Are Communities As Strong As We Think?

Md Abdul Alim, Alan Kuhnle, and My T. Thai
Department of Computer and Information Science and Engineering, University of Florida
Gainesville, FL 32611
Email: {alim,kuhnle,mythai}@cise.ufl.edu

***Abstract*—Many complex systems, from World Wide Web and online social networks to mobile networks, exhibit community structure in which nodes can be grouped into densely interconnected communities. This special structure has been exploited extensively to design better solutions for many operations and applications such as routing in wireless networks, worm containment and interest prediction in social networks. The outcome of these solutions are sensitive to the network structures, which raises an important question: can communities be broken easily in a network? To answer this question, we introduce a density–based problem formulation for analyzing the vulnerability of communities. Our approach includes the NP-completeness and a $O(\log k)$ approximation algorithm for solving the problem where k is the number of communities to be broken. Additionally, we analyze the vulnerability of communities in the context of arbitrary community detection algorithms. The empirical results show that communities are vulnerable to edge removal and in some cases the removal of a small fraction of edges can break the community structure.**

I. Introduction

Complex networks in general exhibit the property of having community structure in which nodes can be grouped into densely interconnected communities. Understanding the behaviors and characteristics of communities is of great advantage. It not only provides helpful information in developing more social-aware strategies for social network problems but also promises a wide range of applications enabled by mobile networking, such as routing in Delay Tolerant Networks (DTNs), worm containment in cellular networks, interest prediction in Online Social Networks (OSNs). Furthermore, communities reveal the core network components together with their mutual interactions, thereby representing the entire network as a compact and more descriptive level. Understanding the community properties can help assess the network vulnerability since changes or failures occurred in one community can have a profound impact which can consequently lead to the transformation of other communities.

Due to the high interaction within a community, we intuitively assume that communities are hard to break; therefore, community-based solutions are very robust. Let us take a community-aware routing protocol in DTNs as an example. In this approach, a group or community in DTNs can be visualized as a group of frequently interacting wireless devices with less connectivity to other groups. Devices in the same community have higher chances to encounter each other to transfer carried messages. Therefore, the knowledge of the community structure could help the routing protocols to wisely choose better forwarding relays for any specific destination, and hence, could significantly improve the chance of message delivery. These approaches have been shown to be very efficient and are among the best methods in DTNs [11], [9]. However, the success of the forwarding clearly depends to a great extent on the internal structure of communities. The non-participation of only some important devices is significant enough to degrade the entire network's performance. Removal of certain edges can lead to unstable behavior of the whole routing process. This raises a question: Are communities really as hard to be broken as believed, even to intentional attacks?

As the first study on this research direction, in this paper, we take the first step on assessing community strength with respect to the removal of edges. The removal of edges can be interpreted as the failures in communication links in wireless networks or DTNs due to energy constraint or the moving of wireless devices. The removal of edges can be also done via unfriending in OSNs. More specifically, in this paper, we choose several combination of different types and sizes of communities and attempt to break them. Clearly, if the number of edges removed is significantly less than the total number of edges in communities, we can say that it is easy to break the communities. Otherwise, we conclude that the communities are very strong.

Unfortunately, identifying these critical edges is very challenging due to several factors: 1) Communities behave very differently based on the location of edge removal. They can either stay intact if the removal edge is less important, or can be broken down into smaller subcommunities which can further be merged to other communities. 2) There is no universally agreed definition of community and there is a vast amount of community detection algorithms in the literature [7], it forces us to define a general method to assess the broken communities for an arbitrary community detection algorithm. And 3) the networks are in large-scale, thus the devised algorithms identifying these critical edges must be scalable.

Aside from the community strength assessment, identifying critical edge provides insights into other applications – for example, limiting misinformation in OSNs. Popularity of OSNs has risen in recent years and these platforms have also become major news sources for many people. In order to control misinformation from being widely propagated, it is important to isolate certain unwanted source users. Breaking a community down into sub-communities by removing the critical edges could enforce such isolation.

The main contributions of this paper are as follows:

- We define the framework for community structure fragility. At first we introduce the density based broken community (DBC) problem for breaking k communities with the minimum number of edge removals and analyze its complexity. We then provide an approximation algorithm with theoretical performance guarantee for the DBC problem.

- To analyze the vulnerability of the community structures in a broader sense, we extend the problem formulation to communities produced from an *arbitrary* community detection algorithm. We offer an efficient heuristic to break the communities and identify the set of critical edges.
- We conduct extensive experiments with different parameters to mine interesting observations about the behavior of broken communities after edge removal. The results show that only a small percentage of edges are enough for breaking the community structure. And thus, the communities are not as strong as we think.

The rest of the paper is organized as follows. Section II introduces the network models and problem definition for density based analysis, along with the NP-completeness proof and the approximation algorithm. In Section III, we consider the general problem for breaking communities along with an efficient heuristic based on a balanced cut method. Extensive experimental results on various datasets are presented in section IV. Section V summarizes the related work and section VI finally concludes the paper.

II. Density-based analysis

A. Network Model and Problem Definition

In this paper a network is represented by a graph $G = (V, E)$ where V is the set of n nodes and E is the set of m edges. A node u in G represents a user while an edge (u, v) represents the interaction between the users u and v in the network. For a community $C \subseteq V$, let m_C and n_C be the number of internal edges and the number of nodes in C, respectively. Let C^{in} denote the set of edges having both endpoints in C. We have used the terms vertex and node interchangeably throughout this work.

There are several quantitative measures to identify communities in a network such as maximizing the modularity based functions [7] and density based functions [8]. In this section, we first consider the density function and discuss other community detection measures later in section III.

The density based function can be defined as $\Psi(C) = \frac{|C^{in}|}{\binom{|C|}{2}}$ to identify a set C of nodes as a community [8]. The more C approaches a clique of its size, the higher its density value $\Psi(C)$.

The threshold on the internal density that suffices for C to be a local community is given by

$$\tau(C) = \frac{\sigma(C)}{\binom{|C|}{2}} \text{ where } \sigma(C) = \binom{|C|}{2}^{1-\frac{1}{\binom{|C|}{2}}} \tag{1}$$

Thus a subgraph induced by C is a local community iff $\Psi(C) \geq \tau(C)$ or equivalently $|C^{in}| \geq \sigma(C)$.

As can be seen, this density function particularly has the advantage of dealing with the candidate group only, not requiring any predefined threshold nor user defined parameter. However, we discuss other community detection measures later in the general framework section. Besides, $\sigma(C)$ is an increasing function which approaches C's full number of connections, i.e., the number of edges in a clique of size $|C|$. Hence, $\sigma(C)$ is a powerful tool for detecting local communities, i.e., densely connected parts of a network.

Based on the definition of the density function, a community C is *broken* if, by removing a set of edges S from E, the density of C falls below $\tau(C)$. Therefore, let k_i denote the number of edges required to be removed from community C_i to make $\Psi(C_i \backslash S_i) < \tau(C_i \backslash S_i)$ where S_i is the set of removed edges in community C_i, then k_i is defined as

$$k_i = \min\{t | (\frac{2(m_{C_i} - t)}{n_{C_i}(n_{C_i} - 1)}) < \tau(C_i)\} \tag{2}$$

The density-based breaking of communities (DBC) problem is defined as follows:

Definition. (DBC) Given an undirected graph $G = (V, E)$, and a set $\mathscr{C}$ of k communities, find a subset $S \subset E$ of minimum cardinality such that removing S from the graph breaks every community in $\mathscr{C}$.

B. Complexity of DBC

Theorem 1. *The DBC problem is NP-complete.*

Proof: The decision version of DBC is defined as follows. Given $(G, \mathscr{C}, l)$, where $G = (V, E)$ is a graph, $\mathscr{C}$ is a set of communities of G, and l is a positive integer, determine whether there exists a set $S \subset E$ such that in $G' = (V, E \backslash S)$, every community in $\mathscr{C}$ is broken, and $|S| \leq l$.

Given a set S of edges, one can efficiently check whether $|S| \leq l$ and whether all communities in $\mathscr{C}$ are broken. Thus DBC is in NP.

To show the NP-hardness, we reduce from the vertex cover problem, defined as follows. Given (G, l), where $G = (V, E)$ is a graph, and l is a positive integer, a vertex cover is a set $A \subset V$ such that for all $e = (u, v) \in E$, $u \in A$ or $v \in A$. The problem is to determine whether a vertex cover A exists with $|A| \leq l$.

First, we need to define the identification of vertices in a graph.

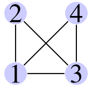
Definition (Vertex identification)**.** Let $H = (V, E)$ be a graph. Let $A = \{u_i : i \in I\}$ be a collection of vertices. Identification of the vertices A is defined to be the following operation.

Let $H' = (V', E')$ be the induced subgraph of H after removing vertices $\{u_i : i \in I\}$. Let u be a new vertex. Then

$$V^* = V' \cup \{u\}$$
$$E^* = E' \cup \{(u, w) : (u_i, w) \in E, w \in V'\}$$

and $H^* = (V^*, E^*)$ is the result of the operation.

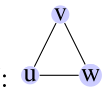
Construction. Let C be the following community with 4 vertices and 5 edges:

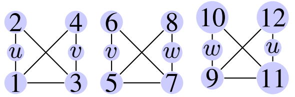


Let (G, l) be an instance of vertex cover. For each edge $e = (u, v) \in G$, we create a copy C_e of C. We associate edge $(1, 2)$ in C_e with u, and edge $(3, 4)$ with v.

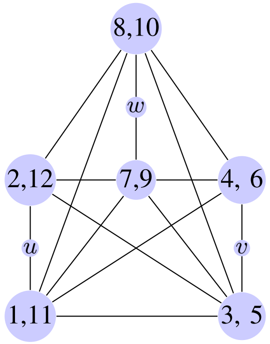
Now form the graph $\hat{G} = \dot{\bigcup}_{e \in E} C_e$, the disjoint union of the C_e. Finally, for each vertex v in G, identify in $\hat{G}$ all incident vertices to the edges to which v is associated. The resulting graph will be called G^*. Together with the collection $\mathscr{C} = \{C_e : e \in E\}$, $(G^*, \mathscr{C}, l)$ form an instance of the decision version of DBC, where we consider C_e in G^* to be the set of vertices of C_e in $\hat{G}$ after identification.

Example. To illustrate the above construction, we will consider an instance of vertex cover (G, l) where G is a triangle, and show $\hat{G}$ and finally G^*.

G: 

$\rightarrow \hat{G}$: 

. Then, vertices $\{1, 11\}$, $\{2, 12\}$ are identified corresponding to the edges associated with u, and likewise for the other vertices in G, and we have G^*:



Given instance (G, l) of vertex cover, it remains to be shown that a solution for $(G^*, \mathscr{C}, l)$ yields a solution for (G, l). Each community C_e meets the density requirement to be a community by a single edge. Since none of the edges in C_e other than $(1, 2)$ and $(3, 4)$ are shared with any other community in $\mathscr{C}$, we can assume that only $(1, 2)$ or $(3, 4)$ (after identification) is removed from any given community. Thus, each edge that is a candidate for removal corresponds to a unique vertex in G.

Thus, given a solution B of at most l edges whose removal breaks $\mathscr{C}$, we get a set A of vertices corresponding to the edges in B. This set A is a vertex cover of G. To see this let $e \in E$. Then C_e is broken by removing B. Thus, one of the edges corresponding to the vertices of e must be in B; hence at least one vertex of e is in A.

By similar argument, a feasible vertex cover for (G, l) gives rise to a feasible solution of $(G^*, \mathscr{C}, l)$. ■

C. Solutions to DBC

In this section, we provide an approximation algorithm for DBC with a theoretical performance guarantee. In doing so, we first reduce DBC to the set multicover problem, in a way that preserves the approximation ratio for set multicover. We then apply solutions of set multicover to our problem. The challenging part of this approach is to reduce a problem to another one while preserving the ratio.

Definition (Approximation ratio preserving reduction)**.** Let Π_1 and Π_2 be minimization problems.

Let f be a polynomial-time algorithm such that if I_1 is an instance of Π_1, $I_2 = f(I_1)$ is an instance of Π_2 with $OPT(I_2)) \leq OPT(I_1)$; that is, the value of the optimal solution to I_2 is at most the value of the optimal solution to I_1.

Let g be a polynomial time algorithm, such that if t is a solution of $I_2 = f(I_1)$, $s = g(I_1, t)$ is a solution of I_1 such that the objective function value of s is not more than the objective function value of t; that is, $\text{obj}_{\Pi_1}(I_1, s) \leq \text{obj}_{\Pi_2}(I_2, t)$.

Then, by use of f and g, an α-approximation for Π_2 yields an α-approximation for Π_1.

Consider the problem

Definition (Set multicover)**.**

$$\begin{aligned} &\text{minimize } x \\ &\text{subject to } Ax \geq b, \\ &\qquad\qquad 0 \leq x \leq u, \ (x \text{ integer}) \end{aligned}$$

where A is n by m matrix (a_{ij}), $a_{ij} \in \{0, 1\}$, $b_i \in \mathbb{N}$ for $i \in \{1, \ldots, n\}$, $u_i \in \mathbb{N}, i \in \{1, \ldots, m\}$.

We have defined set multicover as an integer program, for convenience, but one may think of row i of A as giving the subsets to which element i belongs, b_i as the number of times element i is required to be covered, x_i would correspond to the number of times set i could be picked, bounded above by u_i.

Next, we will define an approximation ratio preserving reduction from DBC to set multicover. Let I_1 be an instance of DBC, consisting of a graph $G = (V, E)$ and a set of communities $\mathscr{C}$ to be broken. Suppose each $C_i \in \mathscr{C}$ to require k_i edges to be removed.

Now, for the set multicover instance, instance I_2 will be defined in the following way. Define the set of elements to be covered to be $\mathscr{C}$, with $b_i = k_i$. For each $e \in E$, define $A_e := \{C \in \mathscr{C} : e \in C\}$. These sets will form the collection of subsets of $\mathscr{C}$ from which we choose the multicover. Finally, define u_e, the maximum times A_e can be chosen, to be $|\{f \in E : A_f = A_e\}|$.

Thus, I_2 is a valid instance of set multicover. Now, any feasible solution s of I_1 corresponds in a natural way to a feasible solution t of I_2 of equal cost. List the edges removed in s: $e_1, e_2, \ldots, e_k$. For each edge e_i, add one to the number of times A_{e_i} is chosen. This procedure clearly results in a feasible solution t of I_2 of equal cost to s. Thence, $OPT(I_2) \leq OPT(I_1)$.

Now, let t be a feasible solution of I_2. It consists of a collection $\{(A_e, x_e)\}$ of subsets of $\mathscr{C}$ together with the number of times each subset is chosen. To construct s: for each subset A_e, pick x_e edges f such that $A_f = A_e$. This is possible since $x_e \leq u_e$, where u_e is the number of edges satisfying this condition. The cost of s is equal to the cost of t. Hence, we have an approximation-preserving reduction.

By [6], set multicover as defined above has a $\log k$-approximation algorithm, where k is the number of elements to be covered. If we combine this algorithm with the above reduction, we have a $\log k$-approximation algorithm for DBC, where k is the number of communities to be broken.

We present the approximation algorithm in Alg. 1 labeled CVA (Community Vulnerability Assessment). The gain function $f(e)$ indicates the number of unbroken communities $L(e)$ that the edge e belongs to. In each iteration we pick the edge with highest gain until all the communities in $\mathscr{C}$ are broken. The $DeletionVector\ D$ contains the number of edges necessary to be removed for each community in order to break it. This vector D is updated each time an edge is removed from a community. Once all the necessary edges to break a community C_i have been removed, i.e. when D_i becomes 0, the community is broken and the gain function $f(e)$ is updated.

Algorithm 1: CVA: An approximation algorithm for finding the critical edges

```
Data: Network $G = (V, E)$, $DeletionVector\ D$,
      $\mathscr{C}, |\mathscr{C}| = k$
Result: A set $S \subseteq E$ edges
$S \leftarrow \emptyset$;
$C \leftarrow \emptyset$;
for each edge $e \in E$ do
    compute the gain $f_X(e)$;
while $|C| < k$ do
    $e' \leftarrow \underset{e \in E}{\text{argmax}}\{f(e)\}$;
    In case of a tie, choose randomly;
    $S \leftarrow S \cup \{e'\}$;
    for $l = 1$ to $k$ do
        if $C_l \notin C$ then
            if $e' \in C_l$ then
                $D_l \leftarrow D_l - 1$;
                if $D_l \leq 0$ then
                    $C \leftarrow C \cup \{C_l\}$;
                    $f(e) = f(e) - 1$ for all $e \in C_l$;
return $S$;
```

III. A GENERAL FRAMEWORK

We now discuss the breaking community problem in the context of a general community detection algorithm. There are a plethora of community detection algorithms with different objective functions. Thus, we define what it means to break a community for an arbitrary community detection algorithm as follows.

Definition. (Broken Community) Consider a community detection algorithm $\mathscr{A}$, which produces a collection $\mathscr{C}$ of communities on graph G (written $\mathscr{C} = \mathscr{A}(G)$). Let G' be a new graph after removal of a set of edges, and let $\mathscr{C}' = \mathscr{A}(G')$. Let $\gamma \in (0, 1)$. A community $C \in \mathscr{C}$ is said to be *broken* in graph G' if there does not exist a community $C' \in \mathscr{C}'$ satisfying

(*i*) $C' \subset C$, and

(*ii*)

$$\frac{|C'|}{|C|} > \gamma.$$

We introduce the strictness threshold γ which defines how much similarity the two structures have in terms of number of common nodes once the community is broken after edge removal. The larger this threshold the less strict the requirement is and vice versa.

Accordingly, **Broken Community Assessment** *BCA* problem is formulated as follows:

Definition. (BCA) Given a network represented by a graph $G = (V, E)$, a specific set $\mathscr{C}$ of k communities, BCA seeks for a minimum cardinality subset $S \subseteq E$ such that removal of S from G breaks every community in $\mathscr{C}$.

A. Solution to BCA

Let $\epsilon > 0$. Define a c-way ϵ-balanced partition of a graph to be a partition with c components, such that for each component A, $|A| < \frac{(1+\epsilon)n}{c}$ [12].

Lemma 1. *Partitioning a community C into at least c ϵ-balanced subparts, where $\gamma c \geq 1 + \epsilon$ makes it broken.*

Proof: After a balanced paritioning of C into c subparts, each partition has less than $(1 + \epsilon)\frac{n}{c}$ vertices, where $n = |C|$. Now, let $\gamma c \geq 1 + \epsilon$, and A be a component. Then,

$$|A| < \frac{(1+\epsilon)n}{c} = \frac{(1+\epsilon)n}{\frac{1}{\gamma}(\gamma c)} \leq \frac{(1+\epsilon)\gamma n}{(1+\epsilon)} = \gamma n$$

Finally, any community C' detected within C must lie in one of the components, A; so $|C'| < \gamma|C|$, and the community C is broken.

■

We devise Alg. 2 for solving the BCA problem based on Lemma 1. In order to find a solution, c should satisfy the condition $\gamma c \geq 1 + \epsilon$. We partition each community into c-balanced components. The proposed Critical Community Fragility (CCF) algorithm follows.

Algorithm 2: CCF: A heuristic algorithm for breaking communities

```
Data: Network $G = (V, E)$, $k$ Communities $\mathscr{C}$,
      strictness threshold $\gamma$
Result: A set $S \subseteq E$ of edges
$S \leftarrow \phi$;
$c \leftarrow z : z$ is least integer satisfying $z\gamma \geq 1 + \epsilon$;
for each community $C_i \in \mathscr{C}$ do
    compute the $c$-way balanced partitioning [12];
    $Cut_i$ = set of edges to cut $C_i$ into $c$ parts;
    $S \leftarrow S \cup Cut_i$;
return $S$;
```

For each of the target k communities, Alg. 2 at first finds out the number of parts it needs to be partitioned for breaking that community as per the general definition III. Each of the target communities are then divided into c parts by balanced partitioning algorithm as proposed in [12]. The edges that lie in between different parts are subsequently removed to ensure that the community is broken.

IV. Experimental Evaluation

Our goal in this section is: 1) Evaluate the performance of our proposed algorithm CVA by comparing it to the optimal solution, and 2) Assess the strength of a community.

A. Data Set

Set up: We use data sets from well-known social, collaboration and communication networks which exhibit inherent community structure in their organization. The Facebook data [23] we are using consists of the social network interactions between Rice University graduate students and contains strongly connected components. The Arxiv Condensed Matter Physics collaboration network is obtained from the e-print database [4] and covers scientific collaborations between authors who have submitted papers to Condensed Matter category. If an author i co-authored a paper with author j, the graph contains an undirected edge from i to j. If the paper is co-authored by k authors this generates a completely connected (sub)graph on k nodes. We have further considered Enron email [16] communication network dataset. A summary of the data sets are given in Table I.

TABLE I: Experimental datasets

Data Set	Node Count	Edge Count
Facebook [23]	4039	88234
Arxiv [4]	23133	93439
Enron [16]	36692	183831

B. Performance evaluation of CVA

To test the effect of DBC on different communities, we compare the result of CVA with the outcome of optimal Integer Programming (IP) solution. We have chosen the k largest communities based on the node numbers. For each k, a minimum number of edges are chosen by Alg. 1 and removed from the network. Total set of edges that are required to break all these k communities according to the definition in Eq. 1 is then plotted to compare the performance with that of the optimal one. All tests are averaged on 500 runs for consistency.

1) IP Formulation: We formulate the DBC problem as an IP problem so that we can compare it with the performance of CVA. This IP will be solved using the CPLEX package [22].

Let the variable z_i represent each edge $e_i \in E$:

$$z_i = \begin{cases} 1, & \text{if } e_i \text{ is selected for removal.} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

For each $C_j \in \mathscr{C} = \{C_1, \ldots, C_k\}$, k_j be the number of edges required to break C_j as defined in Eq. 2. Then we have the following IP:

$$\begin{aligned} \text{minimize} & \sum_{i=1}^{m} z_i \\ \text{subject to} & \sum_{e_i \in C_j} z_i \geq k_j, && \forall C_j \in \mathscr{C}, \\ & z_i \in \{0, 1\}, && \forall i \in \{1, ..., m\} \end{aligned}$$

Fig. 1 depicts the number of edges required to be removed for breaking a total of 100 communities. As can be seen, the comparative performance of CVA is very much close to the optimal one for all the data sets except for a negligible deviation in Arxiv data set as Fig. 1(b) depicts. And thus we can conclude that CVA performs very well.

C. Performance evaluation for generalized framework

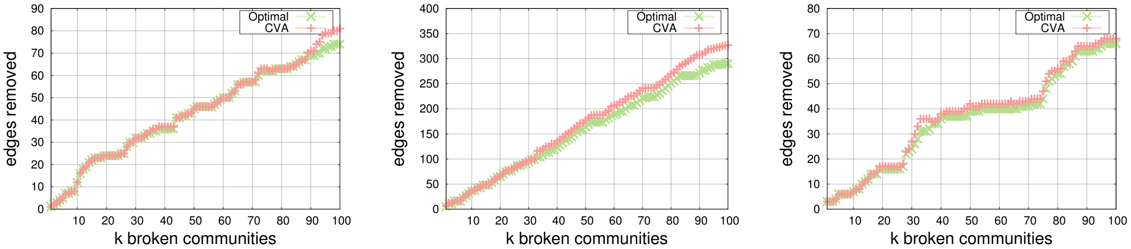
We provide the comparative analysis of the behavior of different networks under two community detection algorithms. For this purpose, we use *Blondel* [2] and *Oslom* [15]. The first one is a modularity based community detection scheme which has been shown to produce very good modular components in timely manner [14]. On the other hand, the latter one is based on statistical properties of the graph which allows overlapping communities. The characteristics of different networks detected by these two community detection algorithms is shown in Table II.

TABLE II: Network Communities

Data Set	Community Count in Blondel	Community Count in Oslom
Facebook	17	118
Arxiv	620	1764
Enron	1265	1374

As a first approach to observe how communities behave under sustained edge removal, we target k large communities with CCF. To this end, we choose two different values of strictness threshold γ, 0.5 and 0.3 all of which follow $\gamma c \geq (1 + \epsilon)$. For all of the experiments, we have considered $\epsilon = 0.03$ for the balanced partitioning. The threshold 0.5 is less strict than 0.3 in the sense that it allows more nodes to be retained even after breaking the community. We also show the behavior of CCF for k randomly selected communities and k smallest communities. For Facebook network with Blondel community detection algorithm, we try to break all 17 communities detected and for all other cases we take 30 communities. The results that we plot are averaged over 100 runs to get rid of inconsistencies as much as possible.

Fig. 2 shows the performance of different types of communities obtained through different community detection algorithms for different strictness threshold (γ) as we remove edges using CCF. For the first two columns, we are considering the k largest communities which were chosen based on their respective number of nodes. From Fig. 2 first column, it is clearly evident that only a small fraction of edge removal causes the communities to be broken for $\gamma = 0.5$. On an average a maximum of 20% edge removal is enough to break



(a) Facebook

(b) Arxiv

(c) Enron

Fig. 1: Density based broken community analysis for k largest community

all 17 communities in Facebook network detected through Blondel as can be seen in Fig. 2(a). In case of Oslom, it takes a large number of edges (a litte more than 40% on an average) which implies larger communities detected by Oslom are more strongly connected internally.

For Arxiv network, the number of edges required to break 30 large communities is only 4% for Blondel community detection algorithm and 15% for Oslom on an average as shown in Fig. 2(e). For Arxiv with the same γ value and equal number of communities, Oslom requires comparatively small number of edges to be removed than in the case of Facebook. It implies members in communities for this particular Facebook network are densely connected internally compared to Arxiv network and as a result it was easier to break Arxiv communities with small number of edges. The same observation is applicable for Enron network as portrayed in Fig. 2(i).

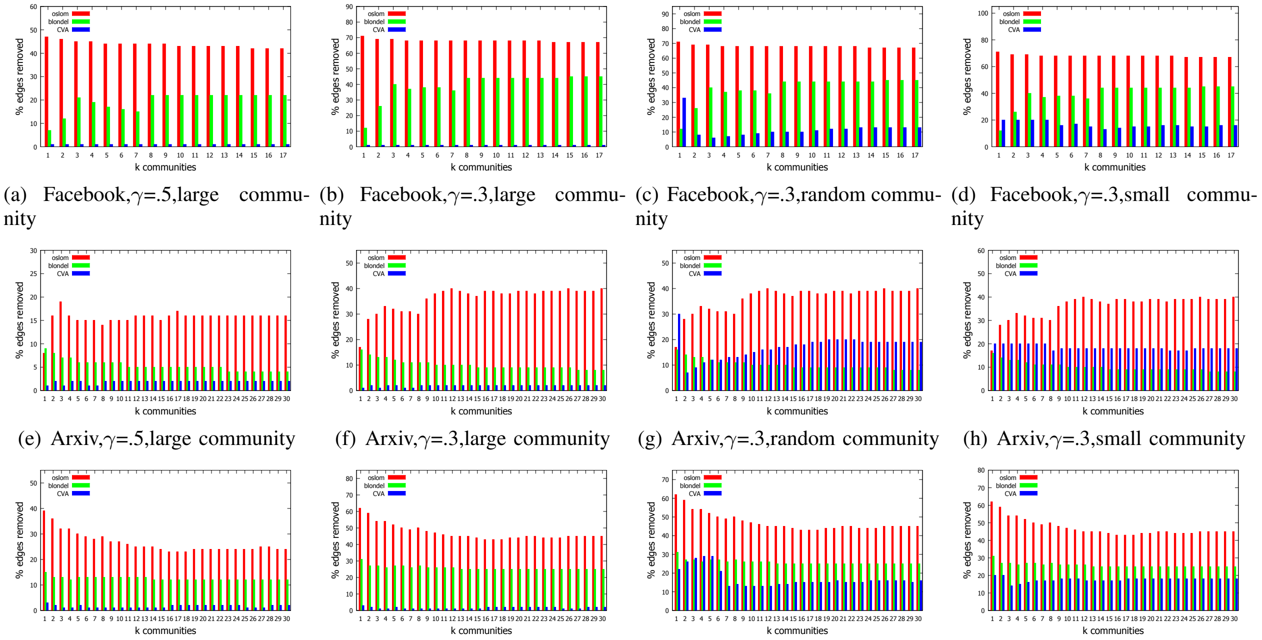
As we break more and more communities in Enron, Oslom requires decreasing number of edges on average to break them. The reason is, the smaller the community becomes the fewer the edges are needed to break them. In all of these cases we have put the performance of CVA in parallel to visualize how communities detected by different community detection algorithms are broken compared to the ones detected by density-based algorithm in terms of breaking k communities. In all of the cases, CVA requires very few edges to break all the communities. In general communities detected by Oslom requires more edge removals compared to any other approach. One of the reasons behind this behavior is that Oslom produces overlapping communities and as a result it requires more edges to break those communities.

The second column of Fig. 2 depicts the behavior of different networks for $\gamma = 0.3$. This means the strictness imposed by γ will necessitate more edges to be removed as few nodes are allowed to be retained if the community is to be broken. This is evident from each of the figures Fig. 2(b), Fig. 2(f) and Fig. 2(j). One thing to notice in this regard, the need for more edge removal is equally true for both of the community detection algorithms: Blondel and Oslom. The percentage of edges needed to break only 1 ($k = 1$) community increase by almost double when we decrease γ from 0.5 to 0.3. Even though we impose more strictness, still, in case of Blondel community detection algorithm, as low as only 7% for Arxiv and 24% for Enron networks on average are required to break k communities. Facebook communities, since they are strongly connected with more internal edges as was seen in earlier cases, require more (35%) edges to break all selected communities. Oslom in this case also needs more edges to be removed compared to Blondel for breaking the same number of communities. This is consistent with the behavior we observed so far in general for Oslom.

Next, we consider k randomly selected communities in the third column of Fig. 2. It corroborates the earlier observations that breaking a community requires more edges in case of Oslom compared to Blondel. Only a small percentage of edge removal breaks all k communities for both Arxiv and Enron networks. Facebook communities, as mentioned for other cases, require more edges to break all the selected communities. For k smallest communities we can see almost similar behavior as large communities except for the fact that this time we need comparatively smaller number edges to be removed for Oslom. The interesting outcome that we can sift from all of these figures points to the fact that in many cases few edges are enough for breaking communities.

Impact of the Location of Communities: In order to understand how the communities are interconnected and intra-connected and what is the impact of their relative structural position in the network on the vulnerability of the communities, we consider three different cases for each of the data set. We choose two communities on random basis using three criteria and find out how difficult it is to break them by removing the optimal number of critical edges. The first criteria chooses two communities who do not have any connecting edges between them, i.e., non-adjacent communities, the second criteria opts for two adjacent communities and tries to break it based on the internal connections of each of the communities only without taking into consideration the inter-community edges. The third criteria does the same as the second one but this time it takes into consideration the inter-community edges while breaking the communities. We call first criteria 'non adjacent community', the second one 'adjacent community inter-edge not considered' and the third one 'adjacent with inter-edge'. Table III shows the percentage of edges needed to break two communities in each of the three criteria. We considered Blondel community detection algorithm for this case with $\gamma = 0.3$ and run over 50 different combination of random communities.

Intuitively, communities with connecting in-between edges are easier to break due to the attraction of the neighboring communities. However, the above analysis from Table III shows that this is not generally true. Moreover, for breaking two non-adjacent and adjacent communities, the results are quite similar to the ones we have seen in Fig. 2 for random communities.



(a) Facebook,γ=.5,large community

(b) Facebook,γ=.3,large community

(c) Facebook,γ=.3,random community

(d) Facebook,γ=.3,small community

(e) Arxiv,γ=.5,large community

(f) Arxiv,γ=.3,large community

(g) Arxiv,γ=.3,random community

(h) Arxiv,γ=.3,small community

(i) Enron,γ=.5,large community

(j) Enron,γ=.3,large community

(k) Enron,γ=.3,random community

(l) Enron,γ=.3,small community

Fig. 2: Edge removal count by greedy algorithm for breaking k largest communities. $\gamma = 0.5$ in first column, $\gamma = 0.3$ in second column and $\gamma = 0.3$ in third column for random communities. The fourth column shows all the small communities with $\gamma = 0.3$

TABLE III: Network Characteristics

Data Set	non adjacent community	adjacent community inter-edge not considered	adjacent community with inter-edge
Facebook	12%	13%	12.4%
Arxiv	15%	11%	11%
Enron	20%	18%	17%

The outcome of this experimental result is consistent with what CCF does. This re-establishes the already claimed conjecture that communities are in fact easy to break. We have also observed that for some cases, as low as 1% edge removal is enough to break the community. To explore one of the reasons behind this more closely, we also consider a small community detected by Oslom community detection algorithm in Enron data set. The observation is depicted in Fig. 3. The internal structure seems to be modular and connected through few number of important edges. This justifies our approach of partitioning each of the communities into parts to break them. Interestingly the critical edges that were selected by CCF are exactly the same one shown in this figure in pink. This shows that communities can be broken by removing some crucial edges that keep different parts inside a community closer.

We also observe similar edges in Fig. 4 in another randomly selected community detected by Oslom in Facebook. These edges act as the connecting force in a community, removal of which results in broken community.

V. Related work

Although a lot of work has been performed on network vulnerability assessment, none of them really targeted the

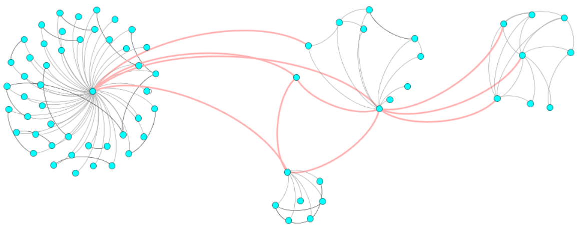


Fig. 3: A small community detected by Oslom for $\gamma = 0.3$ in Enron network. Here the internal structure shows parts are connected through small number of edges. Our greedy algorithm removes the pink cut edges.

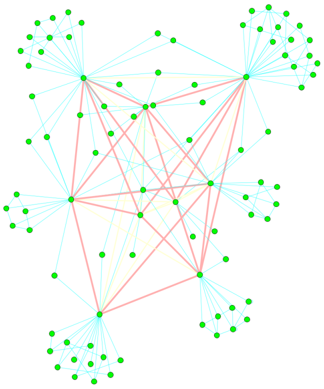


Fig. 4: A community detected by Oslom for $\gamma = 0.3$ in Facebook network. Here the internal structure shows parts are connected through small number of edges in pink.

problem from community structure point of view by defining quantification measure for *broken community*. Nam et al. [19] deals with community structure vulnerability from node point of view based on the Normalized Mutual Information (NMI) measure. They find out how different the network communities are once nodes are removed, but does not address the core issue whether the communities are broken or not. The vulnerability of network function and structure has been examined under the node centrality metrics, such as high degree and betweenness centrality, or under the available number of compromised $s-t$ flows [18], etc. However, none of these works explore how difficult breaking communities is. Due to its crucial role in the network, any significant restructure or transformation of the community structure, resulted from important edge removal, can potentially change the entire network organization and consequently lead to a malfunction or unpredictable performance of the whole network.

The literature on community structure and its detection can be found in an excellent survey of Fortunato et al. [7]. Assessing the vulnerability of network community structure, however, has so far been a relatively untrodden area. In his recent work [3] Borgatti address the problem of discovering key players in a network. A large body of work has been devoted to find the node roles within a community by a link-based technique together with a modification of node degree [20], by using the spectrum of the graph [24], by using a within-module degree and their participation coefficient [10], or by the detection of key nodes, overlapping communities and “date” and “party” hubs [13]. However, none of these approaches discusses whether the communities are strong enough under sustained attack or not.

On the assessment of network vulnerability, existing studies mainly focus on assessing the average shortest path length [1], and the global clustering coefficient [17]. Dinh et al. [5] suggested the β-disruptor problem to find a minimum set of edges or nodes whose removal degrades the total pairwise connectivity to a desired degree. Shen et al. [21] proposed Critical Link Disruptor (CLD) optimization problem to identify critical edges in a network whose removals maximally destroy the networks functions. None of these works consider the assessment of network vulnerability from community structure point of view.

VI. CONCLUSION

We make a novel attempt to study the community vulnerability problem for assessing the system fragility under edge removal. We formulate the density-based broken community problem and show its complexity. We also provide an efficient approximation algorithm for solving this problem after proving its ratio. In addition, we propose a heuristic, CCF, for solving the general version of breaking community problem. Experimental results on real world data under this newly defined framework give us insightful knowledge about the underlying community structure. We find out that communities in real-world networks are susceptible to edge failures and in many cases the failure of only a small number of critical edges can break major communities in the network.

REFERENCES

[1] R. Albert, I. Albert, and G. L. Nakarado. Structural vulnerability of the north american power grid. *Phys. Rev. E*, 69(2), Feb 2004.

[2] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory and Experiment*, 2008.

[3] Stephen P. Borgatti. Identifying sets of key players in a social network. *Comput. Math. Organ. Theory*, 12(1):21–34, 2006.

[4] ArXiv dataset. http://www.cs.cornell.edu/projects/kddcup/datasets.html. *KDD Cup 2003*, Feb 2003.

[5] Thang N. Dinh, Ying Xuan, My T. Thai, Panos M. Pardalos, and Taieb Znati. On new approaches of assessing network vulnerability: hardness and approximation. *IEEE/ACM Trans. Netw.*, 20(2):609–619, April 2012.

[6] Gregory Dobson. Worst-case analysis of greedy heuristics for integer programming with nonnegative data. *Mathematics of Operations Research*, 7(4):515–531, 1982.

[7] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010.

[8] S. Fortunato and C. Castellano. Community structure in graphs. *eprint arXiv: 0712.2716*, 2007.

[9] Wei Gao, Qinghua Li, Bo Zhao, and Guohong Cao. Multicasting in delay tolerant networks: a social network perspective. pages 299–308, 2009.

[10] Roger Guimera and Luis A. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, feb 2005.

[11] Pan Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *Mobile Computing, IEEE Transactions on*, 10(11):1576–1589, Nov.

[12] George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. *SIAM Review*, 2(41), 1998.

[13] Istvan A. Kovacs, Robin Palotai, Mate S. Szalay, and Peter Csermely. Community landscapes: An integrative approach to determine overlapping network module hierarchy, identify key nodes and predict network dynamics. *PLoS ONE*, 5(9):e12528, 09 2010.

[14] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical review. E. 80*, 2009.

[15] Andrea Lancichinetti, Filippo Radicchi, Jos J. Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4):e18961, 04 2011.

[16] J. Leskovec, K. J. Lang, Dasgupta A., and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[17] Luciano, F.A. Rodrigues, G. Travieso, and V. P. R. Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, Aug 2007.

[18] Timothy C. Matisziw and Alan T. Murray. Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure. *Comput. Oper. Res.*, 36(1):16–26, January 2009.

[19] N. P. Nguyen, Md Abdul Alim, Y. Shen, and M. T Thai. Assessing network vulnerability in a community structure point of view. *ASONAM*, pages 231–235, 2013.

[20] Jerry Scripps, Pang-Ning Tan, and Abdol-Hossein Esfahanian. Node roles and community structure in networks. pages 26–35, 2007.

[21] Y. Shen, N. P. Nguyen, Y. Xuan, and M. T. Thai. On the discovery of critical nodes and links for assessing network vulnerability. *IEEE Transactions on Networking*, 21(3):963–973, 2013.

[22] IBM ILOG CPLEX Optimization Studio. http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud. 2014.

[23] Bimal Viswanath, Ansley Post, Krishna P. Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. SIGCOMM ’10, pages 363–374, 2010.

[24] Yang Wang, Zengru Di, and Ying Fan. Identifying and characterizing nodes important to community structure using the spectrum of the graph. *PLoS ONE*, 6(11):e27418, 11 2011.