On Defining and Computing Communities

Martin Olsen

AU Herning
Aarhus University
Birk Centerpark 15, DK-7400 Herning, Denmark
Email: martino@hih.au.dk

Abstract

Inspired by the planted $l$-partition model, the hierarchical random graph model and observations on real networks we define a community structure of a graph as a partition of the nodes into at least two sets with the property that each node has connections to relatively many nodes in its own set compared to any other set in the partition. We refer to the sets in such a partition as communities. We show that it is NP-hard to compute a community containing a given set of nodes. On the other hand, we show how to compute a community structure in polynomial time for any connected graph containing at least four nodes except the star graph $S_n$.

1 Introduction

Much research has been done in the field of detection of community structure in networks with many natural applications (the world wide web, communication networks, social networks, biological networks, etc.) and we refer to the excellent work of Fortunato (2010) for a thorough and up-to-date review of the subject. A community structure can be loosely defined as a partition of the nodes into communities such that there is relatively many connections internally in each community. Even though it is intuitively clear what a community structure is, it seems hard to give a precise quantitative definition of the concept and according to Lancichinetti & Fortunato (2009) "... there is still no agreement among scholars on what a network with communities looks like". Nevertheless, "... there has been a silent acceptance of a simple network model ..." for modeling networks with community structure (again citing (Lancichinetti & Fortunato 2009)). This model is the planted $l$-partition model (Condon & Karp 2001) in which a partition of the nodes is "planted" and nodes within the same group will connect to each other with a higher probability compared to nodes belonging to different groups in the partition. The hierarchical random graph-model (Clauset et al. 2007, 2008) follows the same principle with respect to the probabilities for setting up connections, and this model is described in more detail below. The aim of this paper is to present what we believe to be an obvious and intuitively clear definition of a community based on these models – and supported by observations on real world networks (Choifnes et al. 2010, Guimera et al. 2006) – and analyze the computational complexity of detecting communities.

1.1 Related Work

Newman & Girvan (2004) define the modularity measure for partitions and argue that a high value of the modularity measure indicates strong community structure and Brandes et al. (2008) show that finding a partition with maximum modularity is NP-hard. Many apparently successful heuristics have been proposed to optimize the modularity, but it should be noted that modularity involves "counterintuitive" aspects according to Brandes et al. (2008) and Fortunato & Barthélemy (2007). Flake et al. (2000, 2004) define a community as a set of nodes $C$ such that each node in $C$ has at least as many connections to nodes in $C$ as to nodes outside $C$, and Radicchi et al. (2004) use similar definitions of a community even though Kristiansen et al. (2004) use the term strong defensive alliance for such a set. Flake et al. allow multiple connections between two nodes and show that detecting a community structure based on their definition is NP-hard (Flake et al. 2004). Now consider the graph shown in Fig. 1 with 6 cliques where each clique contains 5 nodes and where each node is connected to the 4 other nodes in their clique and some random nodes not in their own clique. The cliques would not form a partition into communities according to the definitions in (Flake et al. 2000, Radicchi et al. 2004) which seems counterintuitive: The clique in the upper left corner is not a community according to the definition by Flake et al. Cluset et al. (2007, 2008) present a model for producing graphs with hierarchical community structure – communities within communities – so called hierarchical random graphs. Clauset et al. assume that the nodes set up connections independently with probabilities that depend on their "degree of relatedness". Typically, the probabilities increase the more related the nodes are in what Clauset et al. refer to as the "traditional picture of communities". Other definitions of communities are presented in (Olsen 2008, 2009) by the author of this paper. In (Olsen 2009) it is suggested to define a community structure as a partition where each node has at least as many connections to nodes in their own set as to nodes in any other set. Detecting community structures based on this definition is showed to be NP-hard in (Olsen 2009) in the general setting where multiple connections between two nodes are allowed. The suggested definition in (Olsen 2009) leaves room for improvement since it ignores the cardinality of the sets in the partition.
Choffnes et al. (2010) analyze a network of 10,000 BitTorrent (BT) users with a connection between two users if they share interest in the same content. Choffnes et al. show "that strong communities form naturally in BT, with users inside a typical community being 5 to 25 times more likely to connect to each other than with outside users." Guimera et al. (2006) study an e-mail network of approximately 1700 users at the University Rovira i Virgili, Tarragona, Spain. An edge connecting two users a and b shows that a has sent an e-mail to b and that b has replied. The users are affiliated with different centers and Guimera et al. present two "typical cases" of centers where the users on average connect with a higher probability to users in their own center compared to the other centers.

1.2 Contribution and Outline

We present intuitively clear formal definitions of community structures and communities in Sect. 2. The definitions are supported by the observations on real networks (Choffnes et al. 2010, Guimera et al. 2006) just presented and inspired by network models (Clauset et al. 2007, Condon & Karp 2001, Lancichinetti & Fortunato 2009) where nodes belonging to the same group are more likely to connect to each other compared to nodes from different groups. We show in Sect. 3 that the community detection problem is NP-hard when we require the community to contain a given subset of the nodes. Finally, we show in Sect. 4 how to compute a community structure in polynomial time for any connected graph containing at least four nodes except $S_n$. This means that we can efficiently compute a community containing a given node for any node in almost any connected graph.

Before presenting the definitions of community structures and communities we will briefly explain some notation used in this paper. The graphs considered in this paper are unweighted and undirected. For such a graph $G(V,E)$ we will let $N_i(T)$ denote the neighbors of $i$ in $T \subseteq V : N_i(T) = \{ j \in T : \{ i, j \} \in E \}$. A partition $\Pi$ of $V$ is a collection of non-empty disjoint subsets of $V$ with union $V$. We let $\Pi_i$ denote the set in $\Pi$ containing $i$. The star graph $S_n$ contains $n + 1$ nodes where one of the nodes is connected to the $n$ other nodes that all have degree 1.

2 Defining Community Structures and Communities

Inspired by the empirical work of Choffnes et al. (2010) and Guimera et al. (2006) we will use the following generalization of the planted l-partition model (Condon & Karp 2001) and the assortative version of the hierarchical random graph-model (Clauset et al. 2007, 2008) to motivate our definition of community structures and communities in a graph $G(V,E)$:

Let the creation of edges be controlled by a stochastic process where the binary random variable $X_{ij} \in \{0,1\}$ is 1 if and only if $(i,j) \in E$ for $i \neq j \in V$. Now assume, that we have a partition $\Pi$ of the nodes such that each node $i$ connects to another node in $\Pi_i$ with an average probability that is at least as high as the average probability for $i$ connecting to a node in $C$ for any $C \in \Pi$. Formally, the assumption is that the following holds for all $i \in V$ and all $C \in \Pi$:

$$\sum_{j \in \Pi_i \setminus \{i\}} P(X_{ij} = 1) \geq \frac{\sum_{j \in C \setminus \{i\}} P(X_{ij} = 1)}{|C|}.$$  

(1)

This leads us to the following definition of a community structure $\Pi$: For any node $i \in V$ we simply require the number of connections to nodes in $\Pi_i$ divided by $|\Pi_i| - 1$ to be at least as high as the number of connections to any other community $C \in \Pi$ divided by $|C|$. We are not claiming to be the first to present this definition but we have not been able to find it anywhere else in the literature. The formal definition is as follows:

Definition 1. A Community Structure for an undirected graph $G(V,E)$ is a partition $\Pi$ of $V$ such that $|\Pi| \geq 2$ and $|C| \geq 2$ for all $C \in \Pi$ and

$$\forall i \in V, \forall C \in \Pi : \frac{|N_i(\Pi_i)|}{|\Pi_i| - 1} \geq \frac{|N_i(C)|}{|C|}.$$  

(2)

If we use expectations on both sides on the inequality in (2) we get (1) – please note that we are not assuming independence of the random variables. You might argue that our definition of a community structure is too strict since we can not expect that the inequality in (2) holds for all the nodes. It should be noted that our intractability result in Sect. 3 implies that computing non-trivial partitions (ie. with at least two sets) not separating the nodes in a given set is NP-hard if we require (2) to hold for a maximum number of nodes.
A community is simply defined as a member of a community structure:

Definition 2. A community is a member of a community structure.

This definition of a community is in some sense global: you have to be sure that all other nodes belong to some other community before you can conclude that a subset is a community. Consider for example a node with degree 1 that connects to a big clique. According to our definition the clique is not a community. Whether this is reasonable is left to the judgment of the reader.

Before formally studying the computational complexity of computing communities we will look at the definition of community structures and communities from a more informal angle. The setting of our example is a reception with beers and snacks where people walk around and chat. Imagine you are in a group of people where you know 1 4 of the people but you notice another group where you know 1 4 of the members. In this case, you might decide to join that other group. At some point the formation of groups might reach an equilibrium state (a Nash equilibrium in game theoretic terms) where all the guests at the reception are satisfied. This equilibrium is a community structure and the groups are communities according to our definition. We prove that there is a non-trivial equilibrium for (almost) any reception and that this equilibrium is computable in polynomial time. If we on the other hand require that a certain bunch of people should stay together we show that the problem of computing a non-trivial equilibrium becomes NP-hard.

3 Computing Communities Containing Specific Nodes

A graph can allow several community structures according to Definition 1 and as we will see in the next section it is possible to efficiently compute a community structure for any graph containing at least four nodes except for star graphs (that clearly do not allow any community structure). We might obtain three different community structures by dividing users according to their gender, education and center affiliation respectively in the e-mail graph from (Guimera et al. 2006). Given that there may be multiple community structures it would certainly be useful with an algorithm that given partial information on the community structure would compute a community structure consistent with the partial information – or a partition of the nodes consistent with the partial information and with a minimum number of nodes violating (2). As an example, we would maybe like to compute all members of a center given a set of known members of the center in the e-mail graph presented in (Guimera et al. 2006). We now show that it is NP-hard to compute community structures – or partitions with a minimum number of nodes violating (2) – given partial information ruling out a polynomial time algorithm if NP 6= P. To be more precise we will show that the following problem is NP-complete:

Definition 3. The COMMUNITY problem:

• Instance: A pair \((G, S)\) consisting of an undirected graph \(G(V, E)\) and a subset of nodes \(S \subseteq V\).

• Question: Does a community \(C \subset V\) exist such that \(S \subseteq C\)?

We will show that the COMMUNITY problem is intractable by reducing the problem of deciding whether a given boolean formula in conjunctive normal form is satisfiable or not. This problem is known to be NP-complete even for formulas where each clause contains exactly three different literals (Garey & Johnson 1979). The formal definition of the problem we will reduce is the following:

Definition 4. The 3SAT problem:

• Instance: A boolean formula \(f\) in conjunctive normal form where each clause contains exactly three different literals.

• Question: Is \(f\) satisfiable?

We now formally state and prove the intractability of the COMMUNITY problem.

Theorem 1. COMMUNITY is NP-complete.

Proof. The COMMUNITY problem is clearly a member of NP.

We will now show how to reduce the 3SAT problem to the COMMUNITY problem. Let \(f\) represent an instance of 3SAT with \(m\) clauses where each clause consists of three different literals on the boolean variables \(x_1, x_2, \ldots, x_n\). We transform \(f\) into the COMMUNITY instance \((G, S)\) in polynomial time in the following steps – a generic instance \((G, S)\) is depicted in Fig. 2 and Fig. 3 provides a more specific example with the COMMUNITY instance corresponding to \(f = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor \neg x_3) \land (x_3 \lor \neg x_2 \lor \neg x_3)\):

1. We start by adding the clauses \([c_1, c_2, \ldots, c_m]\) and the literals \([x_1, x_2, x_3, \ldots, x_n]\) to \(V\) and connect each clause with its three literals with edges in \(E\).

2. We add another group of nodes \([z_1, z_2, \ldots, z_n]\) to \(V\) and add the edges \([z_i, x_i]\) and \([z_i, x_i]\) to \(E\) for \(i = 1, 2, \ldots, n\).

3. Another group of nodes \([w_1, w_2, \ldots, w_m]\) of nodes is added and a literal now connects to an arbitrarily picked \(u\)-node for each connection to a clause or a \(z\)-node. We can assume that a literal occurs in no more than \(m-1\) clauses – if a literal occurs in \(m\) clauses we transform \(f\) into a yes-instance.

4. A layer of \(m+n\) unnamed nodes is inserted into \(V\) and all of these nodes connect to all \(u\)-nodes.

5. \(G\) is expanded with the nodes \([y_1, y_2, \ldots, y_m]\) and the edges \([y_i, c_i]\) and \([y_i, w_i]\) for \(i = 1, 2, \ldots, m\).

6. Finally, we let \(S\) consist of the clauses and the \(z\)- and \(y\)-nodes.

Before formally showing that \(f\) is a yes-instance of 3SAT if and only if \((G, S)\) is a yes-instance of COMMUNITY, we present some less formal remarks to explain the basic idea of the part of the proof concerning the \(i\)-direction. The edges between the \(z\)-nodes and the literals are there to make sure that at least one of the literals \(x_i\) or \(\neg x_i\) has to join the nodes in \(S\) if we want to form a community containing \(S\) for \(i = 1, 2, \ldots, n\). The \(y\)-nodes, \(u\)-nodes and the unnamed nodes ensure that no more than \(n\) literals join \(S\). This makes it possible for us to interpret the literals joining \(S\) as a truth assignment that satisfies \(f\)
because the clause nodes satisfy the inequality in (2). For the "only if"-direction, it should be noted that the literals connect to the \( w \)-nodes in order to satisfy the inequality in (2) even if they do not join the members of \( S \).

We now show that the two instances are equivalent:

- Now assume, that \( f \) is a yes-instance of 3SAT. Let \( C \subseteq V \) contain \( S \) and all the literals corresponding to the satisfying assignment for \( f \): \( x_i \in S \) if \( x_i = \text{True} \) and \( x_i \in S \) if \( x_i = \text{False} \). Consider as an example the COMMUNITY instance in Fig. 3 where the truth assignment \((\text{False},\text{False},\text{True})\) satisfies \( f \). If we consider the partition \( \Pi = \{C, \bar{C}\} \) with \( C = S \cup \{x_1, x_2, x_3\} \) then we notice that \( |C| = |ar{C}| \) and we also notice that at least half of the neighbours of any node \( i \) are members of \( \Pi \), so \( C \) is clearly a community. It is not hard to verify that \((G, S)\) is a yes-instance by checking (2) for \( \Pi = \{C, \bar{C}\} \) in the general case.

- Let \( C \subseteq V \) be a community containing \( S \). We now argue that \( C \) contains exactly one of the literals \( x_i \) or \( \bar{x}_i \) for \( i = 1, 2, \ldots, n \): If \( C \) neither contains \( x_i \) nor \( \bar{x}_i \) then (2) will not be satisfied for \( z_i \). If, on the other hand, \( C \) contains more than \( n \) literals it forces all other nodes to be members of \( C \), producing a contradiction with \( C \subseteq V \). All the \( w \)-nodes have to join \( C \) to keep the \( y \)-nodes happy. All the \( c \)-nodes have at least one literal neighbour in \( C \) so the literals that join \( C \) define a satisfying assignment for \( f \).

4 Computing Community Structures Without Membership Information

In this section we show how to compute a community structure in polynomial time for almost any connected graph. Basically, it is possible unless it is obvious that such a community structure does not exist.

Theorem 2. A community structure can be computed in polynomial time for any connected undirected graph \( G(V, E) \) containing at least four nodes except \( S_n \) \((n \geq 3)\).

Proof. We will set up a polynomial time local search among certain partitions of \( V \) and prove that the result of the local search is a community structure.

Specifying the Search Space \( S \): If we have a node \( u \) in a subset \( C \) of \( V \) such that \( u \) is connected to all other nodes in \( C \), then we will refer to \( u \) as a center of \( C \) \((\forall v \in C \setminus \{u\} : \{u, v\} \in E)\). We now define the search space \( S \) to be the set of all partitions \( \Pi \) of \( V \) with \(|\Pi| \geq 2 \) such that the following holds for all \( C \subseteq \Pi \) \( |C| \geq 2 \) and \( 2 \) \( C \) has a center. The neighbours of a partition \( \Pi \in S \) are partitions in \( S \) that can be made by moving one node from one set to another set or by letting two nodes form a new set.

The Initial Candidate Solution: We pick four different nodes \( u_1, u_2, u_3, u_4 \) such that \( \{u_1, u_2\}, \{u_3, u_4\} \in E \). Now we expand the collection \( \{\{u_1, u_2\}, \{u_3, u_4\}\} \) with one node at a time carefully making sure that each set contains at least two elements and has a center. Sometimes we might have to form a new set containing the new node and another node. By using induction on the total number of nodes in the sets in the collection it is not hard to show that it is always possible to add a node and still have a collection of sets with at least two nodes and a center. Thus an element \( \Pi^{(1)} \in S \) can be computed in polynomial time.

The Objective Function: For a partition \( \Pi \) we define \( g(\Pi) \) as the number of edges in the complement graph of \( G \) connecting nodes in different members of \( \Pi \). In other words, \( g(\Pi) \) is defined as the following number where \( \bar{E} \) is the complement of \( E \):

\[ |\{\{u, v\} \in \bar{E} : u \in C_1 \text{ and } v \in C_2 \text{ for } C_1 \neq C_2 \subseteq \Pi\}| \ . \]

We now use local search on \( S \) to maximize the following function:

\[ h(\Pi) = |\Pi| + g(\Pi) \ . \]
We start with the candidate solution $\Pi^{(1)} \in S$ and pick a neighbour of $\Pi^{(1)}$ if $h$ increases for that neighbour (if there are several neighbours for which $h$ increases we pick one at random). The process is repeated until $h$ can not be improved locally.

**Analysis:** We will now prove by contradiction that the result $\Pi^{(*)}$ of the local search is a community structure. Assume that there is an $i \in V$ and $C \in \Pi^{(*)}$ violating (2) meaning that

$$\frac{|N_i(\Pi_i)|}{|\Pi_i| - 1} < \frac{|N_i(C)|}{|C|}.$$  

(3)

The set $\Pi_i \in \Pi^{(*)}$ must contain at least three elements. We now consider the following cases:

- $|C| = 2$ and $|\Pi_i| > 3$: The node $i$ is connected to the center of $\Pi_i$ but not to the third node in $\Pi_i$ (otherwise the left hand side of (3) would be 1), and $i$ must have connections to both nodes in $C$. If $i$ is moved to $C$ we would obtain a higher value of $h$ – a contradiction.

- $|C| > 2$ and $|\Pi_i| > 3$: By following the same line of reasoning as above $i$ is only connected to the centers of $C$ and $\Pi_i$. From (3) we now conclude that $|\Pi_i| - 1 > |C|$. Once again, $h$ would increase if $i$ joined $C$ – yet another contradiction.

The local search can be done in polynomial time since each step in the process can be carried out in polynomial time and $h(\Pi^{(*)}) \leq n + \frac{n(n-1)}{2}$. □

As a corollary of Theorem 2 we can efficiently compute a community containing a given node for any node in any connected graph containing at least four nodes except $S_n$. It should be noted that a community may consist of two nodes that are connected with an edge in the graph – if we observe two nodes that are connected we can not exclude the possibility that these nodes form a small community.

We conclude this paper with some suggestions for future work on computing communities following the definitions presented in Sect 2. It would be interesting to analyze the computational complexity for computing communities for various bounds on $|S|$ or to investigate the complexity of computing community structures for various bounds on the number of communities for the case $S = \emptyset$.

**References**


