Social Network Analysis and Mining (2022) 12:74
https://doi.org/10.1007/s13278-022-00896-7

ORIGINAL ARTICLE

Community deception: from undirected to directed networks

Valeria Fionda[1] · Saif Aldeen Madi[2] · Giuseppe Pirrò[2]

Received: 6 March 2022 / Revised: 22 May 2022 / Accepted: 25 May 2022 / Published online: 6 July 2022

Abstract
Community deception is about hiding a target community that wants to remain below the radar of community detection algorithms. The goal is to devise algorithms that, given a maximum number of updates (e.g., edge additions and removal), strive to find the best way to perform such updates in order to hide the target community inside the community structure found by a detection algorithm. So far, community deception has only been studied for undirected networks, although many real-world networks (e.g., Twitter) are directed. One way to overcome this problem would be to treat the network as undirected. However, this approach discards potentially helpful information in the edge directions (e.g., A follows B does not imply that B follows A). The aim of this paper is threefold. First, to give an account of the state-of-the-art community deception techniques in undirected networks underlying their peculiarities. Second, to investigate the community deception problem in directed networks and to show how deception techniques proposed for undirected networks should be modified and adapted to work on directed networks. Third, to evaluate deception techniques both in undirected and directed networks. Our experimental evaluation on a variety of (large) directed networks shows that techniques that work well for undirected networks fail short when directly applied to directed networks, thus underlying the need for specific approaches.

1 Introduction

Complex network analysis is a powerful technique to model and analyze interactions between entities in complex systems (e.g., protein networks, social networks, signaling networks) (Strogatz 2001). One of the major tasks that can be performed over these networks is community detection, that is, the task of identifying a (non-overlapping) partition of nodes of the network, providing some insights about their structure (Fortunato and Hric 2016). Network analysis tools are routinely used by a variety of actors from data analysts that are interested, for instance, in suggesting items to buy to the users of a network. The problem arises when these spontaneously shared pieces of information are improperly used, as in the Cambridge Analytica case, where private personal information about users and their social relationships were used without their consent, or when information about communities is used to block forms of self-organization (King et al. 2013). Another example is the case of Bitcoin trading, where communities were used to identify multiple addresses belonging to the same user (Remy et al. 2017).

Hence, although community detection is an essential tool for discovering functional building blocks within networks, and to provide insights into the dynamics or modes of formation of networks (Leicht and Newman 2008), the question concerning *what disclosing the community structure of networks can cause to the users* remains primarily unsolved. The research community started to look into this problem giving rise to a new strand of research dubbed as *community hiding* (Waniek et al. 2018) or *community deception* (Fionda and Pirrò 2018). The general idea is to promote (simple) techniques that can be used by the participants to a community that wants to remain below the radar of network analysis techniques like community detection. This problem is particularly critical if who wants to *evade* community detection tools are *malevolent* users (e.g., criminals or terrorists) and who want to *identify* the communities are *police* enforcement. More formally, given a target community $\mathscr{C}$

✉ Giuseppe Pirrò
pirro@di.uniroma1.it

Valeria Fionda
valeria.fionda@unical.it

Saif Aldeen Madi
madi@di.uniroma1.it

[1] Department of Mathematics and Computer Science, University of Calabria, via Pietro Bucci 30B, 87036 Rende, CS, Italy

[2] Department of Computer Science, Sapienza University of Rome, Piazzale Aldo Moro 5, 00185 Rome, Italy

inside a network G and a budget β of updates (e.g., edge additions and removals), deception techniques investigate the best way to perform such updates in a way that $\mathscr{C}$ can escape community detection algorithms. To find the best edge updates, some function $\phi_G(\mathscr{C})$ like modularity (Newman and Girvan 2004), safeness (Fionda and Pirrò 2018), permanence (Mittal et al. 2021) is optimized. The desideratum is that, after applying the edge updates, the level of hiding of $\mathscr{C}$ inside a community structure (set of communities) $\overline{C} = \{C_1, C_2, ...C_k\}$ found on the updated network G' will increase.

So far, community deception has only been studied for undirected networks, although many real-world networks are directed. A notable example is Twitter, where users can follow other users. When user A follows user B, an edge is established from A to B. Several studies on community detection (e.g., Leicht and Newman 2008) have shown that edge directions are essential to discovering meaningful communities in directed networks. *The first challenge that we face in this paper is how to devise deception techniques that are aware of edge directions.*

One way to approach this challenge would be to apply deception techniques devised for undirected networks to directed networks by simply ignoring edge directions. However, this approach discards potentially helpful information contained in the edge directions (Leicht and Newman 2008; Malliaros and Vazirgiannis 2013), such as the fact that if there is an edge indicating that A follows B, this does not imply that also B follows A unless there is also the edge from B to A. Therefore, in this paper, we study how current deception optimization measures can be recast to the case of directed networks. Specifically, we study modularity, safeness, and permanence through the lens of edge directions by taking into account the characteristics of directed networks. Returning to the above example, if A is a user with a few followers but follows many other users, one will naturally expect that the edge from A to B is more likely than that from B to A. Thus when this latter edge is present, it has to be considered differently from the edge from A to B.

From a practical perspective, users who want to use these deception tools still need to figure out their peculiarities. *The second challenge that we face in this paper is to offer an overview of the performance of the state-of-the-art deception techniques on a variety of networks.* We analyze the state of the art in terms of: *(i)* ability to *quantify the level of hiding* of $\mathscr{C}$ inside $\overline{C} = \{C_1, C_2, ...C_k\}$; *(ii) scalability* in terms of running time; *(iii) practical applicability*, that is, how to implement the updates in a real network. As a by-product, we make available the implementation of these techniques in a library available online[1].

[1] https://communitydeception.wordpress.com/.

1.1 Contributions and outline

This paper studies the community deception problem from two different angles. On the one hand, we provide a systematic analysis of the state-of-the-art community deception techniques in undirected networks. On the other hand, we study the novel problem of community deception in directed networks. Specifically, we make the following main contributions:

1. A comprehensive overview of deception techniques in undirected networks under a common framework.
2. A study of community deception in directed networks. This problem has not been studied before. We show that dealing with edge directions brings some non-trivial issues since it becomes more involved to tell apart whether a certain category of edge update is convenient deception-wise. In particular, edge directions bring a further intrinsic difficulty toward deception since, in directed networks, only one direction of edges can be managed, that is, edges that $\mathscr{C}$'s members can directly add or delete.
3. An experimental evaluation along three main dimensions: performance in terms of deception score, preservation of the community structure, and running time. As a by-product, we make available a modular Python library where new deception techniques can be easily plugged.

This paper extends a previous paper published in CNA 2021 Fionda and Pirrò (2022). The present paper substantially differs in the following main respects. We expanded the introduction to the community deception problem. We introduce the novel problem of community deception in directed networks (Sect. 4). We conducted a completely new experimental evaluation for directed networks, including three novel deception techniques as well as community detection algorithms specifically devised for directed networks.

The remainder of the paper is organized as follows. Section 2 introduces the community deception problem. Section 3 reviews the state-of-the-art community deception techniques in undirected networks. Section 4 introduces the community deception problem in directed networks. Section 5 reports on an experimental evaluation. We conclude in Sect. 6.

2 Background

The goal of community deception is to design algorithms to deceive community detection algorithms. In particular, *given a community $\mathscr{C}$, the goal is to determine a set β of*

Table 1 Notation table undirected deception techniques

Symbol	Meaning	Formula
δ	Product of the squared total degree of a community structure	$\sum_{C_i \in \overline{C}} \delta(C_i)^2$
η	Sum of the intra-edges of a community structure	$\sum_{C_i \in \overline{C}} \lvert E(C_i)\rvert$
m	Number of edges in the network $G = (V, E)$	$\lvert E\rvert$
$\delta(C_i)$	Total degree of community C_i	$\sum_{u \in C_i} \delta(u)$
$\delta(u)$	Degree of node u	$\lvert\{(u, v) : (u, v) \in E\}$
$V^u(\mathscr{C})$	Set of nodes reachable from u passing only via nodes in $\mathscr{C}$, excluding u itself	
$E(C_i)$	Set of intra-community edges of community C_i	$\{(u, v) : u, v \in C_i\}$
$\widetilde{E}(C_i)$	Set of inter-community edges of community C_i	$\{(u, v) : u \in C_i, v \notin C_i\}$
$E(u, C_i)$	Set of intra-community edges of node u belonging to the community C_i	$\{(u, v) : u, v \in C_i\}$
$\widetilde{E}(u, C_i)$	Set of inter-edges of node u belonging to the community C_i	$\{(u, v) : u \in C_i, v \notin C_i\}$

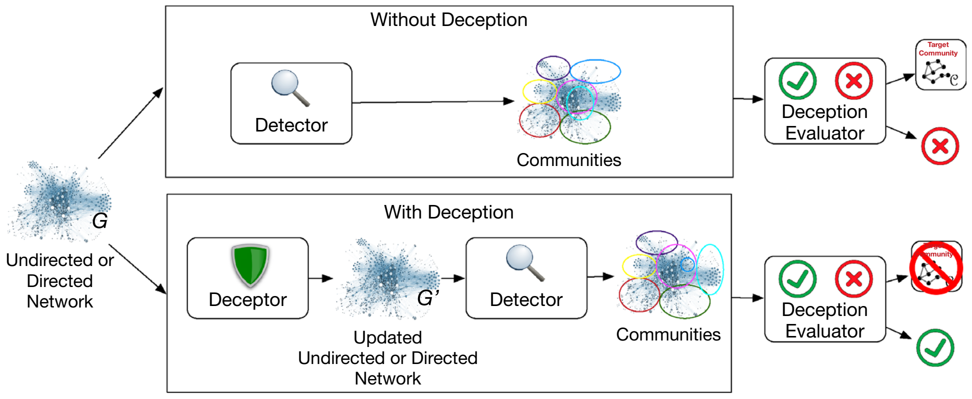


Fig. 1 Community Deception: a general framework

edge updates so that $\mathscr{C}$ will not be discovered by community detection algorithms. A network $G = (V, E)$ is an undirected graph that includes a set of $n{:=}|V|$ vertices and $m{:=}|E|$ edges. We denote by $deg(u){=}|\{(u, v) \in E\}|$ the degree of u. The set of communities (i.e., a community structure), discovered by some community detection algorithm $\mathscr{A}_D$ is denoted by $\overline{C}$ $=\{C_1, C_2, ...C_k\}$; $C_i \in \overline{C}$ denotes the *i-th* community.

Given a community C_i, we distinguish between intra-community edges and inter-community edges. The set of intra-community edges $E(C_i)$ is the set of edges of the form $(u, v) : u, v \in C_i$, where both endpoints are members of C_i. The set of inter-community edges $\widetilde{E}(C_i)$ is the set of edges of the form $(u, v) : u \in C_i, v \notin C_i$, where one of the endpoint is external to C_i. Given a community C_i and a node $u \in C_i$, we indicate by $E(C_i, u)$ (resp., $\widetilde{E}(C_i, u)$) the set of intra-community (respectively, inter-community) edges of u. The degree of a community is denoted by: $\delta(C_i){=}\sum_{u \in C_i} \delta(u)$, where $\delta(u)$ is the degree of node v. Given a network $G = (V, E)$, we indicate by E^+ and E^- the set of edge additions and deletions, respectively, to be applied on G. Table 1 summarizes the notation discussed above.

2.1 Problem statement

Figure 1 reports a general deception framework. Given a network G, the *Detector* module (implementing a community detection algorithm) analyzes G to discover communities. The underlying assumption that stresses the need for deception techniques is that disclosing (part of) $\mathscr{C}$ leads to privacy leaks and should be avoided. The *Deceptor* module (implementing a community deception algorithm) analyzes the network G and suggests a set of edge rewiring involving nodes in $\mathscr{C}$ that help $\mathscr{C}$'s members to be hidden as a group. To find the best set of edge updates, the *Deceptor* is based on some function to be optimized such as modularity (minimization) as in the case of `DICE` Waniek et al. (2018), node safeness (maximization) as for SAFDEC (Fionda and Pirrò 2018), or permanence (maximization) as for NEURAL (Mittal et al. 2021). After applying the modifications suggested by the *Deceptor* and obtaining a new network G', the desideratum is that the *Detector* by analyzing G' is no more able to discover $\mathscr{C}$; ideally because $\mathscr{C}$'s members are scattered among different communities. In order to quantify the privacy leak caused by the *Detector*, the *Deception Evaluator* module leverages some score such as the *Deception Score* (Fionda and Pirrò 2018) reported in Definition 1.

Definition 1 *(Deception Score)*. Given a community $\mathscr{C}$ and a community structure $\overline{C} = \{C_1, C_2, ...C_k\}$ found by some Detector, the community deception score is defined as: $\mathscr{H}(\mathscr{C}, \overline{C})$=

$$\left(1 - \frac{|S(\mathcal{C})| - 1}{|\mathcal{C}| - 1}\right) \times \left(\frac{1}{2}\left(1 - \max_{C_i \in \overline{C}}\{\mathcal{R}\left(C_i, \mathcal{C}\right)\} + \frac{1}{2}\left(1 - \frac{\sum_{C_i \cap \mathcal{C} \neq \emptyset} \mathcal{P}(C_i, \mathcal{C})}{|C_i \cap \mathcal{C} \neq \emptyset|}\right)\right)\right)$$

where $|S(\mathscr{C})|$ is the number of connected components in the subgraph induced by $\mathscr{C}$'s members; $\mathcal{R}(C_i, \mathscr{C}) = \frac{\#\mathscr{C}\text{'s members in } C_i \text{ found by } \mathscr{A}_D}{|\mathscr{C}|} \forall C_i \in \overline{C}$ is the *recall* of the Detector $\mathscr{A}_D$ wrt a target community $\mathscr{C}$; $\mathcal{P}(C_i, \mathscr{C}) = \frac{\#\mathscr{C}\text{'s members in } C_i \text{ found by } \mathscr{A}_D}{|C_i|} \forall C_i \cap \mathscr{C} \neq \emptyset$ is the *precision*.

One way to approach the community deception problem would be to work directly with the deception score $\mathscr{H}$. However, this would require knowing how the community detection algorithm $\mathscr{A}_D$, that generated the community structure $\overline{C} = \{C_1, C_2, ...C_k\}$ used in the computation of $\mathscr{H}$, works. What is needed is a way to increase $\mathscr{H}$ by treating a community detection algorithm $\mathscr{A}_D$ as a black box. One can model community deception in terms of the following optimization problem to tackle this challenge.

Problem 2 [**Community Deception**] Given a network G=(V, E), a target community $\mathscr{C} \subseteq V$ and a budget β of updates, solving the community deception problem amounts at solving the following optimization problem:

$$\underset{G'}{\operatorname{argmax}}\{\phi(G, G', \mathscr{C})\}$$

where G'=(V, E') and E'=$(E \cup E^+) \setminus E^-$ and: $E^+ \subseteq \{(u, v) : u \in \mathscr{C} \vee v \in \mathscr{C}, (u, v) \notin E\}$, $E^- \subseteq \{(u, v) : u \in \mathscr{C} \vee v \in \mathscr{C}, (u, v) \in E\}$, and $|E^+| + |E^-| \leq \beta$.

In the above formulation, $\phi(G, G', \mathscr{C})$ is a function that models a community deception algorithm while the budget β limits the number of possible updates. In particular, the function $\phi(G, G', \mathscr{C})$ computes a numerical value indicating the improvement in the network G' (obtained by applying β modifications) in terms of the hiding of nodes in $\mathscr{C}$. Ideally, the argmax function selects the network G' (and, thus a set of β modifications) where the level of hiding is maximized. The crucial difference between the deception function ϕ and the deception score $\mathscr{H}$ is that the former picks the β changes that maximize ϕ, while $\mathscr{H}$ quantifies (in an axiomatic way) the desirable property that the target community $\mathscr{C}$ is hidden inside $\overline{C} = \{C_1, C_2, ...C_k\}$ (Fionda and Pirrò 2018).

3 Related work

Community deception (Fionda and Pirrò 2018) or hiding (Waniek et al. 2018) studies how to hide a target community $\mathscr{C}$ inside a community structure from community detection algorithms. The idea is to find the best (deception-wise) set of edge updates by optimizing some functions. In what follows, we review the state of the art. The notation for the various deception techniques discussed in this section is summarized in Table 1.

3.1 Modularity-based deception

Waniek et al. (2018) and Fionda and Pirrò (2018) devise deception optimization functions based on (Newman 2006).

Definition 3 [*Modularity*] Given a network G, the modularity of the partition of this network into communities $\overline{C}$ =$\{C_1, C_2, ...C_k\}$ is given by:

$$\mathcal{M}_G(\overline{C}) = \frac{\eta}{m} - \frac{\delta}{4m^2} \tag{1}$$

where η= $\sum_{C_i \in \overline{C}} |E(C_i)|$ and δ= $\sum_{C_i \in \overline{C}} \delta(C_i)^2$.

The intuition behind using modularity for deception can be summarized as follows: community detection quality is related to the value of modularity, the higher, the better. Then, by minimizing modularity wrt edge updates performed by $\mathscr{C}$'s members should lead community detection algorithms astray. In particular modularity-based deception maximizes the modularity loss $\mathcal{ML}$=$\mathcal{M}_G(\overline{C}) - \mathcal{M}_{G'}(\overline{C})$.

In what follows, we focus on the approach described in Fionda and Pirrò (2018) since Waniek et al.'s strategy does not always bring a modularity loss and thus can fail to contribute to the hiding of the members of $\mathscr{C}$ inside the community structure.

Intra-edge addition. The modularity loss of an intra-community edge addition (u, w) s.t. $u, w \in C_i$ and $\{u, v\} \cap \mathscr{C} \neq \emptyset$ giving $G' = (V, E \cup \{(u, w)\})$ is the following:

$$\mathcal{M}_G(\overline{C}) - \mathcal{M}_{G'}(\overline{C}) = \frac{\eta - m}{m(m+1)} + \frac{4m^2(\delta(C_i)+1) - \delta(2m+1)}{4m^2(m+1)^2}.$$

Inter-edge addition. The modularity loss of an inter-community edge addition (u, w): $u \in C_i \cap \mathscr{C}, w \in C_j$, with $C_j \neq C_i$ giving $G' = (V, E \cup \{(u, w)\})$ brings the following potential modularity loss:

$$\mathcal{M}_G(\overline{C}) - \mathcal{M}_{G'}(\overline{C}) = \frac{\eta}{m(m+1)} + \frac{2m^2(\delta(C_i) + \delta(C_j) + 1) - \delta(2m+1)}{4m^2(m+1)^2}$$

Intra-edge deletion. The modularity loss of an intra-edge deletion (u, w) s.t. $u, w \in C_i$ and $\{u, v\} \cap \mathscr{C} \neq \emptyset$ giving $G' = (V, E \setminus \{(u, w)\})$ is the following:

$$\mathcal{M}_G(\overline{C}) - \mathcal{M}_{G'}(\overline{C}) = \frac{\eta - m}{m(m-1)} + \frac{\delta(2m-1) - 4m^2(\delta(C_i)-1)}{4m^2(m-1)^2}.$$

Inter-edge deletion. The modularity loss of an inter-community edge deletion (u, w): $u \in C_i \cap \mathscr{C}, w \in C_j$, with $C_j \neq C_i$ giving $G' = (V, E \setminus \{(u, w)\})$ is the following:

$$\mathcal{M}_G(\overline{C}) - \mathcal{M}_{G'}(\overline{C}) = \frac{\delta(2m-1) - 2m^2(\delta(C_i) + \delta(C_j) + 1)}{4m^2(m-1)^2} - \frac{\eta}{m(m-1)}$$

3.2 Safeness-based deception

Safeness-based deception (Fionda and Pirrò 2018) has been introduced to correct for some drawbacks of modularity-based deception. In particular, with modularity-based deception, one needs to know the entire community structure to pick the best edge update (that depends on the degree of the community toward which a new edge should be inserted). Safeness-based deception only requires information that can be obtained from $\mathscr{C}$'s members.

Definition 4 *(Node Safeness)* Let $G = (V, E)$ be a network, $\mathscr{C} \subseteq V$ a community, and $u \in \mathscr{C}$ a member of $\mathscr{C}$. The safeness of u in G is defined as:

$$\sigma(u, \mathscr{C}) := \tau \frac{|V^u(\mathscr{C})| - |E(u, \mathscr{C})|}{|\mathscr{C}| - 1} + \chi \frac{|\widetilde{E}(u, \mathscr{C})|}{\delta(u)} \tag{2}$$

where $V^u(\mathscr{C}) \subseteq \mathscr{C}$ is the set of nodes reachable from u passing only via nodes in $\mathscr{C}$, $E(u, \mathscr{C})$ (resp., $\widetilde{E}(u, \mathscr{C})$) is the set of intra-$\mathscr{C}$ (resp., inter-$\mathscr{C}$) edges, $\tau, \chi > 0$, and $\eta + \chi$=1.

Definition 5 *(Community Safeness)* Given a network $G = (V, E)$ and a community $\mathscr{C} \subseteq V$, the safeness of $\mathscr{C}$ is defined as: $\sigma(\mathscr{C})$=$\sum_{u \in \mathscr{C}} \sigma(u, \mathscr{C}) / |\mathscr{C}|$

This approach instantiates the function ϕ to be the safeness gain $\xi_{\mathscr{C}}$=$\sigma(\mathscr{C}')$-$\sigma(\mathscr{C})$. It adopts a greedy strategy that, at each step, chooses the edge update that gives the highest $\xi_{\mathscr{C}}$. Therefore, the goal is to understand what kind of update is more profitable safeness-wise (Fionda and Pirrò 2018).

Intra-edge addition. An intra-$\mathscr{C}$ edge addition (u, w) s.t. $\{u, w\} \subset \mathscr{C}$ can increase the safeness of the community only if the edge connects previously disconnected portions of $\mathscr{C}$. The possible safeness gain is:

$$\sum_{v \in C_u \setminus \{u\}} \tau \frac{|C_w|}{2(|\mathscr{C}| - 1)} + \sum_{v \in C_w \setminus \{w\}} \tau \frac{|C_u|}{2(|C_w| - 1)} + \tau \frac{|C_w| - 1}{2(|\mathscr{C}| - 1)} + \tau \frac{|C_u| - 1}{2(|\mathscr{C}| - 1)} + - \chi \frac{|\widetilde{E}(u, \mathscr{C})|}{2\delta(u)(\delta(u)+1)} - \chi \frac{|\widetilde{E}(w, \mathscr{C})|}{2\delta(w)(\delta(w)+1)}$$

where C_u (C_w, respectively) is the connected component of $\mathscr{C}$ to which u (w, respectively) belongs before the edge addition.

Inter-edge addition. The best inter-$\mathscr{C}$ edge addition (u, w) s.t. $u \in \mathscr{C}$ and $w \notin \mathscr{C}$ giving G'=$(V, E \cup \{\mathbf{e}_u w\})$ is given by nodes $u \in \text{argmin}\{\frac{|\widetilde{E}(u, \mathscr{C})|}{\delta(u)}\}$. The safeness gain is:

$$\chi \frac{|\widetilde{E}(u, \mathscr{C})|}{\delta(u) + 1} - \chi \frac{|\widetilde{E}(u, \mathscr{C})|}{\delta(u)}$$

Intra-edge deletion. Intra-$\mathscr{C}$ edges deletions do not always correspond to a safeness gain. The best possible intra-$\mathscr{C}$ edge deletion (u, w): $u, w \in \mathscr{C}$ safeness gain occurs when the value of the following formula is maximum.

$$\chi \frac{|\widetilde{E}(u, \mathscr{C})|}{2\delta(u)(\delta(u)-1)} + \chi \frac{|\widetilde{E}(w, \mathscr{C})|}{2\delta(w)(\delta(w)-1)}$$

Inter-edge deletion. An inter-$\mathscr{C}$ edge deletion (u, w): $u \in \mathscr{C}, w \notin \mathscr{C}$ always corresponds to a safeness decrease.

3.3 Permanence-based deception

Mittal et al. (2021) devised NEURAL, a permanence-based deception strategy, which aims at reducing permanence of the network wrt $\mathscr{C}$. Permanence (2016) is a vertex-centric metric that quantifies the containment of a node u in a network community C:

$$\text{Perm}(u, G) = \frac{|E(u, C)|}{E^{\max}(u)} \times \frac{1}{\delta(u)} - (1 - C_{\text{in}}(u)) \tag{3}$$

where $E^{\max}(u)$ is the maximum number of connections of u to the same neighboring communities, $C_{\text{in}}(u)$ the fraction of actual and possible number of edges among the internal neighbors of u. The permanence for a network G is then defined as $Perm(G) = \frac{\sum_{u \in V} \text{Perm}(u)}{|V|}$. NEURAL instantiates the function ϕ to be the permanence loss $\mathcal{P}_l = \text{Perm}(G) - \text{Perm}(G')$.

Intra-edge addition. An intra-community edge addition (u, w) s.t. $u, w \in C_i$ and $\{u, v\} \cap \mathscr{C} \neq \emptyset$ does not always ensure $\mathcal{P}_l$ >0. The possible permanence loss for node u (a similar loss can be also computed for node w) is:

$$\frac{E(u, C_i) \cdot (E(u, C_i) + 1)}{E^{\max}(u) \cdot \delta(u) \cdot (\delta(u) + 1)} + C_{\text{in}}(u) - C'_{\text{in}}(u)$$

Inter-edge addition. Adding an inter-community edge (u, w) where $u \in C_i \cap \mathscr{C}$ and $w \in C_j$, such that $C_j \neq C_i$, always results in $\mathcal{P}_l$ >0. The loss is more if C_j is the community that provides the maximum external pull for node u. In such a case, the permanence loss is:

$$\frac{E(u, C_i)(1 + E^{\max}(u) + \delta(u))}{E^{\max}(u) \cdot \delta(u) \cdot (E^{\max}(u) + 1) + (\delta(u) + 1)}$$

Intra-edge deletion. An intra-community edge deletion (u, w) s.t. $u, w \in C_i$ and $\{u, v\} \cap \mathscr{C} \neq \emptyset$, always gives $\mathcal{P}_l > 0$

Inter-edge deletion. Deleting an inter-community edge (u, w) where $u \in C_i \cap \mathscr{C}$ and $w \in C_j$ such that $C_j \neq C_i$ never results in $\mathcal{P}_l$ >0.

Other pieces of work (e.g., Nagaraja 2010; Magelinski et al. 2021; Liu et al. 2019) have studied a different problem, that is hiding (or at least changing) the whole community structure instead of a target community. Another line of research (e.g., Jia et al. 2020) has studied countermeasures, that is, the robustness to attacks.

We observe that all these pieces of related work have focused on undirected networks, although many real-world networks (e.g., social networks like Twitter) are directed. As pointed out by fundamental studies (e.g., Leicht and Newman 2008; Malliaros and Vazirgiannis 2013; Fortunato 2010), edge direction can play a fundamental role in revealing more accurate communities in networks. Moreover, devising direction-oblivious deception techniques (e.g., treating directed networks as undirected) to escape from direction-aware detection algorithms like `leiden` (Traag et al. 2019) may undermine the overall goal of protecting a community in a directed network.

4 Deception in directed networks

In this section, we study the community deception problem in directed networks. Although many real-world networks (e.g., Twitter) are intrinsically directed, state-of-the-art deception techniques have only focused on undirected networks. One way to solve the problem would be to ignore edge directions simply. However, this is limiting for at least two reasons. First, meaningful information about edge direction is discarded. In a network like Twitter, the fact that A follows B does not imply that B follows A. Second, as community detection has evolved to take into account edge directions (e.g., Leicht and Newman 2008; Traag et al. 2019), we believe that community deception should evolve to play a fairer game.

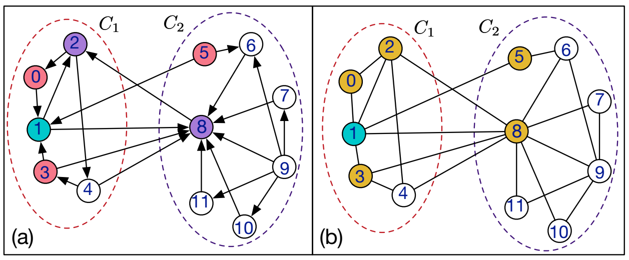


Fig. 2 Example of a directed network (**a**) and its undirected version (**b**)

By referring to the framework reported in Fig. 1, in this new setting, the input network G is now a directed network, the *Detector* module can implement specific community detection algorithms proposed to work on directed networks, and the *Deceptor* will take edge direction into account when suggesting a set of directed edge updates. Therefore, the *Deceptor* can only consider edge additions and deletions whose source node is in the target community.

We show how to derive the counterpart for each of the three main deception strategies available for undirected networks in the directed case. Moreover, to understand the importance of taking edge directions into account and the need for introducing deception approaches specifically tailored to work on directed networks, we will discuss the network reported in Fig. 2. The figure reports the same network when edge direction is considered (Fig. 2(a)) and when it is neglected (Fig.2(b)). We suppose that the *Detector* has identified two communities and the target community is C_1 = {0, 1, 2, 3, 4} and it has been completely disclosed.

4.1 Directed modularity-based deception

In this section, we investigate how the modularity-based deception analyzed in Section 3.1 can be adapted to work on directed networks. In particular, we consider a slightly modified version of the metric described by Leicht and Newman (2008). The notation used to define directed modularity is summarized in Table 2.

Definition 6 Let $G = (V, E)$ be a directed network, the directed modularity of the partition of this network into communities $\overline{C}$={$C_1, C_2, ...C_k$} is given by:

Table 2 Notation table for directed modularity

Symbol	Meaning	Formula
$\vec{\delta}$	Product of the total output degree and total input degree of a community structure	$\vec{\delta} = \delta_o \cdot \delta_i$
δ_o	Total output degree of a community structure $\overline{C}$	$\sum_{C_i \in \overline{C}} \delta_o(C_i)$
δ_i	Total input degree of a community structure $\overline{C}$	$\sum_{C_i \in \overline{C}} \delta_i(C_i)$
$\delta_o(C_i)$	Total output degree of community C_i	$\sum_{u \in C_i} \delta_o(u)$
$\delta_i(C_i)$	Total input degree of community C_i	$\sum_{u \in C_i} \delta_i(u)$
$\delta_o(u)$	Output degree of node u	$\lvert\{(u,v) : (u,v) \in E\}\rvert$
$\delta_i(u)$	Input degree of node u	$\lvert\{(v,u) : (v,u) \in E\}\rvert$

$$\overrightarrow{\mathcal{M}}_G(\overline{C}) = \frac{\eta}{m} - \frac{\vec{\delta}}{m^2}$$

where $\eta = \sum_{C_i \in \overline{C}} |E(C_i)|$, $m = |E|$, and $\vec{\delta} = \delta_o \cdot \delta_i = \sum_{C_i \in \overline{C}} \delta_o(C_i) \cdot \sum_{C_i \in \overline{C}} \delta_i(C_i) = \sum_{C_i \in \overline{C}} \sum_{u \in C_i} \delta_o(u) \cdot \sum_{C_i \in \overline{C}} \sum_{u \in C_i} \delta_i(u)$.

In terms of the general deception formulation (see Definition 2) the function ϕ can be instantiated to be the directed modularity loss $\overrightarrow{\mathcal{ML}} = \overrightarrow{\mathcal{M}}_G(\overline{C}) - \overrightarrow{\mathcal{M}}_{G'}(\overline{C})$. We will analyze the impact of the different types of edge updates on the directed modularity loss.

Example 7 Consider the network in Fig. 2. The directed modularity of the network in Fig. 2(a) is:

$$\begin{aligned}\overrightarrow{\mathcal{M}}_G(\overline{C}) = \frac{\eta}{m} - \frac{\vec{\delta}}{m^2} &= \frac{|E(C_1)| + |E(C_2)|}{21} \\ &\quad - \frac{(\delta_o(C_1) + \delta_o(C_2)) \cdot (\delta_i(C_1) + \delta_i(C_2))}{21^2} \\ &= \frac{6+10}{21} - \frac{(9+11)\cdot(7+13)}{21^2} \\ &= \frac{16}{21} - \frac{20 \cdot 20}{21^2} = -0.145\end{aligned}$$

If we consider now its undirected version, reported in Fig. 2(b), we obtain the following modularity score:

$$\mathcal{M}_G(\overline{C}) = \frac{\eta}{m} - \frac{\delta}{4m^2} = \frac{|E(C_1)| + |E(C_2)|}{21} - \frac{\delta(C_1)^2 + \delta(C_2)^2}{4 \cdot 21^2}$$
$$= \frac{6+10}{21} - \frac{16^2 + 25^2}{4 \cdot 21^2} = \frac{16}{21} - \frac{881}{4 \cdot 21^2} = 0.262$$

From this simple example, it is clear how neglecting edge directions can significantly change the sense of "goodness" assigned to a community structure. Indeed, in the case of directed modularity, the obtained value is lower than 0, indicating the absence of a community structure. In contrast, in the undirected case, the value is higher than 0, indicating the possible presence of a community structure.

4.1.1 Intra-edge addition

An intra-community edge addition (u, w) s.t. $u, w \in C_i$ and $\{u, v\} \cap \mathscr{C} \neq \emptyset$ giving an updated network $G' = (V, E \cup \{(u, w)\})$ does not always correspond to a directed modularity loss. Indeed, the modularity loss is:

$$\begin{aligned}\overrightarrow{\mathcal{ML}} &= \overrightarrow{\mathcal{M}}_G(\overline{C}) - \overrightarrow{\mathcal{M}}_{G'}(\overline{C}) \\ &= \left[\frac{\eta}{m} - \frac{1}{m^2}\left(\delta_o(C_i) \cdot \delta_i(C_i) + \sum_{C_j \in \overline{C} \setminus \{C_i\}} \delta_o(C_j) \cdot \delta_i(C_j)\right)\right] + \\ &\quad - \left[\frac{\eta+1}{m+1} - \frac{1}{(m+1)^2}\right. \\ &\quad \left.\left((\delta_o(C_i) + 1)(\delta_i(C_i) + 1) + \sum_{C_j \in \overline{C} \setminus \{C_i\}} \delta_o(C_j) \cdot \delta_i(C_j)\right)\right] = \\ &= \frac{m-\eta}{m(m+1)} + \frac{1}{m^2(m+1)^2} \\ &\quad \left[m^2(\delta_o(C_i) + \delta_i(C_i) + 1) - (2m+1) \sum_{C_j \in \overline{C}} \delta_o(C_j) \cdot \delta_i(C_j)\right]\end{aligned}$$

With few algebraic manipulations, we obtain the following inequality $\overrightarrow{\mathcal{ML}} < \frac{1}{(m+1)^2}\left(\eta - m + \delta_o(C_i) + \delta_i(C_i) - \frac{2m+1}{m^2} \sum_{C_j \in \overline{C}} \delta_o(C_j) \cdot \delta_i(C_j)\right)$. Since the last term in the bracket is negligibly small, we can conclude that $\overrightarrow{\mathcal{ML}}$ is negative if $\eta - m \geq \delta_o(C_j) + \delta_i(C_i)$.

4.1.2 Inter-edge addition

An inter-community edge addition (u, w) s.t. $u \in C_i \cap \mathscr{C}$ and $w \in C_w \neq C_i$ giving an updated network $G' = (V, E \cup \{(u, w)\})$ does not always correspond to a directed modularity loss. Indeed, the directed modularity loss is:

$$\overrightarrow{\mathcal{ML}} = \frac{\eta}{m} - \frac{1}{m^2}\Bigg(\big(\delta_o(C_i)\delta_i(C_i)\big) + \big(\delta_o(C_w)\delta_i(C_w)\big) + \sum_{C_j\in\bar{C}\setminus\{C_i,C_w\}} \delta_o(C_j)\delta_i(C_j)\Bigg) + - \Bigg[\frac{\eta}{m+1} - \frac{1}{(m+1)^2} \Bigg((\delta_o(C_i)+1)\delta_i(C_i) + \delta_o(C_w)(\delta_i(C_w)+1) + \sum_{C_j\in\bar{C}\setminus\{C_i,C_w\}} \delta_o(C_j)\delta_i(C_j)\Bigg)\Bigg]$$

By some algebraic manipulation, this can be reduced to:

$$\overrightarrow{\mathcal{ML}} = \frac{1}{m^2(m+1)^2}\Bigg[\eta(m^2+m) + m^2\big(\delta_i(\mathscr{C}) + \delta_o(C_w)\big) - (2m+1)\sum_{C_j\in\bar{C}} \delta_o(C_j)\delta_i(C_j)\Bigg]$$

Then, we can conclude that $\overrightarrow{\mathcal{ML}}$ is positive if and only if the term inside square brackets is positive. That is if the following inequality holds:

$$\eta(m^2+m) + m^2\big[\delta_i(C_i) + \delta_o(C_w)\big] \geq (2m+1)\sum_{C_j\in\bar{C}} \delta_o(C_j)\delta_i(C_j)$$

4.1.3 Intra-edge deletion

An intra-community edge deletion (u, w) s.t. $u, w \in C_i$ and $\{u, v\} \cap \mathscr{C} \neq \emptyset$ giving an updated network $G' = (V, E \setminus \{(u, w)\})$ does not always correspond to a directed modularity loss. Indeed, the modularity loss is:

$$\overrightarrow{\mathcal{ML}} = \overrightarrow{\mathcal{M}}_G(\overline{C}) - \overrightarrow{\mathcal{M}}_{G'}(\overline{C}) = \Bigg[\frac{\eta}{m} - \frac{1}{m^2}\Bigg(\delta_o(C_i)\cdot\delta_i(C_i) + \sum_{C_j\in\bar{C}\setminus\{C_i\}} \delta_o(C_j)\cdot\delta_i(C_j)\Bigg)\Bigg] + -\Bigg[\frac{\eta-1}{m-1} - \frac{1}{(m-1)^2}\Bigg((\delta_o(C_i)-1)(\delta_i(C_i)-1) + \sum_{C_j\in\bar{C}\setminus\{C_i\}} \delta_o(C_j)\cdot\delta_i(C_j)\Bigg)\Bigg] =$$

$$= \frac{m-\eta}{m(m-1)} + \frac{1}{m^2(m-1)^2}\Bigg[-m^2(\delta_o(C_i) + \delta_i(C_i) - 1) + (2m-1)\sum_{C_j\in\bar{C}} \delta_o(C_j)\cdot\delta_i(C_j)\Bigg]$$

With few algebraic manipulations, we can conclude that $\overrightarrow{\mathcal{ML}} > \frac{1}{(m-1)^2}\Big(m - \eta + 1 - \delta_o(C_i) - \delta_i(C_i)\Big)$. Such inequality shows that $\overrightarrow{\mathcal{ML}}$ will be positive if the term inside the brackets is positive and thus if $m - \eta \geq \delta_o(C_i) + \delta_i(C_i) - 1$.

4.1.4 Inter-edge deletion

An inter-community edge deletion (u, w) s.t. $u \in C_i \cap \mathscr{C}$ and $w \in C_w \neq C_i$ giving an updated network $G' = (V, E \setminus \{(u, w)\})$ does not always correspond to a directed modularity loss. Indeed, the modularity loss is:

$$\overrightarrow{\mathcal{ML}} = \overrightarrow{\mathcal{M}}_G(\overline{C}) - \overrightarrow{\mathcal{M}}_{G'}(\overline{C}) = \Bigg[\frac{\eta}{m} - \frac{1}{m^2}\Bigg(\sum_{C_j\in\bar{C}} \delta_o(C_j)\cdot\delta_i(C_j)\Bigg)\Bigg] + - \Bigg[\frac{\eta-1}{m-1} - \frac{1}{(m-1)^2}\Bigg((\delta_o(C_i)-1)\delta_i(C_i) + \delta_o(C_w)(\delta_i(C_w)-1) + \sum_{C_j\in\bar{C}\setminus\{C_i,C_w\}} \delta_o(C_j)\delta_i(C_j)\Bigg)\Bigg]$$

$$= \frac{1}{m^2(m-1)^2}\Bigg[-\eta m^2 + \eta m - m^2\delta_i(C_i) - m^2\delta_o(C_w) + (2m-1)\sum_{C_j\in\bar{C}} \delta_o(C_j)\delta_i(C_j)\Bigg]$$

By looking at the above formula, we can conclude that $\overrightarrow{\mathcal{ML}}$ is negative if the term inside the square bracket is negative, that is:

$$-\eta m^2 + \eta m - m^2\delta_i(C_i) - m^2\delta_o(C_w) + (2m-1)\sum_{C_j\in\bar{C}} \delta_o(C_j)\delta_i(C_j) < 0.$$

4.2 Directed Safeness-Based Deception

Starting for the Safeness defined for undirected networks (see equation 2 and equation 5) we can study safeness in directed networks. The notation used to define directed safeness is summarized in Table 3.

We start by defining the safeness of a node in the directed case.

Table 3 Notation table for directed safeness

Symbol	Meaning	Formula
$V_o^u(\mathscr{C})$	Nodes in $\mathscr{C}$ (excluding u itself) reachable from u: *(i)* passing only via nodes in $\mathscr{C}$ and *(ii)* following directed paths originating from u	
$V_i^u(\mathscr{C})$	Nodes in $\mathscr{C}$ (excluding u itself) that can reach u: *(i)* passing only via nodes in $\mathscr{C}$ and *(ii) by following directed paths terminating in u*	
$E_o(u,\mathscr{C})$	Outgoing edges of u to members of $\mathscr{C}$	$\{(u,v) \mid v \in \mathscr{C}\}$
$E_i(u,\mathscr{C})$	Incoming edges of u from members of $\mathscr{C}$	$\{(v,u) \mid v \in \mathscr{C}\}$
$\widetilde{E}_o(u,\mathscr{C})$	Outgoing edges of u to non members of $\mathscr{C}$	$\{(u,v) \mid v \notin \mathscr{C}\}$
$\widetilde{E}_i(u,\mathscr{C})$	Incoming edges of u from non members of $\mathscr{C}$	$\{(v,u) \mid v \notin \mathscr{C}\}$
$\delta_o(u)$	Output degree of node u	$\lvert\{(u,v)\}\rvert$
$\delta_i(u)$	Input degree of node u	$\lvert\{(v,u)\}\rvert$

Definition 8 Consider a directed network $G = (V, E)$, a community $\mathscr{C} \subseteq V$, and a node $u \in \mathscr{C}$. The directed safeness of u in G is defined as:

$$\vec{\sigma}(u,\mathscr{C}) := \tau \frac{\left(|V_o^u(\mathscr{C})| - |E_o(u,\mathscr{C})|\right) + \left(|V_i^u(\mathscr{C})| - |E_i(u,\mathscr{C})|\right)}{2\cdot(|\mathscr{C}|-1)} + \chi\left(\frac{|\widetilde{E}_o(u,\mathscr{C})|}{\delta_o(u)} + \frac{|\widetilde{E}_i(u,\mathscr{C})|}{\delta_i(u)}\right)$$

The above formula split the two components of undirected safeness to take into account edge directions. Indeed, the $V^u(\mathscr{C})$ of the undirected case is split into two terms $V_o^u(\mathscr{C})$ and $V_i^u(\mathscr{C})$ that consider the portion of nodes in $\mathscr{C}$ that can be reached from u via directed paths originating from u and directed paths terminating in u, respectively. Similarly, $E_o(u,\mathscr{C})$ ($E_i(u,\mathscr{C})$, respectively) indicates the set of incoming (outgoing, respectively) edges linking u to other members of $\mathscr{C}$; $\widetilde{E}_o(u,\mathscr{C})$ ($\widetilde{E}_i(u,\mathscr{C})$, respectively) indicates the set of incoming (outgoing, respectively) edges linking u to nodes not in $\mathscr{C}$; and, $\delta_o(u)$ ($\delta_i(u)$, respectively) indicates the outgoing (incoming, respectively) degree of u.

Example 9 Consider again the network reported in Fig. 2 and let $\tau = \chi = \frac{1}{2}$. To discuss safeness, we will focus on node 1; in the directed network, it is connected by two outgoing edges to 2 and 8 and by three incoming edges from nodes 0, 3, and 5. Then, if we compute the directed safeness score of node 3, we obtain the following:

$$\begin{aligned}\vec{\sigma}(3,\mathscr{C}) :=& \frac{1}{2}\frac{\left(|V_o^1(\mathscr{C})| - |E_o(1,\mathscr{C})|\right) + \left(|V_i^1(\mathscr{C})| - |E_i(1,\mathscr{C})|\right)}{2\cdot(|\mathscr{C}|-1)} \\ &+ \frac{1}{2}\left(\frac{|\widetilde{E}_o(1,\mathscr{C})|}{\delta_o(1)} + \frac{|\widetilde{E}_i(1,\mathscr{C})|}{\delta_i(1)}\right) = \\ &= \frac{(4-1)+(4-2)}{2\cdot 2\cdot 4} + \frac{1}{2}\left(\frac{1}{2}+\frac{1}{3}\right) \\ &= 0.313 + 0.417 = 0.73\end{aligned}$$

If we compute the safeness score of node 1 on the undirected version of the network, we obtain:

$$\begin{aligned}\sigma(1,\mathscr{C}) :=& \frac{1}{2}\frac{|V^1(\mathscr{C})| - |E(1,\mathscr{C})|}{|\mathscr{C}|-1} \\ &+ \frac{1}{2}\frac{|\widetilde{E}(1,\mathscr{C})|}{\delta(1)} \\ =& \frac{4-3}{2\cdot 4} + \frac{2}{2\cdot 5} = 0.125 + 0.2 = 0.325\end{aligned}$$

On the one hand, in the directed version of the network, node 1 can reach all the community nodes via outgoing and incoming paths. On the other hand, one out of the three incoming edges of node 1 is an inter-community edge; and one of the two outgoing edges is an inter-community edge. When it comes to the undirected version of the network, it does not matter the direction in which the information can be transmitted; all edges are treated in the same way meaning that node 1 can reach all the other four nodes in C_1 by mean of its three intra-community edges and has two out of five inter-community edges. This causes the decrease of the safeness score from 0.73 to 0.325, meaning node 1 is more subjected to be discovered as a member of C_1 in the undirected version w.r.t the directed one.

Similar to the case of undirected networks, the directed community safeness can be defined by averaging the directed node safeness of the nodes in $\mathscr{C}$.

Definition 10 Given a directed network $G = (V, E)$ and a community $\mathscr{C} \subseteq V$, the directed safeness of $\mathscr{C}$ in G is defined as:

$$\vec{\sigma}(\mathscr{C}) = \sum_{u\in\mathscr{C}} \vec{\sigma}(u,\mathscr{C})/|\mathscr{C}| \tag{4}$$

In terms of the general deception formulation (see Definition 2) this approach instantiates the function ϕ to be the directed safeness gain $\vec{\xi}_{\mathscr{C}}$=$\vec{\sigma}(\mathscr{C}')$-$\vec{\sigma}(\mathscr{C})$. Then, in the following, we will discuss the impact of the different types of edge updates on the directed safeness score.

Table 4 Notation table for directed permanence

Symbol	Meaning	Formula
$E_o(u, C_i)$	Outgoing edges of u to members of C_i	$\{(u, v) \mid u \in C_i\}$
$E_i(u, C_i)$	Incoming edges of u from members of C_i	$\{(v, u) \mid v \in C_i\}$
$E_o^{max}(u)$	Maximum number of edges originated from u connecting u to a neighbour community	$\max_{C\in\overline{C}} \lvert\{(u, v) \mid v \in C\}\rvert$
$E_i^{max}(u)$	Maximum number of incoming edges of u connecting u to a neighbour community	$\max_{C\in\overline{C}} \lvert\{(v, u) \mid v \in C\}\rvert$
$C_{in}(u)$	Clustering coefficient of u's neighbours	$\frac{\lvert\{(v,w)\in E: v,w\in N_u\}\rvert}{\delta(u)(\delta(u)-1)}$
$\delta(u)$	Degree of u	$\lvert\{(u, v)\}\rvert + \lvert\{(v, u)\}\rvert$

4.2.1 Intra-edge addition

An intra-$\mathscr{C}$ edge addition (u, w) s.t. $u, w \in \mathscr{C}$ giving an updated network $G' = (V, E \cup \{(u, w)\})$ does not always introduce a directed safeness gain. Indeed, after the addition of the edge (u, w), if $w \notin V_o^u(\mathscr{C})$, u will be able to reach w and all the nodes in $V_o^w(\mathscr{C}) \setminus V_o^u(\mathscr{C})$. The same holds for w that will be able to reach u and all the nodes in $V_i^u(\mathscr{C}) \setminus V_i^w(\mathscr{C})$, if $u \notin V_i^w(\mathscr{C})$. Then, the possible increase of the safeness score for the nodes u and w is the following:

$$\vec{\sigma}(u, \mathscr{C}') - \vec{\sigma}(u, \mathscr{C}) = \tau\frac{(|V_o^w(\mathscr{C}) \setminus V_o^u(\mathscr{C})| + 1) - 1}{2(|\mathscr{C}| - 1)} + \chi\frac{|\widetilde{E}_o(u, \mathscr{C})|}{\delta_o(u)(\delta_o(u) + 1)}$$

$$\vec{\sigma}(w, \mathscr{C}') - \vec{\sigma}(w, \mathscr{C}) = \tau\frac{(|V_i^u(\mathscr{C}) \setminus V_i^w(\mathscr{C})| + 1) - 1}{2(|\mathscr{C}| - 1)} + \chi\frac{|\widetilde{E}_i(w, \mathscr{C})|}{\delta_i(w)(\delta_i(w) + 1)}$$

Obviously, such edge addition always results in a safeness decrease if $w \in V_o^u(\mathscr{C})$ and $u \in V_i^w(\mathscr{C})$.

4.2.2 Inter-edge addition

Any inter-$\mathscr{C}$ edge addition (u, w) s.t. $u \in \mathscr{C}$ and $w \notin \mathscr{C}$ giving an updated network G'=$(V, E \cup \{(u, w)\})$ always corresponds to a directed safeness increase. Indeed, the directed safeness node increase for u is

$$\begin{aligned}\vec{\sigma}(u, \mathscr{C}') - \vec{\sigma}(u, \mathscr{C}) &= \chi\frac{|\widetilde{E}_o(u, \mathscr{C})| + 1}{\delta_o(u) + 1} - \chi\frac{|\widetilde{E}_o(u, \mathscr{C})|}{\delta_o(u)} \\ &= \chi\frac{\delta_o(u) - |\widetilde{E}_o(u, \mathscr{C})|}{\delta_o(u)(\delta_o(u) + 1)}\end{aligned}$$

that is always greater or equals to 0 since $\delta_o(u) \geq |\widetilde{E}_o(u, \mathscr{C})|$. Note that the maximum increase in directed safeness happens for all the nodes u such that $u \in \text{argmin}\{\frac{|\widetilde{E}_o(u,\mathscr{C})|}{\delta_o(u)}\}$.

4.2.3 Intra-edge deletion

An intra-$\mathscr{C}$ edge deletion (u, w) s.t. $u, w \in \mathscr{C}$ giving an updated network $G' = (V, E \setminus \{(u, w)\})$ does not always bring a directed safeness gain. Indeed, let $V_o^{u-}(\mathscr{C})$ ($V_i^{w-}(\mathscr{C})$, respectively) be the nodes of $\mathscr{C}$ that cannot be reached by following directed paths originating from u (ending in w, respectively) after the deletion of the edge (u, w). Then, the possible increase of the safeness score for the nodes u and w is the following:

$$\vec{\sigma}(u, \mathscr{C}') - \vec{\sigma}(u, \mathscr{C}) = \tau\frac{1-|V_o^{u-}(\mathscr{C})|}{2(|\mathscr{C}| - 1)} + \chi\frac{|\widetilde{E}_o(u, \mathscr{C})|}{\delta_o(u)(\delta_o(u) - 1)}$$

$$\vec{\sigma}(w, \mathscr{C}') - \vec{\sigma}(w, \mathscr{C}) = \tau\frac{1 - V_i^{w-}(\mathscr{C})}{2(|\mathscr{C}| - 1)} + \chi\frac{|\widetilde{E}_i(w, \mathscr{C})|}{\delta_i(w)(\delta_i(w) - 1)}$$

Obviously, such edge addition always results in a safeness increase if $V_o^{u-}(\mathscr{C}) = \emptyset$ and $V_i^{w-}(\mathscr{C}) = \emptyset$, that is u and w will be able to reach exactly the same $\mathscr{C}$'s members if (u, w) is deleted.

4.2.4 Inter-edge deletion

Any inter-$\mathscr{C}$ edge deletion (u, w) s.t. $u \in \mathscr{C}$ and $w \notin \mathscr{C}$ giving an updated network G'=$(V, E \setminus \{(u, w)\})$ always corresponds to a directed safeness decrease. Indeed, the directed safeness node increase for u is

$$\begin{aligned}\vec{\sigma}(u, \mathscr{C}') - \vec{\sigma}(u, \mathscr{C}) &= \chi\frac{|\widetilde{E}_o(u, \mathscr{C})| - 1}{\delta_o(u) - 1} - \chi\frac{|\widetilde{E}_o(u, \mathscr{C})|}{\delta_o(u)} \\ &= \chi\frac{|\widetilde{E}_o(u, \mathscr{C})| - \delta_o(u)}{\delta_o(u)(\delta_o(u) + 1)}\end{aligned}$$

that is always greater or equals to 0 since $\delta_o(u) \geq |\widetilde{E}_o(u,\mathscr{C})|$.

4.3 Directed permanence-based deception

This section shows how permanence can be adapted to directed networks. The notation used in this section is summarized in Table 4. We start by defining directed node permanence.

Definition 11 Let $G=(V,E)$ be a directed network, and $u \in V$ a node in G. The directed node permanence of u is defined as:

$$\overrightarrow{\mathrm{Perm}}(u,G) := \left(\frac{|E_o(u,C_u)|}{2\cdot E_o^{\max}(u)\cdot\delta_o(u)} + \frac{|E_i(u,C_u)|}{2\cdot E_i^{\max}(u)\cdot\delta_i(u)}\right) - \left(1 - C_{in}(u)\right)$$

where $E_o(u,C_u)$ and $E_i(u,C_u)$ denote the internal outgoing and incoming connections of u within its own community C_u resp., $E_o^{\max}(u)$ and $E_i^{\max}(u)$ the maximum number of outgoing and incoming connections of u to its neighboring communities resp., $C_{in}(u)=\frac{|\{(v,w)\in E: v,w\in N_u\}|}{\delta(u)(\delta(u)-1)}$, where $N_u=\{v : (v,u)\in E \vee (u,v)\in E\}$ are the neighbors of u, the fraction of actual and possible number of edges among the internal neighbors of u (i.e., the clustering coefficient among the internal neighbors of u, where $\delta(u)$ indicates the total degree of u). As in the case of undirected networks, also for directed networks for all vertices u that do not have any inter- outgoing and/or incoming connections permanence is considered equal to the clustering coefficient, i.e., $\overrightarrow{\mathrm{Perm}}(u)=C_{\mathrm{in}}(u)$. Moreover, if the total number of internal outgoing and incoming connections of u is less than 2 the clustering coefficient $C_{\mathrm{in}}(u)$ is set to be 0.

Example 12 Consider again the directed network reported in Fig. 2(a), the directed permanence of node 1 is the following:

$$\overrightarrow{\mathrm{Perm}}(1,G) := \left(\frac{|E_o(1,C_1)|}{2\cdot E_o^{max}(1)\cdot\delta_o(1)} + \frac{|E_i(1,C_1)|}{2\cdot E_i^{max}(1)\cdot\delta_i(1)}\right) - \left(1-C_{in}(1)\right) = \left(\frac{1}{2\cdot1\cdot2}+\frac{2}{2\cdot1\cdot3}\right) - (1-\frac{3}{5\cdot4}) = (0.25+0.33)-(1-0.15) = -0.27$$

If we consider the undirected version of the same network, the permanence of node 1 will be the following:

$$\mathrm{Perm}(1,G) := \frac{I(1)}{E^{\max}(1)\cdot\delta(1)} - \left(1-C_{in}(1)\right) = \frac{3}{2\cdot5} - \left(1-\frac{2\cdot3}{5\cdot4}\right) = 0.3-(1-0.3) = -0.4$$

Then, for this example, the permanence of node 1 in the directed version is higher than the permanence in the undirected case, meaning that in the undirected version, node 1 is less committed to staying in C_1 than that in the directed version.

The directed permanence of w.r.t. a target community is defined as:

Definition 13 Given a directed network $G=(V,E)$ and a target community $\mathscr{C}$, the directed permanence of $\mathscr{C}$ in G is defined as:

$$\overrightarrow{\mathrm{Perm}}(\mathscr{C},G) = \frac{\sum_{u\in\mathscr{C}}\overrightarrow{\mathrm{Perm}}(u,G)}{|\mathscr{C}|} \tag{5}$$

In terms of the general deception formulation (see Definition 2) this approach instantiates the function ϕ to be the directed permanence loss $\overrightarrow{\mathcal{P}_l}$=$\overrightarrow{\mathrm{Perm}}(\mathscr{C},G)$-$\overrightarrow{\mathrm{Perm}}(\mathscr{C},G')$. Then, in the following we will discuss the impact of the different types of edge updates on the directed permanence loss:

4.3.1 Intra-edge addition

An intra-community edge addition (u, w) s.t. $u,w\in C_i$ and $\{u,v\}\cap\mathscr{C}\neq\emptyset$ giving an updated network $G'=(V,E\cup\{(u,w)\})$ does not always correspond to a directed permanence loss. In the following we will analyze the directed permanence loss for u, a similar reasoning will apply to w. Then, the directed permanence loss of u is:

$$\frac{|E_o(u,C_i)|}{2E_o^{\max}(u)\delta_o(u)} - (1-C_{in}(u)) - \frac{|E_o(u,C_i)|+1}{2E_o^{\max}(u)(\delta_o(u)+1)} + (1-C'_{in}(u)) = \frac{|E_o(u,C_i)|-\delta_o(u)}{2E_o^{\max}(u)\delta_o(u)\left(\delta_o(u)+1\right)} + (C_{in}(u)-C'_{\mathrm{in}}(u))$$

The first term (i.e., $\frac{|E_o(u,C_i)|-\delta_o(u)}{2E_o^{\max}(u)\delta_o(u)\left(\delta_o(u)+1\right)}$) is always lower or equals to zero since $\delta_o(u)\geq I_o(u)$. The second term (i.e., $(C_{in}(u)-C'_{in}(u))$) can be lower or greater than 0 depending on how the clustering coefficient change after the edge addition.

Note that the addition of the edge (u, w) will also increase the directed permanence of all the nodes v that have both u

and w in its neighborhood, since their clustering coefficient $C_{in}(v)$ will increase.

4.3.2 Inter-edge addition

Any inter-community edge addition (u, w) s.t. $u \in C_i \cap \mathscr{C}$ and $w \in C_w \neq C_i$ giving an updated network $G' = (V, E \cup \{(u, w)\})$ always results in a directed permanence loss. Consider first the case in which $E_o^{max}(u)$ does not change after the edge addition, then the directed permanence loss of u is:

$$\frac{|E_o(u, C_i)|}{2E_o^{\max}(u)\delta_o(u)} - \frac{|E_o(u, C_i)|}{2E_o^{\max}(u)(\delta_o(u)+1)} = \frac{|E_o(u, C_i)|}{2E_o^{max}(u)\delta_o(u)\left(\delta_o(u)+1\right)}$$

that is always greater or equals to 0 since $I_o(u)$ is at least 0.

Consider now the case in which $E_o^{\max}(u)$ changes after the edge addition, then the new value will be $E_o^{\max}(u)+1$. In this case the directed permanence loss of u is:

$$\frac{|E_o(u, C_i)|}{2E_o^{max}(u)\delta_o(u)} - \frac{|E_o(u, C_i)|}{2\left(E_o^{max}(u)+1\right)(\delta_o(u)+1)}$$
$$= \frac{|E_o(u, C_i)|\left(E_o^{\max}(u)+1\right)\left(\delta_o(u)+1\right) - |E_o(u, C_i)|E_o^{\max}(u)\delta_o(u)}{2E_o^{\max}(u)\left(E_o^{\max}(u)+1\right)\delta_o(u)\left(\delta_o(u)+1\right)}$$

that is always greater than 0 since it holds that $|E_o(u, C_i)|\left(E_o^{\max}(u)+1\right)\left(\delta_o(u)+1\right) > |E_o(u, C_i)|E_o^{\max}(u)\delta_o(u)$.

Moreover, note that the maximum permanence loss is obtained in the second case when the edge is added to the community toward which u already has the maximum number of edges.

4.3.3 Intra-edge deletion

An intra-community edge deletion (u, w) s.t. $u, w \in C_i$ and $\{u, v\} \cap \mathscr{C} \neq \emptyset$ giving an updated network $G' = (V, E \setminus \{(u, w)\})$ always results in a directed permanence loss. Indeed, after the deletion of the edge (u, w), we have that $|E_o(u, C_i)|$, $|E_i(w, C_i)|$, $\delta_o(u)$, $\delta_i(w)$ will be decreased by 1. Consider first the directed permanence loss of node u (a similar reasoning will apply also to node w). We will restrict the analysis to edges whose deletion will decrease $C_{in}(u)$, to the new value $C'_{in}(u) \leq C_{in}(u)$. Then the directed permanence loss of u is:

$$\frac{|E_o(u, C_i)|}{2E_o^{\max}(u)\delta_o(u)} - (1 - C_{in}(u)) - \frac{|E_o(u, C_i)| - \}1}{2E_o^{\max}(u)(\delta_o(u) - 1)} + (1 - C'_{in}(u))$$
$$= \frac{\delta_o(u) - |E_o(u, C_i)|}{2E_o^{\max}(u)\delta_o(u)\left(\delta_o(u) - 1\right)} + (C_{in}(u) - C'_{in}(u))$$

that is always greater or equals to 0 since $\delta_o(u) \geq |E_o(u, C_i)|$ and $C'_{in}(u) \leq C_{in}(u)$.

Note that the deletion of the edge (u, w) will also decrease the directed permanence of all the nodes v that have both u

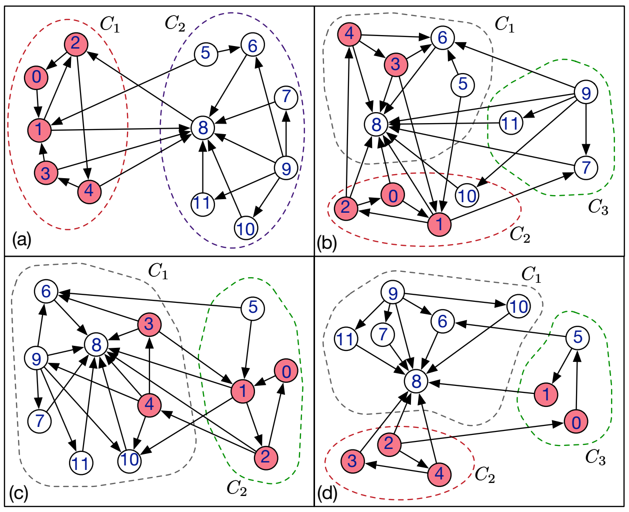


Fig. 3 Deception on directed networks with different deception techniques: (**b**) directed modularity, (**c**) directed safeness, and (**d**) directed permanence

and w in its neighborhood, since their clustering coefficient $C_{in}(v)$ will decrease.

4.3.4 Inter-edge deletion

Any inter-community edge deletion (u, w) s.t. $u \in C_i \cap \mathscr{C}$ and $w \in C_w \neq \mathscr{C}$ giving an updated network $G' = (V, sE \setminus \{(u, w)\})$ never brings a directed permanence loss.

Consider first the case in which $E_o^{\max}(u)$ does not change after the edge deletion, then the directed permanence loss of u is:

$$\frac{|E_o(u, C_i)|}{2E_o^{\max}(u)\delta_o(u)} - \frac{|E_o(u, C_i)|}{2E_o^{\max}(u)(\delta_o(u) - 1)} = \frac{-|E_o(u, C_i)|}{2E_o^{\max}(u)\delta_o(u)\left(\delta_o(u) - 1\right)}$$

that is always lower or equals to 0 since $|E_o(u, C_i)|$ is at least 0.

Consider now the case in which $E_o^{\max}(u)$ changes after the edge deletion, then the new value will be $E_o^{\max}(u) - 1$. In this case the directed permanence loss of u is:

$$\frac{|E_o(u, C_i)|}{2E_o^{\max}(u)\delta_o(u)} - \frac{|E_o(u, C_i)|}{2\left(E_o^{\max}(u) - 1\right)(\delta_o(u) - 1)}$$
$$= \frac{|E_o(u, C_i)|\left(1 - E_o^{\max}(u) - \delta_o(u)\right)}{2E_o^{\max}(u)\left(E_o^{\max}(u)+1\right)\delta_o(u)\left(\delta_o(u)+1\right)}$$

4.4 Directed deception in practice

In this section, we will analyze the behavior of the different deception strategies on the synthetic network reported in

Fig. 3 (a). The two communities shown in the figure have been identified by running the `infomap` detection algorithm (Rosvall and Bergstrom 2008). In the following, we suppose that the target community is $C_1 = \{0, 1, 3, 4, 5\}$ and it has been completely disclosed. Moreover, we will consider a budget of update $\beta = 4$.

Effect of directed modularity deception. The modularity-based deceptor on the directed network in Fig. 3(a) will suggest, in order, the following edge updates:

1. Inter-community edge addition (0, 8);
2. Inter-community edge addition (3, 6);
3. Inter-community edge addition (1, 7);
4. Inter-community edge addition (4, 6);

If we run the same detector on the network obtained after applying the four modifications suggested, we obtain the network reported in Fig 3(b). As it can be noted, the detector identifies three communities, and the target community's members are spread between C_1 and C_2.

Effect of directed safeness deception.

The safeness-based deceptor on the directed network in Fig. 3(a) will suggest, in order, the following edge updates:

1. Inter-community edge addition (4, 10);
2. Inter-community edge addition (3, 6);
3. Inter-community edge addition (1, 10);
4. Inter-community edge addition (4, 9);

If we run the detector on the network obtained after applying the four modifications suggested, we obtain the network reported in Fig 3(c). As it can be noted, the detector identifies two communities with the target community's members spread in both of them.

Effect of directed permanence deception.

The permanence-based deceptor on the directed network in Fig. 3(a) will suggest, in order, the following edge updates:

1. Intra-community edge deletion (1, 2);
2. Intra-community edge deletion (3, 1);
3. Inter-community edge addition (0, 5);
4. Intra-community edge deletion (0, 1);

If we run the detector on the network obtained after applying the four modifications suggested, we obtain the network reported in Fig 3(d). As it can be noted, the detector identifies three communities with the target community's members spread in two of them.

We want to point out that there is a slight difference in the interpretation of intra- and inter-community edge updates among the three different deception strategies described in the previous section. Consider the case in which the target community in the network in Fig. 3 is $\mathcal{C} = \{0, 1, 3, 5, 8\}$. Then, directed modularity and directed permanence categorize intra- and inter-edge updates by considering the community structure identified by the detector, meaning that, for example, the addition of the edge (1,5) would be considered as an inter-edge addition that could decrease both modularity and permanence, while, of course, it is not a good update w.r.t. the target community $\mathcal{C}$. Instead, the directed safeness does not suffer from this problem since it always looks at intra-$\mathcal{C}$ and inter-$\mathcal{C}$ edge updates. Thus, when applying modularity and permanence deception algorithms, one should also check that the intra- and inter-community edge updates suggested meet the requirement related to the identity of the target community $\mathcal{C}$. By considering the example network reported in Fig. 3 such differences cannot be appreciated since the target community is completely revealed.

5 Experimental evaluation

This section reports on an experimental evaluation of the community deception approaches devised for directed networks. We set three main goals. The *first* one is to assess the feasibility of community deception approaches in directed networks. In particular, we want to gain some insight into how our approaches, which rework the state of the art in a directed network context, are effective. *The second* goal is to compare community deception approaches for directed networks with the state of the art that has focused on undirected networks. The comparison will shed further light on our novel techniques' effectiveness in hiding capabilities. *The third* goal is to assess the scalability, in terms of running time, of our novel approaches and make a parallel with deception in undirected networks. We also measure the impact of the deception strategies on the whole community structure by measuring the similarity between communities before and after deception. In what follows, we describe the experimental setting (Sect. 5.1), the datasets (Sect. 5.2), and then report on the experimental results. The algorithms have been implemented in Python. Code and datasets are available online[2]

5.1 Experimental setting

To introduce the experimental setting, we refer to the general framework outlined in Sect. 1 and provide details about the actors involved.

[2] https://communitydeception.wordpress.com/.

Table 5 Datasets and communities found by the Detectors considered

Network	$\|V\|$	$\|E\|$	Number of communities				
			`leiden`	`dm`	`surprise`	`infomap`	`gemsec`
`Freeman`	~50	~500	5	5	7	6	5
`Email`	~1K	~25K	28	32	21	12	16
`AnyBeat`	~12K	~67K	129	81	143	156	112
`WikiVote`	~7K	~103K	30	34	43	51	49
`Facebook-like`	~900	~142K	6	5	6	7	5
`Epinions`	~75K	~508K	795	986	–	–	–
`Slashdot`	~77K	~905K	825	1115	–	–	–
`SocialNet`	~82K	~1M	841	1120	–		–
`Academia`	~200K	~1.4M	89	93	–	–	–
`GooglePlus`	~211K	~1.5M	2105	–	–	–	–

5.1.1 Detectors

We considered a variety of community detection algorithms (detectors) that will act as adversaries to the deception techniques. To make the comparison meaningful for our context, we focus on approaches that work on directed networks. We considered the following algorithms available in the cdlib library:[3]

- Leiden (Traag et al. 2019) (`leiden`): a community detection algorithm that corrects for some issues of the Louvain algorithm (Blondel et al. 2008) and can work on directed networks.
- Directed modularity (Leicht and Newman 2008) (`dm`): this algorithm is an extension of the modularity maximization algorithm devised for undirected networks (Newman 2004).
- Surprise community (Traag et al. 2015) (`surprise`): this algorithm uses the notion of asymptotic surprise, which assesses the quality of the partition of a network into communities.
- InfoMap (Rosvall and Bergstrom 2008) (`infomap`): a detection algorithms that leverages information theory (the shortest description length for a random walk) to return a community structure.
- Gemsec (Rozemberczki et al. 2019) (`gemsec`): an approach that leverages random walks to approximate the point-wise mutual information matrix obtained by pooling normalized adjacency matrix powers. This matrix is decomposed by an approximate factorization technique which is combined with a k-means-like clustering cost.

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

5.1.2 Deceptors

To tackle community deception in directed networks, we considered the following two categories of deceptors:

- Approaches devised for undirected networks: to make the experiments possible for these approaches, we treat the directed network under consideration as undirected. We considered:
  - Delete Internal Connect External (Waniek et al. 2018) (`DICE`): this community deception algorithm is based on the heuristic of deleting intra-community edges and adding inter-community edges. `DICE` is based on the assumption that such kinds of edge updates always minimize modularity.
  - Modularity Minimization (Fionda and Pirrò 2018) (`modMin`): this approach corrects for some issues with `DICE`; the authors of `modMin` showed that in some cases, `DICE` fails to perform edge updates that minimize modularity.
  - Safeness-based deception (Fionda and Pirrò 2018) (`SAF`): this approach introduces safeness maximization for community deception.
  - Permanence-based deception (Mittal et al. 2021) (`NEUR`): this approach is based on permanence minimization.
  - Random edge updates (`RND`): we consider an approach that randomly selects both the type of update and the endpoints of the edge addition/deletion.
- Approaches devised for directed networks: for this category of deceptors we consider edge direction. In this case, we considered all the novel approaches described in the present paper.
  - Directed modularity (`dmod`): this is the approach described in Sect. 4.1

– Directed safeness (`dsaf`): this approach is described in Sect. 4.2
– Directed permanence (`dper`): this approach is described in Sect. 4.3.

5.2 Datasets

As this paper aims to introduce deception for directed networks, we focused on various real directed social networks from a wide range of domains. These networks are available online[4][5][6].

Table 5 gives an overview of the networks considered. The table also reports, for each network, the number of communities found by the Detectors considered. We note that some of the detectors could not complete community detection on the more extensive networks after a timeout of 3h.

5.3 Evaluation methodology

To test deception algorithms, we refer to the methodology introduced in our previous work (Fionda and Pirrò 2018). As an indicator of performance, we measure:

- *Deception Score*: this score, which ranges between 0 and 1, combines a measure of reachability preservation among $\mathscr{C}$ members, community spread (in how many communities are $\mathscr{C}$'s members spread), and community hidings ($\mathscr{C}$'s members should be included in the largest communities) (Fionda and Pirrò 2018).
- *Normalized Mutual Information*(NMI) (Danon et al. 2005): this is a measure that we use to check how deception affects the original community structure. In particular, given the community structure before deception $\overline{C}$ and the community structure after deception $\overline{C}$', we have NMI($\overline{C}, \overline{C}'$) $\in [0, 1]$.
- *Running time*: we also measured deception running time for the various algorithms without considering the time to find communities.

Related pieces of work (Mittal et al. 2021) considered community spread and community hiding separately. However, we believe that also reachability is relevant and that a good deceptor should be evaluate on all the above components simultaneously.

To pick the target community C, we looked at the distribution of the size of the communities. For each detection algorithm, we considered different $\mathscr{C}$ (one for each experiment round) having sizes close to the center of the distribution. Experiments have been conducted on a PC i5 CPU with 3.0 GHz (4 cores) and 16GBs RAM. The results reported are the average (95% confidence interval) of 5 runs.

5.4 Evaluating directed community deception

We start with a discussion about the performance of the novel community deception approaches for directed networks presented in terms of deception score and NMI.

5.4.1 Deception score

Figure 4 shows the results in terms of deception score for medium-size networks. The figure reports, for each column, the network considered, and for each row, the deception score measured as the capability of deceiving a specific detection algorithm. By looking at this figure, we make the following observations.

On small networks (e.g., `Facebook`), obtaining larger values for the deception score seems easier. In general, the deception score always reaches a value greater than 0.5. This can be considered a reasonably good value because the initial deception score was 0 (that is, $\mathscr{C}$ was completely revealed). The deception score increases as the number of edge updates increase; this is consistent for all deception algorithms, detection algorithms, and networks. `dsaf` performs better than the other algorithms in almost all settings. One exception is the `leiden` detection algorithm, where `dmod` seems to perform slightly better than `dsaf`. `dper` seems to be the less performing detection algorithm. We looked into the deception score's community spread and reachability components to shed light on this behavior. In several cases, edge updates suggested by `dper` result in internal edge deletions that result in a disconnection of the community when the number of edge updates increases. `gemsec` seems to be a relatively robust detection algorithm for all three deception approaches. This is especially true in the `Anybeat` network where, when the number of edge updates is below 60% of the number of edges in $\mathscr{C}$, the deception score remains quite low. Figure 5 shows the results for the largest networks considered. We note that only `leiden` and `dm` were able to complete the community detection task within a timeout of 3h. Even in larger networks, it can be observed that larger budget values (x-axis) correspond to larger values of the deception score. In particular, with a budget equal to 60% of the total number of edges in the community in all cases, the deception score is greater than 0.5. We recall that experiments were conducted in the worst-case scenario with a deception score pre-deception equal to 0 ($\mathscr{C}$ completely revealed). However, it is reasonable to assume that the initial deception score is larger; in reality, when deception algorithms are applied, $\mathscr{C}$ is not completely revealed. The larger the network, the more difficult

[4] https://data4goodlab.github.io/dataset.html.
[5] https://snap.stanford.edu.
[6] https://toreopsahl.com/datasets.

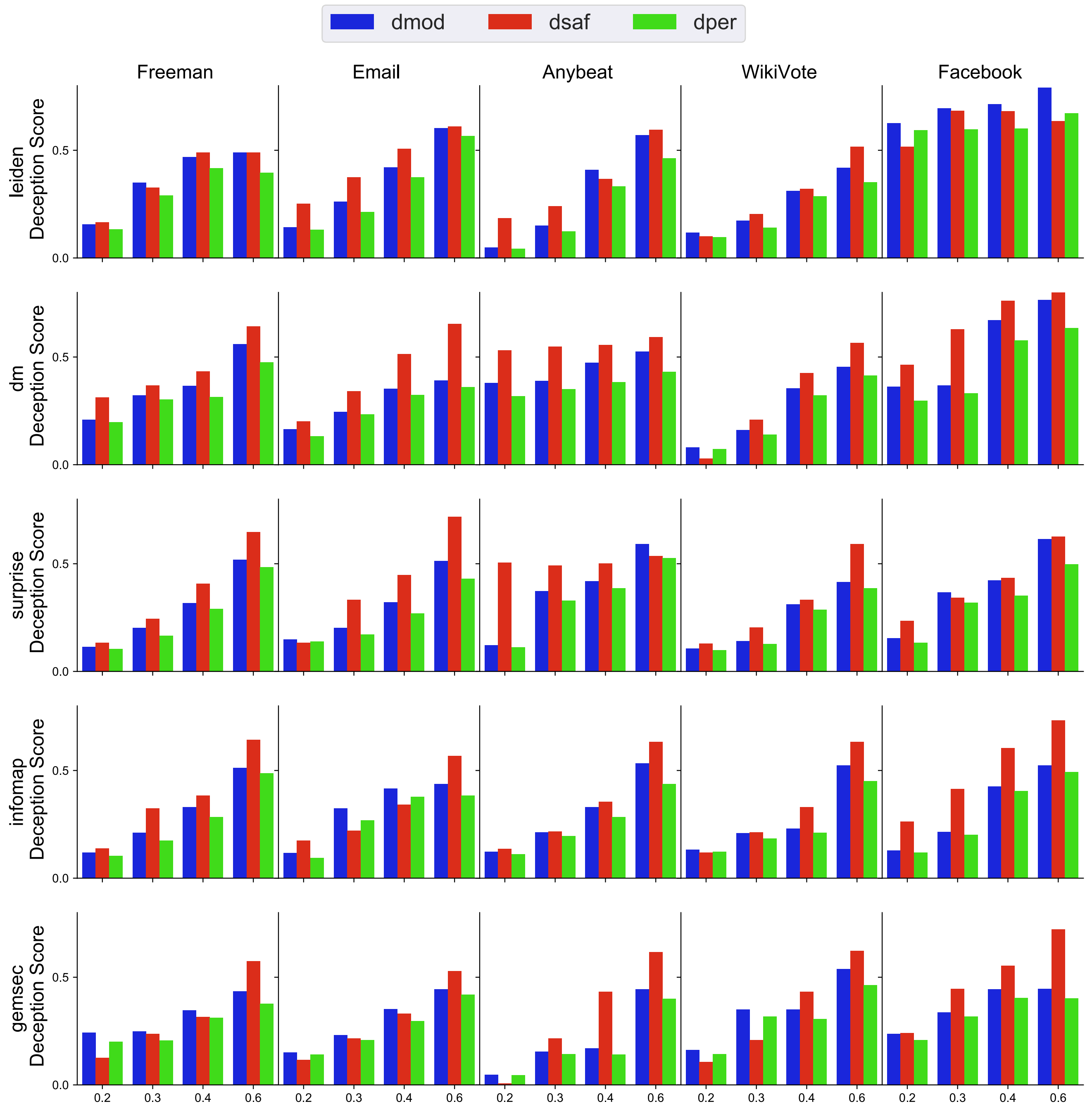


Fig. 4 Directed deception on medium-size networks

it becomes the hide. With the same budget percentage, the results in terms of deception score are lower. By further digging out on the results, we observe that for `Epinions` the number of communities found by `leiden` and `dm` is 795 and 896, respectively. The average size of the $\mathcal{C}$ considered in the experiments is around 400. hence, with 160 updates, `dsaf` can achieve a score greater than 0.5.

On the largest network, google, the number of communities found by `leiden` is 2105, and the average size of the $\mathcal{C}$ considered in the experiments is 500. In this case, with 300 updates, `dsaf` can reach a deception score value greater than 0.5. We again note the `leiden` was the only algorithm able to complete the detection task within a 3h timeout.

Even in this case, we observe that the less effective system is `dper`, which is around 20% and 15% less performing

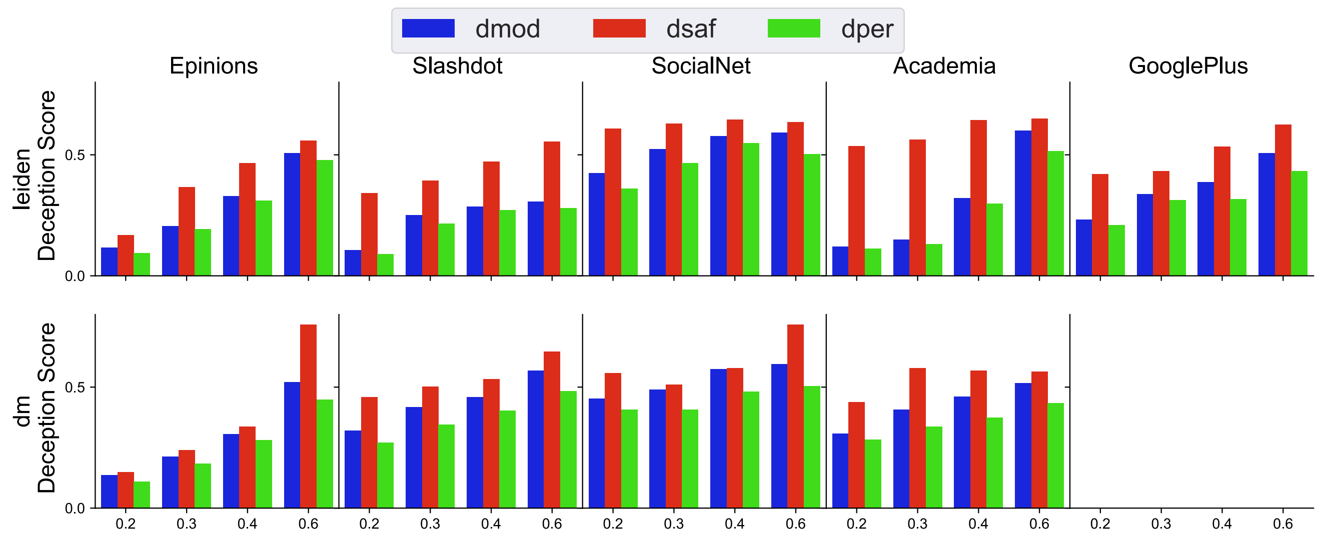


Fig. 5 Directed deception on large-size networks

than dsaf and dmod, respectively. One interesting case is the Academia social network, where nodes represent members that follow other members (hence the network is directed). On the one hand, we observe that dsaf even with lower budget values can achieve a significant result than the other approaches. On the other hand, we observe that dsaf seems to reach a saturation point where further edge updates do not add any benefit. The same is not true for dmod and dper, the performance of which has a significant increase when moving from a 30% to a 60% budget. Here, the average size of $\mathscr{C}$ is 100.

By considering the results from the perspective of detection algorithms, we observe that leiden is more robust to the deception strategies than dm. The reason for this can be found in the fact that leiden finds communities by ensuring that they are well-connected, while dm is an adaptation of modularity optimization to the directed case. Interestingly, the direct competitor of dm would be leiden, also based on directed modularity. However, dsaf consistently performs better both on medium and large networks.

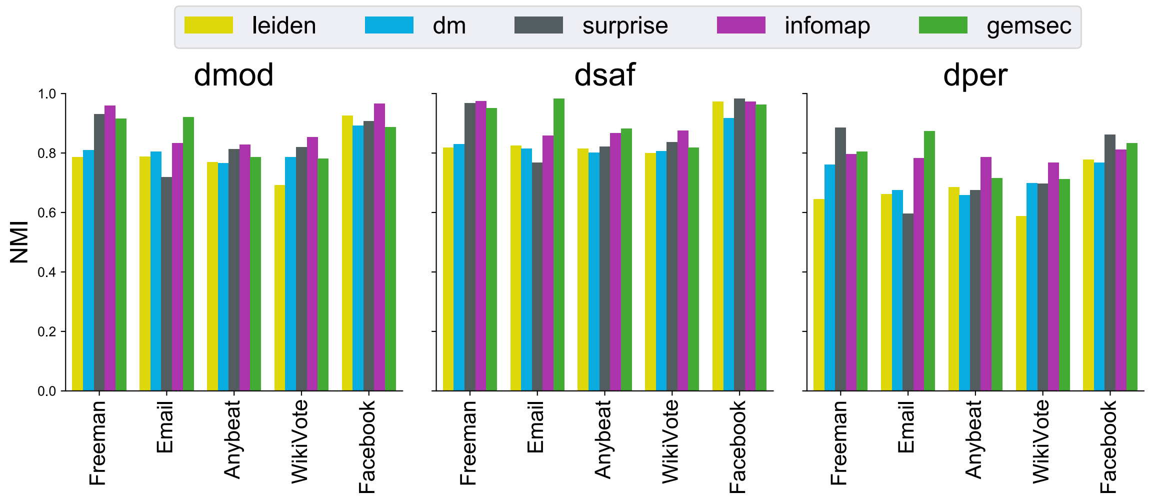
One final observation that we make is related to the characteristics of the community $\mathscr{C}$ considered and the category of edge updates most performed by the deception strategies. We note that the lower the number of intra-$\mathscr{C}$ edges, the easier it becomes to hide it. This sounds natural as more intra-$\mathscr{C}$ edges reinforce the notion of community itself, thus making it challenging to separate nodes within, increasing the deception score.

5.4.2 Normalized mutual information (NMI)

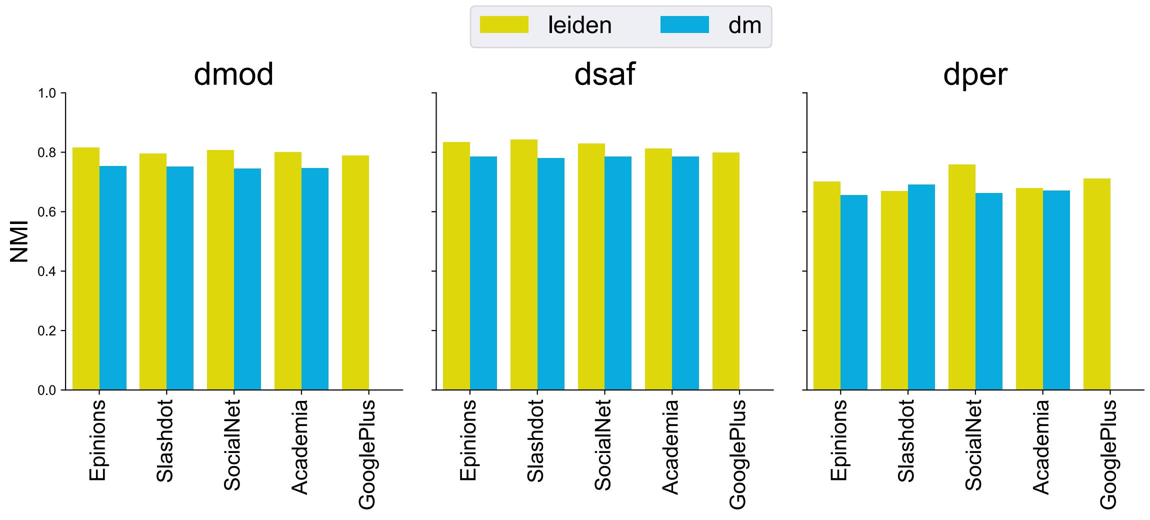
The second dimension of the evaluation considered concerns the impact that community deception has on the original pre-deception communities found. Figure 6 reports the values of the NMI. Each column represents a deception algorithm where the x-axis represents one of the networks and the y-axis the value of NMI.

The values of NMI for medium networks are always around 0.8, meaning that most of the community structure is preserved after applying deception. More specifically, dsaf appears to be the deception algorithm that better preserves the community structure, followed by dmod. The detection algorithms that seem to suffer less from deception in terms of NMI are surprise and infomap. On larger networks (Fig. 6 (b)), the NMI values are a bit higher in general. Still, we have that dper is the deception approach that most changes the community structure. To shed more light on the results, we investigated the relationship between the number of communities before and after deception (referred to as Δ). Figure 7 reports the results for medium size networks when the budget is set to 60% of the number of edges in $\mathscr{C}$. We note that the number of communities after deception decreases; this is always true for the leiden, infomap and gemsec detection algorithms.

By relating the Δ reported in Fig. 7 and the initial number of communities found by each detection algorithm and reported in Table 5 we observe that for the Anybeat network, the number of communities significantly increases (the initial number was 126 for leiden and becomes 167). For the surprise detection algorithm, the larger number of communities is observed in the email network (almost 100 additional communities) for dsaf. In this case, dmod and dper decreased the overall number of communities. By looking at the deception score related to this case (see Fig. 4), we note that dsaf obtained a score higher than dmod and dper. The explanation for this improvement is



(a) NMI on medium networks.



(b) NMI on large networks.

Fig. 6 Directed deception evaluation in terms of NMI

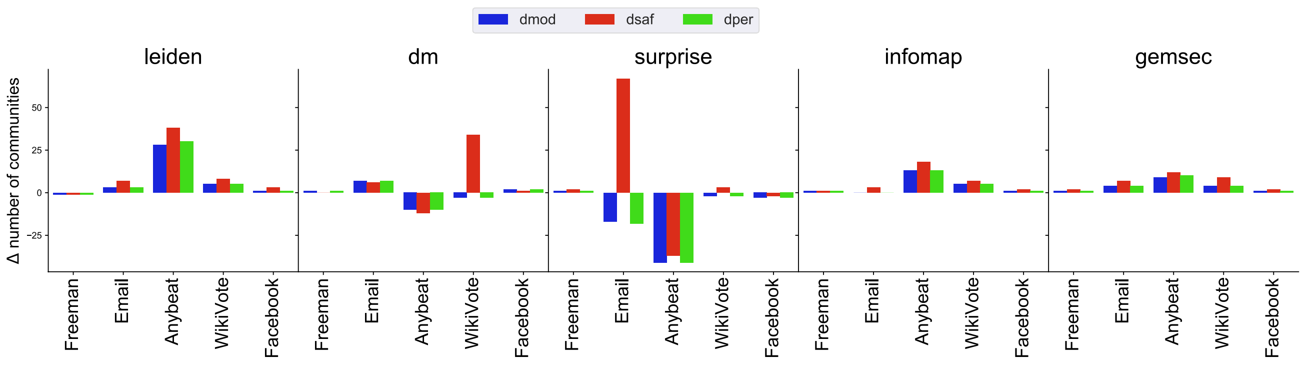


Fig. 7 Variation of the number of communities on medium networks

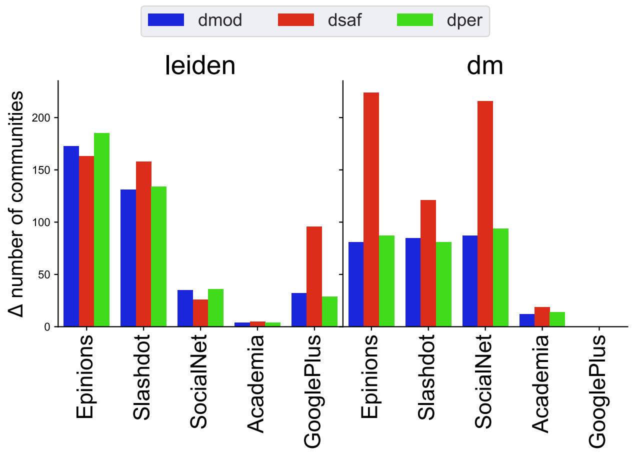


Fig. 8 Variation of the number of communities on large networks

the significant change in the number of communities, which in turn corresponds to an increase in the community spread, that is, the number of communities where $\mathscr{C}$'s members are scattered; indeed, this value went from 1 (the initial setting) to 67. A similar observation can be made for the `WikiVote` network and the `leiden` detection algorithm; here, `dsaf` added a larger number of communities that resulted in a more significant deception score than `dmod` and `dper`.

Figure 8 reports the community variation for larger networks. We observe a similar behavior; the larger the number of new communities after deception, the larger the deception score. This is especially true for `Epinion` and `Social-Net` for the `leiden` detection algorithm and the `dsaf` deception algorithm. We note that `dsaf` reaches a deception score of 0.7 (see Fig. 5).

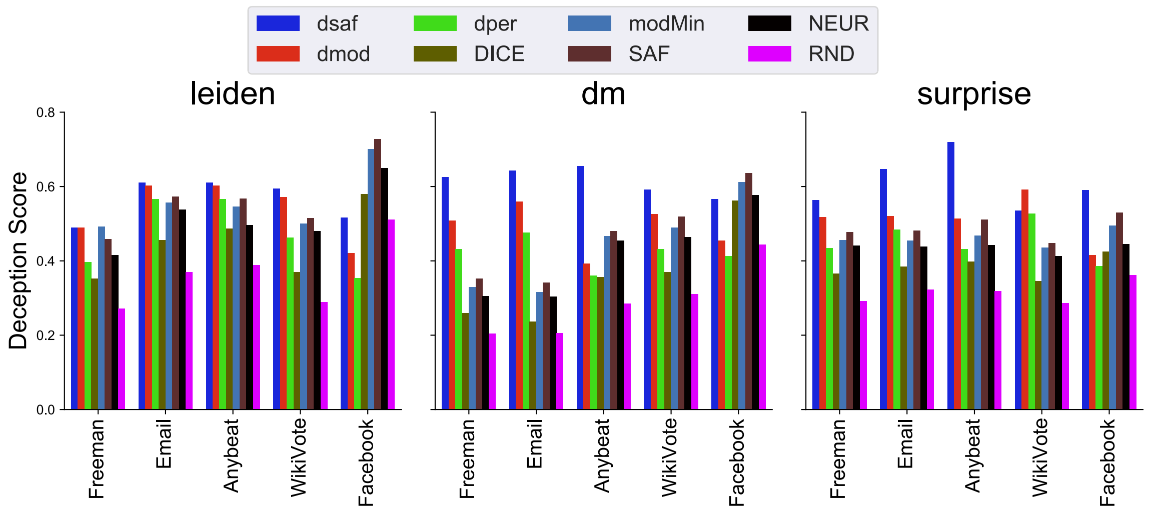


Fig. 9 Comparison between directed and undirected deception approaches on medium networks

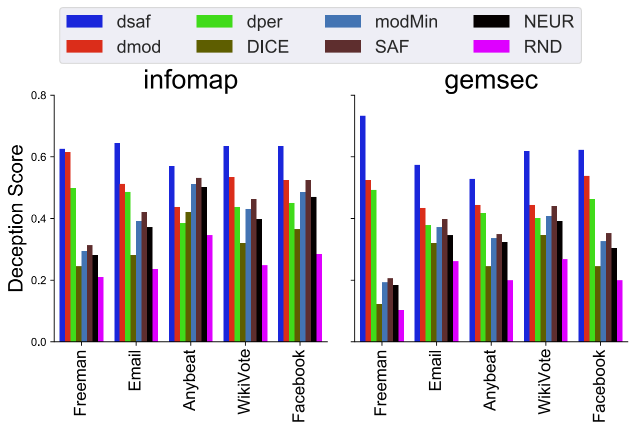


Fig. 10 Comparison between directed and undirected deception approaches on medium networks

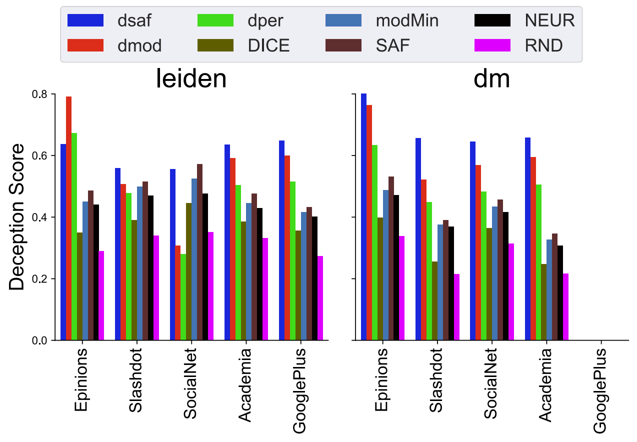


Fig. 11 Comparison between directed and undirected deception approaches on large networks

Table 6 Average deception score with ground-truth communities

	Deception score				
	Leiden	dm	Surprise	Infomap	Gemsec
dsaf	0.623	0.893	0.665	0.635	0.654
dmod	0.598	0.823	0.648	0.642	0.642
dper	0.578	0.789	0.514	0.612	0.591
DICE	0.468	0.658	0.498	0.502	0.471
modMin	0.512	0.662	0.501	0.512	0.484
SAF	0.532	0.698	0.503	0.594	0.493
NEUR	0.516	0.658	0.509	0.449	0.472
RND	0.235	0.123	0.256	0.226	0.194

Table 7 $NMI(\overline{C}_G, \overline{C}_B)$ comparing ground-truth communities and communities returned by a detection algorithm

$NMI(\overline{C}_G, \overline{C}_B)$				
Leiden	dm	Surprise	Infomap	Gemsec
0.891	0.877	0.797	0.862	0.872

Table 8 $NMI(\overline{C}_G, \overline{C}_A)$ comparing ground-truth communities and communities after deception

	$NMI(\overline{C}_G, \overline{C}_A)$				
	Leiden	dm	Surprise	Infomap	Gemsec
dsaf	0.585	0.591	0.563	0.546	0.603
dmod	0.591	0.594	0.574	0.567	0.57
dper	0.581	0.515	0.559	0.536	0.586
DICE	0.593	0.504	0.591	0.504	0.581
modMin	0.533	0.582	0.629	0.597	0.589
SAF	0.514	0.586	0.615	0.512	0.609
NEUR	0.593	0.565	0.625	0.536	0.598
RND	0.589	0.562	0.624	0.547	0.586

5.5 Comparison with undirected deception

We now compare our novel approaches for community deception in directed networks with the state of the art. These approaches were not designed to work on directed networks, which sounds like a limitation. Indeed, several real-world networks, as those considered in our evaluation, are directed, which underlines the importance of adding directions in social network relations. For example, the `Academia` network represents follower-followee relations that naturally carry a direction. To run the experiments with the state-of-the-art deception algorithms, we treated the networks as undirected and considered the same $\mathscr{C}$. In these experiments, we focus on a budget of updates equal to 60% of the edges of $\mathscr{C}$ as this configuration worked best for all approaches. In what follows, we report on comparison in terms of deception score and running time.

5.5.1 Deception score

We compare the deception score of directed and undirected community deception approaches on both medium and large networks. Figures 9 and 10 report results on medium-size networks. The figure considers for each column a detection algorithm. Moreover, the x-axis represents a network in each subfigure while the y-axis is the deception score.

In almost all networks, the directed approaches perform better than the undirected approaches. This is true for all detectors but `leiden`. Here, we observe that for the `Facebook` network, the undirected approaches (excluding the random edge update approach) perform better. To shed more light on this aspect, we looked into the difference between the number of communities after and before deception. In this case, the undirected approaches introduced a larger number of communities than the directed ones. This relation between the number of communities after deception and deception score was also observed when focusing on directed approaches alone (see Sect. 5.4.2).

When moving to the large networks (Fig. 11) we note a clear superiority of the directed approaches. This is especially true for the `Epinions` network. When considering `leiden`, the best performing approach was `dmod` while with `leiden`, `dsaf` obtained slightly better results. One crucial observation is that the undirected algorithms performed significantly worse, reaching in only a few cases a deception score greater than 0.5. As one would expect, the worst-performing deceptor is `RND`, which adds/remove edges randomly starting from $\mathscr{C}$'s members. Also, `NEUR` seems to perform worse than other undirected approaches. To shed more light on this behavior, we looked again at the changes in the community structure and the structure of $\mathscr{C}$; even in this case, we observed that `NEUR` frequently performs internal edge deletions that disconnect $\mathscr{C}$.

5.6 Deception with ground truth communities

In this section, we want to investigate the impact of deception and detection techniques on networks for which the ground truth communities are available. To do so, we do not generate artificial networks and communities but resort to a real-world network of emails for which the communities are available. We are aware that Peel et al. (2017) observed that working with planted communities does not reflect the true data generating process for real networks, which is typically unknown. However, we still believe that the analysis can shed light on how detection algorithms abefore and after applying deception approach these communities. We considered the `email` available from the SNAP repository[7], which represents communication between members of an

[7] https://snap.stanford.edu/data/email-Eu-core.html.

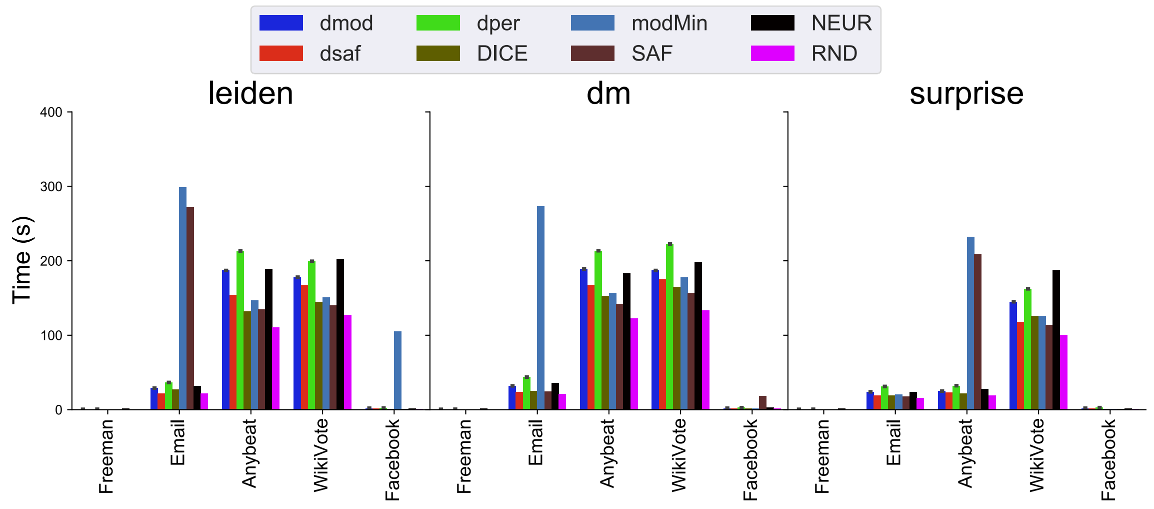


Fig. 12 Running time (s) in medium networks

EU institution with ground-truth communities witnessing the membership of individuals in one of the 42 departments. The network consists of ~1K nodes, 25K edges, and 42 ground-truth communities.

For this experiment, we proceeded as follows. Given a community detection algorithm $\mathcal{D}$, we considered each of the communities returned as the target community $\mathscr{C}$; the number of communities is reported in Table 5. Then, we considered a budget of updates equals to the 50% of the number of nodes in $\mathscr{C}$. In this setting, we have three sets of communities: (i) $\overline{C}_G$: ground-truth communities; (ii) $\overline{C}_B$: communities returned by $\mathcal{D}$ before applying a deception algorithm; (iii) $\overline{C}_A$: communities returned by $\mathcal{D}$ after applying the deception algorithm. We measured the average deception score and NMI values.

Table 6 reports results in terms of deception score. From the table, it emerges that the performance of deception algorithms is consistent with results observed in Fig. 10 where a smaller number of communities (one for each of the 5 experiment rounds) were tested as $\mathscr{C}$. Even in this case, `dsaf` outperforms all the competitors, with approaches devised for undirected networks offering inferior performance. An interesting case is `rnd`, which performs worse than before. This indicates that the deception strategies do not heavily depend on the particular community chosen. However, we noticed that when the size of $\mathscr{C}$ is small, it is, in general, easier to obtain larger values of deception.

We now discuss the different values of NMI score, starting from the analysis of the difference between ground-truth communities and communities returned by a detection algorithm, that is, communities before applying deception algorithms.

Table 7 shows that NMI values are above 0.75, witnessing a quite high level of similarity between the ground-truth communities and the communities found by each detection algorithm. This experiment provides insights into the performance of detection algorithms on this particular network, with `leiden` being the most performing one. We now move to the analysis of the NMI values by comparing ground-truth communities and communities after applying community deception techniques.

We observe from Table 8, that NMI values are much lower than those returned when comparing ground-truth communities with communities returned by a detection algorithm before applying deception techniques. As an example, for the `leiden` detection algorithm and the `dsaf` deception algorithm, which was the best performing detection algorithm, we note that values of NMI drop from 0.891 to 0.585 on average. This means that no matter which of the 42 ground-truth communities we chose as $\mathscr{C}$, there will be a significant difference between the ground truth communities and the communities returned after applying `dsaf`. This same reasoning applies to all other deception techniques. However, we have two observations. First, not always lower values of NMI correspond to higher values of the deception score, which is what ultimately community deception strives to obtain. As an example, although the value of NMI for the `DICE` deception algorithm when considering communities returned by the `dm` detection algorithm is lower than that of `dsaf`, we observe that with the latter, a much larger value of the deception score was obtained (see Table 6). This reasoning is evident when considering the `RND` deception strategy, which adds and removes edges without any clear objective. In fact, while the NMI values are always above 0.5 the corresponding deception score values are very low.

The second observation is that low NMI values after applying deception in a way show that although not specifically designed to hide the whole community structure,

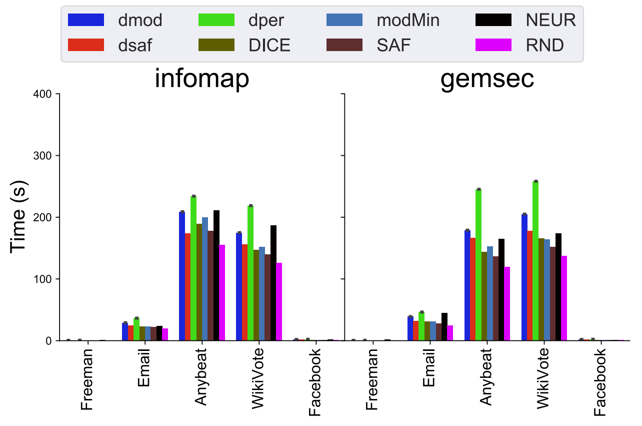


Fig. 13 Running time (s) in medium networks

Table 9 Asymptotic complexity analysis of the deception approaches for undirected and directed networks

Deception measure	Asymptotic complexity
Modularity	$\mathcal{O}(\|E\| + \|V\| + \beta \cdot (k + \|E_{\mathscr{C}}\| + \|V_{\mathscr{C}}\|))$
Safeness	$\mathcal{O}(\beta \cdot (\|E_{\mathscr{C}}\| + \|V_{\mathscr{C}}\|))$
Permanence	$\mathcal{O}(\beta \cdot (\|V_{\mathscr{C}}\| + \|E_{\mathscr{C}}\|^2))$

deception techniques have effects not only on $\mathscr{C}$ but also on the other communities. This comes as no surprise since hiding $\mathscr{C}$, that is, moving its members across communities, changes the structure of each community that releases or receives $\mathscr{C}$'s members.

5.6.1 Deception running time

Our last set of experiments was devoted to investigating the running times of both directed and undirected deception approaches. This will indicate whether considering edge directions brings an additional cost. Moreover, we also insert an asymptotic complexity analysis our novel deception strategies.

In this set of experiments for all approaches, we considered a budget of updates equal to the 60% of the number of edges in $\mathscr{C}$. Figures 12 and 13 show the running time for medium networks. Each column of the figures considers a detection approach. Moreover, in each chart, the x-axis represents a network. We observe that all approaches both directed and undirected run in a few seconds for the smaller networks (e.g., `Freeman`, `Facebook`). An exception is the `modMin` algorithm in the `Facebook` and `Email` networks when considering `leiden` and `dm`, respectively. We further investigated this behavior and hypothesized that it is difficult to exclude bridge edges from possible edge deletions. The same happens in the email network when considering `leiden`. We recall that both `SAF` and `modMin` try to exclude the deletion of internal bridge edges that would disconnect $\mathscr{C}$.

In general, attacks on the output of `leiden` and `dm` appear to be the most costly in terms of running time. We also observe that directed approaches, especially `dper`, require more time than undirected ones in most cases. This comes as no surprise since, from the analysis conducted in this paper (Sect. 4), finding the best edge updates requires a more involved formula where the edge direction and in/out node degrees play a significant role.

Figure 14 reports running time on the largest networks. We observe that here the running time significantly increases; this is especially true for the `GooglePlus` network and the `leiden` algorithm, where the `dper` algorithm required almost 400 seconds to perform the updates. By further analyzing the numbers, we observed that for the largest network, that is, `GooglePlus` the number of updates to be performed was in the order of the hundreds. However, in reality, one can expect that communities that want to implement deception strategies would have a smaller size. This would be reasonable since coordinating among a large group of people can quickly become problematic; in fact, edge updates need to be performed in a real scenario by friending/unfriending or following/unfollowing other nodes.

Asymptotic complexity analysis. For the sake of completeness, Table 9 reports the asymptotic complexity of the deception approaches analyzed in the paper. Note that the asymptotic complexity is the same for the approaches running on directed and undirected networks. The complexity of `DICE` and `RND` is not reported in the table since, at each iteration, the update is selected randomly. Thus, the theoretical complexity only depends on the number of updates (β) that have to be performed. In the case of modularity minimization (row 1 in Table 9) the initialization (corresponding to the computation of δ, η, and the (input/output) total degrees of the communities. Then, the best update to

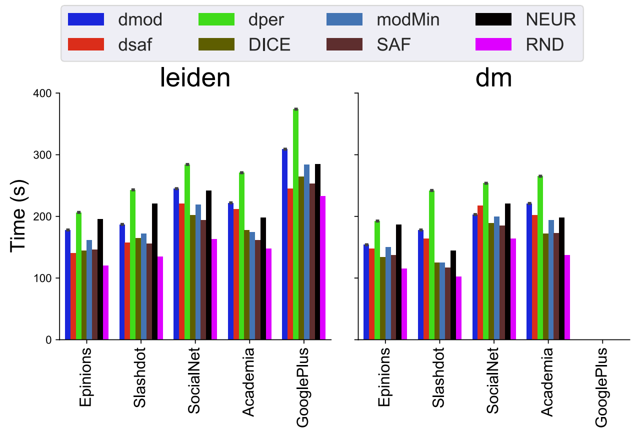


Fig. 14 Running time (s) in large networks

be performed can be computed in $\mathcal{O}(k + |E_{\mathscr{C}}| + |V_{\mathscr{C}}|)$, where k is the number of communities found by the community detection algorithm and derives from the computation of the best inter-edge addition, while the term $|E_{\mathscr{C}}| + |V_{\mathscr{C}}|$ is necessary to compute new bridge edges in $\mathscr{C}$ if the performed update is an intra-edge deletion. The asymptotic complexity of the safeness-based algorithm is dominated by the computation of the best β updates. Indeed, in this case, the initialization has a cost $\mathcal{O}(|E_{\mathscr{C}}| + |V_{\mathscr{C}}|)$ needed for the computation of the connected components of $\mathscr{C}$ and the intra and inter (input/output) degree of the nodes of $\mathscr{C}$. Then, the cost of computing each update is $\mathcal{O}(|E_{\mathscr{C}}| + |V_{\mathscr{C}}|)$, where the cost of computing the best inter-community addition is $\mathcal{O}(|V_{\mathscr{C}}|)$ (to find the node bringing the maximum safeness gain). The cost of computing the best intra-community deletion is $\mathcal{O}(|E_{\mathscr{C}}| + |V_{\mathscr{C}}|)$ since it is dominated by the recomputing of the new bridges after the deletion. Finally, the asymptotic complexity of the permanence-based algorithm is $\mathcal{O}(\beta \cdot (|V_{\mathscr{C}}| + |E_{\mathscr{C}}|^2))$ (as reported in row 3 of Table 9). In this case the cost of the initialization is $\mathcal{O}(|V_{\mathscr{C}}| \cdot |E_{\mathscr{C}}|)$ to compute intra and inter (input/output) degrees and clustering coefficients. Moreover, each update can be computed with a cost of $\mathcal{O}(|V_{\mathscr{C}}| + |E_{\mathscr{C}}|^2)$, where the best intra-community edge addition and intra-community edge deletion can be computed with a cost $\mathcal{O}(|V_{\mathscr{C}}| \cdot |E_{\mathscr{C}}|)$, dominated by the recomputation of the clustering coefficient for each possible modification. The computation of the best inter-community edge addition can be computed in $\mathcal{O}(|V_{\mathscr{C}}|)$.

6 Conclusions

Despite the plethora of approaches to discovering communities, there is not enough awareness that people can act strategically to evade such network analysis tools. This is particularly critical if who wants to *evade* such tools are *malevolent* users and who *run* the tools are *police* enforcement. We introduced the problem of hiding a target community $\mathscr{C}$ from detection algorithms in directed networks. This problem is interesting for two main reasons. First, several real-world networks have edge directions. Therefore, discarding the directions would necessarily result in an information loss. This loss may affect community detection algorithms. This is why specific approaches to finding communities in directed networks have been devised. Second, community deception was only studied in undirected networks. We showed that when throwing out edge direction information, the state of the art fails to reach a reasonable level of hiding of $\mathscr{C}$ inside a community structure. We also showed that it is possible to restore performance similar to that obtained in the undirected scenario when considering direction-aware deception. Specifically, we presented three novel deception strategies. Our theoretical analysis shows that finding the best deception strategy in terms of edge updates is more involved because of the need to distinguish between incoming and outgoing edges for each node. Our extensive experimental evaluation indicates that deception in the directed case is feasible and strictly related to the number of novel communities introduced after applying a deception strategy. Moreover, directed deception is a bit more expensive but still scalable with the size of the network in terms of running time.

There are a number of future research directions. The first is studying deception in the context of network embeddings. Indeed, besides traditional community detection techniques, several approaches perform community discovery via (node and possibly edge) embeddings. Existing deception techniques are not suitable to work in such a setting. The main challenge here consists in the fact that while in a non-embedding setting, one can study the impact of edge updates on some optimization functions (i.e., modularity minimization), understanding how updates reflect into the embedding space is not trivial.

Another exciting line of future research is the investigation of how deception and social bots (Khaund et al. 2022) can benefit from one another. Since social bots mimic the social behaviors of humans, one could think of using social bots to automatize the deception process. The analysis of deception as a cooperative and collective action (Yuce et al. 2014) is also worthy of investigation.

Moreover, we are also interested in applying deception in practice. Indeed, our algorithmic techniques need to be mapped into real-world networks like Facebook or Twitter. The challenge here is how to turn community deception into a collective effort from $\mathscr{C}$'s members that, instructed by deception algorithms, rewire β updates according to a deception function ϕ. Note that while community detection algorithms require complete network knowledge, deception algorithms should ideally only need to know $\mathscr{C}$'s members and their links. In a network like Facebook, intra-$\mathscr{C}$ (resp., inter-$\mathscr{C}$) edge deletions can be simply implemented by "Unfriending" some $\mathscr{C}$'s members (resp., external members). In Twitter, the same behavior can be achieved by "Unfollowing" some $\mathscr{C}$'s members (resp., external members). As for additions, in Facebook, which requires the acceptance of friendship requests, an intra-$\mathscr{C}$ edge addition would not represent a problem. Conversely, an inter-$\mathscr{C}$ edge addition, which requires discovering new network members, can be implemented by picking the target node between colleagues, famous people, classmates, or even random people (by sending several friendship requests). This would reflect in just "Following" some network members on Twitter. Understanding how to implement these policies "silently" is undoubtedly challenging.

Funding Open access funding provided by Università degli Studi di Roma La Sapienza within the CRUI-CARE Agreement.

References

Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast Unfolding of Communities in Large Networks. J Stat Mech 2008(10):P10008

Chakraborty T, Srinivasan S, Ganguly N, Mukherjee A, Bhowmick S (2016) Permanence and community structure in complex networks. ACM TKDD 11(2):1–34

Danon L, Diaz-Guilera A, Duch J, Arenas A (2005) Comparing community structure identification. J Stat Mech Theory Exp 1(9):P09008

Fionda V, Pirrò G (2018) Community deception or: how to stop fearing community detection algorithms. IEEE Trans Knowl Data Eng 30(4):660–673

Fionda V, Pirrò G (2022) Community deception in networks: where we are and where we should go. In: Rosa Maria B, Chantal C, Hocine C, Esteban M, Rocha LM, Sales-Pardo M (eds) Complex networks & their applications X. Springer International Publishing, Cham, pp 144–155

Fortunato S (2010) Community detection in graphs. Phys Rep 486(3):75–174

Fortunato S, Hric D (2016) Community detection in networks: a user guide. Phys Rep 659:1–44

Jia J, Wang B, Cao X, Gong NZ (2020). Certified robustness of community detection against adversarial structural perturbation via randomized smoothing. In Proceedings of The Web Conference 2020, pp 2718–2724

Khaund T, Kirdemir B, Agarwal N, Liu H, Morstatter F (2022) Social bots and their coordination during online campaigns: a survey. IEEE Trans Comput Soc Syst 9(2):530–545

King G, Pan J, Roberts ME (2013) How censorship in china allows government criticism but silences collective expression. Am Polit Sci Rev 107:326–343

Leicht EA, Newman MEJ (2008) Community structure in directed networks. Phys Rev Lett 100(11):118703

Liu Y, Liu J, Zhang Z, Zhu L, Li A (2019) Rem: from structural entropy to community structure deception. Adv Neural Inf Process Syst 32:12938–12948

Magelinski T, Bartulovic M, Carley KM (2021) Measuring node contribution to community structure with modularity vitality. IEEE Trans Netw Sci Eng 8(1):707–723

Malliaros FD, Vazirgiannis M (2013) Clustering and community detection in directed networks: a survey. Phys Rep 533(4):95–142

Mittal S, Sengupta D, Chakraborty T (2021) Hide and seek: outwitting community detection algorithms. IEEE Transactions on Computational Social Systems

Nagaraja S (2010) The impact of unlinkability on adversarial community detection: effects and countermeasures. In PETS, pp 253–272

Newman MEJ (2004) Fast algorithm for detecting community structure in networks. Phys Rev E 69(6):066133

Newman MEJ (2006) Modularity and community structure in networks. PNAS 103(23):8577–8582

Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113

Peel L, Larremore DB, Clauset A (2017) The ground truth about metadata and community detection in networks. Sci Adv 3(5):e1602548

Remy C, Rym B, Matthieu L (2017). Tracking bitcoin users activity using community detection on a network of weak signals. In International conference on complex networks and their applications, pp 166–177. Springer

Rosvall M, Bergstrom CT (2008) Maps of random walks on complex networks reveal community structure. PNAS 105(4):1118–1123

Rozemberczki B, Davies R, Sarkar R, Sutton C (2019). Gemsec: graph embedding with self clustering. In: Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining, pp 65–72

Strogatz SH (2001) Exploring complex networks. Nature 410(6825):268–276

Traag VA, Aldecoa R, Delvenne J-C (2015) Detecting communities using asymptotical surprise. Phys Rev E 92(2):022816

Traag VA, Waltman L, Van Eck NJ (2019) From louvain to leiden: guaranteeing well-connected communities. Sci Rep 9(1):1–12

Waniek M, Michalak TP, Wooldridge MJ, Rahwan T (2018) Hiding individuals and communities in a social network. Nat Human Behav 2(2):139–147

Yuce ST, Agarwal N, Wigand RT, Lim M, Robinson RS (2014) Bridging women rights networks: analyzing interconnected online collective actions. J Glob Inf Manag 22(4):1–20