Node-Centric Community Deception Based on Safeness

Saif Aldeen Madi and Giuseppe Pirrò

***Abstract*— Recent research has highlighted the importance of maintaining user privacy on online social networks. Specifically, despite their appealing applications, community detection algorithms expose personal relationships that might be misused against social network members. This concern has opened a new research area called community deception, which is about hiding the members of a target community from community detection algorithms. State-of-the-art deception methods only look at how community members must perform edge updates to guarantee some level of hiding. This article introduces node-centric deception, a novel approach considering nodes entering and leaving a target community. We theoretically study the effect of node updates by leveraging node safeness as a deception optimization function. Based on this analysis, we present an effective heuristic capable of hiding the target community with minimal node operations. We evaluated our approach against several community detection algorithms and compared it with state-of-the-art deception algorithms with encouraging results.**

***Index Terms*— Community deception, community detection, community hiding, privacy in social networks, social network analysis.**

I. Introduction

COMMUNITY detection is a well-established research problem within social network analysis, concerned with dividing networks into smaller structures, referred to as *communities* [9]. Such grouping of nodes is essential as it can reveal interesting relationships and social dynamics. It also has a range of applications in various domains, including law enforcement [30], e-commerce, and biology [17]. However, privacy concerns have surrounded community detection, mainly when detection algorithms are utilized in human social networks such as social media platforms or transaction networks, where we can imagine detection algorithms being misused to suppress legitimate activities on political grounds or to expose private interuser relationships for mere profit, ignoring fundamental user rights [11].

Such concerns led to a new strand of research called *community deception*. Community deception solves the following problem: given a target community $\mathcal{C}$, find and apply the best set of edge updates to be performed by the members of $\mathcal{C}$ so that $\mathcal{C}$ is better hidden from community detection algorithms in the resulting network.

Manuscript received 31 December 2022; revised 27 April 2023 and 10 July 2023; accepted 9 August 2023. *(Corresponding author: Giuseppe Pirrò.)*

Saif Aldeen Madi is with the Department of Computer Science, University of Rome La Sapienza, 00198 Rome, Italy (e-mail: madi@di.uniroma1.it).

Giuseppe Pirrò is with the Department of Mathematics and Computer Science, University of Calabria, 87046 Arcavacata, Italy (e-mail: giuseppe.pirro@unical.it).

Digital Object Identifier 10.1109/TCSS.2023.3306787

Motivations: Up to this point, research in community deception followed an edge-centric approach where the deception algorithm strategically modifies the network by adding/deleting *edges* to make the target community harder to detect. As a concrete example, Fig. 1(a) shows communities found by the Leiden [28] detection algorithm $\mathcal{A}_{\text{det}}$ on the Zachary's Karate Club network. We observe that $\mathcal{A}_{\text{det}}$ has identified four communities. We assume that the community we want to hide is $\mathcal{C} = \{24, 25, 26, 28, 29, 32\}$ (red-circled nodes); as $\mathcal{C}$ is completely revealed, we are in the worst case scenario, and thus the level of hiding of $\mathcal{C}$ in G measured, for instance, via the *deception score* [7] is $H_s(\mathcal{C},G) = 0$. One way to improve the situation is to use one of the existing edge-centric deception strategies like `dice` [29], `neur` [18], or `saf` [7]. With a budget of $\beta = 5$ edge updates to be performed by $\mathcal{C}$'s members, the deception score after running $\mathcal{A}_{\text{dec}}$ on the updated network G' (not shown for the sake of space) returned by these deception algorithms is of 0.365, 0.487, and 0.524, for `dice` [29], `neur` [18], and `saf` [7], respectively. However, nodes can also be central for hiding a community $\mathcal{C}$. Specifically, nodes *can naturally leave and enter the community* $\mathcal{C}$, and therefore, it is important to understand the deception-wise effect of such movements on the target community. But perhaps more importantly, there are practical advantages to the following a node-centric approach to deception. First, it allows a new incoming node to choose its connections, promoting the hiding of the target community. Referring back to Fig. 1(a), it is not clear, using edge-centric algorithms, what are the best edge updates to handle node 35 entering the community. Second, one could devise strategic node deletions (sacrificing some members) as part of the deception strategy; would the deletion of node 32 be more strategic deception-wise than that of node 24? As we will show, these questions can have answers only by making explicit the role of nodes in formulating the deception problem.

A. Contributions

We study the community deception problem from the perspective of node updates and contribute.

1) A modeling of *node-centric* deception relying on node and community safeness [7] as optimization function. Our modeling goes beyond the state-of-the-art optimization functions, which are oblivious to node updates. We emphasize that, in the previous example, considering nodes allows placing the new node 35 in the best deception-wise community (e.g., the community including the lowest number of $\mathcal{C}$'s members) by adding links to nodes within, and this cannot be achieved by edge-based deception which does not accommodate node additions. Enabling node updates besides edge updates opens

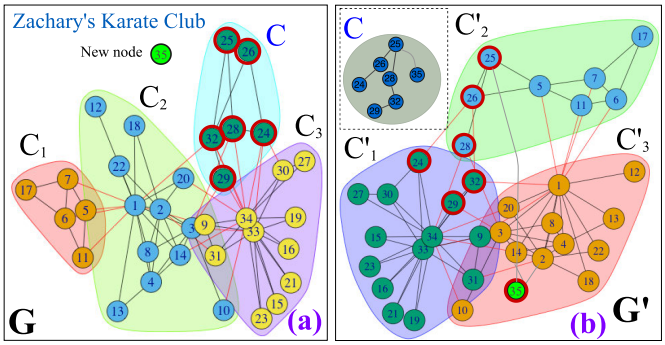

Fig. 1. Node-centric community deception. Communities (a) before and (b) after node-deception.

a new venue of deception schemes; one may consider adding fake nodes (bots) with specific connections as part of the deception strategy (e.g., revealing these fake nodes as part of $\mathcal{C}$ is less harmful than revealing the membership of real nodes). Another possibility is to delete and add back nodes in $\mathcal{C}$, a problem that has not been considered by edge-centric deception strategies yet. Although this operation is ultimately performed by deleting and adding edges, the edge additions returned by edge-centric and node-centric deception strategies are generally not the same. This is because node-centric deception incorporates node updates into the optimization function.

2) A theoretical study of the impact of node updates on safeness. Specifically, we focus on the safeness gain, the difference between the safeness of $\mathcal{C}$ before and after applying node updates. This also allows for singling out simple formulas to assess the most profitable type of update. The perspective of jointly considering node and edge updates discussed in this work opens the way to designing more comprehensive and realistic deception strategies that align with the nature of communities and networks where nodes can leave and join.

3) A greedy algorithm called nSAF to optimize our formulation of node-based deception as an optimization problem. When applying nSAF to the updated network in Fig. 1(a) and running the same detection algorithm $\mathcal{A}_{\text{det}}$, we obtain the network shown in Fig. 1(b) where node 35 was added to nodes in $\mathcal{C}$. In Fig. 1(b), the members of $\mathcal{C}$ are scattered in C_1, C_2, and C_3. However, when $\mathcal{C}$ is not completely revealed to $\mathcal{A}_{\text{det}}$, we can treat it as an additional *virtual community* and devise our deception strategy accordingly.

We emphasize that node-centered deception is neither subsumed by edge-centered deception nor aims at replacing it; these two kinds of updates on a community $\mathcal{C}$ have to be jointly considered. Moreover, the mentioned advantages of node-centric deception are of practical, real-world value. As communities are dynamic, deception algorithms should evaluate the effect of each incoming/leaving member on the community's hiding level. Even when assuming the role of a network owner, like a social media service provider, this research shows that the reliability of community detection can vary with the social network's growth.

B. Outline

The remainder of the article is organized as follows. We introduce some background notation and definitions in Section III. We present an analysis of node-based deception using safeness in Section IV. The nSAF algorithm is introduced in Section V. We report on an extensive experimental evaluation in Section VI. We conclude and sketch future work in Section VII.

TABLE I
COMPARISON OF NSAF WITH RELATED COMMUNITY DECEPTION ALGORITHMS

		nag	dice	saf	mmin	neur	nSAF
Strategy	Edge additions	✓	✓	✓	✓	✓	✓
	Edge deletions		✓	✓	✓	✓	✓
	Node additions						✓
	Node deletions						✓
Metric	Modularity	✓	✓		✓		
	Safeness			✓			✓
	Permanence					✓	

II. Related Work

The problem of *deception* in social networks to maintain user privacy from community detection algorithms has been first considered by some earlier works such as Nagaraja [19], which considered adding a set of edges to deceive detection algorithms. Most work on deception has been concerned with hiding a single community from detection by making edge modifications that optimize some objective function. `dice` [29] is based on the heuristic of deleting intracommunity edges and adding intercommunity edges with the assumption that such kinds of edge updates always minimize modularity. `mmin` [7] corrects for some issues with `dice`; the authors of `mmin` showed that in some cases, `dice` fails to perform edge updates that minimize modularity. `saf` [7] introduced node safeness as a deception objective function and used it to design a deception algorithm that requires limited network knowledge. Chen et al. [3] made small variations to `saf`. `neur` [18] tackled deception by using permanence minimization and showed that it outperformed several baselines. Table I summarizes the mentioned community deception approaches, showing the strategy they apply for deception along with the underlying objective function (metric) and comparing them with our proposed algorithm nSAF.

Note that except for nSAF, which employs node operations, the idea for other algorithms is to determine the best set of edge updates that, when applied to a network G, produces an updated network G' where the members of $\mathcal{C}$ are better hidden. For example, for all approaches adding an intercommunity edge, an edge between a node $u \in \mathcal{C}$ and a node in a different community, increases the level of hiding of $\mathcal{C}$. On the other hand, deleting an intercommunity edge would make $\mathcal{C}$'s members more susceptible to being identified as a community. Recently, this problem has been studied for directed networks [6] and influence networks [15].

A separate line of research (e.g., [14], [16]), instead of focusing on hiding a single community $\mathcal{C}$, strives to hide

the entire community structure. In particular, Liu et al. [14] introduced deception by *residual entropy minimization* (REM), where they followed an information theoretic approach to community structure deception. It is also noteworthy that REM uses only edge additions in its deception strategy. However, we believe it is less *realistic* to assume that *all nodes* in a network can cooperate by deleting or adding edges to escape detection algorithms. On the other side, [29] considered the problem of hiding only a single node and introduced ROAM—a heuristic approach for hiding individual social network users. Other more recent pieces of related work (e.g., [2], [13], [16]) have taken variants of community deception, but still, edge-centered. *This article aims to tackle community deception from the joint perspective of node and edge updates.*

III. Background and Problem Statement

Let $G(V, E)$ be an undirected network, where V is a set of n vertices, and E is the set of edges. A nonoverlapping community detection algorithm $\mathcal{A}_{\text{det}}$ produces a set of k communities $\bar{C} = \{C_1, C_2, \ldots, C_k\}$ or community structure. The community structure satisfies the following conditions: $\forall_{C_i \in \bar{C}} C_i \subseteq V$, $\bigcup_{C_i \in \bar{C}} C_i = V$, and $\forall_{C_i, C_j \in \bar{C}} C_i \cap C_j = \phi$, where $i \neq j$. We also define two types of edges; given a community $C \subseteq V$, an edge $(u, v) \in E$ is called an *intracommunity* edge, if and only if $u, v \in C$. On the other hand, (u, v) is called an *intercommunity* edge, if and only if $u \in C$ and $v \in C'$. Community deception is a conflict between two players with opposing interests: a *detector* and a *deceptor*. On one side, the *detector* is a detection algorithm $\mathcal{A}_{\text{det}}$ to discover the network's underlying communities, among which there may be the *target community* $\mathcal{C}$. Conversely, the deceptor $\mathcal{A}_{\text{dec}}$ strives to determine a set of strategic updates to guarantee that $\mathcal{C}$ remains hidden. By *hidden*, we mean that members of $\mathcal{C}$ should preferably be uniformly distributed [29] among various communities of $\bar{C}$. In this article, we take the role of the deceptor, where we consider community deception as a defense mechanism that aims to enhance the security of $\mathcal{C}$ by concealing its group identity. We assume that if $\mathcal{C}$ is exposed by $\mathcal{A}_{\text{det}}$, its security is breached. To quantify the level of hiding of a community, we rely on the *deception score* [7], which takes into account a more comprehensive range of aspects (e.g., reachability and spread of $\mathcal{C}$'s members among the various communities) than other measures related to community hiding (e.g., [19], [29]). Moreover, we conduct our study considering a worst case scenario—meaning that before applying $\mathcal{A}_{\text{dec}}$, the target community is entirely and uniquely detectable, i.e., $\mathcal{C} \in \bar{C}$. Our analysis will hold for more favorable scenarios, as the worst case scenario is not always actual.

A. Node Deception as an Optimization Problem

In the context of this article, the role of a deception algorithm $\mathcal{A}_{\text{dec}}$ is to find *nodes* to be deleted V^- and/or added V^+ to improve the level of hiding of $\mathcal{C}$. We note that some node additions represent actual community members who want to join, while others are fake members (bots). A sensible deception algorithm should treat these two categories of node additions differently. We will discuss this aspect in detail in Section V. $\mathcal{A}_{\text{dec}}$ also needs to select a set of edges to be deleted E^-, and another set to be added E^+, accordingly. This reasoning can be formalized via the following optimization problem:

$$\begin{aligned} &\text{MAXIMIZE}_{G'} \quad \phi\big(G, G', \mathcal{C}\big) \\ &\text{where} \quad G' = \big(V', E'\big) \\ &\qquad V' = \big(V \cup V^+\big) \backslash V^- \\ &\qquad V^- \subseteq \mathcal{C} \\ &\qquad E' = \big(E \cup E^+\big) \backslash E^- \\ &\qquad E^- = \big\{(u, v) | u \in V^- \vee v \in V^-, (u, v) \in E\big\} \\ &\qquad E^+ \subseteq \big\{(u, v) | u \in V^+ \vee v \in V^+, (u, v) \notin E\big\} \\ &\qquad |E^-| + |E^+| \leq \beta \end{aligned} \tag{1}$$

We note that the nodes to be deleted V^- must be target community nodes as we do not assume control over nodes outside $\mathcal{C}$; in this case, edges for deletion E^- are automatically selected since $\forall_u \in V^-$ it is enough to delete any edge connected with u. Each new node (real or fake) must connect to other network nodes, which forms the set E^+; this set of edge additions must be strategically determined. In (1), the total number of edge updates must not exceed a certain budget indicated with β.

B. Optimizing Safeness for Node Deception

We now need to define the deception function ϕ to be optimized. State-of-the-art edge-based deception strategies leverage different optimization functions. Some are based on modularity minimization (e.g., [7], [29]) others on permanence (e.g., [18]) or community safeness [7]. We focus on community safeness, the state-of-the-art community deception function [8]. *We design our node-based deception algorithm relying on safeness for two main reasons*. First, using safeness instead of modularity has the advantage of not requiring complete knowledge of the whole community structure [7]; safeness can be seen as a local (concerning only $\mathcal{C}$'s members) measure. Second, safeness is entirely independent of existing community-centric metrics like modularity. Community safeness is defined starting from node safeness.

Another important aspect of safeness is that it emphasizes that a community is *safer* when hidden. Indeed, as pointed out earlier, exposing relationships between nodes might put them in a vulnerable position, especially if they belong to a disadvantaged group, say a political or a religious minority. Moreover, community detection allows inferring personal data that some users would not otherwise share on a social network. It has been shown, for instance, that user addresses can be re-identified in a Bitcoin transaction network [24]. Note also that safeness, as we shortly show, explicitly handles the trade-off between connectivity (or communication) and security.

Definition 1 (Node Safeness): Let G be a network, $\mathcal{C} \subseteq V$ a community, and $u \in \mathcal{C}$ a member of $\mathcal{C}$. The safeness of u in G is defined as

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

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

Definition 2 (Community Safeness): Given a network $G = (V, E)$ and a community $\mathcal{C} \subseteq V$, the safeness of $\mathcal{C}$ is defined as: $\sigma(\mathcal{C}) = \sum_{u \in \mathcal{C}} \sigma(u, \mathcal{C})/|\mathcal{C}|$

The first part of (2) incentives higher node reachability (i.e., $|V^u(\mathcal{C})|$), while keeping a minimum intracommunity degree (i.e., $|E(u, \mathcal{C})|$). The second part of the equation awards higher intercommunity degrees of u. The values of τ and χ are set by the deceptor to balance the trade-off between reachability and intercommunity edge ratio. Note that including reachability is another advantage of safeness over modularity. In this article, we instantiate ϕ as the *safeness gain* $\Delta_S^\beta = \sigma(\mathcal{C}') - \sigma(\mathcal{C})$ where $\mathcal{C}'$ is the updated target community after applying β node updates. At this point, we need to study the impact of individual *node updates* on safeness.

IV. Node Updates and Safeness

As stated in the problem introduced in Section III-A, we note that the optimization of the safeness gain needs to produce four sets: V^- (node deletions from $\mathcal{C}$), V^+ (node additions), E^+ (edge additions toward nodes in $\mathcal{C}$ and node in other communities), and E^- (edge deletions from the nodes in $\mathcal{C}$). To move forward toward defining a node-centric deception algorithm, we first need to understand the effect of deleting/adding nodes on safeness. The results we present here are used to define the nSAF algorithm described in Section V.

A. Effect of Node Deletions on Safeness

Let $\mathcal{C}$ be the target community that wants to hide from community detection algorithms. Assuming $|\mathcal{C}| > 1$, we have that the deletion of a node $\gamma \in \mathcal{C}$ brings the following safeness gain:

$$\Delta S = \sigma(\mathcal{C}^-) - \sigma(\mathcal{C}) = \underbrace{\sum_{v \in \mathcal{C}^-} \frac{\sigma(v, \mathcal{C}^-)}{|\mathcal{C}^-|}}_{\text{after node deletion}} - \underbrace{\sum_{u \in \mathcal{C}} \frac{\sigma(u, \mathcal{C})}{|\mathcal{C}|}}_{\text{before node deletion}} \tag{3}$$

where $\mathcal{C}^- = \{u | u \in (\mathcal{C} \backslash \{\gamma\})\}$ is the set of the remaining nodes after deleting γ. It is important to note that the safeness of nodes in $\mathcal{C}^-$ (after deletion) may differ from that of nodes in $\mathcal{C}$ (before deletion). This is because deleting γ implies changes in the number of edges of the remaining nodes, which affect their safeness [see (2)]. Now, since $|\mathcal{C}^-| = |\mathcal{C}| - 1$, we can reduce (3) to

$$\Delta S = \frac{1}{|\mathcal{C}|(|\mathcal{C}| - 1)} \left[|\mathcal{C}| \sum_{v \in \mathcal{C}^-} \sigma(v, \mathcal{C}^-) - (|\mathcal{C}| - 1) \sum_{u \in \mathcal{C}} \sigma(u, \mathcal{C}) \right]. \tag{4}$$

Thus, $\Delta S > 0$ if and only if

$$\sum_{v \in \mathcal{C}^-} \sigma(v, \mathcal{C}^-) > \frac{(|\mathcal{C}| - 1)}{|\mathcal{C}|} \sum_{u \in \mathcal{C}} \sigma(u, \mathcal{C}). \tag{5}$$

We can conclude that deleting a node from the target community will result in a safeness gain if and only if the total safeness of the remaining nodes is greater than the threshold defined in (5). The left side of (5) is simply the sum of the remaining nodes' safeness. The best node to delete is the one with the least individual safeness, increasing the remaining nodes' average safeness. We will discuss how this result can be used in the nSAF deception algorithm in Section V.

B. Effect of Node Additions on Safeness

Let us now consider the addition of a new node ζ. We observe that ζ can be a completely new node or a node that previously existed and was deleted and readded as part of the deception strategy. We have that

$$\Delta S = \sigma(C^+) - \sigma(C) = \underbrace{\sum_{v \in C^+} \frac{\sigma(v, C^+)}{|C^+|}}_{\text{after node addition}} - \underbrace{\sum_{u \in C} \frac{\sigma(u, C)}{|C|}}_{\text{before node addition}} \tag{6}$$

where $C^+ = \{u | u \in (C \cup \{\zeta\})\}$ is the community after adding ζ. In this case, it is strategic to add edges from ζ toward nodes outside $\mathcal{C}$ so that ζ will end up in a community $C \neq \mathcal{C}$. Noting that $|C^+| = |C| + 1$, we can reduce (6) to

$$\Delta S = \frac{1}{|C|(|C| + 1)} \times \left[|C| \sum_{v \in C^+} \sigma(v, C^+) - (|C| + 1) \sum_{u \in C} \sigma(u, C) \right]. \tag{7}$$

Thus, $\Delta S > 0$ if and only if

$$\sum_{v \in C^+} \sigma(v, C^+) > \frac{(|C| + 1)}{|C|} \sum_{u \in C} \sigma(u, C). \tag{8}$$

Based on (8), we can say that node addition is not guaranteed to cause a safeness gain. Therefore, the edges of the new node $E(\zeta)$ must be chosen to maximize the left side of (8).

1) Selecting Edges of the New Node: As we saw in Section IV-B, maximizing the target community's safeness involves maximizing its component nodes' safeness. In what follows, we assume a worst case scenario where the target community is already revealed –$\mathcal{C} \in \bar{C}$. In a real-world setting, however, this is not necessarily the most probable situation, but it can provide a good account of the effectiveness of our proposed technique. We need to consider that ζ can end up (after running a detection algorithm $\mathcal{A}_{\text{det}}$) in one of two states—either it is added to $\mathcal{C}$ forming $\mathcal{C}^+$, or it is added to some other community in $C \in \bar{C}$. In the beginning, the new node ζ must have *at least one connection* with one of the nodes in $\mathcal{C}$; without this requirement, ζ would not be able to communicate with any $\mathcal{C}$'s member and will not have any effect on the safeness of the community. We need to quantify the extent of this change precisely. Theorem 3 shows that the safeness change resulting from connecting ζ with a node $u \in \mathcal{C}$ depends on whether this connection is the first made by ζ. Specifically, in the beginning, ζ is an isolated node, but after the first edge addition, it already forms a single connected component with $\mathcal{C}$ (we assume that $\mathcal{C}$ is not disconnected).

Theorem 3: Let $\mathcal{C}^+ = \{\zeta\} \cup \mathcal{C}$, where ζ is a new node that wants to join $\mathcal{C}$, then adding the edge (ζ, u), where $u \in \mathcal{C}$, will always decrease $\sigma(u, \mathcal{C}^+)$.

Proof: First, the safeness of u before edge addition is

$$\sigma(u, \mathcal{C}^+) = \tau \frac{|V^u(\mathcal{C}^+)| - |E(u, \mathcal{C}^+)|}{|\mathcal{C}^+| - 1} + \chi \frac{|\widetilde{E}(u, \mathcal{C}^+)|}{\delta(u)}. \tag{9}$$

Now, we need to consider two cases.

Case 1: if $\mathcal{C}^+$ does not form a single component, i.e., ζ is isolated from $\mathcal{C}$ before edge addition, then $|V^u(\mathcal{C}^+)|$ will

increase by 1. Thus,

$$\sigma(u, \mathcal{C}^+)^* = \tau \frac{\left(|V^u(\mathcal{C}^+)| + 1\right) - \left(|E(u, \mathcal{C}^+)| + 1\right)}{|\mathcal{C}^+| - 1} + \chi \frac{|\widetilde{E}(u, \mathcal{C}^+)|}{\delta(u) + 1}. \quad (10)$$

Moreover, the change in the safeness of u is

$$\Delta\sigma(u, \mathcal{C}^+) = -\chi \frac{\widetilde{E}(u, \mathcal{C}^+)}{\delta(u)^2 + \delta(u)}. \quad (11)$$

Case 2: $\mathcal{C}^+$ already forms a single component before edge addition. Let $\sigma(u, \mathcal{C}^+)^*$ be the safeness of u after adding the edge (ζ, u). Since $\mathcal{C}^+$ forms a single component, adding (ζ, u) will not affect $|V^u(\mathcal{C}^+)|$, and therefore

$$\sigma(u, \mathcal{C}^+)^* = \tau \frac{\left(|V^u(\mathcal{C}^+)|\right) - \left(|E(u, \mathcal{C}^+)| + 1\right)}{|\mathcal{C}^+| - 1} + \chi \frac{|\widetilde{E}(u, \mathcal{C}^+)|}{\delta(u) + 1}. \quad (12)$$

Now, we compute the change in the safeness of u

$$\Delta\sigma(u, \mathcal{C}^+) = \sigma(u, \mathcal{C}^+)^* - \sigma(u, \mathcal{C}^+) = \frac{-\tau}{|\mathcal{C}^+| - 1} - \chi \frac{\widetilde{E}(u, \mathcal{C}^+)}{\delta(u)^2 + \delta(u)}. \quad (13)$$

As shown above, in both cases, the safeness of u will decrease after adding an edge with ζ. However, the magnitude of the decrease will be smaller if (ζ, u) is the first connection between ζ and $\mathcal{C}$. ■

Next, we move to the case where ζ will not explicitly be part of $\mathcal{C}$ while still being part of the *virtual community* $\mathcal{C}$. This can happen because of successive edge operations by ζ, which might render it closer to other communities. Indeed, it is clear that if ζ adds only one edge toward nodes in $\mathcal{C}$ and several edges to nodes outside $\mathcal{C}$, a sensible detection strategy will place ζ in another community due to this weak tie with $\mathcal{C}$.

Theorem 4: Let $\zeta \in C^*$ be an added node, where $C^* \in \bar{C}$. Then, adding the edge (ζ, u), where $u \in \mathcal{C}$, will always increase $\sigma(u, \mathcal{C})$.

Proof: The safeness of u before edge addition is

$$\sigma(u, \mathcal{C}) = \tau \frac{|V^u(\mathcal{C})| - |E(u, \mathcal{C})|}{|\mathcal{C}| - 1} + \chi \frac{|\widetilde{E}(u, \mathcal{C})|}{\delta(u)}. \quad (14)$$

Noting that ζ is not part of $\mathcal{C}$, let the safeness of u after adding the edge (ζ, u) be expressed as $\sigma(u, \mathcal{C})^*$, then

$$\sigma(u, \mathcal{C})^* = \tau \frac{|V^u(\mathcal{C})| - |E(u, \mathcal{C})|}{|\mathcal{C}| - 1} + \chi \frac{|\widetilde{E}(u, \mathcal{C})| + 1}{\delta(u) + 1}. \quad (15)$$

Then we compute the change in safeness

$$\Delta\sigma(u, \mathcal{C}) = \sigma(u, \mathcal{C})^* - \sigma(u, \mathcal{C}) = \chi \frac{\delta(u) - \widetilde{E}(u, \mathcal{C})}{\delta(u)^2 + \delta(u)}. \quad (16)$$

Since $\mathcal{C}$ forms a connected component, $\delta(u) > \widetilde{E}(u, \mathcal{C})$, and therefore, we conclude that $\Delta\sigma(u, \mathcal{C}) > 0$. ■

V. Node-Based Deception via Safeness

We now outline an algorithm to tackle community deception based on node updates and safeness. We have discussed that adding a new node ζ to the network is trickier than deleting a node. Indeed, *when adding a new node, we have to strategically add new edges to promote our primary goal, namely, hiding* $\mathcal{C}$. Moreover, the overall deception strategy may need the addition of fake nodes. The node-based deceptor called nSAF is summarized in Algorithm 1. The algorithm at each iteration starts by computing the possible safeness gain for adding a new node ζ. Specifically, in line 3, the algorithm according to Theorem 3 (case 1), which states that the addition of a new edge with a node in $\mathcal{C}$ decreases safeness, tries to find the node $n_p \in \mathcal{C}$ for which the ration external-internal edges is maximum. Finding the node n_p allows establishing an edge that enables a communication channel between the new node ζ and nodes in the target community. Note that if one establishes more edges, one would need the result in Theorem 3 (case 2).

As described in Section III-A, new nodes can be either natural or fake (bots). The following example best explains the difference: suppose $\mathcal{C}$ is a group of students using a social network. If the new member is a natural one, then it is a real student that shares a genuine relationship with group members, and its profile would naturally be similar to other students in $\mathcal{C}$. On the other hand, $\mathcal{C}$ may choose to design a fake node that does not belong to students but to a different group, say, employees. We make this distinction because in the case of a natural node, $\mathcal{A}_{\text{det}}$ is likely to attach ζ to $\mathcal{C}$, and Theorem 3 applies. This means that adding new edges between ζ and other target community members becomes inadvisable. However, fake nodes do not suffer from this problem because they would not be assigned by $\mathcal{A}_{\text{det}}$ to $\mathcal{C}$ since they do not resemble the rest of $\mathcal{C}$. Thus, Theorem 4 applies, and adding new edges connecting to the rest of $\mathcal{C}$ becomes effective. Although Algorithm 1 does not distinguish between natural and fake nodes explicitly, it is more practically successful in the case of fake nodes since it relies heavily on making external connections (in line 4).

The problem of adding a new node reduces to strategically finding the edges of ζ. This can be achieved by solving the following optimization problem:

$$\begin{aligned} &\text{MAXIMIZE}_{E(\zeta)} \quad \sigma(\mathcal{C}) \\ &\text{where} \quad E(\zeta) \subseteq \{(\zeta, v) : v \in V \wedge v \neq \zeta\} \\ &\qquad |E(\zeta)| \leq \beta. \end{aligned} \quad (17)$$

The brute-force way to approach this problem would be to try every possible edge set $E(\zeta)$. However, this solution suffers from two serious drawbacks. First, it requires global knowledge of all nodes in the network G, which is, at best, rarely available. Second, even if such knowledge was available, connecting to nodes too far from the target community requires more effort in designing a fake profile (i.e., ζ) to increase the chances of outlying communities accepting ζ's proposed connections. Without such effort, connection attempts will probably fail. For instance, if Bob sends a friendship request on Facebook to a person from another country who does not share any mutual friends or interests with him, Bob's request would probably be ignored.

Algorithm 1 The nSAF Deception Algorithm

Input: Network G, Target community $\mathcal{C}$, Budget β
Output: Updated Network G'

1: **do**
2: /* **Node addition** */
3: select(ζ, n_p), $n_p \in$ MAXIMIZE$(\frac{|\tilde{E}(u,\mathcal{C})|}{\delta(u)})$
4: E(ζ)={(ζ, n_p)} ∪`getBExtEdges`$(\mathcal{C} \cup \{\zeta\}, \beta(\zeta)-1)$
5: $\sigma^+ \leftarrow$ `getAddGain`$(\mathcal{C} \cup \{\zeta\})$ /* equation (6) */
6: /* **Node deletion** */
7: **if** $|\mathcal{C}| > 0$ **then**
8: u^-, E_u^- = `getDelNode`$(G, \mathcal{C})$ /* equation (5) */
9: σ^-= `getDelGain`$(\mathcal{C} \backslash \{u^-\})$ /* equation (3) */
10: **else**
11: $\sigma^- = 0$
12: /* **Choosing the kind of update** */
13: **if** $\sigma^- > \sigma^+$ and $\sigma^- > 0$ **then**
14: $G' = (V \backslash \{u^-\}, E \backslash E_u^-)$
15: β=β-$|E_u^-|$
16: **else**
17: **if** $\sigma^+ > 0$ **then**
18: $G' = (V \cup \{\zeta\}, E \cup \{E(\zeta)\})$
19: β=β-$|E(\zeta)|$
20: **while** $\beta > 0$ and $(\sigma^+ > 0$ or $\sigma^- > 0)$

Instead, we can utilize the following two heuristic conditions that are derived from the results of Theorems 3 and 4 (references to algorithm 1 appear in parentheses).

1) Select an edge (ζ, n_p), where $n_p \in \mathcal{C}$, such that n_p has the maximum intercommunity edge ratio (line 3).
2) Construct the remaining $\beta(\zeta)-1$ edges of ζ with nodes from an external community, say C_j, that has the highest number of connections with n_p. (line 4).

Following is a brief outline of the procedures used in Algorithm 1.

`getBExtEdges:` After the first step of the heuristic above, which is needed to maintain a connection between ζ and the rest of $\mathcal{C}$, this procedure performs the second step. This step increases the chances of ζ being allocated to the community C_j by $\mathcal{A}_{\text{det}}$, making Theorem 4 apply. Finally, in line 5, the algorithm computes the safeness gain for the addition according to (6).

`getDelNode`**:** This procedure identifies the best candidate for deletion based on the threshold defined in (5) by looping over all $\mathcal{C}$'s members. The deletion gain is computed in line 9. Then, the algorithm, which adopts a greedy strategy to solve node deception, compares the gain and applies the most convenient deception-wise node update. The output of the algorithm is the updated network G'. Fig. 2 provides more insight into the workflow of nSAF. nSAF receives the network G, the target community $\mathcal{C}$, and the edge budget β as input. It then chooses one addition candidate and one deletion candidate along with their respective edges and computes the safeness gain for each. Finally, it updates the network with the operation yielding the best gain to get a new network G' (G'_+ in the case of an addition and G'_- in the case of a deletion).

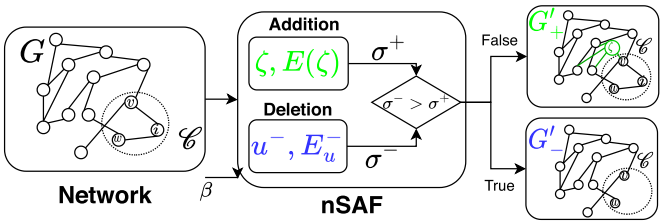

Fig. 2. High-level description of nSAF.

VI. Experimental Evaluation

We conducted an experimental evaluation centered on the following aspects. First, compare our novel node-centric nSAF with edge-centric community deception approaches (**RQ1**, Section VI-B). Second, analyze the effects of deception on the whole community structure (**RQ2**, Section VI-C). Third, give an account of the relative running times of deception and detection algorithms (**RQ3**, Section VI-D). Results reported are the average of ten runs and have been performed on a 2.7 GHz Quad-Core Intel Core i7 with 16GB of memory (95% c.i.). For each of the ten runs performed, a different $\mathcal{C}$ is chosen to have a size close to the average size of the communities the detection algorithm finds to avoid picking too small or too large $\mathcal{C}$s.

A. Experimental Setting

We now outline the experimental setting. We implemented all approaches, but `neur`, in Python[1] using the `igraph`[2] and `cdlib` libraries.[3] For `neur`, we used the existing implementation provided by the authors.[4]

1) Datasets: We considered real-world networks (see Table II) available online.[5] Besides real-world networks, we also considered a variety of networks generated by using the LFR benchmark [12] and varying the number of network nodes (N), the power-law exponent for the degree distribution of the graph (τ_1), the power-law exponent for the community size distribution (τ_2), and the fraction of intercommunity edges incident to each node (μ). The ground truth communities are available for each of the ten generated networks and for `dblp`, `amz`, and `you`. However, we know that using such ground-truth communities may not reflect the actual data-generating process for real networks, which is typically unknown [21]. Some detection algorithms could not complete on the most extensive networks with a timeout of 3 h; we could not apply deception algorithms in these cases.

2) Competitors: We compared nSAF with the following *edge-centric* deception techniques: delete internal connect external (`dice`) [29], modularity minimization (`mmin`) [7], safeness-based deception (`saf`) [7], and neural (`neur`) [18]. We consider an approach (`rnd`), which randomly selects both the type of update and the endpoints of the edge addition/deletion.

3) Detection Algorithms: As adversaries to the deception algorithms, we considered several community detection

[1]https://github.com/giuseppepirro/NdecSAF/
[2]https://igraph.org/
[3]https://cdlib.readthedocs.io
[4]https://github.com/mittalshravika/HideAndSeek-neurAL
[5]http://snap.stanford.edu

TABLE II
REAL NETWORKS CONSIDERED

Network	$\|V\|$	$\|E\|$
Zachary Karate Club (kar)	34	78
Anonymized Facebook sub-net (fb)	6.3K	217K
Dolphin association (dol)	62	159
Co-appearances in Les Miserables (les)	77	154
Madrid bombing terrorist network (mad)	1.3K	400
USA Power Grid (pow)	5K	6.5K
Books about US politics (pol)	105	441
Terrorist network (terr)	2K	1K
Co-authors of diabetes publications (pub)	19K	44K
Networks theory co-authors (nets)	1.5K	5.5K
A subset of DBLP (dblp)	540K	15.2M
Amazon metadata (amz)	548K	925K
YouTube dataset (you)	1M	3M

algorithms available in the cdlib library. We picked detection algorithms based on different optimization techniques.

a) Modularity-based: Leuven [1] (louv): the algorithm optimizes the modularity in two elementary phases: 1) local moving of nodes; and 2) aggregation of the network; Leiden [28] (lei): the Leiden algorithm is an improvement of the Leuven algorithm; Greedy [4] (gred): at every step of the algorithm two communities that contribute maximum positive value to global Modularity are merged; combo [27] (cmb): the fulcrum of the algorithm is the choice of the best recombination of vertices between two communities, as splits and mergers are particular cases of this operation; leading eigenvectors [20] (eig): Newman's leading eigenvector method for detecting community structure based on modularity.

b) Spectral-based: Spectral [10] (spec): it is based on Fielder's vector (obtained from the eigenvector related to the second eigenvalue of the normalized Laplacian) that is leveraged to extract the communities using clustering; knowledge cut [26] (kcut): is designed to provide a unique combination of recursive partitioning and direct k-way methods.

c) Information theory based: InfoMap [25] (inf): the algorithm uses the probability flow of random walks on a network as a proxy for information flows in the whole system, and it decomposes the network into modules by compressing a description of the probability flow.

d) Network structure-based: label propagation [23] (lp): The label propagation algorithm detects communities using network structure; scalable community detection [22] (scd): the procedure optimizes the approximate weighted community clustering metric.

4) Evaluation Methodology: To test deception algorithms, we refer to the methodology introduced in Fionda and Pirrò [7] and measure.

1) *Deception Score* (H_s): This score, which ranges between 0 and 1, combines a measure of reachability preservation among $\mathcal{C}$ members, community spread (in how many communities are $\mathcal{C}$'s members spread), and community hidings ($\mathcal{C}$'s members should be included in the most prominent communities) [7].
2) *Normalized Mutual Information* (NMI) [5]: Given the community structure before deception $\bar{C}$ and after deception $\bar{C}'$, we have $\text{NMI}(\bar{C}, \bar{C}') \in [0, 1]$.
3) *Running Time*: Time to find the updates by a deception strategy and apply them.

Related pieces of work [18] considered community spread and community hiding separately. However, we believe that a good deceptor should simultaneously be evaluated on all the above components. We note no direct relationship between safeness and the deception score. While safeness is the function we optimize to hide a community $\mathcal{C}$, the deception score quantifies the hiding level of a community following an axiomatic definition (i.e., establishing some desiderata on a good hiding of $\mathcal{C}$). Moreover, while the deception score needs information about the overall community structure, safeness does not (see Definition 2).

B. RQ1: Node-Centric Deception

This experiment evaluates node-centric deception in the worst case scenario $\mathcal{C} = C_i$ with $C_i \in \bar{C}$; this is the classical setting followed by the state-of-the-art. To set the budget update for both nSAF and edge-based deception approaches, we looked at the size of the community to be hidden and considered $\beta \in \{0.2 \cdot |\mathcal{C}|, 0.3 \cdot |\mathcal{C}|, 0.5 \cdot |\mathcal{C}|, 0.7 \cdot |\mathcal{C}|\}$. We observe that each node operation involves a certain number of edge operations; for instance, adding a node involves adding a certain number of edges to existing nodes. We count these edge updates as part of the budget for node updates.

Fig. 3 shows the results of applying nSAF in terms of the cumulative deception score H_s (y-axis), where each subplot represents one of the 12 datasets. The x-axis shows increasing budget values. Each bar on the x-axis shows the cumulative H_s computed by adding deception scores achieved against each of the ten detectors, such that the highest possible value would be 10.0 (by getting $H_s = 1.0$ against each detector). Every single color represents one of the detection algorithms. Looking at Fig. 3 on a macro-level, we can observe a steady increase in the cumulative score with higher budget values. This tendency is natural as higher budgets imply more node modifications and stronger deception. A more detailed view shows that, in almost all networks, with budgets 0.5 and 0.7, the deception score against most detectors passes 0.5.

We observe that in the smaller networks, values of the deception score are generally higher, meaning that our strategy is more effective in such networks. This is for two main reasons; first, we noted that target communities have relatively minor dimensions as compared to communities to be hidden in more extensive networks; second, there are fewer possible choices for our algorithm intended as, for instance, communities toward which adding novel edges or nodes in $\mathcal{C}$ to be deleted and re-inserted. We also observe that info seems less sensitive to our deception strategy in the largest networks than other algorithms. However, this fact is not reflected in smaller networks (e.g., nets), where it does not emerge as a less sensitive detection algorithm. We recall here that this experiment was performed in a worst case scenario (starting from $H_s = 0$) and that even in the largest network (you), the

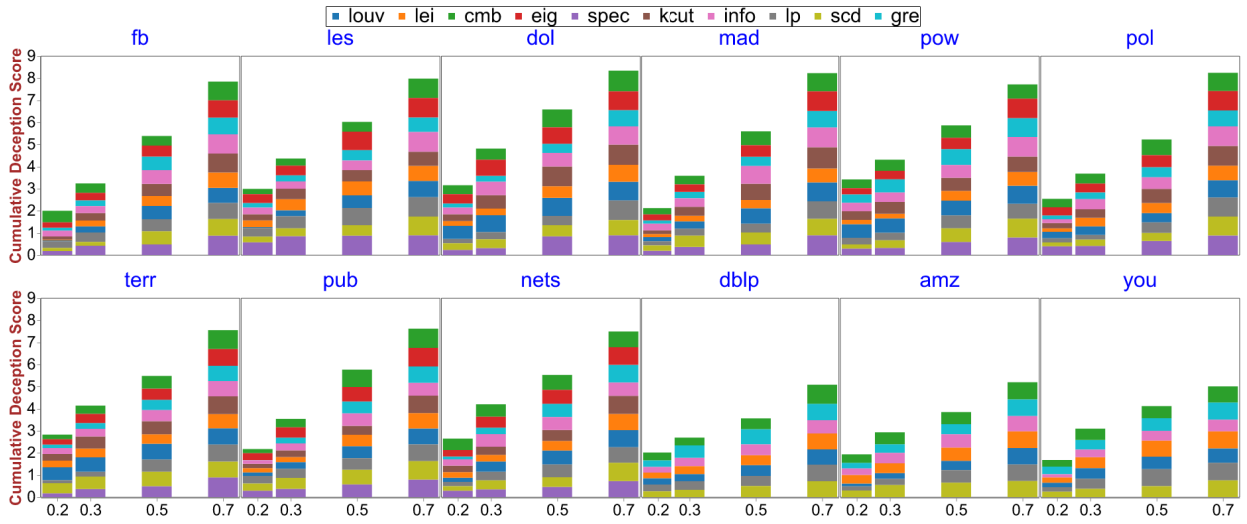

Fig. 3. Evaluation of nSAF. Cumulative min–max normalized H_s score (y-axis) for different budget values (x-axis).

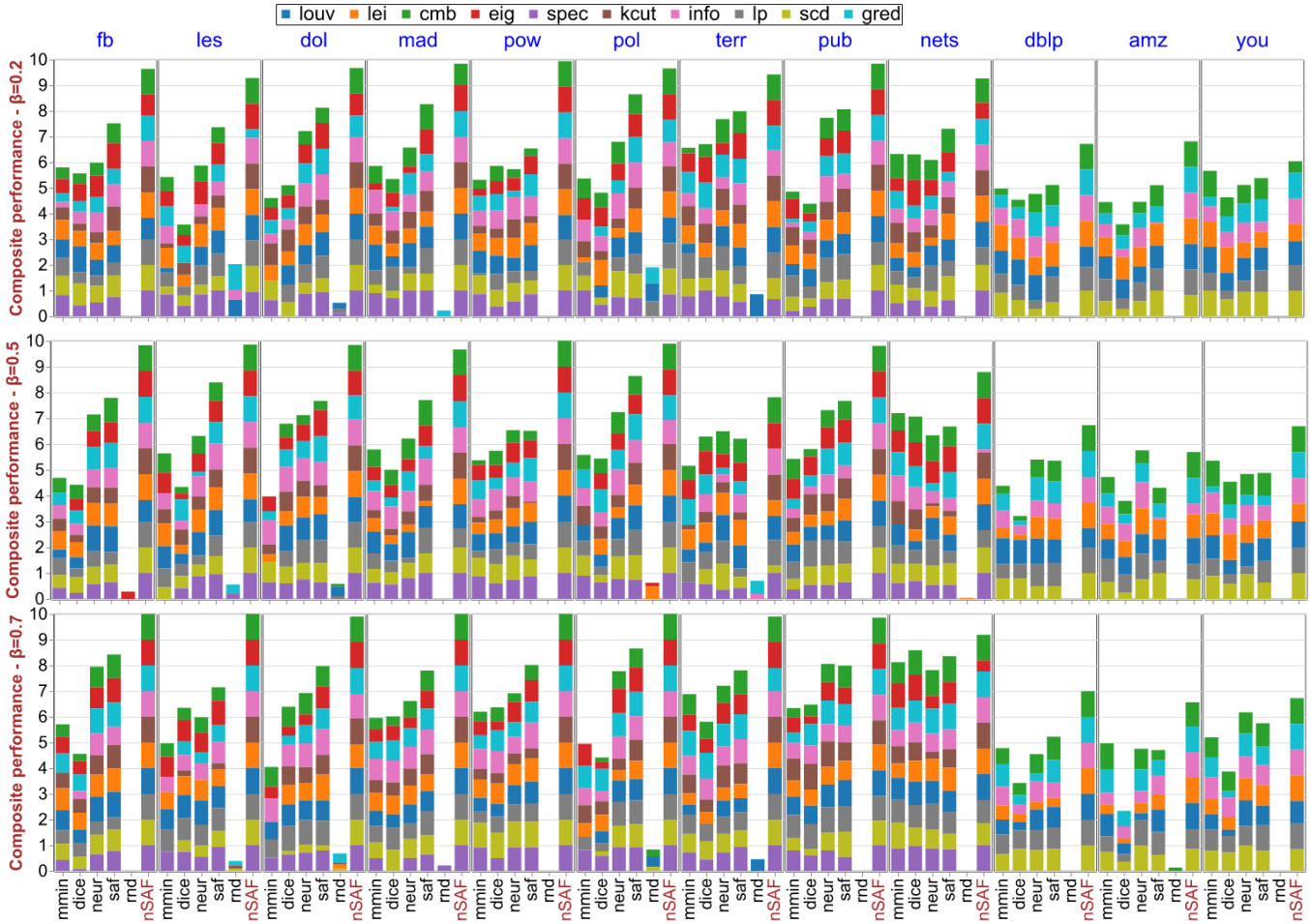

Fig. 4. Comparison between nSAF and edge-based deception approaches for different budget (β) values. The x-axis reports the deception algorithms. The y-axis reports the composite performance intended as the capability to deceive the ten detection algorithms (after min–max normalization of deception scores).

value of H_s was around 0.5 also in the case of `info`, and higher for `lp`.

Now, we compare deception scores H_s of nSAF against four state-of-the-art edge-based community deception algorithms. Again, since we start from a worst case point, where $\mathcal{C}$ is already detected, the deception score's initial value (before deception) is 0. Therefore, any positive increase in its value can be deemed an improvement. Fig. 4 shows three large plots on each horizontal line for three different budget values $\beta = \{0.2, 0.5, 0.7\}$, representing percentages of the target community size. These values are meant to illustrate the effectiveness of more substantial deception and the robustness of various detection algorithms against increasing network changes. In each plot, the y-axis shows the

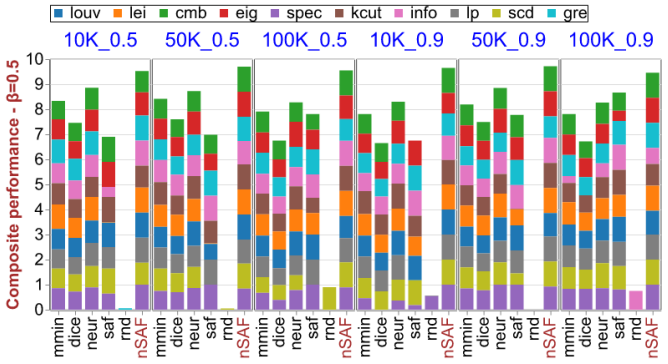

Fig. 5. Comparison between nSAF and edge-based deception approaches on artificially generated networks. The x-axis reports the deception algorithms. The y-axis reports the composite performance intended as the capability to deceive the ten detection algorithms.

composite performance (cumulative deception score after min-max normalization), and the x-axis shows smaller subplots for each of the 12 networks. Each vertical bar shows the performance of one of the six competing deception algorithms (including nSAF), where every color represents the deception score resulting against one of the detection algorithms. Note that for the last three networks, fewer detectors (i.e., colors) are depicted because `spec` detection algorithm, along with `kcut` and `eig`, were not able to complete before a timeout of 3 h. Fig. 4 shows that nSAF outperforms all competitors as reflected by greater composite scores. Nevertheless, the advantage is more apparent in some networks like `pow`. We also note that, except for the largest networks, nSAF achieves $H_s > 0.8$ against virtually all detection algorithms. Additionally, `saf` emerges as our closest competitor, which also uses safeness, and it performs relatively well but fails to scale as much as nSAF on the largest three networks. We note that `mmin` is not as much affected by the change in network size and even outperforms `saf` in some of the largest networks like `amz`. The random approach to deception, `rnd`, almost consistently fails to achieve any meaningful improvement in the deception score, highlighting the necessity of algorithms that strategically choose network modifications.

Fig. 4 can be considered from two other perspectives: the robustness of detectors and the effectiveness of deception algorithms. For the first, a quick scan reveals that `lei` is relatively robust, especially across smaller networks (e.g., `fb`, `les`, `dol`, and `mad`), and less so for some of the larger ones (e.g., `pol`). On the other hand, `gred` seems less affected by the variation in the network size. Larger datasets (i.e., `dblp`, `amz`, and `you`) present a more complex challenge for deception algorithms, and this is reflected by the fact that deception scores are rarely greater than 0.7.

We now move to a similar experiment, but this time using artificial networks (Fig. 5). As before, we plot the composite performance of each deception algorithm against all detectors, using cumulative normalized deception scores. With this experiment, we want to shed light on the performance of deception approaches when imposing network characteristics like the number of intra/intercommunity edges and the total number of nodes. Considering composite performance, we observe that deception score values are generally higher for all deception approaches in these networks. However, the random deceptor `rnd` remains ineffective across all networks. Surprisingly, there seems to be less variance for all deception algorithms. Again, the performance of nSAF is better than other competitors, but we notice some change in their relative rank. Specifically, while in real networks, `saf` was our closest rival, using artificial networks, we see that it was outperformed by `mmin` and `neur`. By digging for further details into the results, we observe that the average H_s value when considering all networks and deception algorithms ranges from ∼0.6 for `dice` to ∼0.85 for nSAF. With the node-based nSAF system, the improvement in performance ranges from ∼10% with respect to`saf` to ∼20% with respect to`dice` on average. Although these results show the superiority of nSAF again, our primary goal with this experiment was to investigate the deception strategies' potential dependencies on the networks' characteristics. We did not observe significant changes in deception when considering the same network regarding the number of nodes and varying the μ parameter controlling the number of intercommunity edges incident to each node.

C. RQ2: Impact of Deception on Communities

In this section, we discuss the impact of deception on the ground-truth communities that are available for both the artificially generated networks and the `dblp`, `amz`, and `you` networks. In particular, via the NMI score, we measure how similar the community structure is after deception to the ideal community structure. However, we know the limitations of working with ground-truth communities [21].

Each of the nine subplots in Fig. 6 represents results using a different network. The y-axis shows the composite performance (computed by min-max NMI normalization) for each competing deception algorithm (x-axis). We notice from the figure that nSAF exhibits relatively more preservation of the community structure after deception. Specifically, running nSAF resulted in normalized NMI $\approx$ 1.0 against all detection algorithms and across all tested networks. Because of time-out, however, we could not show results against three detectors (i.e., `eig`, `spec`, and `kcut`) for the last three real networks. Generally, nSAF keeps ∼20% higher normalized NMI compared to other deception algorithms.

We note that the relative performance of other deception algorithms also seems preserved here. In particular, `saf` remains the closest to nSAF, but on average, it did not produce normalized NMI values higher than ∼0.75–distorting the community structure more than nSAF. On the other hand, random network modification with `rnd` produced the lowest NMI indicating strong disruption of the original community structure. Considering detection algorithms, on the other side, `info` appears to be more sensitive to changes in the community structure as reflected by smaller NMI values on some of the artificial networks. This behavior is consistent with what was observed in the previous experiments where `info` emerged as the most robust detection algorithm. To detail how NMI values are affected by the updates suggested by the deception strategies, we investigated the difference (referred to as Δ) between the number of ground-truth communities and the number of communities after deception. Table III reports the results.

We note that the number of communities after deception generally *increases* when deceiving each of the seven tested detection algorithms. This may indicate that deception

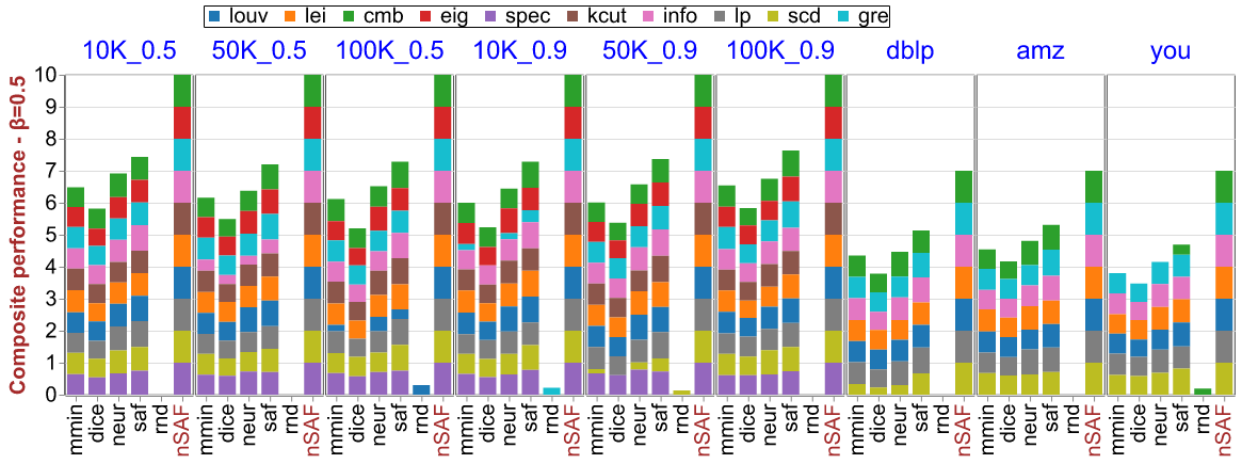

Fig. 6. Impact of Deception on the ground-truth communities. The x-axis reports the deception algorithms. The y-axis reports the composite performance intended as the capability to deceive the ten detection algorithms (min–max NMI normalization).

TABLE III
DIFFERENCE (Δ) IN THE NUMBER OF COMMUNITIES ON LARGE NETWORKS

net	det	mmin	dice	saf	neur	nSAF	rnd
dblp	louv	5	5	23	5	30	3
amz	louv	3	3	18	5	26	5
you	louv	8	3	6	7	18	6
dblp	lei	6	6	34	8	55	3
amz	lei	8	8	14	11	21	8
you	lei	4	6	7	9	16	4
dblp	gre	4	12	34	19	60	8
amz	gre	10	10	19	11	34	8
you	gre	5	10	7	2	11	6
dblp	cmb	8	12	34	4	41	8
amz	cmb	5	15	12	9	32	10
you	cmb	5	5	6	3	17	5
dblp	info	6	6	58	11	38	11
amz	info	6	12	19	7	21	5
you	info	19	19	93	19	22	10
dblp	lp	12	11	21	22	65	95
amz	lp	4	4	15	7	41	4
you	lp	6	4	11	9	18	6
dblp	scd	6	3	14	6	58	3
amz	scd	6	8	11	3	29	6
you	scd	4	8	9	4	15	6

TABLE IV
RUNNING TIME (S) ON LARGE NETWORKS

net	det	mmin	dice	saf	neur	nSAF	rnd	det
dblp	louv	77.49	72.17	102.04	121.46	105.1	44.11	207.24
amz	louv	51.48	75.33	64.72	63.52	69.9	59.46	124.11
you	louv	96.74	130.6	136.05	149.33	141.49	52.88	434.27
dblp	lei	61.77	59.62	72.47	98.6	79.72	35.17	58.31
amz	lei	40.18	57.01	51.42	46.64	57.08	48.95	45.56
you	lei	74.46	115.23	98.31	127.42	105.2	42.89	187.74
dblp	cmb	63.32	50.96	80.63	94.95	83.05	36.93	489.34
amz	cmb	40.7	66.05	56.6	47.27	59.43	50.73	678.87
you	cmb	76.39	103.48	109.2	119.96	122.3	37.08	1865
dblp	info	56.35	55.29	75.53	87.67	80.82	34.29	134.82
amz	info	37.61	64.54	55.95	54.26	62.11	47.17	41.78
you	info	82.19	99.56	96.95	119.96	102.77	41.84	159.45
dblp	lp	64.87	54.57	83.7	102.24	91.23	35.17	1194.56
amz	lp	39.67	55.5	46.89	44.73	48.77	47.17	1436.45
you	lp	73.49	93.03	111.92	116.97	123.11	43.95	2547
dblp	scd	64.87	56.73	76.55	99.81	85.74	36.05	87.67
amz	scd	39.15	66.05	46.89	52.99	49.24	41.81	49.48
you	scd	82.19	100.87	106.48	122.94	111.8	41.31	91.78
dblp	gre	67.97	57.46	85.74	98.6	95.17	31.2	435.73
amz	gre	40.7	63.03	48.19	55.53	50.6	45.98	287.22
you	gre	72.52	94.34	99.68	121.45	107.65	42.36	873.56

strategies can somehow spread nodes (including those belonging to $\mathcal{C}$) in more communities. It is interesting to observe that nSAF brings the greatest increase in the number of communities compared to the baseline algorithms, followed by `saf`. It is noteworthy also that Δ produced by nSAF *decreases* for larger networks, going from `dblp` to `you`. This is apparently because of the larger communities and the need for greater budgets to achieve comparable change.

D. RQ3: Scalability

We tested the scalability regarding running time and reported the community detection time. Results are reported in Table IV for the most extensive networks; for small-scale networks, the times of detection/deception were negligible.

We have the following observations. As the network size increases, detection is much more time-consuming than deception. The main reason is that optimizing a detection function considering all network nodes is intrinsically more involved than focusing on a small subset of nodes represented by $\mathcal{C}$. We note that nSAF requires a running time similar to `saf`, and also to `scd`, the fastest detection algorithm. This is because both nSAF and `saf` adopt safeness as an optimization function, although nSAF centers the deception strategy on nodes. Although simple heuristic methods such as `dice`, and sometimes `mmin`, seem to have relatively lower running times, we think the advantage of nSAF in terms of its notably higher deception scores outweighs the difference in execution time (especially given the scale of the tested network). As a general comment, we observe that the deception time is significantly lower than the detection time, which shows the actual applicability of deception strategies even to large networks. Although the running times observed show the feasibility of deception even in large

networks, running time loses significance in a real world like Twitter and Facebook; adding and deleting nodes and edges requires friending/unfriending or following/unfollowing people, respectively.

VII. Conclusions and Future Work

This article starts with a simple consideration: different actors, such as community members, analyze networks to obtain indirect information. Since community detection is part of any network analysis toolkit, we consider it necessary to think of counter-tools, such as community deception, that can restore people's freedom in a network. Existing approaches have addressed community deception by considering only updates at the edges; this is limiting because nodes can join and leave a community. Therefore, this work provides a more comprehensive view of community deception with two main features. First, our solution relies on safeness, a simple and intuitive measure of the hiding of a community independent of any cluster quality measure. Second, our approach considers the insertion and deletion of nodes, all operations neglected so far. We have shown that all operations can better be linked to hiding $\mathcal{C}$.

However, other strands of research need to be addressed in the future; first, despite the merits of choosing safeness as the deception optimization function, node-centric deception based on *other optimization functions* has to be considered to have a broader perspective on node-centric deception. Moreover, this work assumes a worst case scenario where the deception starts from a completely revealed target community. Introducing algorithms that can *compromise on the worst case assumption* might notably improve the scalability and performance of deception approaches. Our proposed deception algorithm targets only nonoverlapping community detection, leaving room for further research *considering overlapping detection algorithms*. Another open research question is how to explore deception in richer networks such as knowledge graphs and signed networks.

References

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.

[2] J. Chen, Y. Chen, L. Chen, M. Zhao, and Q. Xuan, "Multiscale evolutionary perturbation attack on community detection," *IEEE Trans. Computat. Social Syst.*, vol. 8, no. 1, pp. 62–75, Feb. 2021.

[3] X. Chen, Z. Jiang, H. Li, J. Ma, and P. S. Yu, "Community hiding by link perturbation in social networks," *IEEE Trans. Computat. Social Syst.*, vol. 8, no. 3, pp. 704–715, Jun. 2021.

[4] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 70, no. 6, pp. 66–111, Dec. 2004.

[5] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *J. Stat. Mech., Theory Exp.*, vol. 2005, no. 9, Sep. 2005, Art. no. P09008.

[6] V. Fionda, S. A. Madi, and G. Pirró, "Community deception: From undirected to directed networks," *Social Netw. Anal. Mining*, vol. 12, no. 1, p. 74, Dec. 2022.

[7] V. Fionda and G. Pirró, "Community deception or: How to stop fearing community detection algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 4, pp. 660–673, Apr. 2018.

[8] V. Fionda and G. Pirró, "Community deception in networks: Where we are and where we should go," in *Complex Networks and Their Applications X*. Cham, Switzerland: Springer, 2022, pp. 144–155.

[9] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Phys. Rep.*, vol. 659, pp. 1–44, Nov. 2016.

[10] D. J. Higham, G. Kalna, and M. Kibble, "Spectral clustering and its use in bioinformatics," *J. Comput. Appl. Math.*, vol. 204, no. 1, pp. 25–37, Jul. 2007.

[11] G. King, J. Pan, and M. E. Roberts, "How censorship in China allows government criticism but silences collective expression," *Amer. Political Sci. Rev.*, vol. 107, no. 2, pp. 326–343, May 2013.

[12] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 80, no. 1, Jul. 2009, Art. no. 016118.

[13] D. Liu, G. Yang, Y. Wang, H. Jin, and E. Chen, "How to protect ourselves from overlapping community detection in social networks," *IEEE Trans. Big Data*, vol. 8, no. 4, pp. 894–904, Aug. 2022.

[14] Y. Liu, J. Liu, Z. Zhang, L. Zhu, and A. Li, "REM: From structural entropy to community structure deception," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 12938–12948.

[15] S. A. Madi and G. Pirró, "Influence-based community deception," in *Complex Networks and Their Applications XI*, H. Cherifi, R. N. Mantegna, L. M. Rocha, C. Cherifi, and S. Micciche, Eds. Cham, Switzerland: Springer, 2023, pp. 175–187.

[16] T. Magelinski, M. Bartulovic, and K. M. Carley, "Measuring node contribution to community structure with modularity vitality," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 707–723, Jan. 2021.

[17] I. Manipur, M. Giordano, M. Piccirillo, S. Parashuraman, and L. Maddalena, "Community detection in protein–protein interaction networks and applications," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 20, no. 1, pp. 217–237, Jan. 2023.

[18] S. Mittal, D. Sengupta, and T. Chakraborty, "Hide and seek: Outwitting community detection algorithms," *IEEE Trans. Computat. Social Syst.*, vol. 8, no. 4, pp. 799–808, Aug. 2021.

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

[20] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 74, no. 3, Sep. 2006, Art. no. 036104.

[21] L. Peel, D. B. Larremore, and A. Clauset, "The ground truth about metadata and community detection in networks," *Sci. Adv.*, vol. 3, no. 5, 2017, Art. no. e1602548.

[22] A. Prat-Perez, 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*, 2014, pp. 225–236.

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

[24] C. Remy, B. Rym, and L. Matthieu, "Tracking Bitcoin users activity using community detection on a network of weak signals," in *Complex Networks*. Berlin, Germany: Springer, 2018, pp. 166–177.

[25] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Nat. Acad. Sci. USA*, vol. 105, no. 4, pp. 1118–1123, Jan. 2008.

[26] J. Ruan and W. Zhang, "An efficient spectral algorithm for network community discovery and its applications to biological and social networks," in *Proc. 7th IEEE Int. Conf. Data Mining (ICDM)*, Oct. 2007, pp. 643–648.

[27] S. Sobolevsky, R. Campari, A. Belyi, and C. Ratti, "General optimization technique for high-quality community detection in complex networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 90, no. 1, Jul. 2014, Art. no. 012811.

[28] V. A. Traag, L. Waltman, and N. J. van Eck, "From Louvain to Leiden: Guaranteeing well-connected communities," *Sci. Rep.*, vol. 9, no. 1, pp. 1–12, Mar. 2019.

[29] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan, "Hiding individuals and communities in a social network," *Nature Hum. Behav.*, vol. 2, no. 2, pp. 139–147, Jan. 2018.

[30] X. Zhao and J. Tang, "Crime in urban areas: A data mining perspective," *ACM SIGKDD Explor. Newslett.*, vol. 20, no. 1, pp. 1–12, May 2018.