and vertices are moved among communities iteratively. This movement tries to maximize the vertices' local contribution to the overall modularity score. When the algorithm reaches a consensus (i.e., no single move would increase the modularity score), every community in the original graph is shrunk to a single vertex (while keeping the total weight of the adjacent edges). The process restarts on the next level. The algorithm stops when it is impossible to increase the modularity anymore after shrinking the communities to vertices.

The *Leiden* algorithm is considered an improvement to the Louvain algorithm [54], consisting of three phases: local moving of nodes, refinement of the partition, and aggregation of the network based on the refined partition by using the non-refined one to create an initial partition for the aggregate network.

The modularity optimization approach is a simple and direct method to compute the quality of a community structure in a network. However, it is not unanimous in the literature. One negative issue related to modularity is its resolution limit when dealing with large networks, as presented by [32]. In general, the resolution limit problem says that we must make a definite statement about modules found through modularity optimization with a method that verifies whether the modules are indeed single communities or a combination of communities. It is then necessary to inspect the structure of each of the modules found.

However, the resolution parameter inclusion can overcome this limitation [55]. Moreover, another approach to overcome the resolution limit problem is to use a new metric derived from modularity called modularity density Q_{ds}, defined by [56]. This metric uses the number of nodes of each community to normalize the objective value instead of the total edge number. The function maximizes the difference between the number of internal and external edges of each community, according to the definition:

$$Q_{ds} = \sum_{c \in C} \left[\frac{m_c}{m} p_c - \left(\frac{2m_c + c_e}{2m} p_c \right)^2 - \sum_{c' \neq c} \frac{m_{cc'}}{2m} p_{cc'} \right], \tag{2.4}$$

where $m_{cc'}$ is the number of edges between communities c and c', $p_c = \frac{2m_c}{n_c(n_c-1)]}$ is the density of links inside c, $p_{cc'} = \frac{m_{cc'}}{n_c n_{c'}}$ is the density of edges between c and c', n_c is the number of nodes in c, and the other quantities are the same as in Equation 2.3. Again, the partition that maximizes Q_{ds} corresponds to the community structure.

Random Walk

The random walk approach implements algorithm *Walktrap* [26]. The basic idea of the Walktrap is that short random walks tend to stay in the same community.

Let us consider a discrete random walk process (or diffusion process) on graph G. At each step, a walker is on a vertex i and moves to a vertex j chosen randomly and uniformly among neighbors. The sequence of visited vertices represents a Markov chain, the states of which are the graph vertices. At each step, the transition probability from vertex i to vertex j is $P_{ij} = A_{ij}/d(i)$. The transition probability defines the transition matrix P of random walk processes. One can also write $P = D^{-1}A$ where D is the diagonal matrix of the degrees $\forall i, D_{ii} = d(i)$ and $D_{ij} = 0$ for $i \neq j$. The process is driven by the powers of the matrix P. The probability of going from i to j through a random walk of length t - $(P^t)_{ij}$.

Propagation

The propagation approach deploys on the spread of something over the network, represented by the *Label Propagation* algorithm, where each vertex is assigned a different label. In each iteration, a vertex chooses the dominant label in its neighborhood. The vertices are updated in a randomized order before every iteration. The algorithm ends when vertices reach a consensus. Since ties are broken randomly, there is no guarantee that the algorithm returns the same community structure after each run, and they frequently differ. Another approach based on propagation is *Fluid* [44], based on the idea of fluids interacting in an environment, expanding and contracting as an interaction result.

Other approaches

There are other approaches based on many different ways to model the network, as *Spin glass* [57], the *Semantic network* [58] that creates a semantic network of internet content and tries to reveal relationships between users. The *Matrix Eigenvector* approach introduced by [1] implements the *Leading Eigenvector* algorithm. This algorithm splits the network into two components according to the leading eigenvector of the modularity matrix derived from the adjacency matrix. Then, recursively takes the given number of steps by splitting the communities as individual networks.

2.2.2 Non-classical Approaches

Non-classical approaches refer to methods that mainly rely on probabilistic or ML techniques to tackle the CD problem or that compose some more elaborated approach. The job is to learn an adequate representation of the network structure of a given network. A common strategy in these methods splits the work into two stages: representation and refinement. The approach finds candidate subgraphs to communities in the representation stage. In the refinement stage, the approach decides when the previously identified subgraphs are natural communities.

In the following sections, we present techniques that describe the two stages of learning-based approaches to CD, such as Graph Embedding, Node2Vec, Stochastic Block Model (SBM), and GNN. In the following sections, we present CD metrics and RL.

Graph Embedding

In a general form, Graph Embedding, or network embedding, represents an approach to translate an entire network into a low-dimensional space to be handled by ML algorithms. Mathematically, the graph embedding problem is a function that maps a graph (or an adjacency matrix) into a set of dense continuous vectors, as presented in the schema of Figure 2.3.

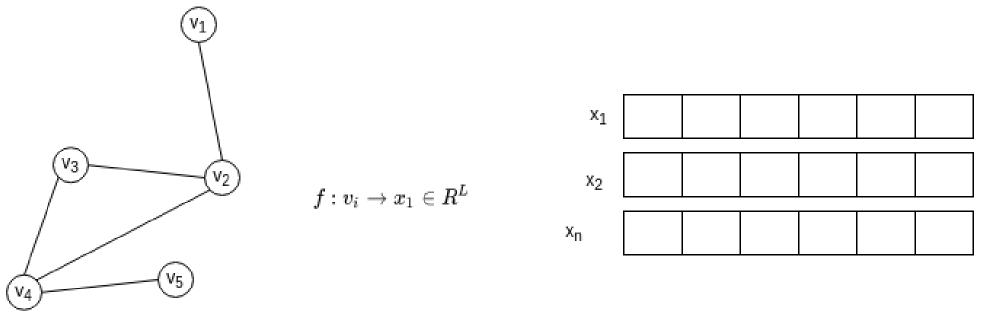


Figure 2.3: A graph embedding schema.

Structural information extraction from graphs using traditional machine approaches often relies on summary graph statistics (e.g., degrees or clustering coefficients), kernel functions, or carefully hand-engineered features to measure local neighborhood structures. However, these approaches are limited because these hand-engineered features are inflexible — i.e., they cannot adapt during the learning process — and designing these features can be time-consuming and expensive.

According to [59], a taxonomy of graph embedding methods presented in the literature points to three major method categories: matrix factorization-based, random walk-based, and Neural Network (NN)-based. Matrix factorization-based methods construct a high-order proximity matrix based on transition probabilities and factorize it to obtain the node embeddings [47–50]. In addition to the static graph embedding methods, in this work, we also discuss the emerging deep learning-based dynamic graph embedding methods [60, 61].

More recently, there has been a surge of approaches seeking to learn representations that encode structural information about the graph. These approaches learn a mapping that embeds nodes, or entire (sub)graphs, as points in a low-dimensional vector space $\mathbb{R}^d$. The goal is to optimize this mapping so that geometric relationships in the embedding space reflect the graph's original structure. After optimizing the embedding space, the learned embeddings are feature inputs for downstream ML tasks. The main distinction between representation learning approaches and previous work is how they treat the problem of representing a graph structure. Previous work treated this problem as a pre-processing step, using hand-engineered statistics to extract structural information. In contrast, representation learning approaches treat this problem as an ML task, using a data-driven approach to learn embeddings that encode graph structure [62].

Node2Vec

Node2Vec is a class of network embedding solutions introduced by [31] that defines a flexible notion of a node's network neighborhood. By choosing an appropriate neighborhood notion, Node2Vec can learn representations that organize nodes based on their network roles and the communities they belong. It achieves this by developing a family of biased random walks, which efficiently explore diverse neighborhoods of a given node. The resulting algorithm is flexible, giving us control over the search space through tunable parameters, in contrast to rigid search procedures in prior works.

Consequently, Node2Vec generalizes modeling the full spectrum of equivalences observed in networks. The parameters governing the search strategy have an intuitive interpretation and bias the walk towards different network exploration strategies. A semi-supervised approach learns these parameters directly using a tiny fraction of labeled data. Algorithm 1 describes Node2Vec according to [31]. The algorithm's input is the graph G, the dimension of the output d, i.e., the length of the vector representing each node, and some other hyperparameters. Node2Vec operates in three phases, preprocessing to compute transition probabilities for each pair of nodes (Line 1) and generate the probability matrix π, random walk simulations of length l (Line 6), and optimization using Stochastic Gradient Descent (SGD) (Line 9). The method *node2vecWalk*, defined at Line 13 and called at Line 6, creates a sequence of nodes, called a walk, to visit the neighborhood of a node u (Line 5) and contribute to the embedding of u (Lines 5-8).

Stochastic Block Model

SBM is an effective generative model of network block structures adopting statistical modeling for CD for the first time [45]. The method probabilistically assigns nodes in a network to different communities (block structures) using a node membership likelihood function. Then, progressively infers the probabilities of node memberships by inferencing on the likelihood function to derive hidden communities in the network. Note that several SBM variants exist for CD, but their core generation process is the same. There are two steps in the generation process: (1) iteratively assign a community to each node in the network, and (2) compute or update the probability of two nodes connected by an edge [18].

Taking a social network as an example, SBM can be used to capture a probabilistic generation process with community distribution as a hidden variable. The communities

Algorithm 1: The Node2Vec algorithm.

```
Input: Graph G=(V,E,W), Dimensions $d$, Walks per node $r$, Walk length $l$,
       Context size $k$, Return $p$, In-out $q$
1  $\pi$ = PreprocessModifiedWeights(G,p,q);
2  G' = (V,E,$\pi$) ;
3  Initialize walks to Empty;
4  for iter = 1 to $r$ do
5      forall node $u \in V$ do
6          walk = node2vecWalk(G', $u$, $l$);
7          Append walk to walks;
8      end
9      $f$ = StochacticGradientDescent($k$, $d$, walks);
10 end
11 return $f$
12 ____________________________________________
13 node2vecWalk(Graph G' = (V,E,$\pi$), Start node u, Length l)
14 Initialize walk to [$u$];
15 for walk_iter = 1 to $l$ do
16     curr = walk[$-1$];
17     $V_{curr}$ = GetNeighbors(curr, Ǵ);
18     s = AliasSample($V_{curr}$, $\pi$);
19     Append $s$ to walk;
20 end
21 return walk ;
```

are reconstructed by maximizing the likelihood function of the node community membership. In the social network, the nodes are partitioned into k disjoint communities with probability $\omega = \{\omega_1, ..., \omega_k\}$. Assuming there are two nodes v_i and v_j belonging to two communities C_r and C_s, represented by c_{ir} and c_{js}. The probability that nodes v_i and v_j connected by an edge, i.e., a_{ij} (0 or 1), obeys a Bernoulli distribution with parameter π_{rs} [63].

Convolutional Neural Networks

CNN constitutes a class of NN commonly applied to image processing using the convolution operation instead of matrix multiplication. The author in [64] states that the primary design goal of CNNs was to create a network where the neurons in the early layer of the network would extract local visual features, and neurons in later layers would combine these features to form high-order features. A local visual feature is a feature whose extent is limited to a small patch, a set of neighboring pixels in an image. Another important concept of CNNs is pooling, a form of non-linear down-sampling. There are several non-linear functions to implement pooling, where max pooling is the most common. It partitions the input image into rectangles and outputs the maximum for each sub-region.

In the field of CD, the use of CNNs follows the use of matrix factorization techniques as user interactions within different OSNs represented as sparse, high dimensional adjacency matrices. In this way, [37] present a semi-supervised community detection approach, combining deep learning techniques (i.e., CNN layers) with topological properties of a social network.

Graph Neural Networks

GNN is a neural network architecture based on an information diffusion mechanism defined by [65]. Figure 2.4 depicts the branches of AI approaches and locates GNN as a specialization of the supervised learning method. GNN can directly process most graph types, e.g., acyclic, cyclic, directed, and undirected. GNN implements a function $\tau(G, n) \in \mathbb{R}^m$ that maps a graph G and one of its n nodes into an m-dimensional Euclidean space. A set of units process a graph in GNN, each corresponding to a graph node linked according to their connectivity. The units update their states and exchange information until they reach a stable equilibrium. The output of a GNN is then computed locally at each node on the united state base. The diffusion mechanism is constrained to ensure that a unique stable equilibrium always exists. Some specializations of GNNs are GCN and GAT.

GCN is a GNN architecture inspired by the Convolutional Neural Network (CNN), presented by [66]. GCN was initially used in the nodes classification problem (e.g., documents) in a graph (e.g., citation network) using semi-supervised learning, where labels are only available for a small subset of nodes. Additionally, this approach joining GCN and Markov Random Fields (MRF) is applied successfully to the CD as described by [67].

GCN defines a spectral graph convolution by multiplying a signal $x \in \mathbb{R}^N$ with a spectral filter in the Fourier domain $g_\theta = diag(\theta)$, i.e.

$$g_\theta \star x = U g_\theta U^\top x \tag{2.5}$$

where U is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-1/2} A D^{-1/2} = U \Lambda U^\top$, with a diagonal matrix of its eigenvalues Λ and $U^\top x$ being the graph Fourier transform of x. It uses two graph convolution layers to derive a network embedding and then applies the *softmax* function to classify nodes into different categories. In training, the prior information on community memberships of a few nodes, network topology, and node attributes are input to learn the neural network weight parameters. Similar to CNN, GCN has an excellent global search capability, i.e., it can extract complex features or patterns from a myriad of local features by a stack of convolution operations.

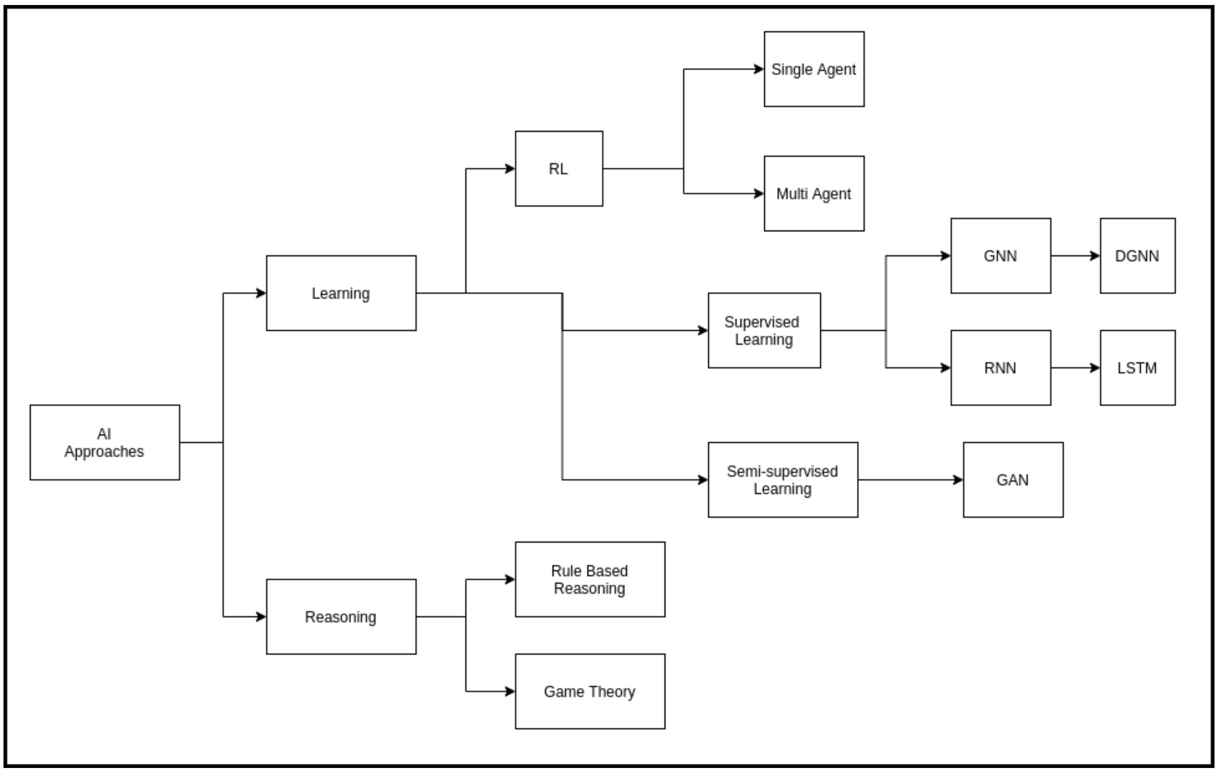


Figure 2.4: Taxonomy of AI approaches applied to the CD problem.

According to [67], GCN has at least two drawbacks. First, GCN aims primarily at deriving a network embedding of the input data in the hidden layers of CNN. However, such an embedding is not community oriented and does not consider community properties. Second, GCN can only obtain a relatively coarse community result since it lacks smoothness constraints to reinforce similar or nearby nodes to have compatible community labels.

GAT is another GNN architecture, proposed by [68], that represents an attention-based neural network architecture designed to perform node classification of graph-structured data. This architecture aims to compute the hidden representations of each node in the graph by attending to its neighbors, a self-attention strategy. The attention architecture has several interesting properties: (1) the operation is efficient since it is parallelizable across node neighbor pairs; (2) it applies to graph nodes having different degrees by specifying arbitrary weights to the neighbors; and (3) the model is directly applicable to inductive learning problems, including tasks where the model has to generalize to completely unseen graphs. Figure 2.5 presents the node representation update implemented by the GAT architecture, where the representation of Node 1 ($\overrightarrow{h_1'}$) is the result of an aggregation (concat/avg) representing its neighborhood with prior representation ($\overrightarrow{h_1}$). Mathematically, we can describe the update strategy by Equation 2.6, where W is the weight matrix, K is the number of head attentions, and α is the normalized attention coefficient.

$$\overrightarrow{h_1'} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\overrightarrow{h_j}\right) \tag{2.6}$$

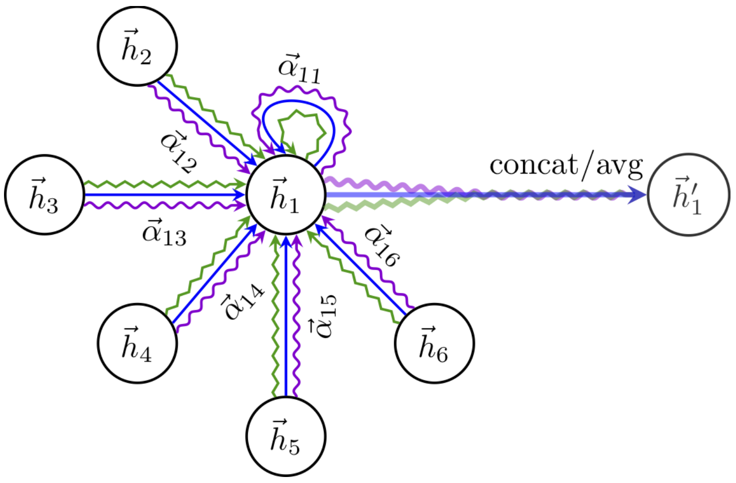


Figure 2.5: GAT node representation update. Source: [68].

The attention mechanism has become a fundamental standard in many sequence-based tasks. One of the benefits of attention mechanisms is that they allow for dealing with variable-sized inputs, focusing on the most relevant input parts to make decisions. A self-attention or intra-attention is commonly referred to when an attention mechanism computes a representation of a single sequence. Together with Recurrent Neural Network (RNN) or convolutions, self-attention has proven to be useful for tasks such as machine reading [69] and learning sentence representations [70]. Additionally, [71] showed that self-attention can improve a method based on RNNs or convolutions and is sufficient for constructing a powerful model obtaining state-of-the-art performance on the machine translation task.

2.2.3 CD Metrics

CD metrics are scores used to evaluate the outcome of a solution method, as modularity presented in Section 2.2.1. The information retrieval field inherits many CD metrics. We can classify these metrics as internal when considering only the internal nodes of each community and external when comparing two communities.

F1-score is the harmonic measure of precision P and recall R, according to [72]. Micro-averaged F1-score (Micro-F1) and macro-averaged F1-score (Macro-F1) are ways to aggregate the F1-score measuring the performance of a classifier in a multi-label categorization. The authors in [73] defined these two measures as:

- Micro-F1: the harmonic mean of the micro-precision and micro-recall computed based on the sum of true positives, false positives, and false negatives values.

$$\mathbb{F}_1 = H(\bar{P}, \bar{R}) = \frac{2\bar{P}\bar{R}}{\bar{P} + \bar{R}} = 2\frac{(\frac{1}{n}\sum_x P_x)(\frac{1}{n}\sum_x R_x)}{\frac{1}{n}\sum_x P_x + \frac{1}{n}\sum_x R_x}$$

- Macro-F1: the arithmetic means of F1-scores of all categories.

$$\mathcal{F}_1 = \frac{1}{n}\sum_x F1_x = \frac{1}{n}\sum_x \frac{2P_x R_x}{P_x + R_x}$$

The Normalized Mutual Information (NMI) is one external metric [74, 75]. Given a reference community structure A and a detected community structure B, NMI computes the overlapping nodes in A and B. To define NMI, we need to approximate the marginal probability of a randomly selected node being in the community a, with $a \in A$, and b, with $b \in B$, by $P_A(a) = \frac{n_a}{n}$ and $P_B(b) = \frac{n_b}{n}$, where n_a and n_b denote community size of a and b. Moreover, $P_{AB}(a, b) = \frac{n_{ab}}{n}$, where n_{ab} is the number of nodes that are both in the community of partition A and group b of partition B. Equation 2.9 mathematically presents the NMI, where I represents the mutual information (Equation 2.7), and H is the entropy (Equation 2.8). Since $H(P_{AB}) \leq H(P_A) + H(P_B)$, $NMI(P_A, P_B)$ is bounded below by 0. Also note that $H(P_{AB}) = H(P_A) = H(P_B)$ when A and B are identical, which means, in this case, $NMI(P_A, P_B) = 1$ [75].

$$I(P_A, P_B) = \sum_k \sum_j P_{AB}(a, b) log \frac{P_{AB}(a, b)}{P_A(a)P_B(b)} \tag{2.7}$$

$$H(P_A) = -\sum_a P_A(a) log P_A(a) \tag{2.8}$$

$$\text{NMI}(P_A, P_B) = \frac{I(P_A, P_B)}{[H(P_A) + H(P_B)]/2} \tag{2.9}$$

2.3 Reinforcement Learning

RL is a subfield of Artificial Intelligence (AI) that explicitly considers the whole problem of a goal-directed agent interacting with an uncertain environment [76]. Figure 2.6 shows the general architecture of an RL model with the two main elements: agent and environment.

The environment is the *locus* where the agent operates. The environment represents the problem an agent tries to solve, commonly implemented as a sequence of states. At each received action A_t, the environment issues a new State S_{t+1} and the corresponding Reward R_{t+1}. Thus, a common approach to solving the problem is through an MDP.

The agent is the RL element responsible for sensing the environment and taking actions that change the environment. The set of all possible actions in an environment is the *action space*. For each state S_t, an action A_t is issued with its associated probability the way an agent distinguishes the different types of RL.

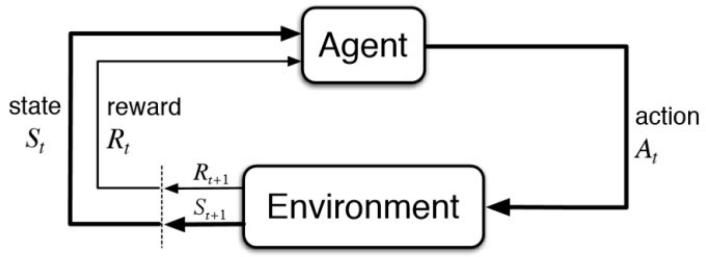


Figure 2.6: RL general architecture. Source: [76].

In [77], the author correlates some knowledge areas to RL as depicted in Figure 2.7, where six domains heavily overlap each other on the methods and specific topics related to decision-making. At the intersection of all those related scientific areas is RL, which can take the available information from these domains as follows:

- Computer Science: More specifically, ML, where RL's goal is to learn how an agent should behave when it faces imprecise observational data.

- Engineering: This helps to take a sequence of optimal actions to get the best result (Optimal Control).

- Neuroscience: The human brain acts similarly to the RL model, with its dopamine system acting in the Reward System.

- Psychology: Behavior studies in various conditions, such as how people react and adapt, are close to the RL model (Classical/Operant Conditioning).

- Economics: One important topic is to maximize reward in terms of imprecise knowledge and the changing conditions of the real world (Bounded Rationality).

- Mathematics: This works with idealized systems and devotes significant attention to finding and reaching the optimal conditions in the Operations Research field.

The literature classifies RL algorithms, at a high level, as MDP or Bandits-based. The Bandits problem can be formalized as a one-state MDP because it does not depend on the previous state (i.e., arm pulled). Following the taxonomy of RL methods developed by [78]. Figure 2.8 shows this classification. This research concentrates on the branch MDP

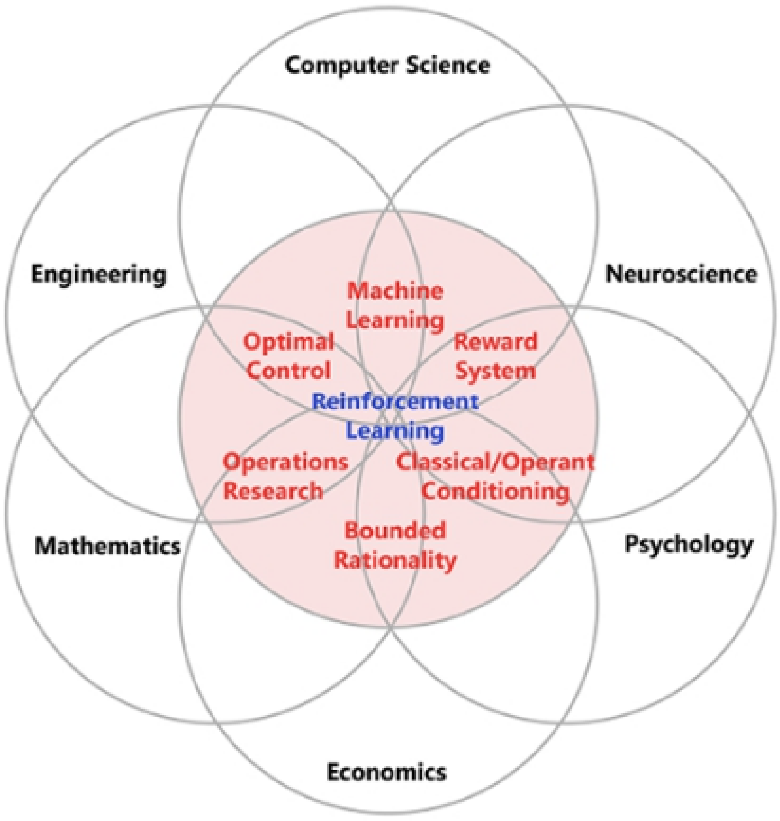


Figure 2.7: RL related areas. Source: [77].

→ Model-Free → Policy-Based → Gradient-Based → Trust Region Policy Optimization (TRPO)/Proximal Policy Optimization (PPO) (the blue path).

Although the taxonomy presented by [78] in Figure 2.8 classifies MDP as Model-based and Model-free, the authors in [76] differentiate these two methods as Model-based methods rely on planning as their primary component. In contrast, Model-free methods primarily rely on learning. Moreover, all these methods look ahead to future events, computing a backed-up value and then using it as an update target for an approximate value function, such as Monte Carlo (MC) and Temporal Difference (TD). In both methods, the training update is implemented either after the control episode finishes (MC) or after one or more steps execute in an episode (TD).

In Figure 2.8, following the Value-Based branch, we find the Q-Learning Off-Policy method and its specialization DQN using a deep neural network. The authors in [76] consider Q-learning one of the most important breakthroughs in RL. Q-learning was presented by [79] as an off-policy TD control algorithm. The simplest form, one-step Q-learning, is defined by Equation 2.10.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1}\gamma \max_a Q(S_{t+1}, a - Q(S_t, A_t)] \tag{2.10}$$

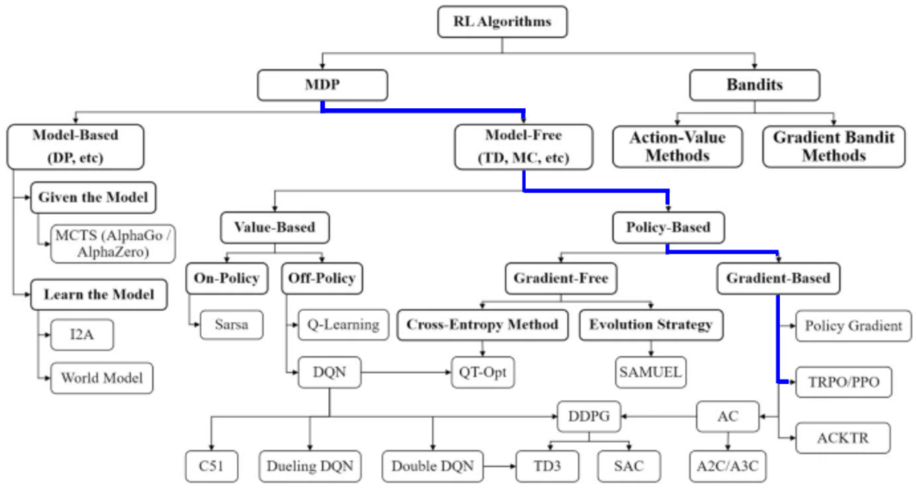


Figure 2.8: RL taxonomy. Source: [78].

In this case, the learned action-value function, Q, directly approximates to $q*$, i.e., the optimal action-value function, independent of the policy being followed. This strategy dramatically simplifies the algorithm analysis enabling early convergence proofs. The policy still has an effect as it determines which state-action pairs are visited and updated. However, it is required for correct convergence as all pairs continue to be updated.

In the Policy-based branch, the Policy Gradient is a class of gradient-based methods employing an estimator to maximize the long-term reward. Policy Gradient methods present some advantages to other methods as the guaranteed convergency to local optimum and its fitness to discrete and continuous action and state scenarios. However, there are some negative aspects to using the Policy Gradient methods, as they are not guaranteed to converge to a global maximum, and they are sample inefficient because they need to discard some episode data to avoid bias to the gradient estimator. Policy Gradient methods implemented by Equation 2.11, where we initialize the parameter ω_0 with random values and iteratively improve by the loss function $\mathcal{L}$ gradient. The hyperparameter η is the learning rate and controls the improvement achievement.

$$\omega_{n+1} = \omega_n - \eta \nabla \mathcal{L}_{|\omega}(\omega_n) \tag{2.11}$$

The PPO is a kind of Policy Gradient method defined in [80]. The PPO algorithm has some of the benefits of TRPO, is much simpler to implement, more general, and has better sample complexity (empirically), according to the authors of both algorithms. In this research, we concentrate on the clipped version of PPO with the objective function as presented in Equation 2.12.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)], \tag{2.12}$$

In Equation 2.12, ϵ is a hyperparameter. The expectation $\hat{\mathbb{E}}_t[...]$ indicates the empirical average over a finite batch of samples. The first term inside the min function is $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, which is the probability of taking action a_t at state s_t in the current policy divided by the previous one. The second term, $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$, modifies the surrogate objective by clipping the probability ratio, which removes the incentive for moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$. In this equation, $\hat{A}_t$ represents the estimated advantage at time t. Thus, the minimum of the clipped and unclipped objectives are taken, and the final objective is lower (i.e., pessimistic bound) than the unclipped objective. With this scheme, it is possible to ignore only the change in probability ratio when it would improve the objective and then include when it turns worse the objective. The action selection is a core component of any RL system. One approach for this task is to use a neural network to choose the best action given a state.

2.3.1 Actor-Critic Method

The actor-critic follows the RL approach, where two components compound the learning agent. According to [76], if we learn an approximation for a value function in addition to the policy approximation, we have an actor-critic method, where the actor refers to the learned policy π and the critic to the learned value-function V. Figure 2.9 presents the RL loop in an actor-critic architecture highlighting the two main components and the messages exchanged in the form of the action and the TD error.

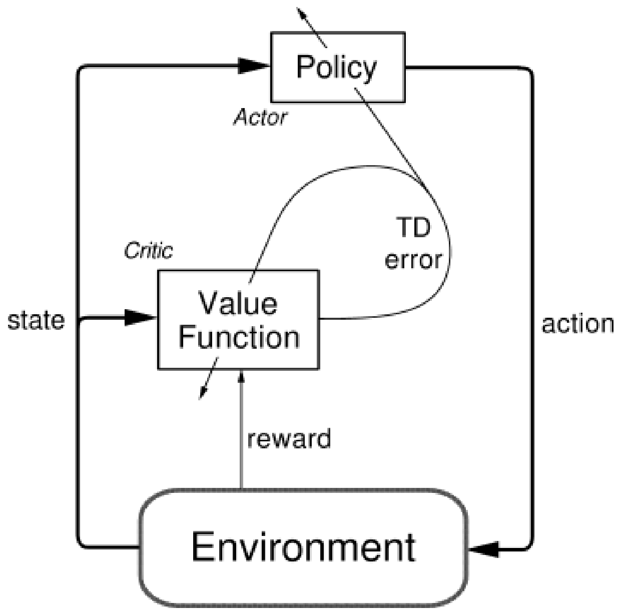


Figure 2.9: The actor-critic architecture. Source: [76].

TD error is the difference between the estimated value and the current value of a state s_t, given by the Equation 2.13, where V is the current value function implemented by the critic, r is the current reward and γ is the discount factor, which quantifies the importance of future rewards. If the TD error is positive, it suggests that the tendency to select an action a_t should be strengthened for the future. Whereas the TD error is negative, the tendency should be weakened.

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{2.13}$$

In DRL, the policy implemented by the actor and the value function implemented by the critic can use deep neural networks. The most common situation em DRL is the combination of the Actor-Critic method with some implementation of policy gradient, such as PPO.

Chapter 3 presents the literature review protocol and its execution, including the studies that inspired this research and the contributions compared to the related work.

Chapter 3

Literature Review

This Section presents the literature review method used in this research. The review protocol follows the definition of [81], which includes three main phases: planning, conducting, and reporting the review. Section 3.1 describes the planning phase of the systematic review protocol with the main characteristics of the mnemonic PICO (Population, Intervention, Comparison, and Outcome) used as a literature search strategy to ensure comprehensive and bias-free searches. Section 3.2 describes the review conduction highlighting the used search bases and the selected studies. It is worth noting that we made some manual interventions to add relevant studies with a refinement review process. Section 3.3 presents related work indicating contributions compared to this work.

3.1 Planning

According to the protocol proposed in [81], the planning stage includes identifying the need for a review and developing a protocol. The need for a literature review arose to identify the featured research in a vibrating area such as CD with ML techniques, where some research paths become obsolete too fast. More specifically, we would like to investigate the state-of-the-art approaches to CD in complex dynamic networks that apply some ML approach. We undertook the literature review with works published from 2015 to 2020. However, we continued to evaluate new publications until 2023.

The mnemonic PICO (Population, Intervention, Comparison, and Outcome) is instantiated during the protocol planning phase as follows:

- Population: works on CD of social networks.
- Intervention: works using RL.
- Comparison: we intend to compare our work with classical and ML approaches to CD.

- Outcome: create a model to detect communities in OSNs.

The research questions that guided the literature review were:

1. What are the state-of-the-art approaches to CD in complex dynamic networks?
2. What is the performance of the approaches to dynamic CD with RL?

Table 3.1 summarizes the main features of our literature review protocol.

Table 3.1: Literature review protocol features.

Feature	**Description**
Period	from 2015 to 2020 (refined to 2023)
Tool	Parsifal
Publication type	conference and journal articles
Repository	ACM Digital Library, IEEE Xplore, and Springer Link

3.2 Conducting

The literature review uses the Parsifal tool [82][1] to search the CD state-of-the-art approaches to social networks, employing mainly RL. We search publications on the Springer Link,[2] IEEE Xplore,[3] and ACM Digital Library.[4] The search string used in all search engines was *"reinforcement learning" AND "community detection" AND "social network"*. Table 3.2 presents the inclusion and exclusion criteria for filtering the studies. Figure 3.1 presents the percentage referring to the number of publications imported per source. Note that 81.6% of publications were from the Springer Link, 10.2% from the IEEE Xplore, and 8.2% from the ACM Digital Library.

Table 3.2: Inclusion and exclusion criteria.

Inclusion criteria	**Exclusion criteria**
big data context	not applied on social network
other ML approach	text not available in English
statistical approach	publication year minor than 2015
survey or literature review	

We selected publications with full text available in English covering at least one topic of interest. One relevant issue detected was that even those papers that use RL for CD have

[1]Parsifal is an online tool designed to support researchers performing a systematic literature review. Available at `https://parsif.al/`. Accessed on 17 April 2020.

[2]`https://link.springer.com/`

[3]`https://ieeexplore.ieee.org/Xplore/home.jsp`

[4]`https://dl.acm.org/`

some restrictions. For example, the work of [83] uses RL for CD considering temporal aspects. However, the author focuses on undirected edges and disjoint communities, making the study so restrictive for our purpose. Another issue is that some studies were not in the repositories' search engines like [84] and [85].

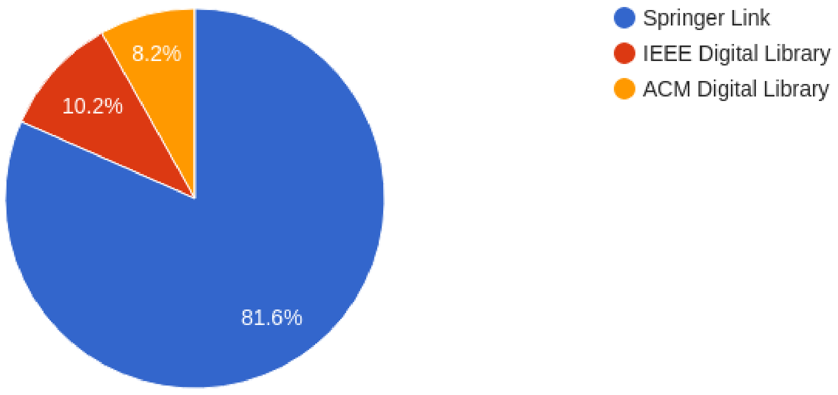


Figure 3.1: Percentage of publications imported per source.

A Quality Assessment Checklist (QAC) was defined to rank the select studies with a cutoff score to filter the studies considering this research scope. The QAC was composed of the following questions:

1. Does the study exploit more than one approach?

2. Does the study treat large-scale networks?

3. Does the study use a comparative approach?

4. Is it possible to replicate the study (data source available)?

The conduction of the literature review started with importing the publications, 40 from Springer Link, five from IEEE Xplore, and four from ACM Digital Library. From 49 imported studies, we found three duplicates, 29 were rejected for not being compliant with the inclusion criteria or having matched some exclusion criteria, and 17 were accepted. However, six studies did not reach the minimum score in QAC. Figure 3.2 presents the histogram of selected and accepted papers by source. It is possible to see that the number of selected studies from Springer Link is the biggest one. However, ACM Digital Library presents the highest acceptance rate with 3 of 4 accepted studies (75%), against 0 of 5 from IEEE Xplore and 14 out of 40 from Springer Link. Figure 3.3 presents the distribution of selected publications by year and highlights the growth of ascending number of publications matched by the selection criteria in 2019 and 2020.

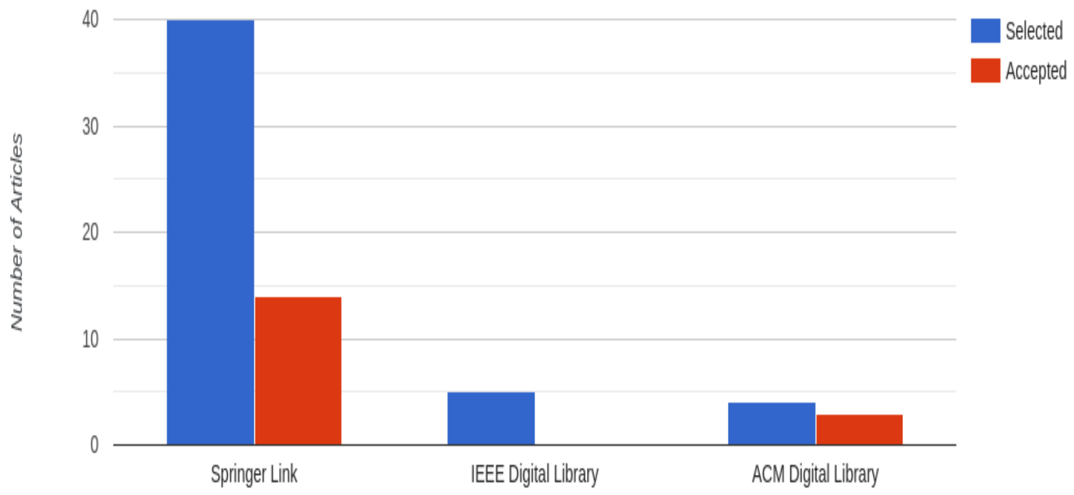


Figure 3.2: Accepted publications per source.

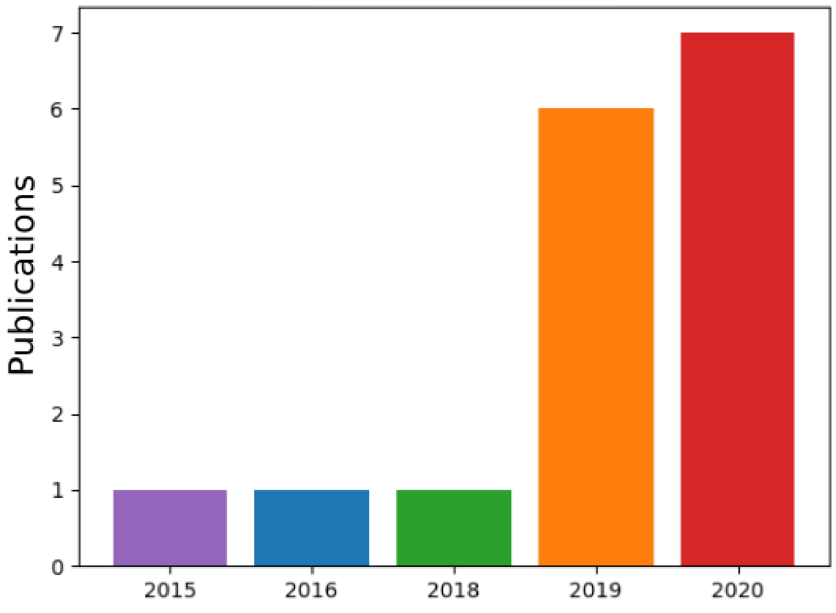


Figure 3.3: Selected publications by year.

3.3 Reporting

The literature review results include 20 publications in Table 3.3. There are two publications added after the refining process of the review: [86], and [87]. We present the studies in reverse-chronological order. It is worth noting that some studies do not apply AI approaches. We may explain this aspect due to the early application of ML to NA problems. The *N.A.* in the AI approach column presents the works without AI approaches. We consider these studies relevant in the NA field as they apply methods usable with little adaptation.

The authors in [86] present the Dual Structural Consistency Preserving CD (DSCPCD) method to uncover the hidden overlapping community structure on social networks. The authors claim that DSCPCD investigates the implicit friendship relationship from a global perspective. Specifically, to unleash the power of the game theory of complex relationship characterization. The DSCPCD introduced an evolutionary game to remedy the "complete rationality" defect in past game theoretic methods, making DSCPCD closer to real scenarios. The authors also defined a dynamic payoff matrix to characterize the community-aware interaction state. Besides the good results achieved in the experiments and the lightweight architecture that enables DSCPCD to cope with large networks, the authors tested only on static networks limiting the scope of use.

In [87], the authors propose a novel overlapping community-detection algorithm based on adaptive Density Peak Clustering (DPC) using an iterative partition strategy (ODPI). In particular, the authors emphasized that different cut-off distances d_c may lead to highly varying algorithm performance for diverse social networks. To this end, the proposed ODPI can calculate d_C adaptively based on different network scales and features, which indicates that d_c can adapt to diverse network topologies. Besides, there were tests of the performance of the ODPI algorithm on complex networks by setting different values of d_c, and the final results demonstrate the effectiveness in adaptive d_c. The tests did not include a dynamic network or a larger one. ODPI lies on a distance matrix, which limits the networks to be processed.

In [84], the NA focuses on a relationship network of citations and keywords using this analysis to detect indirect links between keywords with higher semantic relationships. Authors identify vital knowledge units and discover the topics with greater significance. This study can give us insight into using other ways to model a network from a set of documents. Other studies that do not employ AI approaches are [88] and [85], which make a systematic literature review on network representation learning. [88] focus their study on generating an efficient network representation to deal with complex high dimensional data, and [89] focus on applying node attributes to improve CD.

The work of [90] proposes a unique social graph hybrid model named Representation Learning Via knowledge-graph Embeddings and ConvNet (RLVECN). RLVECN is defined to study and extract meaningful representations from social graphs to aid in node classification, CD, and link prediction problems. RLVECN utilizes an edge sampling approach for exploiting features of the social graph via learning the context of each actor concerning its neighboring actors. RLVECN also applies a convolutional neural network to extract facts from a social network and proceed with node classification and CD. However, when conducting the CD, RLVECN treats network clustering in a simplified way using node similarity, not considering node relationships.

The authors in [22] propose a semi-supervised solution named Seed Expansion with generative Adversarial Learning (SEAL). SEAL is based on a GAN, acting as the discriminator module. Their solution finds communities considering network topology and node attributes. The authors in [91] use GNN too, however, focusing on the overlapping aspect of communities in the CD problem. Both studies focus on static networks. On the other hand, [92] introduce a GNN model named Dynamic Graph Neural Network (DGNN) to deal with NA on dynamic networks. However, some aspects in their study, such as the model performance evaluation for CD and deleting edges or nodes as valid changes network changes.

In [93], the authors implement an architecture based on a network diffusion module to capture malicious behavior in a network. The model underlying this module represents a message passing through the network. In [37], a semi-supervised CD solution is implemented using GNN. The study undertaken by [38] also implemented a semi-supervised CD solution based on GCN and RL.

The studies described in this paragraph use RL to deal with the CD problem. The authors in [24] use the modularity maximization classical approach computed through a multi-agent system. In [94], the authors use *particle competition*, a different approach to CD. The authors in [23] use a classical approach to CD named *random walks* to aggregate an intelligent model of random walks as a problem-solving method.

The study conducted by [8] is a survey on the CD problem solved by game theory techniques. A highlighted game theory technique applied to CD is the *Evolutionary Game* with an evolutionarily stable strategy. The policy implemented on the RL agent can employ this approach to select the best action. The study of [83] also uses a game theory-based strategy.

The study of [95] focuses on benchmarking CD algorithms with a graph named *Lancichinetti-Fortunato-Radicchi* to choose the best CD algorithm for a given network. This study inspired the selection of the graph embedding technique in the proposed solution to evaluate the CD process.

This chapter presented the literature review results, including the protocol used with the planning, conducting, and reporting phases. Considering the studies written, the focus of AI approaches applied to the CD problem is increasing, as presented in Figure 3.3, with a visible growth beginning in 2019. Chapter 4 presents the CD proposed method.

Table 3.3: Related work overview.

Reference	**Problem**	**AI approach**	**Contribution**	**Advantage**	**Disadvantage**
[86]	Overlapping CD in social networks	N.A.	introduced an evolutionary game to remedy the defect of "complete rationality" that existed in past game theoretic methods	lightweight implementation able to process large networks	Tested only in static networks
[87]	Overlapping CD in social networks	N.A.	A new method to use the DPC	Fits well on heterogeneous community sizes	Tested only in static networks, it is not adequate to large networks
[38]	Improve the accuracy in finding communities	RL, GNN-autoencoder	A different strategy to find communities, a novel framework for CD using DRL, experiments with CD and semi-supervised CD	Improve the accuracy with good performance	Needs test with other approaches of RL
[37]	CD in OSN	GCN	Implemented an operator to compute convolutions on sparse matrices	Can operate with big networks	Tested only with macro-F1 and micro-F1 scores
[84]	Discover related topics and classify the more influential ones	N.A.	Provides a novel perspective for discipline knowledge structure analysis, which transcends conventional methods to map the knowledge domain based on the cooccurrence of keywords	Compares two ways to create a relationship network: co-word and Keyword-Citation-Keyword (KCK) using different techniques of NA, page rank analysis, and research topic analysis	Use only one data source - ACM combined with its area classification system. Restricts the analysis to compare co-word and KCK networks

Reference	Problem	AI approach	Contribution	Advantage	Disadvantage
[85]	How to present networked data to classic ML algorithms	N.A.	Survey different methods to transform graph data into vector space	Shows different ways to make embeddings, from network embeddings to node and edge embeddings	Do not apply any ML algorithm
[90]	Extract meaningful facts from social network structures to aid in node classification as well as CD tasks	CNN	Proposition of a DL-based and hybrid model, RLVECN, which aims at solving node classification problems in SNA, Detailed benchmark reports concerning classic objective functions used for classification tasks, Comparative analysis between RLVECN and state-of-the-art methodologies.	Define formally every aspect	Treat network clustering in a simplified way using node similarity
[22]	Using semi-supervised CD, try to find mode communities given several communities as training data	GAN, RL, Semi-supervised learning	search communities using a heuristic method, Uses dual learn to CD	Consider the topology of network and node attributes, Considers overlapping communities	complexity, requires training data
[93]	Tackle the malicious behavior in social media using ontological reasoning	Rule-based learning	powerful knowledge representation formalism with the capability to generate hypotheses via the application of existential rules	Shows a complete framework to deal with rule-based reasoning	The focus is on malicious behavior domain. There is no implementation with comparative results.

Reference	Problem	AI approach	Contribution	Advantage	Disadvantage
[24]	Modularity optimization through a decentralized approach	Multi-agent RL	Uses modularity maximization approach to identify communities in a decentralized way	Used real-world datasets. Can be adapted to deal with dynamic networks. The use of multi-agent allows distributed computation to deal with large networks	Tested only for disjoint communities
[94]	The focus is on the problem of concept drift in large data sets represented by temporal networks or data that can be transformed	RL	The study case is guided toward concept drift and concept relationships. Uses *particle competition* to detect communities	Defines a flexible framework that is compatible with other similarity functions or CD methods	Uses non-determinism to enhance the learning process. Does not compare with another solution.
[92]	Analysis of dynamic social networks	DGNN using Recurrent Neural Network (RNN) implemented by Long Short-Term Memory (LSTM)	A GNN model for dynamic NA and a comparative experiment for link prediction and node classification	Captures the dynamic aspects of social networks and outperform the state-of-the-art solutions in terms of Recall in link prediction and node classification	CD were not analyzed, only link prediction and node classification, the datasets used where small, the bigger one has 6,224 nodes and 19,496 edges. They only consider edge and node creations as network changes.

Reference	Problem	AI approach	Contribution	Advantage	Disadvantage
[83]	CD considering temporal dynamic aspects (dynamics consensus CD problem)	RL	Model CD as a Combinatorial Multi-armed Bandit problem	It is not restricted to graph model or specific community-change events	Uses only undirected edges, Considers only disjoint communities, Does not have a study case
[91]	Detection of overlapping communities	GNN	Introduce a GNN model for overlapping CD, makes available four datasets for overlapping CD and performs experiments with state-of-the-art solutions	Show superior performance when compared to different paradigms as probabilistic inference, non-negative- matrix factorization and deep learning	Restrict its analysis to static graphs, splits the model that uses the adjacency matrix (NOCD-G) from the model that uses node attributes (NOCD-X)
[23]	The weakness of non-intelligent models of the random walk as a problem-solving method	RL (learning automata)	Presents the Self-Avoiding Random Walks	Focus on using an intelligent approach using RL in random walks	The full-text is not available for free, and the author only sent the first chapter
[88]	Represent networked data as low-dimensional space	N.A.	It's a survey that compares methods for embedding homogeneous, heterogeneous, dynamic, attributed, and signed networks	Considers attributed networks when doing network embedding, analyzes many solutions	Doesn't give a clear conclusion. The focus is on methods for network embedding for link prediction

Reference	Problem	AI approach	Contribution	Advantage	Disadvantage
[89]	Detect overlapping communities in social networks using information about nodes to boost the qualify of communities detected	N.A.	Identify topics shared among nodes based on their content information	Consider overlapping community	Do not have a ground truth for topics discovered, Uses only algorithms for static detection (Louvain)
[8]	Distinguish groups of more densely connected nodes in a social network	N.A.	A survey on game theoretic approaches to CD in social networks	Covers many subfields of game theory applied to dynamic and static networks	There is no use of ML approaches
[95]	Benchmark CD algorithms using an approach different than Girvan & Newman	N.A.	Use another method to benchmark CD algorithms, i.e., LFR (Lancichinetti, Fortunato & Radichi) benchmark	The LRF benchmark generates networks with aspects that better represent real-world networks (largely heterogeneous degree distribution)	Only uses synthetic dataset

Chapter 4

Proposal

This research proposes a model for the CD problem in dynamic social networks. The Actor-Critic RL model is the basis for optimizing a community structure's modularity density. The gradient descent strategy called PPO helps to achieve optimization. The model core is the GNN architecture that moves the agent and the critic components using GAT layers. The AC2CD architecture is configurable to use GAT or GCN as neural network implementations. Chapter 5 presents the AC2CD implementation evaluation using two case studies. Though the main objective of our proposal is to tackle CD in dynamic social networks, our tests show application flexibility in dynamic and static contexts. Using a RL approach allows us to tackle CD in dynamic networks grouping the changes in snapshots and interactively dealing with them.

This chapter presents in Section 4.1 the formalization aspects of CD in dynamic social networks as an MDP. Section 4.2 details the AC2CD architecture describing each component. Section 4.3 shows the technologies used to implement the architecture and helped the case studies executions. Section 4.4 presents the AC2CD execution process describing the necessary tasks to start with a dataset and achieve detected communities in the dataset.

4.1 Model Formalization

CD through modularity optimization, as presented in Section 2.2.1, can be seen as a sequence of steps to improve the modularity score continually. Moreover, this stepwise improvement keeps the Markov property because the next improvement step depends only on the current one.

Given a social network modeled as a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V}$ is a set of nodes v and $\mathcal{E}$ a set of edges e, and a community structure $\mathcal{C}$. In this research, the CD problem in OSNs is formalized as an MDP $< S, A, R, p >$, where:

- S – The set of states s_t, in the time step t. Each state represents a community structure, i.e., given a node $v \in \mathcal{V}$ and a community $c \in \mathcal{C}$, a community structure is a set of pairs $< v, c >$.

- A – The set of actions a_t, in the time step t. The action space, i.e., the set of actions allowed in our model, adds a node to a community or removes a node from a community. Thus, given a node v as input, the output of the Agent's neural network is the probability distribution of v being in the communities c_i, with $i \in [1..|\mathcal{C}|]$.

- R – The reward function, with r_i be the reward received by the agent in the step i, such that:

$$R(.) = \begin{cases} 1, & \text{if the action received increased modularity density} \\ -1, & \text{otherwise} \end{cases}$$

- p – the transition function implemented by the environment to change between two consecutive states s_t and s_{t+1} when receiving an action a_t. In probability terms $P(s_{t+1}|s_t, a_t)$.

4.2 Architecture

The AC2CD architecture consists of a DRL approach based on GAT to find the optimal community structure in a dynamic social network. We use the message-passing feature of GAT as an element to propagate the label for each community, thus improving the modularity density of the community structure. The RL method chosen is Actor-Critic, implemented with PPO in the clipped version and Generalized Advantage Estimation (GAE) to compute the surrogate function of the policy gradient. According to [96], PPO performs the best in terms of profit and loss, training time, and data needed compared to Q-learning and deep Q-learning. It is worth noting that the proposed architecture can accommodate other implementations of GNN. In the current implementation, one can use GAT or GCN. The source code is in Python language and available to the research community.[1]

Figure 4.1 shows the AC2CD architecture overview highlighting the Actor-Critic components in gray. We find the layers of our Graph Neural Network (GNN) inside these components. The first layer is a *Dropout* regularization layer, followed by the first attention layer, named *GATConv1*. The *ReLU* activation and *Dropout* layers, and the second attention layer, *GATConv2*, the output of this GNN is a *Softmax* activation layer. The

[1]`https://gitlab.com/InfoKnow/SocialNetwork/ac2cd`

input data corresponds to network snapshots taken at each network change and embedded by the encoder in the data manipulation defining a DTDG (Section 2.2). The Edge list.txt is a file in the Coordinate Format (COO), or *ijv* format, where each line corresponds to an edge, i.e., a pair of node ids and possibly edges attributes, e.g., timestamp, weight, and other domain-specific features.

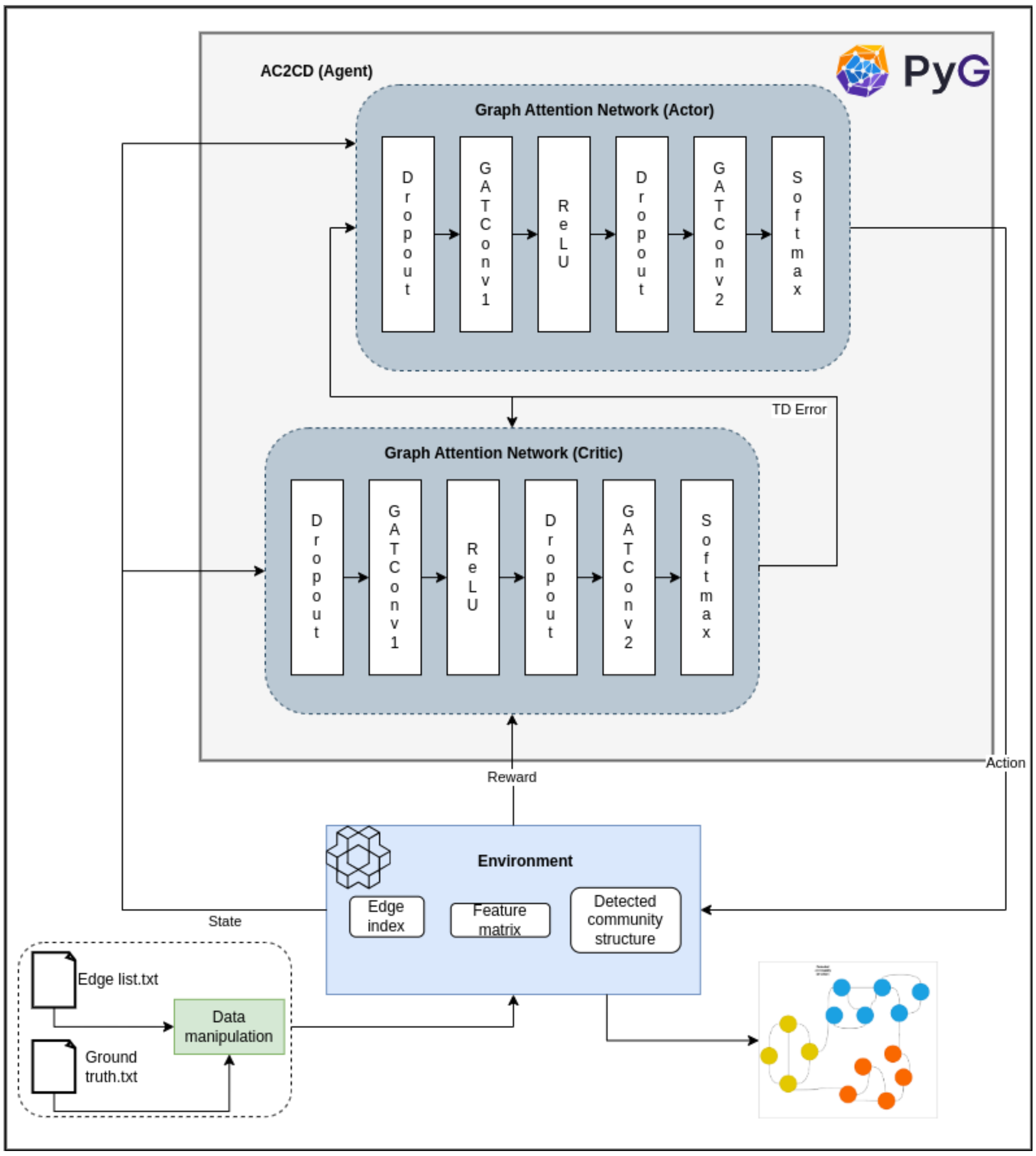


Figure 4.1: The AC2CD architecture.

Figure 4.2 highlights the content of the input files in the Data manipulation module of AC2CD, using Edge list.txt and Ground truth.txt files to generate embeddings. The Ground truth.txt file stores the node assignment to each community, i.e., for each node line in the file, there is a pair of node identification and community identification. Furthermore, Figure 4.1 presents the interaction between the agent (light gray), environment (blue), and internal aspects of these entities.

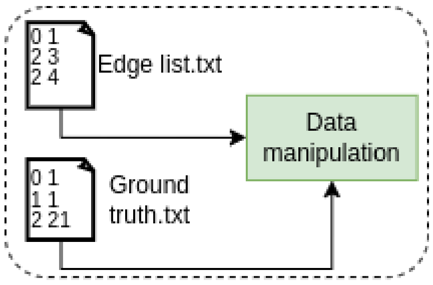


Figure 4.2: The AC2CD data manipulation

Each node is represented as embedded in a vector with 256 positions using the Node2Vector method defined by [31]. This results in a matrix M_{256xn}, where n is the number of vertexes. The Edge list.txt might include different network formats, where dynamic networks are a function of time (temporal) or static networks (non-temporal). The edge list in a temporal dataset has the following format:

```
582 364     0
168 472  2797
168 912  3304
  2 790  4523
  2 322  7926
  2 790  8061
 42 402 19403
870 337 19560
663 362 21077
663 410 21280
```

The presented temporal dataset in COO format has its third column representing the edge's timestamp with ten lines (first ten edges) from the Email-Eu-core detailed in Section 5.1. The BlogCatalog non-temporal dataset takes the following format:

```
1,176
1,233
1,283
1,371
1,394
1,446
1,585
1,645
1,667
1,696
```

The Ground truth.txt file includes the valid assignment of each node to its corresponding community. It is a comma-separated file with two fields. The first corresponds to the node identification and the second to the community identification. The following list represents the ten first lines of the Ground truth.txt file of the BlogCatalog dataset.

```
 28,1
 32,1
 36,1
 37,1
 84,1
129,1
138,1
169,1
172,1
218,1
```

The Data manipulation module's role is to make uniform these differences in the original representation of the datasets (temporal and non-temporal). Based on the hyperparameter *snapshots*, the Data manipulation module splits the input data after splitting the training, test, and validation data. Then, the split data is embedded in a low-dimensional space using the Node2Vec. This way, we assume that every network node is present in all snapshots until the learning process ends.

Figure 4.1 also shows the *Environment* component presenting its data structures, as illustrated in Figure 4.3, including the edge index of the network, and a COO list representing the connectivity of the network. The role of the environment component is to simulate the network dynamics at each timestep. The environment manages the action and observation spaces, which are initialized at the beginning of the execution, i.e., at the time step t_0. At each time step t, the environment receives an action a_t generated by the agent and computes the effects produced a_t, developing a new state s_{t+1}, a reward r_{t+1}, and indicating if the state is terminal (line 9 of the Algorithm 2).

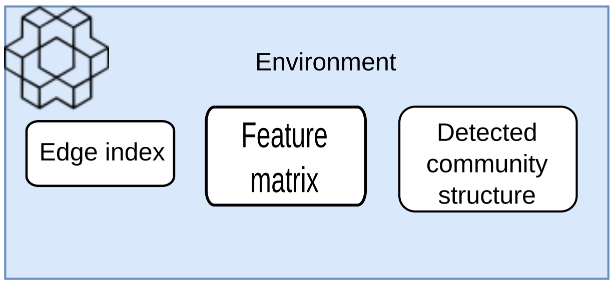


Figure 4.3: The AC2CD environment.

AC2CD Algorithm

The learning process begins with the agent receiving the current state from the environment. The current state is a representation of the community structure of the network. After the recent state observation, the Actor agent passes through the node list issuing the probability distribution of each node in each community. The Critic agent, in its turn, computes the modularity density for the community structure. The difference between the current value of the modularity density from ground truth and the previous prediction issued by the Critic corresponds to the TD error.

The RL action space represents the possible combination of assignments between node and community. The reward function is implemented as the difference between the current value for modularity density for the community structure, as described in Equation 2.4 for each network snapshot and the previous one. A positive reward indicates an improvement in the modularity density. Otherwise, a negative reward is issued.

The Algorithm 2 presents the main loop of AC2CD. The input of the algorithm (represented by Input:) is the *Dataset* consisting of the two input files (Edge list.txt and Ground truth.txt) and the hyperparameters (*Hp*) file described in Appendix A.1. In Line 1, the Data manipulation module (*DataManip(Dataset)*) first creates the node embeddings of the input network. It splits the dataset, i.e., the result of the *DataManip* method is the list of node embeddings (*node_ emb*) and the list of edges grouped by snapshots (*edge_ index*), according to the hyperparameter snapshots and the split ratio for train and test.

In Line 2, the environment is created (*env*) by the method (*GATEnv*), which receives the *node_ emb* and *edge_ index* to create the features matrix to represent nodes, create its copy of *edge_ index* and initialize the detected communities.

In Line 3, the Agent (*agent*) is created by the method *Agent(Hp)* according to *Hp*. The creation of the Agent implicitly creates the Actor, the Critic networks, and the experience memory to store the last episodes of the execution.

In Line 4, we note that a list of modularities (*score_ history*) stores the scores produced for each episode.

In the first interaction of the algorithm, the community structure initializes with each node receiving a random attribution to a community. Line 5, the environment is "reset" at Line 6, and the main loop begins. At each iteration, while not in a terminal state, the Agent chooses an action based on the observation provided by the environment (Line 8) and passes that action to the environment (Line 9), which executes a step returning a new observation, the reward, and a flag indicating whether the new state is terminal. For our implementations, the environment returns a terminal state when all nodes update their community assignment, even if the new community is the same as the older one. In

Line 10, the current observation, the last action, reward, Actor output (*prob*), and Critic output (*val*) are stored in the Agent's experience memory. At $mod(Hp.train_interval)$ interactions (Line 11), the Agent goes for a train session, Line 12, to update its policy π_θ and value function $\hat{q}_\omega(s,a)$. The training process follows the regular training for any GNN-based RL model using the policy gradient PPO, which consists of a loop where the parameter θ of the objective function is updated. After this process, we have the model file ready to be used to infer the community assignment for nodes in a network and the evolution of the score stored in the *score_history* variable.

Algorithm 2: Community Detection in AC2CD

```
   Input: Dataset
   Input: Hp
1  $node\_emb, edge\_index \leftarrow DataManip(Dataset)$;
2  $env \leftarrow GATEnv(node\_emb, edge\_index)$;
3  $agent \leftarrow Agent(Hp)$;
4  $score\_history \leftarrow []$;
5  while $n < Hp.max\_iter$ do
6      $obs \leftarrow env.reset()$;
7      while not done do
8          action, prob, val ← agent.choose_action(obs);
9          new_obs,reward,done ← env.step(action);
10         agent.remember(obs, action, reward,prob,val);
11         if $n \% Hp.train\_interval == 0$ then
12             agent.learn();
```

4.3 Technologies

This section describes the technologies used to implement each component of the AC2CD architecture and the hardware used to run the study cases. This description aims to aid the proposed model's reproducibility and extensibility by showcasing the implementations' library dependencies.

We used some libraries in the experiments to avoid re-implement established solutions, such as the basic library blocks of GNN and the library for the environment block of RL. The PyTorch Geometric library [97][2] was used to implement the GNN-related stuff like GAT layers (*GATConv1* and *GATConv2*) and the datasets of the experiments. The environment component of RL was implemented as a custom environment using the Gym library [98] as the basis. The matrix manipulation and computation out of tensor arith-

[2]Available at: `https://pytorch-geometric.readthedocs.io/`

metic were done with the Numpy library [99].[3] The metrics used in the experiments and presented in Section 2.2.3 are available on the Scikit learn library.[4]

4.4 Executing AC2CD

The AC2CD execution process includes three main sets of activities, pre-processing, training and testing, and execution. Figure 4.4 presents the process from defining the *Hp* for the network embedding and model training until the Show results task with the communities assigned to each node.

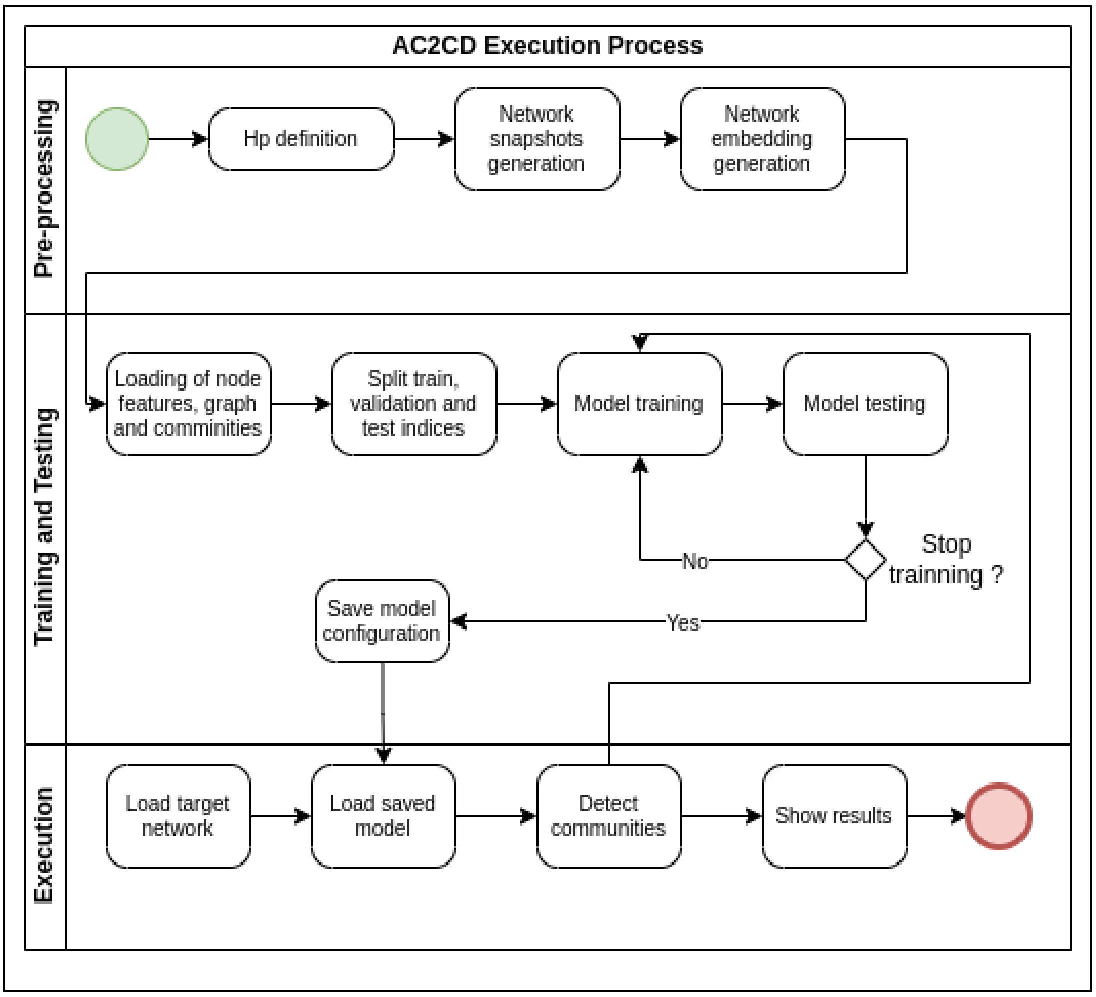


Figure 4.4: The AC2CD execution process.

The Pre-processing is the starting point of the AC2CD execution. The first task in this process is the definition of the *Hp*. These hyperparameters are responsible for configuring the framework, as presented in Section A.1. The second task is the generation of the

[3]Available at: `https://numpy.org/`
[4]Available at: `https://scikit-learn.org/stable/`

snapshots in case the dataset is not temporal (i.e., the edges without timestamps). The third task is the generation of the network embedding using the Node2Vec. However, any other embedding technique that preserves the community structure can be applied.

The Training and Test process takes the vectors embedding the input graph and the corresponding edge index, splitting the nodes to train and test the model. The training process runs until the maximum iteration or the "patient threshold" are reached. The patient threshold is an empirically defined value corresponding to the number of training iterations without modification in the loss. The model configuration is saved once the training is over (i.e., the matrices with weights for the Actor and Critic networks). During the training session, we can use the model to make predictions, but the accuracy of the model is not guaranteed.

The execution process is responsible for loading the network to be analyzed, loading the previously trained model, and making the predictions of the community, showing the community structure in a set of text files, each corresponding to the network structure detected for the respective network snapshot.

To execute the AC2CD architecture, one must install the dependencies in the requirements.txt file, described in Section A.2, which contains a list of libraries required to execute the entire workflow. Once installing the required libraries, you can configure the framework with an environment file, choose the dataset, and finally run the main file, i.e., main_ppo.py. During the execution, a new embedding file is created in the current directory if there is no embedding file for the desired dataset. Once completing the network embedding, the training and test process begins following the respective set of *Hp*. When this process finishes, we have the trained model persisted as two files, one for the Actor network and another for the Critic network.

Chapter 5 presents the experiments executed to validate the proposed model. We conduct a comparison with classical and the state of the art solutions for CD using real-world datasets and discuss the results. The conducted experiments to evaluate the accuracy of AC2CD-implemented architecture consisted of feeding the environment with an embedded network version by the Data manipulation module. At each iteration, the cycle triggers the training and testing phases. The agent computes a node assignment (action) and submits this action to the environment, returning the reward and the corresponding network structure as the next state. The objective is continual to improve the modularity density at the end of each episode or until reaching a stalled state for the modularity density.

Chapter 5

Experiments and Discussion

This section describes the experiments conducted to verify the efficacy of the AC2CD architecture proposed. Section 5.1 describes the datasets used to validate the proposed model highlighting the different aspects of static and dynamic networks. Section 5.2 presents a comparative study with four implementations representing different theoretical approaches. Section 5.3 offers a second round of experiments focusing on the heterogeneity of the datasets and the impact of this aspect on the performance of the proposed model.

5.1 Datasets

The dataset contains the friendship network crawled and group memberships. The experiment used five real-world datasets: Email-Eu-core, BlogCatalog3, Flickr , Youtube2, and High School, categorized in static or dynamic depending on the presence of annotated timestamps in the edge list. We choose some datasets to allow a direct comparison with state-of-the-art studies as [37] and [38]. We found the datasets with the help of the Colorado Index of Complex Network Project (ICON),[1] downloaded from the Social Computing Data Repository of Arizona State University [101],[2] and the Stanford Network Analysis Project.[3] Table 5.1 describes basic statistics of the used datasets, including the number of nodes, edges, communities, and the indication of whether the dataset is dynamic.

The Email-Eu-core dataset was generated using email data from a large European research institution. The emails represent communication between institution members (the core). The dataset does not contain incoming messages from or outgoing messages to the rest of the world. In the list of edges file, a line represents a directed edge (u, v, t),

[1]Available at `https://icon.colorado.edu/`. Accessed on 2022-01-10.
[2]Available at `http://datasets.syr.edu/pages/datasets.html`. Accessed on 2022-01-16.
[3]Available at `https://snap.stanford.edu/data/index.html`. Accessed on 2022-01-16.

which means that person u sent an email to person v at time t. We created a distinct edge for each email recipient. The communities of this dataset represent departments, i.e., community members are persons who work together in the same department.

Table 5.1: Dataset characterization with the number of nodes, edges, communities, and dynamicity.

Dataset name	**# Nodes**	**# Edges**	**# Communities**	**Dynamic**
High School	329	45047	9	Yes
Email-Eu-core	1,005	25,571	42	Yes
BlogCatalog3	10,312	333,983	39	No
Flickr	80,513	5,899,882	195	No
Youtube2	1,138,499	2,990,443	47	No

Intuition tells us that people inside a community are more connected than outside. Thus, people working in the same department have more probability of sending emails in the same department. Figure 5.1 presents the community distribution in this dataset. The Email-Eu-core dataset shows a heavy tail in the distribution of members for communities and the concentration of almost 10% (109 members) of members in a single community and two districts with one member. However, a community with only one member seems like a paradox, but we extracted this information directly from the ground truth.

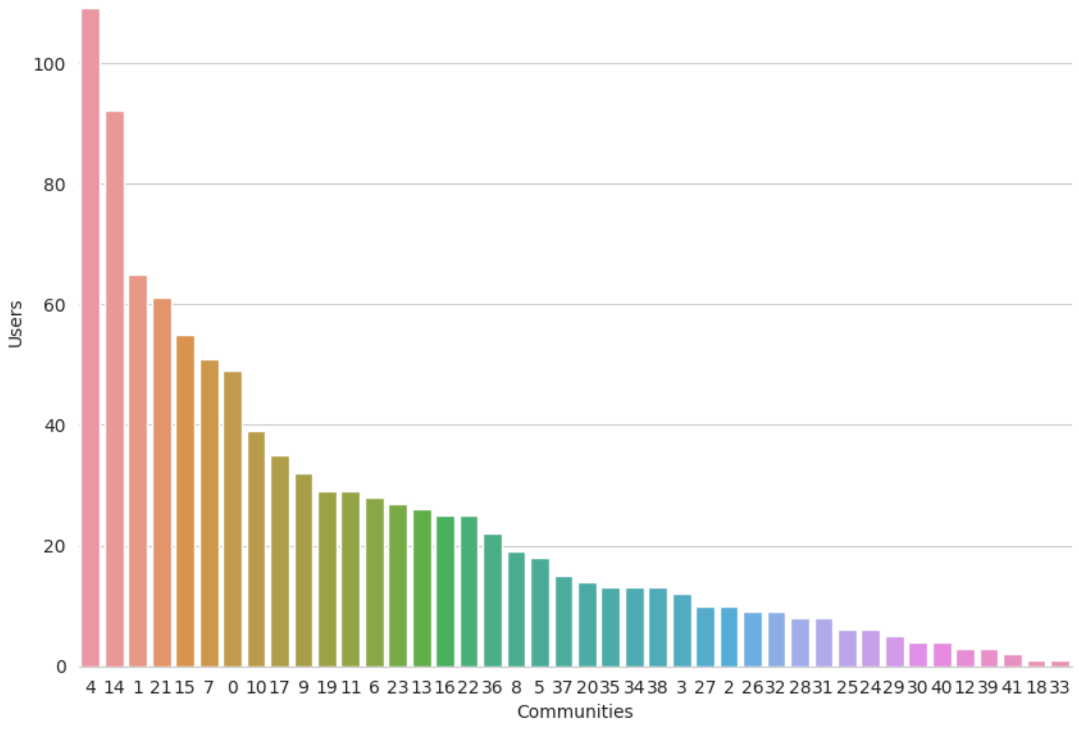


Figure 5.1: The Email-Eu-core communities histogram.

The BlogCatalog3 is a non-temporal dataset created from the network of a social blog directory that manages the bloggers and their blogs. We crawled this dataset from the BlogCatalog website.[4] The contact network and selected group membership information are in two files: the edge index and the community attribution. This dataset implements an undirected network with an edge *1,2* means blogger with id "1" is a friend with blogger id "2". Thus, when handling it, the data manipulation creates the snapshots as configured by the *Hp*. Figure 5.2 presents the distribution of participants in each community of this dataset. Here, we see a heavy tail profile but with a concentration of 16% of members in the most extensive community. The second biggest community with almost half members of the first one (1623/986).

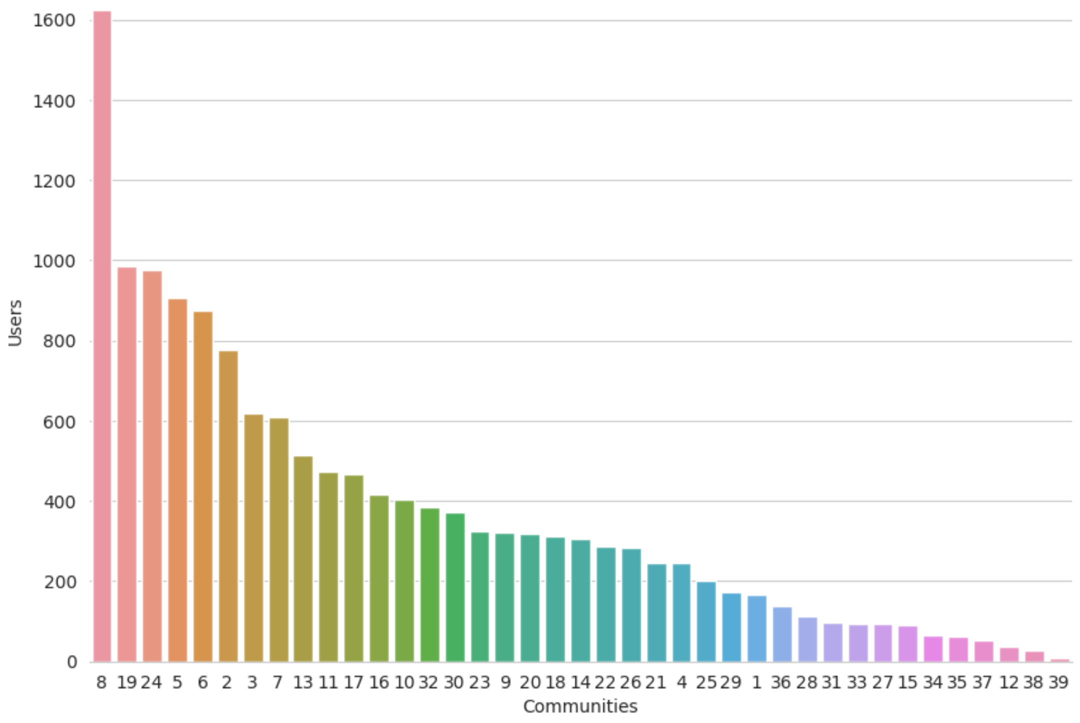


Figure 5.2: The BlogCatalog3 communities histogram.

The Flickr dataset represents the network crawled from the Flickr website.[5] Flickr is an image and video hosting website, web services suite, and online community. Both the contact network and selected group membership information are in this dataset. Each edge implements a friendship relationship. Nevertheless, this dataset represents a social network, and the timestamps of the edges are not present. Thus, this dataset is considered non-temporal. Figure 5.3 presents the community distribution in this dataset. This dataset has the most extensive number of communities (195). However, this represents

[4]`http://www.blogcatalog.com`
[5]`https://www.flickr.com/`

the more significant relationship between members of the community. The more extensive community concentrates 17% members, and the second one only 8%. The concentration of members in only two communities is 25%, i.e., 1% of communities focus 25% of members. This dataset is among the most challenging for its number of nodes, number of communities, and heterogenous distribution of members for each community.

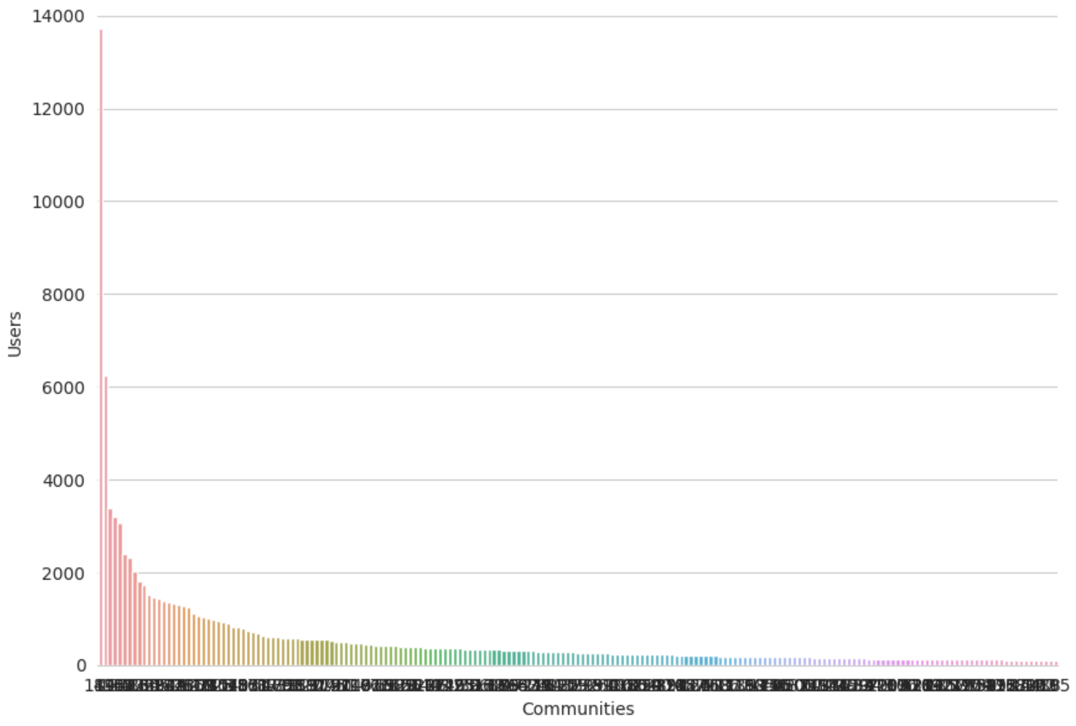


Figure 5.3: The Flickr communities histogram.

The Youtube2 dataset crawled from the YouTube website,[6], and each edge represents a friendship relationship in the network. Youtube is a video-sharing website where users can upload, share, and view videos. Figure 5.4 presents the community distribution in this dataset. Once more, this network offers a heavy tail. However, showing a homogeneous distribution of community members, the more extensive community has only ten times more members than the smaller one, and the 13 smaller communities have less than 120 members.

The High School dataset is a collection of networks containing the temporal network of contacts between students in a high school in Marseilles, France. The first network presents the students' connections of three classes over four days in December 2011. The second network corresponds to the students' contacts of five courses during seven days in November 2012 (from Monday to Tuesday of the following week). Figure 5.5 presents the community distribution in this dataset representing a class the student is enrolled.

[6]`http://www.youtube.com`

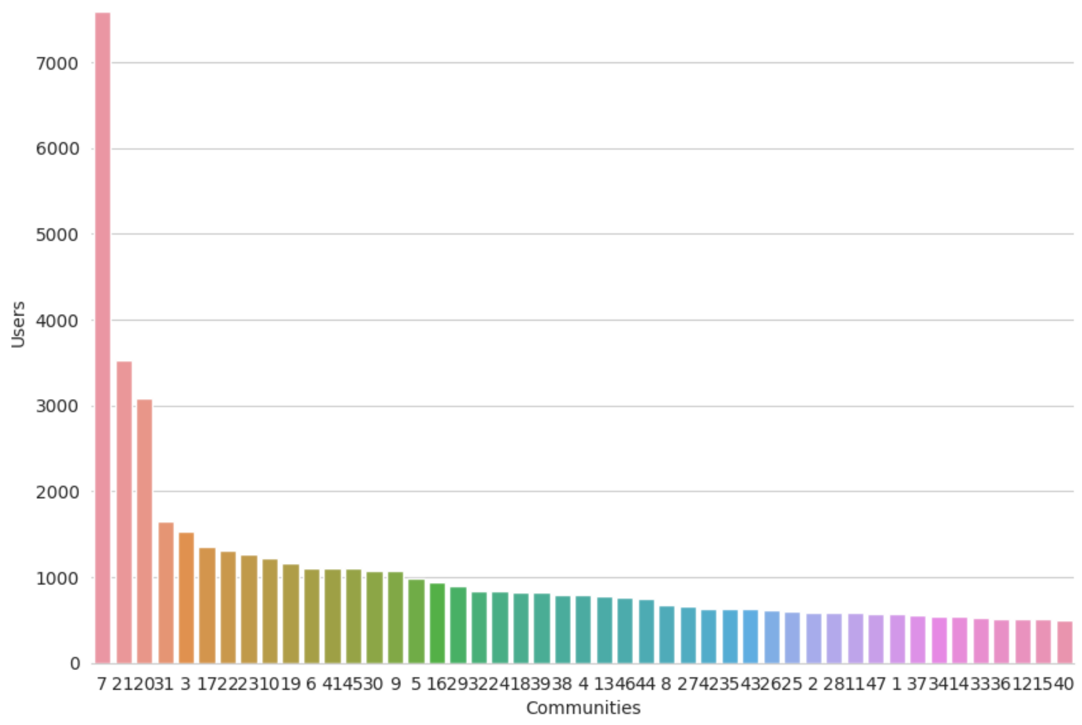


Figure 5.4: The Youttube2 communities histogram.

Thus, each color is a class with nine communities (i.e., PC, 2BIO3, PC*, MP*2, 2BIO1, 2BIO2, PSI*, MP, MP*1). The High School is the most homogeneous dataset used in the experiments, with the smaller community representing 66% the number of members of the bigger one.

In the training context, our proposal adopts inductive learning using a different data set to train and test. The labeled nodes of these datasets are used according to a baseline of comparison in [37]. We split the BlogCatalog3 dataset in the proportion of 10%, 30%, 60%, 90% of labeled nodes, and the other datasets are 1%, 3%, 6%, and 9%. This progression makes the prediction task more manageable, as used in the baseline. Figure 5.6 presents the distribution of members in communities for each dataset. As previously described, the Flickr dataset carries the most heterogeneous members distribution, with many communities represented as outliers. Almost the same is valid for the Youtube2 dataset, where three communities are considered outliers. As the statistics tell, an outlier is a value far away from the focal group of values.

As shown in Table 5.1 and Figure 5.6, the Flickr network has the most extensive number of communities, and Youtube2 has the most considerable number of nodes and edges. Both datasets present a heterogeneous distribution of nodes by communities. Those aspects make these two datasets the most challenging to find communities, mainly for the number of communities and the possible diversity in these communities. High School and Email-EU-Core present a more homogeneous distribution of nodes by communities. However, they are temporal, making them interesting for our analysis.

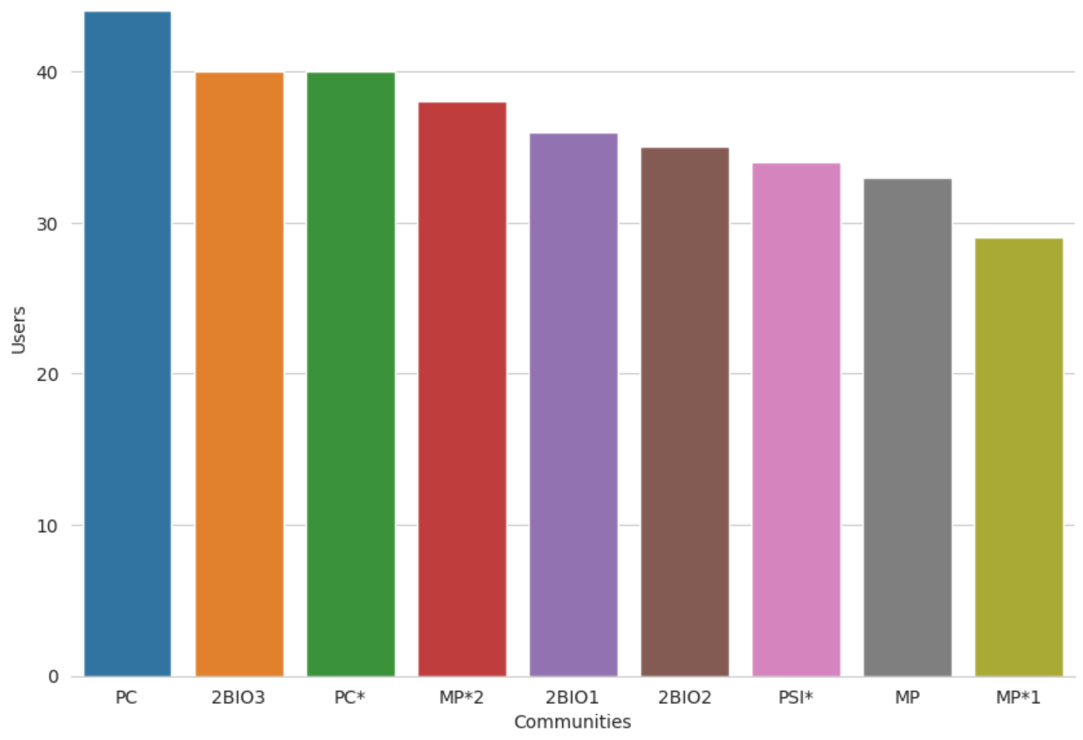


Figure 5.5: The High School communities histogram.

5.2 Case Study 1

This experiment analyzed the performance of the AC2CD compared to different methods and implementations. We choose structural embedding, community embedding, GAN, and GCN methods for their highlighted relevance in CD. The comparison study was executed in four different datasets as presented in Section 5.1.

The experiment used real-world datasets comparing the results with state-of-art approaches presented by [37], named CNN, and [38], named CLARE. It is worth observing that the experimental results presented in [37], especially concerning the CNN algorithm, which is not available, were directly imported into our results containing the use of the same evaluation metrics. We used the confidence interval metrics for micro-averaged Micro-F1 and Macro-F1 rendered by [72] and presented in Section 2.2.3. Furthermore, to evaluate the accuracy of CD methods, we used the NMI score defined by [75], comparing our results to the algorithms described in Section 5.2 using Email-EU-core dataset (i.e., SDNE, ComE, GraphGAN, and CLARE).

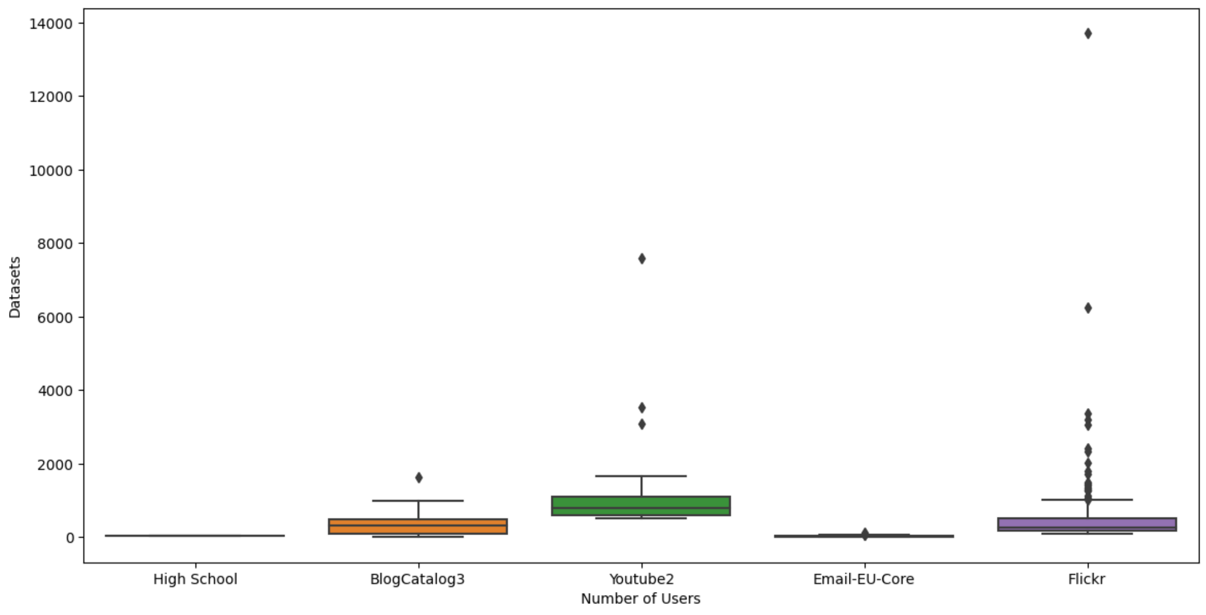


Figure 5.6: The community members distribution.

5.2.1 State-of-the-art Comparisons

The methods present in this comparative study represent different ways to use ML for CD. They are representatives of solution categories and have implementations available in public repositories. We used the methods in [37] and [38], allowing us to compare the approaches directly.

SDNE

The Structural deep network embedding (SDNE) is a semi-supervised deep network model that exploits first-order and second-order proximity to preserve the network structure [36].[7] We used the second-order proximity by the unsupervised component to capture the global network structure. At the same time, the first-order proximity is used as the supervised information in the supervised part to preserve the local network structure. By jointly optimizing them in the semi-supervised deep model, this method can maintain both the local and global network structure and is robust to sparse networks. The SDNE architecture overview is present in Figure 5.7, which comprises multiple nonlinear mapping functions to map the input data to a highly nonlinear latent space to capture the network structure. In this model, each vertex is embedded using an unsupervised approach and then pair-wised using a supervised component based on Laplacian Eigenmaps.

[7] Available at `https://github.com/suanrong/SDNE`

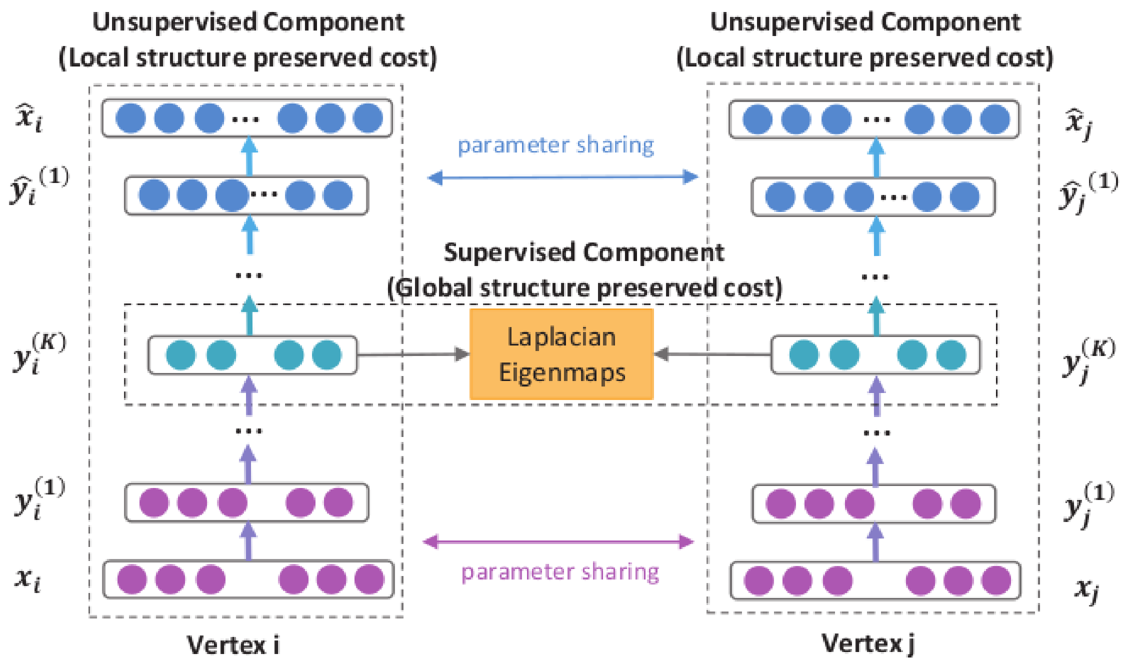


Figure 5.7: The SDNE architecture. Source: [36].

ComE

The ComE defines a method that relies on the node and community embedding for learning graph embeddings in a closed loop among community embedding, CD, and node embedding [35].[8] On the one hand, node embedding can help improve CD, which outputs good communities for fitting better community embedding. On the other hand, we can use community embedding to optimize the node embedding by introducing community-aware high-order proximity. ComE closed loop for learning community embedding is present in Figure 5.8, where each edge denotes the transition between each task. The first one is the Node embedding. This task is inspired by the Deep Walk [102], followed by the CD using spectral clustering and the community embedding using Gaussian Mixture Model (GMM).

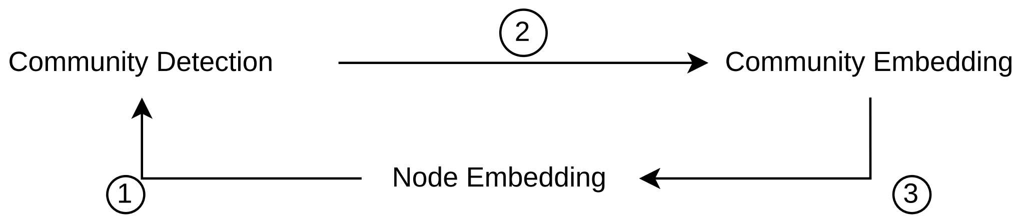


Figure 5.8: The ComE architecture. Source: [35].

[8]Available at `https://github.com/andompesta/ComE`

GraphGAN

The GraphGAN is a graph representation framework proposed by [34][9] that implements the GAN approach unifying generative and discriminative thinking for graph representation learning.

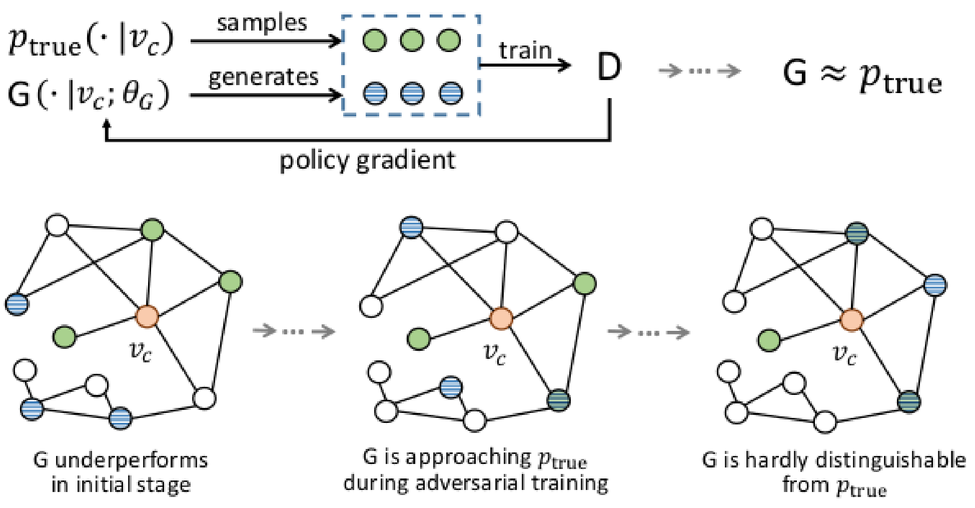
The GAN formulation in GraphGAN follows the terms. Let $G = (\mathcal{V}, \mathcal{E})$ be a given graph, where $\mathcal{V} = \{v_1, ..., v_V\}$ represents the set of vertices and $\mathcal{E} = \{e_{ij}\}_{i,j=1}^{V}$ represents the set of edges. For a given vertex v_c, $\mathcal{N}(v_c)$ is defined as the set of vertices directly connected to v_c, the size of which is typically much smaller than the total number of vertices V. The conditional probability $p_{true}(v|v_c)$ denotes the underlying true connectivity distribution for vertex v_c, which reflects v_c's connectivity preference distribution over all other vertices in $\mathcal{V}$. From this point of view, $\mathcal{N}(v_c)$ can be seen as a set of observed samples drawn from $p_{true}(v|v_c)$.

Specifically, GraphGAN aims to train two models during the learning process: (1) Generator $G(v|v_c)$, which tries to fit the underlying true connectivity distribution $p_{true}(v|v_c)$ as much as possible, and generates the most likely vertices to be connected with v_c, and (2) Discriminator $D(v, v_c)$, which tries to distinguish well-connected vertex pairs from ill-connected ones, and calculates the probability of whether an edge exists between v and v_c. In the proposed GraphGAN, the generator G and the discriminator D act as two players in a *minimax* game: the generator tries to produce the most indistinguishable "fake" vertices under guidance provided by the discriminator, while the discriminator tries to draw a clear line between the ground truth and "counterfeits" to avoid being fooled by the generator. Competition in this game drives both to improve their capability until the generator is indistinguishable from the accurate connectivity distribution. Figure 5.9 presents the architecture overview of GraphGAN and the evolution of an execution highlighting the role of the Generator G and Discriminator D.

CLARE

The study undertaken by [38] presented CLARE[10] framework consisting of two key components: Community Locator and Community Rewriter. The community locator can quickly locate potential communities by seeking subgraphs similar to training ones. Specifically, CLARE encodes communities into vectors, measures the similarities between communities in the latent space, and then discovers candidates based on the similarities with the nearest neighbors matching strategy. The community rewriter further adjusts those candidate communities by introducing global structural patterns. CLARE frames such refinement process as a DRL task and optimizes this process via policy gradient. For

[9]Available at `https://github.com/hwwang55/GraphGAN`
[10]Available at `https://github.com/FDUDSDE/KDD2022CLARE`



$p_{\text{true}}(\cdot|v_c)$
samples
$G(\cdot|v_c;\theta_G)$
generates
train
D
$G \approx p_{\text{true}}$
policy gradient
v_c
G underperforms in initial stage
G is approaching p_{true} during adversarial training
G is hardly distinguishable from p_{true}

Figure 5.9: The GraphGAN architecture. Source: [34].

located communities, the rewriter provides two actions: adding outer nodes or dropping existing nodes, thus refining their structures flexibly and intelligently.

The core of CLARE is a GCN that learns to encode nodes and community representations. Figure 5.10 presents the CLARE architecture overview with emphasis on the two main components, at left the Community Locator and right the Community Rewriter. Note that DRL is used only in the Community Rewriter.

The Community Rewriter component implements DRL with the following specification. The state is a predicted community united with its outer boundary. The action is a combination of $(a_t^{exclude}, a_t^{expand})$, i.e., at each time t, one node is excluded, and another one is included in a community C_t. The reward signal is taken directly from the F1 score of the community structure.

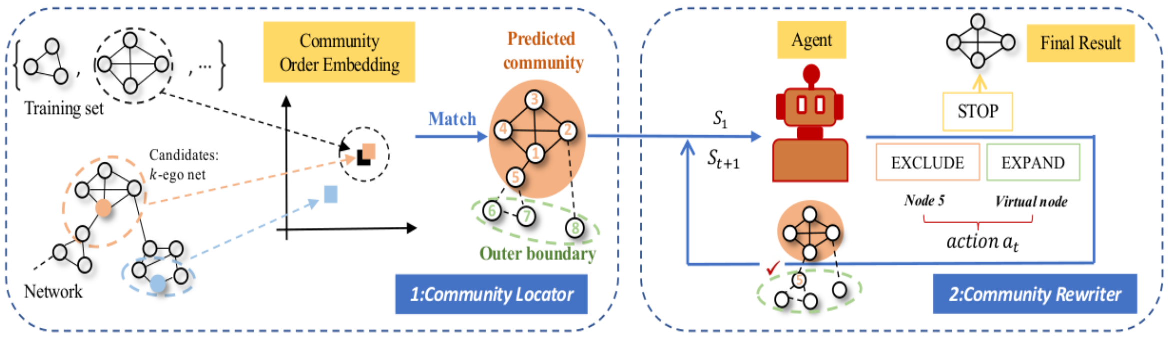


Figure 5.10: The CLARE architecture. Source: [38].

Results

The analysis of the Email-Eu-Core network presented by [103] illustrates the community network topology as in Figure 5.11. As noted by [101], the largest community has 98% of node concentration. Table 5.2 summarizes the results highlighting the excellent performance of AC2CD. The results of the Macro-F1 score in Figure 5.12 resemble the excellent performance of CLARE architecture using the GCN algorithm to learn such concentrated edges in one node. The results of the Micro-F1 score in Figure 5.13 are not so good to capture such regularity in the density distribution of communities.

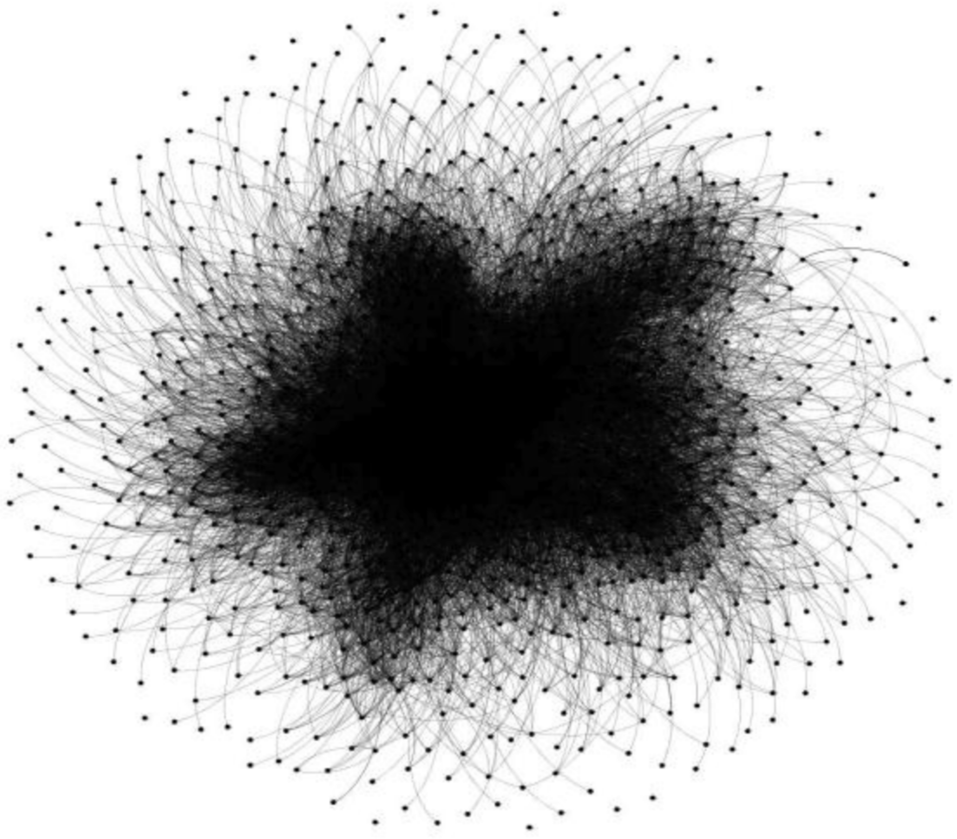


Figure 5.11: The Email-EU-Core community network topology. Source: [103].

Table 5.2: Experimental results comparison with Email-Eu-core dataset.

	Lableled Nodes							
	1%		**3%**		**6%**		**9%**	
	Micro-F1	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro F1**	**Macro-F1**
GraphGAN	25.66	14.88	36.01	20.54	39.71	26.87	44.36	31.55
ComE	24.65	13.21	35.64	18.78	39.21	25.98	39.71	29.26
SDNE	28.02	16.88	33.70	21.94	38.48	25.11	39.08	29.32
CLARE	38.32	31.72	38.67	31.98	38.78	32.92	38.92	34.10
AC2CD	**30.22**	**17.26**	**40.47**	**23.14**	**46.21**	**31.36**	**46.60**	**35.69**

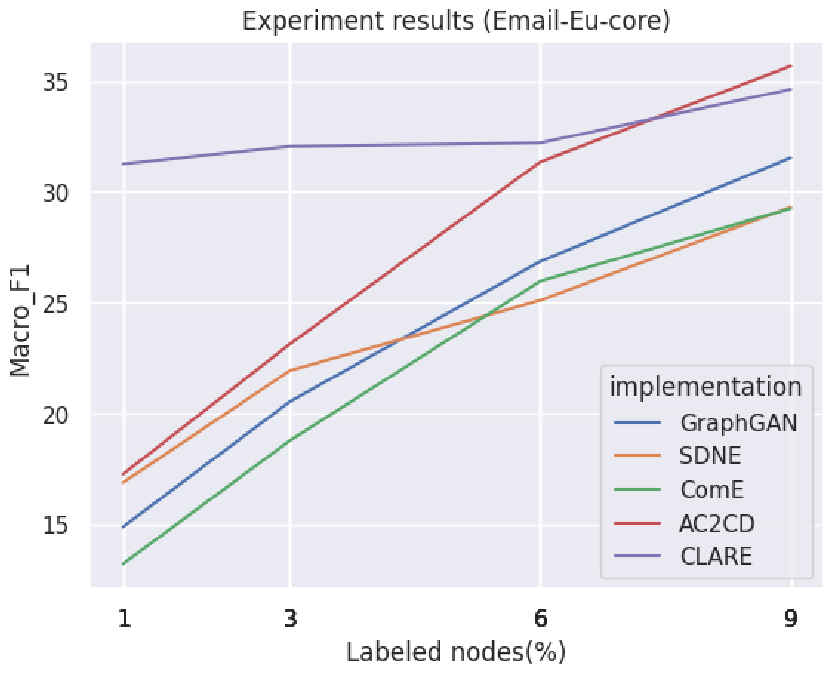


Figure 5.12: Macro-F1 assessment for Email-EU-Core dataset.

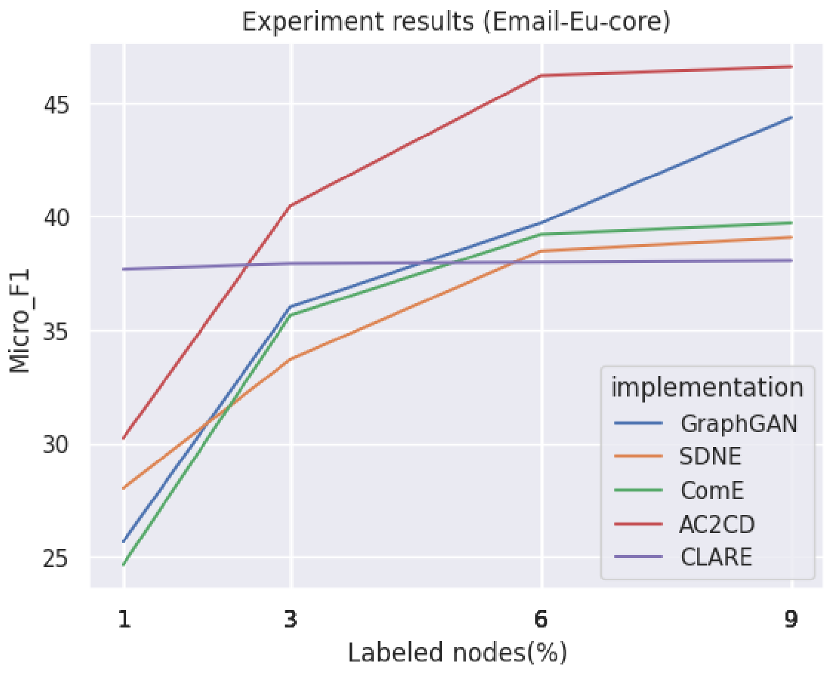


Figure 5.13: Micro-F1 assessment for Email-EU-Core dataset.

The authors in [104] propose a novel method for multi-task learning-based network embedding as presented in Figure 5.14. Note that there is a regular distribution of nodes in each community. Table 5.3 summarize the results highlighting the excellent performance of AC2CD. The results of Macro-F1 in Figure 5.15 present good performance with the CLARE using the GCN algorithm to learn the community structure as the network is composed of a homogeneous community distribution. However, the Micro-F1 in Figure 5.16 presents not-so-good results of the GCN since this metric computes the sensitivity of the difference among density distribution of communities, where more elaborate learning algorithms are needed to capture such complex features.

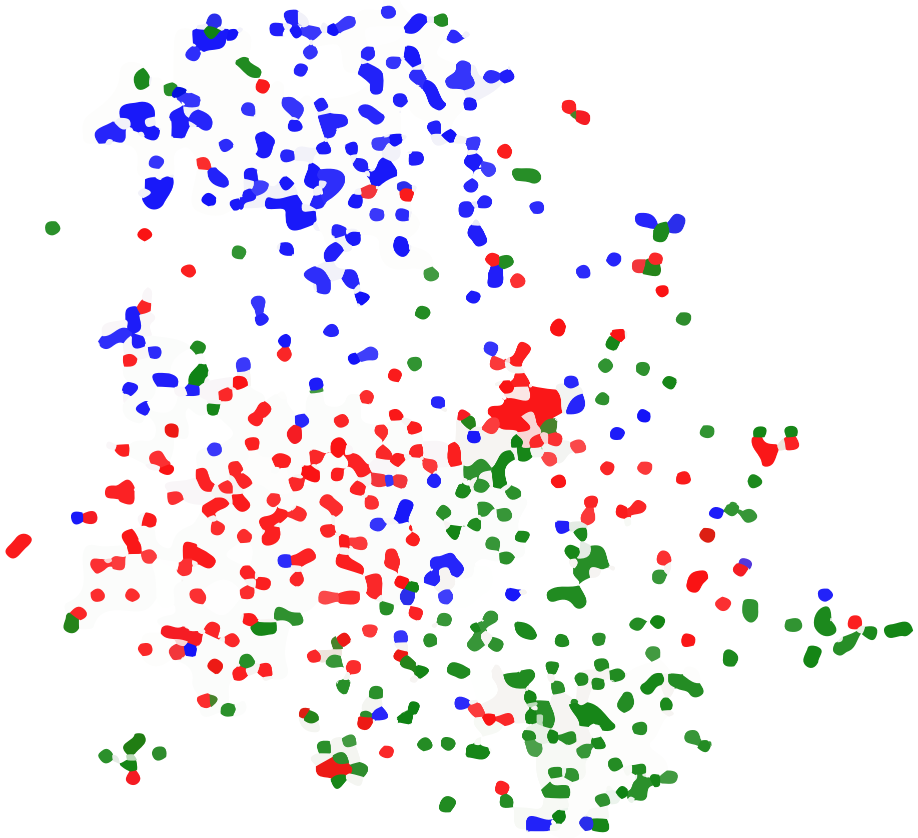


Figure 5.14: The BlogCatalog community network topology. Source: [104].

Table 5.3: Experimental results comparison with BlogCatalog3 dataset.

	Lableled Nodes							
	10%		**30%**		**60%**		**90%**	
	Micro-F1	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**
GraphGAN	28.78	17.88	39.01	23.54	42.71	29.87	44.76	33.01
ComE	27.18	16.21	38.64	22.78	41.68	28.98	44.12	32.46
SDNE	31.11	19.88	36.70	24.94	41.88	28.11	44.88	31.22
CLARE	30.51	36.72	31.67	36.98	31.70	37.02	31.92	38.12
AC2CD	**33.18**	**20.41**	**43.62**	**27.34**	**49.36**	**35.56**	**51.85**	**40.35**

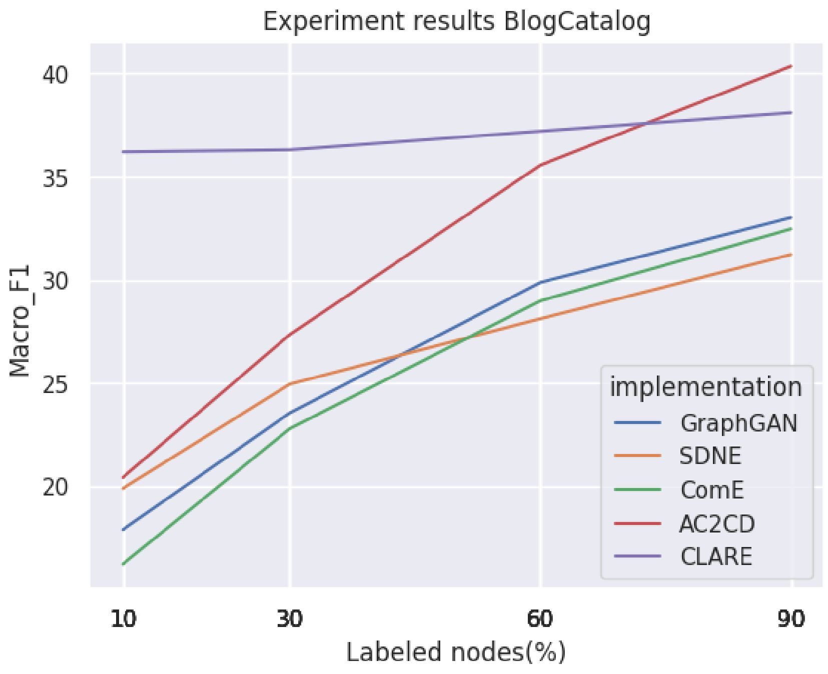


Figure 5.15: Macro-F1 assessment for BlogCatalog dataset.

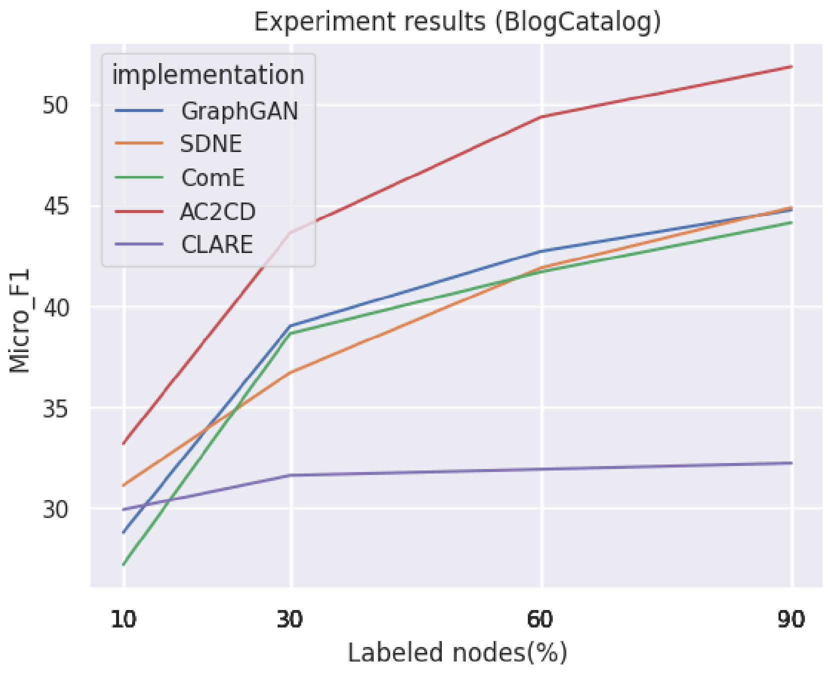


Figure 5.16: Micro-F1 assessment for BlogCatalog dataset.

The authors in [105] use the Flickr network dataset as presented in Figure 5.17 to validate their peer prediction based trustworthy service rating system for social networks. Note the complexity of the community network topology. Table 5.4 summarizes the results highlighting the excellent performance of AC2CD. The results of Macro and Micro-F1 scores in Figures 5.18 and 5.19 resemble the GCN difficulty to detect the community structure, where the actor-critic approach presents the best results being able to capture such complex features in the community structure. We highlight that such a dynamic community network topology associated with the Micro-F1 metric sensitivity resembles the importance of more robust approaches to CD.

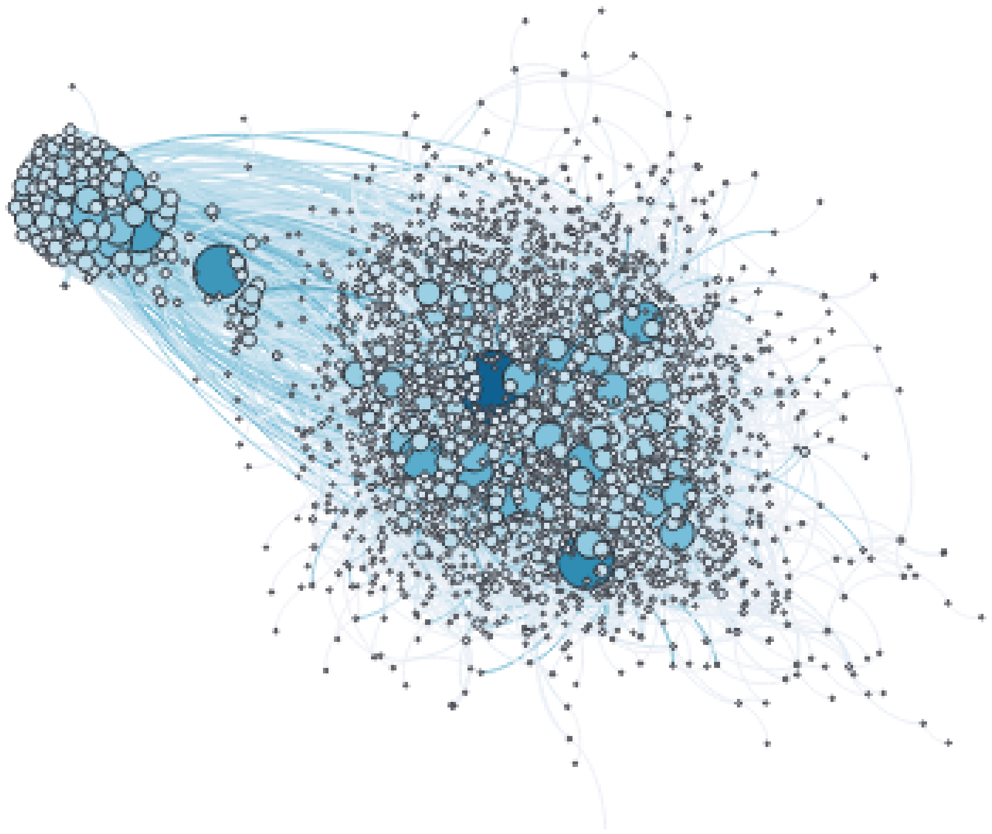


Figure 5.17: The Flickr community network topology. Source: [105].

Table 5.4: Experimental results comparison with Flickr dataset.

	Lableled Nodes							
	1%		**3%**		**6%**		**9%**	
	Micro-F1	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**
GraphGAN	23.01	13.92	33.10	19.91	37.77	25.63	42.32	27.39
ComE	22.66	12.78	32.68	19.12	36.59	25.11	41.67	26.88
SDNE	23.74	11.69	34.76	19.87	37.83	23.29	41.14	26.13
CLARE	35.94	14.15	36.88	17.91	46.43	24.46	46.51	24.74
AC2CD	**28.07**	**14.67**	**37.81**	**22.10**	**43.08**	**29.50**	**48.77**	**33.26**

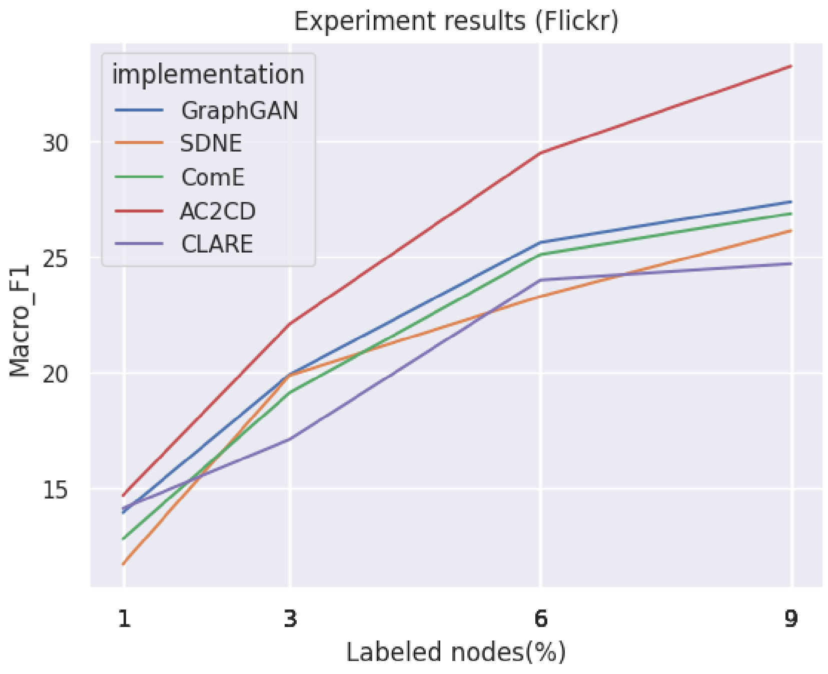


Figure 5.18: Macro-F1 assessment for Flickr dataset.

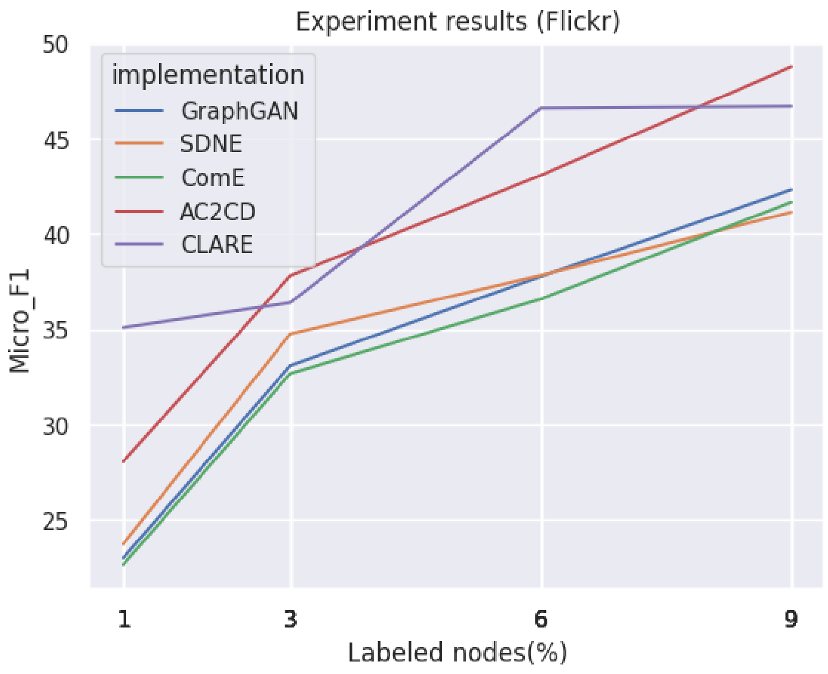


Figure 5.19: Micro-F1 assessment for Flickr dataset.

The results of the NMI score generate a box plot graph as in Figures 5.20 and 5.21. The box plot describes the mean, median, and standard deviation for the executions of each ML approach. Note that the GNN-based strategies (i.e., GraphGAN, AC2CD, CLARE) presented on average superior performance than others. The GraphGAN presents a symmetric profile related to the medium. The SDNE shows a high median where most results are near the maximum NMI. The ComE with a single deep learning approach presents a low median where most results are near the minimum NMI value. On average, the AC2CD offers the best NMI result. CLARE's standard deviation is minimal compared to the other approaches with a stable profile.

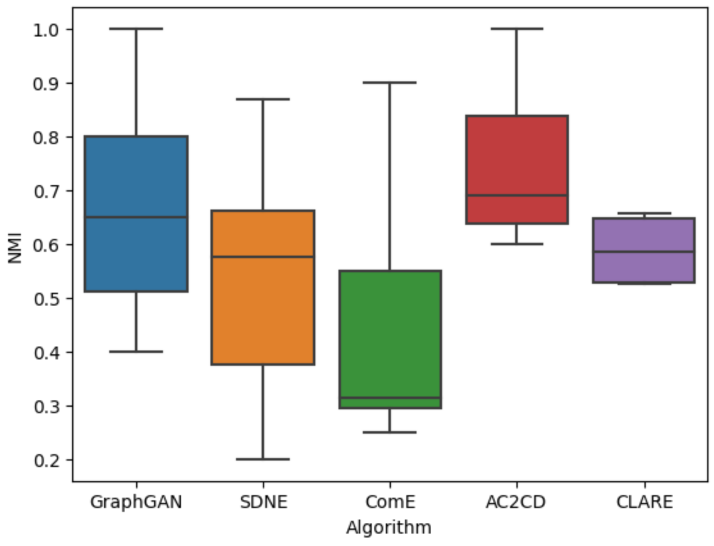


Figure 5.20: Comparative box plot of NMI values for the Email-EU-Core dataset.

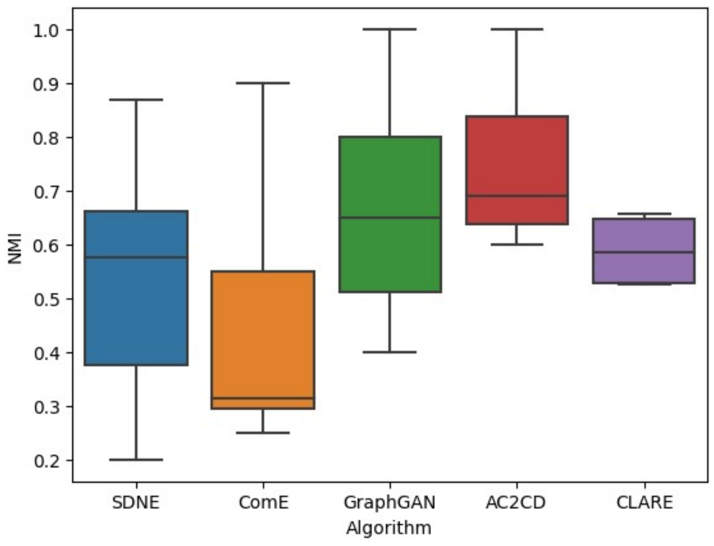


Figure 5.21: Comparative box plot of NMI values for the BlogCatalog2.

The results achieved by CLARE motivated us to undertake a new case study to investigate the GNN implementations varying the dataset, focusing on the implementation performance and the corresponding community profile in each network.

5.3 Case Study 2

The research methodology adopted in this study case mixes exploratory, experimental, and quantitative analysis. Case Study 2 focuses on the heterogeneity of the community size of three datasets and its impact on the performance of the CD methods. The datasets used in this experiment are BlogCatalog3, Email-EU-Core, and High School, as described in Section 5.1. The motivation to conduct Case Study 2 is the work of [38], where the CD framework implementation takes two different GNN including GAT, and GCN. For comparison, we used the GCN implementation in both the Actor and Critic components.

The experiment objective is to compare two implementations of GNN available in the AC2CD with GAT and GCN to evaluate their accuracy considering the heterogeneity of the datasets. The NMI score evaluates the performance of the implementations. The selection of which implementation to run is in the hyperparameter *nn_ type*. Section 5.3.1 presents the results achieved with a discussion.

5.3.1 Results and Discussion

The intuition tells us that the GAT implementation can be more flexible to cope with more complex network structures for using the attention mechanism as a substitute for the statically normalized convolution operation of the GCN.

Table 5.5 presents the results of the experiment using the NMI score. For each dataset, AC2CD was configured to use GAT and GCN. Note the GAT implementation presents a slightly better performance with the three datasets. We consider this aspect a consequence of using the learnable attention mechanism. We also highlight the better performance using both implementations in datasets with fewer classes (i.e., High School). Figure 5.22 presents the NMI comparison of the three datasets highlighting the slightly superior performance of GAT implementation.

Table 5.5: The AC2CD results using NMI score.

Dataset	**GAT**	**GCN**
BlogCatalog3	**0.75**	0.70
Email-EU-Core	**0.72**	0.68
High School	**0.80**	0.72

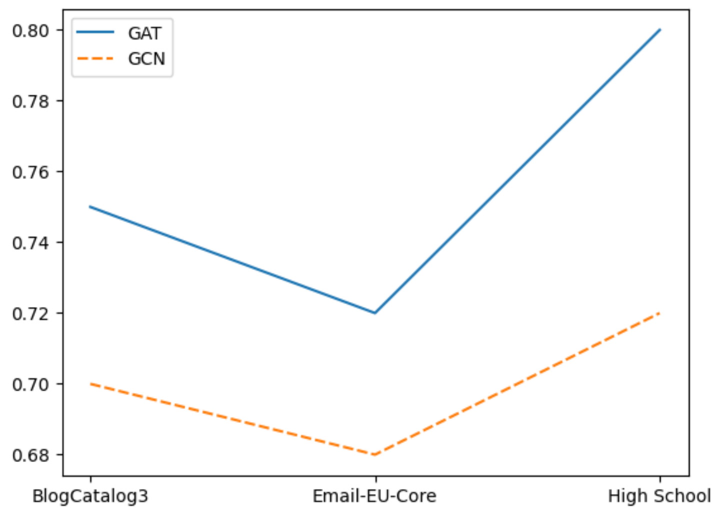


Figure 5.22: The AC2CD results with GAT and GCN using NMI score.

Table 5.6, reproduced from [38], presents the comparison of CLARE using three different datasets (i.e., Amazon, DBLP, Livejournal). The objective is to investigate the accuracy of different GNN architectures in the same task of CD. Note a slight difference among the implementations using GCN, Graph Isomorphism Network (GIN), and GAT. These results confirm our finding that using GAT or GCN presents almost no difference in accuracy.

Table 5.6: Comparison with different graph neural network encoders. Locator results are reported by [38].

	Amazon			**DBLP**			**Livejournal**		
	F1	Jaccard	ONMI	F1	Jaccard	ONMI	F1	Jaccard	ONMI
GCN	0.7438	0.473	0.686	0.3819	0.3116	0.2585	0.4899	0.393	0.3592
GIN	0.7169	0.6196	0.6313	0.3841	0.3100	0.2561	0.4943	0.4004	0.3660
GAT	0.7231	0.6235	0.6318	0.3751	0.3021	0.2446	0.4745	0.3806	0.3405

Moreover, a fake news detection study of [106] using three types of GNN architectures (GAT, GCN, and GraphSAGE) concludes the performance of GNN (i.e., GCN or GAT) produced an entirely insignificant difference. In their implementation, they chose GraphSAGE to be less time-consuming in the text classification task of real or fake.

Thus, GNN exhibits robust results independent of the specific architecture chosen. However, the AC2CD actor-critic approach provides better outputs associated with GNN models, as presented in the conducted experiments with three different datasets. In Chapter 6, we delineate the conclusions of this work and give some hints for future work.

Chapter 6

Conclusion

This research presents the design and implementation of an adaptive model to CD in dynamic social networks based in RL. We proceed with two rounds of study cases comparing our model with classical and state-of-the-art solutions to CD. In both studies, our model achieved relevant results.

Although the research of CD solutions may seem mature, in recent years, we perceive a growing volume of publications seeking to improve the research on CD solutions performance towards the usage of high volume datasets [3] or using ML to enhance the quality of scoring response [22–24]. However, as the authors in [25] indicate, the aggregating topological and content information can enable a more informative CD, in which cues from different sources integrate into more powerful models. This research brought a candle to help clarify CD in the context of Dynamic Social Networks. Furthermore, we demonstrated the effectiveness of using GNN and RL as an alternative to solve the CD problem in OSN.

The two rounds of experiments presented the relevant results of ML-based solutions to CD, mainly the stability of the results when facing the complexity of datasets like Flickr or Youtube. In Case Study 1, we demonstrated the efficacy of our proposed model compared to classical and state-of-the-art solutions. In Case Study 2, we focused on the robustness and flexibility of our model applied to datasets with different features and using two types of GNN, concluding that there is an irrelevant difference in accuracy in the considered GNN architectures (GAT and GCN) and promoting the Actor-Critic architecture with PPO as highlighted aspects. These studies validated our hypothesis that the application of the DRL approach to continually improve the modularity density of a community structure dealing with dynamic social networks considering the state of the art of classical and ML-based solutions. In addition, DRL improve the accuracy of CD in dynamic social networks considering the state of the art of classical and ML-based solutions.

Some questions arise during the conduction of this research in a divergent way. These questions are left as future work since they have the potential to originate another thesis:

- The use of explainable AI to better understand each component's influence on the results. We aim to investigate which part contributes more and how much to the model's results. We may improve the model's accuracy by experimenting with different components. In addition, the resolution limit presented by [32] might be explored more profoundly in the semi-supervised case focusing on unbalanced community sizes to test the DRL method.

- Improve the visualization of the detected communities and verify the accuracy inside each community. This research path sheds more light on the results achieved by the model, giving insights into the model's performance by comparing the results achieved with ground truth.

- Explore using distributed training to reduce the time necessary to train the model. In this same direction, another approach may be to explore recurrent CNN layers. In the data manipulation module, we might try another embedding algorithm.

- Testing the architecture with other temporal datasets and running datasets with more snapshots to verify whether there is an improvement in the accuracy score. Such future work is vital to expose the proposed model to diverse network topologies.

- Compare with probabilistic-based methods such as MRF. The authors in [107] present a technique that formalizes modularity as an energy function based on the structures of MRF associated with the belief propagation method to find communities in a network.

- The context of this research was in OSN. However, we could apply the AC2CD to other contexts like image segmentation, where objects in an image are communities.

References

[1] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004. 1, 12, 13, 15

[2] Stanley Wasserman and Kathrine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994. 1, 3

[3] Rahil Sharma and Suely Oliveira. Community detection algorithm for big social networks using hybrid architecture. *Big data research*, 10:44–52, 2017. 1, 3, 67

[4] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 990–998. ACM, 2008. 1

[5] Gil Amitai, Arye Shemesh, Einat Sitbon, Maxim Shklar, Dvir Netanely, Ilya Venger, and Shmuel Pietrokovski. Network analysis of protein structures identifies functional residues. *Journal of molecular biology*, 344(4):1135–1146, 2004. 1

[6] Lovro Šubelj, Štefan Furlan, and Marko Bajec. An expert system for detecting automobile insurance fraud using social network analysis. *Expert Systems with Applications*, 38(1):1039–1052, 2011. 1

[7] Edward J. S. Hearnshaw and Mark M. J. Wilson. A complex network approach to supply chain network theory. *International Journal of Operations & Production Management*, 2013. 1, 4

[8] Annapurna Jonnalagadda and Lakshmanan Kuppusamy. A survey on game theoretic models for community detection in social networks. *Social Network Analysis and Mining*, 6(1):83, 2016. 1, 33, 38

[9] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4):175–308, 2006. 1

[10] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010. 1, 11

[11] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007. 1

[12] Mark EJ Newman and Michelle Girvan. Mixing patterns and community structure in networks. In *Statistical mechanics of complex networks*, pages 66–87. Springer, 2003. 2

[13] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002. 2

[14] Richard M Karp. *Reducibility among combinatorial problems.* Springer, 2010. 2

[15] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the national academy of sciences*, 101(9):2658–2663, 2004. 2

[16] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015. 2

[17] Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, Jian Yang, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, et al. A comprehensive survey on community detection with deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 3

[18] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Dongxiao He, Jia Wu, Philip Yu, and Weixiong Zhang. A survey of community detection approaches: From statistical modeling to deep learning. *IEEE transactions on knowledge and data engineering*, pages 1–22, 2021. 3, 17

[19] Jianyong Sun, Wei Zheng, Qingfu Zhang, and Zongben Xu. Graph neural network encoding for community detection in attribute networks. *arXiv preprint arXiv:2006.03996*, 2020. 3

[20] Hocine Cherifi, Gergely Palla, Boleslaw K Szymanski, and Xiaoyan Lu. On community structure in complex networks: challenges and opportunities. *Applied Network Science*, 4(1):1–35, 2019. 3

[21] Martin Rosvall, Jean-Charles Delvenne, Michael T Schaub, and Renaud Lambiotte. Different approaches to community detection. *Advances in network clustering and blockmodeling*, pages 105–119, 2019. 3, 11, 13

[22] Yao Zhang, Yun Xiong, Yun Ye, Tengfei Liu, Weiqiang Wang, Yangyong Zhu, and Philip S Yu. Seal: Learning heuristics for community detection with generative adversarial networks. In *Proc. of the 26th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, pages 1103–1113, 2020. 3, 33, 35, 67

[23] Ali Mohammad Saghiri, M. Daliri Khomami, and Mohammad Reza Meybodi. *Random Walk Algorithms: Definitions, Weaknesses, and Learning Automata-Based Approach*, pages 1–7. Springer International Publishing, Cham, 2019. 3, 33, 37, 67

[24] Eduardo C. Paim, Ana L. C. Bazzan, and Camelia Chira. Detecting communities in networks: a decentralized approach based on multiagent reinforcement learning. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2225–2232, 2020. 3, 33, 36, 67

[25] Ana P. Appel, Renato L. F. Cunha, Charu Aggarwal, and Marcela Megumi Terakado. Temporally evolving community detection and prediction in content-centric networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Dublin, Ireland, September 2018. Springer. 3, 67

[26] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Int. symposium on computer and information sciences*, pages 284–293. Springer, 2005. 3, 14

[27] John Scott. Social network analysis. *Sociology*, 22(1):109–127, 1988. 3

[28] Fang Zhang, Anjun Ma, Zhao Wang, Qin Ma, Bingqiang Liu, Lan Huang, and Yan Wang. A central edge selection based overlapping community detection algorithm for the detection of overlapping structures in protein–protein interaction networks. *Molecules*, 23(10):2633, 2018. 4

[29] Márcia Oliveira and Joao Gama. An overview of social network analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):99–115, 2012. 4

[30] Gerardus Blokdyk. *Dynamic Network Analysis A Complete Guide.* 5STARCooks, 2020. 5

[31] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016. 6, 17, 42

[32] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the national academy of sciences*, 104(1):36–41, 2007. 6, 14, 68

[33] Aurélio Ribeiro Costa and Célia Ghedini Ralha. AC2CD: An actor-critic architecture for community detection in dynamic social networks. *Knowledge-Based Systems*, 261:110202, February 2023. 6

[34] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2508–2515. AAAI Press, 2018. 6, 56, 57

[35] Sandro Cavallari, Vincent W. Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 377–386, New York, NY, USA, 2017. Association for Computing Machinery. 6, 55

[36] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'16, page 1225–1234, New York, NY, USA, 2016. Association for Computing Machinery. 6, 54, 55

[37] Aniello De Santo, Antonio Galli, Vincenzo Moscato, and Giancarlo Sperlì. A deep learning approach for semi-supervised community detection in online social networks. *Knowledge-Based Systems*, 229:107345, 2021. 6, 19, 33, 34, 48, 52, 53, 54

[38] Xixi Wu, Yun Xiong, Yao Zhang, Yizhu Jiao, Caihua Shan, Yiheng Sun, Yangyong Zhu, and Philip S Yu. Clare: A semi-supervised community detection algorithm. In *Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'22*, pages 2059–2069. ACM, 2022. 7, 33, 34, 48, 53, 54, 56, 57, 65, 66

[39] Richard P. Smiraglia. 4 - empirical techniques for visualizing domains. In Richard P. Smiraglia, editor, *Domain Analysis for Knowledge Organization*, pages 51–89. Chandos Publishing, 2015. 9

[40] A.M. Chiesi. Network analysis. In Neil J. Smelser and Paul B. Baltes, editors, *International Encyclopedia of the Social & Behavioral Sciences*, pages 10499–10502. Pergamon, Oxford, 2001. 9

[41] Gilbert Strang. *Linear algebra and its applications.* Belmont, CA: Thomson, Brooks/Cole, 2006. 10

[42] Daniel R. Figueiredo. Introdução a redes complexas. *Atualizações em Informática*, pages 303–358, 2011. 10

[43] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1975–1985, 2021. 10

[44] Ferran Parés, Dario Garcia Gasulla, Armand Vilalta, Jonatan Moreno, Eduard Ayguadé, Jesús Labarta, Ulises Cortés, and Toyotaro Suzumura. Fluid communities: A competitive, scalable and diverse community detection algorithm. In *Int. Conf. on Complex Networks and their Applications*, pages 229–240. Springer, 2017. 11, 15

[45] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983. 11, 17

[46] Mark E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006. 11

[47] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015. 11, 16

[48] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS)*, page 585–591, Cambridge, MA, USA, 2001. MIT Press. 11, 16

[49] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 1105–1114, 2016. 11, 16

[50] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000. 11, 16

[51] Mason A Porter, Jukka-Pekka Onnela, and Peter J Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009. 12

[52] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6):066122, 2011. 12

[53] Mark E. J. Newman. Modularity and community structure in networks. *Proc Natl Acad Sci USA*, 103(23):8577–8582, 2006. 12, 13

[54] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019. 13

[55] Alex Arenas, Alberto Fernandez, and Sergio Gomez. Analysis of the structure of complex networks at different resolution levels. *New journal of physics*, 10(5):053039, 2008. 14

[56] Zhenping Li, Shihua Zhang, Rui-Sheng Wang, Xiang-Sun Zhang, and Luonan Chen. Quantitative function for community detection. *Physical Review E*, 77(3):036109, 2008. 14

[57] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006. 15

[58] ZhengYou Xia and Zhan Bu. Community detection based on a semantic network. *Knowledge-Based Systems*, 26:30–39, 2012. 15

[59] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833–852, 2018. 16

[60] Yu Chen, Lingfei Wu, and Mohammed J Zaki. Reinforcement learning based graph-to-sequence model for natural question generation. In *International Conference on Learning Representations*, 2019. 16

[61] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018. 16

[62] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017. 17

[63] Clement Lee and Darren J Wilkinson. A review of stochastic block models and extensions for graph clustering. *Applied Network Science*, 4(1):1–50, 2019. 18

[64] John D Kelleher. *Deep learning*. MIT press, 2019. 18

[65] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. 19

[66] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 19

[67] Di Jin, Ziyang Liu, Weihao Li, Dongxiao He, and Weixiong Zhang. Graph convolutional networks meet markov random fields: Semi-supervised community detection in attribute networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI/IAAI/EAAI)*. AAAI Press, 2019. 19, 20

[68] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 20, 21

[69] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016. 21

[70] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017. 21

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of the 31st Int. Conf. on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. 21

[72] Kanae Takahashi, Kouji Yamamoto, Aya Kuchiba, and Tatsuki Koyama. Confidence interval for micro-averaged F_1 and macro-averaged F_1 scores. *Applied intelligence (Dordrecht, Netherlands)*, 52(5):4961–4972, 2021. 21, 53

[73] Juri Opitz and Sebastian Burst. Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*, 2019. 21

[74] L.N.F. Ana and A.K. Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II, 2003. 22

[75] Pan Zhang. Evaluating accuracy of community detection using the relative normalized mutual information. *Journal of Statistical Mechanics: Theory and Experiment*, 2015(11):P11006, 2015. 22, 53

[76] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018. 22, 23, 24, 26

[77] Maxim Lapan. *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more.* Packt Publishing Ltd, 2018. 23, 24

[78] Hongming Zhang and Tianyang Yu. *Taxonomy of Reinforcement Learning Algorithms*, pages 125–133. Springer Singapore, Singapore, 2020. 23, 24, 25

[79] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992. 24

[80] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 25

[81] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004. 28

[82] Vitor Freitas. Parsifal. Available at `https://parsif.al/`. Accessed on: 2023-02-24, 2014. Parsifal is an online tool designed to support researchers to perform systematic literature reviews within the context of Software Engineering. 29

[83] Domenico Mandaglio and Andrea Tagarelli. Dynamic consensus community detection and combinatorial multi-armed bandit. In *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 184–187. IEEE, 2019. 30, 33, 37

[84] Qikai Cheng, Jiamin Wang, Wei Lu, Yong Huang, and Yi Bu. Keyword-citation-keyword network: a new perspective of discipline knowledge structure analysis. *Scientometrics*, 124(3):1923–1943, 2020. 30, 32, 34

[85] Bentian Li and Dechang Pi. Network representation learning: a systematic literature review. *Neural Computing and Applications*, pages 1–33, 2020. 30, 32, 35

[86] Yuyao Wang, Jie Cao, Zhan Bu, Jia Wu, and Youquan Wang. Dual structural consistency preserving community detection on social networks. *IEEE Transactions on Knowledge and Data Engineering*, 2023. 31, 32, 34

[87] Yunyun Niu, Detian Kong, Ligang Liu, Rong Wen, and Jianhua Xiao. Overlapping community detection with adaptive density peaks clustering and iterative partition strategy. *Expert Systems with Applications*, 213:119213, 2023. 31, 32, 34

[88] Anuraj Mohan and KV Pramod. Network representation learning: Models, methods and applications. *SN Applied Sciences*, 1(9):1014, 2019. 32, 37

[89] Elyazid Akachar, Brahim Ouhbi, and Bouchra Frikh. Community detection in social networks using structural and content information. In *Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services*, pages 282–288, 2018. 32, 38

[90] Bonaventure C Molokwu, Shaon Bhatta Shuvo, Narayan C Kar, and Ziad Kobti. Node classification in complex social graphs via knowledge-graph embeddings and convolutional neural network. In *Proc. of the Int. Conf. on Computational Science*, pages 183–198. Springer, 2020. 32, 35

[91] Oleksandr Shchur and Stephan Günnemann. Overlapping community detection with graph neural networks. *arXiv:1909.12201*, 2019. 33, 37

[92] Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. Streaming graph neural networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–728, 2020. 33, 36

[93] Jose N Paredes, Gerardo I Simari, Maria Vanina Martinez, and Marcelo A Falappa. Netder: An architecture for reasoning about malicious behavior. *Information Systems Frontiers*, pages 1–17, 2020. 33, 35

[94] Xubo Gao, Qiusheng Zheng, Didier A Vega-Oliveros, Leandro Anghinoni, and Liang Zhao. temporal network pattern identification by community modelling. *Scientific Reports*, 10(1):1–12, 2020. 33, 36

[95] Zhao Yang, René Algesheimer, and Claudio J Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific reports*, 6:30750, 2016. 33, 38

[96] Jiayi Du, Muyang Jin, Petter N Kolm, Gordon Ritter, Yixuan Wang, and Bofei Zhang. Deep reinforcement learning for option replication and hedging. *The Journal of Financial Data Science*, 2(4):44–57, 2020. 40

[97] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019. 45

[98] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540*, 2016. 45

[99] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011. 46

[100] R. Zafarani and H. Liu. Social computing data repository at ASU, 2009.

[101] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014. 48, 58

[102] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014. 55

[103] A. Bharali. An analysis of email-eu-core network. *International Journal of Scientific Research in Mathematical and Statistical Sciences*, 5:100–104, 8 2018. 58

[104] Shanfeng Wang, Qixiang Wang, and Maoguo Gong. Multi-task learning based network embedding. *Frontiers in Neuroscience*, 13:1387, 2020. 60

[105] Jun Du, Erol Gelenbe, Chunxiao Jiang, Haijun Zhang, Yong Ren, and H Vincent Poor. Peer prediction-based trustworthiness evaluation and trustworthy service rating in social networks. *IEEE Transactions on Information Forensics and Security*, 14(6):1582–1594, 2018. 62

[106] Andrea Stevens Karnyoto, Chengjie Sun, Bingquan Liu, and Xiaolong Wang. Augmentation and heterogeneous graph neural network for aaai2021-covid-19 fake news detection. *International journal of machine learning and cybernetics*, 13(7):2033–2043, 2022. 66

[107] Di Jin, Binbin Zhang, Yue Song, Dongxiao He, Zhiyong Feng, Shizhan Chen, Weihao Li, and Katarzyna Musial. Modmrf: A modularity-based markov random field method for community detection. *Neurocomputing*, 405:218–228, 2020. 68

Appendix A

AC2CD configurations

This appendix shows the necessary configurations to run the AC2CD. Section A.1 presents the list of *Hp* adopted to run the study cases. Section A.2 presents the required libraries and how to install them to be able to run the AC2CD.

A.1 Hyperparameters

The list of *Hp* used in the case studies.

```
emb_dim=256
emb_walk_len=20
emb_walks_per_node=10
alpha=3e-04
batch_size=10
checkpoint_interval=5000
device="cuda"
learn_rate=40
max_epochs=20000
nn_type="gat"
n_epochs=20
n_games=100
patience_threshold=100
snapshots=1
timespan=100
validation_interval=50
```

A.2 Dependencies

The required libraries to execute AC2CD. It can be saved as requirements.txt, and before running pip install -r requirements.txt to install all libraries.

```
networkx==2.8.3
matplotlib==3.5.2
gym==0.21.0
--extra-index-url https://download.pytorch.org/whl/cu113
torch==1.11.0
torchvision==0.12.0+cu113
torchaudio==0.11.0+cu113
-f https://data.pyg.org/whl/torch-1.11.0+cu113.html
torch-scatter==2.0.9
torch-sparse==0.6.14
torch-cluster==1.6.0
torch-spline-conv ==1.2.1
torch-geometric==2.0.4
numpy==1.23.1
seaborn==0.11
pydantic[dotenv]==1.10
cdlib[extras]
```