Community Deception or: How to Stop Fearing Community Detection Algorithms

Valeria Fionda and Giuseppe Pirrò

Abstract—In this paper, we research the community deception problem. Tackling this problem consists in developing techniques to hide a target community ($\mathcal{C}$) from community detection algorithms. This need emerges whenever a group (e.g., activists, police enforcements, or network participants in general) want to observe and cooperate in a social network while avoiding to be detected. We introduce and formalize the community deception problem and devise an efficient algorithm that allows to achieve deception by identifying a certain number (β) of $\mathcal{C}$'s members connections to be rewired. Deception can be practically achieved in social networks like Facebook by friending or unfriending network members as indicated by our algorithm. We compare our approach with another technique based on modularity. By considering a variety of (large) real networks, we provide a systematic evaluation of the robustness of community detection algorithms to deception techniques. Finally, we open some challenging research questions about the design of detection algorithms robust to deception techniques.

Index Terms—Community deception, community detection, community hiding

1 Introduction

Many aspects of everyday life involve networks; social networks, biological networks and the World Wide Web are just a few examples. The study of networks touches many disciplines ranging from physics to computer and social science. One important task in network analysis is the identification of communities, that is, regions (subsets of vertices) of a network that help to gain insights into the structure of the network itself [13]. Detecting communities is useful for several purposes such as: identifying topics in information networks [11], [29], criminal organizations from mobile networks [8], motifs in biological networks [9], [10] or understanding friendship in social networks [37].

While community detection is a well-understood and studied problem, little has been done in terms of techniques that allow to *hide a target community* $\mathcal{C}$ from community detection algorithms [21], [35]. The goal of this paper is to provide an in-depth study of this problem that we name as *community deception*. We believe that studying community deception is intriguing from at least two different perspectives. Deception techniques are useful for users that want to hide (as a group) in social networks like Facebook or Twitter; concrete examples could be activists in despotic regimes or police enforcements. One may argue that a community that wants to hide should not be part of the network. However, we believe that both individuals and groups have the right to observe/cooperate in a network without being tracked by social network analysis (e.g., community detection) tools.

Another perspective that makes deception worth of studying is the fact that deception techniques could be used for malicious purposes (e.g., terrorists that want to communicate while remaining hidden). Therefore, our study raises awareness for the design of novel community detection algorithms robust to deception techniques.

Research Challenges. When embarking on the study of community deception we identified a set of research challenges.

The first challenge is about modeling deception. We formulate deception as an optimization problem by keeping an eye on the realistic amount of network knowledge that $\mathcal{C}$'s members have. Then, we introduce *community safeness*, a targeted-to-deception objective function. We also study another approach where modularity [23] is recast for deception purposes [21], [35]. We show that an optimal modularity-based deception strategy requires knowledge of the communities where to hide $\mathcal{C}$, thus depending from the community detection algorithm that produced these communities. Safeness-based deception does not suffer from this problem.

The second challenge regards how to quantify the level of deception of a detection algorithm for a particular target community $\mathcal{C}$. We give an axiomatic definition of this aspect by introducing the deception score $\mathcal{H}$. The deception score takes into account a wider range of aspects (e.g., reachability, the spread of $\mathcal{C}$'s members) than other measures related to community hiding (e.g., [21], [35]).

The third challenge concerns the design of efficient algorithms. High values of *community safeness* correspond to finding the right set of updates for a given budget of updates β; one way to find such updates could be to search through all possible candidates. This approach does not scale and requires global network knowledge. We will show, on a variety of real networks, how good (deception-wise) solutions can be found via approximate optimization techniques that do not require global network knowledge. Our algorithm

- *V. Fionda is with the Department of Mathematics and Computer Science, University of Calabria, Arcavacata di Rende, CS 87036, Italy. E-mail: fionda@mat.unical.it.*
- *G. Pirrò is with the Institute of High Performance Computing and Networking of the Italian National Research Council (ICAR-CNR), Rome 00185, Italy. E-mail: pirro@icar.cnr.it.*

Manuscript received 24 Feb. 2017; revised 7 Sept. 2017; accepted 6 Nov. 2017. Date of publication 22 Nov. 2017; date of current version 5 Mar. 2018. (Corresponding author: Giuseppe Pirrò.) Recommended for acceptance by H. Xiong. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2017.2776133

TABLE 1
Comparison with Related Research

Approach	Updates Considered	Technique	Network Knowledge Required	Effect of Deception on Detection
Nagaraja [21]	Additions	Modularity	Vertex centrality	No
Waniek et al. [35]	Additions & Deletions	Modularity	$\mathcal{C}$'s members links	No
This paper	Additions & Deletions	Safeness & Modularity	$\mathcal{C}$'s members links	Yes

can scale to networks with millions of nodes and edges and is considerably faster than community detection algorithms.

1.1 Related Work

Community detection algorithms [12] strive to maximize cluster quality measures such as modularity (e.g., [2], [22]), adopt probabilistic approaches based, for instance, on random walks [30] or use network attributes [37]. Yet other approaches study the problem of finding a community given a set of vertices [1]. Fortunato [13] provides a comprehensive study on this topic; other studies focus on the evaluation of community detection algorithms (e.g., [19], [33], [36]).

In this paper, we tackle the problem of designing algorithms to deceive community detection algorithms. *Our goal is to hide a community $\mathcal{C}$ from being discovered by community detection algorithms*. A related problem was first introduced by Nagaraja [21]. The author studied to what extent a community can hide by rewiring some edges; the technique only focused on edge additions toward nodes with high centrality values (e.g., degree centrality). Experiments were conducted on a single small email network and against a single modularity-based detection algorithm. Waniek et al. [35] also touched upon this problem by hinting a strategy inspired by modularity [23]. Their algorithm called DICE works by randomly deleting edges between members of $\mathcal{C}$ (internal edges) and adding edges between members of $\mathcal{C}$ and non-members (external edges). Both approaches also defined a measure of community hiding.

Table 1 compares our approach with related research. Our work differs in several respects. First, previous work focused on modularity; however, there exists a plethora of community detection algorithms that adopt other objective functions (e.g., random walks [30], label propagation [27]). We introduce an approach based on a targeted-to-deception objective function, called community safeness (Section 3).

Second, we observe that neither of the two approaches provides an analysis of the amount of network knowledge an optimal modularity-based deception strategy requires. We fill this gap and show that to pick the best updates (deception-wise) knowledge of the set of communities where $\mathcal{C}$ has to be hidden is needed. Our safeness-based deception algorithm does not suffer from this problem.

Third, our deception score is more comprehensive; it includes reachability, which is at the core of the notion of community itself (if all members of $\mathcal{C}$ were disconnected then there would be no community). Fourth, we conducted an extensive experimental evaluation (Section 5) on a variety of real-world networks up to millions of nodes/edges. Finally, we also measured the overall impact of deception techniques on the output of detection algorithms.

Another vein of research, somewhat further from our approach is community preservation. This latter problem is tackled via techniques such as k-anonymity, k-structural diversity [31], k-degree anonymity [4] or k-isomorphism [5] and is focused on how well the anonymization preserves communities [4]. Differently from these approaches, tackling community deception does not require anonymization. Indeed, as we will show neither our approach assume to have complete network knowledge nor access to the types of edges. Indeed, it is unrealistic to change or add arbitrary edge types in a social network like Facebook.

1.2 Contributions and Outline

In this paper, we formalize and study the community deception problem. From an evaluation point of view, we discuss extensive experiments aimed at assessing the robustness of community detection algorithms to deception techniques. From a practical point of view, we show how applying deception "in practice" amounts at friending/unfriending network members as indicated by deception algorithms. Specifically, we make the following main contributions:

- introduce and formalize the community deception problem;
- introduce a measure to quantify the level of hiding of a target community $\mathcal{C}$ within the output of a detection algorithm;
- present an efficient algorithm for community deception based on safeness and provide a detailed analysis of another algorithm based on modularity;
- compare the deception capabilities of our algorithm with those of modularity-based algorithms against several existing community detection algorithms on real networks drawn from a variety of domains.
- measure the impact of deception techniques on the output of detection algorithms.

The remainder of the paper is organized as follows. Section 2 introduces the community deception problem. Section 3 presents our algorithm for community deception based on safeness. We outline modularity-based deception in Section 4. Section 5 discusses the evaluation on a variety of real-world networks. We conclude in Section 6.

2 Overview of the Approach

We are ready to state the community deception problem and provide an example. We start with some preliminary definitions. A network $G = (V, E)$ is an undirected graph that includes a set of vertices V and a set of edges E.

The set of communities (i.e., a community structure), discovered by some community detection algorithm $\mathcal{A}_D$ is denoted by $\overline{C} = \{C_1, C_2, \ldots C_k\}$, with $C_i \subseteq V$ and $C_i \cap C_i = \emptyset$, for all $i, j \in \{1, \ldots k\}$ and $i \neq j$. Given a community $\bar{V} \subseteq V$, an intra-$\bar{V}$ edge (or intra-community edge) has the form (u, v):$\{u, v\} \subseteq \bar{V}$ while inter-$\bar{V}$ edge (or inter-community edge) has the form (u, v):$u \in \bar{V}$, $v \notin \bar{V}$.

2.1 Measuring Deception

Given a network $G = (V, E)$, we denote by $\mathcal{C} \subseteq V$ the target community, that is, the community that want to escape community detection algorithms. If $\mathcal{C} \in \overline{C} = \{C_1, C_2, \dots C_k\}$, we are in the worst case scenario: the target community has been completely found. On the other hand, if $\mathcal{C} \notin \overline{C}$ then there can be different ways in which members of $\mathcal{C}$ are hidden within $\overline{C}$. Before introducing deception, we shall establish some desiderata for a good hiding of $\mathcal{C}$ in $\overline{C}$.

- D1: *Reachability Preservation*: $\mathcal{C}$'s members should be reachable from one another to preserve the information flow;
- D2: *Community Spread*: $\mathcal{C}$'s members should be spread in as many communities in $\overline{C}$ as possible;
- D3: *Community Hiding*: $\mathcal{C}'s$ members should be distributed in the largest communities of $\overline{C}$.

These desiderata characterize the community *deception score*. Before formally defining the deception score we introduce some notation. Given a community structure $\overline{C} = \{C_1, C_2, \dots C_k\}$, we define the *recall* of a detection algorithm $\mathcal{A}_D$ wrt a target community $\mathcal{C}$ as follows:

$$\mathcal{R}(C_i, \mathcal{C}) = \frac{\#\ \mathcal{C}\text{'s members in } C_i \text{ found by } \mathcal{A}_D}{|\mathcal{C}|}\ \forall C_i \in \overline{C}.$$

Similarly, we define *precision* as follows:

$$\mathcal{P}(C_i, \mathcal{C}) = \frac{\#\ \mathcal{C}\text{'s members in } C_i \text{ found by } \mathcal{A}_D}{|C_i|}\ \forall C_i \cap \mathcal{C} \neq \emptyset.$$

We are now ready to introduce the deception score.

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

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

where $|S(\mathcal{C})|$ is the number of connected components in the subgraph induced by $\mathcal{C}'s$ members.

$\mathcal{H}$ captures *reachability preservation* (desideratum D1) by the first multiplicative factor in Definition 1. The best situation is when all nodes are in a single connected component; conversely, the worst case occurs when they all belong to a different connected component. *Community spread* (desideratum D2) is captured by the first term in the parenthesis, which includes the maximum recall $\mathcal{R}$. The ideal situation occurs when each member of $\mathcal{C}$ is placed (by a detection algorithm) in a different community and the value of the maximum recall is as low as possible (viz. $1/|\mathcal{C}|$). Finally, *community hiding* (desideratum D3), is captured by the second term in the parenthesis, which is based on the average precision $\mathcal{P}$. The ideal situation is when each $C_i \in \overline{C}$ contains a little percentage of $\mathcal{C}'s$ nodes.

Overall, we have that $\mathcal{H} \sim 1$ if: (i) $\mathcal{C}'s$ nodes are in a single connected component (to fulfill D1) and (ii) each node in $\mathcal{C}$ belongs to a different (to fulfill D2) and large (to fulfill D3) community; $\mathcal{H} = 0$ if (i) each member of $\mathcal{C}$ belongs to a different component or (ii) $\mathcal{C} \in \overline{C}$. We shall now characterize community deception from a computational point of view.

2.2 Problem Formulation

The deception score $\mathcal{H}$ assesses the level of hiding of a community $\mathcal{C}$ within a community structure $\overline{C}$ found by some detection algorithm. *The central idea of this paper is to devise algorithms that, by carefully rewiring β edges of $\mathcal{C}$'s members, improve the deception score of $\mathcal{C}$.*

At this point, the question arises about how to tackle community deception. One way to approach the problem would be to work directly with the deception score $\mathcal{H}$ (viz. trying to maximize it). However, $\mathcal{H}$ incorporates knowledge about the community structure $\overline{C}$ and thus would require to have knowledge of the community detection algorithm $\mathcal{A}_D$ that generated $\overline{C}$. In practical terms, one would need to come up with a deception strategy that depends on $\mathcal{A}_D$ (see Fortunato [13] and Leskovec et al. [19] for comprehensive surveys about detection algorithms). Last but not least, in a realistic context $\mathcal{C}$'s members do not have knowledge of the community structure $\overline{C}$. Therefore, we treat detection algorithms as a black-box. Moreover, to make deception applicable by $\mathcal{C}$'s members, we focus on algorithms that do not require global network knowledge. In what follows, given a community $\bar{V} \subseteq V$, we indicate with $E(\bar{V})$ (resp., $\tilde{E}(\bar{V})$) the set of intra-community (resp., inter-community) edges. Furthermore, E^+ (resp., E^-) denotes a set of edge additions (resp., deletions). We shall now give a general formulation of community deception in terms of an optimization problem.

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

$$\arg\max_{\{E'(\mathcal{C}), \tilde{E}'(\mathcal{C})\}} \{\phi(\mathcal{C}, E(\mathcal{C}), \tilde{E}(\mathcal{C}), \beta, E'(\mathcal{C}), \tilde{E}'(\mathcal{C}))\}$$

where $E'(\mathcal{C}) \cup \tilde{E}'(\mathcal{C}) = (E(\mathcal{C}) \cup \tilde{E}(\mathcal{C}) \cup E^+) \setminus E^-$ and:

$$E^+ \subseteq \{(u, v) : u \in \mathcal{C} \vee v \in \mathcal{C}, (u, v) \notin E(\mathcal{C}) \cup \tilde{E}(\mathcal{C})\},$$
$$E^- \subseteq \{(u, v) : u \in \mathcal{C} \vee v \in \mathcal{C}, (u, v) \in E(\mathcal{C}) \cup \tilde{E}(\mathcal{C})\}, \text{and}$$
$$|E^+| + |E^-| \leq \beta.$$

The function ϕ models a community deception algorithm; the budget β limits the number of edge updates. The crucial difference between the deception function ϕ and the deception score $\mathcal{H}$ is that the former picks the β changes that maximize ϕ, while $\mathcal{H}$ quantifies (in an axiomatic way) the desirable property of our target community $\mathcal{C}$, that is, being as much as possible hidden within the output of a detection algorithm. The desideratum is that high values of ϕ bring good (deception-wise) partitions of the network into communities when applying a detection algorithm. One way to maximize ϕ would be to search within the space of candidates updates for a given budget β. However, finding the optimal solution (i.e., the set of β updates that maximizes ϕ) requires to explore an enormous search space, thus making the maximization problem prohibitively difficult to solve for large graphs. Pursuing this way would also require complete network knowledge (e.g., to try all possible set of combinations of additions toward nodes external to $\mathcal{C}$). Therefore, in this paper we focus on approximate (greedy) optimization techniques that are effective (i.e., allow to obtain high values of the deception

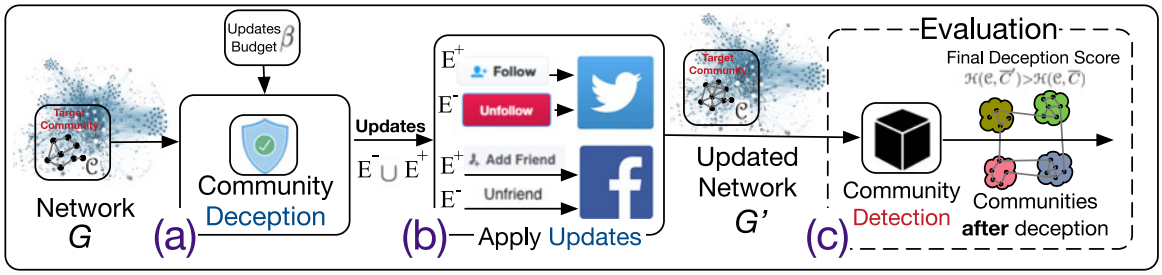

Fig. 1. Given a network G and a target community $\mathscr{C}$, a deception algorithm suggests a set of updates $E^{-} \cup E^{+}$. These updates when applied by $\mathscr{C}$'s members (e.g., by friending/unfriending in Facebook or following/unfollowing in Twitter) give an updated network G'. The deception score of $\mathscr{C}$ in the community structure $\overline{C}'$ (computed from G') is expected to be higher than that in $\overline{C}$ (computed from G).

score $\mathcal{H}$), can scale to networks with millions of nodes/edges and do not require global network knowledge.

2.3 Applying Deception in Practice

Fig. 1 gives an overview of community deception. The input to a community deception algorithm (see Fig. 1a) is a community $\mathcal{C}$ and a budget of updates β. Note that while community detection algorithms require complete network knowledge, deception algorithms only need to know $\mathcal{C}$'s members and their links. We think that deception techniques are useful for users that wish not to be identified as belonging to a certain community or group even if they participate in it. Being part of a social network exposes members of $\mathcal{C}$ to community detection since the network topology inherently contains information about community memberships. We see community deception as a collective effort from $\mathcal{C}$'s member that, instructed by deception algorithms, rewire β updates according to a deception function ϕ (see Problem 2).

Connection rewiring (see Fig. 1b) amounts at intra-$\mathcal{C}$ (resp., inter-$\mathcal{C}$) edge deletion or intra-$\mathcal{C}$ (resp., inter-$\mathcal{C}$) edge additions. To give a concrete example, in a network like Facebook, intra-$\mathcal{C}$ (resp., inter-$\mathcal{C}$) edge deletions can be simply implemented by "Unfriending" some $\mathcal{C}$'s members (resp., external members). In Twitter, the same behavior can be achieved by "Unfollowing" some $\mathcal{C}$'s members (resp., external members). As for additions, in Facebook, which requires the acceptance of a friendship requests, an intra-$\mathcal{C}$ edge addition would not represent a problem. Conversely, an inter-$\mathcal{C}$ edge addition, which requires discovering new network members, can be implemented by picking the target node between colleagues, popular people, classmates, or even random people (by sending several friendship requests). In Twitter, this would reflect in just "Following" some member of the network.

Fig. 1c also shows how to evaluate deception algorithms. The idea is to compare the deception scores of $\mathcal{C}$ in $\overline{C}$, the initial community structure, and $\overline{C}'$, the community structure after applying the β updates.

2.4 Running Example

The community deception algorithm $\mathcal{D}_s$, that we are going to detail in Section 3, is based on the notion of community safeness $\sigma(\mathcal{C})$. $\mathcal{D}_s$ is a greedy algorithm, which looks at reachability between $\mathcal{C}'s$ members (i.e., tries to keep reachability limiting the number of intra-$\mathcal{C}$ edges) and the intra/inter edge ratio (i.e., preferring inter edge additions from nodes with the lowest intra/inter edge ratio). The crucial point is that $\mathcal{D}_s$ does not require knowledge about the community structure $\overline{C}$, thus being independent from detection algorithms and only requiring limited network knowledge. The link between $\mathcal{D}_s$ and the problem formulation given in Problem 2 is that the function ϕ to be maximized is the *safeness gain* $\xi_{\mathcal{C}} = \sigma(\mathcal{C}') - \sigma(\mathcal{C})$.

Consider the Zachary's Karate Club network and the community structure $\overline{C} = \{C_1, C_2, C_3, \mathcal{C}\}$ shown in Fig. 2a obtained by the Louvain community detection algorithm ($\mathcal{A}_D$ =`louv`) [2]. To model the worst-case scenario from the deception point of view, we assume $\mathcal{C} = \{24, 25, 26, 28, 29, 32\}$; this means that $\mathcal{C} \in \overline{C}$ and then $\mathcal{H}(\mathcal{C}, \overline{C}) = 0$ (the community $\mathcal{C}$ is completely revealed). When considering a budget of updates $\beta = 4$, the output of $\mathcal{A}_D$ after applying the updates in the inner-box is shown in Fig. 2b. We note that: (i) $\mathcal{C}'s$ members are now equally spread in two communities; (ii) $\mathcal{C}'s$ members are now better "hidden" with nodes in $\bar{C}_1$ (there are 3 members of $\mathcal{C}$ out of 15) and $\bar{C}_2$ (there are 3 members of $\mathcal{C}$ out of 6); (iii) all nodes of $\mathcal{C}$ are reachable from one another. Therefore, with a few updates the level of hiding of $\mathcal{C}$ increased from 0 to 0.602. The steps of $\mathcal{D}_s$ are discussed in detail in Example 9. We will discuss in Section 5 on a variety of real networks how $\mathcal{D}_s$, with limited network knowledge, is effective in hiding a community $\mathcal{C}$.

3 Safeness-Based Deception

The goal of our safeness-based deception algorithm $\mathcal{D}_s$ is to deceive a detection algorithm by rewiring the connections of $\mathcal{C}$'s members. This way the choice $\mathcal{C} \in \overline{C}$ (i.e., grouping all nodes in $\mathcal{C}$ in a single community) will not be *a good choice* for the detection algorithm. Our rewiring strategy leverages node safeness. In what follows, given a community $\bar{V} \subseteq V$, we indicate with $E(u, \bar{V})$ (resp., $\widetilde{E}(u, \bar{V})$) the set of intra-community (resp., inter-community) edges for a node $u \in \bar{V}$.

Definition 3 (Node Safeness). *Let $G = (V, E)$ be a network, $\mathcal{C} \subseteq V$ a community, and $u \in \mathcal{C}$ a member of $\mathcal{C}$. The safeness $\sigma(u, \mathcal{C})$ of u in G is defined as*

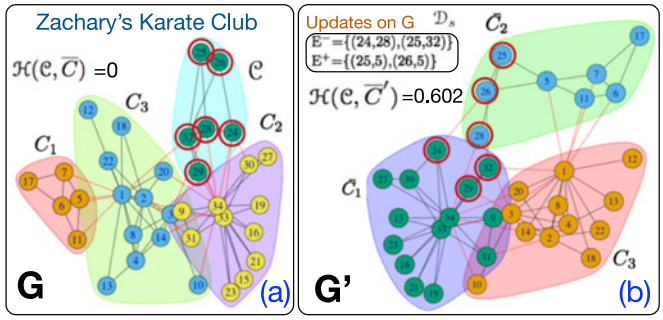

Fig. 2. (a) Communities found by the `louv` community detection algorithm on the Zachary's Karate Club network. (b) Output of `louv` after applying the updates suggested by the deception algorithm $\mathscr{D}_s$.

$$\sigma(u,\mathcal{C}) := \frac{1}{2}\frac{|V_{\mathcal{C}}^u| - |E(u,\mathcal{C})|}{|\mathcal{C}|-1} + \frac{1}{2}\frac{|\widetilde{E}(u,\mathcal{C})|}{deg(u)}, \quad (1)$$

where $V_{\mathcal{C}}^u \subseteq \mathcal{C}$ is the set of nodes reachable from u passing only via nodes in $\mathcal{C}$.

We explain the rationale behind Eq. (1). In the first term ($\frac{|V_{\mathcal{C}}^u|-|E(u,\mathcal{C})|}{|\mathcal{C}|-1}$), the numerator $|V_{\mathcal{C}}^u| - |E(u,\mathcal{C})|$ computes the difference between the number of nodes in $\mathcal{C}$ that are in the same connected component of u and the number of edges that connect u with other $\mathcal{C}$'s members. Values of this term are in the range $[0, |\mathcal{C}|-1]$; the value 0 is obtained when u cannot reach any node in $\mathcal{C}$, i.e., $|V_{\mathcal{C}}^u| = 0$ (and, thus, also $|E(u,\mathcal{C})| = 0$). On the other extreme, the value $|\mathcal{C}|-1$ is obtained if: (i) $\mathcal{C}$ forms a single connected component (i.e., $|V_{\mathcal{C}}^u| = |\mathcal{C}|$) and (ii) there is only one edge connecting u to nodes in $\mathcal{C}$ (i.e., $|E(u,\mathcal{C})| = 1$). As for the denominator, the value $|\mathcal{C}|-1$ is used to normalize the value $|V_{\mathcal{C}}^u| - |E(u,\mathcal{C})|$ in the range [0,1]. Overall, this term takes into account the portion of nodes in $\mathcal{C}$ that can be reached only via other nodes in $\mathcal{C}$ balanced by the number of intra-community edges, and gives an account of how-well u can transmit information in $\mathcal{C}$. In the ideal situation a member of $\mathcal{C}$ will be able to reach all the other members of $\mathcal{C}$ with the minimum number of intra-community edges.

The second term in Eq. (1) gives the portion of u's edges (wrt all of its edges) that point outside $\mathcal{C}$ and gives an account on how u is "hidden" inside the network with respect to its degree. To increase its safeness u should diversify its connections, that is, have a high proportion of links with nodes not in $\mathcal{C}$. We equally weights the two components of Eq. (1) and return node safeness values in the interval [0,1].

Definition 4 (Community Safeness). *Let $G = (V, E)$ be a network and $\mathcal{C} \subseteq V$ a community, the safeness of $\mathcal{C}$ is defined as: $\sigma(\mathcal{C}) = \sum_{u\in\mathcal{C}} \sigma(u,\mathcal{C})/|\mathcal{C}|$*

Defining the safeness of $\mathcal{C}$ starting from the safeness of its members allows to identify the least safe members and rewire their links to increase the safeness score of the whole $\mathcal{C}$. Safeness controls different aspects of a community such as reachability and internal/external edge balance.

In terms of the general deception formulation (see Definition 2) $\mathcal{D}_s$ instantiates the function ϕ to be the safeness gain $\xi_{\mathcal{C}} = \sigma(\mathcal{C}') - \sigma(\mathcal{C})$ where $\mathcal{C}'$ is the community after applying one or more edge updates. $\mathcal{D}_s$ adopts a greedy strategy that at each step chooses the edge update that gives the highest $\xi_{\mathcal{C}}$.

3.1 Impact of Edge Updates on Safeness

We treat edge additions and deletions separately.

3.1.1 Edge Additions

We start by analyzing inter-$\mathcal{C}$ edge additions. Let $G = (V, E)$ be a network and $\mathcal{C} \subseteq V$ a community having safeness $\sigma(\mathcal{C})$. Let $\xi_{\mathcal{C}} = \sigma(\mathcal{C}') - \sigma(\mathcal{C})$ be the safeness gain.

Theorem 5. *For any inter-$\mathcal{C}$ edge addition (u, w) s.t. $u \in \mathcal{C}$ and $w \notin \mathcal{C}$ giving $G' = (V, E \cup \{(u, w)\})$ we have that: (i) $\xi_{\mathcal{C}} \geq 0$ always holds; (ii) the maximum increase in safeness happens for the nodes $u \in \arg\min\{\frac{|\widetilde{E}(u,\mathcal{C})|}{deg(u)}\}$.*

Proof. Checking if $\xi_{\mathcal{C}} = \sigma(\mathcal{C}') - \sigma(\mathcal{C}) \geq 0$ amounts at checking whether

$$\frac{1}{2}\frac{|\widetilde{E}(u,\mathcal{C})|+1}{deg(u)+1} - \frac{1}{2}\frac{\widetilde{E}(u,\mathcal{C})}{deg(u)} \geq 0$$

Condition (i) holds since $deg(u) \geq \widetilde{E}(u,\mathcal{C})$. Moreover, the maximum increase happens when condition (ii) holds. □

Theorem 6. *An intra-$\mathcal{C}$ edge addition (u, w) s.t. $\{u, w\} \subset \mathcal{C}$ giving an updated network $G' = (V, E \cup \{(u, w)\})$ does not always introduce a safeness gain.*

Proof. We show that an intra-$\mathcal{C}$ edge addition does not always correspond to an increase of safeness. First of all let us observe that if u and w belong to the same component of $\mathcal{C}$ in G, then by adding the edge (u, w) we do not have any benefit; it is immediate to check that this always brings a safeness decrease. Let us now consider the case in which u and w belongs to two disconnected components of $\mathcal{C}$, say $\mathcal{C}_u$ and $\mathcal{C}_w$, respectively. After the addition of the edge (u, w) all $\mathcal{C}_u$'s members (resp., $\mathcal{C}_w$'s members) will be able to reach all $\mathcal{C}_w$'s members (resp., $\mathcal{C}_u$'s members). The insertion of (u, w) brings a safeness increase if, and only if

$$\begin{aligned}
&\sum_{v\in\mathcal{C}_u\setminus\{u\}}\left(\frac{|V_{\mathcal{C}}^u| + |\mathcal{C}_w| - |E(v,\mathcal{C})|}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(v,\mathcal{C})|}{2deg(v)}\right) \\
&+ \frac{|V_{\mathcal{C}}^u| + |\mathcal{C}_w| - |E(u,\mathcal{C})| - 1}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(u,\mathcal{C})|}{2(deg(u)+1)} \\
&+ \frac{|V_{\mathcal{C}}^w| + |\mathcal{C}_u| - |E(w,\mathcal{C})| - 1}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(w,\mathcal{C})|}{2(deg(w)+1)} \\
&> \sum_{v\in\mathcal{C}}\left(\frac{|V_{\mathcal{C}}^v| - |E(v,\mathcal{C})|}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(v,\mathcal{C})|}{2deg(c)}\right)
\end{aligned}$$

that corresponds to check whether

$$\begin{aligned}
&\sum_{v\in\mathcal{C}_u\setminus\{u\}}\frac{|\mathcal{C}_w|}{2(|\mathcal{C}|-1)} + \sum_{v\in\mathcal{C}_w\setminus\{w\}}\frac{|\mathcal{C}_u|}{2(|\mathcal{C}|-1)} + \frac{|\mathcal{C}_w|-1}{2(|\mathcal{C}|-1)} \\
&+ \frac{|\mathcal{C}_u|-1}{2(|\mathcal{C}|-1)} - \frac{|\widetilde{E}(u,\mathcal{C})|}{2deg(u)(deg(u)+1)} - \frac{|\widetilde{E}(w,\mathcal{C})|}{2deg(w)(deg(w)+1)} \\
&> 0
\end{aligned}$$

If the condition is satisfied the safeness gain is

$$\begin{aligned}
\xi_{\mathcal{C}} = &\sum_{v\in\mathcal{C}_u\setminus\{u\}}\frac{|\mathcal{C}_w|}{2(|\mathcal{C}|-1)} + \sum_{v\in\mathcal{C}_w\setminus\{w\}}\frac{|\mathcal{C}_u|}{2(|\mathcal{C}|-1)} + \frac{|\mathcal{C}_w|-1}{2(|\mathcal{C}|-1)} \\
&+ \frac{|\mathcal{C}_u|-1}{2(|\mathcal{C}|-1)} - \frac{|\widetilde{E}(u,\mathcal{C})|}{2deg(u)(deg(u)+1)} - \frac{|\widetilde{E}(w,\mathcal{C})|}{2deg(w)(deg(w)+1)}.
\end{aligned}$$

□

The above theorem deals with the addition of an intra-$\mathcal{C}$ edge (viz., both members belong to $\mathcal{C}$). The possibility for such an edge to increase the safeness of the community occurs when the edge connects previously disconnected portions of $\mathcal{C}$. If no new communication path among $\mathcal{C}$'s members is made available, the new edge will certainly decrease the safeness score.

Intuitively, this is justified by the fact that if the edge does not bring any advantage in terms of connectivity among $\mathcal{C}$'s nodes it will get u and w more connected to members of $\mathcal{C}$ thus strengthening the community. In some cases $\mathcal{C}$'s members could strive to increase reachability between $\mathcal{C}$'s members via paths involving other $\mathcal{C}$'s members so that the induced subgraph of G on $\mathcal{C}$'s nodes has a single connected component.

3.1.2 Edge Deletions

We now move to edge deletions starting from inter-$\mathcal{C}$ edges.

Theorem 7. *An inter-$\mathcal{C}$ edge deletion (u,w): $u \in \mathcal{C}, w \notin \mathcal{C}$ giving $G' = (V, E \setminus \{(u,w)\})$ always gives $\xi_{\mathcal{C}} \leq 0$.*

Proof. Checking if $\xi_{\mathcal{C}} = \sigma(\mathcal{C}') - \sigma(\mathcal{C}) \geq 0$ amounts at checking whether

$$\frac{1}{2}\frac{|\widetilde{E}(u,\mathcal{C})|-1}{deg(u)-1} - \frac{1}{2}\frac{\widetilde{E}(u,\mathcal{C})}{deg(u)} \geq 0,$$

that clearly never holds since $deg(u) \geq \widetilde{E}(u,\mathcal{C})$. □

We now analyze the case of an intra-$\mathcal{C}$ edge deletion and show that it does not always bring a safeness increase.

Theorem 8. *An intra-$\mathcal{C}$ edge deletion (u,w): $\{u,w\} \subseteq \mathcal{C}$ does not always bring a safeness gain.*

Proof. Let us assume that after the deletion of the edge (u,w), the nodes u and w still remain in the same connected component of $\mathcal{C}$. It is enough to note that the following condition (corresponding to $\xi_{\mathcal{C}} > 0$) always holds

$$\begin{aligned}
&\sum_{v\in\mathcal{C}\setminus\{u,w\}} \sigma(v,\mathcal{C}) + \frac{|V^u_{\mathcal{C}}| - |E(u,\mathcal{C})| + 1}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(u,\mathcal{C})|}{2(deg(u)-1)} \\
&+ \frac{|V^w_{\mathcal{C}}| - |E(w,\mathcal{C})| + 1}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(w,\mathcal{C})|}{2(deg(w)-1)} \\
&> \sum_{v\in\mathcal{C}\setminus\{u,w\}} \sigma(v,\mathcal{C}) + \frac{|V^u_{\mathcal{C}}| - |E(u,\mathcal{C})|}{2(\mathcal{C}|-1)} + \frac{|\widetilde{E}(u,\mathcal{C})|}{2deg(u)} \\
&+ \frac{|V^w_{\mathcal{C}}| - |E(w,\mathcal{C})|}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(w,\mathcal{C})|}{2deg(w)}.
\end{aligned}$$

From the above inequality it can be easily checked that the edge deletion bringing the highest increase in the safeness score is the one for which $\frac{|\widetilde{E}(u,\mathcal{C})|}{2deg(u)(deg(u)-1)} + \frac{|\widetilde{E}(w,\mathcal{C})|}{2deg(w)(deg(w)-1)}$ has the maximum value.

Let us now consider the case in which, after the deletion of (u,w), u and w belong to two disconnected components of $\mathcal{C}$, say $\mathcal{C}_u$ and $\mathcal{C}_w$, respectively. Then, after the deletion of (u,w) all $\mathcal{C}_u$'s members (resp., $\mathcal{C}_w$) will not be able to reach all $\mathcal{C}_w$'s members (resp., $\mathcal{C}_u$). Then, $\xi_{\mathcal{C}} > 0$ if, and only if

$$\begin{aligned}
&\sum_{v\in\mathcal{C}\setminus(\mathcal{C}_u\cup\mathcal{C}_w)} \left(\frac{|V^v_{\mathcal{C}}| - |E(v,\mathcal{C})|}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(v,\mathcal{C})|}{2deg(v)}\right) \\
&+ \sum_{v\in\mathcal{C}_u\setminus\{u\}} \left(\frac{|\mathcal{C}_u| - |E(v,\mathcal{C})|}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(v,\mathcal{C})|}{2deg(v)}\right) \\
&+ \sum_{v\in\mathcal{C}_w\setminus\{w\}} \left(\frac{|\mathcal{C}_w| - |E(v,\mathcal{C})|}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(v,\mathcal{C})|}{2deg(v)}\right) \\
&+ \frac{|\mathcal{C}_u| - |E(u,\mathcal{C})| - 1}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(u,\mathcal{C})|}{2(deg(u)-1)} \\
&+ \frac{|\mathcal{C}_w| - |E(w,\mathcal{C})| - 1}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(w,\mathcal{C})|}{2(deg(w)-1)} \\
&> \sum_{v\in\mathcal{C}} \left(\frac{|V^v_{\mathcal{C}}| - |E(v,\mathcal{C})|}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(v,\mathcal{C})|}{2deg(v)}\right)
\end{aligned}$$

that corresponds to check whether

$$\begin{aligned}
&\sum_{v\in\mathcal{C}_u\setminus\{u\}} \frac{-|\mathcal{C}_w|}{2(|\mathcal{C}|-1)} + \sum_{v\in\mathcal{C}_w\setminus\{w\}} \frac{-|\mathcal{C}_u|}{2(|\mathcal{C}|-1)} - \frac{|\mathcal{C}_w|+1}{2(|\mathcal{C}|-1)} \\
&- \frac{|\mathcal{C}_u|+1}{2(|\mathcal{C}|-1)} + \frac{|\widetilde{E}(u,\mathcal{C})|}{2deg(u)(deg(u)-1)} + \frac{|\widetilde{E}(w,\mathcal{C})|}{2deg(w)(deg(w)-1)} > 0.
\end{aligned}$$

□

3.2 The $\mathcal{D}_s$ Algorithm

The community deception algorithm $\mathcal{D}_s$ outlined in Algorithm 1 builds upon the analysis performed in Section 3.1. In particular, in terms of additions the algorithm only considers inter-$\mathcal{C}$ edge additions since according to Theorem 5 this type of additions always brings a safeness gain. The algorithm does not consider intra-$\mathcal{C}$ edge additions since according to Theorem 5 there can be a safeness gain only if the intra-$\mathcal{C}$ edge allows to connect two components of $\mathcal{C}$ that were previously disconnected. It is realistic to assume that $\mathcal{C}$'s members cooperate to keep the community connected. Hence, if the community $\mathcal{C}$ is disconnected from the beginning then we assume that it is a decision of $\mathcal{C}$'s members to keep the community disconnected. In terms of edge deletions, Algorithm 1 only considers intra-$\mathcal{C}$ edge deletion that do not disconnect a previously connected component of $\mathcal{C}$ (see Theorem 8). The algorithm does not consider inter-$\mathcal{C}$ edge deletions since according to Theorem 7 they always bring a safeness loss. We also note that to pick the best updates the only piece of knowledge required is the ratio inter-$\mathcal{C}$ edges/node degree. Therefore, the amount of network knowledge required by $\mathcal{D}_s$ is limited.

Algorithm 1. $\mathcal{D}_s$ - Community Deception via Safeness

```
1:  procedure comDeceptSafeness(G = (V, E), C, β)
2:    do
3:      n_p = getNodeMinimumAddRatio(C) /* Theorem 5 */
4:      n_t = findExternalNode(n_p, C, G)
5:      ξ^add_C ← getAdditionGain(n_p, n_t, C)
6:      (n_k, n_l) = getBestDelExclBridges(C) /* Theorem 8 */
7:      ξ^del_C ← getDeletionGain(n_k, n_l, C)
8:      if ξ^add_C ≥ ξ^del_C and ξ^add_C > 0 then
9:        G ← (V, E ∪ {(n_p, n_t)})
10:     else if ξ^del_C > 0 then
11:       G ← (V, E \ {(n_k, n_l)})
12:     β = β − 1
13:   while β > 0 and (ξ^add_C > 0 or ξ^del_C > 0)
```

The procedure `getNodeMinimumAddRatio` (line 3) computes, for each $n \in \mathcal{C}$, the fraction of n's edges that point outside $\mathcal{C}$. The procedure `findExternalNode` (line 4) find a node (not in $\mathcal{C}$) such that the edge (n_p, n_t) does not exist (see Section 2.3). The procedure `getBestDelExclBridges` (line 6) works in two phases; it excludes bridge edges that, if deleted, could disconnect $\mathcal{C}$; then, for each remaining edge, it computes the value specified in Theorem 8. Finally, $\mathcal{D}_s$ selects the most convenient (safeness-wise) edge update (line 8) and applies it to the network (lines 9-11).

Example 9. Fig. 3 shows both the candidate updates and the actual update choses (in bold) made by $\mathcal{D}_s$ when considering a budget $\beta = 4$, the target community $\mathcal{C}$ and the network in Fig. 2a. For the first update, the algorithm picks the nodes 25 and 26 *as candidates for an inter-community edge addition*. This is because these are the nodes having the minimum

Safeness ($\mathcal{D}_s$)				
#Update	Addition	$\xi_{\mathcal{C}}^{add}$	Deletion	$\xi_{\mathcal{C}}^{del}$
1	(25,_)	.125	**(24,28)**	.358
2	(25,_)	.125	**(25,32)**	.25
3	**(25,_)**	.125	-	0
4	**(26,_)**	.125	-	0

Fig. 3. Updates made by $\mathscr{D}_s$ for the network and $\mathscr{C}$ in Fig. 2a.

ratio according to Theorem 5 (line 3). Indeed, both node 25 and 26 have 0 inter-community edges. The safeness increase is $\xi_{\mathcal{C}}^{add} = 0.125$ in this case (line 5). The algorithm also picks the edge $(24, 28)$ as an *intra-community deletion candidate*; this is because this edge maximizes the factor $\frac{|E(u,\mathcal{C})|}{2deg(u)(deg(u)-1)} + \frac{|E(w,\mathcal{C})|}{2deg(w)(deg(w)-1)}$ as per Theorem 8. It is interesting to note that the edge $(29, 32)$ is not selected as a candidate because it is a bridge and its deletion will disconnect $\mathcal{C}$. The safeness increase brought by the deletion of $(24, 28)$ is $\xi_{\mathcal{C}}^{del} = 0.358$ (line 7). Finally, since $\xi_{\mathcal{C}}^{del} > \xi_{\mathcal{C}}^{add}$ (line 8) the first update applied (and reported in bold in Fig. 3) is the deletion of the intra-community edge $(24, 28)$ (line 11). The algorithm proceeds in a similar way for the other updates in Fig. 3. The updated network is shown in Fig. 2b.

Theorem 10. *Algorithm 1 runs in time $\mathcal{O}(|\mathcal{C}| + |E(\mathcal{C})|)$.*

Proof. The node for the best addition and the edge for the best deletion can be computed by checking all the nodes and edges in $\mathcal{C}$. In fact, the identification of bridges can be done in $\mathcal{O}(|\mathcal{C}| + |E(\mathcal{C})|)$. Addition and deletion gains can be computed in $\mathcal{O}(|E(\mathcal{C})|)$. □

Correctness of the $\mathcal{D}_s$ algorithm. Given a community $\mathcal{C}$ and a budget β, the $\mathcal{D}_s$ Algorithm updates the initial network G by obtaining a network G' such that $\sigma(\mathcal{C}') \geq \sigma(\mathcal{C})$. In fact, it is enough to note that each edge update is selected, as per Theorem 5 and Theorem 8. According to Theorem 5, inter-$\mathcal{C}$ edge additions (lines 3-5) always bring an increase in safeness. As for intra-$\mathcal{C}$ edge deletions, since bridge edges are never deleted (lines 6-7) the connectivity of nodes in $\mathcal{C}$ is preserved (see the proof of Theorem 8).

4 Modularity-Based Deception

We now outline a second community deception algorithm $\mathcal{D}_m$, which shows how modularity [23] can be recast for deception purposes. We denote by $deg(v)$ the degree of $v \in V$, that is, the number of neighbors of v. The degree of a community C_i is denoted by: $deg(C_i) = \sum_{v \in C_i} deg(v)$.

Definition 11 (Modularity). *Given a network G having m edges, the modularity of the partition of this network into communities $\overline{C} = \{C_1, C_2, \ldots C_k\}$ is given by*

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

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

Modularity measures the number of edges falling within groups minus their expected number in an equivalent network with edges placed at random. The objective of many detection algorithms is to maximize modularity [3].

To achieve deception, an intuitive strategy would be to leverage the modularity loss $\mathcal{ML} = \mathcal{M}_G(\overline{C}) - \mathcal{M}_{G'}(\overline{C})$; here, the goal is to find the set of β edge updates where $\mathcal{ML}$ is maximized. Maximizing the loss will make the partitioning $\overline{C}$ given by $\mathcal{A}_D$ no more optimal and thus (hopefully) will increase the level of hiding of $\mathcal{C}$. However, this approach would not conform to the general function ϕ in Problem 2. This is because, we are going to show in Section 4.1 that a deception strategy based on the modularity loss requires knowledge of the community structure $\overline{C}$ to pick the best updates and thus depends on the community detection algorithm that generated the structure. We showed in Section 3 that safeness-based deception does not suffer from this problem.

Nevertheless, we study modularity-based deception to compare our approach with that of Waniek et al. [35] that hinted a somehow related strategy. Their study suggests a heuristic (called DICE) based on intra-community edge deletions and inter-community edge additions for some communities in $\overline{C}$. However, as we are going to show Waniek et al.'s strategy does not always bring a modularity loss. Indeed, in some cases the edge update suggested by their strategy may actually increase modularity.

4.1 Impact of Edge Updates on Modularity

Let $G = (V, E)$ be a network, $\overline{C} = \{C_1, C_2, \ldots C_k\}$ be a partitioning having modularity $\mathcal{M}_G(\overline{C})$, $\mathcal{C} \subseteq V$ be a community and $\mathcal{ML} = \mathcal{M}_G(\overline{C}) - \mathcal{M}_{G'}(\overline{C})$ be the modularity loss.

4.1.1 Edge Additions

We first consider inter-community edge additions.

Lemma 12. *An inter-community edge addition (u, w): $u \in C_i \cap \mathcal{C}, w \in C_j$, with $i \neq j$ giving $G' = (V, E \cup \{(u, w)\})$ does not always bring a modularity loss.*

Proof. By manipulating Eq. (2) we have that η (sum of edges within communities) remains unchanged while δ (sum of the degrees in all communities) becomes $\delta = \delta + 2 + 2deg(C_i) + 2deg(C_j)$. This gives the new modularity

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

The possible modularity loss is then

$$\begin{aligned}\mathcal{ML} &= \frac{\eta}{m} - \frac{\delta}{4m^2} - \frac{\eta}{m+1} + \frac{\delta + 2 + 2deg(C_i) + 2deg(C_j)}{4(m+1)^2} \\ &= \frac{\eta}{m(m+1)} + \frac{2m^2(deg(C_i) + deg(C_j) + 1) - \delta(2m+1)}{4m^2(m+1)^2}.\end{aligned}$$

Note that the above $\mathcal{ML}$ is not always greater than zero as it depends on the degree of the communities involved. □

The highest possible loss is obtained by picking as source and target communities for the edge addition, the communities having the highest degrees. If $\mathcal{C} \in \overline{C}$, the possible modularity loss depends on the rank (in terms of degree) of $\mathcal{C}$ in $\overline{C}$. If $\mathcal{C} \notin \overline{C}$, the result of Lemma 12 still holds; the edge insertion with the highest possible loss is (u, w): $u \in C_i \cap \mathcal{C}, w \in C_j$ and $deg(C_i) + deg(C_j)$ is maximal.

We now consider the addition of an intra-community edge. We omit the proofs of the following lemmas as they are similar to that of Lemma 12.

Lemma 13. *An intra-community edge addition (u, w): $u \in C_i \cap \mathcal{C}, w \in C_i$ giving $G' = (V, E \cup \{(u, w)\})$ does not always bring a modularity loss. We have that $\mathcal{ML} > 0$ if, and only if*

$$\frac{\eta - m}{m(m+1)} + \frac{4m^2(deg(C_i)+1) - \delta(2m+1)}{4m^2(m+1)^2} > 0. \quad (3)$$

Given an inter-edge addition between communities C_i and C_j (giving G') and an intra-community edge addition in the community C_i (giving the network G'') we have that: $\mathcal{M}_{G'}(\overline{C}) - \mathcal{M}_{G''}(\overline{C}) = \frac{deg(C_i)-deg(C_j)-2m-1}{2(m+1)^2}$. Since $deg(C_i) \leq 2m$, we have that $\mathcal{M}_{G'}(\overline{C}) - \mathcal{M}_{G''}(\overline{C}) < 0$ and, thus, $\mathcal{M}_{G''}(\overline{C}) > \mathcal{M}_{G'}(\overline{C})$ and thus the largest modularity loss can be obtained via an inter-community edge addition.

4.1.2 Edge Deletions

We start with inter-community edge deletions.

Lemma 14. *For any inter-community edge deletion (u, w): $u \in C_i \cap \mathcal{C}, w \in C_j$, with $i \neq j$ giving $G' = (V, E \setminus \{(u, w)\})$: $\mathcal{ML} > 0$ if, and only if*

$$\frac{\delta(2m-1) - 2m^2(deg(C_i) + deg(C_j) + 1)}{4m^2(m-1)^2} - \frac{\eta}{m(m-1)} > 0.$$

If $\mathcal{C} \in \overline{C}$, the (possible) modularity loss depends on the rank (in terms of degree) of $\mathcal{C}$ in $\overline{C}$. If $\mathcal{C} \notin \overline{C}$, the result of Lemma 14 still holds; the edge deletion with the possible highest loss is (u, w): $u \in C_i \cap \mathcal{C}$, $w \in C_j$, with $i \neq j$, where the sum of $deg(C_i)$ and $deg(C_j)$ is minimal. We now consider the deletion of an intra-community edge.

Lemma 15. *For any intra-edge deletion (u, w): $u \in C_i \cap \mathcal{C}$, $w \in C_i$ giving $G' = (V, E \setminus \{(u, w)\})$, $\mathcal{ML} > 0$ if and only if*

$$\frac{m - \eta}{m(m-1)} + \frac{\delta(2m-1) - 4m^2(deg(C_i) - 1)}{4m^2(m-1)^2} > 0.$$

The best edge deletion, in terms of modularity loss, is an intra-community edge in the community C_i having the lowest degree and such that $C_i \cap \mathcal{C} \neq \emptyset$. The modularity loss is the same no matter the pair of nodes $u \in C_i \cap \mathcal{C}$ and $w \in C_i$.

4.2 The $\mathcal{D}_m$ Algorithm

The deception algorithm $\mathcal{D}_m$, outlined in Algorithm 2, builds upon the analysis performed in Section 4.1. $\mathcal{D}_m$ at each step compares the two most convenient edge updates (line 8) as per Lemma 12 (line 4) and Lemma 15 (line 6) and picks the update giving the highest modularity loss. Finally, the update is applied to the network (lines 9 or 11).

Algorithm 2 requires knowledge of the community structure to compute degrees (line 3). Knowledge about the degrees is essential to pick the edge update that could give the highest modularity loss. We observe that the strategy adopted by Waniek et al. [35] is sub-optimal for two main reasons: (i) intra-community edge deletions and inter-community edge additions are picked randomly (without looking at the degrees of communities in $\overline{C}$); (ii) intra-community edge deletions and inter-community edge additions do not always bring a modularity loss.

Example 16. Fig. 4 shows the modularity losses computed by $\mathcal{D}_m$ when considering a budget $\beta = 4$, the target community $\mathcal{C}$ and the network in Fig. 2a. Note that the modularity loss for each edge update only depends on the (degrees of the) communities. As an example, for the first update, the best communities for an inter-community edge addition are $\mathcal{C}$ and C_3; this is because C_3 has the maximum degree (see Theorem 12); in this case, the modularity loss is $\mathcal{ML}_{add} = 0.0082$. For an intra-community edge deletion the loss is $\mathcal{ML}_{del} = 0.0079$ (see Theorem 15). In this case, the algorithm picks the inter-community edge addition and specifically two random nodes, one belonging to $\mathcal{C}$ and the other belonging to C_3. Note that since $\mathcal{C} \in \overline{C}$ (we assume in this example that our target community is completely revealed) the function `getLast` (that is supposed to select the intra-community edge to delete from the community having the minimum degree among the communities that contains at least one member of $\mathcal{C}$) has no effect since all the members of $\mathcal{C}$ belongs to the same community. The other updates are chosen in a similar way although the loss decreases as the algorithm suggests new edge additions between $\mathcal{C}$ and C_3.

Modularity($\mathcal{D}_m$)		
#Update	$\mathcal{ML}_{add}$	$\mathcal{ML}_{del}$
1	**.0082**	.0079
2	**.0082**	.0078
3	**.0080**	.0077
4	**.0078**	.0076

Fig. 4. Updates made by $\mathscr{D}_m$ for the network and $\mathscr{C}$ in Fig. 2a.

Theorem 17. *Algorithm 2 runs in time $\mathcal{O}(|E| + |\overline{C}|^2)$.*

Proof. Computing community degrees and sorting them can be done in $\mathcal{O}(|E| + |\overline{C}|\log(|\overline{C}|))$. By keeping information about the presence/absence of $\mathcal{C}$'s nodes in the various communities, the selection of the best edge insertion/deletion can be done in $\mathcal{O}(|\overline{C}|^2)$ (by comparing the communities pairwise). Finally, the comparison of losses and the update of the network can be done in constant time. □

Algorithm 2. $\mathcal{D}_m$ - Deception via Modularity

1: **procedure** comDeceptModularity($G = (V, E), \overline{C}, \mathcal{C}, \beta$)
2: **do**
3: degC ← `computeandSortComDegrees`$(\overline{C}, G)$
4: `select`$(n_p, n_t) \notin E, n_p \in C_i, n_t \in \mathcal{C} \cap C_j, (C_i, C_j) \in \arg\max(deg(C_i) + deg(C_j))$
5: $\mathcal{ML}_{add} \leftarrow$ `getAddLoss`$(n_p, n_t, \overline{C}, G)$ // Lemma 12
6: `get`$(n_k, n_l) \in E, \{n_k, n_t\} \subseteq$ `getLast`$(\text{degC}, \mathcal{C}), n_t \in \mathcal{C}$
7: $\mathcal{ML}_{del} \leftarrow$ `getDelLoss`$(n_k, n_l, \overline{C}, G)$ // Lemma 15
8: **if** $\mathcal{ML}_{del} \geq \mathcal{ML}_{add}$ and $\mathcal{ML}_{del} > 0$ **then**
9: $G \leftarrow (V, E \setminus \{(n_k, n_l)\})$
10: **else if** $\mathcal{ML}_{add} > 0$ **then**
11: $G \leftarrow (V, E \cup \{(n_p, n_t)\})$
12: $\beta \leftarrow \beta - 1$
13: **while** $\beta > 0$ and ($\mathcal{ML}_{add} > 0$ or $\mathcal{ML}_{del} > 0$)

Correctness of the $\mathcal{D}_m$. Given a community $\mathcal{C}$ and a budget β, the deception algorithm $\mathcal{D}_m$ updates the network G and produce a network G' such that $\mathcal{M}_{G'}(\overline{C}) \geq \mathcal{M}_G(\overline{C})$. In fact, it is enough to note that each update is selected, as per Lemmas 12 and 15, in order to maximize the modularity loss. Note that (lines 8-11) the best inter-community edge addition or intra-community edge deletion are applied only if they actually bring a modularity loss.

5 Experimental Evaluation

We now report on an extensive experimental evaluation of community deception algorithms. We describe the experimental setting, the evaluation methodology, and then report on the results. We implemented the deception algorithms in R[1] by also using the `igraph`[2] and `parallel` packages.

5.1 Experimental Setting

We consider the deception algorithm $\mathcal{D}_s$ (Algorithm 1) and $\mathcal{D}_m$ (Algorithm 2). We also considered a baseline algorithm ($\mathcal{D}_r$), which randomly selects both the type of update and the endpoints of the edge addition/deletion. Finally, we implemented the approach by Waniek et al. [35] ($\mathcal{D}_w$), which randomly adds external edges and deletes internal edges. We note that the algorithm $\mathcal{D}_m$ is actually an improved version of $\mathcal{D}_w$, as discussed in Section 4.

Detection Algorithms. As adversaries to the deception algorithms we considered ten detection algorithms. The first nine are available in the R version of the `igraph` library. The code of the tenth algorithm is available online.[3]

(1) Louvain [2] (`louv`): a multi-level modularity optimization algorithm, which runs in time $\mathcal{O}(|V| \log |V|)$;
(2) WalkTrap [25] (`walk`): based on the idea that random walks are more likely to stay in the same community. This algorithm runs in time $\mathcal{O}(|V|^2 \log|V|)$;
(3) Greedy [6] (`gre`): based on a greedy modularity maximization strategy and running in time $\mathcal{O}(|V|\log^2|V|)$;
(4) InfoMap [30] (`inf`): it returns a community structure, which provides the shortest description length for a random walk. This algorithm runs in time $\mathcal{O}(|E|)$;
(5) Label propagation [27] (`lab`): it assigns to every node one of k labels and then re-assigns labels to nodes in a way that each node takes the most frequent label of its neighbors. This algorithm runs in time $\mathcal{O}(|E|)$;
(6) Leading Eigenvectors [22] (`eig`): a top-down hierarchical approach based on modularity. Its complexity is $\mathcal{O}(|V|(|E|+|V|))$, or $\mathcal{O}(|V|^2)$ on a sparse graph;
(7) Edge-Betweeness [17] (`btw`): a hierarchical decomposition process where edges are removed in the decreasing order of their edge betweenness scores. This algorithm runs in time $\mathcal{O}(|E|^2|V|)$;
(8) SpinGlass [28] (`spin`): it reduces community detection to the problem of finding the ground state of an infinite spin glass. The complexity is $\mathcal{O}(V^{3.2})$;
(9) Optimal [3] (`opt`): it uses integer programming and has exponential complexity.
(10) Scalable Community Detection [26] (`scd`), which partitions the graph by maximizing the Weighted Community Clustering, a community detection metric based on triangle analysis. The time complexity is $\mathcal{O}(|E| \log|V|)$.

Experiments have been conducted on a PC i5 CPU 3.0 GHz (4 cores) and 16 GBs RAM. Results reported are the average (95 percent confidence interval) of 10 runs.

1. https://www.r-project.org
2. http://igraph.org/r
3. https://github.com/DAMA-UPC/SCD

5.2 Evaluation Methodology

Since there is no standard benchmark to test deception algorithms we adopted the methodology in Algorithm 3. We start with a community structure $\overline{C}$ found by some detection algorithm $\mathcal{A}_D$ that we want to deceive (line 2)—as our goal is to show to what extent deception algorithms can hide $\mathcal{C}$ inside $\overline{C}$, the performance of $\mathcal{A}_D$, that is, its ability to return the right communities is orthogonal to our goal. We ran the experiments in the worst-case scenario (line 3), that is, assuming that the target community $\mathcal{C}$ is fully discovered (i.e., $\mathcal{C} \in \overline{C}$). We pick target communities of various sizes by looking at the community distribution for a specific detection algorithm (see Fig. 9 in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2017.2776133). We compute (lines 4-6) the values of modularity, safeness and deception score; then, we apply a budget of changes β such that $|E^+| + |E^-| \leq \beta$ to the initial network according to one of the considered deception algorithms ($\mathcal{D}_s, \mathcal{D}_m, \mathcal{D}_r, \mathcal{D}_w$) and obtain an updated networks G' (line 7).

Algorithm 3. Evaluating Community Deception Algorithms

1: **procedure** EVALUATEDECEPTIONALGOG, (β, $\mathcal{A}_D$, $\mathcal{D}$)
2: $\overline{C} \leftarrow \mathcal{A}_D(G)$ /* Communities before deception */
3: $\mathcal{C} \leftarrow$ `getTargetCommunity`($\overline{C}$)
4: $\mathcal{M}_G(\overline{C}) \leftarrow$ `getModularity`($\overline{C}, G$)
5: $\sigma(\mathcal{C}) \leftarrow$ `getSafeness`($\mathcal{C}, G$)
6: $\mathcal{H}(\mathcal{C}, \overline{C}) \leftarrow$ `getDeception`($\mathcal{C}, G, \overline{C}$)
7: $G' \leftarrow$ applyDeception($G, \mathcal{C}, \beta, \mathcal{D}_{\mathbf{x}}$) /* $\mathbf{x} \in \{s, m, r, w\}$*/
8: $\overline{C}' \leftarrow \mathcal{A}_D(G')$; /* Coms. after deception */
9: $\mathcal{M}_{G'}(\overline{C}') \leftarrow$ `getModularity`($\overline{C}', G'$);
10: $\sigma(\mathcal{C}') \leftarrow$ `getSafeness`($\mathcal{C}, G'$)
11: $\mathcal{H}(\mathcal{C}, \overline{C}') \leftarrow$ `getDeception`($\mathcal{C}, \overline{C}', G'$)

At this point, we run (again) $\mathcal{A}_D$ on G' (lines 8) and compute the values of modularity, safeness and deception score on the updated networks to measure the impact of the deception strategies. Finally, the discovery of a new node in the case of edge additions performed by the $\mathcal{D}_s$ and $\mathcal{D}_w$ algorithms is simulated by picking a random node.

The Goals of the Evaluation are to Investigate. (i) how deception algorithms can hide a community $\mathcal{C}$ from detection algorithms; (ii) the safeness ($\mathcal{D}_s$), modularity ($\mathcal{D}_m$), random-based ($\mathcal{D}_r$) and Wakiek's et al. [35] ($\mathcal{D}_w$) deception algorithms; (iii) how to set the budget β; (iv) how deception affects the community structure; (v) the running time of deception and detection algorithms.

5.3 Datasets

We considered real networks from a wide range of domains, including social networks, collaboration networks, communication networks, citation networks, affiliation networks, and product co-purchasing networks. Table 2 gives an overview of the networks considered. Due to their variety, we do not report experiments on artificially generated networks for which planted communities are produced (e.g., [18]). Peel et al. [24] recently observed that working with planted communities does not reflect the true data generating process for real networks, which is typically unknown. Table 2 also reports the number of communities found by each detection

TABLE 2
Real Networks Considered

Net	$\|V\|$	$\|E\|$	Description	Number of communities									
				scd	louv	walk	gre	inf	lab	eig	btw	spin	opt
kar	34	78	Zachary Karate's Club[a]	8	4	6	3	3	3	4	5	4	4
dol	62	159	Dolphins association[a]	26	5	4	4	5.5	3.5	5	5	5	5
mad	62	159	Madrid Train Bombing[b]	26	6	6	7	7	3	4.5	12		6
lesm	77	154	Co-appearances of characters in "Les Miserables"[a]	35	6	8	5	9	7	8	11	6	6
polb	105	441	Books about US politics[a]	23	4	4	4	6	3	4	5	6	5
words	112	425	Adjectives/nouns in the novel David Copperfield[a]	57	7	25	7	2	1	10	69	7.5	
nets	1589	5486	Net Theory co-authors[a]	631	406	416	403	442	457	404	405		
erdos	6100	9939	Erdős collaborations[c]	196	47	214	57	313	310	44	88	26	
fb-75	6387	217K	Facebook [32])	998	20	358	25	135	18	14	152	20	
pow	4941	6594	USA Power Grid[a]	4326	40	364	41	487	515	35	45	25	
astr	17.9K	197K	Astrophysics co-authors[b]	5116	32	2264	146	890	341	28		1988	
4sq	639K	3.2M	Foursquare [16]	563K	62	1345	345	1097	25	39			
dblp	540.5K	15.2M	DBLP[b]	62K	234	11.5K	2331	15.1K	21K			25	
amz	548K	925K	Amazon metadata[b]	141K	32		215K	230K	236K				
you	1M	3M	Youtube dataset[b]	999K	36K		43K	97K	60K				
ork	3M	117.2M	Orkut social network[b]	634K	228								

[a] *http://www-personal.umich.edu/∼mejn/netdata;*
[b] *http://snap.stanford.edu/data;*
[c] *http://vlado.fmf.uni-lj.si/pub/networks/pajek/data/gphs.htm.*

algorithm. Some algorithms could not handle networks above a certain size. The distribution of communities for the considered networks and algorithms is reported in the Appendix (Fig. 9), available in the online supplemental material.

5.4 Deception Evaluation

We start with the first six networks in Table 2. Fig. 5 provides a detailed view of the deception score when considering $\mathcal{D}_m$ and $\mathcal{D}_s$ and budgets values β from 1 to 5. Since both the random algorithm $\mathcal{D}_r$, and the $\mathcal{D}_w$ algorithm performed poorly (viz., $\mathcal{H} \sim 0$), we do not report their values.

Remarks. We can see that safeness-based deception (i.e., $\mathcal{D}_s$) performs better than modularity-based deception (i.e., $\mathcal{D}_m$) as the budget (x-axis) increases. The deception score increases with the budget β, for almost all the community detection algorithms. Nevertheless, there are a few exceptions when using the modularity-based deception algorithm $\mathcal{D}_m$. As an example, on the network kar, modularity-based deception applied to the inf algorithm reaches its maximum value with $\beta = 1$; a similar behavior can be observed on the words network and the spin and lab algorithms.

We can explain such behavior with the fact that these algorithms are not based on modularity maximization and thus the $\mathcal{D}_m$ algorithm, which generates an updated network with a lower value of modularity does not bring enough disruption to the functioning of inf for larger budget values. Note also that modularity-based algorithms like louv and btw always show an increase of the deception score for an increased budget value when applying $\mathcal{D}_m$. On the other hand, we observe that $\mathcal{D}_s$ consistently increases the value of deception (also) when considering non-modularity-based detection algorithms. $\mathcal{D}_s$ represents a more general way of attacking community deception than the approach implemented by $\mathcal{D}_m$ based on modularity. While $\mathcal{D}_m$ aims at maximizing the modularity loss only, $\mathcal{D}_s$ also looks at reachability, which is also taken into account in the deception score (see Definition 1). We observed that in some cases after applying the updates suggested by $\mathcal{D}_m$ the community $\mathcal{C}$ is disconnected; this is avoided by $\mathcal{D}_s$. We observed that (results not reported for sake of space) safeness increases as the budget increases while modularity behaves the opposite way in most of the cases; there are cases (e.g., the network words) where no modularity loss could be observed. This is in-line with the analysis performed in Sections 3 and 4, respectively.

We now move our analysis to larger networks (i.e., power, fb75, erdos, and nets). The values of the deception score when increasing the budget β from 1 to 5 are reported in Fig. 6. Here, we also report the deception score for the $\mathcal{D}_w$ algorithm and baseline algorithm $\mathcal{D}_r$ (last row).

Remarks. The baseline $\mathcal{D}_r$, which picks β edge updates randomly performs poorly. We note an improvement when using the $\mathcal{D}_w$ algorithm, which differently from $\mathcal{D}_r$, constraints the possible updates to only intra-edge deletion and inter-edge additions. $\mathcal{D}_s$ and $\mathcal{D}_m$, despite the moderate budget value ($\beta \leq 5$), perform reasonably well considering that the experiments are conducted in the worst case scenario (i.e., starting from $\mathcal{H} = 0$). We also observe that $\mathcal{D}_s$ performs better than $\mathcal{D}_m$ in almost all scenarios and that the latter performs consistently better than $\mathcal{D}_w$. This is in line with the analysis performed in Section 4 where we showed that the best modularity loss (at the core of both $\mathcal{D}_w$ and $\mathcal{D}_m$) can be obtained when considering the degrees of communities; this piece of information is available to $\mathcal{D}_m$ but neither it is to $\mathcal{D}_w$ nor $\mathcal{D}_s$. On these networks, it emerges that: (i) $\mathcal{D}_s$ gives higher values

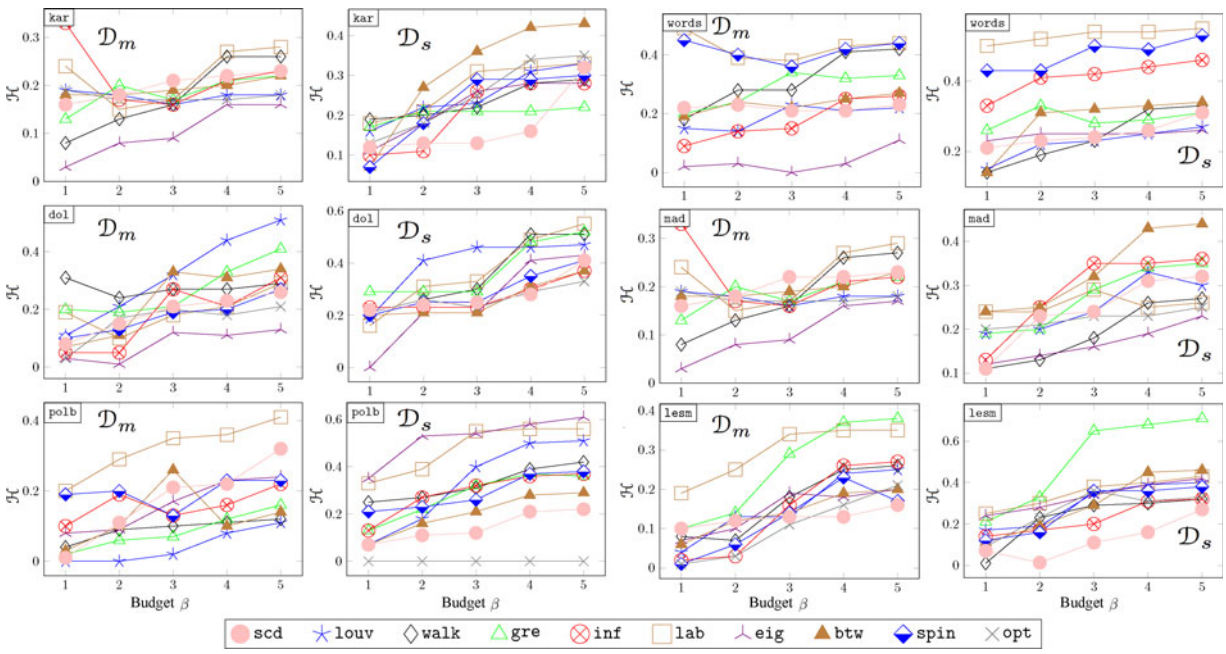

Fig. 5. Deception score on small networks. The y-axis represents the deception score ($\mathscr{H}$) obtained either by using either $\mathscr{D}_m$ or $\mathscr{D}_s$.

USA Power Grid (power)

$\mathcal{H}$	$\beta=1$			$\beta=2$			$\beta=3$			$\beta=4$			$\beta=5$		
	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$
scd	.25	.31	.34	.31	.34	.39	.40	.46	.46	.46	.48	.47	.43	.49	.50
louv	.16	.18	.26	.32	.35	.26	.32	.38	.35	.45	.47	.50	.39	.42	.50
walk	.31	.36	.48	.24	.29	.44	.37	.38	.46	.45	.47	.46	.46	.48	.48
gre	.44	.47	.46	.39	.48	.46	.44	.45	.49	.50	.50	.49	.50	.52	.49
inf	.13	.14	.26	.15	.17	.27	.37	.37	.36	.37	.38	.37	.43	.43	.44
lab	.21	.24	.28	.32	.36	.35	.35	.37	.35	.39	.40	.39	.37	.37	.42
eig	.27	.29	.25	.26	.27	.35	.34	.35	.38	.32	.36	.38	.36	.38	.39
btw	.16	.19	.21	.29	.31	.33	.32	.31	.42	.29	.44	.43	.41	.43	.46
spin	.32	.35	.32	.33	.34	.38	.39	.40	.41	.38	.42	.45	.40	.40	.45
$\mathcal{D}_r$	.05			.11			.09			.12			.11		

Facebook (fb-75)

$\mathcal{H}$	$\beta=1$			$\beta=2$			$\beta=3$			$\beta=4$			$\beta=5$		
	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$
scd	.00	.00	.00	.10	.11	.13	.10	.11	.13	.10	.12	.15	.18	.21	.26
louv	.00	.00	.00	.01	.02	.27	.21	.27	.30	.21	.29	.27	.21	.29	.28
walk	.00	.00	.00	.00	.00	.02	.00	.00	.02	.24	.25	.24	.24	.25	.24
gre	.02	.04	.04	.12	.27	.29	.23	.27	.29	.19	.20	.27	.21	.27	.30
inf	.31	.37	.37	.40	.50	.50	.43	.48	.50	.43	.50	.50	.42	.43	.50
lab	.26	.28	.28	.26	.28	.28	.48	.50	.28	.41	.50	.50	.20	.27	.50
eig	.01	.02	.02	.20	.21	.29	.30	.32	.30	.32	.35	.37	.36	.40	.41
btw	.20	.21	.21	.28	.31	.27	.29	.31	.29	.26	.29	.38	.29	.32	.41
spin	.01	.02	.01	.01	.03	.09	.10	.12	.11	.19	.21	.18	.19	.21	.23
$\mathcal{D}_r$	.01			.02			.01			.03			.06		

Erdos Number (erdos)

$\mathcal{H}$	$\beta=1$			$\beta=2$			$\beta=3$			$\beta=4$			$\beta=5$		
	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$
scd	.09	.11	.12	.21	.21	.22	.20	.31	.24	.20	.21	.27	.27	.29	.32
louv	.01	.02	.02	.24	.27	.26	.22	.27	.26	.26	.28	.47	.30	.36	.47
walk	.21	.25	.26	.22	.26	.34	.21	.35	.34	.31	.34	.34	.42	.46	.41
gre	.04	.05	.05	.22	.26	.26	.34	.37	.26	.10	.14	.32	.31	.32	.36
inf	.21	.25	.33	.24	.25	.29	.41	.45	.35	.41	.45	.45	.40	.44	.46
lab	.42	.43	.47	.40	.43	.46	.39	.44	.44	.42	.44	.44	.39	.44	.48
eig	.00	.00	.00	.36	.37	.26	.36	.44	.45	.30	.33	.45	.40	.46	.45
btw	.01	.01	.02	.09	.11	.13	.19	.21	.22	.29	.31	.29	.32	.36	.38
spin	.02	.03	.06	.20	.21	.23	.21	.31	.25	.28	.30	.35	.36	.38	.40
$\mathcal{D}_r$	.06			.09			.08			.11			.09		

Netscience Collaborations (nets)

$\mathcal{H}$	$\beta=1$			$\beta=2$			$\beta=3$			$\beta=4$			$\beta=5$		
	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$
scd	.20	.21	.23	.21	.22	.26	.13	.23	.26	.38	.43	.32	.22	.32	.36
louv	.06	.08	.25	.20	.22	.25	.32	.36	.25	.09	.12	.25	.32	.36	.37
walk	.21	.25	.24	.19	.22	.28	.26	.25	.28	.23	.45	.32	.45	.45	.43
gre	.12	.15	.28	.23	.25	.28	.21	.25	.28	.36	.45	.28	.35	.44	.29
inf	.21	.22	.22	.18	.21	.22	.15	.16	.23	.21	.22	.25	.21	.24	.25
lab	.10	.12	.13	.13	.14	.15	.31	.33	.22	.28	.29	.25	.20	.23	.39
eig	.21	.25	.13	.21	.26	.13	.24	.25	.47	.43	.46	.47	.42	.45	.47
btw	.14	.15	.15	.11	.12	.25	.16	.17	.25	.32	.34	.25	.31	.33	.37
$\mathcal{D}_r$	.07			.08			.06			.07			.11		

Astrophysics (astr)

$\mathcal{H}$	β=1%·|$\mathcal{E}$|			β=5%·|$\mathcal{E}$|			β=10%·|$\mathcal{E}$|			β=20%·|$\mathcal{E}$|			β=30%·|$\mathcal{E}$|		
	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$
scd	.18	.19	.12	.22	.29	.39	.36	.38	.47	.47	.51	.51	.45	.58	.74
louv	.28	.29	.14	.36	.40	.27	.37	.41	.42	.50	.57	.78	.46	.63	.81
walk	.12	.14	.13	.34	.38	.29	.38	.42	.41	.36	.49	.66	.47	.57	.72
gre	.10	.10	.16	.38	.40	.23	.50	.57	.33	.49	.60	.44	.65	.71	.73
inf	.04	.06	.12	.06	.09	.12	.32	.35	.22	.23	.42	.26	.43	.50	.72
lab	.20	.21	.13	.31	.30	.28	.31	.39	.36	.40	.70	.58	.69	.72	
eig	.16	.19	.07	.22	.26	.21	.29	.30	.44	.44	.45	.78	.54	.63	.83
spin	.21	.22	.21	.41	.44	.44	.42	.46	.50	.48	.54	.53	.39	.58	.57

DBLP (dblp)

$\mathcal{H}$	β=1%·|$\mathcal{E}$|			β=5%·|$\mathcal{E}$|			β=10%·|$\mathcal{E}$|			β=20%·|$\mathcal{E}$|			β=30%·|$\mathcal{E}$|		
	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$
scd	.01	.02	.06	.09	.10	.33	.29	.33	.66	.29	.33	.70	.49	.64	.82
louv	.00	.00	.15	.10	.11	.20	.12	.14	.30	.21	.23	.67	.50	.62	.74
walk	.05	.06	.15	.12	.14	.35	.11	.14	.49	.22	.26	.75	.49	.66	.79
gre	.06	.08	.09	.20	.20	.13	.14	.24	.40	.28	.39	.46	.50	.59	.80
inf	.07	.08	.08	.12	.14	.16	.16	.19	.31	.31	.32	.40	.38	.49	.72
lab	.09	.11	.07	.13	.14	.40	.21	.24	.59	.30	.31	.68	.51	.53	.52
spin	.10	.12	.13	.16	.17	.16	.22	.25	.26	.32	.33	.35	.52	.57	.59

Youtube (you)

$\mathcal{H}$	β=1%·|$\mathcal{E}$|			β=5%·|$\mathcal{E}$|			β=10%·|$\mathcal{E}$|			β=20%·|$\mathcal{E}$|			β=30%·|$\mathcal{E}$|		
	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$
scd	.10	.11	.15	.21	.24	.26	.39	.59	.32	.39	.59	.56	.47	.65	.80
louv	.00	.00	.04	.29	.35	.10	.32	.38	.26	.41	.46	.39	.55	.60	.43
gre	.00	.00	.06	.25	.28	.12	.30	.31	.14	.43	.50	.31	.48	.62	.55
inf	.00	.00	.09	.00	.00	.11	.21	.32	.42	.29	.33	.47	.39	.53	.56
lab	.03	.05	.10	.09	.11	.19	.18	.14	.44	.29	.38	.54	.58	.62	.83

Foursquare (4sq)

$\mathcal{H}$	β=1%·|$\mathcal{E}$|			β=5%·|$\mathcal{E}$|			β=10%·|$\mathcal{E}$|			β=20%·|$\mathcal{E}$|			β=30%·|$\mathcal{E}$|		
	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$
scd	.09	.12	.08	.12	.15	.05	.26	.31	.24	.31	.39	.30	.47	.59	.80
louv	.00	.00	.04	.21	.23	.10	.32	.32	.53	.39	.42	.63	.51	.59	.83
walk	.09	.12	.03	.20	.21	.13	.23	.28	.22	.44	.51	.39	.67	.70	.77
gre	.04	.07	.14	.22	.28	.57	.32	.37	.72	.42	.46	.79	.45	.75	.84
inf	.01	.02	.02	.09	.09	.32	.09	.14	.72	.30	.36	.72	.69	.71	.78
lab	.03	.05	.08	.04	.07	.14	.19	.20	.40	.31	.35	.48	.44	.61	.52
eig	.04	.09	.35	.35	.40	.55	.45	.48	.68	.44	.53	.80	.51	.58	.87

Amazon (amz)

$\mathcal{H}$	β=1%·|$\mathcal{E}$|			β=5%·|$\mathcal{E}$|			β=10%·|$\mathcal{E}$|			β=20%·|$\mathcal{E}$|			β=30%·|$\mathcal{E}$|		
	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$
scd	.00	.00	.12	.23	.30	.15	.32	.39	.16	.29	.44	.27	.53	.61	.37
louv	.00	.00	.15	.21	.31	.25	.33	.40	.59	.44	.47	.63	.47	.56	.73
gre	.10	.16	.13	.12	.20	.27	.12	.21	.36	.21	.28	.44	.30	.32	.60
inf	.21	.38	.10	.34	.40	.11	.24	.43	.46	.30	.39	.74	.23	.38	.82
lab	.01	.02	.06	.21	.24	.22	.31	.31	.45	.28	.34	.64	.19	.20	.69

Orkut (ork)

$\mathcal{H}$	β=1%·|$\mathcal{E}$|			β=5%·|$\mathcal{E}$|			β=10%·|$\mathcal{E}$|			β=20%·|$\mathcal{E}$|			β=30%·|$\mathcal{E}$|		
	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$	$\mathcal{D}_w$	$\mathcal{D}_m$	$\mathcal{D}_s$
scd	.09	.12	.12	.11	.15	.19	.14	.17	.54	.21	.25	.66	.56	.71	.80
louv	.08	.10	.15	.23	.33	.37	.43	.51	.66	.49	.57	.77	.52	.67	.90

Fig. 6. Deception score $\mathscr{H}$ when the number of updates β is fixed (power, fb-75, erdos, nets) and when it is a percentage of the size of $\mathscr{E}$ (astr, dblp, amz, 4sq, you, ork). Darker colors represent larger values of $\mathscr{H}$ and thus better performance for the deception algorithm considered.

of $\mathcal{H}$ in all the four networks as β increases; (ii) the most difficult algorithm to deceive appears to be inf for all networks but fb-75 (in this case it is walk); (iii) the most vulnerable algorithms are gre, eig, lab, and inf for pow, nets, erdos, and fb-75, respectively; these rankings are consistent, no matter the deception algorithm used.

From these experiments, it emerged that the larger the size of $\mathcal{C}$ the lower the values of $\mathcal{H}$ when the total number of communities is in the order of a hundred. This is true, for example, for netsc and the louv algorithm, where there are around 100 communities the average size of which is larger than that found by other detection algorithms [14]. Observe also that spin is the least robust to deception on fb-75; in this case, we noted that despite the average size of $\mathcal{C}$ is the largest, the algorithm returns a few communities, some of which bigger than those returned by other algorithms. By looking at inf we observed that the average size of $\mathcal{C}$ is the smallest in all networks but fb-75; in all these networks inf is the most robust-to-deception algorithm. On fb-75 where the size of $\mathcal{C}$ is the second-most large and the distribution of communities is long tailed (see Fig. 9d in the Appendix, available in the online supplemental material) inf performs poorly deception-wise. Hence, the level of $\mathcal{H}$ is influenced both by the average size of $\mathcal{C}$ and the distribution of communities. Building

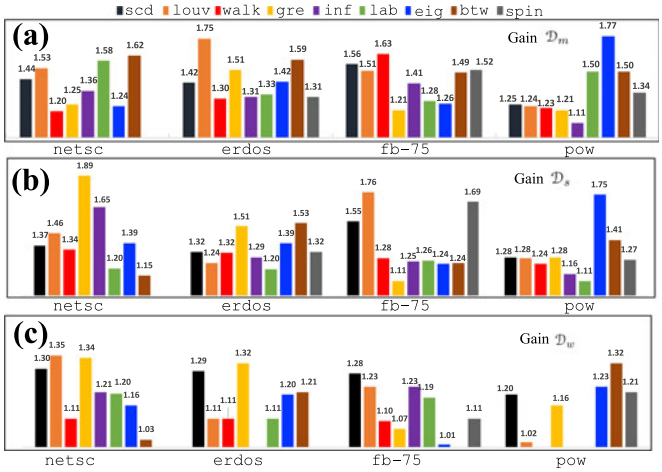

Fig. 7. Gain for $\mathscr{D}_m$ (a) $\mathscr{D}_s$ (b), and $\mathscr{D}_w$ (c).

upon this observation, we now study a different strategy to decide the budget of updates for deception algorithms.

5.4.1 Automatic Budget

We devised a strategy that sets the budget β as a function of the size of the target community $\mathcal{C}$. The rational behind this choice is that picking a percentage of $\mathcal{C}$ members that should be involved in the deception process reflects the fact that deception is a cooperative process. We considered 5 different budget values: 1, 5, 10, 20, and 30 percent of the number of nodes in $\mathcal{C}$. We apply this strategy to the $\mathcal{D}_m$, $\mathcal{D}_s$, and $\mathcal{D}_w$ algorithms and define the gain as the ratio between the deception score for β depending on the size of $\mathcal{C}$ and a fixed β. Fig. 7 reports the value of the gain by considering budget values of $\beta = 5$ and $\beta = 0.3|\mathcal{C}|$, respectively.

Remarks. Setting the budget as a function of $\mathcal{C}$'s size pays back (the gain is, in all cases, greater than one). This comes as no surprise as, in general, the new strategy considers a larger number of updates (when $|\mathcal{C}| > 18$) than the fixed-budget strategy. As an example, for `louv` and the `erdos` networks, where the gain is ∼75 percent for $\mathcal{D}_m$, ∼24 percent for $\mathcal{D}_s$, and ∼11 percent for $\mathcal{D}_w$; the maximum number of updates is now 30 instead of 5. The best gain, for all the deception algorithms, was obtained when deceiving the `eig` algorithm on the network `pow`; in this case, the maximum number of updates was 8 (instead of 5). A further consideration that underlines the need to have a dynamic budget is the fact that we observed that with very large networks like `ork`, deciding how to pick the right value of β is challenging. As an example, when fixing the budget to $\beta = 5$ we obtained a deception score equals to zero. As a final note, considering larger budget values up to 30 percent of the size of $\mathcal{C}$ is bearable in terms of running time (see Section 5.6).

We now discuss experiments on `astr`, `4sq`, `dblp`, `amz`, `you`, and `ork`. Here, we adopted the strategy that sets the budget as a function of $\mathcal{C}$'s size.

Remarks. We can observe (Fig. 6) that in networks like `astr` modularity-based deception ($\mathcal{D}_m$) is able to increase the deception score from 0 to 0.4 with $\beta =$∼10 percent of the number of nodes in $\mathcal{C}$ while $\mathcal{D}_s$ gives slightly lower values of $\mathcal{H}$ for this budget. As for $\mathcal{D}_w$, its performance, in terms of deception score, are always worse than that of $\mathcal{D}_m$ and $\mathcal{D}_s$. This is in line with the results obtained on smaller networks.

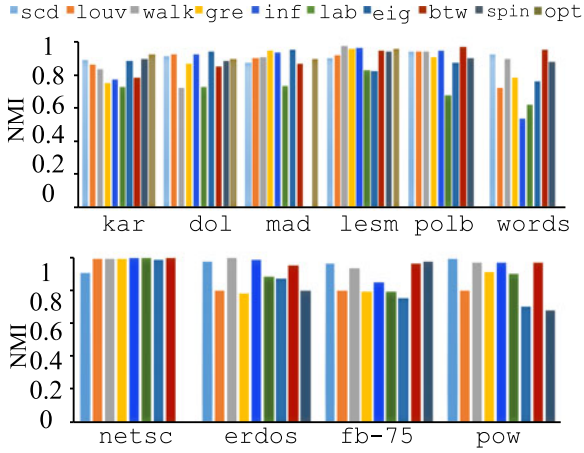

Fig. 8. NMI after Safeness-based ($\mathscr{D}_s$) deception.

Generally, we note that $\mathcal{D}_s$ performs consistently better than $\mathcal{D}_m$ and $\mathcal{D}_w$ for $\beta = 30$ percent of $|\mathcal{C}|$. The number of updates in this case is in the order of 100 for `louv` while it is in the order of 10 for the other algorithms. On larger networks like `4sq` to obtain a deception score of 0.4 a budget of ∼30 percent is required for $\mathcal{D}_m$ while $\mathcal{D}_s$ reaches that value with β ∼10 percent. In this case, the number of updates is around 100 for `louv` while it is around 30 for the other algorithms. For `dblp` we can observe that the deception score is, in general, lower for budget values < 30 percent while it reaches roughly the same values (∼0.6), as in `astr` and `4sq`, when the budget is 30 percent (roughly for `louv`). Note also that the most difficult algorithm to deceive when using $\mathcal{D}_m$ is `inf`. This behavior is consistent with what observed with smaller networks; it confirms that the fact that this algorithm is not based on modularity, as $\mathcal{D}_m$ and partially $\mathcal{D}_w$ are, and thus rewiring connections to reduce modularity may not directly affect the working of `inf` (which strive to maximize a different objective function). On the largest network (`ork`) we can note that the `scd` algorithm is more robust than `louv`, even if with $\beta = 30$ percent the deception score is higher. For this network, the number of updates is in the order of hundreds.

5.5 Effects of Deception on Detection

We now study the impact of deception algorithms on $\overline{C}$ by investigating how much deception algorithms change the original (pre-deception) community structure. We use Normalized Mutual Information (NMI), a measure for evaluating the accuracy of detection algorithms. We treat as ground-truth the initial community structure $\overline{C} = \{C_1, C_2, \ldots C_k\}$ and measure the level of agreement with $\overline{C}' = \{C'_1, C'_2, \ldots C'_p\}$. The NMI is defined as

$$\text{NMI} = \frac{-2\sum_{i,j} N_{ij} log \frac{N_{ij} N_t}{N_i N_j}}{\sum_i N_i log \frac{N_i}{N_t} + \sum_j N_j log \frac{N_j}{N_t}}, \tag{4}$$

where N is the confusion matrix whose element N_{ij} is the number of the shared members between an initially detected community C_i and the community C_j after deception. N_i and N_j are the sum over row i and column j respectively, and $N_t = \sum_i \sum_j N_{ij}$. Fig. 8 reports the values of NMI when using safeness-based deception ($\mathcal{D}_s$) on small/medium networks.

TABLE 3
Time of Detection versus Deception (Secs.)

Network	Load	Upd-$\mathscr{D}_m$	Upd-$\mathscr{D}_s$	Detec
netsc	0.5	0.07	0.08	0.07
fb-75	1.13	0.58	0.54	1.95
astro	1.23	0.62	0.71	1.45
dblp	67.35	15.65	16.58	150.6
4sq	19.38	18.36	21.39	322.56
ork	493.45	34.56	39.84	754.23

Remarks. We observed smaller NMI values on smaller networks (top part of Fig. 8). As the size of the network increases the NMI approaches 1. We can explain such behavior with the fact that smaller networks have typically a smaller number of communities and thus to hide $\mathcal{C}$ a larger portion of the communities should be affected. In fact, our deception algorithm that in these experiments works in the worst case scenario, aims at spreading $\mathcal{C}$'s members in as many as possible communities in $\overline{C}$. If each $\mathcal{C}$'s member after deception becomes part of a $C_i \in \overline{C}$, such that $C_i \neq \mathcal{C}$, the structure of C_i is changed thus increasing the NMI. It is interesting to observe Fig. 8 also on the light of the number of (initial) communities found by deception algorithms (Table 2). For instance, on kar we note that algorithms such as gre, inf, and lab returning a lower number of communities as compared to the others, have lower NMI values. We can observe the same trend also on words, erdos, fb-75 and pow. As for netsc, we observe values NMI≅1 for all algorithms but scd. By looking at Table 2 we note that scd returns the highest number of communities; the other algorithms return almost the same number of communities. We observed the same NMI trends (i.e., larger values for smaller numbers of communities) also on larger networks.

5.6 Running Time Evaluation

We compared the running time of deception and detection algorithms when considering networks of different size. We considered the $\mathcal{D}_m$ and $\mathcal{D}_s$ algorithms; $\mathcal{D}_w$ gave running times similar to $\mathcal{D}_m$. The running time of deception and detection algorithms, reported in Table 3, has been split in two components: Load refers to the time spent to load the network in memory; Upd-$\mathcal{D}_s$ (resp., Upd-$\mathcal{D}_m$) refers to the time taken by $\mathcal{D}_s$ (resp., $\mathcal{D}_m$) to apply the β=0.3×$|\mathcal{C}|$ updates. Finally, Detection refers to the time taken by the fastest detection algorithm (scd in almost all the cases).

Remarks. On the smallest network (i.e., netsc) we note that the running time of $\mathcal{D}_m$ and $\mathcal{D}_s$ is comparable to the time taken to find the communities; with networks of increasing size there is a significant difference. On the largest network ork (the Orkut social network) counting 3M nodes and 117.2 M edges deception algorithms are ∼19 times faster than the fastest detection algorithm (scd in this case). This comes as no surprise since the complexity of scd is $\mathcal{O}(|E| \log |V|)$ while the complexity of deception algorithms depends on $|\overline{C}|$ for $\mathcal{D}_m$ and the number of nodes and edges in $\mathcal{C}$ for $\mathcal{D}_s$. We want also to mention that despite the fact that $\mathcal{D}_m$ is faster than $\mathcal{D}_s$, the former algorithm, to pick the best update in terms of modularity loss requires knowledge of the community structure given by a detection algorithm $\mathcal{A}_D$. Therefore, the "real" running time of $\mathcal{D}_m$ would require to also count the time taken by the detection algorithm to find the community structure. We want to mention that running times as discussed here have a limited practical utility. This is because applying deception (i.e., adding/deleting edges) in a real-life scenario (see Section 2.3) is more involved than performing edge updates as done in our experiments.

5.7 Evaluation Summary

From the extensive evaluation conducted it emerges that all deception algorithms are effective, also considering that experiments have been performed in the worst case scenario (i.e., $\mathcal{C}$ is completely revealed). We showed that our greedy algorithm behave well in real networks by allowing to reach high deception score values starting from the worst case scenario $\mathcal{H} = 0$. As an example, in ork (the largest network considered) the score was always > 0.5. In more detail, we observed that: (i) safeness-based deception ($\mathcal{D}_s$) performs better in almost all scenarios; (ii) $\mathcal{D}_m$ and $\mathcal{D}_w$ gives good results when deceiving modularity-based algorithms (e.g., louv, eig); (iii) setting a fixed budget value β works only for small networks while allocating the budget as a function of the number of $\mathcal{C}$ members is a viable solution especially on large networks.

We found that there exists a correlation between the distribution of communities and the success of deception strategies; typically, hiding a small $\mathcal{C}$ within a large number of (small) communities is easier as one would naturally expect. We also discussed running times, although they are not really informative in a real scenario, where applying updates is a different process than computer-simulated update. As for the impact of deception on the output of a detection algorithm, we noted that it is negligible as the size of the network and the number of communities increase. Finally, we mention that when performing the evaluation *not* in the worst case scenario (as done in this paper) the deception score can be significantly higher (up to 40 percent).

6 Conclusions and Future Work

So far the literature has focused on the design of community detection algorithms. However, in some others contexts there is the need to hide the presence of a community. In this paper, we studied this problem that we named *community deception*. We introduced a formalization of the problem as an optimization problem and presented concrete instantiations of our formulation based on greedy strategies. One of the main results of this paper is that community deception based on safeness merely requires local knowledge ($\mathcal{C}$'s members) and can bring clear privacy gains (in terms of deception score) even when faced with an adversary with global knowledge (i.e., community detection algorithms).

There are several avenues for future research. Devising other instantiations of the general ϕ function (see Definition 2) is certainly interesting. Also investigating detection algorithms that incorporate additional information including overlapping communities [15], [34], multidimensional networks [20], or network attributes [37] is intriguing.

In this paper, we have studied how to *deceive* detection algorithms. However, our study leaves open another problem worthy of attention: how can detection algorithms be made deception-aware? We are currently tackling this

problem by leveraging network history [7]; the idea is to analyze the changes in the community structure (e.g., via NMI as discussed in Section 5.5) and detect anomalies (e.g., a specific group of people that is now loosely connected). A more speculative yet intriguing line of future work is to investigate whether complex networks such as biological networks exhibit some (natural) form of deceptive behavior. The question that we asked ourselves is: do some sub-networks types in the cells rewire their connections to hide?

REFERENCES

[1] N. Barbieri, F. Bonchi, E. Galimberti, and F. Gullo, "Efficient and effective community search," *Data Mining Knowl. Discovery*, vol. 29, no. 5, pp. 1406–1433, 2015.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Statistical Mech.*, vol. 2008, no. 10, 2008.

[3] U. Brandes, et al., "On modularity clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 2, pp. 172–188, Feb. 2008.

[4] A. Campan, Y. Alufaisan, T. M. Truta, and T. Richardson, "Preserving communities in anonymized social networks," *Trans. Data Privacy*, vol. 8, no. 1, pp. 55–87, 2015.

[5] J. Cheng, A. W.-C. Fu, and J. Liu, "K-isomorphism: Privacy preserving network publication against structural attacks," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 459–470.

[6] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, no. 6, pp. 66–111, 2004.

[7] N. Du, X. Jia, J. Gao, V. Gopalakrishnan and A. Zhang, "Tracking temporal community strength in dynamic networks," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 11, pp. 3125–3137, Nov. 2015.

[8] E. Ferrara, P. De Meo, S. Catanese, and G. Fiumara, "Detecting criminal organizations in mobile phone networks," *Expert Syst. Appl.*, vol. 41, no. 13, 2014.

[9] V. Fionda, L. Palopoli, S. Panni, and S. E. Rombo, "Protein-protein interaction network querying by a "focus and zoom" approach," *Bioinf. Res. Develo.*, vol. 13, pp. 331–346, 2008.

[10] V. Fionda, L. Palopoli, S. Panni, and S. E. Rombo, "A technique to search for functional similarities in protein-protein interaction networks," *Int. J. Data Mining Bioinf.*, vol. 3, no. 4, pp. 431–453, 2009.

[11] V. Fionda, C. Gutierrez, and G. Pirrò, "Building knowledge maps of Web graphs," *Artif. Intell.*, vol. 239, pp. 143–167, 2016.

[12] F. Folino and C. Pizzuti, "An evolutionary multiobjective approach for community discovery in dynamic networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1838–1852, Aug. 2014.

[13] S. Fortunato, "Community detection in graphs," *Physics Rep.*, vol. 486, no. 3, pp. 75–174, 2010.

[14] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proc. Nat. Academy Sci. United States America*, vol. 104, no. 1, pp. 36–41, 2007.

[15] E. Galbrun, A. Gionis, and N. Tatti, "Overlapping community detection in labeled graphs," *Data Mining Knowl. Discovery*, vol. 28, no. 5–6, pp. 1586–1610, 2014.

[16] H. Gao, J. Tang, and H. Liu, "Exploring social-historical ties on location-based social networks," in *Proc. 6th Int. AAAI Conf. Weblogs Social Media*, pp. 114–121, 2012.

[17] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proc. Nat. Academy Sci. United States America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[18] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," Ph. Rev. E, vol. 80, no. 1, 2009, Art. no. 016118.

[19] J. Leskovec, K. J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proc. 19th Int. Conf. World Wide Web*, pp. 631–640, 2010.

[20] X. Li and M. K. Ng, and Y. Ye, "MultiComm: Finding community structurein multi-dimensional networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 929–841, Apr. 2014.

[21] S. Nagaraja, "The impact of unlinkability on adversarial community detection: Effects and countermeasures," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, pp. 253–272, 2010.

[22] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, 2006, Art. no. 036104.

[23] M. E. Newman, "Modularity and community structure in networks," *Proc. Nat. Academy Sci. United States America*, vol. 103, no. 23, pp. 8577–8582, 2006.

[24] L. Peel, D. B. Larremore, and A. Clauset, "The ground truth about metadata and community detection in networks," *Sci. Advances*, vol. 3, no. 5, pp. 1–8, 2017.

[25] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Proc. Int. Symp. Comput. Inf. Sci.*, 2005, pp. 284–293.

[26] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey, "High quality, scalable and parallel community detection for large real graphs," in *Proc. 23rd Int. Conf. World Wide Web*, pp. 225–236, 2014.

[27] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E*, vol. 76, no. 3, 2007, Art. no. 036106.

[28] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Phys. Rev. E*, vol. 74, no. 1, 2006, Art. no. 016110.

[29] M. Revelle, C. Domeniconi, M. Sweeney, and A. Johri, "Finding community topics and membership in graphs," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2015, pp. 625–640.

[30] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Nat. Academy Sci. United States America*, vol. 105, no. 4, pp. 1118–1123, 2008.

[31] C. H. Tai, P. S. Yu, D. Yang, and M. Chen, "Structural diversity for resisting community identification in published social networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 235–252, Jan. 2014.

[32] A. L. Traud, P. J. Mucha, and M. A. Porter, "Social structure of facebook networks," *Physica A: Statistical Mech. Appl.*, vol. 391, no. 16, pp. 4165–4180, 2012.

[33] M. Wang, C. Wang, J. X. Yu, and J. Zhang, "Community detection in social networks: An in-depth benchmarking study with a procedure-oriented framework," *Proc. VLDB Endowment*, vol. 8, no. 10, pp. 998–1009, 2015.

[34] J. J. Whang, D. F. Gleich, and I. S. Dhillon, "Overlapping community detection using neighborhood-inflated seed expansion," *IEEE Trans. Knowl. Data Eng.*, vol 28, no. 5, pp. 1272–1284, May 2016.

[35] M. Waniek, T. Michalak, T. Rahwan, and M. Wooldridge, "Hiding individuals and communities in a social network," CoRR, abs/1608.00375, pp. 1–29, 2016.

[36] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 181–213, 2015.

[37] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *Proc. IEEE 13th Int. Conf. Data Mining*, 2013, pp. 1151–1156.

Valeria Fionda received the MSc degree in computer engineering and the PhD degree in mathematics and computer science from the University of Calabria. She is a researcher in DeMaCS, University of Calabria. Her research interests include in the areas of semantic web, process mining, and algorithms for graph querying and manipulation. She has published papers in top journals and conferences including *Artificial Intelligence*, the *ACM Transactions on the Web*, *VLDB Endowment*, IJCAI, AAAI, WWW, and CIKM.

Giuseppe Pirrò received the master's degree in computer engineering and the PhD degree in computer and system engineering from the University of Calabria. He is a researcher in the Institute for High Performance Computing and Networking (ICAR-CNR). His research interests include semantic web, graph databases, and distributed systems. He has published papers in top journals and conferences including *Artificial Intelligence*, the *ACM Transactions on the Web*, PVLDB, ISWC, AAAI, WWW, and CIKM.