SAPIENZA
UNIVERSITÀ DI ROMA

Community Deception in online Social Networks

Facoltà: Ingegneria dell'informazione, informatica e statistica
Master Degree in Computer Science

Andrea Bernini
ID number 2021867

Advisor
Prof. Tolomei Gabriele

Co-Advisor
Prof. Silvestri Fabrizio

President of the study course
Prof. Casalicchio Emiliano

Academic Year 2022/2023

Community Deception in online Social Networks
Master thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Website: https://github.com/AndreaBe99/community-deception-thesis
Author's email: bernini.2021867@studenti.uniroma1.it

Abstract

Community detection techniques are useful tools for social media platforms to discover tightly connected groups of users who share common interests. However, this functionality often comes at the expense of potentially exposing individuals to privacy breaches by inadvertently revealing their sensitive data. Therefore, some users may wish to safeguard their anonymity and choose to opt out of community detection for various reasons, such as affiliation with political or religious organizations.

In this study, we address the challenge of *community membership hiding*, which involves strategically altering the structural properties of a network graph to prevent one or more nodes from being identified by a given community detection algorithm. We tackle this problem by formulating it as a constrained counterfactual graph objective, and we solve it via deep reinforcement learning. We validate the effectiveness of our method through two distinct tasks: node and community deception. Extensive experiments show that our approach overall outperforms existing baselines in both tasks.

Contents

1 Introduction **1**

2 Related Work **3**
2.1 Community Detection 3
2.2 Community Deception 4

3 Background and Preliminaries **5**
3.1 Community Detection 5
3.2 Graph Neural Networks 6
3.3 Deep Reinforcement Learning 8

4 Community Membership Hiding **10**
4.1 Problem Formulation 10
4.2 Counterfactual Graph Objective 12
4.3 Markov Decision Process 13
4.3.1 States ($\mathcal{S}$) 13
4.3.2 Actions ($\mathcal{A}$) 13
4.3.3 Transitions Probability ($\mathcal{P}$) 13
4.3.4 Reward (r) 13
4.3.5 Policy ($\pi_{\boldsymbol{\theta}}$) 14

5 Proposed Method **15**
5.1 Advantage Actor-Critic Agent 16
5.1.1 Policy function (Actor) 17
5.1.2 Value function (Critic) 17
5.2 Environment 19
5.3 Performance Evaluation 19

6 Experiments **21**
6.1 Experimental Setup 21
6.1.1 Dataset 21
6.1.2 Community Detection Algorithms 21
6.1.3 Distance Metrics 22
6.1.4 Tasks 23
6.2 Node Deception Task 23
6.2.1 Beselines 23
6.2.2 Evaluation Metrics 23

6.3 Community Deception Task 24
6.3.1 Baselines 24
6.3.2 Evaluation Metrics 24
6.4 Results and Discussion 25
6.4.1 Node Deception Task 25
6.4.2 Community Deception Task 36
6.5 Parameter Sensitivity 47

7 Ethical Implications **48**
7.1 Possible Benefits 48
7.2 Potential Drawbacks 48
7.3 Responsible Development and Use 49

8 Conclusion and Future Work **50**
8.1 Recap of Achievements 50
8.2 Future Directions 50

Bibliography **52**

Chapter 1

Introduction

In the ever-expanding landscape of interconnected entities, such as social networks, online platforms, and collaborative ecosystems, the identification of coherent *communities* plays a pivotal role in understanding complex network graph structures [11]. The task of identifying these communities is typically accomplished by running *community detection* algorithms on the network graphs, which allows the unveiling of hidden patterns and relationships among nodes within these structures. These algorithms effectively group entities, users, or objects based on shared characteristics or interactions, shedding light on the underlying organization and dynamics of the network.

The successful detection of communities in complex network graphs is useful in several application domains [18]. For instance, the insights gained from accurately identifying communities can significantly impact business strategies, leading to better monetization opportunities through targeted advertising. By understanding the distinct interests and behaviors of community members, businesses can provide better service solutions and tailor advertisements that are more relevant and appealing to their customer segments, resulting in higher engagement and increased revenue [27].

However, while community detection algorithms offer immense benefits, they also raise concerns regarding individual privacy and data protection. In some cases, certain nodes within the network might prefer to remain undetected as part of any specific community. These nodes could belong to sensitive or private groups (e.g., political organizations or religious associations) or may simply wish to retain their anonymity for personal reasons. Anyway, the right to opt out of community detection becomes crucial to strike a balance between preserving privacy and maximizing the utility of these algorithms.

Motivated by the need above, in this work, we address the intriguing challenge of *community membership hiding*. Taking inspiration from the concept of *counterfactual reasoning* [39, 38], specifically for graph data [23], our objective is to offer users personalized recommendations to preserve their anonymity from community detection. We effectively guide them in adjusting their connections so that they cannot be identified as members of a given community. For instance, we might suggest to a hypothetical user of a social network, "*If you unfollow users X and Y, you will no*

longer be recognized as a member of community Z."

The core challenge of this problem lies in determining how to strategically modify the structural properties of a network graph, effectively excluding one or more nodes from being identified by a given community detection algorithm. To the best of our knowledge, we are the first to formulate the community membership hiding problem as a constrained counterfactual graph objective. Furthermore, inspired by [7], we cast this problem into a Markov decision process (MDP) framework and we propose a deep reinforcement learning (DRL) method to solve it.

Our method works as follows. We start with a graph and a set of communities identified by a specific community detection algorithm whose inner logic remains unknown or undisclosed. When given a target node within a community, our objective is to find the optimal structural adjustment of the target node's neighborhood. This adjustment should enable the target node to remain concealed when the same community detection algorithm is reapplied to the modified graph. We refer to this primary task as *node deception*, and we consider it successful when a predefined similarity threshold between the original community and the new community containing the target node is met. In addition to this, we define a *community deception* task – already investigated in the literature [10] – whose goal is to mask *all* the nodes of a given community.

We validate our approach on four real-world datasets, and we demonstrate that it outperforms existing baselines using standard quality metrics both on node and community deception tasks. Furthermore, unlike their competitors, our method maintains its effectiveness even when used in conjunction with a community detection algorithm that was not seen during the training phase. The main contributions of our work are summarized below.

- We formulate the community membership hiding problem as a constrained counterfactual graph objective.
- We cast this problem within an MDP framework and solved it via DRL.
- We utilize a graph neural network (GNN) representation to capture the structural complexity of the input graph, which in turn is used by the DRL agent to make its decisions.
- We validate the performance of our method in comparison with existing baselines using standard quality metrics both on node and community deception tasks.
- We publicly release both the source code and the data utilized in this study to encourage reproducibility.[1]

The remainder of this work is structured as follows. In Section 2, we review related work. Section 3 contains background and preliminaries. We present our problem formulation in Section 4. In Section 5, we describe our method, which we validate through extensive experiments in Section 6. Finally, we conclude in Section 8.

[1] `https://github.com/AndreaBe99/community-deception-thesis`

Chapter 2

Related Work

This chapter embarks on a dual exploration of interconnected domains—Community Detection and Community Deception—each fundamental to our research.

2.1 Community Detection

Community detection algorithms serve as indispensable tools in the realm of network analysis, playing a pivotal role in the exploration of densely interconnected groups of nodes within intricate graphs. These algorithms find their applications across a wide spectrum of domains, ranging from the intricate webs of social networks to the intricate tapestry of biological systems and the intricate dynamics of economic networks. Broadly classified, community detection algorithms fall into two overarching categories: non-overlapping and overlapping community detection.

Non-overlapping community detection methods are designed to assign each node within a network to a singular community. This task is achieved through a plethora of sophisticated techniques, each tailored to the unique characteristics of the data. These approaches may involve leveraging concepts like Modularity Optimization [2], Spectrum Analysis [36], Random Walk [32], or the intriguing Label Propagation [33].

On the other hand, overlapping community detection ventures into the realm of complex networks where nodes are not confined to exclusive memberships but rather exhibit the flexibility to participate in multiple communities concurrently. In addressing this intricate challenge, advanced methods have emerged, exemplified by the likes of NISE (Neighborhood-Inflated Seed Expansion) [42]. Additionally, there are innovative techniques centered around the minimization of the Hamiltonian associated with the Potts model [35]. These approaches capture the nuanced relationships that exist in real-world networks.

For those seeking a more comprehensive understanding of the fascinating landscape of community detection algorithms, we highly recommend consulting the insightful work authored by Jin et al. [16]. This seminal resource provides a thorough and up-to-date overview of the field, shedding light on its evolution and showcasing its vital role in modern network analysis.

2.2 Community Deception

The realm of community deception is intimately intertwined with the intricate problem of concealing community memberships, a central focus of our research in this endeavor. In essence, community deception can be viewed as a specialized facet of the broader community membership hiding challenge, where the primary objective is to obscure an entire community from the prying eyes of community detection algorithms.

Community deception is a multifaceted concept, offering a range of potential applications and consequences. One of its positive applications lies in safeguarding the anonymity of individuals in online monitoring scenarios, particularly evident in the context of social networks. By employing community deception techniques, users can protect their privacy and prevent the unauthorized disclosure of their affiliations. Furthermore, community deception can contribute to the field of public safety by enabling the identification and mitigation of online criminal activities, thereby enhancing the security of digital spaces.

Nevertheless, it is crucial to acknowledge the darker side of these deception techniques. Malicious actors can exploit them with nefarious intentions, evading detection algorithms and clandestinely conducting activities that may infringe upon the law, thereby posing significant challenges for law enforcement agencies and cybersecurity experts.

Various techniques have been devised for the purpose of concealing communities within networks, with each approach tailored to address specific nuances and requirements. Noteworthy examples within the Modularity-based category include the pioneering algorithm proposed by Nagaraja [28], which marked a significant milestone in the field. The DICE algorithm, introduced by Waniek et al. [40], further advances the Modularity-based deception methodology. Additionally, another innovative approach within this realm has been presented by Fionda and Pirrò [10], underscoring the rich diversity of techniques within this category.

In addition, some deception techniques revolve around the concept of Safeness, as initially defined by Fionda and Pirrò [10] and subsequently explored in greater depth by Chen et al. [6]. This approach offers an alternative perspective on community deception, focusing on the security and resilience aspects of network concealment. Additionally, the concept of Permanence, as articulated by Chakraborty et al. [5], offers yet another intriguing dimension to the field, aiming to ensure the persistence and durability of hidden communities.

For a comprehensive and in-depth exploration of these diverse community deception techniques and their applications, we strongly recommend referring to the comprehensive survey available in [17]. This invaluable resource offers a thorough overview of the field, presenting a panorama of methodologies, insights, and challenges that continue to shape the landscape of community deception.

Chapter 3

Background and Preliminaries

This chapter provides a comprehensive backdrop for our study, introducing three crucial domains of knowledge that underpin our research: Community Detection, Graph Neural Networks (GNNs), and Deep Reinforcement Learning (DRL). Each section illuminates key concepts and principles vital to our investigation.

3.1 Community Detection

In this section, we will briefly review the definition of the well-known community detection problem. Afterward, we will utilize this definition as the foundation to provide our formulation of the community membership hiding problem.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an arbitrary (directed) graph, where $\mathcal{V}$ is a set of n nodes ($|\mathcal{V}| = n$), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of m edges ($|\mathcal{E}| = m$). Optionally, an additional set of p node attributes may also be present. In such cases, each node $u \in \mathcal{V}$ is associated with a corresponding p-dimensional real-valued feature vector $x_u \in \mathbb{R}^p$. These node features collectively form a $n \times p$ real matrix $X \in \mathbb{R}^{n \times p}$, where x_u constitutes the u-th row of the matrix. Furthermore, the underlying link structure of $\mathcal{G}$ is represented using a binary adjacency matrix $A \in \{0,1\}^{n \times n}$, where $A_{u,v} = 1$ if and only if the edge $(u, v) \in \mathcal{E}$, and it is 0 otherwise.

The *community detection* problem aims to identify clusters of nodes within a graph, called *communities*. Due to the intricate nature of the concept and its reliance on contextual factors, establishing a universally accepted definition for a "community" is challenging. Intuitively, communities exhibit strong intra-cluster connections and relatively weaker inter-cluster connections [45].

More formally, in this work, we adhere to the definition used by Fionda and Pirrò [10], and we consider a function $f(\cdot)$ that takes a graph as input and generates a partition of its nodes $\mathcal{V}$ into a set of non-empty, non-overlapping, communities $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ as output, i.e., $f(\mathcal{G}) = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$, where k is usually unknown. Within this framework, every node $u \in \mathcal{V}$ is assigned to *exactly one* community. This assignment can be captured using a k-dimensional stochastic vector c_u, where $c_{u,i} = P(u \in \mathcal{C}_i)$ measures the probability that node u belongs to community $\mathcal{C}_i$, and $\sum_{i=1}^{k} c_{u,i} = 1$. Eventually, we use the notation $i_u^* = \arg\max_i(c_{u,i})$ to represent the index of the community to which a specific node u belongs to based on the

outcome of $f(\mathcal{G})$. Note that in the case of hard node partitioning, there exists only one non-zero entry in the vector c_u, which evaluates to 1. However, this framework can be extended to scenarios with overlapping communities, where each node can belong to multiple clusters, resulting in more than one non-zero entry in c_u. We leave the exploration of overlapping communities for future work.

Typically, community detection methods operate by maximizing a specific score that measures the intra-community cohesiveness (e.g., Modularity [30]). However, this usually translates into solving NP-hard optimization problems. Hence, some convenient approximations have been proposed in the literature to realize $f(\cdot)$ in practice, e.g., Louvain [1], WalkTrap [32], Greedy [3], InfoMap [4], Label Propagation [33], Leading Eigenvectors [29], Edge-Betweeness [13], SpinGlass [34]. Anyway, the rationale behind how communities are found is irrelevant to our task, and, hereinafter, we will treat the community detection technique $f(\cdot)$ as a "black box".

3.2 Graph Neural Networks

Graph Neural Networks (GNNs) have emerged as a powerful tool in the domain of deep learning, expanding its capabilities beyond the realm of Euclidean data. These neural networks have become essential for handling inherently non-Euclidean data types, including social networks, recommendation systems, molecular structures, and more. In this section, we embark on an exploration of GNNs, delving into their core concepts, revealing their mathematical foundations, exploring their versatile applications, and illustrating their effectiveness with real-world examples.

GNNs are a specialized type of neural network architecture designed for processing graph-structured data. Their primary goal is to learn representations that capture both node features and the intricate web of interactions within the graph. GNNs excel at preserving graph symmetries and prioritize permutation invariance as a core principle. Essentially, GNNs encapsulate a transformational function that reshapes a graph representation, making it invariant to node shuffling.

One of the most renowned and widely used GNN architectures is the Graph Convolutional Network (GCN) [19]. Conceptually, GCN can be thought of as a natural extension of convolutional neural networks (CNNs), tailored to address the unique challenges presented by graph-structured data. The mathematical formulation of a GCN layer can be expressed as:

$$H = \sigma(\bar{D}^{-\frac{1}{2}}\tilde{A}\bar{D}^{-\frac{1}{2}}X\Theta) \tag{3.1}$$

Here, $\tilde{A}$ represents the adjacency matrix augmented with the identity matrix I, $\bar{D}$ signifies the diagonal node degree matrix derived from $\tilde{A}$, $\sigma(\cdot)$ denotes a non-linear activation function (e.g., ReLU), and Θ represents a matrix containing learnable parameters. The elegance of graph convolutions lies in their innate ability to define a propagation rule that remains invariant to permutations. They gracefully update node-level features, orchestrating an exquisite information aggregation across

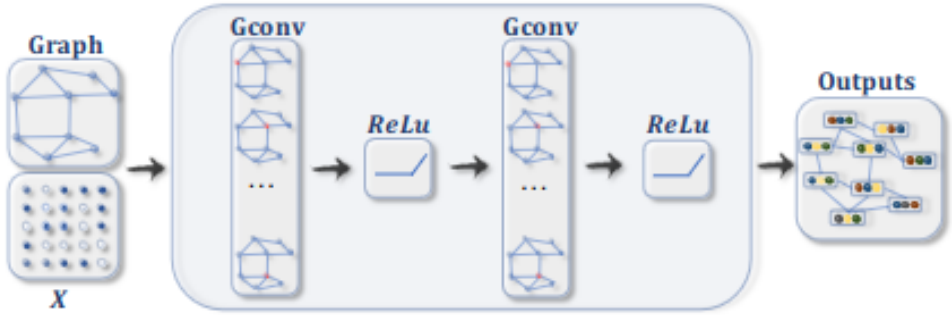


Figure 3.1. Our approach draws inspiration from the comprehensive research on Graph Neural Networks (GNNs) by Wu et al. [44]. It incorporates a Convolutional Graph Neural Network (ConvGNN) characterized by its use of multiple graph convolutional layers. This sophisticated design harnesses the power of GNNs to process and extract meaningful information from graph-structured data, making it well-suited for a wide range of applications.

neighboring nodes, all through the lens of a shared local filter spanning the entire graph.

The realm of Graph Neural Networks (GNNs) presents a compelling tapestry of advantages. GNNs possess a remarkable ability to master the intricacies of graph-structured data, adeptly navigating the complex interconnections that permeate diverse domains. Their inherent capacity to uncover hidden relationships within these intricate data webs underscores their prowess.

Furthermore, GNNs exhibit an exceptional talent for feature learning, akin to a sorcerer conjuring profound embeddings for nodes. Within these embeddings, GNNs encapsulate both the intrinsic attributes of nodes and their intricate connections within the network, a testament to their ability to capture the essence of the data.

In the realm of generalization, GNNs display an elegant grace. They transcend the boundaries of the known, extending their benevolent influence to uncharted territory, seamlessly adapting to unseen graphs and nodes. This adaptability unfurls their utility across a myriad of real-world scenarios, where their versatility shines through.

In the arena of prediction, GNNs dazzle with their might. They ascend to the throne in a regal display of skill, reigning supreme across a diverse spectrum of domains. GNNs excel in tasks such as node classification, link prediction, recommendation systems, and the intricate art of community detection.

In summary, Graph Neural Networks (GNNs) establish a principled foundation for extracting knowledge from the labyrinthine expanse of graph-structured data. Their mathematical elegance, harmoniously intertwined with their ability to unravel intricate relationships, positions them as invaluable assets in the evolving field of modern machine learning and data analysis. GNNs shine most brightly in applications where the quest for uncovering and harnessing the hidden riches embedded within network structures is paramount.

3.3 Deep Reinforcement Learning

Reinforcement Learning (RL) constitutes a pivotal subfield within the domain of machine learning, with its primary focus residing in the empowerment of autonomous agents to execute sequential decisions within an environment, thereby maximizing an accumulated reward signal. Its profound impact extends across an extensive spectrum of applications, encompassing fields as diverse as robotics, strategic game-playing, and autonomous navigation. At the core of RL lies the fundamental notion of agent-environment interaction, a dynamic interplay where an agent undertakes actions within an environment, reaps feedback in the form of rewards, and, over time, refines its decision-making processes to optimize its overall performance.

The fundamental framework that structures RL is the Markov Decision Process (MDP), a refined mathematical model specifically designed to capture decision-making problems that exist in uncertain environments. An MDP is essentially a set of five elements, namely $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where:

- $\mathcal{S}$ symbolizes the collection of states that encapsulate the environment's conceivable configurations or scenarios. Each state within this set represents a unique configuration or context in which the decision-making agent may find itself.
- $\mathcal{A}$ conveys the assortment of actions at the agent's disposal. In every state, the agent retains the prerogative to select an action from this predefined set.
- $\mathcal{P}$, the transition probability function, delineates the intricate dynamics that dictate the evolution of the system. It serves as the conduit through which the probability distribution of transitioning from one state to another is articulated, contingent upon the agent's selected action. Formally, $\mathcal{P}(s'|s, a)$ embodies the probability of transitioning to state s' from the current state s, given the action a taken by the agent.
- $\mathcal{R}$ assumes the role of the reward function, a decisive arbiter that bestows immediate rewards upon the agent for specific actions performed within particular states. Precisely, $\mathcal{R}(s, a, s')$ quantifies the reward accrued by the agent as it moves from state s to state s' by virtue of action a.
- γ, denoting the discount factor, resides within the range of 0 to 1, serving as a pivotal compass that guides the agent's predilection towards immediate rewards in contrast to rewards that lie in the distant future. A heightened value of γ signifies an elevated penchant for considering rewards bestowed in the long term.

The agent, traversing discrete time increments, meanders through a succession of states, effecting transitions from one to the next by selecting actions, all in accordance with a predefined policy π. This policy, akin to a strategic blueprint, maps states to actions and stands as the cornerstone of the agent's pursuit: to ascertain an optimal policy that maximizes the expected cumulative reward over its enduring odyssey.

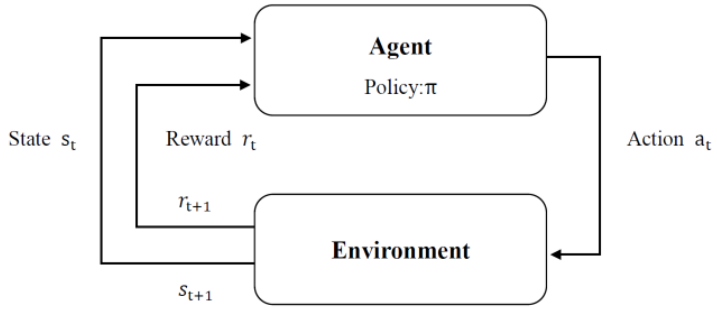


Figure 3.2. Interaction process between agent and environment [31].

Solving an MDP hinges upon the quest for this optimal policy. However, traditional RL methodologies often encounter formidable challenges when grappling with high-dimensional state spaces and labyrinthine environments. Enter the domain of Deep Reinforcement Learning (DRL), a pioneering frontier that merges RL with the formidable capabilities of deep neural networks. DRL algorithms leverage these neural architectures to approximate value functions or policies, thereby gaining the competence to navigate through high-dimensional input domains, replete with raw sensory data—think images or sensor readings. Prominent algorithms within the DRL paradigm include Q-Learning [41], Reinforce [43], Actor-Critic [20], and the illustrious Deep Q-Network (DQN) [25]. Their deployment is geared towards the noble objective of resolving complex decision-making challenges, fortified by the study and analysis of MDPs, an indomitable stronghold that empowers intelligent systems to craft informed choices amid the tempest of uncertainty.

Chapter 4

Community Membership Hiding

In a nutshell, community membership hiding aims to enable a target node within a graph to elude being recognized as a member of a particular node cluster, as determined by a community detection algorithm. This objective is accomplished by granting the node in question the ability to strategically modify its connections with other nodes. Therefore, our primary focus is on making changes to the graph's structure, represented by the adjacency matrix. While altering node features holds potential interest, that aspect is reserved for future exploration. We depict our approach in Fig. 4.1.

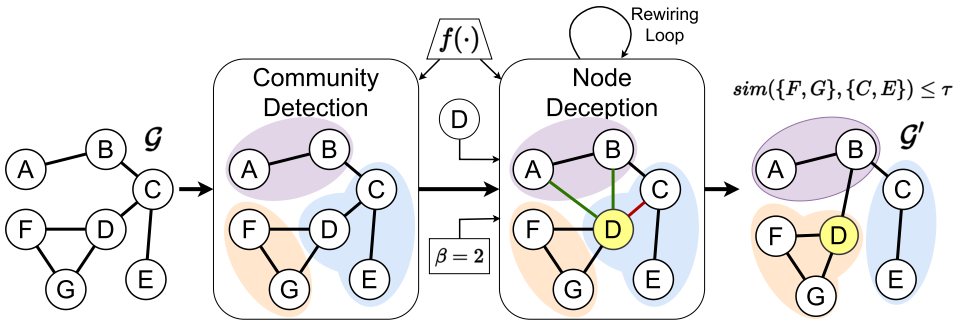


Figure 4.1. Given a graph $\mathcal{G}$, a node u (in this case $u = D$), a budget of actions β, and the set of communities identified by the community detection algorithm $f(\cdot)$ (including the community $\mathcal{C}_i$ to which the node belongs), community membership hiding consists of adding inter-community edges $\mathcal{E}^{+}_{u,i}$ (green edges), or removing intra-community edges $\mathcal{E}^{-}_{u,i}$ (red edge), so that the value returned by the similarity function $sim(\cdot,\cdot)$, between the new community to which the node belongs after rewiring, and the original one, is lower than the τ constraint.

4.1 Problem Formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph and $f(\mathcal{G}) = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ denote the community arrangement derived from applying a detection algorithm $f(\cdot)$ to $\mathcal{G}$. Furthermore, suppose that f has identified node $u \in \mathcal{V}$ as a member of the community $\mathcal{C}_i \in f(\mathcal{G})$ – i.e., $i^*_u = i$ – denoted as $u \in \mathcal{C}_i$. The aim of community membership hiding is to formulate a function $h_{\boldsymbol{\theta}}(\cdot)$, parametrized by $\boldsymbol{\theta}$, that takes as input the initial graph $\mathcal{G}$ and

produces as output a *new* graph $h_{\boldsymbol{\theta}}(\mathcal{G}) = \mathcal{G}' = (\mathcal{V}, \mathcal{E}')$. Among all the possible graphs, we seek the one which, when input to the community detection algorithm f, disassociates a target node u from its original community $\mathcal{C}_i$. To achieve that goal, suppose that the target node u is associated with a new community $\mathcal{C}'_i \in f(\mathcal{G}')$. Hence, we can define the objective of community membership hiding by establishing a threshold for the similarity between $\mathcal{C}'_i$ and $\mathcal{C}_i$, excluding the target node u, which, by definition, belongs to both communities. In other words, we set a condition: $sim(\mathcal{C}_i - \{u\}, \mathcal{C}'_i - \{u\}) \leq \tau$, where $\tau \in [0, 1]$.[1]

Several similarity measures can be used to measure $sim(\cdot, \cdot)$ depending on the application domain, e.g., the overlap coefficient (a.k.a. Szymkiewicz–Simpson coefficient) [24], the Jaccard coefficient [15], and the Sørensen-Dice coefficient [8].

Setting $\tau = 0$ represents the most stringent scenario, where we require zero overlaps between $\mathcal{C}'_i$ and $\mathcal{C}_i$, except for the node u itself. At the other extreme, when $\tau = 1$, we adopt a more tolerant strategy, allowing for maximum overlap between $\mathcal{C}'_i$ and $\mathcal{C}_i$. However, it is important to note that except for the overlap coefficient, which can yield a value of 1 even if one community is a subset of the other, the Jaccard and Sørensen-Dice coefficients yield a value of 1 only when the two communities are identical. In practice, setting $\tau = 1$ may lead to the undesired outcome of $\mathcal{C}'_i$ being equal to $\mathcal{C}_i$, thus contradicting the primary goal of community membership hiding. Therefore, it is common to let $\tau \in [0, 1)$ to avoid this scenario.

Moreover, it is essential to emphasize that executing f on $\mathcal{G}'$ instead of the original $\mathcal{G}$ could potentially influence (*i*) the community affiliations of nodes beyond the selected target, u, and (*ii*) the eventual count of recognized communities (i.e., $|f(\mathcal{G}')| = k' \neq k = |f(\mathcal{G})|$), providing that f does not need this number fixed apriori as one of its inputs. Therefore, community membership hiding must strike a balance between two conflicting goals. On the one hand, the target node u must be successfully elided from the original community $\mathcal{C}_i$; on the other hand, the cost of such an operation – i.e., the "distance" between $\mathcal{G}$ and $\mathcal{G}'$, and between $f(\mathcal{G})$ and $f(\mathcal{G}')$ – must be as small as possible.

Overall, we can define the following loss function associated with the community membership hiding task:

$$\mathcal{L}(h_{\boldsymbol{\theta}}; \mathcal{G}, f, u) = \ell_{\text{decept}}(\mathcal{G}, h_{\boldsymbol{\theta}}(\mathcal{G}); f, u) + \lambda \ell_{\text{dist}}(\mathcal{G}, h_{\boldsymbol{\theta}}(\mathcal{G}); f). \tag{4.1}$$

The first component (ℓ_{decept}) penalizes when the goal is *not* satisfied. Let Γ be the set of input graphs which do *not* meet the membership hiding objective, i.e., those which retain node u as part of the community $\mathcal{C}_i$. More formally, let $\widetilde{\mathcal{C}}_i$ be the community to which node u is assigned when f is applied to the input graph $\widetilde{\mathcal{G}}$. We define $\Gamma = \{\widetilde{\mathcal{G}} \mid sim(\mathcal{C}_i - \{u\}, \widetilde{\mathcal{C}}_i - \{u\}) > \tau\}$. Thus, we can compute ℓ_{decept} as follows:

$$\ell_{\text{decept}}(\mathcal{G}, h_{\boldsymbol{\theta}}(\mathcal{G}); f) = \mathbb{1}_{\Gamma}(h_{\boldsymbol{\theta}}(\mathcal{G})), \tag{4.2}$$

where $\mathbb{1}_{\Gamma}(h_{\boldsymbol{\theta}}(\mathcal{G}))$ is the well-known 0-1 indicator function, which evaluates to 1 if $h_{\boldsymbol{\theta}}(\mathcal{G}) \in \Gamma$, or 0 otherwise.

[1] We assume $sim(\cdot, \cdot)$ ranges between 0 and 1.

The second component, denoted as ℓ_{dist}, is a composite function designed to assess the overall dissimilarity between two graphs and their respective communities found by f. This function serves the dual purpose of (i) discouraging the new graph $h_{\boldsymbol{\theta}}(\mathcal{G})$ from diverging significantly from the original graph $\mathcal{G}$ and (ii) preventing the new community structure $f(h_{\boldsymbol{\theta}}(\mathcal{G}))$ from differing substantially from the prior community structure $f(\mathcal{G})$.

4.2 Counterfactual Graph Objective

Given the target community $\mathcal{C}_i$, from which we want to exclude node u, we can classify the remaining nodes $\mathcal{V} - \{u\}$ of $\mathcal{G}$ into two categories: nodes that are inside the same community $\mathcal{C}_i$ as u and nodes that belong to a different community from u. This categorization helps us define which edges the target node u can control and, thus, directly manipulate under the assumption that $\mathcal{G}$ is a directed graph.[2] Specifically, following [10], we consider that u can (i) *remove* existing outgoing edges to nodes that are inside u's community (ii) *add* new outgoing edges to nodes that are outside u's community. We explicitly neglect two theoretically plausible operations, namely (iii) removing outgoing links to outside-community nodes and (iv) adding outgoing links to inside-community nodes. The rationale for this choice is twofold. On the one hand, allowing (iii) would increase the risk of further isolating u and its original community $\mathcal{C}_i$. On the other hand, allowing (iv) would strengthen the connectivity between u and the other nodes of $\mathcal{C}_i$. Both events contradict the goal of community membership hiding.

Overall, we can define the set of candidate edges to remove ($\mathcal{E}^-_{u,i}$) and to add ($\mathcal{E}^+_{u,i}$) as follows.

$$\mathcal{E}^-_{u,i} = \{(u,v) \mid u,v \in \mathcal{C}_i \wedge (u,v) \in \mathcal{E}\},$$
$$\mathcal{E}^+_{u,i} = \{(u,v) \mid u \in \mathcal{C}_i, v \notin \mathcal{C}_i \wedge (u,v) \notin \mathcal{E}\}.$$

If we suppose the target node u has a fixed budget $\beta > 0$, we can find the optimal model $h^* = h_{\boldsymbol{\theta}^*}$ as the one whose parameters $\boldsymbol{\theta}^*$ minimize Eq. (4.1), i.e., by solving the following constrained objective:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \Big\{ \mathcal{L}(h_{\boldsymbol{\theta}}; \mathcal{G}, f, u) \Big\} \quad \text{subjetc to: } |\mathcal{B}_{u,i}| \leq \beta, \tag{4.3}$$

where $\mathcal{B}_{u,i} \subseteq \mathcal{E}^-_{u,i} \cup \mathcal{E}^+_{u,i}$ is the set of graph edge modifications selected from the candidates.

Note that Eq. (4.3) resembles the optimization task to find the best *counterfactual graph* $\mathcal{G}^* = h^*(\mathcal{G})$ that, when fed back into f, changes its output to hide the target node u from its community.

[2] We can easily extend this reasoning if $\mathcal{G}$ is *undirected*.

4.3 Markov Decision Process

We approach the community membership hiding problem defined in Eq. (4.3) as a sequential decision-making process, following standard reinforcement learning principles. In this framework, at each time step, an agent: *(i)* takes an action (choosing to add or remove an edge based on the rules defined above), and *(ii)* observes the new set of communities output by f when this is fed with the graph modified according to the action taken before. The agent also receives a scalar reward from the environment. The process continues until the agent eventually meets the specified deception goal and the optimal counterfactual graph $\mathcal{G}^*$ – i.e., the optimal h^* – is found. We formalize this procedure as a typical Markov decision process (MDP) denoted as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, p_0, r, \gamma\}$. Below, we describe each component of this framework separately.

4.3.1 States ($\mathcal{S}$)

At each time step t, the agent's state is $S_t = s_t$, where $s_t = \mathcal{G}^t \in \mathcal{S}$ is the current modified input graph. In practice, though, we can replace $\mathcal{G}^t$ with its associated adjacency matrix $A_t \in \{0,1\}^{n \times n}$. Initially, when $t = 0$, $\mathcal{G}^0 = \mathcal{G}$ ($A^0 = A$).

4.3.2 Actions ($\mathcal{A}$)

The set of actions is defined by $\mathcal{A} = \{a_t\}$, which consists of all valid graph rewiring operations, assuming node u belongs to the community C_i.

$$\mathcal{A} = \{\text{del}(u,v) | (u,v) \in \mathcal{E}^-_{u,i}\} \cup \{\text{add}(u,v) | (u,v) \in \mathcal{E}^+_{u,i}\}. \tag{4.4}$$

According to the allowed graph modifications outlined in Section 4.2, the agent can choose between two types of actions: deleting an edge from u to any node within the same community C_i or adding an edge from u to any node in a different community.

4.3.3 Transitions Probability ($\mathcal{P}$)

Let $a_t \in \mathcal{A}$ be the action taken by the agent at iteration t. This action deterministically guides the agent's transition from the state s_t to the state s_{t+1}. In essence, the transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$, which associates a transition probability with each state-action pair, assigns a transition probability of 1 when the subsequent state s_{t+1} is determined by the state-action pair (s_t, a_t) and 0 otherwise. Formally:

- $\mathcal{P}(s_{t+1}|s_t, a_t) = 1$ if s_{t+1} is the next state resulting from the application of action a_t in state s_t.
- $\mathcal{P}(s_{t+1}|s_t, a_t) = 0$ otherwise.

4.3.4 Reward (r)

The reward function of the action a_t which takes the agent from state s_t to state s_{t+1} can be defined as:

$$r(s_t, a_t) = \begin{cases} 1 - \lambda(\ell^t_{\text{dist}} - \ell^{t-1}_{\text{dist}}) & \text{, if "the goal is met"} \\ -\lambda(\ell^t_{\text{dist}} - \ell^{t-1}_{\text{dist}}) & \text{, otherwise.} \end{cases} \tag{4.5}$$

The goal is considered successfully achieved when $f(\mathcal{G}^t)$ leads to $u \in \mathcal{C}_i^t \neq \mathcal{C}_i$ such that $sim(\mathcal{C}_i - \{u\}, \mathcal{C}_i^t - \{u\}) \leq \tau$. In addition, $\ell_{\text{dist}}^t = \ell_{\text{dist}}(\mathcal{G}, \mathcal{G}^t; f)$ measures the penalty computed on the graph before and after action a_t, and $\lambda \in \mathbb{R}_{>0}$ is a parameter that controls its weight. More precisely, the penalty is calculated as follows:

$$\ell_{\text{dist}}(\mathcal{G}, \mathcal{G}^t; f) = \alpha \times d_{\text{community}} + (1 - \alpha) \times d_{\text{graph}}, \tag{4.6}$$

where $d_{\text{community}}$ computes the distance between the community structures $f(\mathcal{G})$ and $f(\mathcal{G}^t)$, d_{graph} measures the distance between the two graphs $\mathcal{G}$ and $\mathcal{G}^t$, and the parameter $\alpha \in [0, 1]$ balances the importance between the two distances.

Hence, the reward function encourages the agent to take actions that preserve the similarity between the community structures and the graphs before and after the rewiring action.

4.3.5 Policy ($\pi_{\boldsymbol{\theta}}$)

We first define a parameterized policy $\pi_{\boldsymbol{\theta}}$ that maps from states to actions. We then want to find the values of the policy parameters $\boldsymbol{\theta}$ that maximize the expected reward in the MDP. This is equivalent to finding the optimal policy π^*, which is the policy that gives the highest expected reward for any state. We can find the optimal policy π^* by minimizing the Eq. (4.3). This minimization leads to the discovery of the optimal model h^*, which is the model that best predicts the rewards in the MDP.

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \ell_{\text{decept}}(h_{\boldsymbol{\theta}}; \mathcal{G}, f, u) + \lambda \ell_{\text{dist}}(\mathcal{G}, h_{\boldsymbol{\theta}}(\mathcal{G}); f) \\
&= \arg\max_{\boldsymbol{\theta}} \; - \ell_{\text{decept}}(h_{\boldsymbol{\theta}}; \mathcal{G}, f, u) - \lambda \ell_{\text{dist}}(\mathcal{G}, h_{\boldsymbol{\theta}}(\mathcal{G}); f) \\
&= \arg\max_{\boldsymbol{\theta}} \begin{cases} 1 - \lambda(\ell_{\text{dist}}^t - \ell_{\text{dist}}^{t-1}) & \text{, if “the goal is met”} \\ -\lambda(\ell_{\text{dist}}^t - \ell_{\text{dist}}^{t-1}) & \text{, otherwise} \end{cases} \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{t=1}^{T} r(s_t, \pi_{\boldsymbol{\theta}}(s_t)),
\end{aligned} \tag{4.7}$$

where T is the maximum number of steps per episode taken by the agent and is therefore always less than the allowed number of graph manipulations, i.e., $T \leq \beta$.

Chapter 5

Proposed Method

In this comprehensive exploration of the Proposed Method, we dive into the intricate world of the Advantage Actor-Critic (A2C) agent and its underlying architecture, and then, moving on to the implementation of the code.

In particular, the development of our agent and its interaction with the environment were carried out using Python and several indispensable libraries, including `PyTorch`[1], `PyTorch Geometric`[2], and `NetworkX`[3]. These libraries played a pivotal role in facilitating various graph operations and enhancing the capabilities of our agent.

In the following sections, we will provide detailed descriptions of each of the key modules within our project. Fig. 5.1 displays the UML class diagram, which serves as a visual representation of the modules and their interconnections, providing an overview of what we are about to delve into.

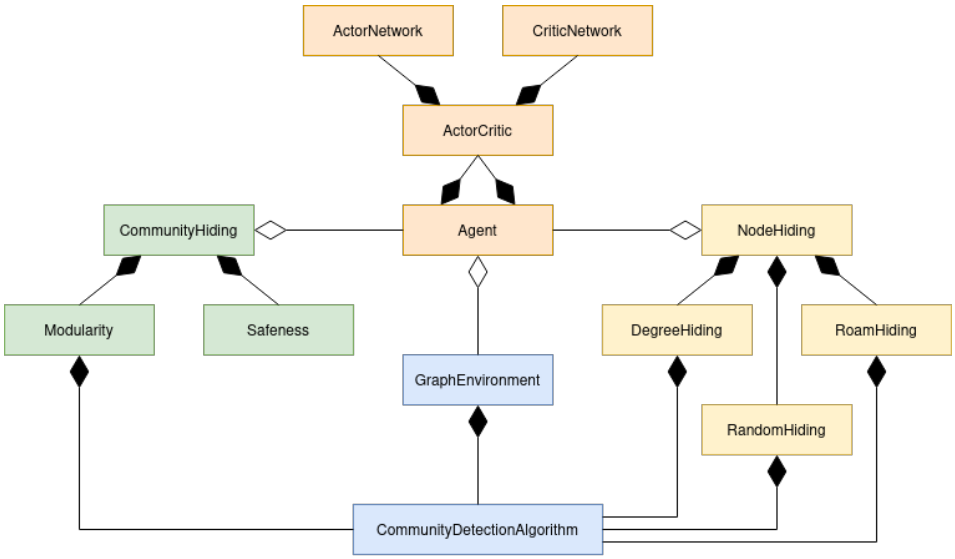


Figure 5.1. UML diagram of the main classes.

[1] `https://pytorch.org/`
[2] `https://pyg.org/`
[3] `https://networkx.org/`

5.1 Advantage Actor-Critic Agent

To learn the optimal policy for our agent defined above, we use the *Advantage Actor-Critic* (A2C) algorithm [26], a popular deep reinforcement learning technique that combines the advantages of both policy-based and value-based methods. Specifically, A2C defines two neural networks, one for the policy function ($\pi_{\boldsymbol{\theta}}$) and another for the value function estimator (V_v), such that:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) \sim \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) A(s_t, a_t),$$
$$\text{with } A(s_t, a_t) = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t),$$

where $\mathcal{J}(\boldsymbol{\theta})$ is the reward (objective) function, and the goal is to find the optimal policy parameters $\boldsymbol{\theta}$ that maximize it. Instead, $A(s_t, a_t)$ is the advantage function, which quantifies how good or bad an action a_t is compared to the expected value of taking actions according to the current policy.

Below, we describe the policy (*actor*) and value function network (*critic*) separately.

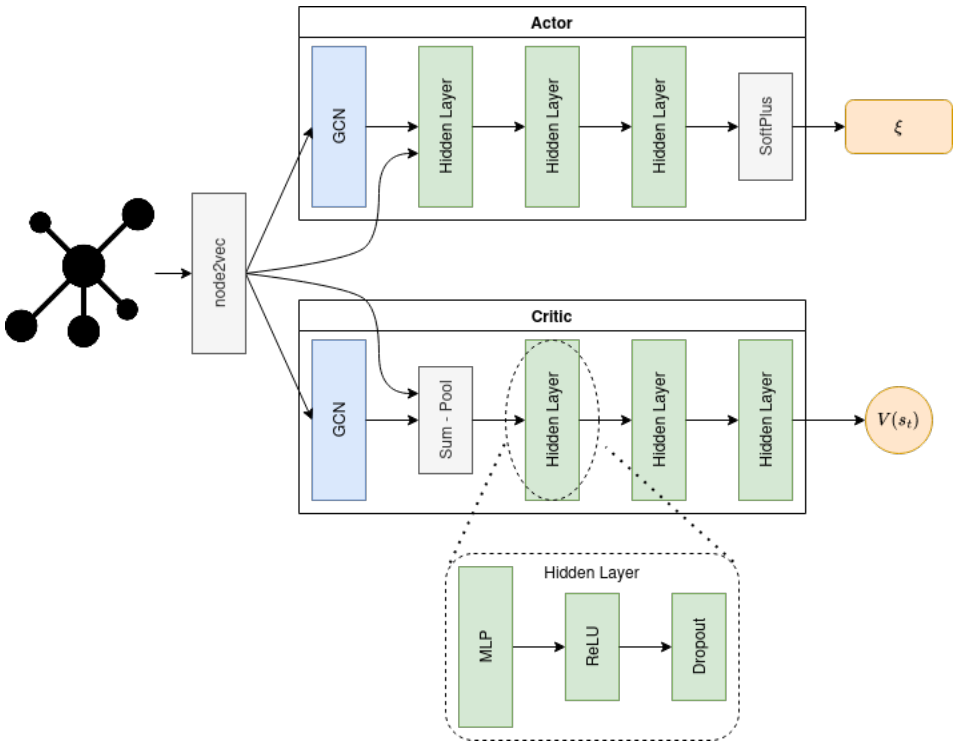


Figure 5.2. Network architecture overview of the *Actor* and *Critic*. Initially, the node's continuous feature vectors are acquired by employing *node2vec*, subsequently modified through the graph convolutions and processed through non-linearities to establish the concentration parameters $\xi \in \mathbb{R}_+^{|\mathcal{V}|}$ (i.e. correlated with the probability density on the shares) and the estimated value function $V(s_t)$.

5.1.1 Policy function (Actor)

The policy function network is responsible for generating a probability distribution over possible actions based on the input, which consists of a list of nodes and the graph's feature matrix. However, some graphs may lack node features. In such cases, we can extract continuous node feature vectors (i.e., node embeddings) with graph representational learning frameworks like *node2vec* [14]. These node embeddings serve as the feature matrix.

Our neural network implementation comprises a primary graph convolution layer (GCNConv [19]) for updating node features. The output of this layer, along with skip connections, feeds into a block consisting of three hidden layers. Each hidden layer includes multi-layer perception (MLP) layers, ReLU activations, and dropout layers. The final output is aggregated using a sum-pooling function. In building our network architecture, we were inspired, in part, by the work conducted by Gammelli et al. [12], adapting it to our task. The policy is trained to predict the probability that node v is the optimal choice for adding or removing the edge (u, v) to hide the target node u from its original community. The feasible actions depend on the input node u and are restricted to a subset of the graph's edges as outlined in Section 4.2. Hence, not all nodes $v \in \mathcal{V}$ are viable options for the policy.

5.1.2 Value function (Critic)

This network resembles the one employed for the policy, differing only in one aspect: it incorporates a global sum-pooling operation on the convolution layer's output. This pooling operation results in an output layer with a size of 1, signifying the estimated value of the value function. The role of the value function is to predict the state value when provided with a specific action a_t and state s_t.

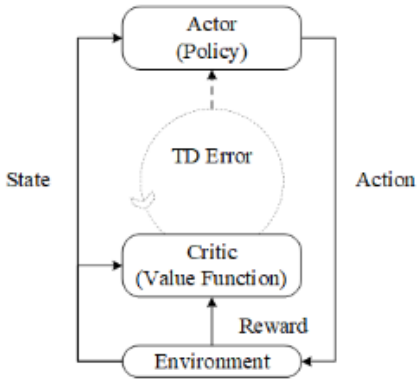


Figure 5.3. The foundation of the Actor-Critic algorithm, as outlined in the research by Nie et al. [31], operates as follows: The Actor component takes a state from the environment and decides on an action to execute. Simultaneously, the Critic component receives both the current state and the state resulting from the previous interaction. It computes the TD (Temporal Difference) error, leveraging this error to refine and update both the Critic and Actor components.

Algorithm 1 Advantage Actor-Critic Training

procedure N-Step Advantage Actor-Critic
    Initialize policy network $\pi_{\boldsymbol{\theta}}(s, a)$ *and value network* $V_\nu(s)$
    Initialize optimizer for policy η *and for value network* ϕ
    Initialize graph $\mathcal{G}$*environment*
    for *episode* $= 1$ **to** 10000
        Reset environment to initial state s, *and get new budget* β
        Initialize empty buffers: $s_{buffer}, a_{buffer}, r_{buffer}$
        $t = 0$
        while *not* `done` **or** $t < \beta$:
            Sample action a_t *from policy* $\pi_{\boldsymbol{\theta}}(s_t, a_t)$
            Execute action a_t
            Observe reward r_t, *and new state* s_{t+1}, *and check* `done`
            Store (s_t, a_t, r_t) *in buffers*
            $t = t + 1$
        endwhile
        `// Compute the discounted future rewards`
        `// for each reward in the buffer`
        $R \leftarrow 0$
        for $i = t$ **to** 0
            $R = r_i * \gamma R$
        endfor
        `// Compute advantage` A
        for $i = 0$ **to** t
            $A(s_i, a_i) = R - V_\nu(s_i)$
        endfor
        `// Compute policy loss`
        $L_{policy} = \sum_{i=1}^{t} \log(\pi_{\boldsymbol{\theta}}(a_i|s_i)) \cdot A(s_i, a_i)$
        `// Compute value loss`
        $L_{value} = \frac{1}{2} \sum_{i=1}^{t} (V_\nu(s_i) - R)^2$
        `// Compute the combined loss:`
        $L = L_{policy} + L_{value}$
        `// Update policy network weights`
        $\theta \leftarrow \theta - \eta \nabla_\theta L_{policy}$
        `// Update value network weights`
        $\nu \leftarrow \nu - \phi \nabla_\phi L_{value}$
    endfor
end procedure

5.2 Environment

The second area revolves around the agent's environment and its associated libraries and modules. To effectively train and assess the performance of our A2C agent in the task of hiding a node within its initial community, we have created a specialized environment. This environment serves as the stage where the agent interacts with the graph executes rewiring actions, computes rewards, and manages episode progression.

The setup process begins with the initialization of the graph data to create a suitable environment for the task at hand. Subsequently, we calculate the budget, taking into account the average degree of the graph and a given multiplier, beta. We also define the detection algorithm, which plays a crucial role in calculating rewards. At each time step within an episode, the agent selects an action corresponding to a specific node in the graph. This action instructs the agent to reconfigure connections between the chosen node and the node to be concealed while adhering to a predetermined budget parameter (β). The agent can either delete existing connections or establish new ones, always staying within the budget constraint.

The calculation of rewards is a critical aspect of training the A2C agent. Our reward mechanism is designed to support the community structure by penalizing disruptive actions while preserving the overall graph structure. For a comprehensive explanation of this reward system, please consult Section 4.3.4. An episode concludes when a predetermined number of steps have been executed or the goal of concealing nodes has been achieved. At the end of each episode, the environment is reset to its original state, restoring the community's graphical structure and labels. This approach ensures that the agent can adapt to various situations and maintain community cohesion during subsequent episodes.

Within this environment, a crucial module is responsible for implementing Community Detection algorithms, which are utilized at each step to calculate rewards. These algorithms are realized using the Python `iGraph`[4] library, which not only manages the graph but also incorporates diverse clustering algorithms.

In summary, our customized environment provides a comprehensive platform for training and evaluating the A2C agent's ability to adeptly rewire nodes while preserving community structure and adhering to budget constraints.

5.3 Performance Evaluation

This module serves a crucial role in our study, as it enables you to perform comprehensive performance comparisons of the agent. It does so by executing a substantial number of iterations, during each of which a different node is selected from a community distinct from the one in the previous iteration. The primary objective here is to conceal this node from its initial community, all while adhering to a specified constraint denoted as τ.

To provide a robust benchmark for our comparisons, we employ three baseline methods: Random, Degree, and Roam. These baseline techniques will be thoroughly

[4] https://igraph.org/

elucidated in Section 6.2, offering you a detailed understanding of their functioning and purpose in our evaluation process.

Moving forward, the final module of our research focuses on testing the agent's capabilities in the context of the community deception task. In this challenging task, the agent's goal, in each iteration, is to select a different community to hide from the detection algorithms. This intricate test scenario enables us to scrutinize the agent's performance vis-à-vis the Safeness and Modularity criteria, which are further expounded upon in Section 6.3. By analyzing the agent's behavior and effectiveness in evading detection within diverse communities, we gain valuable insights into its adaptability and resilience, thereby contributing to a more comprehensive evaluation of its performance.

In addition to these core modules, various others are employed for defining and calculating various metrics essential for reward computation and performance assessment. For more detailed information, please refer to the GitHub repository[5].

[5]https://github.com/AndreaBe99/community-deception-thesis

Chapter 6

Experiments

6.1 Experimental Setup

6.1.1 Dataset

To keep the training time for our DRL agent computationally feasible, we train it on the real dataset `words`,[1] This dataset strikes a favorable balance in terms of the number of nodes, edges, and discovered communities. In fact, a swift growth in the number of nodes and, consequently, edges would result in an exponential surge in potential actions that the agent can choose from, thereby leading to unmanageable training times.

In addition, we evaluate the performance of our method on two additional datasets: `kar`[1] and Wikipedia's `vote`.[2]

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	**Community Detection Algorithm**		
			`gre`	`louv`	`walk`
`kar`	34	78	3	4	5
`words`	112	425	7	7	25
`vote`	889	2,900	12	10	42

Table 6.1. Properties of the datasets used along with the size of the communities detected by `greedy`, `louvain`, and `walktrap`.

6.1.2 Community Detection Algorithms

The DRL agent is trained using a single detection algorithm, i.e., our $f(\cdot)$, namely the modularity-based Greedy (`gre`) algorithm [3].

During the testing phase, we employ two additional algorithms. One of them, the Louvain (`louv`) algorithm [1], falls within the same family as the one used for training. The other, the WalkTrap (`walk`) [32], takes a distinct approach centered on Random Walks.

[1] http://konect.cc/

[2] https://networkrepository.com

Table 6.1 provides an overview of the datasets used, including key properties and the number of communities detected by each community detection algorithm. The modularity-based algorithms (`gre` and `louv`) yield comparable community counts, while `walk` identifies a generally higher number of communities.

All three, are community detection algorithms, quite robust and available in various libraries related to graph manipulation (`CDlib` [3], `iGraph` [4]).

6.1.3 Distance Metrics

As described in Eq. (4.6), the penalty function consists of two mutually balanced factors, namely $d_{\text{community}}$ and d_{graph}, which quantify the dissimilarity between community structures and graphs before and after the action. During model training, we operationalize these distances using the Normalized Mutual Information (NMI) score for community comparison and the Jaccard distance for graph comparison.

The NMI score [37, 21], utilized for measuring the similarity between community structures, ranges from 0 (indicating no mutual information) to 1 (indicating perfect correlation). Following the formulation by Lancichinetti et al. [22], it can be expressed as follows:

$$\text{NMI}(\mathcal{K}, \mathcal{K}^t) = I_{\text{norm}}(X : Y) = \frac{H(X) + H(Y) - H(X, Y)}{(H(X) + H(Y))/2}, \tag{6.1}$$

where $H(X)$ and $H(Y)$ denote the entropy of the random variables X and Y associated with partitions $\mathcal{K} = f(\mathcal{G})$ and $\mathcal{K}^t = f(\mathcal{G}^t)$, respectively, while $H(X, Y)$ denotes the joint entropy. Since we need to transform this metric into a distance, we calculate 1-NMI.

The Jaccard distance can be adapted to the case of two graphs, as described in [9], as follows:

$$\text{Jaccard}(\mathcal{G}, \mathcal{G}^t) = \frac{|\mathcal{G} \bigcup \mathcal{G}^t| - |\mathcal{G} \bigcap \mathcal{G}^t|}{|\mathcal{G} \bigcup \mathcal{G}^t|} = \frac{\sum_{i,j} |A_{i,j} - A^t_{i,j}|}{\sum_{i,j} \max(A_{i,j}, A^t_{i,j})}, \tag{6.2}$$

where $A_{i,j}$ denotes the (i, j)-th entry of the adjacency matrix for the original graph $\mathcal{G}$. The Jaccard distance yields a value of 0 when the two graphs are identical and 1 when they are entirely dissimilar.

Finally, to assess the achievement of the goal, i.e., whether the new community $\mathcal{C}^t_i$ of node u at step t can no longer be considered the same as the initial community $\mathcal{C}_i$, we employed the Sørensen-Dice coefficient [8], which is defined as follows:

$$\text{DSC}(\mathcal{C}_i, \mathcal{C}^t_i) = \frac{2|\mathcal{C}_i \bigcap \mathcal{C}^t_i|}{|\mathcal{C}_i| + |\mathcal{C}^t_i|}, \tag{6.3}$$

which returns a value between 0 (no similarity) and 1 (strong similarity). If that value is less than or equal to the parameter τ, we consider the deception goal successfully met.

[3] https://cdlib.readthedocs.io/en/latest/

[4] https://igraph.org/

6.1.4 Tasks

We validate our method on two separate tasks: *node deception* and *community deception*. The former directly stems from the community membership hiding problem defined in Section 4. Its goal is to conceal an individual node from the community to which it was initially identified. The latter, instead, can be seen as a specific instance of node deception, where the objective is to hide an entire community from a designated detection algorithm. In essence, this involves performing multiple node deception tasks, one for each node within the community to be masked. Actually, this is a simplification of the process since each node deception task may indirectly hide other nodes within the same community.

6.2 Node Deception Task

6.2.1 Beselines

We evaluate the performance of our method on the node deception task by comparing it to three baseline approaches, described below.

1. *Random-based.* This baseline operates by randomly selecting one of the nodes in the graph. If the selected node is a neighbor of the node to be hidden (i.e., there is an edge between them), the edge is removed; otherwise, it is added. The randomness of these decisions aims to obscure the node's true community membership.

2. *Degree-based.* This approach adopts a different strategy for node concealment. Specifically, it selects nodes with the highest degrees within the graph and rewires them. By prioritizing nodes with higher degrees, this baseline seeks to disrupt the node's central connections within its initial community, thus promoting concealment.

3. *Roam-based.* Our third baseline is based on the Roam heuristics [40], originally designed to reduce a node's centrality within the network. This hiding approach aims to diminish the centrality and influence of the target node within its initial community, making it less conspicuous and favoring its deception.

6.2.2 Evaluation Metrics

We measure the performance of each method in solving the node deception task using the following metrics.

1. *Success Rate* (SR). This metric calculates the success rate of the node deception algorithm by determining the percentage of times the target node is successfully hidden from its original community. If, after applying the node deception algorithm, the target node no longer belongs to the original community (as per Eq. (6.3) and the τ constraint), we consider the goal achieved. By repeating this procedure for several nodes and communities, we can estimate the algorithm's success rate. Obviously, a higher value of this metric indicates better performance.

2. *Normalized Mutual Information* (NMI). To quantify the impact of the function $h(\cdot)$ on the resulting community structure, denoted as the output of $f(\mathcal{G}')$ where $\mathcal{G}'$ is the graph created by modifying the original graph $\mathcal{G}$, we compute $\text{NMI}(\mathcal{K}, \mathcal{K}')$, as outlined in Eq. (6.1). This score measures the similarity between the two structures, $\mathcal{K} = f(\mathcal{G})$ and $\mathcal{K}' = f(\mathcal{G}')$. A higher value for this metric indicates a greater degree of similarity between the original and modified community structures.

6.3 Community Deception Task

We adapt our method, originally conceived for the node deception task, to the community deception task as follows. We iterate the execution of our agent on every node of the community to hide. In doing so, we fulfill the constraint on the number of possible actions (i.e., graph rewirings). Specifically, we set the same constraint as in [10], which limits the percentage of edges of the graph that can be modified. To make more efficient use of its budget, our agent, at each step, ranks the nodes within the community to hide based on their centrality degrees, starting with the most central node and progressing to the least central. The reasoning behind this selection process is that by prioritizing highly central nodes, the agent is more likely to mask more nodes within the target community using the same budget compared to selecting nodes randomly. In the next step, based on the number of actions performed, the agent chooses another node to hide from the remaining nodes within the target community, with the budget adjusted accordingly. We describe this procedure in Algorithm 1, provided in Appendix B.

6.3.1 Baselines

We compare our method with two of the most popular community deception algorithms available in the literature, both proposed by Fionda and Pirrò [10]. Those approaches are *Modularity-based* and *Safeness-based*.

6.3.2 Evaluation Metrics

We validate the quality of each method considered by comparing the community structures obtained before and after applying the node deception function $h(\cdot)$. In particular, we calculate the *Deception Score* as defined in [10] and the NMI described in Eq. (6.1).

The Deception Score ($\mathcal{H}$) evaluates the level of concealment of a set of nodes $\mathcal{C}_i$, within a community structure $\mathcal{K}$ identified by a detection algorithm $f(\cdot)$, i.e., $\mathcal{K} = f(\mathcal{G})$. It is defined as follows:

$$\mathcal{H}(\mathcal{C}_i, \mathcal{K}) = \left(1 - \frac{|S(\mathcal{C}_i)| - 1}{|\mathcal{C}_i| - 1}\right) \quad \times$$
$$\left[\frac{1}{2}(1 - \max_{\mathcal{C}_j \in \mathcal{K}}\{R(\mathcal{C}_j, \mathcal{C}_i)\}) + \frac{1}{2}\left(1 - \frac{\sum_{\mathcal{C}_j \cap \mathcal{C}_i \neq \emptyset} P(\mathcal{C}_j, \mathcal{C}_i)}{|\mathcal{C}_j \cap \mathcal{C}_i \neq \emptyset|}\right)\right],$$

where $R(\mathcal{C}_j, \mathcal{C}_i)$ is the *recall*, $P(\mathcal{C}_j, \mathcal{C}_i)$ is the *precision*, and $|S(\mathcal{C}_i)|$ is the number of connected components in the subgraph induced by the members of $\mathcal{C}_i$.

The Deception Score yields values ranging from 0 to 1. When $\mathcal{H} = 0$, it means that the members of a community $\mathcal{C}_i$ are completely enclosed within one of the communities of $\mathcal{C}$ and are part of different connected components. Conversely, when $\mathcal{H} \approx 1$, it indicates that each node in $\mathcal{C}_i$ belongs to a distinct, larger community within the same connected component.

6.4 Results and Discussion

The experiments were conducted on the Kaggle platform,[5] which provides an Intel Xeon CPU at 2.2 GHz (4 cores) and 18 GB RAM. This section is divided into two paragraphs to present the results of the two distinct tasks: *node deception* and *community deception*.

6.4.1 Node Deception Task

In this study, we evaluate our *DRL-Agent*'s performance against the baseline methods discussed in Section 6.2. We investigate various parameter settings, including different values for the similarity constraint τ $(0.3, 0.5, 0.8)$ and the budget β $(\frac{1}{2}\mu, 1\mu, 2\mu$, where $\mu = \frac{|\mathcal{E}|}{|\mathcal{V}|}$). We evaluate all possible combinations of these parameters. For each parameter combination, dataset, and community detection algorithm, we conducted a total of 100 experiments. In each iteration, we randomly select a node from a different community than the previous one for concealment. The reported results are based on the average outcomes across all runs.

Here, we emphasize two primary findings. Firstly, our approach consistently surpasses baseline techniques when trained on the *same* community detection algorithm employed for the node deception task (*symmetric* setup). In Fig. 6.1, we present the Success Rate and NMI score for our *DRL-Agent* trained and tested on the modularity-based Greedy algorithm.

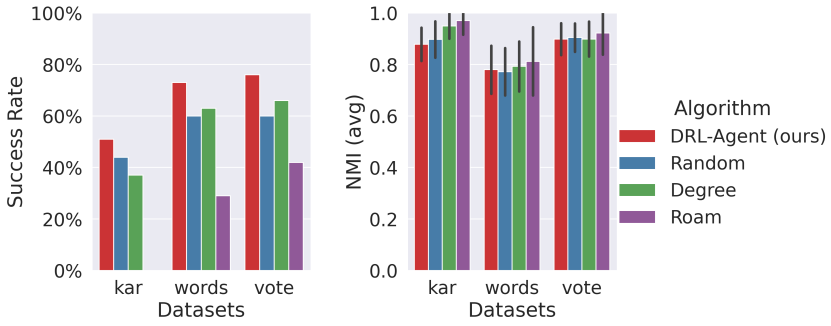


Figure 6.1. SR and NMI using the Greedy algorithm, as $f(\cdot)$, on all datasets with $\tau = 0.3$, and $\beta = 1\mu$, both at training and node deception time (*symmetric*).

A crucial observation is that our method strikes the best balance between the Success Rate and the NMI score. On the one hand, it achieves a higher success rate in hiding the selected target node than competing methods (reaching up to 80%

[5] https://www.kaggle.com/

with an improvement of approximately 15% over the best-performing baseline). On the other hand, our approach pays a limited price for this success (i.e., the modified graph remains relatively similar to the original one), as evidenced by the NMI score compared to other methods. In fact, for the `words` and `vote` datasets, the NMI score achieved by our *DRL-Agent* is similar to that of the baselines. In the case of the `kar` dataset, our method's NMI score is slightly lower than that of, for example, Roam. However, while our method accomplishes the node deception task around 50% of the time, Roam always fails.

The second key finding concerns the ability of our *DRL-Agent* to *transfer* to a different community detection algorithm (*asymmetric* setup). Specifically, we evaluate the performance of our method, which was trained on the modularity-based Greedy algorithm, in the context of a node deception task that utilizes a different community detection algorithm, namely the Louvain algorithm. As illustrated in Fig. 6.2, both the baselines and our *DRL-Agent* achieve even higher success rates in this asymmetric setting, with our method maintaining its superiority. Naturally, this outcome has a more substantial impact on the NMI score. However, intriguingly, the disparity between our method's NMI and other approaches becomes even less pronounced than in the previous symmetric setup. Similar conclusions can be drawn for the other community detection algorithms considered and using different combinations of values for the key parameters τ and β.

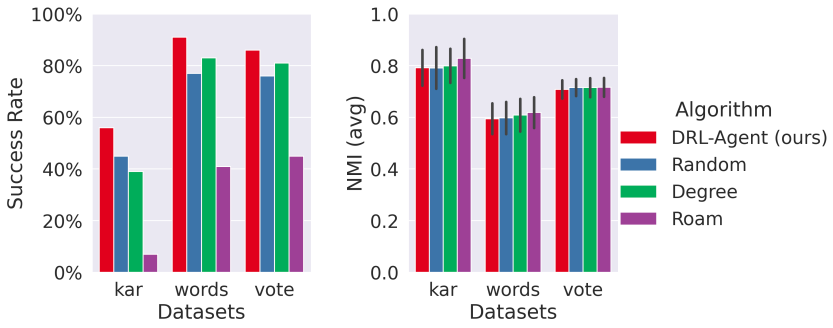
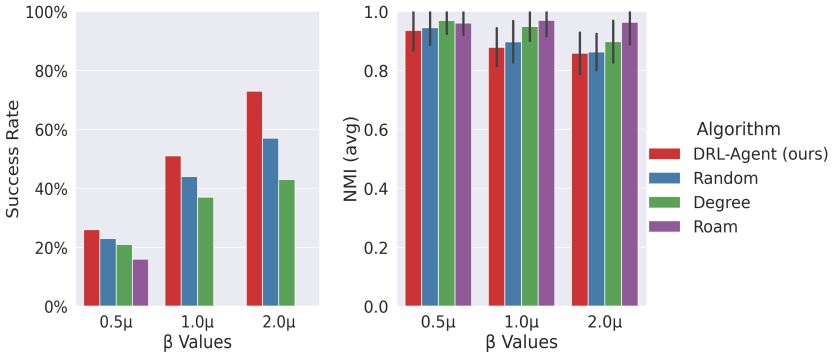
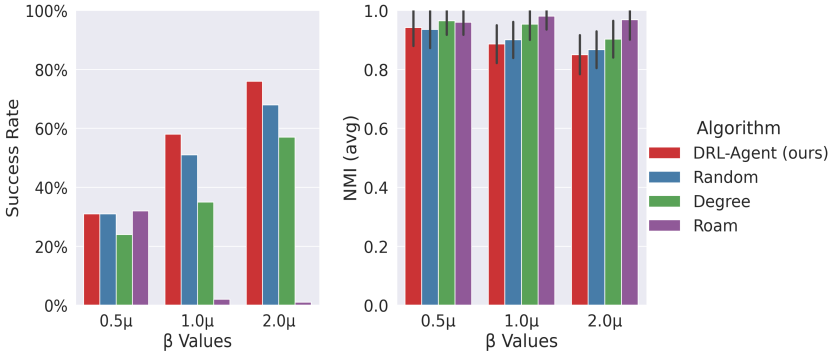


Figure 6.2. SR and NMI using the Greedy algorithm at training time and the Louvain algorithm at node deception time, as $f(\cdot)$, on all datasets with $\tau = 0.3$, and $\beta = 1\mu$ (*asymmetric*).

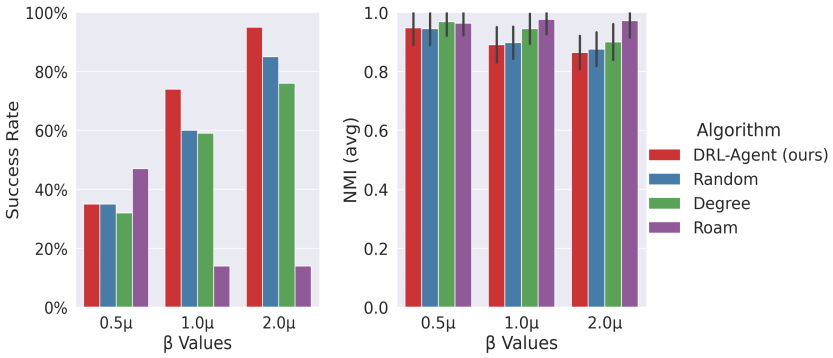
Below we report the complete results of the node deception task, using the Greedy algorithm at training time, on various datasets and algorithms, and above all on varying τ and β. Briefly, the *DRL-Agent* always performs better than the baselines when using modularity-based algorithms. Instead, when the WalkTrap (`walk`) detection algorithm is used, the performances of the three algorithms are comparable, with the *Degree-based* achieving high success rates in some cases. This could be attributed to WalkTrap's ability to generate a larger number of communities than the other two detection algorithms, resulting in fewer nodes per community, thus facilitating easier community transitions for a node. However, this is not the case for the smaller `kar` dataset, where the *DRL-Agent* achieves better performance than the baselines using the algorithm based on Random Walks.



(a) $\tau = 0.3$

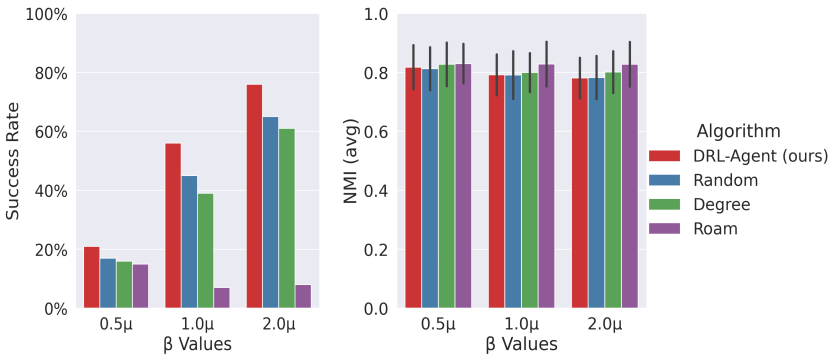


(b) $\tau = 0.5$

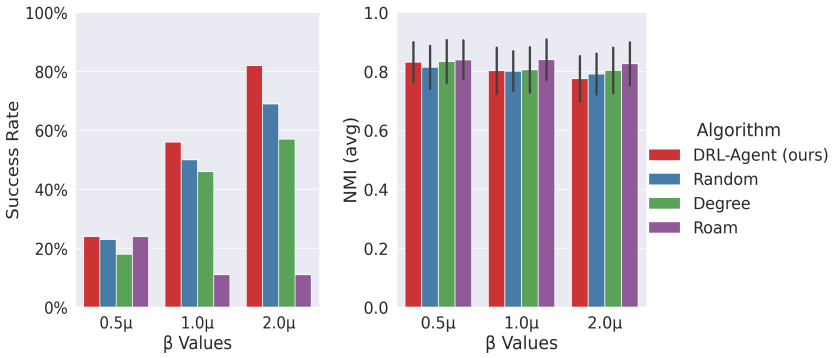


(c) $\tau = 0.8$

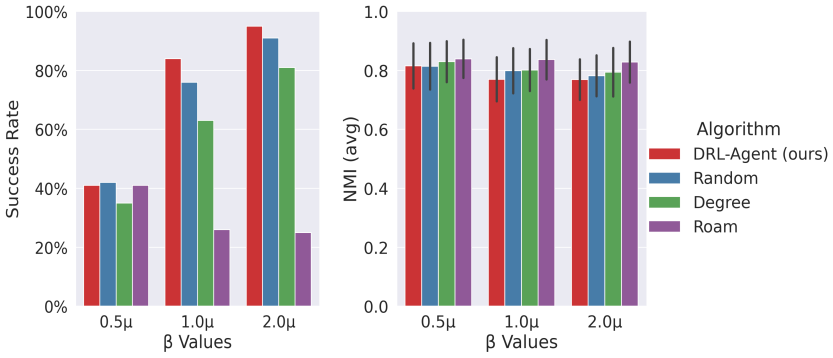
Figure 6.3. Node Deception, SR and NMI with `gre` as $f(\cdot)$, on `kar` dataset.



(a) $\tau = 0.3$

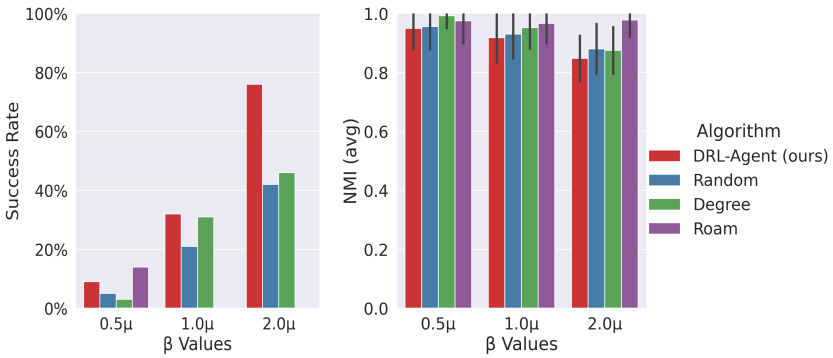


(b) $\tau = 0.5$

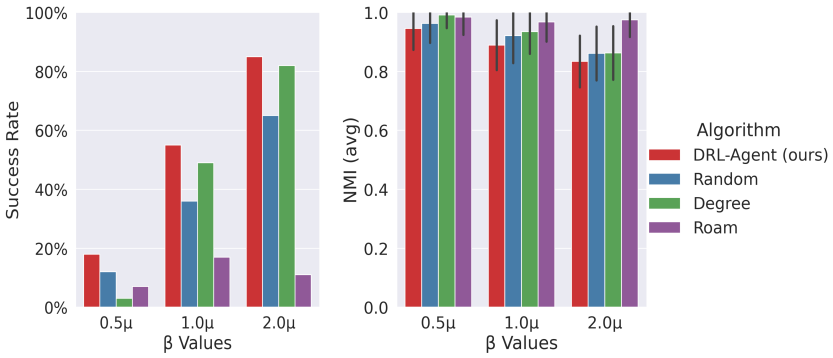


(c) $\tau = 0.8$

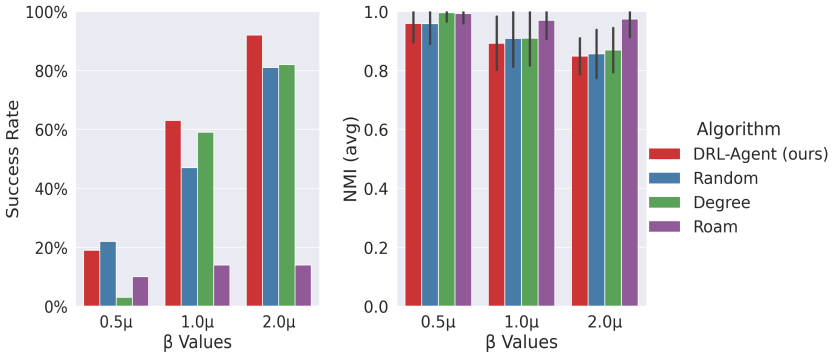
Figure 6.4. Node Deception, SR and NMI with `louv` as $f(\cdot)$, on `kar` dataset.



(a) $\tau = 0.3$

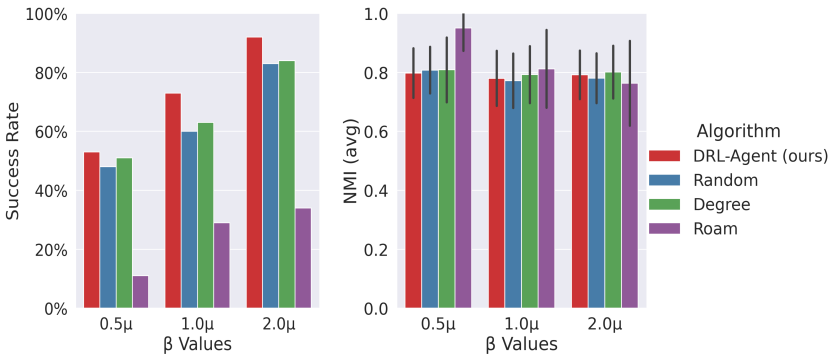


(b) $\tau = 0.5$

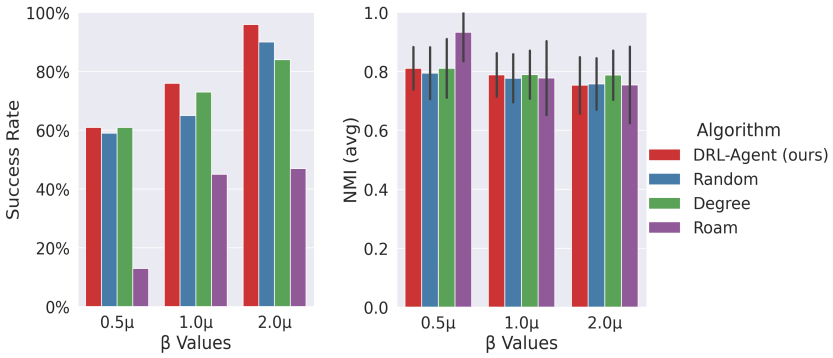


(c) $\tau = 0.8$

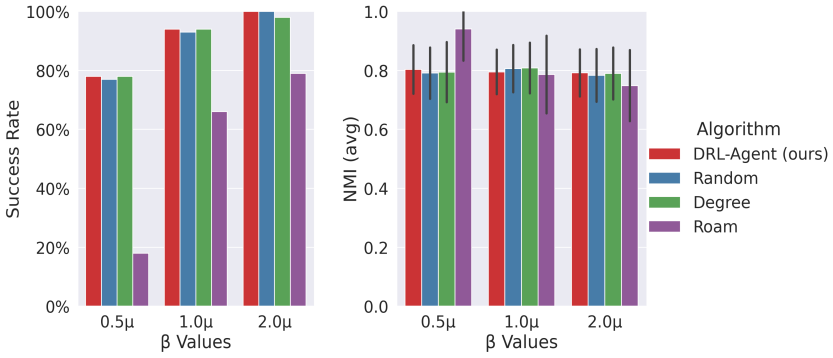
Figure 6.5. Node Deception, SR and NMI with `walk` as $f(\cdot)$, on `kar` dataset.



(a) $\tau = 0.3$

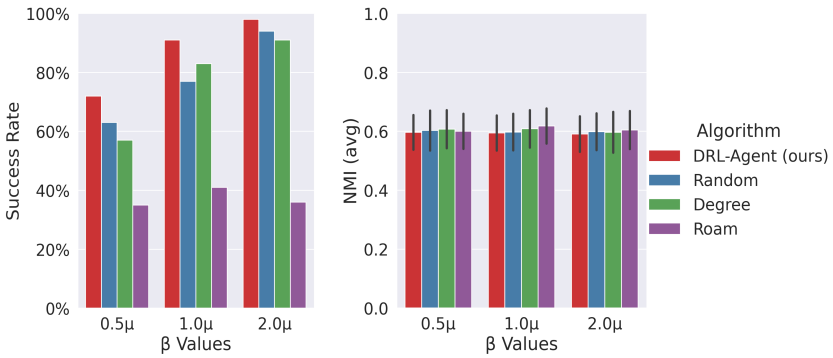


(b) $\tau = 0.5$

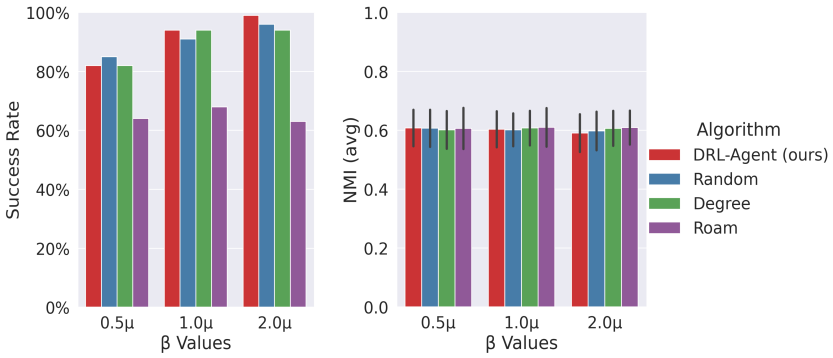


(c) $\tau = 0.8$

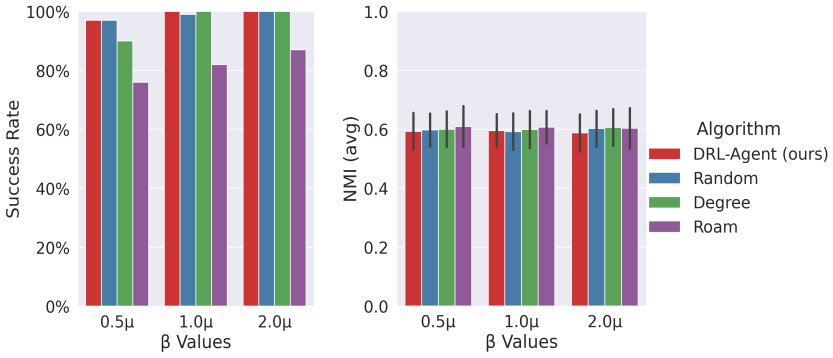
Figure 6.6. Node Deception, SR and NMI with `gre` as $f(\cdot)$, on `words` dataset.



(a) $\tau = 0.3$

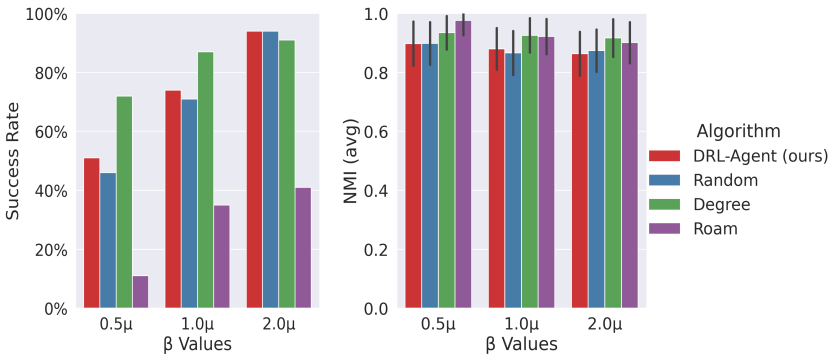


(b) $\tau = 0.5$

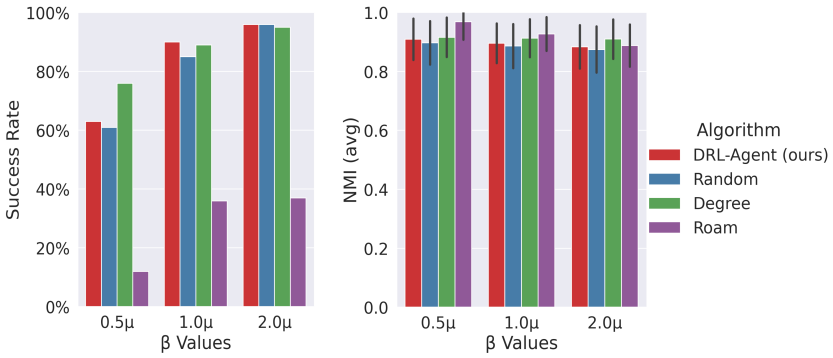


(c) $\tau = 0.8$

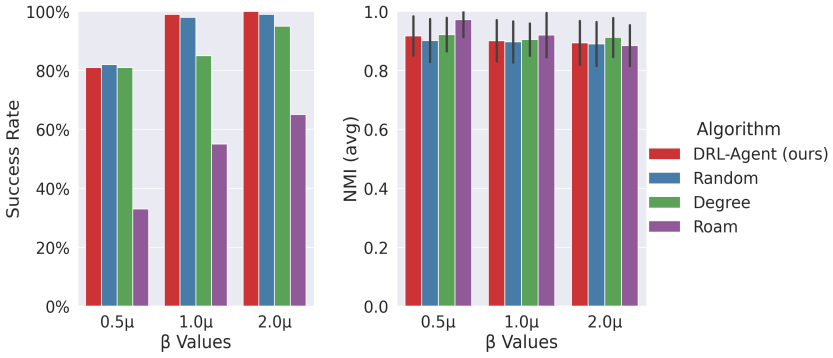
Figure 6.7. Node Deception, SR and NMI with `louv` as $f(\cdot)$, on `words` dataset.



(a) $\tau = 0.3$

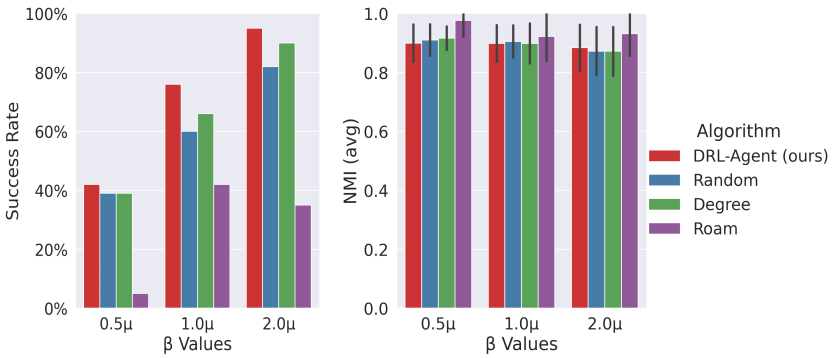


(b) $\tau = 0.5$

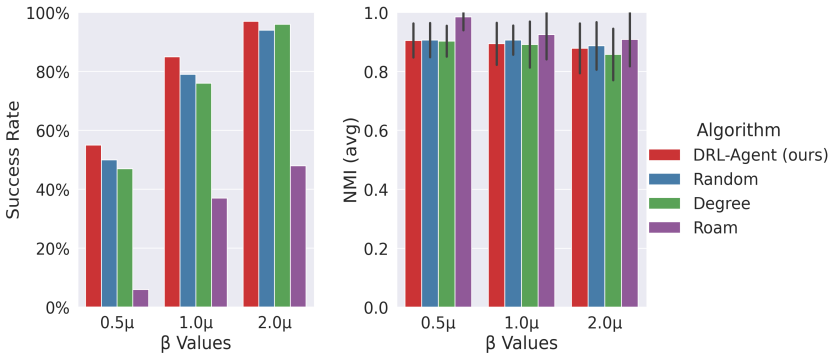


(c) $\tau = 0.8$

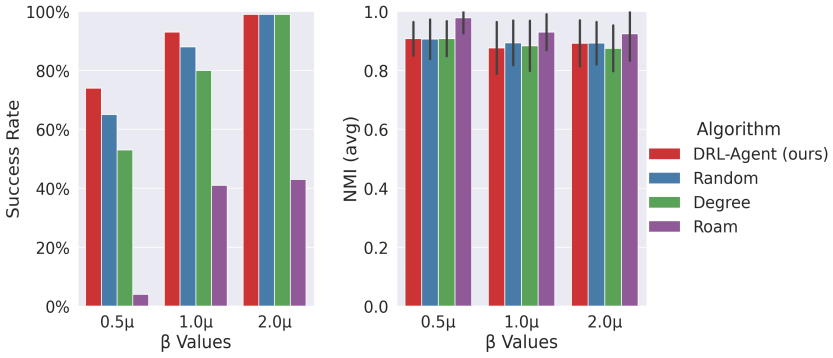
Figure 6.8. Node Deception, SR and NMI with `walk` as $f(\cdot)$, on `words` dataset.



(a) $\tau = 0.3$

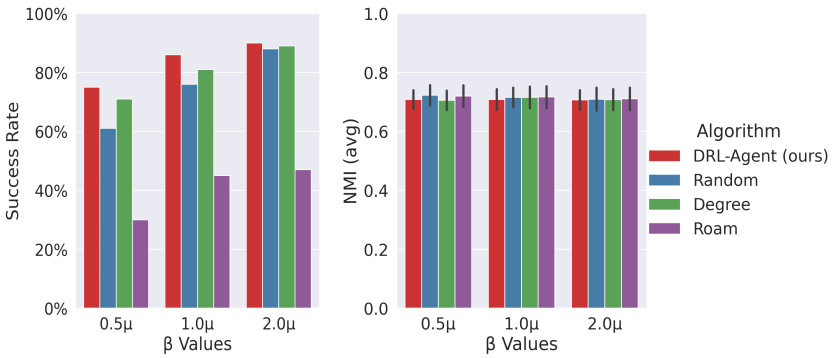


(b) $\tau = 0.5$

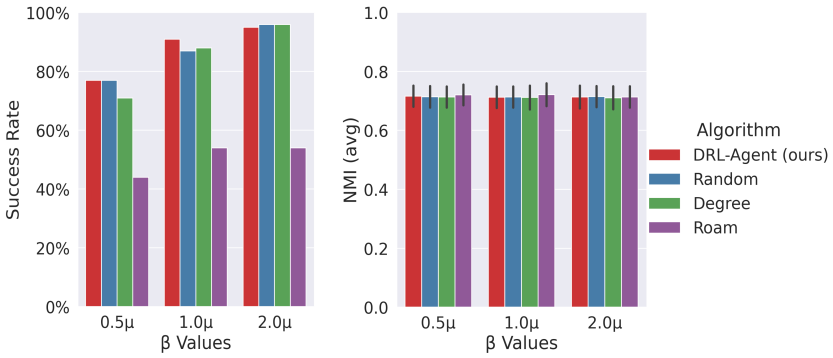


(c) $\tau = 0.8$

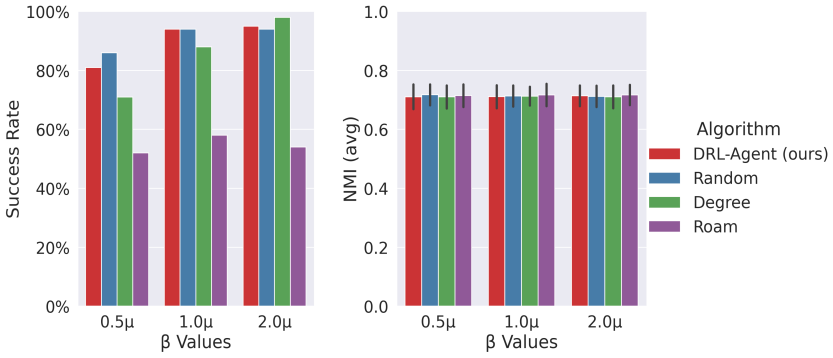
Figure 6.9. Node Deception, SR and NMI with `gre` as $f(\cdot)$, on `vote` dataset.



(a) $\tau = 0.3$

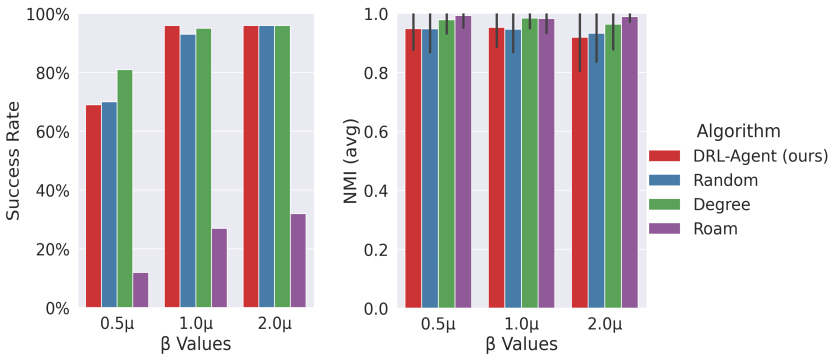


(b) $\tau = 0.5$

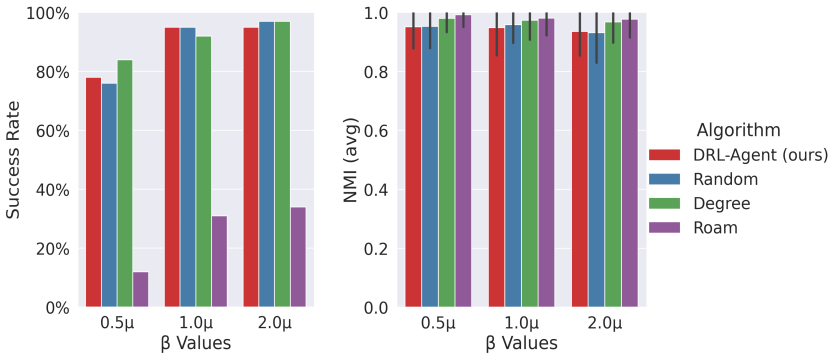


(c) $\tau = 0.8$

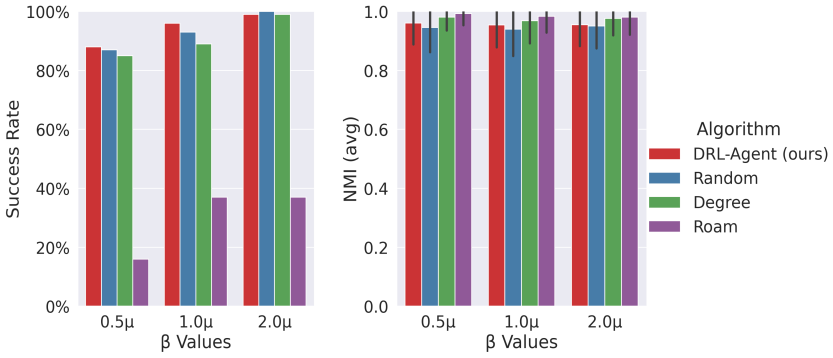
Figure 6.10. Node Deception, SR and NMI with `louv` as $f(\cdot)$, on `vote` dataset.



(a) $\tau = 0.3$



(b) $\tau = 0.5$



(c) $\tau = 0.8$

Figure 6.11. Node Deception, SR and NMI with `walk` as $f(\cdot)$, on `vote` dataset.

6.4.2 Community Deception Task

We evaluate our method against the baselines outlined in Section 6.3, using the same datasets, detection algorithms, and τ constraint values as the node deception task, with the only difference being the variation of β $(1, 3, 5)$ due to its distinct interpretation. For smaller to medium-sized datasets, the number of rewirings is fixed based on [10]. Conversely, for larger datasets, we define β as a ratio of the community size to hide (e.g., $\beta = 0.1|\mathcal{C}_i|$).

As with the node deception task, we examine two distinct scenarios: *symmetric* and *asymmetric*. The former involves our *DRL-Agent* trained on the same modularity-based community detection algorithm as used during deception, i.e., the Greedy algorithm. The latter, instead, assesses the agent's capability to adapt to a different community detection algorithm (Louvain) during deception, despite being still trained on the Greedy algorithm (*transferability*).

In Fig. 6.12 and Fig. 6.13, we report the Deception Score and NMI score for our *DRL-Agent* in the symmetric and asymmetric setup, respectively.

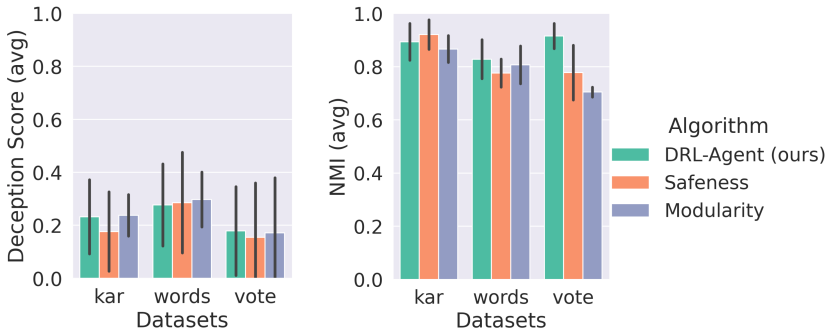


Figure 6.12. $\mathcal{H}$ and NMI using the Greedy algorithm, as $f(\cdot)$, on all datasets with $\tau = 0.3$, and $\beta = 1$, both at training and community deception time (*symmetric*).

Much like our findings in the node deception task, our approach finds the optimal trade-off between these two quality metrics. This observation holds even more significance for the larger dataset (`vote`), where our agent not only achieves the highest Deception Score but also the best NMI score. Moreover, our *DRL-Agent* demonstrates the ability to transfer its performance to a community detection algorithm that differs from the one it was trained on.

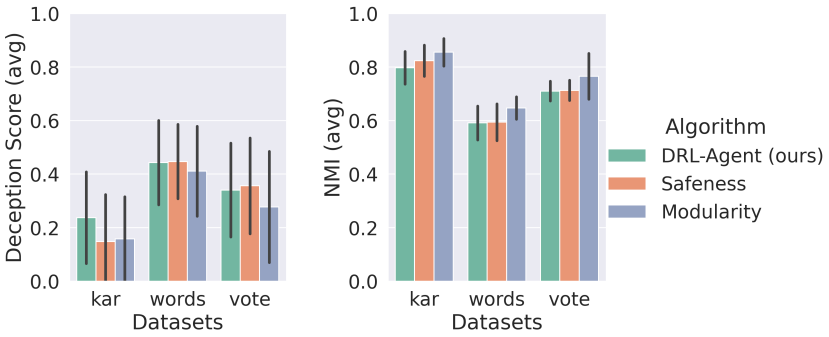
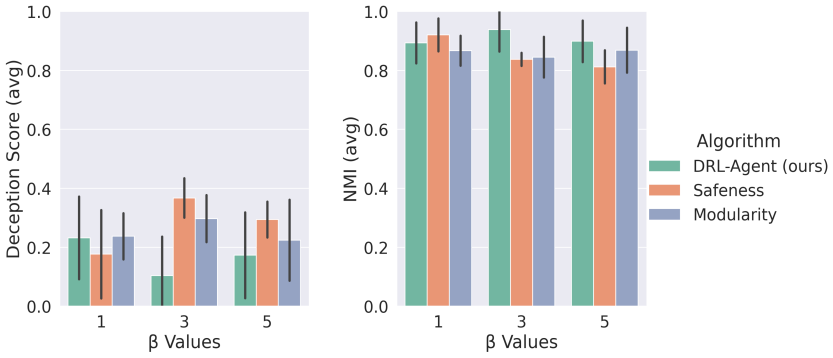


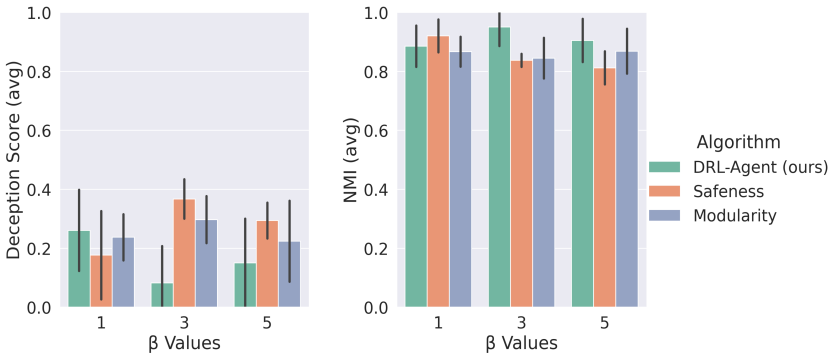
Figure 6.13. $\mathcal{H}$ and NMI using the Greedy algorithm at training time and the Louvain algorithm at community deception time, as $f(\cdot)$, on all datasets with $\tau = 0.3$, and $\beta = 1\mu$ (*asymmetric*).

In summary, across various algorithms and datasets, a common trend emerges: as the τ similarity constraint increases, the performance of the *DRL-Agent* on the community deception task typically decreases. This phenomenon may arise because the agent achieves the node deception objective with fewer rewirings, resulting in fewer actions targeting nodes with high centrality, as they are typically among the first analyzed by the algorithm.

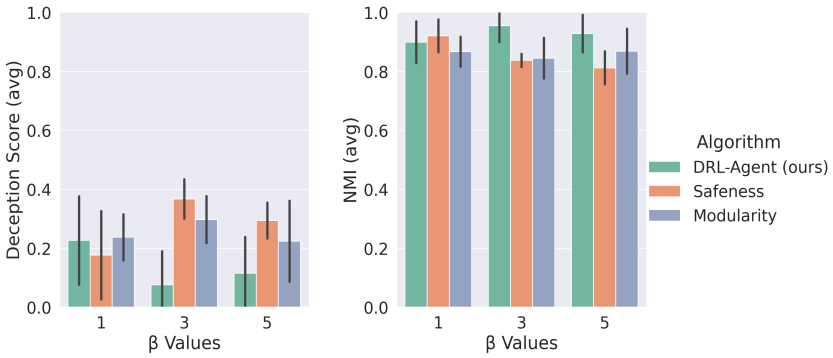
Below are the results of all the tests carried out for the community deception task, using the Greedy algorithm at training time. Generally, it is evident that the *DRL-Agent* achieves better outcomes, for the Deception Score, on larger datasets (`words` and `votes`) when compared to the Modularity-based algorithm, particularly with the use of WalkTrap algorithm as $f(\cdot)$. On the other hand, the algorithm based on Safeness typically obtains superior results than our agent, particularly as β increases.



(a) $\tau = 0.3$

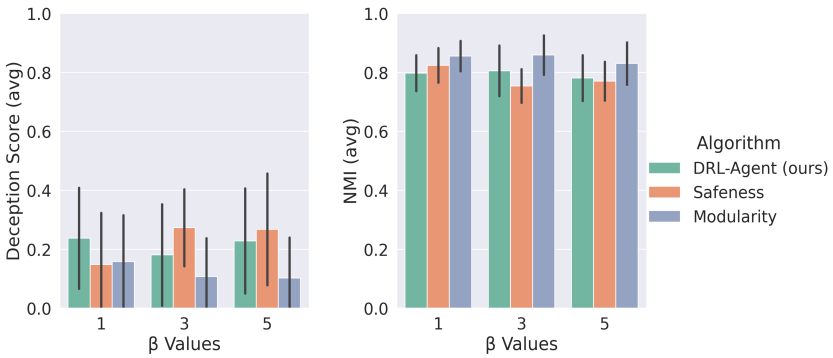


(b) $\tau = 0.5$

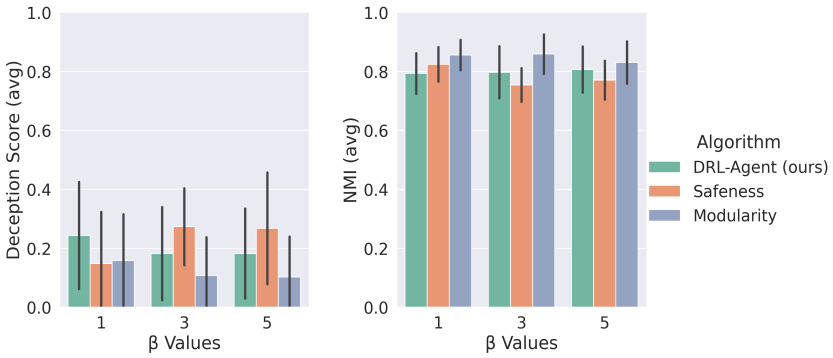


(c) $\tau = 0.8$

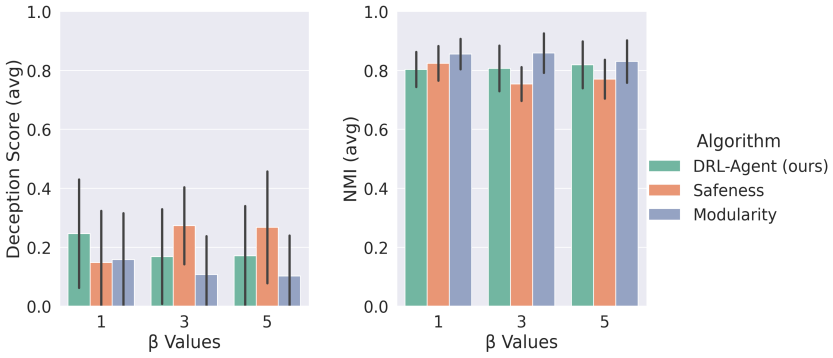
Figure 6.14. Community Deception, $\mathcal{H}$ and NMI with `gre` as $f(\cdot)$, on `kar` dataset.



(a) $\tau = 0.3$

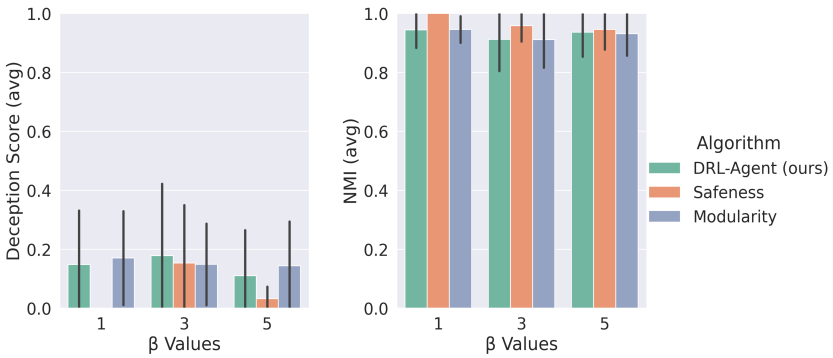


(b) $\tau = 0.5$

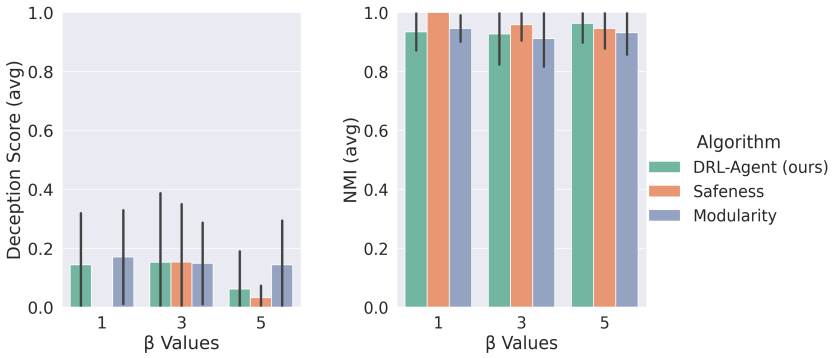


(c) $\tau = 0.8$

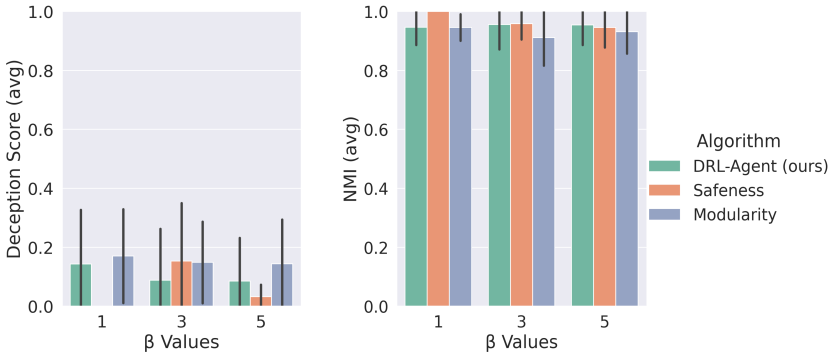
Figure 6.15. Community Deception, $\mathcal{H}$ and NMI with `louv` as $f(\cdot)$, on `kar` dataset.



(a) $\tau = 0.3$

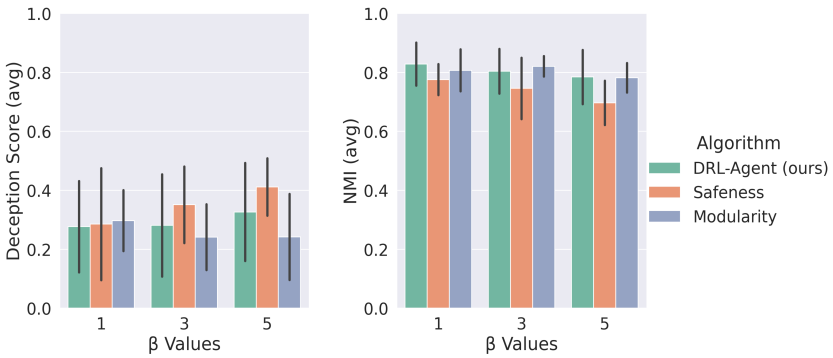


(b) $\tau = 0.5$

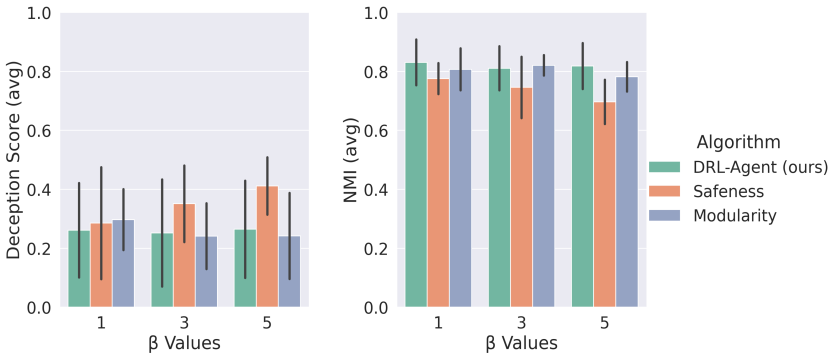


(c) $\tau = 0.8$

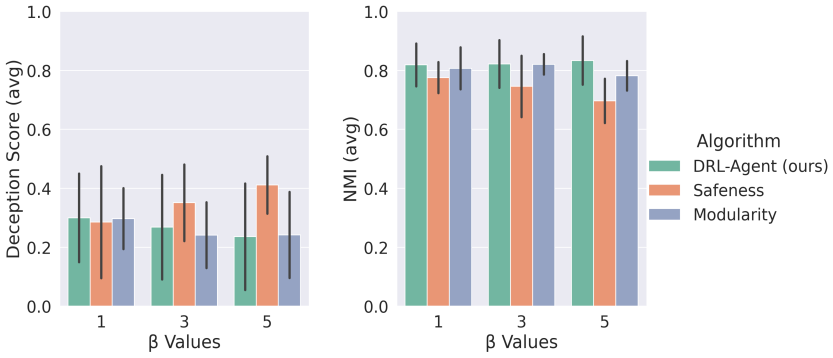
Figure 6.16. Community Deception, $\mathcal{H}$ and NMI with `walk` as $f(\cdot)$, on `kar` dataset.



(a) $\tau = 0.3$

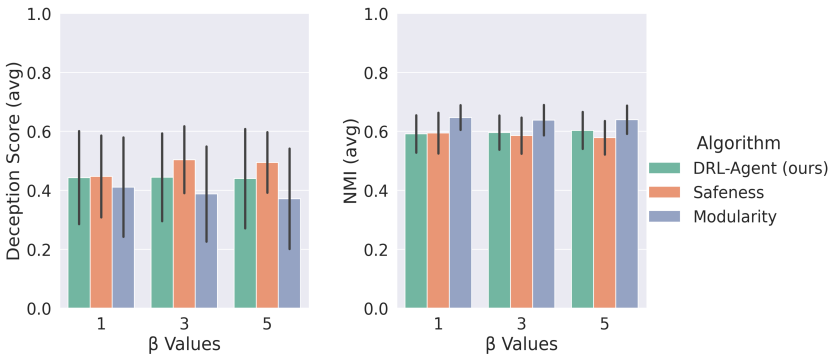


(b) $\tau = 0.5$

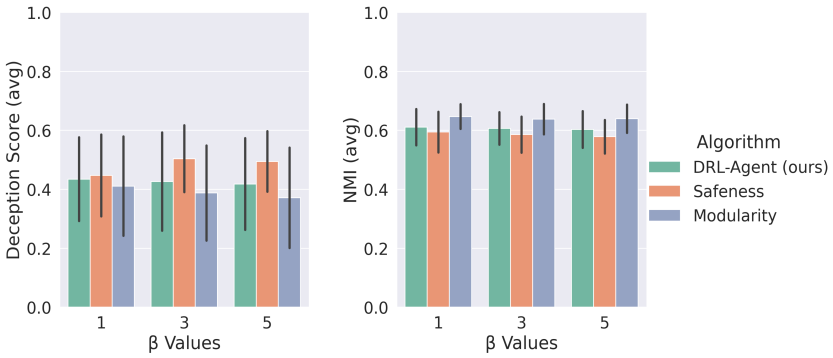


(c) $\tau = 0.8$

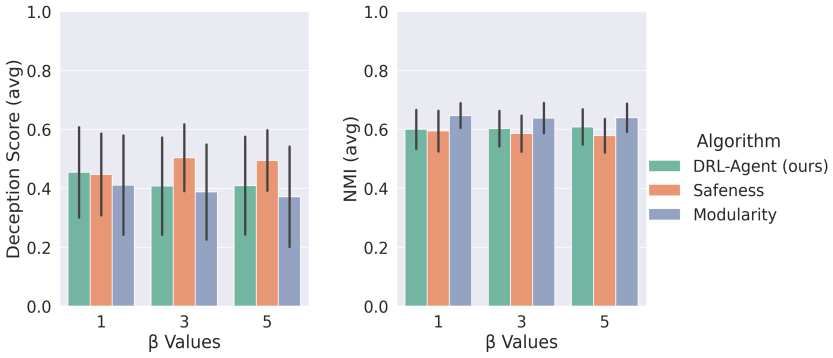
Figure 6.17. Community Deception, $\mathcal{H}$ and NMI with `gre` as $f(\cdot)$, on `words` dataset.



(a) $\tau = 0.3$

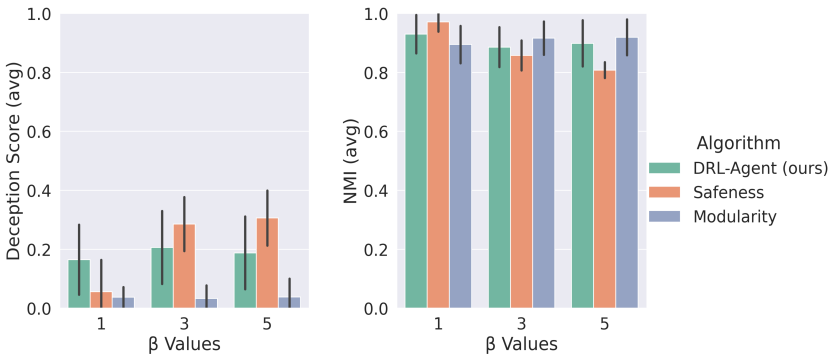


(b) $\tau = 0.5$

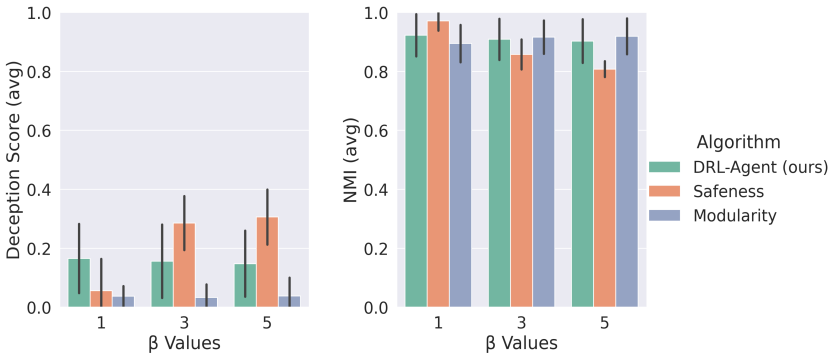


(c) $\tau = 0.8$

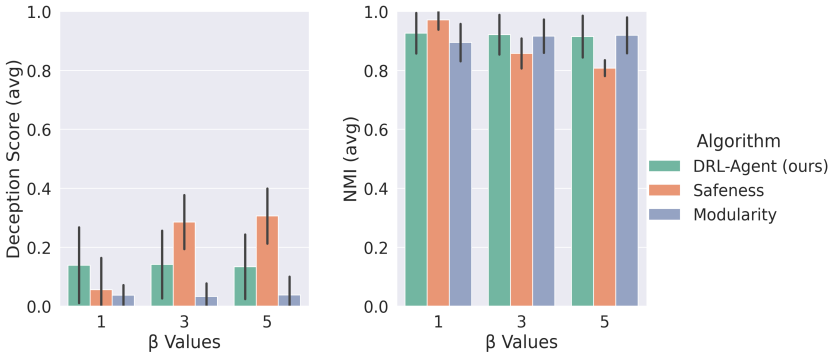
Figure 6.18. Community Deception, $\mathcal{H}$ and NMI with `louv` as $f(\cdot)$, on `words` dataset.



(a) $\tau = 0.3$

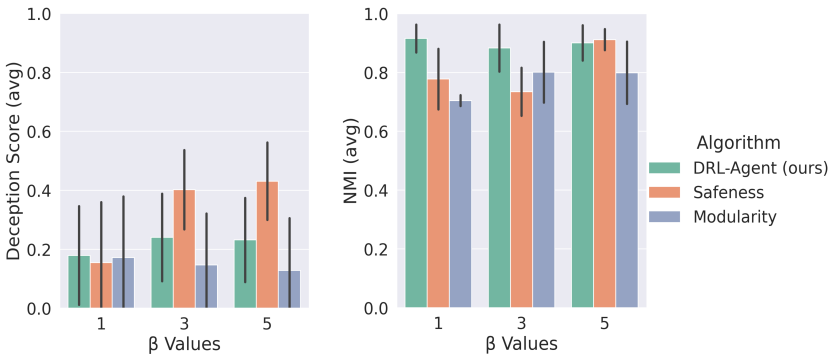


(b) $\tau = 0.5$

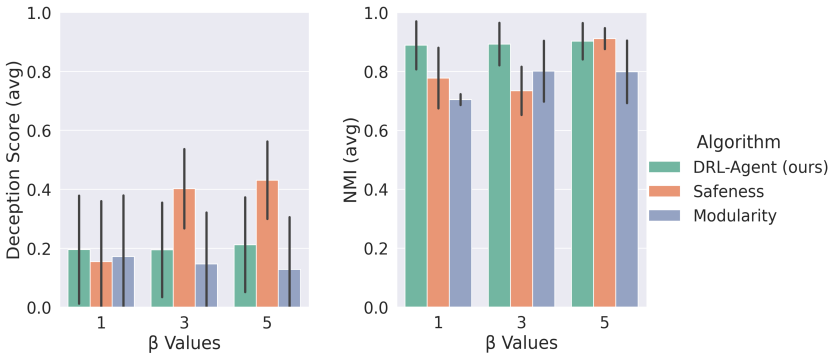


(c) $\tau = 0.8$

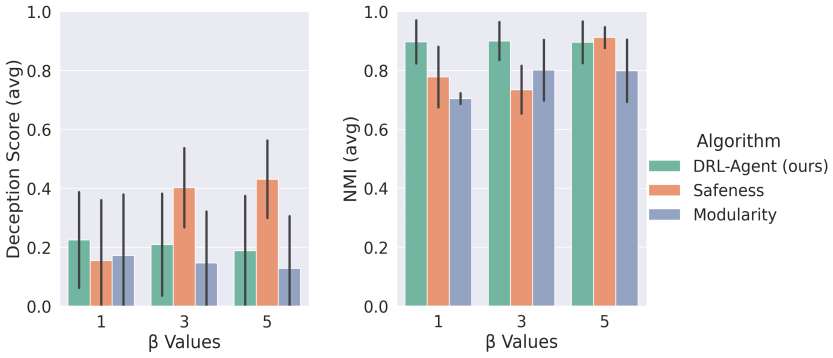
Figure 6.19. Community Deception, $\mathcal{H}$ and NMI with `walk` as $f(\cdot)$, on `words` dataset.



(a) $\tau = 0.3$

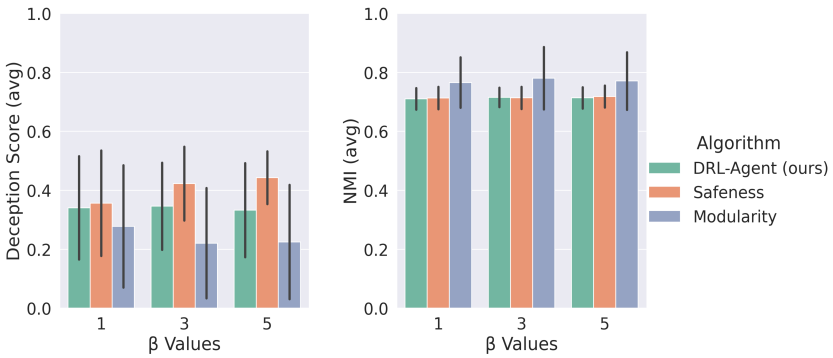


(b) $\tau = 0.5$

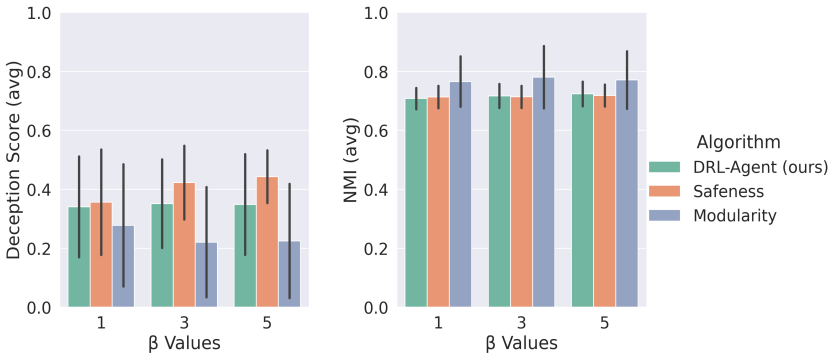


(c) $\tau = 0.8$

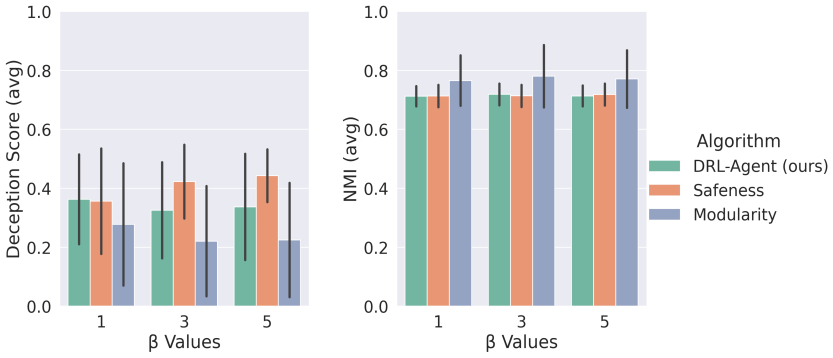
Figure 6.20. Community Deception, $\mathcal{H}$ and NMI with `gre` as $f(\cdot)$, on `vote` dataset.



(a) $\tau = 0.3$

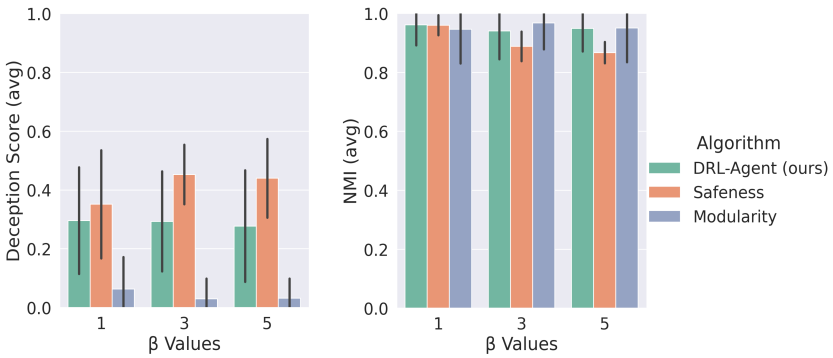


(b) $\tau = 0.5$

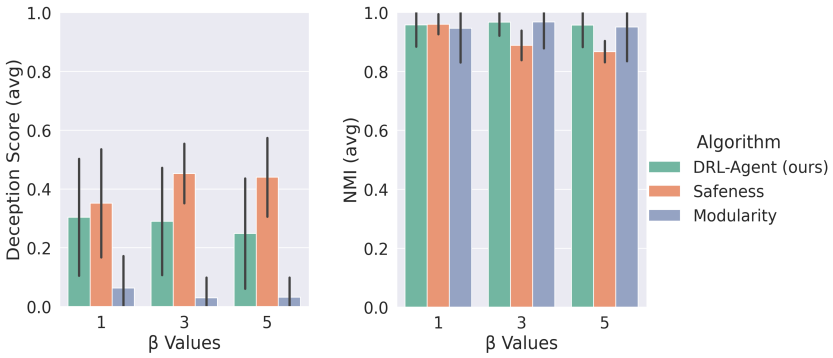


(c) $\tau = 0.8$

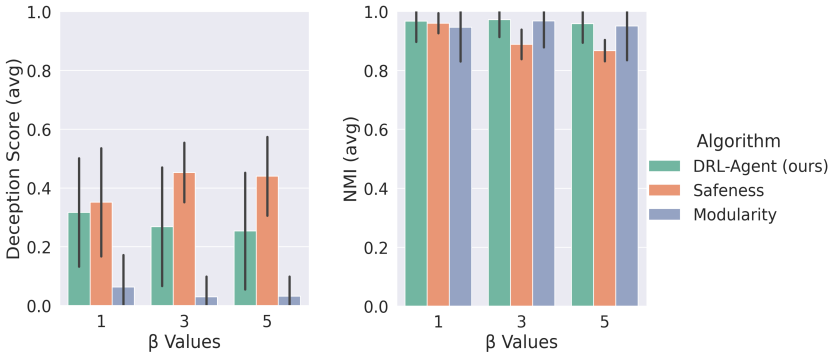
Figure 6.21. Community Deception, $\mathcal{H}$ and NMI with `louv` as $f(\cdot)$, on `vote` dataset.



(a) $\tau = 0.3$



(b) $\tau = 0.5$



(c) $\tau = 0.8$

Figure 6.22. Community Deception, $\mathcal{H}$ and NMI with `walk` as $f(\cdot)$, on `vote` dataset.

6.5 Parameter Sensitivity

The effectiveness of our *DRL-Agent* relies on two critical parameters: (*i*) the similarity threshold (τ) used to determine whether the node deception goal has been achieved or not, and (*ii*) the budget (β) to limit the effort – i.e., graph modifications – performed to achieve the goal. In this section, we analyze their impact. Specifically, in Table 6.2 and Table 6.3, we explore how the SR and NMI metrics for the node deception task are influenced by varying the values of τ and β, while keeping the detection algorithm $f(\cdot)$ and dataset fixed.

Table 6.2. SR for node detection task, using the Greedy community detection algorithm on `words` dataset.

τ	β	**Node Deception Algorithm**			
		DRL-Agent (ours)	*Random*	*Degree*	*Roam*
0.3	$\frac{1}{2}\mu$	**53**%	48%	51%	11%
	1μ	**73**%	60%	63%	29%
	2μ	**92**%	83%	84%	34%
0.5	$\frac{1}{2}\mu$	**61**%	59%	**61**%	13%
	1μ	**76**%	65%	73%	45%
	2μ	**96**%	90%	84%	47%
0.8	$\frac{1}{2}\mu$	**78**%	77%	**78**%	18%
	1μ	**94**%	93%	**94**%	66%
	2μ	**100**%	**100**%	98%	79%

Table 6.3. NMI (avg. ± std.) for node detection task, using the Greedy community detection algorithm on `words` dataset.

τ	β	**Node Deception Algorithm**			
		DRL-Agent (ours)	*Random*	*Degree*	*Roam*
0.3	$\frac{1}{2}\mu$	0.79 ± 0.08	0.80 ± 0.07	0.80 ± 0.10	0.95 ± 0.07
	1μ	0.77 ± 0.09	0.77 ± 0.09	0.79 ± 0.09	0.81 ± 0.13
	2μ	0.79 ± 0.08	0.78 ± 0.08	0.80 ± 0.08	0.76 ± 0.14
0.5	$\frac{1}{2}\mu$	0.81 ± 0.07	0.79 ± 0.08	0.81 ± 0.09	0.93 ± 0.09
	1μ	0.78 ± 0.07	0.77 ± 0.08	0.78 ± 0.08	0.77 ± 0.12
	2μ	0.75 ± 0.09	0.75 ± 0.08	0.78 ± 0.08	0.75 ± 0.12
0.8	$\frac{1}{2}\mu$	0.80 ± 0.08	0.79 ± 0.08	0.79 ± 0.10	0.94 ± 0.10
	1μ	0.79 ± 0.07	0.80 ± 0.07	0.80 ± 0.08	0.78 ± 0.13
	2μ	0.79 ± 0.07	0.78 ± 0.08	0.79 ± 0.08	0.74 ± 0.11

These results reinforce the previously mentioned findings: our method consistently achieves a superior balance between SR and NMI. It substantially outperforms baseline approaches in the former without overly compromising the latter.

Chapter 7

Ethical Implications

In this section, we will delve into the ethical considerations surrounding the use of community membership hiding algorithms. These algorithms, as explored in this work, can serve as potent tools for safeguarding the privacy of social network users and have the potential to impact various aspects of society.

7.1 Possible Benefits

One of the primary motivations for the development of community membership hiding algorithms is to protect the privacy of individuals in online social networks. By enabling users to hide their community affiliations, these methods empower individuals to control the dissemination of their personal information and interactions within digital communities. This can be particularly valuable for vulnerable groups, such as journalists or opposition activists, who operate in regions governed by authoritarian regimes. Shielding their online identities can help protect them from potential harm and persecution.

Community membership hiding techniques can also be harnessed to combat online criminal activities. By strategically modifying network connections, these methods can be used to infiltrate espionage agents or disrupt communications among malicious users. Law enforcement agencies and cybersecurity experts can employ such techniques to identify and neutralize threats posed by criminal organizations, terrorist groups, and other malicious entities. This application can enhance public safety and national security.

7.2 Potential Drawbacks

However, the use of node-hiding techniques is not without ethical concerns and potential drawbacks:

1. *Privacy Violations:* While community membership hiding can protect the privacy of individuals, it can also be exploited for unethical purposes. Malicious actors may use these methods to conceal their illicit or criminal activities, making it challenging for law enforcement agencies to track and apprehend wrongdoers.

2. *Potential for Abuse:* There is a risk that community membership hiding algorithms may be misused for purposes that infringe upon the rights and safety of others. For example, individuals with harmful intentions could use these techniques to impersonate innocent users or evade legitimate network analysis efforts.

3. *Unintended Consequences:* Modifying network structures to hide community memberships may have unintended consequences, such as disrupting the integrity of online communities or inadvertently facilitating cyberbullying or harassment.

4. *Bias and Discrimination:* The deployment of these algorithms should be closely monitored to prevent potential biases and discrimination. If not used responsibly, they may disproportionately impact certain groups or individuals, potentially exacerbating existing inequalities.

7.3 Responsible Development and Use

To address these ethical concerns, it is crucial for researchers, developers, and policymakers to adopt a responsible approach when developing and deploying community membership hiding algorithms. This includes:

1. *Transparency:* Developers should be transparent about the capabilities and limitations of these algorithms, ensuring that users are aware of the potential risks and benefits.

2. *Ethical Guidelines:* Establishing clear ethical guidelines and regulations for the use of node-hiding techniques can help prevent misuse and abuse.

3. *Oversight:* There should be mechanisms for oversight and accountability to monitor the responsible use of these algorithms and address any ethical violations.

4. *Interdisciplinary Collaboration:* Collaboration between experts from various fields, including computer science, ethics, law, and sociology, can help address the multifaceted ethical challenges associated with community membership hiding.

In conclusion, while community membership hiding algorithms offer significant benefits in terms of privacy protection and countering criminal activities, they also present ethical dilemmas that require careful consideration. Responsible development, regulation, and ethical oversight are essential to ensure that these techniques are used for the greater good while minimizing their potential for harm and misuse.

Chapter 8

Conclusion and Future Work

In this concluding section, we will summarize the key achievements and contributions of this work in addressing the challenging problem of community membership hiding, as well as outline potential avenues for future research and development.

8.1 Recap of Achievements

This work has successfully tackled the intricate problem of community membership hiding, which involves strategically modifying the structural characteristics of a network graph to prevent specific nodes from being detected by community detection algorithms. Our approach is rooted in a novel formulation of this problem as a constrained counterfactual graph objective, and we harnessed the power of deep reinforcement learning to provide a solution.

Throughout our research journey, we conducted extensive experiments to rigorously evaluate the effectiveness of our proposed method. We focused on two distinct tasks: node and community deception. The results of these experiments have been immensely promising and have demonstrated that our approach excels in achieving a balance between the deception goal and the cost incurred due to graph modifications when compared to existing baseline methods in both tasks.

8.2 Future Directions

While this work represents a significant step forward in the realm of community membership hiding, there are several exciting avenues for future research and development that can build upon our contributions:

1. *Scalability to Large-scale Graphs:* Our experiments thus far have mainly been conducted on relatively small-scale graphs. Future work should focus on scaling up our approach to handle large-scale networks commonly encountered in real-world applications. This involves optimizing the efficiency of our algorithms to process and modify graphs with millions of nodes and edges.

2. *Integration of Node Features:* Currently, our method primarily focuses on modifying the structural properties of the graph. In future research, we aim to

extend our approach to incorporate modifications to node features alongside structural changes. This will provide a more comprehensive solution to the community membership hiding problem, where both network topology and node attributes play crucial roles.

3. *Robustness and Adversarial Attacks:* Investigating the robustness of our method against adversarial attacks and exploring potential countermeasures is essential. Future work can delve into designing algorithms that can withstand sophisticated attempts to unveil hidden nodes or communities.

4. *Real-world Applications:* Applying our techniques to practical, real-world scenarios is of paramount importance. Research should aim to identify specific domains where community membership hiding has practical implications and apply our method to solve real-world problems.

5. *Interdisciplinary Collaboration:* Collaborations between computer scientists, sociologists, and experts from other domains can enrich our understanding of community detection and community membership hiding. Future research should foster interdisciplinary partnerships to tackle these challenges from multiple perspectives.

6. *Ethical Considerations:* As with any technology that involves hiding or manipulating information, ethical considerations must be at the forefront. Research should continue to explore the ethical implications of community membership hiding and develop guidelines for responsible use.

In conclusion, this work has laid a solid foundation for addressing the intricate problem of community membership hiding through the innovative use of deep reinforcement learning and counterfactual graph objectives. The future directions outlined above provide a roadmap for further advancements in this field, promising to make our communities and networks more secure and resilient in the face of evolving challenges.

Bibliography

[1] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008. doi: 10.1088/1742-5468/2008/10/P10008. URL https://dx.doi.org/10.1088/1742-5468/2008/10/P10008.

[2] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008. doi: 10.1088/1742-5468/2008/10/p10008. URL https://doi.org/10.1088%2F1742-5468%2F2008%2F10%2Fp10008.

[3] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20:172–188, 2008. URL https://api.semanticscholar.org/CorpusID:150684.

[4] Alina Campan, Yasmeen Alufaisan, and Traian Marius Truta. Preserving communities in anonymized social networks. *Trans. Data Privacy*, 8(1):55–87, dec 2015. ISSN 1888-5063.

[5] Tanmoy Chakraborty, Sriram Srinivasan, Niloy Ganguly, Animesh Mukherjee, and Sanjukta Bhowmick. Permanence and community structure in complex networks, 2016.

[6] Xianyu Chen, Zhongyuan Jiang, Hui Li, Jianfeng Ma, and Philip S. Yu. Community hiding by link perturbation in social networks. *IEEE Transactions on Computational Social Systems*, 8(3):704–715, 2021. doi: 10.1109/TCSS.2021.3054115.

[7] Ziheng Chen, Fabrizio Silvestri, Jia Wang, He Zhu, Hongshik Ahn, and Gabriele Tolomei. ReLAX: Reinforcement Learning Agent Explainer for Arbitrary Predictive Models. In Mohammad Al Hasan and Li Xiong, editors, *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 252–261. ACM, 2022. doi: 10.1145/3511808.3557429. URL https://doi.org/10.1145/3511808.3557429.

[8] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945. doi: https://doi.org/10.2307/1932409.

URL https://esajournals.onlinelibrary.wiley.com/doi/abs/10.2307/1932409.

[9] Claire Donnat and Susan Holmes. Tracking network dynamics: a survey of distances and similarity metrics, 2018.

[10] Valeria Fionda and Giuseppe Pirrò. Community deception or: How to stop fearing community detection algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 30(4):660–673, 2018. doi: 10.1109/TKDE.2017.2776133.

[11] Santo Fortunato. Community Detection in Graphs. *Physics Reports*, 486 (3):75–174, 2010. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2009.11.002. URL https://www.sciencedirect.com/science/article/pii/S0370157309002841.

[12] Daniele Gammelli, Kaidi Yang, James Harrison, Filipe Rodrigues, Francisco C. Pereira, and Marco Pavone. Graph neural network reinforcement learning for autonomous mobility-on-demand systems, 2021.

[13] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002. doi: 10.1073/pnas.122653799. URL https://www.pnas.org/doi/abs/10.1073/pnas.122653799.

[14] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks, 2016.

[15] Paul Jaccard. The distribution of the flora in the alpine zone.1. *New Phytologist*, 11(2):37–50, 1912. doi: https://doi.org/10.1111/j.1469-8137.1912.tb05611.x. URL https://nph.onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-8137.1912.tb05611.x.

[16] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Dongxiao He, Jia Wu, Philip S. Yu, and Weixiong Zhang. A survey of community detection approaches: From statistical modeling to deep learning, 2021.

[17] N. Kalaichelvi and K. S. Easwarakumar. A comprehensive survey on community deception approaches in social networks. In Erich J. Neuhold, Xavier Fernando, Joan Lu, Selwyn Piramuthu, and Aravindan Chandrabose, editors, *Computer, Communication, and Signal Processing*, pages 163–173, Cham, 2022. Springer International Publishing. ISBN 978-3-031-11633-9.

[18] Arzum Karataş and Serap Şahin. Application Areas of Community Detection: A Review. In *Proc. of the International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pages 65–70, 2018. doi: 10.1109/IBIGDELFT.2018.8625349.

[19] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[20] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. 12, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[21] J. Kreer. A question of terminology. *IRE Transactions on Information Theory*, 3(3):208–208, 1957. doi: 10.1109/TIT.1957.1057418.

[22] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, mar 2009. doi: 10.1088/1367-2630/11/3/033015. URL https://doi.org/10.1088%2F1367-2630%2F11%2F3%2F033015.

[23] Ana Lucic, Maartje A. ter Hoeve, Gabriele Tolomei, Maarten de Rijke, and Fabrizio Silvestri. CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, volume 151 of *Proceedings of Machine Learning Research*, pages 4499–4511. PMLR, 2022. URL https://proceedings.mlr.press/v151/lucic22a.html.

[24] Vijaymeena M.K and Kavitha K. A survey on similarity measures in text mining. 2016. URL https://api.semanticscholar.org/CorpusID:62118842.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. URL https://api.semanticscholar.org/CorpusID:205242740.

[26] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016.

[27] Mohammad Javad Mosadegh and Mehdi Behboudi. Using Social Network Paradigm for Developing a Conceptual Framework in CRM. *Australian Journal of Business and Management Research*, 1(4):63, 2011.

[28] Shishir Nagaraja. The impact of unlinkability on adversarial community detection: Effects and countermeasures. In Mikhail J. Atallah and Nicholas J. Hopper, editors, *Privacy Enhancing Technologies*, pages 253–272, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-14527-8.

[29] Mark E. J. Newman. Finding Community Structure in Networks Using the Eigenvectors of Matrices. *Phys. Rev. E*, 74:036104, Sep 2006. doi: 10.1103/PhysRevE.74.036104. URL https://link.aps.org/doi/10.1103/PhysRevE.74.036104.

[30] Mark E. J. Newman. Modularity and Community Structure in Networks. *Proceedings of the National Academy of Sciences*, 103 (23):8577–8582, June 2006. doi: 10.1073/pnas.0601602103. URL http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=retrieve&db=pubmed&list_uids=16723398&dopt=AbstractPlus.

[31] Mingshuo Nie, Dongming Chen, and Dongqi Wang. Reinforcement learning on graphs: A survey, 2023.

[32] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. pages 284–293, 2005.

[33] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, 76:036106, Sep 2007. doi: 10.1103/PhysRevE.76.036106. URL https://link.aps.org/doi/10.1103/PhysRevE.76.036106.

[34] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Phys. Rev. E*, 74:016110, Jul 2006. doi: 10.1103/PhysRevE.74.016110. URL https://link.aps.org/doi/10.1103/PhysRevE.74.016110.

[35] Peter Ronhovde and Zohar Nussinov. Multiresolution community detection for megascale networks by information-based replica correlations. *Physical Review E*, 80(1), jul 2009. doi: 10.1103/physreve.80.016109. URL https://doi.org/10.1103%2Fphysreve.80.016109.

[36] XingMao Ruan, YueHeng Sun, Bo Wang, and Shuo Zhang. The community detection of complex networks based on markov matrix spectrum optimization. In *2012 International Conference on Control Engineering and Communication Technology*, pages 608–611, 2012. doi: 10.1109/ICCECT.2012.192.

[37] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[38] Gabriele Tolomei and Fabrizio Silvestri. Generating Actionable Interpretations from Ensembles of Decision Trees. *IEEE Trans. Knowl. Data Eng.*, 33(4): 1540–1553, 2021. doi: 10.1109/TKDE.2019.2945326. URL https://doi.org/10.1109/TKDE.2019.2945326.

[39] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 465–474. ACM, 2017. doi: 10.1145/3097983.3098039. URL https://doi.org/10.1145/3097983.3098039.

[40] Marcin Waniek, Tomasz P. Michalak, Michael J. Wooldridge, and Talal Rahwan. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139–147, jan 2018. doi: 10.1038/s41562-017-0290-3. URL https://doi.org/10.1038%2Fs41562-017-0290-3.

[41] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

[42] Joyce Jiyoung Whang, David F. Gleich, and Inderjit S. Dhillon. Overlapping community detection using neighborhood-inflated seed expansion, 2015.

[43] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 2004. URL https://api.semanticscholar.org/CorpusID:19115634.

[44] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, jan 2021. doi: 10.1109/tnnls.2020.2978386. URL https://doi.org/10.1109%2Ftnnls.2020.2978386.

[45] Yunpeng Zhao, Elizaveta Levina, and Ji Zhu. Consistency of Community Detection in Networks Under Degree-Corrected Stochastic Block Models. *The Annals of Statistics*, 40(4):2266–2292, 2012. ISSN 00905364, 21688966. URL http://www.jstor.org/stable/41806535.