Evading Community Detection via Counterfactual Neighborhood Search

Andrea Bernini
Sapienza University of Rome
Italy
andreabe99@gmail.com

Fabrizio Silvestri
Sapienza University of Rome
Italy
fsilvestri@diag.uniroma1.it

Gabriele Tolomei
Sapienza University of Rome
Italy
tolomei@di.uniroma1.it

Abstract

Community detection techniques are useful 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 tastes or preferences. Therefore, some users may wish to preserve their anonymity and opt out of community detection for various reasons, such as affiliation with political or religious organizations, without leaving the platform. 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. Extensive experiments demonstrate that our method outperforms existing baselines, striking the best balance between accuracy and cost.

CCS Concepts

• **Human-centered computing** → **Social network analysis**; • **Security and privacy** → **Social network security and privacy**; • **Theory of computation** → **Reinforcement learning**; • **Computing methodologies** → **Markov decision processes**.

Keywords

Community detection; Community membership hiding; Node hiding; Node deception; Counterfactual graph

ACM Reference Format:
Andrea Bernini, Fabrizio Silvestri, and Gabriele Tolomei. 2024. Evading Community Detection via Counterfactual Neighborhood Search. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3637528.3671896

1 Introduction

Identifying *communities* is crucial for understanding the intricacies of complex graph structures like social networks [9]. This is typically achieved by *community detection* algorithms, which group nodes 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 [17]. For instance, the insights gained from accurately identifying communities can significantly impact business strategies, leading to better monetization opportunities through targeted advertising [27].

However, community detection algorithms also raise concerns on individual privacy and data protection. In some cases, certain nodes within the network might prefer not to be identified as part of a particular community. These nodes could be associated with sensitive or private groups (e.g., political or religious organizations) or may wish to protect their anonymity for personal reasons. An option for these users would be to leave the platform, but such a decision might be too extreme. A more flexible approach would allow users to opt out of community detection while staying on the platform. This strategy strikes the optimal balance between preserving privacy and maximizing the utility of community detection.

Motivated by the need above, in this paper, we address the intriguing challenge of *community membership hiding*.

Drawing inspiration from *counterfactual reasoning* [38, 39], particularly in the realm of graph data [23], we aim to provide users with personalized recommendations that safeguard their anonymity from community detection. To illustrate this with an example, consider a scenario in which a social network is equipped with a tool that offers guidance to its users on how to modify their connections to prevent them from being recognized as members of a particular community. For instance, a recommendation could take the form: "*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 [5], we cast this problem into a Markov decision process (MDP) framework and we propose a deep reinforcement learning (DRL) approach 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 can be unknown or undisclosed. When given a target node within a community, we aim 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 as *community membership hiding task*,

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671896

and we consider it successful when a predefined similarity threshold between the original community and the new community containing the target node is met. It is worth noting that while related, this task differs significantly from the *community deception task* explored in existing literature [8]. Specifically, community deception aims to hide an entire community from community detection algorithms by rewiring connections of *some* members within the target community. However, the community deception task, as defined by Fionda and Pirrò [8], does not have a binary outcome, unlike our community membership hiding goal. Instead, the authors introduce a smooth measure, the *Deception Score*, that combines three criteria for effective community masking: reachability, spreadness, and hiding. While it might seem plausible to extend our method for community deception by running multiple membership hiding tasks for *each* node in the target community, this straightforward strategy might be too aggressive due to our more stringent (i.e., binary) definition of deception goal. A more nuanced approach could involve leveraging the structural properties of each node in the community to mask (e.g., their degree) to cleverly select target nodes for membership hiding. Further exploration of this strategy is left for future research.

We validate our approach on five real-world datasets, and we demonstrate that it outperforms existing baselines using standard quality metrics. Notably, our method maintains its effectiveness even when used in conjunction with a community detection algorithm that was not seen during the training phase. We call this key property: *transferability*.

Our main contributions 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.
- We publicly release both the source code and the data utilized in this study to encourage reproducibility.[1]

The remainder of this paper 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. Section 7 discusses the implementation challenges along with the potential security and ethical impact of our method. Finally, we conclude in Section 8.

2 Related Work

Community Detection. Community detection algorithms are essential tools in network analysis, aiming to uncover densely connected groups of nodes within a graph. They apply to various domains, such as social network analysis, biology, and economics.

These algorithms can be broadly categorized into two types: non-overlapping and overlapping community detection.

Non-overlapping community detection assigns each node to a single community, employing various techniques such as Modularity Optimization [1], Spectrum Analysis [35], Random Walk [31], or Label Propagation [32]. On the other hand, overlapping community detection seeks to represent better real-world networks where nodes can belong to multiple communities. Mature methods have been developed for this purpose, including NISE (Neighborhood-Inflated Seed Expansion) [42] and techniques based on minimizing the Hamiltonian of the Potts model [34]. For a more comprehensive overview, we recommend consulting the work by Jin et al. [15].

Community Deception. As already discussed in Section 1 above, community deception is closely related to the community membership hiding problem we investigate in this work. Somehow, it is a specialization of community membership hiding, where the objective is to hide an entire community from community detection algorithms. In essence, this should involve performing multiple membership hiding tasks, one for each node within the community to be masked. However, this is a pretty raw simplification of the process since each node hiding task, for how we define it below, may, once solved, already achieve a satisfactory degree of community hiding (i.e., obtain a high Deception Score, according to [8]), indirectly concealing multiple nodes within the same community.

Community deception can serve various purposes, such as preserving the anonymity of sensitive groups of individuals in online monitoring scenarios, like social networks, or aiding public safety by identifying online criminal activities. Yet, these techniques also pose risks, as malicious actors can use them to evade detection algorithms and operate covertly, potentially violating the law.

Several techniques exist for hiding communities, such as those based on the concept of Modularity. Notable examples in this category include the approach proposed by Nagaraja [28], the DICE algorithm introduced by Waniek et al. [41], and the method developed by Fionda and Pirrò [8]. Other deception techniques are founded on the idea of Safeness, as defined by Fionda and Pirrò [8] and further explored by Chen et al. [4], as well as the notion of Permanence used by Mittal et al. [24]. An exhaustive summary of these methods is provided by Kalaichelvi and Easwarakumar [16].

3 Background and Preliminaries

In this section, we briefly review the well-known community detection problem and utilize its definition as the basis for formulating 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 $\boldsymbol{x}_u \in \mathbb{R}^p$. Furthermore, the underlying link structure of $\mathcal{G}$ is represented using a binary adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$, where $\boldsymbol{A}_{u,v} = 1$ 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 [43].

[1] https://github.com/AndreaBe99/community_membership_hiding

More formally, in this work, we adhere to the definition widely used in the literature [8, 20, 22, 24], 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 $\{C_1, \ldots, C_k\}$ as output, i.e., $f(\mathcal{G}) = \{C_1, \ldots, C_k\}$, where k is usually unknown. Within this framework, every node $u \in \mathcal{V}$ is assigned to *exactly one* community. However, this can be extended to accommodate scenarios with overlapping communities, where each node can belong to multiple clusters. Indeed, to represent node-community assignments, we can use a k-dimensional stochastic vector $\boldsymbol{c}_u$, where $c_{u,i} = P(u \in C_i)$ measures the probability that node u belongs to community 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 define the index of the community to which a specific node u belongs based on the outcome of $f(\mathcal{G})$. Note that in the case of hard node partitioning, the vector $\boldsymbol{c}_u$ has *only one* non-zero entry, which equals 1. 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 [31], Greedy [2], InfoMap [3], Label Propagation [32], Leading Eigenvectors [29], Edge-Betweeness [12], SpinGlass [33]. 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."

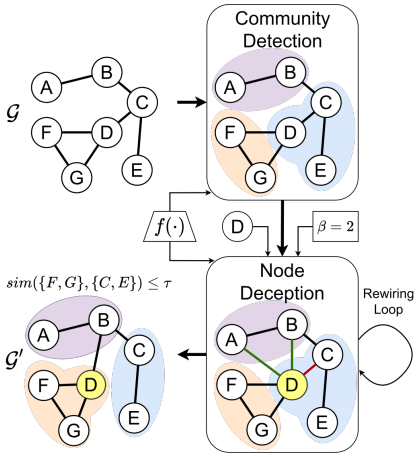

Figure 1: Given a graph $\mathcal{G}$, a node u (in this case $u = D$), a budget of actions β, and the set of communities identified by the community detection algorithm $f(\cdot)$ (including the community C_i to which the node belongs), *community membership hiding* consists of adding inter-community 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 Community Membership Hiding

In a nutshell, community membership hiding seeks to allow a target node in a graph to avoid being identified as a member of a specific node cluster, as determined by a community detection algorithm. This objective is achieved by suggesting to the node in question how to strategically modify its connections with other nodes. Our primary focus is to change the graph's structure, represented by the adjacency matrix. While altering node features holds potential interest, that aspect is reserved for future work.

Assumptions: Whoever runs our community membership hiding algorithm must be able to *execute* the community detection algorithm $f(\cdot)$ even without access to its internal logic and possesses *full knowledge* of the graph. Nevertheless, our method might work under more relaxed conditions (e.g., partial graph knowledge), as discussed in Section 7.1. We illustrate our approach in Figure 1.

4.1 Problem Formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph and $f(\mathcal{G}) = \{C_1, \ldots, C_k\}$ 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 $C_i \in f(\mathcal{G})$ – i.e., $i_u^* = i$ – denoted as $u \in C_i$. The aim of community membership hiding is to formulate a function $h_{\boldsymbol{\theta}}(\cdot)$, parameterized 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 C_i. This might lead to formulating various objectives depending on how we define community membership hiding. Let us consider the scenario where the target node u is associated with a new community $C'_i \in f(\mathcal{G}')$. One possible way to characterize community membership hiding for u is to aim for a small similarity between the new community and the original community to which u belonged. Alternatively, one may opt to enforce specific changes, ensuring that the target node u no longer shares the same community with *some* nodes. For example, if $C_i = \{s, t, \underline{u}, v, w, x, y, z\}$, we might wish to assign u to a new community C'_i such that $s, w, z \notin C'_i$.

In this work, we adopt the first definition, leaving exploration of other possibilities for future research. Practically, we set a similarity threshold between C'_i and C_i, excluding the target node u, which belongs to both communities by definition. This condition can be expressed as $sim(C_i - u, C'_i - u) \leq \tau$, where $\tau \in [0, 1]$. (Note: We assume that $sim(\cdot,\cdot)$ ranges between 0 and 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) [25], the Jaccard coefficient [14], and the Sørensen-Dice coefficient [6]. Setting $\tau = 0$ represents the most stringent scenario, where we require zero overlaps between C'_i and 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 C'_i and C_i. However, 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 C'_i being equal to 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. Thus, 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 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:

$$\mathcal{L}(h_{\boldsymbol{\theta}}; \mathcal{G}, f, u) = \ell_{\text{decept}}(\mathcal{G}, h_{\boldsymbol{\theta}}(\mathcal{G}); f, u) + \lambda \ell_{\text{dist}}(\mathcal{G}, h_{\boldsymbol{\theta}}(\mathcal{G}); f). \quad (1)$$

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 C_i. More formally, let $\widetilde{C}_i$ be the community to which node u is assigned when f is applied to the input graph $\widetilde{\mathcal{G}}$. We define $\Gamma = \{\widetilde{\mathcal{G}} \mid sim(C_i - \{u\}, \widetilde{C}_i - \{u\}) > \tau\}$. Thus, we can compute ℓ_{decept} as follows:

$$\ell_{\text{decept}}(\mathcal{G}, h_{\boldsymbol{\theta}}(\mathcal{G}); f) = \mathbb{1}_{\Gamma}(h_{\boldsymbol{\theta}}(\mathcal{G})), \quad (2)$$

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.

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 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 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 [8], we assume 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 intentionally exclude two possible actions: (iii) removing outgoing links to outside-community nodes and (iv) adding outgoing links to inside-community nodes. Two primary reasons drive this choice. On the one hand, allowing (iii) could isolate u and its original community C_i further. On the other hand, allowing (iv) would enhance connectivity between u and other nodes in C_i. Both events contradict the goal of community membership hiding.

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

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

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

If we suppose the target node u has a fixed budget $\beta > 0$, solving the community membership hiding task resorts to finding the optimal model $h^* = h_{\boldsymbol{\theta}^*}$ as the one whose parameters $\boldsymbol{\theta}^*$ minimize Eq. (1), i.e., by solving the following constrained objective:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left\{ \mathcal{L}(h_{\boldsymbol{\theta}}; \mathcal{G}, f, u) \right\} \quad \text{subject to: } |\mathcal{B}_{u,i}| \leq \beta, \quad (3)$$

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. (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.

4.3 Markov Decision Process

The community membership hiding problem defined in Eq. (3) requires minimizing a discrete, non-differentiable loss function. Thus, standard optimization methods like stochastic gradient descent are unsuitable for this task. One potential solution is to smooth the loss function using numerical techniques, such as applying a real-valued perturbation matrix to the original graph's adjacency matrix like in [23, 40]. Another option is to directly define ℓ_{decept} as a smooth similarity between the original and the newly obtained community, sacrificing control over the threshold τ. We leave the exploration of these alternatives for future work. Instead, we take a different approach and cast this problem 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 node hiding-goal and the optimal counterfactual graph $\mathcal{G}^*$ – i.e., the optimal h^* – is found.

We formalize this scheme as a discounted Markov decision process (MDP) denoted as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, p_0, r, \gamma\}$ and detailed below.

***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 $\boldsymbol{A}_t \in \{0,1\}^{n \times n}$. Initially, when $t = 0$, $\mathcal{G}^0 = \mathcal{G}$ ($\boldsymbol{A}^0 = \boldsymbol{A}$).

***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.

$$\mathcal{A} = \{\text{del}(u,v) | (u,v) \in \mathcal{E}^-_{u,i}\} \cup \{\text{add}(u,v) | (u,v) \in \mathcal{E}^+_{u,i}\}. \quad (4)$$

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.

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.

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:

$$r(s_t, a_t) = \begin{cases} 1 - \lambda(\ell^t_{\text{dist}} - \ell^{t-1}_{\text{dist}}) & \text{, if "the goal is met"} \\ -\lambda(\ell^t_{\text{dist}} - \ell^{t-1}_{\text{dist}}) & \text{, otherwise.} \end{cases} \tag{5}$$

The goal is considered successfully achieved when $f(\mathcal{G}^t)$ leads to $u \in C^t_i \neq C_i$ such that $sim(C_i - \{u\}, C^t_i - \{u\}) \leq \tau$. In addition, $\ell^t_{\text{dist}} = \ell_{\text{dist}}(\mathcal{G}, \mathcal{G}^t; f)$ 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:

$$\ell_{\text{dist}}(\mathcal{G}, \mathcal{G}^t; f) = \alpha \times d_{\text{community}} + (1 - \alpha) \times d_{\text{graph}}, \tag{6}$$

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.

Policy **(π_θ).** We first define a parameterized policy π_θ 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. (3). This minimization leads to the discovery of the optimal model h^*, which is the model that best predicts the rewards in the MDP.

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \ell_{\text{decept}}(h_{\boldsymbol{\theta}}; \mathcal{G}, f, u) + \lambda \ell_{\text{dist}}(\mathcal{G}, h_{\boldsymbol{\theta}}(\mathcal{G}); f) \\ &= \arg\max_{\boldsymbol{\theta}} \; - \ell_{\text{decept}}(h_{\boldsymbol{\theta}}; \mathcal{G}, f, u) - \lambda \ell_{\text{dist}}(\mathcal{G}, h_{\boldsymbol{\theta}}(\mathcal{G}); f) \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{t=1}^{T} \begin{cases} 1 - \lambda(\ell^t_{\text{dist}} - \ell^{t-1}_{\text{dist}}) & \text{, if "the goal is met"} \\ -\lambda(\ell^t_{\text{dist}} - \ell^{t-1}_{\text{dist}}) & \text{, otherwise} \end{cases} \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{t=1}^{T} r(s_t, \pi_{\boldsymbol{\theta}}(s_t)), \end{aligned} \tag{7}$$

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., $T \leq \beta$.

5 Proposed Method

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 (π_θ) and another for the value function estimator (V_v), such that:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) &\sim \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) A(s_t, a_t), \\ \text{with } A(s_t, a_t) &= r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t), \end{aligned} \tag{8}$$

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

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

Policy Function Network (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* [13]. These node embeddings serve as the feature matrix.

Our neural network implementation comprises a primary graph convolution layer (GCNConv [18]) 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 multilayer 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. [11], 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.

Value Function Network (Critic). This network is similar to the one used for the policy, with one distinction: it includes a global sum-pooling operation on the convolution layer's output. This pooling operation results in an output layer with a size of 1, indicating the estimated value of the value function. The role of the value function is to predict the state value given a specific action a_t and state s_t.

6 Experiments

6.1 Experimental Setup

Datasets. We train our DRL agent on the real dataset `words`.[3] This dataset strikes a favorable balance regarding the number of nodes, edges, and discovered communities. In addition, we evaluate the performance of our method on four additional datasets: `kar`[3], Wikipedia's `vote` [4], `pow`[3], and Facebook `fb-75`[4].

Community Detection Algorithms. The DRL agent is trained using a single detection algorithm, namely the modularity-based Greedy (`greedy`) algorithm [2]. However, at test time, we use two additional algorithms: the Louvain (`louvain`) algorithm [1], which

[3] http://konect.cc/
[4] https://networkrepository.com

Table 1: Properties of the datasets used along with the size of the communities detected by `greedy`, `louvain`, and `walktrap`.

Dataset	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	Community Detection Algorithm		
			`greedy`	`louvain`	`walktrap`
`kar`	34	78	3	4	5
`words`	112	425	7	7	25
`vote`	889	2,900	12	10	42
`pow`	4,941	6,594	40	41	364
`fb-75`	6,386	217,662	29	19	357

is in the same family as `greedy`, and WalkTrap (`walktrap`) [31], which takes a distinct approach centered on Random Walks.

Table 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 (`greedy` and `louvain`) yield comparable community counts, while `walktrap` identifies a generally higher number of communities.

Similarity/Distance Metrics. To assess the achievement of the community membership hiding goal, i.e., whether the new community C_i^t of the target node u at step t can no longer be considered the same as the initial community C_i, we need to define the $sim(\cdot)$ function used within ℓ_{decept}. We employ the Sørensen-Dice coefficient [6], which is defined as follows:

$$\text{DSC}(C_i, C_i^t) = \frac{2|C_i \bigcap C_i^t|}{|C_i| + |C_i^t|}. \tag{9}$$

This metric 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 node-hiding goal successfully met.

Furthermore, as described in Eq. (6), we must specify the penalty function (ℓ_{dist}), which consists of two mutually balanced factors, namely $d_{\text{community}}$ and d_{graph}. These factors 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 [19, 36], utilized for measuring the similarity between community structures, ranges from 0 (indicating no mutual information) to 1 (indicating perfect correlation).

Following the formulation by [21], it can be expressed as follows:

$$\text{NMI}(\mathcal{K}, \mathcal{K}^t) = I_{\text{norm}}(X : Y) = \frac{H(X) + H(Y) - H(X,Y)}{(H(X) + H(Y))/2}, \tag{10}$$

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 [7], as follows:

$$\text{Jaccard}(\mathcal{G}, \mathcal{G}^t) = \frac{|\mathcal{G} \bigcup \mathcal{G}^t| - |\mathcal{G} \bigcap \mathcal{G}^t|}{|\mathcal{G} \bigcup \mathcal{G}^t|} = \frac{\sum_{i,j} |\boldsymbol{A}_{i,j} - \boldsymbol{A}_{i,j}^t|}{\sum_{i,j} \max(\boldsymbol{A}_{i,j}, \boldsymbol{A}_{i,j}^t)}, \tag{11}$$

where $\boldsymbol{A}_{i,j}$ 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.

6.2 Community Membership Hiding Task

Baselines. We compare the performance of our method (*DRL-Agent*) against the five baselines 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 target 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 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) Betweenness-based. This baseline prioritizes nodes with the highest *betweenness centrality* [10] as those candidates to be disconnected. The betweenness centrality of a node u, denoted as $b(u)$, is computed as $b(u) = \sum_{s \neq u \neq t} \frac{\sigma_{s,t}(u)}{\sigma_{s,t}}$, where $\sigma_{s,t}$ is the total number of shortest paths from node s to node t and $\sigma_{s,t}(u)$ is the number of those paths that pass through u (where u is *not* an end point).

4) Roam-based. This method is based on the Roam heuristics [41], originally designed to reduce a node's centrality within the network. It aims to diminish the centrality and influence of the target node within its initial community, favoring its deception.

5) Greedy-based. We develop a "relaxed" greedy heuristic that selects the most promising rewiring action without exhaustively exploring all the possible operations by prioritizing higher-degree nodes.

Let u be the target node to be masked from the community C. At each step, we still can choose between (i) adding an edge between u and a node v outside the community or (ii) deleting an existing edge between u and a node w inside the same community. To greedily determine what action provokes the largest loss reduction as of Eq. (1), we restrict the two actions above as follows.

Concerning (i), we do not consider all the possible nodes $v \in \mathcal{V} \setminus C$ but only the node v^* with the highest out-degree. Then, we measure the loss reduction once the edge (u, v^*) is added. The rationale is similar to the one already used in the Degree-based baseline, where the goal is to connect u to a popular node and, hopefully, mitigate its "centrality" in its current community C. Concerning (ii), we select the edge to remove as follows. For each node $w \neq u$ that belongs to the same community C, such that $(u, w) \in \mathcal{E}$, we calculate the intra-community degree $deg_C(w) = |\{(w, x) \in \mathcal{E} | x \in C\}|$. Then, we delete the edge (u, w^*) with the node w^* having the highest intra-community degree. The rationale of this choice is to remove the link between u and the node with the largest number of connections within the community. By doing so, we expect to increase the chance for u to be masked from C. Again, we measure the loss reduction if the edge (u, w^*) is deleted.

At each step, we decide between adding (u, v^*) or deleting (u, w^*) by selecting the action that results in the highest loss reduction. This relaxed greedy approach is computationally more tractable than the full-greedy heuristic, as it allows us to run the community detection algorithm $f(\cdot)$ only twice at each step, once for each of the two actions. We keep doing this until the goal is achieved or the budget β is exhausted.

Evaluation Metrics. We measure the performance of each method in solving the community membership hiding task using the following metrics.

1) Success Rate (SR). This metric calculates the success rate of the membership hiding algorithm by determining the percentage of times the target node is successfully hidden from its original community. If the target node no longer belongs to the original community (as per Eq. (9) 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. A higher value of this metric indicates better performance.
2) Normalized Mutual Information (NMI). To quantify the impact of the function $h_{\theta}(\cdot)$ 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 NMI($\mathcal{K}, \mathcal{K}'$), as outlined in Eq. (10). 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 and, therefore, a smaller cost.

In general, SR and NMI are two contrastive metrics; a higher SR corresponds to a lower NMI and vice versa. Therefore, similar to the $F1$ score, which balances precision and recall, we calculate the harmonic mean between SR and NMI using the formula $\frac{2\times SR\times NMI}{SR+NMI}$ to evaluate which method achieves the optimal trade-off.

6.3 Results and Discussion

We assess 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. Furthermore, we explore two distinct setups: *symmetric* and *asymmetric*. In the symmetric case, our *DRL-Agent* is trained and tested using the *same* community detection algorithm used for the membership hiding task. Instead, the asymmetric setup evaluates the performance of our method when tested on a community detection algorithm different from the one used for training. This second setting allows us to assess the *transferability* of our method to community detection algorithms unseen at training time. Specifically, the results below showcase the outcome when we train our *DRL-Agent* using the modularity-based `greedy` algorithm for both setups. In the symmetric setting, `greedy` is also used at test time, while in the asymmetric setup, we employ the `louvain` or `walktrap` algorithm.

All the experiments were conducted on a 4-core Intel Xeon CPU running at 2.2 GHz with 18 GB RAM.

In Tables 2 and 3, we display the Success Rate (SR) as a function of budget (β) for each dataset in the symmetric and two asymmetric settings, respectively, with a tolerance threshold of $\tau = 0.5$.

In Figures 2, 3, and 4, we show the harmonic mean ($F1$ score) between the Success Rate and NMI for all datasets in symmetric and asymmetric setups, still with a tolerance threshold of $\tau = 0.5$ and a budget $\beta = 1\mu$.

From the results above, we emphasize two primary findings. Firstly, our approach strikes the best balance between accuracy and

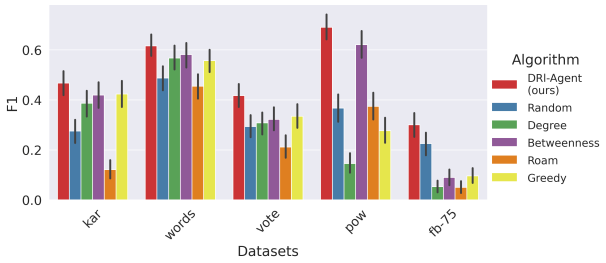

Figure 2: F1 score of SR and NMI in *symmetric* setting (training: `greedy`; testing: `greedy`; $\tau = 0.5$; $\beta = 1\mu$).

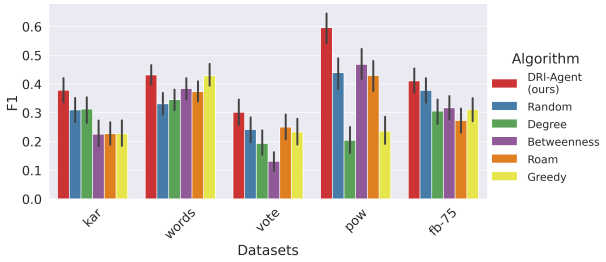

Figure 3: F1 score of SR and NMI in *asymmetric* setting (training: `greedy`; testing: `louvain`; $\tau = 0.5$; $\beta = 1\mu$).

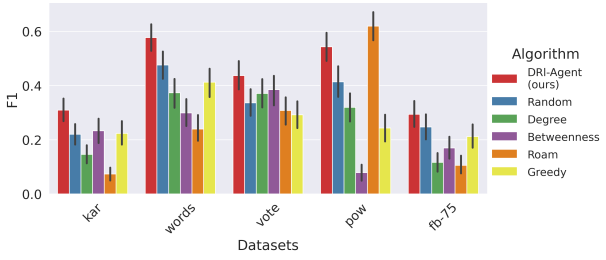

Figure 4: F1 score of SR and NMI in *asymmetric* setting (training: `greedy`; testing: `walktrap`; $\tau = 0.5$; $\beta = 1\mu$).

cost. Indeed, for a fixed budget value, our *DRL-Agent* achieves the highest Success Rate in hiding the selected target node compared to other competing methods. The only exception occurs in a single dataset (pow) and for a specific budget value ($\beta = 2\mu$), where the method based on *Betweenness* outperforms all others. Moreover, our approach pays a limited "price" for this success. Specifically, the modified graph retains much of the structural properties of the original. This is demonstrated by the highest $F1$ score between SR and NMI compared to other methods.

The second key finding concerns the transferability of our *DRL-Agent* to a different community detection algorithm than the one seen during training (*asymmetric* setup). As illustrated in Table 2, our approach generally works better in the symmetric setup (like other baseline methods). Still, Table 3 shows that our *DRL-Agent* also dominates over competitors in the asymmetric settings. This quality makes our approach effective and applicable even when the underlying community detection algorithm is unknown.

Table 2: Success Rate (SR) vs. budget (β) for community membership hiding task in the *symmetric* setting ($\tau = 0.5$).

Dataset	β	*Symmetric* (training: greedy; testing: greedy)					
		DRL-Agent (ours)	*Random*	*Degree*	*Betweenness*	*Roam*	*Greedy*
kar	$\frac{1}{2}\mu$	**26.3% ± 4.9%**	22.0% ± 4.6%	16.3% ± 4.1%	14.6% ± 4.0%	13.6% ± 3.8%	0.6% ± 0.9%
	1μ	**54.0% ± 5.6%**	31.6% ± 5.2%	42.3% ± 5.5%	46.0% ± 5.6%	12.6% ± 3.7%	45.3% ± 5.6%
	2μ	**70.3% ± 5.1%**	46.6% ± 5.6%	43.6% ± 5.6%	55.6% ± 5.6%	13.6% ± 3.8%	38.3% ± 5.5%
words	$\frac{1}{2}\mu$	**49.7% ± 5.7%**	48.7% ± 5.7%	48.0% ± 5.7%	46.0% ± 5.6%	45.3% ± 5.6%	29.0% ± 5.1%
	1μ	**71.7% ± 5.1%**	57.0% ± 5.6%	62.3% ± 5.5%	64.6% ± 5.4%	51.7% ± 5.7%	64.0% ± 5.4%
	2μ	**88.3% ± 3.6%**	71.0% ± 5.1%	77.3% ± 4.7%	79.6% ± 4.5%	44.7% ± 5.6%	79.7% ± 4.5%
vote	$\frac{1}{2}\mu$	**38.6% ± 5.5%**	16.6% ± 4.2%	20.6% ± 4.5%	21.0% ± 4.6%	26.0% ± 4.9%	3.3% ± 2.0%
	1μ	**49.6% ± 5.6%**	35.3% ± 5.4%	37.3% ± 5.4%	38.6% ± 5.5%	22.3% ± 4.7%	38.3% ± 5.5%
	2μ	**65.3% ± 5.3%**	47.0% ± 5.6%	53.0% ± 5.6%	55.0% ± 5.6%	40.0% ± 5.5%	53.3% ± 5.6%
pow	$\frac{1}{2}\mu$	**40.0% ± 5.5%**	33.3% ± 5.3%	19.3% ± 4.4%	20.3% ± 4.5%	32.3% ± 5.2%	6.7% ± 2.8%
	1μ	**71.3% ± 5.1%**	38.0% ± 5.4%	15.0% ± 4.0%	64.0% ± 5.4%	38.6% ± 5.5%	28.7% ± 5.1%
	2μ	91.6% ± 3.1%	47.3% ± 5.6%	91.6% ± 3.1%	**93.0% ± 2.8%**	28.6% ± 5.1%	92.0% ± 3.0%
fb-75	$\frac{1}{2}\mu$	**29.0% ± 5.1%**	20.3% ± 4.5%	8.3% ± 3.1%	10.3% ± 3.4%	6.3% ± 2.7%	11.3% ± 3.6%
	1μ	**33.6% ± 5.3%**	24.3% ± 4.8%	5.6% ± 2.6%	9.6% ± 3.3%	5.3% ± 2.5%	10.3% ± 3.4%
	2μ	**45.0% ± 5.6%**	36.3% ± 5.4%	11.0% ± 3.5%	16.3% ± 4.1%	8.0% ± 3.0%	17.3% ± 4.3%

Table 3: Success Rate (SR) vs. budget (β) community membership hiding task in two *asymmetric* settings ($\tau = 0.5$).

Dataset	β	*Asymmetric* (training: greedy; testing: louvain)						*Asymmetric* (training: greedy; testing: walktrap)					
		DRL-Agent (ours)	*Random*	*Degree*	*Betweenness*	*Roam*	*Greedy*	*DRL-Agent (ours)*	*Random*	*Degree*	*Betweenness*	*Roam*	*Greedy*
kar	$\frac{1}{2}\mu$	**33.3% ± 5.3%**	31.3% ± 5.2%	24.0% ± 4.8%	7.6% ± 3.0%	27.0% ± 5.0%	1.0% ± 1.1%	**10.0% ± 3.4%**	6.7% ± 2.8%	6.7% ± 2.8%	4.0% ± 2.2%	6.0% ± 2.7%	4.0% ± 2.2%
	1μ	**50.0% ± 5.6%**	41.0% ± 5.5%	39.6% ± 5.5%	25.0% ± 4.9%	28.3% ± 5.1%	25.0% ± 4.9%	**41.7% ± 5.6%**	30.3% ± 5.2%	19.7% ± 4.5%	26.7% ± 5.0%	9.3% ± 3.3%	25.0% ± 4.9%
	2μ	**66.6% ± 5.3%**	46.6% ± 5.6%	41.0% ± 5.5%	37.0% ± 5.4%	24.6% ± 4.8%	46.0% ± 5.6%	**88.3% ± 3.6%**	61.0% ± 5.5%	58.3% ± 5.6%	70.3% ± 5.2%	7.7% ± 3.0%	74.3% ± 4.9%
words	$\frac{1}{2}\mu$	**57.6% ± 5.5%**	44.0% ± 5.6%	48.6% ± 5.6%	52.0% ± 5.6%	50.6% ± 5.6%	50.0% ± 5.7%	**43.0% ± 5.6%**	35.7% ± 5.4%	20.3% ± 4.5%	29.7% ± 5.2%	32.0% ± 5.3%	21.0% ± 4.6%
	1μ	**68.3% ± 5.2%**	51.6% ± 5.6%	54.6% ± 5.6%	60.3% ± 5.5%	59.3% ± 5.5%	60.0% ± 5.5%	**64.0% ± 5.4%**	52.7% ± 5.7%	41.0% ± 5.6%	31.3% ± 5.2%	26.0% ± 5.0%	43.7% ± 5.6%
	2μ	**84.0% ± 4.1%**	61.6% ± 5.5%	59.6% ± 5.5%	69.0% ± 5.2%	56.3% ± 5.6%	83.0% ± 4.2%	**76.7% ± 4.8%**	66.3% ± 5.3%	65.0% ± 5.4%	51.7% ± 5.7%	42.0% ± 5.6%	57.0% ± 5.6%
vote	$\frac{1}{2}\mu$	**22.0% ± 4.6%**	19.0% ± 4.4%	14.3% ± 3.9%	8.0% ± 3.0%	20.3% ± 4.5%	7.0% ± 2.9%	**27.0% ± 5.0%**	23.0% ± 4.8%	13.3% ± 3.9%	12.7% ± 3.8%	25.0% ± 4.9%	7.0% ± 2.9%
	1μ	**35.6% ± 5.4%**	28.6% ± 5.1%	23.0% ± 4.7%	16.3% ± 4.1%	29.6% ± 5.1%	26.7% ± 5.0%	**45.7% ± 5.6%**	35.3% ± 5.4%	38.7% ± 5.5%	39.3% ± 5.5%	32.0% ± 5.3%	29.7% ± 5.2%
	2μ	**45.0% ± 5.6%**	35.6% ± 5.4%	39.6% ± 5.5%	21.6% ± 4.6%	31.0% ± 5.2%	40.0% ± 5.5%	**66.7% ± 5.3%**	51.3% ± 5.7%	64.0% ± 5.4%	65.0% ± 5.4%	42.7% ± 5.6%	64.7% ± 5.4%
pow	$\frac{1}{2}\mu$	**56.6% ± 5.6%**	41.3% ± 5.5%	19.3% ± 4.4%	15.0% ± 4.0%	45.3% ± 5.6%	11.3% ± 3.6%	49.7% ± 5.7%	37.7% ± 5.5%	29.0% ± 5.1%	5.3% ± 2.5%	**65.3% ± 5.4%**	7.3% ± 2.9%
	1μ	**63.3% ± 5.4%**	46.6% ± 5.6%	21.6% ± 4.6%	49.6% ± 5.6%	45.6% ± 5.6%	25.0% ± 4.9%	56.0% ± 5.6%	42.7% ± 5.6%	33.0% ± 5.3%	8.0% ± 3.1%	**64.0% ± 5.4%**	24.7% ± 4.9%
	2μ	82.3% ± 4.3%	55.3% ± 5.6%	32.3% ± 5.2%	**96.0% ± 2.2%**	47.0% ± 5.6%	43.3% ± 5.6%	**62.0% ± 5.5%**	46.0% ± 5.6%	48.0% ± 5.7%	27.7% ± 5.1%	44.3% ± 5.6%	36.3% ± 5.4%
fb-75	$\frac{1}{2}\mu$	**48.3% ± 5.6%**	42.0% ± 5.5%	37.0% ± 5.4%	41.0% ± 5.5%	36.3% ± 5.4%	41.0% ± 5.6%	**31.7% ± 5.3%**	20.3% ± 4.5%	10.3% ± 3.4%	12.0% ± 3.7%	12.0% ± 3.7%	14.3% ± 3.9%
	1μ	**56.0% ± 5.6%**	48.6% ± 5.6%	41.6% ± 5.5%	43.3% ± 5.6%	34.3% ± 5.3%	42.3% ± 5.6%	**32.3% ± 5.3%**	27.0% ± 5.0%	13.0% ± 3.8%	19.0% ± 4.4%	12.3% ± 3.7%	23.3% ± 4.8%
	2μ	**66.3% ± 5.3%**	63.6% ± 5.4%	60.0% ± 5.5%	64.0% ± 5.4%	48.0% ± 5.6%	62.0% ± 5.5%	**36.0% ± 5.4%**	33.7% ± 5.3%	16.3% ± 4.2%	22.3% ± 4.7%	18.3% ± 4.4%	25.7% ± 4.9%

6.4 Parameter Sensitivity

The effectiveness of our *DRL-Agent* relies on two critical parameters: (i) the similarity threshold (τ) used to determine whether the node-hiding 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 4, we explore how the Success Rate for the community membership hiding task is influenced by varying the values of τ and β, while keeping the detection algorithm $f(\cdot)$ and dataset fixed. The results shown refer to a specific symmetric setup using the `greedy` community detection algorithm on the `words` dataset.

Table 4: The impact of τ and β on Success Rate (SR), using the `greedy` community detection algorithm on the `words` dataset.

τ	β	Community Membership Hiding Algorithm					
		DRL-Agent (ours)	*Random*	*Degree*	*Betweenness*	*Roam*	*Greedy*
0.3	$\frac{1}{2}\mu$	**41.7% ± 5.6%**	39.0% ± 5.5%	39.3% ± 5.5%	40.7% ± 5.7%	36.7% ± 5.4%	26.7% ± 5.0%
	1μ	**61.7% ± 5.5%**	46.0% ± 5.6%	53.7% ± 5.6%	55.0% ± 5.6%	41.3% ± 5.6%	54.7% ± 5.6%
	2μ	**77.7% ± 4.7%**	62.7% ± 5.5%	64.7% ± 5.4%	67.7% ± 5.3%	50.7% ± 5.7%	71.0% ± 5.1%
0.5	$\frac{1}{2}\mu$	**49.7% ± 5.7%**	48.7% ± 5.7%	48.0% ± 5.7%	46.0% ± 5.6%	45.3% ± 5.6%	29.0% ± 5.1%
	1μ	**71.7% ± 5.1%**	57.0% ± 5.6%	62.3% ± 5.5%	64.7% ± 5.4%	51.7% ± 5.7%	64.0% ± 5.4%
	2μ	**88.3% ± 3.6%**	71.0% ± 5.1%	77.3% ± 4.7%	79.7% ± 4.6%	44.7% ± 5.6%	79.7% ± 4.5%
0.8	$\frac{1}{2}\mu$	**62.7% ± 5.5%**	55.3% ± 5.6%	52.0% ± 5.7%	49.0% ± 5.7%	45.3% ± 5.6%	32.0% ± 5.3%
	1μ	**90.0% ± 3.4%**	73.7% ± 5.0%	75.0% ± 4.9%	77.7% ± 4.7%	58.0% ± 5.6%	71.0% ± 5.1%
	2μ	**96.3% ± 2.1%**	82.3% ± 4.3%	86.3% ± 3.9%	88.3% ± 3.6%	62.0% ± 5.5%	92.7% ± 2.9%

As one might expect, by increasing the similarity threshold (τ), the node deception goal is easier to achieve for our *DRL-Agent* and, therefore, the Success Rate is higher (for a given fixed budget β). This happens because we pose a less strict requirement on the distance between the original community of the target node to hide and the one where it eventually ends up after the graph modifications induced by our method. Similarly, granting a larger budget (β) allows our method to alter more substantially the neighborhood of the target node to hide, hence increasing its chance of success.

6.5 Computational Complexity

The size of the graph datasets used in our experiments ranges from small to large (e.g., the Facebook dataset `fb-75` has thousands of nodes and hundreds of thousands of edges). However, we acknowledge that real-world network graphs may count millions, if not billions, of nodes. Therefore, we analyze the computational complexity of our proposed method to assess its feasibility and potential deployment into extremely large-scale production environments.

The primary computational challenge of our method lies in training our *DRL-Agent*. In each time step of the training process, the agent can select from a set of actions $\mathcal{A}$. Each action corresponds to either removing an existing edge or adding a new one, as detailed in Section 4.2. Thus, $|\mathcal{A}| \leq |\mathcal{V}|^2 = n^2$, considering the worst-case scenario of the target node being part of a fully connected graph.

The number of possible states ($\mathcal{S}$), instead, equates to the potential adjacency matrices for a graph with $|\mathcal{V}| = n$ nodes, i.e., $|\mathcal{S}| = 2^{n^2}$.

In general, given a discounted MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, p_0, r, \gamma\}$ and assuming that sampling state-action pairs from the transition function $\mathcal{P}$ takes $O(1)$ time, Sidford et al. [37] show that the upper bound on the time spent and number of samples taken for computing an ϵ-optimal policy with probability $1 - \delta$ is:

$$O\left[\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2}\log\left(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)\delta\epsilon}\right)\log\left(\frac{1}{(1-\gamma)\epsilon}\right)\right].$$

Given the exponential number of states in our worst case setting, our *DRL-Agent* may result impractical to train for graphs with a large number n of nodes. Nonetheless, in practice, the number of states and actions allowed are significantly smaller. Experimental results, even on graphs with thousands of nodes like `fb-75`, demonstrate that convergence to the optimal policy occurs at a faster rate, and the theoretical complexity bound above might not be tight.

7 Discussion

7.1 Who Can Run Our Method?

In our scenario, we assume that a social network runs the community detection algorithm $f(\cdot)$. Moreover, this social network may offer its end-users the capability to opt out of being detected, accommodating their privacy needs. In such a case, the platform has *full knowledge* of the graph, and our community membership hiding algorithm can be run seamlessly. Thus, the resulting counterfactual graph can be used to suggest to the target end-user what links they should add/remove to meet their deception goal.

On the contrary, if the social network does not offer this opt-out feature, the end-user may still attempt to remain concealed from the community detection algorithm. However, in this case, two critical considerations come into play, as the average end-user typically: (i) lacks access to the full graph structure, and (ii) cannot directly execute the community detection algorithm $f(\cdot)$.

To address (i), the target end-user who wants to get masked off using our community membership hiding algorithm must first perform web scraping to obtain their local graph structure, albeit only the community to which they belong. For example, this community can be constructed by examining the followers of the people the target end-user follows. However, complications may arise, particularly on platforms like Instagram or Facebook, where viewing a user's followers is generally possible if you follow them back.

To tackle (ii), the target end-user must establish empirical indicators that serve as proxies to determine if their hiding goal has been met. For instance, this could involve criteria such as "not receiving any more following suggestions or advertisements that are clearly targeted for the community they want to be masked from."

Moreover, in a real-world scenario, multiple nodes may require concealment independently. There is a risk that modifications aiding the concealment of one node might hinder another's, potentially reducing the overall success rate. This issue arises when the nodes to be hidden share non-trivial neighborhood overlaps. To address this, we propose extending our framework to handle node sets rather than individuals, akin to a multi-agent reinforcement learning (MARL) problem. Each agent, representing a node, collaborates to achieve their hiding objectives.

7.2 Security and Ethical Implications

As highlighted in the motivation for this work, community membership hiding algorithms can serve as valuable tools for safeguarding the privacy of social network users. Furthermore, these methods can be used to protect individuals at risk, including journalists or opposition activists, in regions governed by authoritarian regimes. Additionally, these techniques can combat online criminal activities by modifying network connections to infiltrate espionage agents or disrupt communications among malicious users.

However, node-hiding techniques can also be exploited to pursue harmful goals. For instance, malicious individuals can strategically use these methods to evade network analysis tools, often employed by law enforcement for public safety, enabling them to mask their illicit or criminal activities on the network. Furthermore, in this work, we focus on a *single* node u to be hidden from a community without considering potential side effects on other nodes v. As we only manipulate edges controlled by u, the chance of another node v unintentionally joining the community without any direct action from v itself (e.g., adding a link to a node w in the community) seems rare, albeit possible. To deal with such potential side effects, we can either limit the set of feasible actions or conduct post-processing sanity checks to ensure that no node that was not originally part of the community ends up included therein.

Overall, for a social network offering community membership hiding capabilities, it is essential to thoroughly assess the impact of this feature *before* granting users the actual ability to conceal themselves from community detection algorithms.

8 Conclusion and Future Work

This paper tackled the challenge of *community membership hiding*, which entails strategically modifying the structural characteristics of a network graph to prevent a target node from being detected by a community detection algorithm. To address this problem, we formulated it as a constrained counterfactual graph objective and solved it via deep reinforcement learning. We conducted extensive experiments to validate our method's effectiveness. Results demonstrated that our approach strikes the best balance between achieving the desired node-hiding goal and the required cost of graph modifications compared to existing baselines.

In future work, we aim to explore different definitions of community membership hiding and incorporate node feature modifications alongside structural alterations to the counterfactual graph objective. Furthermore, we plan to apply our method to extremely large network graphs. Finally, we will generalize our method to address the community deception task or scenarios where multiple users simultaneously request node membership hiding.

Acknowledgments

This work was partially supported by the projects FAIR (PE0000013), SERICS (PE00000014), and SoBigData.it (IR0000013) under the National Recovery and Resilience Plan funded by the European Union NextGenerationEU, as well as HypeKG – Hybrid Prediction and Explanation with Knowledge Graphs (H53D23003700006) under the PRIN 2022 program funded by the Italian MUR.

References

[1] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast Unfolding of Communities in Large Networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (Oct 2008), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

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

[3] Alina Campan, Yasmeen Alufaisan, and Traian Marius Truta. 2015. Preserving Communities in Anonymized Social Networks. *Transactions on Data Privacy* 8, 1 (Dec 2015), 55–87.

[4] Xianyu Chen, Zhongyuan Jiang, Hui Li, Jianfeng Ma, and Philip S. Yu. 2021. Community Hiding by Link Perturbation in Social Networks. *IEEE Transactions on Computational Social Systems* 8, 3 (2021), 704–715. https://doi.org/10.1109/TCSS.2021.3054115

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

[6] Lee R. Dice. 1945. Measures of the Amount of Ecologic Association Between Species. *Ecology* 26, 3 (1945), 297–302. https://doi.org/10.2307/1932409 arXiv:https://esajournals.onlinelibrary.wiley.com/doi/pdf/10.2307/1932409

[7] Claire Donnat and Susan Holmes. 2018. Tracking Network Dynamics: A Survey of Distances and Similarity Metrics. arXiv:1801.07351 [stat.AP]

[8] Valeria Fionda and Giuseppe Pirrò. 2018. Community Deception or: How to Stop Fearing Community Detection Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 30, 4 (2018), 660–673. https://doi.org/10.1109/TKDE.2017.2776133

[9] Santo Fortunato. 2010. Community Detection in Graphs. *Physics Reports* 486, 3 (2010), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

[10] Linton C. Freeman. 1977. A Set of Measures of Centrality Based on Betweenness. *Sociometry* 40, 1 (1977), 35–41. http://www.jstor.org/stable/3033543

[11] Daniele Gammelli, Kaidi Yang, James Harrison, Filipe Rodrigues, Francisco C. Pereira, and Marco Pavone. 2021. Graph Neural Network Reinforcement Learning for Autonomous Mobility-on-Demand Systems. arXiv:2104.11434 [eess.SY]

[12] M. Girvan and M. E. J. Newman. 2002. Community Structure in Social and Biological Networks. *Proceedings of the National Academy of Sciences* 99, 12 (2002), 7821–7826. https://doi.org/10.1073/pnas.122653799 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.122653799

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. arXiv:1607.00653 [cs.SI]

[14] Paul Jaccard. 1912. The Distribution of the Flora in the Alpine Zone. *New Phytologist* 11, 2 (1912), 37–50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

[15] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Dongxiao He, Jia Wu, Philip S. Yu, and Weixiong Zhang. 2021. A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning. arXiv:2101.01669 [cs.SI]

[16] N. Kalaichelvi and K. S. Easwarakumar. 2022. A Comprehensive Survey on Community Deception Approaches in Social Networks. In *Computer, Communication, and Signal Processing*, Erich J. Neuhold, Xavier Fernando, Joan Lu, Selwyn Piramuthu, and Aravindan Chandrabose (Eds.). Springer International Publishing, Cham, 163–173.

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

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 [cs.LG]

[19] J. Kreer. 1957. A Question of Terminology. *IRE Transactions on Information Theory* 3, 3 (1957), 208–208. https://doi.org/10.1109/TIT.1957.1057418

[20] Suchi Kumari, Riteshkumar Jayprakash Yadav, Suyel Namasudra, and Ching-Hsien Hsu. 2021. Intelligent Deception Techniques against Adversarial Attack on the Industrial System. *International Journal of Intelligent Systems* 36, 5 (2021), 2412–2437.

[21] Andrea Lancichinetti, Santo Fortunato, and János Kertész. 2009. Detecting the Overlapping and Hierarchical Community Structure in Complex Networks. *New Journal of Physics* 11, 3 (Mar 2009), 033015. https://doi.org/10.1088/1367-2630/11/3/033015

[22] Yiwei Liu, Jiamou Liu, Zijian Zhang, Liehuang Zhu, and Angsheng Li. 2019. *REM: From Structural Entropy to Community Structure Deception.* Curran Associates Inc., Red Hook, NY, USA.

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

[24] Shravika Mittal, Debarka Sengupta, and Tanmoy Chakraborty. 2021. Hide and Seek: Outwitting Community Detection Algorithms. *IEEE Transactions on Computational Social Systems* 8, 4 (2021), 799–808. https://doi.org/10.1109/TCSS.2021.3062711

[25] Vijaymeena M.K and Kavitha K. 2016. A Survey on Similarity Measures in Text Mining. https://api.semanticscholar.org/CorpusID:62118842

[26] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783 [cs.LG]

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

[28] Shishir Nagaraja. 2010. The Impact of Unlinkability on Adversarial Community Detection: Effects and Countermeasures. In *Privacy Enhancing Technologies*, Mikhail J. Atallah and Nicholas J. Hopper (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 253–272.

[29] Mark E. J. Newman. 2006. Finding Community Structure in Networks Using the Eigenvectors of Matrices. *Physical Review E* 74 (Sep 2006), 036104. Issue 3. https://doi.org/10.1103/PhysRevE.74.036104

[30] Mark E. J. Newman. 2006. Modularity and Community Structure in Networks. *Proceedings of the National Academy of Sciences* 103, 23 (Jun 2006), 8577–8582. https://doi.org/10.1073/pnas.0601602103

[31] Pascal Pons and Matthieu Latapy. 2005. Computing Communities in Large Networks Using Random Walks. (2005), 284–293.

[32] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near Linear Time Algorithm to Detect Community Structures in Large-Scale Networks. *Physical Review E* 76 (Sep 2007), 036106. Issue 3. https://doi.org/10.1103/PhysRevE.76.036106

[33] Jörg Reichardt and Stefan Bornholdt. 2006. Statistical Mechanics of Community Detection. *Physical Review E* 74 (Jul 2006), 016110. Issue 1. https://doi.org/10.1103/PhysRevE.74.016110

[34] Peter Ronhovde and Zohar Nussinov. 2009. Multiresolution Community Detection for Megascale Networks by Information-Based Replica Correlations. *Physical Review E* 80, 1 (Jul 2009). https://doi.org/10.1103/physreve.80.016109

[35] XingMao Ruan, YueHeng Sun, Bo Wang, and Shuo Zhang. 2012. The Community Detection of Complex Networks Based on Markov Matrix Spectrum Optimization. In *2012 International Conference on Control Engineering and Communication Technology*. 608–611. https://doi.org/10.1109/ICCECT.2012.192

[36] C. E. Shannon. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal* 27, 3 (1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[37] Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. 2018. Near-Optimal Time and Sample Complexities for Solving Markov Decision Processes with a Generative Model. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/bb03e43ffe34eeb242a2ee4a4f125e56-Paper.pdf

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

[39] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. 2017. 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*. ACM, 465–474. https://doi.org/10.1145/3097983.3098039

[40] Giovanni Trappolini, Valentino Maiorca, Silvio Severino, Emanuele Rodola, Fabrizio Silvestri, and Gabriele Tolomei. 2023. Sparse Vicious Attacks on Graph Neural Networks. *IEEE Transactions on Artificial Intelligence* (2023).

[41] Marcin Waniek, Tomasz P. Michalak, Michael J. Wooldridge, and Talal Rahwan. 2018. Hiding Individuals and Communities in a Social Network. *Nature Human Behaviour* 2, 2 (Jan 2018), 139–147. https://doi.org/10.1038/s41562-017-0290-3

[42] Joyce Jiyoung Whang, David F. Gleich, and Inderjit S. Dhillon. 2015. Overlapping Community Detection Using Neighborhood-Inflated Seed Expansion. arXiv:1503.07439 [cs.SI]

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