A Classification for Community Discovery Methods in Complex Networks

Michele Coscia[a,b,c], Fosca Giannotti[b], Dino Pedreschi[a,b]

[a] *Computer Science Department, University of Pisa, Pisa, Italy*
[b] *KDDLab, ISTI-CNR, Pisa, Italy*
[c] *Barabasi Lab, Northeastern University, Boston, USA*

Abstract

In the last years it has been found that many real-world networks show a so-called community structure organization. Much effort has been devoted in literature in order to develop methods and algorithms that can efficiently highlight this hidden structure of the network, traditionally operating a partition of the graph. Since the network representation can be very complex and can contain different variants in the traditional graph model, each algorithm present in literature focuses on some of these properties and establishes its own definition of community. Then it extracts communities according to this definition, able to reflect only some of the features of real communities. Aim of this survey is to provide a manual for the community discovery problem. Given a meta definition of what is a community in a social network, our aim is to organize the main categories of community discovery approaches basing on their own definition of community. Given a desired definition of community and the features of a problem (size of network, direction of edges, multidimensionality, and so on) we aim to provide the set of approaches a researcher might focus on.

Keywords: Community discovery, Social network, Groups, Complex network, Graph partitioning, Graph clustering, Graph mining, Information propagation.

1. Introduction

A complex network is a mathematical model able to represent in a machine readable form many interaction phenomena that take place in the real world. A critical feature, widely studied in literature since the early complex network analysis, is the possibility to identify groups and communities inside the structure of many phenomena represented with this model.

Community detection is important for many reasons. Identifying communities and their boundaries allows for a classification of vertices, according to their topological position in the groups. It is therefore possible to identify community leaders, a task that can be very useful in order to identify, for example, critical actors in the spreading of influence in an information propagation setting. Or it is possible to identify individuals lying at the boundaries between groups, that play an important role of mediation and lead the relationships and exchanges among different communities. Finally, one can study the graph where entities are the communities and edges are set between these groups if there are connections between some of their entities in the original graph and/or if the groups overlap. In this way one attains a coarse-grained description of the original graph, which unveils the relationships between modules.

In general, a "Community" is considered to be a set of entities where each entity is closer to the other entities within the community than to entities outside it. Communities are groups of entities which probably share common properties and/or play similar roles within the interacting phenomenon we are representing. Communities may correspond to groups of pages of the World Wide Web dealing with related topics [1], to functional modules such as cycles and pathways in metabolic networks [2, 3], to groups of related individuals in social networks [4] and so on.

The community discovery task is very close to the clustering problem, a traditional data mining task. In data mining, clustering is an unsupervised learning task, which aims to assign a large set of data into homogeneous groups (clusters). In fact, community discovery can be viewed as a data mining analysis on graphs: an unsupervised classification of its nodes. Moreover, community discovery is the most studied data mining application on social networks. Other applications, such as graph mining [5], are in an early phase of their development. Community discovery has instead reached a more advanced development with contributions from different fields like physics.

Nevertheless, the community discovery problem is something more. In the classical data mining clustering we have data which is not in a relational form. Thus, in this general form, the fact that the entities are nodes connected each other through edges is traditionally not deeply considered. Therefore, it is needed to map the concept of spatial proximity among entities (i.e. vertices) in the graph

Email addresses: coscia@di.unipi.it (Michele Coscia), fosca.giannotti@isti.cnr.it (Fosca Giannotti), pedre@di.unipi.it (Dino Pedreschi)

representation setting.

The traditional and most accepted definition of proximity in a network is based on the topology of its edges. In this case the definition of community is formulated according to the differences in the densities of links in different parts of the network. Many networks, it is found, are inhomogeneous, consisting not of an undifferentiated mass of vertices, but of distinct groups. Within these groups there are many edges between vertices, but between groups there are fewer edges. Aim of a community detection algorithm is, in this case, to divide the vertices of a network into some number k of groups, while maximizing the number of edges inside these groups and minimizing the number of edges established between vertices in different groups. These groups are the desired communities of the network.

In the last years, this definition is no longer suitable due to the increasing complexity of the network representations and of the novel analytical settings, such as information propagation or multidimensional network analysis. For example, in a temporal evolving setting, two entities can be considered close each other if they share a common action profile even if they are not directly connected. Thus each novel approach to the community discovery has faced this problem and has developed its own definition of community for its own solution.

Beside this variety of different definitions of community, there are a number of interesting features of these communities. Such features can be a hierarchical or overlapping configuration of the groups inside the network. Or the graph can include directed edges, thus giving importance to this direction when considering the relations among entities. The communities can be dynamic, i.e. evolving over time, or multi-relational, i.e. there might be multiple relations and there might be sets of individuals that behave as isolated entities in each relation of the network forming a dense community when considering all the possible relations at the same time.

This extreme richness of different definitions and features had as a result the publication of an impressive number of excellent solutions to the community discovery problem. It is not a surprise, then, that a number of review papers, collecting a description of all these methods, has been proposed in literature (even very recently [6]). However, we believe that a novel point of view on this topic is needed. The present reviews, in fact, are devoted to analyze the different techniques from a very technical perspective. They do not consider to organize the impressive number of algorithms according to their definition of community. In other words, the aim of those reviews is to talk to people interested in build a new community detection algorithm, not to people who want to use the methods present in literature, which is the audience of this paper.

In order to do this, we have chosen to cluster the community discovery algorithms by considering their definition of what is a community, used as basis of what kind of groups they aim to extract from the network. For each algorithm we record the characteristics of the output of the method, thus highlighting for which set of features the reviewed algorithm is suitable and for which one it is not. Further, we consider some general frameworks able to provide both a community discovery approach and a general technique that is applicable to other graph partitioning algorithms adding new features to these other methods.

The remainder of the paper is organized as follows. In Section 2 we will provide the community discovery problem statement in the most general way possible and the meta definition of what is a community. In Section 3 is explained the classification of algorithms based on the community definitions. Then, from Section 4 to Section 11, we will present the main categories of approaches given our problem definition, along with the most important and known works in each given category. In Section 12 we will briefly present some other related works, surveys about community discovery in social networks, along with the motivations of a novel point of view about these methods, that is this paper. Finally, Section 13 concludes the survey and provides a perspective about some possible future works.

2. Problem Definition

2.1. Problem Representation

We are given a graph G denoted by a quadruple $G = (V, E, L, C)$, where V is a set of labeled nodes, E is a set of labeled edges, L is a set of edge labels and C is a set of node labels. E is a set of quadruples of the form (u, v, l, w) where $u, v \in V$ are nodes, $l \in L$ is a label and w is an integer that represent the weight of the relation. We assume that given a pair of nodes $u, v \in V$ and a label $l \in L$ it may exist only one edge (u, v, l, w), however the direction of the edge is considered in the model, thus edges (u, v, l, w) and (v, u, l, w) are considered distinct. We assume also that each node can be labeled with one or more category $c \in C$. Moreover, we consider the temporal evolution of the network. Thus each edges, and node, can be labeled with an arbitrary number of timestamps that represent the time in which the edge appear and disappear in the network. Also the labels of a given node can change over time. Please note that nodes can create/delete edges in the network and/or change/introduce/delete one or more label in their category set $C_v = \{c_1, c_2, ..., c_n\}$. We call such events "actions" performed by the nodes.

With this complex model we are able to represent all possible variants in the graph representation of a complex real world phenomenon. For example we can model multi-relational networks by considering the edge labels L as the different relations (dimensions) of the network. We can also represent simpler models, such as unweighted networks, by assigning to every edge in the network the same weight $w = 1$.

In the remainder of the paper we will use the notation presented in Table 1. We will introduce new symbols and notations when needed in the presentation of a particular method and not useful for the others.

Symbol	Description
n	Number of vertices of the network
m	Number of edges of the network
k	Number of communities of the network
$\bar{K}$	Avg degree of the network
K	Max degree in the network
T	Number of action in the network
A	Max number of actions for a node
D	Number of dimensions (if any)
c	Number of vertex types (if any)
t	Number of time step (if any)

Table 1: Resume of the main notation used in the paper.

2.2. Community Meta Definition

We now present our meta definition of community in a complex network. With this meta definition we create an underlying concept which poses the basis of this survey and it is able to include all the possible definition variants present in literature.

Meta Definition 1 (Community). *A community in a complex network is a set of entities that share some closely correlated sets of actions with the other entities of the community. Here we consider the direct connection a particular, and most important, kind of action.*

Aim of a community discovery algorithm is to identify these communities in the network. The desired result is the list of sets of entities grouped together. Starting from this meta definition we can model the main aspects of the current issue of discovering communities in complex networks.

Density-based definitions. In this classical setting, as we said in the Introduction, the definition is entirely based on the topology of the network edges. The community is defined as a group in which there are many edges between vertices, but between groups there are fewer edges. Aim of a community detection algorithm is, in this case, to divide the vertices of a network into some number k of groups, while maximizing the number of edges inside these groups and minimizing the number of edges that run between vertices in different groups. In our definition we consider the connection between two vertices a particular kind of action. Hence, if we group entities maximizing their common actions, we group them also maximizing the edges inside the community. The community discovery is exactly the same if the edge creation is the only action recorded in the network representation. Further, by considering in the meta definition different kinds of sets of action, we can also model the overlapping situation: for certain set of actions (i.e. connections) a node belongs to one community, for another set of actions it belongs to another community.

Vertex similarity-based definitions. As pointed out by Fortunato [6], it is natural to assume that communities are groups of vertices similar to each other. One can compute the similarity between each pair of vertices with respect to some reference property, local or global, no matter whether they are connected by an edge or not. Each vertex ends up in the cluster whose vertices are most similar to it. By considering an evolving setting in our problem representation, also the presence or the absence of a particular property (i.e. a label of the vertex), we are able to model the similarity measures as the similarity of the set of actions.

Action-based definitions. In this setting, that is gaining increasing attention in literature, entities can be grouped by the set of actions they perform inside the network. For example, in [7] authors consider a multi-mode network in which users are connected to queries and ads. Two user can be considered part of the same community if they are connected to the same queries (i.e. they perform the same action) even if they are not directly linked each other. The discovery of communities based on this definition can be performed both considering or not considering the presence of a direct link among entities. Both cases are included in our meta definition.

Influence Propagation-based definitions. In some works, the concept of "tribe" has been introduced. In [8], a tribe is defined as a set of entities that are influenced by the same leaders. A node is a leader if it performed an action and, within a chosen time bound after this action, a sufficient number of other users performed the same action. The role of social ties in this influence spread is considered. Thus, according to our definition, the set of users that frequently perform the same actions due to the influence of their leaders are considered a community.

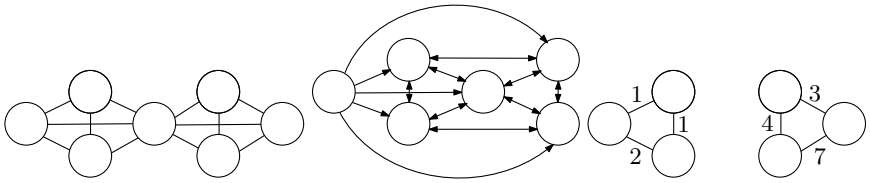
2.3. Problem Features

The features to be considered in the complex task of detecting communities in graph structures are many. In this section we will provide a resume of the considered features. This, and the following sub section, is the description of the data included in Table 2, that collects the main characteristics and the classification of all the methods here reviewed.

First, Table 2 records the main properties of a community discovery algorithm. These properties can be grouped in two classes. The first class consider the features of the problem representation, the second one are the characteristics of the approach.

Inside the first class of features we group all the possible variants in the representation of the original real world phenomenon. The most important features we will consider in this paper are the following:

- **Overlapping**. In some real world networks the communities can share one or more common nodes. For example, in social networks actors may be part of different communities: the work, the family, the friends and so on. All these communities will share a common member, and usually more since a work colleague can also be your friend outside the working



(a) Overlapping Communities (b) Directed Community (c) Weighted Communities

Figure 1: Different community features.

environment. Consider, for example, the structure depicted in Figure 1(a): the central node is shared by the two communities. A community discovery algorithm should consider the possibility of identify these overlapping situations. In Table 2 we record if an algorithm consider this feature in the "Overlap" column.

- **Directed**. Some phenomena in real world must be represented with edges and links that are not reciprocal. This, for example, is the case of the web graph: an hyper-link from one page to another is directed and the other page may not have another hyper-link pointing in the other direction. In Figure 1(b) we have depicted an example in which the direction of the edges should be considered. The leftmost node is connected to the community, but only in one direction. If reciprocity is an important feature, the leftmost node should be considered outside the depicted community. In Table 2 we record if an algorithm consider this feature in the "Dir" column.

- **Weighted**. A group of connected vertices can be considered a community only if the weights of their connections are strong enough, over a given threshold. In the case of Figure 1(c), the left group might not be strong enough to form a community. In Table 2 we record if an algorithm consider this feature in the "Weight" column.

- **Dynamic**. Following our problem representation in Section 2.1, in our setting we have a set of edges that can appear and disappear. Thus, also communities might evolve over time. In Table 2 we record if an algorithm consider this feature in the "Dyn" column.

The second class of features collects some desired properties that an approach might have. These features can specify constraints for input data, improve expressive power of the results or may facilitate the community discovery task.

- **Parameter free**. A desired feature of an algorithm, especially in data mining research, is the absence of parameters. In other words, an algorithm should be able to make explicit the knowledge that is hidden inside the data without needing any previous information about the data nor the problem (for instance the number of the communities). A parameter free community discovery algorithm is able to return a partition of the network without further indications from the analyst. In Table 2 we record if an algorithm provide this feature in the "NoPar" column.

- **Multidimensional input**. Multidimensionality in networks is one of the emerging topic in recent years. A network is said to be multidimensional if it contains a number of different kinds of relation that are established among the nodes of the network. Thus, when dealing with multiple dimensions, the notion of community changes. Our proposed Meta Definition 1 captures this complex environment, by representing the creation or the absence of a particular edge in a particular dimension with an action. This concept of multidimensionality is used (with various names: multi-relational, multiplex, and so on) by some approaches as a feature of the input considered by the approach. In Table 2 we record if an algorithm provide this feature in the "MDim" column.

- **Incremental**. Another desired feature of an algorithm is its ability to provide an output without an exhaustive search of the entire input. An incremental approach to the community discovery can be to classify a node in one community by looking only its neighborhood, or the set of nodes two hops away, or to put the newcomers in one of the previously defined communities without starting the community detection process from the beginning. In Table 2 we record if an algorithm provides this feature in the "Incr" column.

- **Multipartite input**. Many community discovery approaches are able to work even if the network has the particular form of a multipartite graph. The multipartite graph, however, is not entirely a feature of the input that we might want to consider for the output. Many algorithms often use a (usually) bipartite projection of a classical graph in order to apply some efficient computations. As in the case of multidimensionality, this is the reason of including the multipartite input as a feature of the approach and not of the output. In Table 2 we record if an algorithm provide this feature in the "Multip" column.

There is a last "meta feature" that we will consider. This feature is the possibility of apply the considered approach to another community discovery technique adding new features to the "guest method". This meta feature will be highlighted with an asterisk note next to the algorithm name.

Table 2 records other informations. In "Complexity" column we record the time complexity of the presented methods. The two "BES" columns present the Biggest Experiment Sizes, in terms of nodes ("BESn") and edges ("BESm"), that are included in the original paper reviewed. Please note that the Complexity and BES columns often offers an evaluation of the actual values, since the original work did not provide an explicit and clear analysis of complexity or their experimental setting. When it is considered not trivial to evaluate the complexity, or no experimental details are provided, we insert a question mark in the column.

3. The Definition-based classification

In the following we will present an extensive review of community detection approaches. We will group together in each section all the algorithms that share the same definition of what is a community, i.e. same conditions satisfied by a group of entities that allow to cluster them together in a community. The proposed categories are the following:

- **Feature Distance** (Section 4). Here we will collect all the community discovery approaches that starts from the assumptions that a community is composed by entities who share ubiquitously a very precise set of features, with similar values (i.e. defining a distance measure on their features, the entities are all close to each other). A common feature can be an edge or any attribute linked to the entity (in our problem definition: the action). Usually, these approaches propose this community definition in order to apply classical data mining clustering techniques, such as the Minimum Description Length principle [48].
- **Internal Density** (Section 5). In this group we will consider the most important articles that define a community as a group in which the number of edges present among its members is considerably higher than the expected number of edges in a random graph.
- **Bridge Detection** (Section 6). This section includes the community discovery approaches based on the concept that communities are dense parts of the graph among which there are very few edges able to break down in pieces the network if removed. These edges are "bridges" and the components of the network resulting by their removal are the desired communities.
- **Diffusion** (Section 7). Here we will include all the approaches to the community discovery task relying on the idea that communities are groups of nodes that can be influenced by the propagation (diffusion) of a certain property or information inside the network. In addiction, the community definition can be narrowed to the groups that are influenced only by the very same set of sources of these diffusions.
- **Closeness** (Section 8). A community can be defined also as a group of entities that can reach each of its own community companions with very few hops on the edges of the graph, while the entities outside the community are significantly farther.
- **Structure** (Section 9). Another approach to community discovery is to define exactly the community as a very precise and almost immutable structure of edges. Often these structures are defined as combination of smaller network motifs. The algorithms following this approach define some kind of these structures and then they try to find them efficiently inside the graph.
- **Link Clustering** (Section 10). The last class of solutions that we will consider can be viewed as a projection of the community discovery problem. Instead of clustering the nodes of a network, these approaches state that is the relation that belongs to a community, not the node. Therefore they cluster the edges of the network and then the nodes belong to the set of communities of their edges.
- **No Definition** (Section 11). There are a number of community discovery frameworks which have not a basic definition of the characteristic of the community they want to explore. Instead they define some operations and algorithms to combine the results of various community discovery approaches and then use the target method community definition for their results, or they let the analyst define its own notion of community and search it into the graph.

In each Section we will provide, if possible, a simple graphical example of the definition considered.

Please note that in the following we will not focus our attention to historical approaches. Some examples of classical clustering algorithms that are not extensively reviewed are the Kernighan-Lin algorithm [49] or the classical spectral bisection approach [50]. We have chosen this approach because this survey is meant to be devoted to the most recent approaches and to the more general definitions of community. Thus, for an historical point of view of the community discovery problem, we refer to other review papers.

3.1. The Classification Overlap

It is worthwhile noting that there is a sort of overlap for some community definitions. For example a defini-

	Name	Overlap	Dir	Weight	Dyn	NoPar	MDim	Incr	Multip	Complexity	BESn	BESm	Year	Ref
Feature Distance	Evolutionary*				✓			✓		$\mathcal{O}(n^2)$	5k	?	2006	[9]
	MSN-BD			✓					✓	$\mathcal{O}(n^2ck)$	6k	3M	2006	[10]
	SocDim	✓		✓			✓			$\mathcal{O}(n^2 \log n)*$	80k	6M	2009	[11]
	PMM			✓			✓			$\mathcal{O}(mn^2)$	15k	27M	2009	[12]
	MRGC		✓		✓		✓		✓	$\mathcal{O}(mD)$	40k	?	2007	[13]
	Infinite Relational					✓	✓			$\mathcal{O}(n^{2c}D)$	160	?	2006	[14]
	Find-Tribes				✓				✓	$\mathcal{O}(mnK^2)$	26k	100k	2007	[15]
	AutoPart		✓			✓		✓		$\mathcal{O}(mk^2)$	75k	500k	2004	[16]
	Timefall				✓	✓			✓	$\mathcal{O}(mk)$	7.5M	53M	2008	[17]
	Context-specific Cluster Tree					✓			✓	$\mathcal{O}(mk)$	37k	367k	2008	[18]
Internal Density	Modularity			✓						$\mathcal{O}(mk \log n)$	400k	2.5M	2004	[19]
	Directed modularity		✓	✓						$\mathcal{O}(n^2 \log n)$	50	?	2008	[20]
	External Optimization		✓	✓						$\mathcal{O}(n^2 \log n)$	27k	?	2005	[21]
	Local modularity			✓				✓		$\mathcal{O}(n^2)$	400k	2.5M	2005	[22]
	Modularity Unfolding			✓						$\mathcal{O}(mk)$	118M	1B	2008	[23]
	Multislice modularity		✓	✓	✓		✓		✓	$\mathcal{O}(mkD)$	2k	?	2010	[24]
	MetaFac				✓		✓			$\mathcal{O}(mnD)$	?	2M	2009	[25]
	Variational Bayes		✓			✓				$\mathcal{O}(mk)$	115	613	2008	[26]
	$LA \rightarrow IS^2*$	✓	✓							$\mathcal{O}(mk+n)$	16k	?	2005	[27]
	Local Density		✓			✓		✓		$\mathcal{O}(nK \log n)$	108k	330k	2005	[28]
Bridge	Edge Betweenness		✓							$\mathcal{O}(m^2n)$	271	1k	2002	[4]
	CONGO*	✓								$\mathcal{O}(n \log n)$	30k	116k	2008	[29]
	L-Shell	✓						✓		$\mathcal{O}(n^3)$	77	254	2005	[30]
	Internal-External Degree	✓								$\mathcal{O}(n^2 \log n)$	775k	4.7M	2009	[31]
Diffusion	Label Propagation			✓		✓		✓		$\mathcal{O}(m+n)$	374k	30M	2007	[32]
	Node Colouring				✓				✓	$\mathcal{O}(ntk^2)$	2k	?	2007	[33]
	Kirchhoff	✓		✓						$\mathcal{O}(m+n)$	115	613	2004	[34]
	Communication Dynamic	✓	✓		✓			✓		$\mathcal{O}(mnt)$	160k	530k	2008	[35]
	GuruMine		✓		✓					$\mathcal{O}(TAn^2)$	217k	212k	2008	[8]
	DegreeDiscountIC		✓							$\mathcal{O}(k \log n + m)$	37k	230k	2009	[36]
	MMSB	✓	✓							$\mathcal{O}(nk)$	871	2k	2007	[37]
Close	Walktrap									$\mathcal{O}(mn^2)$	160k	1.8M	2006	[38]
	DOCS	✓								?	325k	1M	2009	[39]
	Infomap		✓	✓						$\mathcal{O}(m \log^2 n)$	6k	6M	2008	[40]
Structure	K-Clique	✓								$\mathcal{O}(m^{\frac{\ln m}{10}})$	20k	127k	2005	[3]
	S-Plexes Enumeration									$\mathcal{O}(mn)$	?	?	2009	[41]
	Bi-Clique	✓							✓	$\mathcal{O}(m^2)$	200k	500k	2008	[42]
	EAGLE	✓	✓	✓						$\mathcal{O}(3^{\frac{n}{3}})$	16k	31k	2009	[43]
Link	Link modularity	✓		✓					✓	$\mathcal{O}(2mk \log n)$	20k	127k	2009	[44]
	Link Jaccard*	✓		✓					✓	$\mathcal{O}(n\bar{K}^2)$	885k	5.5M	2010	[45]
NoD	Hybrid*	✓	✓	✓		✓				$\mathcal{O}(nk\bar{K})$	325k	1.5M	2010	[46]
	Multi-relational Regression			✓			✓			?	?	?	2005	[47]

Table 2: Resume of the community discovery methods.

tion of internal density may find also communities with sparse external links, i.e. bridges. We will see in Section 5 that in this definition a key concept is modularity [19]. Modularity is a quality function which both consider the internal density of a community and the absence of edges among communities. Thus methods based on it could be clustered in both category. However, the underlying definition of modularity has the focus on the internal density, and this is the reason for the proposed classification. To give another example, a diffusion approach may detect the same communities whose member can reach each other with few hops. But this is not necessary: the diffusion approach may find communities also with an arbitrary distance among its members.

In order to have a stronger evidence of this fact consider Figures 2, 4, 7, 9, 12 and 13. In these figures are depicted the most simple typical communitties identified, respectively, by the definitions of Feature Distance, Internal Density, Bride Detection, Diffusion, Closeness e Structure Definition. As one can see, there are a number of differences among these toy examples. The Bridge Detection example (Figure 7) is a random graph, thus with no community structure by definition for the algorithms in the Internal Density category. Also the Diffusion example (Figure 9) is a random graph, but even if the diffusion process identify two communities, no clear bridges can be detected.

The overlap is due to the fact that many algorithms work with some general "background" meta definitions of community. The categories here proposed can be clustered together in a hierarchy with the four main categories exposed in Section 2.2. Further, many algorithms may present common strategies in the exploration of the search space or in evaluating the quality of their partition in order to refine it. Consider for example [51] and [52]. In these two paper there is a deep theoretical study about modularity and its most general form. In [51], for example, authors were able to derive modularity as a random walk exploration strategy, thus highlighting its overlap with the algorithms here clustered in the "Closeness" category.

Evaluating the overlap and the relationships among the most important community discovery approaches is not a trivial task, and it is outside the aim of this survey. Here we are interested in focusing on the connection between an algorithm and its particular definition of community. In this way we can create the "inverted index" useful to connect particular analysis needs to the tools available in literature. To study how to derive one algorithm in terms of another, thus creating a graph of algorithms and not a classification, is an open interesting problems we will leave to future works.

4. Feature Distance

In this Section we will review the community discovery methods that define a community according this meta definition:

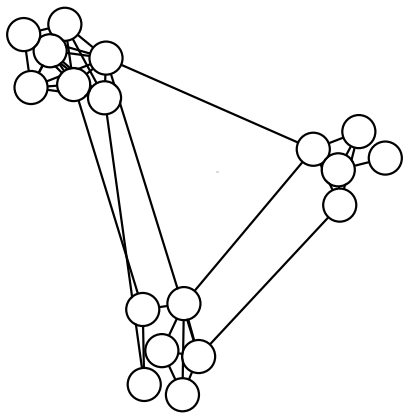


Figure 2: An example of graph which can be partitioned with a notion of "distance" among its nodes.

Meta Definition 2 (Community). *A community in a complex network is a set of entities that share a precise set of features (including the edge as a feature). Defining a distance measure based on the values of the feature, the entities inside a community are very close to each other, more than entities outside the community.*

Using this definition the task of finding communities is very similar to the classical clustering problem in data mining. In data mining, clustering is an unsupervised learning task. Aim of a clustering algorithm is to assign a large set of data into groups (clusters) so that the data in the same clusters are similar each other more than with any other data in any other cluster. Similarity is defined through a distance measure, usually based on the number of common features of the entities, or on similar values of these attributes.

Consider Figure 2. Here we have depicted a network whose nodes are positioned according to a distance measure. It is important to note that this measure could consider the direct edge connection, but it is not mandatory. The nodes are then grouped in the same community if they are close in this space (which may be highly dimensional according to the number of features considered).

An example of clustering technique is K-means [53]. A natural clustering approach to the community discovery are some evolutions of co-clustering [54, 55] and/or some spectral approaches to the clustering problem [56]. A survey focused on co-clustering algorithms is [57].

Here we will focus on some clustering techniques able to provide some very interesting features: the Evolutionary clustering [9]; RSN-BD [10], a k-partite graph based approach; MRGC [13], that is a clustering technique working with tensors; two approaches that use modularity for the detection of latent dimensions for a multidimensional community discovery with a machine learning classifier that maximize the number of common features ([11] and [12]); a Bayesian approach to clustering based on the predictabil-

ity of the features for nodes belonging to the same group [14]; and an analysis of the shared attribute connections in a bipartite graph entity-attribute [15].

An interesting clustering principle is the Minimum Description Length principle [48]. In MDL the main concept is that any regularity in the data (i.e. common features) can be used to compress it, i.e. to describe it using fewer symbols than the number of symbols needed to describe the data literally (see also [58] and [59]). The more regularities there are, the more the data can be compressed. This is a very interesting approach since it allows to perform the community discovery without setting any parameter at all. After considering the classical clustering approaches, in this Section we will also present three main algorithms that implement an MDL community discovery approach: Autopart [16] (that is, to the best of our knowledge, the first popular community discovery that formulates the ground theory for the MDL community detection), the Context-specific cluster tree [18] and Timefall [17].

4.1. Evolutionary [9]*

In [9] authors tackle the classical clustering problem adding the temporal dimension. This novel situation implies some constraints:

- **Consistency**. Any insights derived from a study of previous clusters are more likely to apply to future clusters.
- **Noise Removal**. Historically consistent clustering provides greater robustness against noise by taking previous data points into effect.
- **Smoothing**. The true clusters shift over time.
- **Cluster Correspondence**. It is generally possible to place today's clusters in correspondence with yesterday's clusters, so the user will still be situated within the historical context.

In order to consider these constraints, some measures of a proposed clustering division are defined: the *snapshot quality* and the *history cost*. The snapshot quality of C_t, a proposed cluster division, measures how well C_t represents the data at time-step t. The history cost of the clustering is a measure of the distance between C_t and C_{t-1}, the clustering used during the previous time-step.

This setting can be considered similar, but with some differences, to incremental clustering [60]. The main differences are two. First: here the focus is upon optimizing a new quality measure which incorporates deviation from history. Second, it works on-line (i.e. it must provide a clustering of the data during time-step t before seeing any data for time-step $t+1$) while other frameworks works on data streams [61].

This framework can be added to any clustering algorithm. Particularly on the agglomerative hierarchical clustering, used for the examples in the original paper, the time complexity will be $\mathcal{O}(n^2)$, although some authors claim that it is possible a quasi-linear implementation [62]. However, this framework is presented here because it is possible to apply its principles to any other community discovery algorithm presented in this survey. In particular there are two framework application worth noting. The first one is FacetNet [63], in which a framework for evaluating the evolution of the communities is developed. The second one is [64]. In this work are introduced the concepts of nano-communities an k-clique-by-clique, useful to evaluate the snapshots and historical quality of the communities identified in various snapshots with any given method.

4.2. RSN-BD [10]

RSN-BD (acronym for Relation Summary Network with Bregman Divergence) is a community discovery approach focused on that examples of real-world data that involve objects of multiple types that are related to each other. A natural representation of this setting is a k-partite graph of heterogeneous types of nodes. This method is suitable for general k-partite graph and not only some special cases such as [65], that has the restriction that the numbers of clusters for different types of nodes must be equal and the clusters for different types of objects must have one-to-one associations.

The key idea of Relation Summary Network is to add a small number of *hidden nodes* to the original k-partite graph to make the hidden structures of the graph explicit. RSN must be as close as possible to the original graph. To express this closeness a distance function $\mathfrak{D}$ is given: basically every original node can be linked only with one hidden node and two hidden nodes are linked only if their original node were linked. Then the distance function sum up all the Euclidean distances between the weights of the edges in the original graph and in the transformed graph. It is also possible to choose as distance function any definition that can be generalized as a Bregman divergence [66].

Authors expect that in a sparse k-partite graph two nodes are similar when they are connected to similar nodes even though they are not connected to the same nodes.

Since RSN problem is $\mathcal{NP}$-Hard, authors formulates some approximations. They represent a k-partite graph as a set of matrices. Then, the distance between the two graphs (original and manipulated) can be formulated as the distances between a set of matrices and a set of matrix products: the distance between two matrices $D(X, Y)$ denotes the sum of the distances of each pair of elements, i.e., $D(X, Y) = \sum_{h,l} D(X_{hl}, Y_{hl})$. The proposed algorithm updates each element of the matrix that represents the graph and this updating involves only edges between v_{ih} and the related nodes, not all the edges.

4.3. MRGC [13]

In this model, each relation between a given set of entity classes is represented as a multi-dimensional tensor (or data cube) over an appropriate domain, with the dimensions being associated with the various entity classes. Further, each cell in the tensor encodes the relation between a particular set of entities and can either take real values, i.e., the relation has single attribute, or itself is a vector of attributes.

Although defined for relations graphs, this model can be used also for identify community structures in social networks.

It is based on the minimum Bregman information principle [67], which generalizes both least squares and maximum entropy principles, to the relation graph setting. The result is a general framework in which previously coclustering approach, such as [68] can be viewed as special cases.

MRGC stands for "Multi-way Relation Graphs Clustering". The multi-way clustering ρ is defined as a n-uple $(\rho_1, \rho_2, ..., \rho_n)$ where each $\rho_i : \{1, 2, ..., m_i\} \rightarrow \{1, 2, ..., k_i\}$ denotes a mapping from the entities to their respective clusters (m_i is the cardinality of the entities and k_i the desired number of clusters). The quality of the multi-way clustering ρ can be readily measured in terms of the approximation error or the expected Bregman distortion between the random variables Z (the original tensor) and $\hat{Z}$ (the approximate tensor that follows ω, the measure inducted on Z). The multi-way clustering problem is then to find the optimal ρ that minimizes the Bregman distortion. To characterize $\hat{Z}$, one needs to specify what summary statistics are to be preserved, and how to get an approximation based on the summary statistics.

The MBI principle [67] posits that the best approximation $\hat{Z}$ is the random variable that has the minimum Bregman information, i.e. the "best" approximation given certain information is one that does not make any extra assumptions over the available information. In general, the solution cannot be expressed in closed form as a function of the summary statistics, except for certain special bases (Block Multi-way Clustering and Bias-Adjusted Multi-way Clustering).

The proposed algorithm is an alternate minimization scheme for optimizing the multi-way clustering objective function. The algorithm considers each dimension in turn, finds the optimal clustering with respect to that dimension keeping everything else fixed, and recomputes the MBI solution, and this process is repeated till convergence.

4.4. SocDim [11]

SocDim is a relational learning framework whose aim is to build a classifier of the interacting entities based on latent social dimensions. Each dimension can be considered as the description of a likely affiliation between social actors. These affiliations have to be considered when building the classifier, thus going beyond the Markov assumption that the label of a node is only dependent on the labels of all its neighbors. But affiliations are usually not recorded in public available data, so this methods aims to identify them even if they are latent.

SocDim is based on the concept of homophily [69]: actors sharing certain properties tend to form groups and within-group interactions are more dense than interactions with members outside the group.

SocDim has two steps. First: it extracts latent social dimensions based on network connectivity: it uses modularity (Section 5) in order to find in the structure of the network the dimensions in which the nodes are placed. Second, it constructs a discriminative classifier: the extracted social dimensions are considered as normal features (including some possible other sources) in the classical supervised learning task. It is then possible to use the predicted labels of the classifier in order to reconstruct the community organization of the entities.

This is a multidimensional community discovery because the classifier will determine which dimensions are relevant to a class label. Authors considered one-vs-rest linear SVM due to its simplicity and scalability [70] and more powerful methods like structural SVM [71].

This work is at the basis of a further evolution [72] that has an edge-centric view of the communities. The partition is not about nodes, but about edges. In this way it is possible to have a more scalable clustering procedure.

4.5. PMM [12]

This work, presented in [73] and evolved in [12], presents a variation of the modularity approach on a multidimensional setting. The goal of PMM (the acronym stands for "Principal modularity Maximization") algorithm is to infer the shared latent community structure among the actors given a multi-dimensional network, taking into account all the dimension directly instead of applying the modularity algorithm on one dimension at time. Since the modularity is computed with Lanczos method [74], the initial time complexity is at least $\mathcal{O}(mn^2)$.

There are two basic steps. The first is "Structural Feature Extraction": for a multidimensional network, authors extract social features from each dimension of the network, i.e. those eigenvectors with a larger positive eigenvalue. The second is "Cross-Dimension Integration" that starts from the assumption that the structural features of different dimensions are highly correlated after transformation. To capture the correlations between multiple sets of variables authors use (generalized) canonical correlation analysis [75] (CCA attempts to find a transformation for each set of variables such that the pairwise correlations are maximized).

After this step, authors obtain the lower-dimensional embedding which captures the principal pattern across all the dimensions of the network. Then they can perform k-means [53] on this embedding to find out the discrete community assignment.

4.6. Infinite Relational [14]

Suppose that are given one or more relations (i.e. edges) involving one or more types (i.e. nodes). The goal of the Infinite Relational Model is to partition each type into clusters (i.e. communities), where a good set of partitions allows relationships between entities to be predicted by their cluster assignments. Authors goal is to organize the entities into clusters that relate to each other in predictable ways, by simultaneously clustering the entities and the relations.

Formally, suppose that the observed data are m relations involving n types. Let R^i be the ith relation, T^j be the jth type, and z^j be a vector of cluster assignments for T^j. The task is to infer the cluster assignments, and the ultimate interest lays in the posterior distribution $P(z_1, ..., z_n \mid R_1, ..., R_m)$.

To allow the IRM the ability to discover the number of clusters in type T, authors use a prior [76] that assigns some probability mass to all possible partitions of the type.

Consider an m dimensional relation R involving n different types. Let d_k be the label of the type that occupies dimension k: for example, the three place relation $R : T^1 \times T^1 \times T^2 \to \{0, 1\}$ has $d1 = d2 = 1$, and $d3 = 2$. The probability that the relation holds between a group of entities depends only on the clusters of those entities: $R(i_1, ..., i_m) \mid z^1, ..., z^n, \eta \backsim Bernoulli(\eta(z_{i_1}^{d_1}, ..., z_{i_m}^{d_m}))$. In settings with multiple relations, authors introduce a parameter matrix η_i for each relation R_i. The parameter $\eta(a, b)$ specifies the probability that a link exists between any given pair (i, j) where i belongs to cluster a and j belongs to cluster b.

Inference can be carried out using Markov chain Monte Carlo methods to sample from the posterior on cluster assignments.

4.7. Find-Tribes [15]

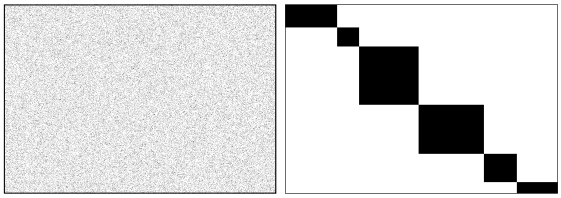
Find-Tribes is an algorithm not explicitly developed for community discovery purposes. However, the technique can still be used in order to identify some kind of community. In particular, it is very close to our "action" definition of a community: the entities in a group tend to behave in the same way.

As input, authors require a bipartite graph $G = (R \cup A, E)$ of entities R and attributes A. The entities should connect to at least several attributes. Aim of the algorithm is to return those group sharing "unusual" combinations of attributes. This restriction can be easily generalized in order to obtain as output also the "usual" groups.

The strategy for the desired task revolves around developing a good definition of "unusual". For an entity group to be considered anomalous, the shared attributes themselves need not be unusual, but the particular configuration of them should be.

The algorithm begins with listing all node pairs. Authors then summarize the direct entity relationships F, keeping track of the attributes where they coincide. Additional informations, such as dates, are noted. The result of this step is a new, non bipartite, graph $H'(R, F)$. The algorithm proceeds by identifying all significant pairs in H'. For each edge a score c_{ij} is computed, measuring how significant or unusual its sequence of shared attributes is. Once the significance scores are computed, authors pick a threshold d for the scores and remove all edges f_{ij} for which $c_{ij} < d$. Then, authors compute the connected components of H', which are the desired tribes.

Authors propose a number of possible scores: the number of attributes in the shared sequence, the number of time steps of overlap, a probabilistic Markov chain of attributes and so on.



(a) The original matrix (b) Reordered matrix

Figure 3: An example of MDL principle for matrices: the matrix at the left is exactly the same matrix at the right, but reordered according to its simplest way to be described.

4.8. AutoPart [16]

In Autopart we have the basic formulation of the MDL approach to the community discovery problem. We have a binary $n \times n$ matrix that represents associations between the n nodes of the graph (corresponding to rows and columns in the adjacency matrix). In this binary matrix there is a 1 at position (i, j) if node i is directly connected to node j. An example of a possible adjacency matrix is depicted in Figure 3(a).

The main idea is to reorder the adjacency matrix so that similar nodes, i.e. nodes that are connected to the same set of nodes, are grouped with each other. Then the adjacency matrix would consist of homogeneous rectangular/square blocks of high (low) density, representing the fact that certain node groups have more (less) connections with other groups. The final output of the proposed procedure should be similar to the reordered matrix in Figure 3(b). Clearly the reordered matrix can be encoded with a great compression of the data, especially when the node blocks are very homogeneous.

Following this procedure one must identify a trade-off point that indicates the best number of groups k. For the identification of this trade-off authors rely on the application of the overall MDL philosophy, that enables a parameter free mining that is used in many applications [77], where the compression costs are based on the number of bits required to transmit both the "summary" of the node groups, as well as each block given the groups. Therefore, a total encoding cost function is defined, and

aim of the algorithm is to identify the best grouping that minimize this function.

Authors solved this problem by a two-step iterative process: first, they find a good node grouping G for a given number of node groups k that minimize entropy; and second, they search for the number of node groups k by splitting the previous identified groups and verifying if there is a possible gain in the total encoding cost function.

4.9. Context-specific Cluster Tree [18]

In this variant of the MDL approach the binary $n_s \times n_d$ matrix represents a bipartite graph with n_s source nodes and n_d destination nodes. This is a hierarchical evolution of the existing flat method described in [55]. Please note that here hierarchy will not be considered an overlapping setting because it is only a particular overlapping case in which at each level of the hierarchy the identified clusters are completely disjoint.

In this work, aim of the authors is to automatically construct a recursive community structure of a large bipartite graph at multiple levels, namely, a Context-specific Cluster Tree (CCT). The resulting CCT can identify relevant context-specific clusters. The main idea is to subdivide the adjacency matrix in tiles, or "contexts", with possible reordering of rows and columns, and compress them, either as-is (if they are homogeneous enough) or by further subdividing.

The process is the following. The entire graph is considered as a whole community. If the best representation of the considered (sub)graph is the random graph then the community cannot be split into two sub-community, because by definition the random graph has no community structure at all. Otherwise, if the random graph cannot describe the considered community, then this graph is split and the algorithm is reapplied in a recursive fashion. In order to verify if a community is almost random, its representing graph is compressed using a total encoding cost function.

The result is a tree of communities in which the bottom levels are context specialization of the generic communities at the top of the tree. For instance in a co-authorship network will identify at the top level the computer science and sociology communities of authors, and at the lower level inside computer science community may identify algorithm and computer theory communities.

This idea of recursive clustering is also applied to streaming setting [78, 79], although with a number of parameters.

4.10. Timefall [17]

Timefall is a MDL approach that can be described as a parameter-free network evolution tracking. Given n time-stamped events (like, e.g., papers published), each related to several of m items (like, title-words), it finds simultaneously (a) the communities, that is, item-groups (e.g., research topics and/or research communities) and (b) a description of how the communities evolve over time (e.g., appear, disappear, split, merge) and (c) a selection of the appropriate cut-points in time when existing community structure change abruptly.

With this approach, authors first represent the graph in the form of an adjacency matrix (step 1). The algorithm then splits the rows according to their time stamps (step 2) and uses the Cross Association algorithm [55], a MDL based algorithm similar to the previously presented, to cluster the columns of the connection matrices (step 3). It then utilizes the MDL principle to connect the column clusters of the matrices (step 4): if two column clusters can be encoded together with a low encoding cost then they are connected. Finally it reduces the unimportant time points in the graph evolution history (step 5).

The differences in clustering of two matrices can be efficiently described by using mutual information between the matrices, i.e., authors use the clusters (communities) at time t to efficiently describe the communities at time $t + 1$.

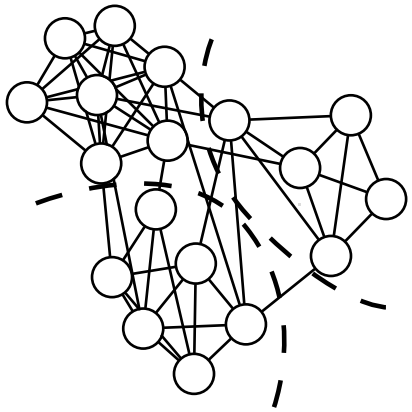


Figure 4: An example of graph which can be partitioned with a notion of internal density among its nodes.

5. Internal Density

For this group of approaches, the underlying meta definition is the following:

Meta Definition 3 (Community). *A community in a complex network is a set of entities that are densely connected. In order to be densely connected a group of vertices must have a number of edges significantly higher than the expected number of edges in a random graph (which has no community structure).*

In Figure 4 we have depicted a network in which the identified communities are significantly denser than a random graph with the same degree distribution.

A key concept for satisfying this meta definition is modularity [80]. Briefly, consider dividing the graph into c non-overlapping communities. Let c_i denote the community membership of vertex v_i , k_i represents the degree

of vertex i. Modularity is like a statistical test in which the null model is a uniform random graph model: in this model one entity connects to others with uniform probability. For two nodes with degree k_i and k_j respectively, the expected number of edges between the two in a uniform random graph model is $\frac{k_i k_j}{2m}$, where m is the number of edges in the graph. Modularity measures how far the interaction deviates from a uniform random graph with the same degree distribution. It is defined as:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

where $\delta(c_i, c_j) = 1$ if $c_i = c_j$ (i.e. the two nodes are in the same community), and 0 otherwise, and A_{ij} is the number of edges between nodes i and j. A larger modularity indicates denser within-group interaction. Note that Q could be negative if the vertices are split into bad clusters. $Q > 0$ indicates the clustering captures some degree of community structure. In general, one aims to find a community structure such that Q is maximized.

Modularity is involved in the community discovery problems at two level. First: it is a measure able to quantify how good is a given network partition. It is a very good quantitative evaluation of the performance of a community discovery algorithm because it can be applied to any kind of graph (single graph, multi graph, weighted graph and so on). Furthermore, it gives a result of the quality of the partition even without any knowledge about the actual communities of the network. This is suitable especially for very large networks. On the other hand, modularity is not the perfect solution for evaluating a proposed community partition. It suffers of well known problems, in particular the resolution problem: modularity fails to identify communities smaller than a scale which depends on the total size of the network and on the degree of interconnectedness of the communities, even in cases where modules are unambiguously defined. Further, with modularity you can only evaluate communities extracted according to the meta definition proposed in this section. Any other kind of definition of communities will result in meaningless evaluations by applying modularity. For an extensive review of the known problems of modularity please see [6] and [81].

The second level of the modularity usage in the graph partitioning task is represented by community discovery algorithms that are based on modularity maximization. These algorithms suffers of the aforementioned problems of the usage of modularity as quality measure. However, modularity maximization is a very prolific field of research, and there are many algorithms relying on some heuristics and strategies for finding the best network partition.

We will now present some of the main examples of modularity-based approaches, whose number is intractable in this survey (we will, however, provide also references for minor modularity maximization algorithms. A good review of the eigenvector modularity based works is presented in [82]). We will focus particularly on: the first classical efficient modularity approach by Clauset et al. [19], an extension of modularity for directed networks [20] and some heuristic strategies developed to tackle the $\mathcal{NP}$-completeness characteristics of the modularity maximization problem (the external optimization [21], the local modularity computation [22] and the very efficient modularity unfolding [23]). A very recent work [24] extends modularity in order to be able to extract multidimensional communities.

Modularity is not the only cost function able to quantify if a set of entities are more related than expected and thus has to be considered a community. The other reviewed methods that rely on different techniques, but share the same meta definition of community proposed in this Section, are: MetaFac [25], an hypergraph factorization technique; a physical-chemical algorithm using a Bayesian approach [26]; a local density-based approach called $LA \rightarrow IS^2$ [27]; and another proposed function used to measure the internal local density of a cluster [28].

5.1. Classical modularity [19]

This is the first efficient algorithm that is based on the modularity maximization strategy. To find the partition providing the maximum value of modularity is an NP-complete problem. Therefore, many heuristics has been proposed. The presented "Classical modularity" is an advanced implementation of the very first proposed heuristic, presented in [83].

In the previous work, Newman proposed an efficient strategy for modularity maximization. The algorithm proposed uses a greedy optimization in which, starting with each vertex being the sole member of a community of one, repeatedly join together the two communities whose amalgamation produces the largest increase in Q. For a network of n vertices, after $n - 1$ such joins the result is a single community and the algorithm stops. The entire process can be represented as a tree whose leaves are the vertices of the original network and whose internal nodes correspond to the joins. This dendrogram represents a hierarchical decomposition of the network into communities at all levels. Since a partition grouping all vertices in a single community provides a modularity equal to 0, the dendrogram must be cut in the modularity peak in order to obtain the communities, as depicted in Figure 5. Please note that Figure 5 also depicts another problem of modularity maximization heuristics: it has been discovered that modularity has not a single peak given all the possible partitions, but there are several local optima, thus making more complex the task of finding the actual partition with the largest modularity value. Moreover, real networks have many near-global-optima at various places [81] (in Figure 5 the rightmost peak) and we cannot know where the algorithm locates its solution.

In the first algorithm this operation is done explicitly on the entire matrix. Here, the idea is to store a matrix

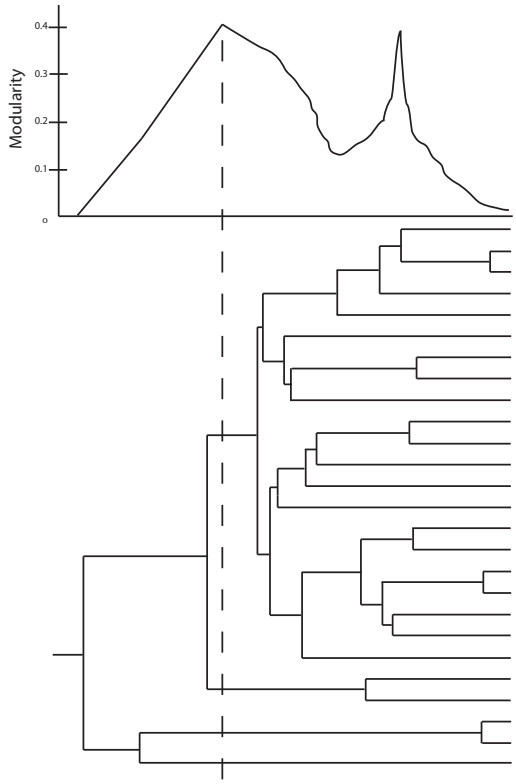


Figure 5: A dendrogram result for the modularity maximization algorithm, with a plot of resulting values of modularity given the partition.

containing only the $\Delta Q_{i,j}$ values of the communities, i.e. the modularity change when joining the communities i and j. The algorithm can now be defined as follows. Calculate the initial values of $\Delta Q_{i,j}$ and keep track of the largest element of each row of the matrix ΔQ. Select the largest $\Delta Q_{i,j}$ among these largest elements, join the corresponding communities, update the matrix ΔQ and the collection of the largest elements and increment Q by $\Delta Q_{i,j}$. Repeat this last step until the dendrogram is complete.

This was, at the time of publication, the quickest of modularity-based methods.

5.2. Directed modularity [20]

This modularity maximization approach is adapted in the case of a directed network. Therefore, in this case we have a matrix representation of the graph, but the matrix is not symmetric.

The algorithm is based on [84]. Authors consider first the simplified problem of dividing a directed network into just two communities, with a restored symmetric adjacency matrix. The basic idea is to calculate the eigenvector corresponding to the largest positive eigenvalue of the symmetrized matrix and then assign communities based on the signs of the elements of the eigenvector.

This spectral method typically provides an excellent guide to the broad outlines of the optimal partition, but may err in the case of individual vertices, a situation that can be remedied by adding a "fine-tuning" stage to the algorithm in which the vertices are moved back and forth between communities in an effort to increase the modularity, until no further improvements can be made.

There are a variety of ways of generalizing the approach to more than two communities but the simplest, adopted by the authors, is repeated bisection. Further, more recent works point to apply also the modularity approach to directed and overlapping communities [85].

5.3. External Optimization [21]

This algorithm represents an evolution and optimization of modularity-based approaches. Is basically a divisive algorithm that optimizes the modularity Q using a heuristic search. This search is based on a measure that depends on the node degree, and its normalization involve all the links in the network after summation. This measure is called λ_i, and it is related to the contribution of individual nodes i to the final computation of the global modularity value (for more details see [21]).

The node selected, in original EO algorithm [86] is always the node with the worst λ_i-value. There is a τ-EO version [87] that is less sensitive to different initializations and allows escape from local maxima (for what is a local maximum in modularity maximization, consider again the rightmost peak in Figure 5).

A number of other optimization strategies has been proposed. Most efficient is the size reduction [88], that works better than simulated annealing [89].

5.4. Local modularity [22]

Intuitively, in order to obtain a local modularity evaluation, there should be no need to know nothing about the global structure of the network. The local version of modularity was developed in order to tackle its high complexity in maximization for huge networks.

Authors define a core of vertices and they know everything about this core $\mathcal{C}$. Around $\mathcal{C}$ there is a boundary zone $\mathcal{B}$. In $\mathcal{B}$ we have all the nodes adjacent to the ones present in $\mathcal{C}$, and we know only the edges that are directly connected to $\mathcal{C}$. We don't know nothing about the other edges that connect the boundary to the remaining part of the network, unexplored $\mathcal{U}$ ($\mathcal{C} \cup \mathcal{B}$ is a sort of ego-network [90] of $\mathcal{C}$).

The local modularity measure $Q_{\mathcal{C}}$ (generalizable for weighted networks) takes into account the number of edges with one or more endpoints in $\mathcal{B}$ on the number of those edges with neither endpoint in $\mathcal{U}$.

Initially, the algorithm places the source vertex in the community, $v_0 = \mathcal{C}$, and places its neighbors in $\mathcal{U}$. At each step, the algorithm adds to $\mathcal{C}$ (and to $\mathcal{B}$, if necessary) the neighboring vertex that results in the largest increase (or smallest decrease) in $Q_{\mathcal{C}}$, breaking ties randomly. Finally, it adds to $\mathcal{U}$ any newly discovered vertices, and updates its estimate of $Q_{\mathcal{C}}$. This process continues until it has agglomerated either a given number of vertices k, or it

has discovered the entire enclosing component, whichever happens first.

5.5. Modularity Unfolding [23]

This algorithm was developed to be applicable to huge graphs. The previous largest graph used for modularity testing was 5.5 millions nodes [91], with this improvement it is possible to scale up to 100 millions nodes.

The algorithm is divided in two phases that are repeated iteratively. It starts with a weighted network of n nodes. It assigns a different community to each node of the network. Then, for each node i authors consider the neighbors J of i and evaluate the gain of modularity that would take place by removing i from its community and by placing it in the community of J. The node i is then placed in the community for which this gain is maximum, but only if this gain is positive. If no positive gain is possible (or if it is not greater than a given threshold, used to reduce the running time), i stays in its original community. This first phase stops when a local maximum of the modularity is attained, i.e. when no individual move can improve the modularity. The output of the algorithm depends on the order in which the nodes are considered.

The second phase of the algorithm consists in building a new network whose nodes are now the communities found during the first phase. To do so, the weights of the links between the new nodes are given by the sum of the weight of the links between nodes in the corresponding two communities. Once this second phase is completed, it is then possible to reapply the first phase of the algorithm to the resulting weighted network and to iterate.

This method has been tested on the Uk-Union WebGraph [92], and it has been used also for co-citation networks [93].

5.6. Multislice modularity [24]

The multislice modularity framework was developed starting from the assumption that none of the available community-detection methods is directly applicable to dynamic and/or multidimensional networks. In these cases the analyst was forced to rely on detecting communities separately at each time and then to use ad hoc methods or special constraints to piece together the structures obtained at different times/dimensions (and this is consistent with this survey, please check Table 2).

In order to generalize modularity for multidimensional and dynamic networks, authors extend the null model of modularity (the random graph) to this novel setting. In order to do so, they use several generalizations, like an additional parameter that controls the coupling between dimensions, basing their operation on the equivalence between modularity-like quality functions and Laplacian dynamics of populations of random walkers [51]. In that work a quality function equivalent to modularity was derived for unipartite, undirected and monodimensional networks. In the multislice formulation authors extend the work of Lambiotte et al. in three directions.

First, they restrict the independent joint probability contribution to one that is conditional on the type of connection (dimension or temporal link to pass from a network snapshot to another) necessary to step between two nodes. This yields to some known null model generalizations for bipartite networks [94]. Second, they generalize the Laplacian dynamics to include motion along multiple types of connections. Also in this case, the generalization of the Laplacian dynamics recovers, via a similar derivation, a recently-developed null model for signed networks [95]. Third, they interpret the Laplacian dynamics flexibly to permit different spreading weights on the different dimensions and temporal snapshots. Also in this third case there is another different equivalent generalization [96].

In order to represent both snapshots and dimensions of the network, authors use the concept of slice. Each slice s of a network is represented by adjacencies A_{ijs} between nodes i and j. Authors specify also inter-slice couplings C_{jrs} that connect node j in slice r to itself in slice s. They notate the strengths of each node individually in each slice, so that $k_{js} = \sum_i A_{ijs}$ and $c_{js} = \sum_r C_{jsr}$, and define the multislice strength $\kappa_{js} = k_{js} + c_{js}$. Authors then specify the associated multislice null model using the probability $\rho(is|jr)$ of sampling node-slice is conditional on whether the multislice structure allows one to step from node-slice jr to node-slice is (considering intra- and inter-slice steps separately). The resulting multislice extended definition of modularity is the following:

$$Q = \frac{1}{2\mu} \sum_{ijsr} \left\{ \left(A_{ijs} - \gamma_s \frac{k_{is} k_{js}}{2m_s} \delta_{sr} \right) + \delta_{ij} C_{jsr} \right\} \delta(c_{is}, c_{jr}).$$

In this extension γ_s is the resolution parameter, that may be or may be not different for each slice. If $\forall s \gamma_s = 1$ then this formula degenerates on the usual interpretation of modularity as a count of the total weight of intra-slice edges minus the weight expected at random. Otherwise the inter-slice coupling C_{jsr} are considered. C_{jsr} takes values from 0 to ∞. If $C_{jsr} = 0$ we degenerate again in the usual modularity definition. Otherwise the quality-optimizing partitions force the community assignment of a node to remain the same across all slices in which that node appears, and the multislice quality reduces to that of an adjacency matrix summed over the contributions from the individual slices with a null model that respects the degree distributions of the individual contributions. The generality of this framework also allows one to include different weights across the C_{jsr} couplings.

After defined the new quality function, the algorithm needed to extract communities can be one of the many modularity based algorithm, thus using the modularity unfolding technique [23] the complexity drops to $\mathcal{O}(mk)$ per dimension.

5.7. MetaFac [25]

In this work, authors introduce the concept of *metagraph*. The metagraph is a relational hypergraph for representing multi-relational and multi-dimensional social data.

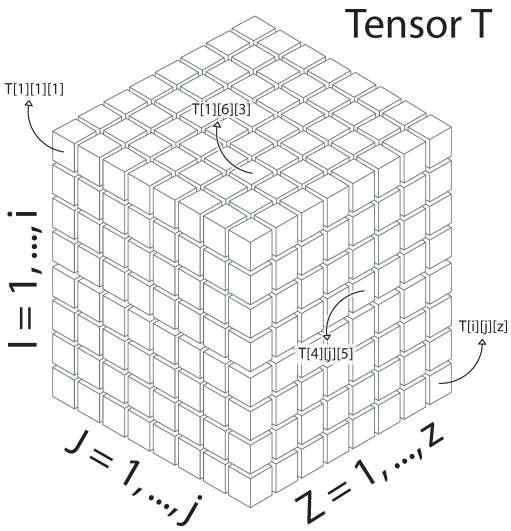


Figure 6: A third-order tensor.

First authors begin to define the elements of the metagraph. In this model, a set of entities of the same type is called a *facet*. The interaction among (two or more) facets is called a *relation*.

An hypergraph is a well know graph generalization. In this representation the edges, called hyperedges, connect any number of vertices. The idea of the authors is to use an M-way hyperedge to represent the interactions of M facets: each facet as a vertex and each relation as an hyperedge on an hypergraph. A *metagraph* defines a particular structure of interactions among facets (groups of entities of the same type), not among facet elements (the entities themselves).

The goal of the authors is to discover latent community structures in the *metagraph* that represent the context of user actions in social media networks. They are interested in clusters of people who interact with each other in a coherent manner.

In order to do so, the metagraph is defined as a set of data tensors. A tensor is an array with N dimensions. In these N dimensions we can identify the Nth-order tensor as an element of the tensor product of N vector spaces, each of which has its own coordinate system. This is a mathematical and computer science definition of tensors, for the tensor notion in physics and engineering please refer to [97]. A "third-order" tensor has three indexes, as shown in Figure 6; a "first-order" tensor is a vector; a "second-order" tensor is a matrix, and tensors of order three or higher are called higher-order tensors. For an extensive review about tensors, tensor decomposition and their applications and tools please refer to [98] (in this work there are also provided some examples of possible applications of tensor decomposition: signal processing [99], numerical linear algebra [100] and, more close to our area of interest, data mining [101, 102], graph analysis tasks [103, 104] and recommendation systems [105]).

Given the metagraph and its defined data tensors, authors want to find a non negative core tensor and factors for a corresponding set of facets. The tensor decomposition and factorization is a very hard task with a known number of issues. To the best of our knowledge, only recently some memory and time efficient techniques have been developed, such as [106]. In the metagraph approach the tensor decomposition can also be viewed as a dynamic analysis, when the set of tensors are temporally annotated and the resulting core tensor refers to a specific time-step t. This is called metagraph factorization (for time evolving data).

Finally, the MF problem can be stated in terms of optimization, i.e. minimizing a given cost function, thus obtaining facet communities.

5.8. Variational Bayes [26]

In this work, authors specify an $N - node$ network by its adjacency matrix A and define $\sigma_i \in \{1, ..., K\}$ to be the unobserved module membership of the i^{th} node.

Authors define a joint probability by considering the number of edges present and absent within and among the K communities of a network. Traditional methods [107] needs to specify K, this one is parameter free, using particular distributions on the vectors.

Given the adjacency matrix, authors determine the most probable number of modules (i.e. occupied spin states) $K = argmax_K p(K|A)$ and infer posterior distributions over the model parameters (i.e. coupling constants and chemical potentials) $p(\pi, \theta|A)$ and the latent module assignments (i.e. spin states) $p(\sigma|A)$. The problem to assign each node to a module in the network is tackled by solving the disorder-averaged partition function of a spin-glass, calculated by marginalizing over the possible quenched values of the parameters of the system.

The computationally intensive solution is tackled using the variational Bayes approach [108].

To minimize the given confusion parameter the algorithm needs multiple random-chosen initialization of the parameters.

*5.9. $LA \rightarrow IS^2$ * [27]*

In this work, authors adopt this definition of community: a group C of actors in a social network forms a community if its communication density function achieves a local maximum in the collection of groups that are close to C [109]. Basically, a group is a community if adding any new member to, or removing any current member from, the group decreases the average of the communication exchanges.

This work is an evolution of [110]. It is build on two distinct phases: Link Aggregate (LA) and the real core of the community detection (IS^2). Authors need a two steps approach because the IS^2 algorithm performs well at discovering communities given a good initial guess, for example when its initial guesses are the outputs of another clustering algorithm, in this case called Link Aggregate (LA).

In LA, the nodes are ordered according to some criterion, for example decreasing Page Rank [111], and then processed sequentially according to this ordering. A node is added to any cluster if adding it improves the cluster density. If the node is not added to any cluster, it creates a new cluster. The complexity of this stage is $\mathcal{O}(mk+n)$.

IS^2 explicitly constructs a cluster that is a local maximum w.r.t. a density metric by starting at a seed candidate cluster and updating it by adding or deleting one node at a time as long as the metric strictly improves. The algorithm stops when no further improvement can be obtained with a single change. The nodes capable of increasing the cluster density are the members of the cluster itself (which could be removed) or members of the cluster immediate neighborhood, defined by those nodes adjacent to a node inside the cluster.

As one can see, the IS^2 algorithm can be applied to the results of any other clustering technique, thus making this approach a general framework useful to improve some incomplete results.

5.10. Local Density [28]

In order to compute the "Local Density", authors define the internal degree of a cluster C to be the number of edges connecting vertices in C to each other, $deg_{int}(\mathcal{C}) = |\{(u,v) \in E | u,v \in \mathcal{C}\}|$. Thus it is possible to define the local density of cluster as

$$\delta_l(\mathcal{C}) = \frac{2deg_{int}(\mathcal{C})}{|\mathcal{C}|(|\mathcal{C}|-1)}.$$

Optimizing $\delta \in [0,1]$ alone makes small cliques superior to larger but slightly sparser sub-graphs, which is often impractical. For clusters to have only a few connections to the rest of the graph, one may optimize the relative density

$$\delta_r(\mathcal{C}) = \frac{deg_{int}(\mathcal{C})}{deg_{int}(\mathcal{C}) + deg_{ext}(\mathcal{C})},$$

where $deg_{ext}(\mathcal{C}) = |\{(u,v) \in E | u \in \mathcal{C}, v \in V \setminus C\}|$. The final quality measure used is $f(\mathcal{C}) = \delta_l(\mathcal{C})\delta_r(\mathcal{C})$.

A good approximation of the optimal cluster for a given vertex can be obtained by local search. To locate a cluster containing a given vertex $v \in V$ from a graph $G = (V, E)$, authors stochastically examine subsets of V containing v, and choose the candidate with maximal f as $C(v)$. The initial cluster $C(v)$ of a vertex v contains v itself and all vertices adjacent to v. Each search step may either add a new vertex that is adjacent to an already included vertex, or remove an included vertex.

Authors guide the local search with simulated annealing [112].

6. Bridge Detection

The meta definition of community for the algorithms in this Section is the following:

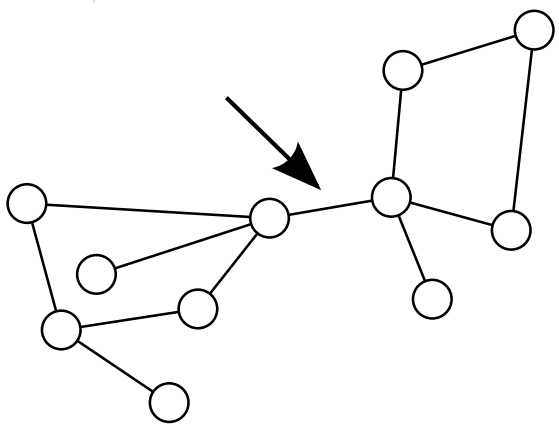


Figure 7: An example of graph which can be partitioned identifying a "bridge".

Meta Definition 4 (Community). *A community in a complex network is a component of the network obtained by removing from the structure all the sparse bridges that connects the dense parts of the network.*

The bridge identified by the arrow in Figure 7 is a perfect example of edge to be removed in order to decompose the network into disconnected components which represent our communities.

The main focus for these approaches is how to find these bridges (that can be both nodes or edges) inside the network. The most popular approach in this category is to use a centrality measure.

In social network analysis a centrality measure is a metric defined in order to obtain a quantitative evaluation of the structural power of an entity in a network [113]. An entity does not have power in the abstract, it has power because it can dominate others: ego's power is alter's dependence. There are a number of measures defined to capture the power of an entity in a network. Some of them are: **Degree centrality**, actors who have more ties to other actors may be advantaged positions; **Closeness centrality**, the more close an entity is to any other entity in the network, the more power it has; **Betweenness centrality**, the most important entity in the network is the entity present in the majority of the shortest paths among all other entities.

Here we will focus on two methods based on an edge definition of the traditional node betweenness centrality: the very first edge betweenness community discovery algorithm [4], that recently has been object of further evolutions, and a general approach that uses the split betweenness concept in order to obtain an overlapping community discovery framework [29]. We then consider also two alternative methods [30, 31] that try to detect the bridges by expanding the community structure and computing a community fitness function.

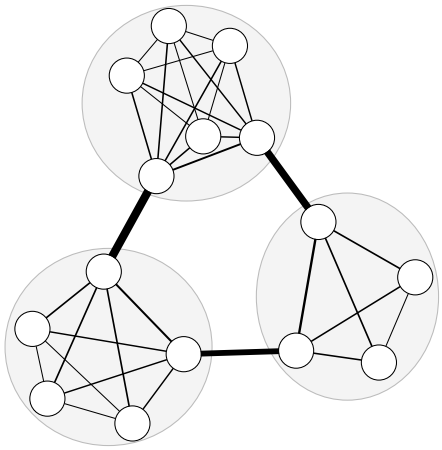


Figure 8: An intuitive example of bridge detection approach. In this graph the edge width is proportional to the edge betweenness value. Wider edges are more likely to be a bridge among communities.

6.1. Edge Betweenness [4]

This is one of the first community discovery algorithms developed after the renew interest in social network analysis started at the end of the 90s. Previously the traditional graph partitioning approach was constructing communities by adding the strongest edges to an initially empty vertex set (such as in hierarchical clustering [114]). Here, authors construct communities by progressively removing edges from the original graph.

In order to find which edges in a network are most between other pairs of vertices, authors generalize Freeman's betweenness centrality [115] to edges and define the "edge betweenness" of an edge as the number of shortest paths between pairs of vertices that run along it.

If a network contains communities or groups that are only loosely connected by a few inter-group edges, then all shortest paths between different communities must go along one of these few edges. Thus, the edges connecting communities will have high edge betweenness. In Figure 8 we have depicted an example. In this Figure, the size of the edges is proportional to their edge betweenness. As one can notice, the higher edge betweenness values are taken by the edges between communities. By removing these edges, it is possible to separate a group from one another and so reveal the underlying community structure of the graph

While the classical implementation of the edge betweenness algorithm is $\mathcal{O}(mn)$, recently has been proposed a speed-up for parallel systems that is linear [116]. Thus without the parallel algorithm the worst case time complexity is $\mathcal{O}(m^2n)$.

A slight variation of this method instead of edge betweenness centrality uses a local measure called *edge clustering coefficient* as a criterion for removing edges [117], or other definitions [118]. The edge clustering coefficient is defined as the fraction of number of triangles a given edge participates in, to the total number of possible such triangles. The clustering coefficient of an edge is expected to be the least for those running between communities and hence the algorithm proceeds by removing edges with low clustering coefficients. The total running time of this divisive algorithm is $\mathcal{O}(\frac{m^4}{n^2})$.

6.2. CONGA [29]

CONGA (Cluster-Overlap Newman Girvan Algorithm) is based on the well known edge betweenness community discovery algorithm [4], described in Section 6.1. It adds to this algorithm the ability to split vertices between communities, based on the new concept of "split betweenness".

The split betweenness [119] of a vertex v is the number of shortest paths that would pass between the two parts of v if it was split. There are many ways to split a vertex into two, the best split is the one that maximizes the split betweenness.

The proposed algorithm:

1. Calculate edge betweenness of edges and split betweenness of vertices.
2. Remove edge with maximum edge betweenness or split vertex with maximum split betweenness, if greater.
3. Recalculate edge betweenness and split betweenness.
4. Repeat from step 2 until no edges remain.

In CONGA, edge betweenness and split betweenness are calculated by counting the number of short paths. This is not an exact solution, but it is defined a greedy heuristic. In particular, the shortest path considered are those no longer than h (a parameter).

CONGA is also used in the Peacock framework in which it is possible to use a classical disjoint community discovery algorithm in order to find overlapping communities [120]. The network is transformed to a new, larger, network. Each step of the transformation splits a vertex into two vertices and one edge, then it is applied a classical CD algorithm and the result is a (possible) overlapping community structure of the network.

6.3. L-Shell [30]

The strategy of this method consists of an $l-shell$ spreading outward from a starting vertex. An $l-shell$ is a group of l vertices whose aim is to grow and occupy an entire community. As the starting vertex's nearest neighbors and next nearest neighbors (and so on) are visited by the $l-shell$, two quantities are computed: the *emerging degree* and *total emerging degree*. The emerging degree of a vertex is defined as the number of edges that connect that vertex to vertices the $l-shell$ has not already visited as it expanded from the previous $(l-1)$, $(l-2)$, ... $-shells$. The total emerging degree K_j of an $l-shell$ is then the sum of the emerging degrees of all vertices on the leading edge of the $l-shell$.

For a starting vertex j the algorithm starts an $l-shell$, $l=0$, at vertex j (add j to the list of community members) and compute the total emerging degree of the shell. Then

it spreads the $l-shell$, $l=1$, it adds the neighbors of j to the list, and compute the new total emerging degree. Now it is able to compute the change in the emerging degree of the shell. If the total emerging degree is increased less than a given threshold α, then a community has been found. Otherwise it increases the size of the shell (posing $l = l+1$) until α is crossed or the entire connected component is added to the community list. The assumption is that a community is a structure in which the total emerging degree cannot be significantly increased, i.e. the vertices at the border of the community have few edges outside it and these edges are the bridges among different communities.

It is possible for the $l-shell$ to spill over the community it is detecting. This is dependent on how the starting vertex is situated within the graph: if it is closer (or equally close) to some non-community vertex, the $l-shell$ may spread along two or more communities at the same time. To alleviate this effect, one can run the algorithm n times, using each vertex as a starting vertex, and then achieve a group consensus as to which vertices belong to which communities.

6.4. Internal-External Degree [31]

An approach close to $l-shell$ starts from the similar basic assumption that communities are essentially local structures, involving the nodes belonging to the modules themselves plus at most an extended neighborhood of them. A community is a sub-graph identified by the maximization of a property or fitness of its nodes. The fitness chosen here is the total internal degree of nodes on the sum of internal and external degrees to the power of a positive real-valued parameter (α).

The internal degree of a module equals the double of the number of internal links of the module. The external degree is the number of links joining each member of the module with the rest of the graph.

Given a fitness function, the fitness of a node A with respect to sub-graph $\mathcal{G}$, f_G, is defined as the variation of the fitness of sub-graph $\mathcal{G}$ with and without node A.

The steps of the algorithm are significantly different from the $l-shell$ formulation. A loop is performed over all neighboring nodes of $\mathcal{G}$ not included in $\mathcal{G}$. Then neighbor with the largest fitness is added to $\mathcal{G}$, yielding a larger sub-graph $\mathcal{G}$'. After the new node is added, the fitness of each node in $\mathcal{G}$' is recalculated. If a node turns out to have negative fitness, it is removed from $\mathcal{G}$', yielding a new sub-graph $\mathcal{G}$". In this case the fitness needs to be recomputed for each node, otherwise a new neighboring node of the new $\mathcal{G}$ is added. The process stops when the nodes examined in the neighborhood of $\mathcal{G}$ all have negative fitness, i.e. their external edges are all bridges.

To cover all the network one pick at random a node, compute and then pick at random another node that is not in the first community and so on.

Large values of α yield very small communities, small values instead deliver large modules. For $\alpha = 1$ this method is equivalent to [117], another algorithm that falls in this category. By going from $\alpha = 0.5$ to $\alpha = 2$ one can discover the hierarchical structure of the network.

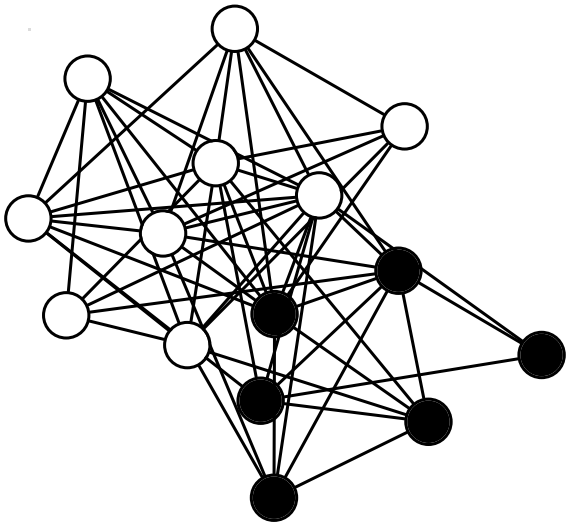


Figure 9: An example of graph partitioned with a diffusion process.

7. Diffusion

A diffusion is a process in which vertices or edges of a graph are randomly designated either "occupied" or "unoccupied" and one asks about various properties of the resulting patterns of vertices [121] (see Figure 9). A generalization of a diffusion process can be used as technique for community discovery in complex networks, according to the following definition of community:

Meta Definition 5 (Community). *A community in a complex network is a set of nodes who are grouped together by the propagation of the same property, action or information in the network.*

We recall that, according to this meta definition, a community can be also defined as a set of entities influenced by a fixed set of sources. This consideration is important because consider as a community discovery method also algorithms which are not explicitly developed as approaches for the graph partitioning. Basically, this definition of the problem overlaps with another well known data mining problem: influence spread and flow maximization [1], often for viral marketing purposes [122]. In this field some preliminary ideas can be found in [123], but in this work only a novel centrality measure is defined, and the approach can be mapped in the Newman edge betweenness algorithm [4].

Other interesting works in viral marketing are, given a community partition, the analysis of the group characteristics in order to predict their evolution [124]. Moreover, it is possible to predict if a single vertex will be attached

to that group, or even classify some features (and the evolution of these features) of a group. While it is not a community discovery work, this can be used as a framework after a community detection algorithm in order to obtain a temporal evolving description of the identified groups.

To sum up, the classical community discovery diffusion-based algorithms here presented are: a label propagation technique [32], dynamic node coloring for temporal evolving communities [33] and an edge resistor algorithms that consider the original graph as an electric circuit [34].

The influence propagation approaches here reviewed are: an analytical description of a network representing an exchange of information [35]; GuruMine [8], a framework whose aim is to analyze the so called "tribes", DegreeDiscountIC [36], a classical spread maximization algorithm and a mixed membership stochastic blockmodels algorithm [37], which uses Bayesian inferences in order to compute the final state of the influence vectors for each node in the network.

7.1. Label Propagation [32]

Suppose that a node x has neighbors $x_1, x_2, ..., x_k$ and that each neighbor carries a label denoting the community to which it belongs to. Then x determines its community based on the labels of its neighbors. A three step example of this principle is depicted in Figure 10.

Authors assume that each node in the network chooses to join the community to which the maximum number of its neighbors belong to. As the labels propagate, densely connected groups of nodes quickly reach a consensus on a unique label. At the end of the propagation process, nodes having the same labels are grouped together as one community.

Ideally the iterative process should continue until no node in the network changes its label. However, there could be nodes in the network that have an equal maximum number of neighbors in two or more communities. Authors perform the iterative process until every node in the network has a label to which the maximum number of its neighbors belong to. The stop criterion is only a condition and not a measure that is being maximized or minimized. Consequently there is no unique solution and more than one distinct partition of a network into groups satisfies the stop criterion.

As one can expect, an equal maximum number of neighbors in two or more communities can belong to both communities, thus identifying overlapping communities. It is easy to define an overlapping version of this algorithm. Recently, this overlapping formulation has been proposed [125].

7.2. Node coloring [33]

In this approach, that represents an evolution of [126], the base input representation is a bipartite graph of individuals connected to events. At each snapshot of the graph some individuals form groups by attending at the same event. The rules to identify a community are:

1. In each time step, every group is a representative of a distinct community;
2. An individual is a member of exactly one community at any one time (but can change community affiliation over time);
3. An individual tends not to change its community affiliation very frequently;
4. If an individual keeps changing its affiliations among many different communities, then it is not a true member of any of those communities;
5. An individual is frequently present in the group representing the community with which it is affiliated.

Authors define the community interpretation of a graph G a function $f : V \to \mathbb{N}$. Each individual belongs to exactly one community in each time-step, and each group represents exactly one community. Thus, even if the affiliation can change over time, this is a disjoint community detection algorithm, not an overlapping one.

To measure the quality of a community interpretation, authors use costs to penalize violations of Postulates 3 and 5. There are three different types of costs: *individual* (incurred whenever an individual changes its color), *group* (if an individual vertex does not have an edge to the group of the same color or if an individual vertex has an edge to a group of a color different from its own) and *color* (for each color an individual uses beyond its first). The optimization problem is then to find the valid community interpretation minimizing the total cost resulting from the individual edges, group edges and color usage.

Authors present an exhaustive global optimum algorithm with exponential time complexity (that with dynamic programming tries all possible coloring of the graph) and then some heuristics. First one: a group coloring is good if most of the individuals can retain their color from one step to the next. Thus it is possible to enumerate all or many maximal matchings, and choose from among them based on the actual coloring cost. Second: the matching algorithm tries to maximize the amount of *similarity* (using Jaccard index) preserved from one time step to the next. The greedy algorithm repeatedly selects the groups pair (g, g') with the highest similarity, and decide that g, g' should have the same color.

In a further work [127] the authors present a set of heuristics and optimizations that can lead to lower time complexity of their algorithm.

7.3. Kirchhoff [34]

In this paper, the basic idea is to imagine each edge to be a resistor with the same resistance. It is then possible to connect a "virtual battery" between some chosen vertices so that they have fixed voltages. Having made these assumptions the graph can be viewed as an electric circuit with current flowing through each edge (resistor). By solving Kirchhoff equations authors can obtain the voltage value of each node. Authors claim that, from a node's

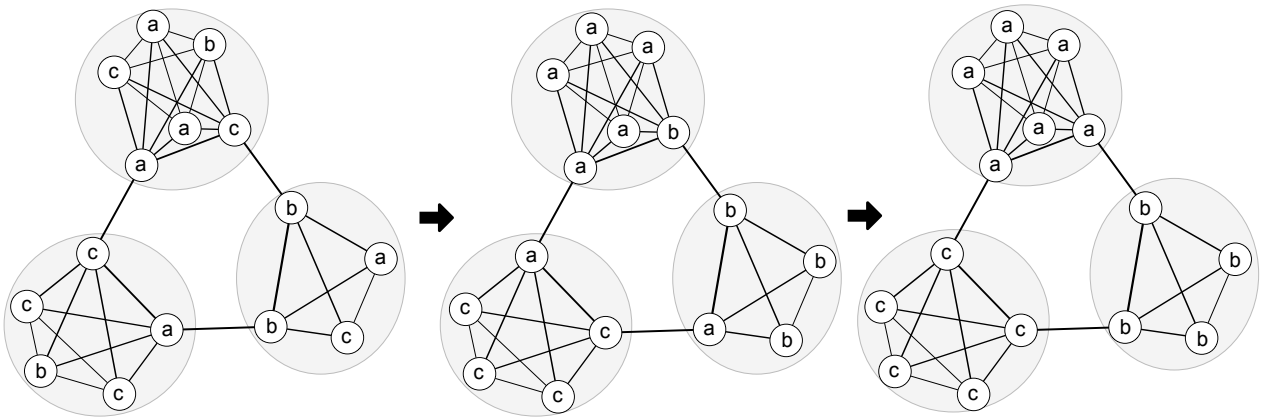


Figure 10: Possible steps of a label propagations-based community discoverer.

voltage value they are able to judge whether it belongs to one community or another.

There are three critical questions. In order to answer to these questions authors enumerates a series of heuristics. Main issues and solutions are the following:

1. *How to pick the two poles so that they lie in different communities?* The proposed solution is to pick poles randomly using some characteristics (such as geodesic distance) to be sure that the two nodes lies in different communities.
2. *What threshold should be used to separate the two communities?* A tolerance parameter is defined. It expresses how much a node belongs to a particular community. This can also be generalized for overlapping community discovery.
3. *How to generalize this method to networks with more than two communities?* Authors propose to repeat the process in order to find the maximum voltage gap and apply a majority vote.

A further expansion [128] apply a walk-based approach in order to unveil the hidden hierarchical structure of the network and identify good choices for the seed poles. Then authors apply a very similar implementation of this method using a Kirchhoff matrix.

7.4. Communication Dynamic [35]

The focus of the paper is on an analytical description of the evolution of a network, whose size is stable over time and represent the exchange of communication among individuals. The authors present a locality based model for communication dynamics, that can be used in order to identify the mechanisms of community creation and evolution over time in a social network.

Existing similar approaches, such as preferential attachment [129], are applicable only when the communications are open (observable to all nodes). Authors instead present a locality based model which relies on two fundamental principles: first the concept of locality reduces the set of nodes a node can attach to in the next time step, and second after obtaining a node's locality, one must specify the attachment mechanism, used by the individual to select the nodes in its locality to which it will connect at the next time step. This is a Markov chain-like approach.

For the preliminary community structure that identifies the local environment of a node authors use an existing method based on density [27]. Authors define the blogograph as a directed, unweighted graph representing the communication of the blog network within a fixed time-period. There is a vertex in the blogograph representing each blogger and a directed edge from the author of any comment to the owner of the blog. Authors consider consecutive weekly snapshots of the network.

Authors recorded statistics as number of vertices, edges, power-law degree distribution exponent, giant component size and so on, observing that they are stable over time, consistent with previous observation (like in [130] and [131]). They also provide an indicator of a community vitality over time.

The goal of the model is to produce a sequence of graphs which simulate the connection and reconnection of vertices and can be used for community validation.

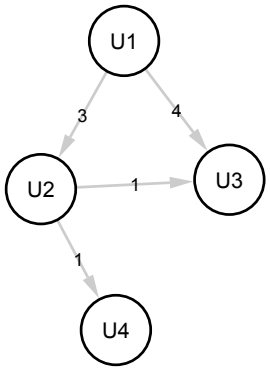
7.5. GuruMine [8]

Aim of GuruMine is to investigate how influence (for performing certain actions) propagates from users to their network friends, potentially recursively, thus identifying a group of users that behaves homogeneously (i.e. a tribe, or a community). A node u is a leader if u performed a and within a chosen time bound after u performed a, a sufficient number of other users performed a. Furthermore these other users must be reachable from u thus capturing the role social ties may have played.

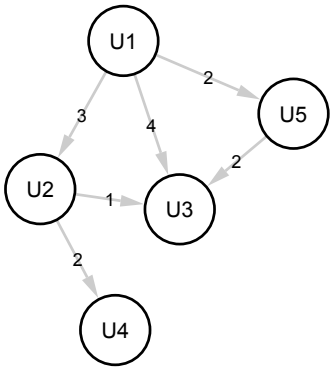
For instance, in Table 3 we have a possible action table with two actions, α and β, and five users. Figure 11(b) and 11(c) represent the influence graphs of these two actions. $U1$ can be considered a tribe leader in both cases. However, for action α, $U1$ can be considered not a leader if the threshold regarding the minimum number of influenced users is equal to 4.

User	Action	Time
U1	β	12
U5	β	14
U1	α	15
U2	β	15
U3	β	16
U4	β	17
U2	α	18
U4	α	19
U3	α	19

(a) Action Table



(b) Influence Graph, Action α



(c) Influence Graph, Action β

Figure 11: The GuruMine data structures.

A stronger notion of leadership might be based on requiring that w.r.t. each of a class of actions of interest, the set of influenced users must be the same: this will identify a tribe leader, meaning the user leads a fixed set of users (tribe) w.r.t. a set of actions, that can be considered a community. The general goal is similar to some recent works (such as [132], [133] and [134]). However, here the input includes not just a graph (which is not edge-weighted) but an action table which plays a central role in the definition of leaders. Secondly, the focus is on finding all leaders based on the frequent pattern mining approach, which is different from the approaches employed in the above works.

Besides a classical social network, the method has as input an action log, which contains a tuple $(u; t; a)$ indicating that user u performed action a at time t. Users in the Actions table correspond to nodes of the graph. For each action, authors define a propagation graph whose nodes are users who had performed the action at least once, and directed edges whenever the action propagates from user i to user j. Each propagation graph can have more than one source node; it is directed, and cycles are impossible due to the time constraint which is the basis for the definition of propagation. Authors also define a maximum propagation time threshold π, at the basis of the influence graph of each node (all nodes in the propagation graph that are reachable from u with a maximum elapsed time equal to π). To be a leader the influence graph must be greater than a threshold ψ (for a number of actions greater than σ), to be a tribe leader the composition of the influenced graph must be the same.

Authors introduce also other constraints such as the confidence threshold, similar to the confidence of the association rules [135], and the genuine leader concept, i.e. leaders that are not influenced by another leader. All thresholds and parameters are optional.

Any algorithm for extracting leaders must scan the action log table and traverse the graph. The implementation works with only one scan, with the action log stored in chronological order, thus it is possible to use a (backward moving) time window of size π. With this scan one can compute the influence matrix $IM_\pi(U; A)$. (Genuine) Leaders are easily computed from this matrix.

For tribe leaders one needs the influence cube $Users \times Actions \times Users$, with cells containing boolean entries if user v was influenced by user u w.r.t. action a. We can see a tribe as an item-set, and finding tribe leaders as finding frequent item-sets larger than a given threshold ψ minimum acceptable tribe size, that is the definition of common behavior community. The implementation of this phase is provided with ExAMiner [136]. Once obtained the tribes and the tribe leaders it is possible to merge tribes influenced by the same group of leaders thus obtaining the communities. Even if this step is not provided by GuruMine, the idea can be easily implemented with a brute force approach. Some efficient techniques can be developed.

This work is part of a bigger framework that supports also a query interface [137].

7.6. DegreeDiscountIC [36]

In this work we are in the classical data mining influence spread setting. The problem definition consists in defining whom to include in the initial set of targeted users so that they eventually influence the largest number of people in the network. This knowledge can be used for community discovery: each seed node is the head of a community that acts uniformly and the set of these influenced nodes are the community members. This work is an implementation of the idea in [134] and the improvement of the algorithm proposed in [133].

Influence is propagated in the classical network representation of social interactions according to a stochastic cascade model. The influence maximization problem is to find k vertices in the graph (referred to as "seeds") such that under the influence cascade model, the expected number of vertices influenced by the k seeds is the largest possible.

Let S be the subset of vertices selected to initiate the influence propagation. First optimization lays in the cas-

cade model (IC). Let A_i be the set of vertices that are activated in the i-th round, and $A_0 = S$. For any $\overline{uv} \in E$ such that $u \in A_i$ and v is not yet activated, v is activated by u in the $(i+1)$-th round with an independent probability p, which authors call the propagation probability. In other words, if there are neighbors of v that are in A_i, $v \in A_{i+1}$ with probability p. This process is repeated until A_{i+1} is empty. Then authors remove all edges not for propagation from G to obtain a new graph G'. With this treatment, the set of influenced vertices is simply the set of vertices reachable from S in G'.

A second optimization is applied in the weighted cascade (WC) model. If u is activated in round i, then with probability $\frac{1}{d_v}$, v is activated by u in round $i+1$. Similar to the IC model, each neighbor can activate v independently. Therefore, if a not-yet-activated vertex v has neighbors activated during the i-th round, the probability that v is activated in round $i+1$ is $\frac{1}{d_v}$.

As last optimization, if u has been selected as a seed, then when considering selecting v as a new seed based on its degree, authors not count the edge $\overline{vu}$ towards its degree. Thus authors discount v degree by one due to the presence of u in the seed set, and do the same discount on v degree for every neighbor of v that is already in the seed set. With this and more fine tuned heuristics on degree, authors can develop a well performing algorithm in reasonable complexity.

7.7. MMSB [37]

In the mixed membership stochastic blockmodels approach (in short MMSB) each node is associated with a randomly drawn vector, say $\overrightarrow{\pi_i}$ for node i, where $\pi_{i,g}$ denotes the probability of node i belonging to group g. Each node can simultaneously belong to multiple groups with different degrees of affiliation strength. The indicator vector $\overrightarrow{z}_{p \to q}$ denotes the group membership of node p when it is approached by node q and $\overrightarrow{z}_{p \leftarrow q}$ denotes the group membership of node q when it is approached by node p.

If the network has K groups, the algorithm is the following: for each node i draw a K dimensional mixed membership vector $\overrightarrow{\pi_i}$. For each node couple draw a membership indicator for the initiator, $\overrightarrow{z}_{p \to q}$ and one for the initiator $\overrightarrow{z}_{p \leftarrow q}$ and finally sample the value of their interaction.

Each node may assume different membership when interacting to or being interacted by different peers. Authors also introduce a sparsity parameter to calibrate the importance of non-interaction.

As for other mixed membership models, this is intractable to compute. The normalizing constant of the posterior is the marginal probability of the data, which requires an intractable integral over the simplicial vectors $\overrightarrow{\pi_i}$. A number of approximate inference algorithms for mixed membership models have appeared in recent years such as mean-field variational methods [138], expectation propagation [139] and Monte Carlo Markov chain sampling [140]. In this work authors apply mean-field variational methods to approximate the posterior of interest.

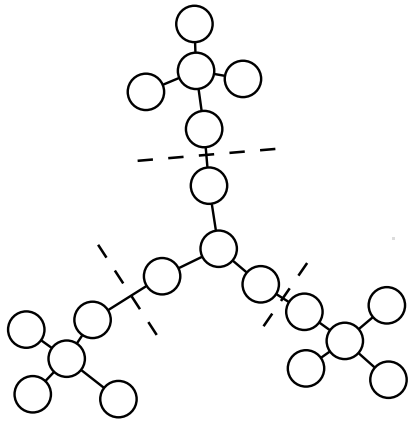


Figure 12: An example of graph which can be partitioned considering the relative distance, in terms of number of edges, among its vertices.

8. Closeness

A very intuitive notion of community in a complex network is based on the concept of reachability of its members. A community in practice is a set of individuals who can communicate each other very easily because they have a direct link with almost anybody in the community. We have depicted a simple example for this configuration in Figure 12. The underlying definition of community in this case is the following:

Meta Definition 6 (Community). *A community in a complex network is a set of nodes who can reach any member of its group usually crossing a very low number of edges, significantly lower than the average shortest path in the network.*

A very efficient approach used with this problem definition relies on random walks. In a network a random walk is a process in which at each time step a walker is on a vertex and moves to a vertex chosen randomly and uniformly among its neighbors. The same procedure is followed for the new selected vertex. This is a Markov process. However, some different strategies has been formulated in the past in order to obtain very sophisticated random walk based application. Just as an example, the popular link analysis PageRank algorithm [111] is based on random walks.

To the best of our knowledge there are three main community discoverer which use random walks in order to find community whose members are very close to each other: Walktrap [38], based on the assumption that when performing random walks the virtual surfer is trapped in the high density regions of the graph (i.e. the communities); DOCS [39], a more complex framework that uses also modularity as a fitness function; and Infomap [40], which applies an information-theoretic approach.

8.1. Walktrap [38]

Walktrap approach is based on the following intuition: random walks on a graph tend to get "trapped" into densely connected parts corresponding to communities. Authors consider some properties of random walks on graphs. With there properties it is possible to define a measurement of the structural similarity between vertices and between communities, thus defining a distance function. With this approach, one can merge iteratively the vertices into communities. The final output is a hierarchical community structure that may be represented in a dendrogram.

The key problem is the definition of the distance function, computed from the information given by random walks in the graph. This distance must be large if the two vertices are in different communities, and small otherwise. In the original paper this distance is defined as:

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{d(k)}}$$

where P_{ik}^t is the probability to go from i to j in t steps and $d(k)$ is the degree of vertex k.

A critical parameter is the length t of the random walks: it must be sufficiently long to gather enough information about the topology of the graph, but on the other hand it must not be too long because when the length of a random walk starting at vertex i tends towards infinity, the probability of being on a vertex j only depends on the degree of vertex j (and not on the starting vertex i).

High values of this measure mean that the two vertices i and j "see" the network in a very similar way, thus they belong to the same community.

Similar random walk approaches are [141, 142]. However they are more inefficient comparing the medium case complexity of Walktrap.

8.2. DOCS [39]

This method is based on spectral partition and random walk expansion, and represent an extension of [143].

The first step is to create the "seed groups". DOCS coarsens the original graph into a series of higher coarsening level graphs. Then the algorithm starts to agglomerate these graphs. Agglomeration is measured by modularity: if merging two neighbor groups brings good change to the modularity value of whole graph, they will be collapsed. The coarsest graph is split into two groups. The recursive partition is executed until there is no possible further partitioning step. Finally, the partition is projected back to the original vertices by going through a series of lower level graphs.

As second step, the seed groups are expanded with a lazy random walks technique. The fixed-depth expansion corresponds to a Breadth First Search tree. Its expansion process never considers the vertex filtering conditions and make the low-probability vertices have same expansion weights with high-probability ones, which easily lead to skewed community structures. The random walks technique extend the neighbors of a vertex with random probabilities. The probabilities measure the expansion weights. At each add the modularity [19] change is computed. If the value is above zero, the new vertex is named as contributing vertex. Otherwise, it is called non-contributing one. In random walks, the algorithm also sorts all the expanded vertices in a descending order by their contributing values. Now the target communities can be extracted.

In order to perform this partition strategy, authors introduce several concepts in their work. They define the volume of a community as the total degrees of its members; and the edge border as a set of those edges whose vertices do not both belong to the community (but one of them does). Closely related to te community volume, another concept used by the authors is the community sparsity (presented in [144] with a number of community discovery algorithms). Also the concept of "Community Overlapping Rate" is introduced as the fractions of the cardinality of all the intersections of a given community with all other communities on the cardinality of the given community.

8.3. Infomap [40]

Infomap algorithm is considered one of the most accurate community discovery methods currently available. It is based on a combination of information-theoretic techniques and random walks. Authors want to explore the graph structure with a number of random walks of a given length and with a given probability of jumping to a random node. This approach is equivalent to the random surfer of the PageRank algorithm [111]. Intuitively, the random walkers are trapped into a community and exit from it very rarely. Thus if we have a division in communities we can efficiently describe these random walks as a series of intra community steps followed by an inter community jump. The formal equation described by these concepts is the following:

$$L(M) = qH(Q) + \sum_{i=1}^{m} p_i H(P_i)$$

where L is the lower bound for the number of bits needed in the description of the nodes of the network, M is the community partition, q is the probability that the random walk jumps from a community to another on any given step, $H(Q)$ is the entropy of the description of the community, m is the number of communities in the network, p_i is the fraction of within-community movements that occur in community i and $H(P_i)$ is the entropy of the within-community movements, including the exit code for community i.

Trying any possible community partition in order to minimize $L(M)$ is inefficient and intractable. Authors narrow the space of the candidate partitions with several iteration of a greedy modularity community discoverer [19].

They then refine the partition of the graph with simulated annealing [2]. This optimization is driven by the information-theoretic principle of reducing the number of bits needed to encode the information of the structure (i.e. they minimize $L(M)$). This is done by assigning an Huffman coding to the nodes of the network.

The description of the nodes of the network is divided into two levels. Authors retain unique names for large-scale objects, the communities identified within the network in the first step, and they reuse the names associated with fine-grain details, the individual nodes within each community. This two-level description allows to describe the path in fewer bits, relying on the fact that a random walker is statistically likely to spend long periods of time within certain clusters of nodes.

9. Structure Definition

A number of works tackle the community discovery with a very strong assumption: to be called a community a group of vertices must follow a very strict structural property. In other words, they use the following meta definition of community:

Meta Definition 7 (Community). *A community in a complex network is a set of nodes with a precise number of the edges among them, distributed in a very precise topology defined by a number of rules. Set of nodes which do not satisfy these structural rules are not communities.*

Aim of the community discovery algorithm is to find all the maximal structures in the network satisfying the desired constraints.

This task can be considered very similar to a very well known data mining problem in network analysis: the graph mining. Some examples of graph mining algorithms are [5, 145, 146, 147]. However, traditional graph mining algorithms only returns all the single different structure patterns with their support. In community discovery there is only one important structure and the desired result is the list of all vertex groups forming that structure in the network.

Thus in this survey we will ignore the pure graph mining algorithms and we will focus on the structural community discovery approaches. The methods here reviewed are: clique percolation [3] and its evolution for bipartite graphs [42], the s-plexes detection [41] (s-plexes are particular structures later defined) and a maximal clique approach [43]. We will not focus on other minor evolutions, like the k-dense approaches ([148]).

9.1. K-Cliques [3]

Palla et al. suggest that community can be interpreted as a union of smaller complete (fully connected) sub-graphs that share nodes. Authors define a k-clique-community as the union of all k-cliques that can be reached from each other through a series of adjacent k-cliques. Two

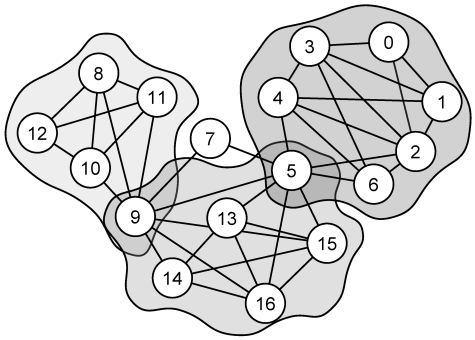


Figure 13: The overlapping community structure detected by a clique-percolation approach.

k-cliques are said to be adjacent if they share $k-1$ nodes. Authors define also a k-clique chain as the union of a sequence of adjacent k-cliques. Further they introduce the concept of k-clique connectedness: two k-cliques are k-clique-connected if they are parts of a k-clique chain.

A 2-clique is simply an edge and a 2-clique-community is the union of those edges that can be reached from each other through a series of shared nodes. Similarly, a 3-clique-community is given by the union of triangles that can be reached from one an other through a series of shared edges.

Consider Figure 13. In this case the clique percolation approach detects $\{0, 1, 2, 3\}$ as a 4-clique. Then it considers $\{1, 2, 3, 4\}$: it is again a 4-clique and it shares 3 vertices with the previous one. Thus the two cliques are joined in one community. Same considerations can be done for the 4-cliques $\{2, 3, 4, 6\}$ and $\{2, 4, 5, 6\}$, thus identifying the community $\{0, 1, 2, 3, 4, 5, 6\}$. Please notice that with this process two communities can have an overlap of some vertices. In this example both vertices 5 and 9 belongs to two overlapping communities.

The algorithm first extracts all complete sub-graphs of the network that are not parts of larger complete sub-graphs. Aim of the first phase is to populate a clique-clique overlap matrix. In this data structure each row (and column) represents a clique and the matrix elements are equal to the number of common nodes between the corresponding two cliques. The diagonal entries are equal to the size of the clique. The k-clique-communities can be found by erasing every off-diagonal entry smaller than $k-1$.

9.2. S-Plexes Enumeration [41]

An s-plex is a relaxed concept of the c-isolated clique [149]. Let $G = (V, E)$ be an undirected graph. A set $S \subseteq V$ of k vertices is called c-isolated if it has less than ck outgoing edges, where an outgoing edge is an edge between a vertex in S and a vertex in $V \backslash S$. Clearly, a c-isolated clique is a concept close to a community, but it is considered too restrictive. The c-isolated cliques problem has a known solution linear on number of edges and exponential on the c isolation parameter [150].

Instead of this concept authors use a relaxed version of a c-isolated clique called s-plex [151]: in an undirected graph $G = (V, E)$, a vertex subset $S \subseteq V$ of size k is called an s-plex if the minimum degree in $G[S]$ is at least $k - s$. Hence, cliques are exactly 1-plexes. Authors also define novel and different isolation concepts that make the s-plexes enumeration closer to the community discovery problem.

Since in an s-plex S of size k every vertex $v \in S$ is adjacent to at least $k - s$ vertices, the sub-graph induced by S in the complement graph (the graph with the same set of vertices and complementary edge set) $G[S]$ is a graph with a maximum degree of at most $s-1$. The idea is to enumerate maximal s-plexes in G by deleting minimal sub-graph with maximal degree equal to $s - 1$ in the complement graph.

Another key concept for this solution is the pivot set P. The pivot set contains the pivot vertex v and those vertices that belong to the s-plex but are not adjacent to v. The pivot vertex is defined as the vertex with the lowest index among the vertices with less than c outgoing edges.

There is already a slower similar algorithm [152]. Here, the simple idea is to pick a vertex v with more than d neighbors, and then to branch into $d+2$ cases corresponding to the deletion of v or the deletion of one vertex of an arbitrary set of $d + 1$ neighbors of v.

The algorithm removes vertices from candidate set C having too few neighbors in C. It builds the complement graph, then for each possible pivot set P applies the deletion of minimal sub-graph in the complement graph. Finally, it removes enumerated s-plexes that either have pivot $u \neq v$ or are not maximal.

9.3. Bi-Clique [42]

This is a bipartite graph version that solves some issues about the k-clique approach [3]. The k-clique algorithm is unable to analyze sparse network regions, due to the fact that 2-clique communities are simply the connected components of the network. The first non-trivial k-clique has size k = 3 and nodes must have at least two links in order to qualify for participation in a 3-clique. In networks with heavy tailed degree distributions, a large fraction of the nodes have degree less than two and an even larger fraction of nodes do not participate in cliques of size three or greater.

Bi-clique is a natural approach for affiliation networks, where a one-mode projection disregards important network information: all (sparse) information about the bipartite linkages is reduced to a dense network of two-point correlations; and all of the information contained in the weights is typically discarded in a subsequent thresholding operation. The presented Bi-Clique algorithm is able to detect structures between 2-clique communities and 3-clique communities where the k-clique algorithm fails to locate structure. By changing two parameters the analyst can identify both large groups of users with few interactions among them and small groups that belongs to many common affiliations.

Algorithm begins isolating the N maximal bi-cliques in the bipartite network using [153]. Using this list the authors create two symmetric clique overlap matrix for the two classes of nodes. Then for both matrix diagonal elements respectively greater than or equal to a and b, the two parameters of the algorithm, are set to one. All other diagonal elements are set to zero. All off-diagonal elements that correspond to a zero diagonal element are set to zero. Thus, only elements respectively greater than or equal to $a - 1$ and $b - 1$ are kept. The final overlapping matrix is obtained by the matrix intersection, using the AND operator. The final step is to determine the connected components of L; each component corresponds to a bi-clique community

It is possible to construct a network consisting of the bi-clique communities. In this network, each community is a node and two communities are linked if they have nodes in common.

9.4. EAGLE [43]

A community is commonly defined as an high link-density portion of the graph. EAGLE starts from the assumption of the presence of a large clique in a dense-linked community. This clique could be considered the core of the community. EAGLE algorithm deals with the set of maximal cliques, a clique which is not a subset of any other cliques, rather than the set of sole vertices.

Firstly find out all the maximal cliques in the network with Bron-Kerbosch algorithm [154] (complexity $\mathcal{O}(3^{\frac{n}{3}})$). The maximal cliques, whose vertices are from some other larger maximal cliques, are called subordinate maximal cliques (discarded). Authors also set a threshold k and neglect all the maximal cliques with the size smaller than k (between 3 and 6). Some vertices (subordinate vertices) do not belong to any remaining maximal cliques.

First stage: build the dendrogram. EAGLE finds out all maximal cliques in the network, neglecting, as we said, subordinate maximal cliques. The remainder cliques are the set of the initial communities. Each subordinate vertex is also considered an initial community comprising the sole vertex. EAGLE then calculates the similarity between each pair of communities. After this step it selects the pair of communities with the maximum similarity, incorporating them into a new one community and calculate the similarity between the new community and other communities. The similarity measure is the modularity [19]. This calculation is repeated until only one community remains, thus completing the dendrogram.

The second stage is the cut of the dendrogram. Any cut through the dendrogram produces a cover of the network. To determine the place of the cut, a measurement is required to judge the quality of a cover, computed with a given variant of modularity.

10. Link Clustering

In literature there are some recent approaches based on the idea that the community is not a partition of the nodes of the network, but a partition of the links. In other words, is the relationship between two entities that belongs to a particular environment and the entities belong to all the communities of their edges (or a subset of them). The meta definition is the following

Meta Definition 8 (Community). *A community in a complex network is a set of nodes which share a number of relations clustered together since they belong to a particular relational environment.*

The basic approaches to the link clustering problem is to define a projection graph in which the nodes represent the links of the original graph and the definition of a proximity value in order to understand how close two edges of the network are. In both cases the critical point is to measure the relations among the edges. Then a classical clustering algorithm could be applied.

The two methods here reviewed reflect both approaches. The first one [44] define the projection graph with a random walk measure for the proximity of the projected edges, then uses modularity to compute the modules of the network. The second one [45] is a general framework in which it is possible to define any distance measure for the nodes (such as the Jaccard index) and then apply a classical hierarchical clustering technique based on this distance definition.

10.1. Link modularity [44]

In this first link modularity work, authors suggest that it is possible to define communities as a partition of the links rather than of the set of nodes. A node may then have links belonging to several communities and in this case it belongs to several communities.

Authors interpret the usual modularity Q in terms of a random walker moving on the nodes. They further define two walking strategy: a link-link and a link-node-link random walk. They project the adjacency matrix into a bipartite incidence matrix. The elements $B_{i\alpha}$ of this $n \times m$ matrix are equal to 1 if link α is related to node i and 0 otherwise.

The incidence matrix is then projected into a line graph: a link is added between two nodes in this projected graph if these two nodes had at least one node of the other type in common in the original incidence bipartite graph. Then, modularity is computed on this line graph.

10.2. Link Jaccard [45]*

In this approach, authors start from the assumption that whereas nodes belong to multiple groups (individuals have families, co-workers and friends), links often exist for one dominant reason (two people are in the same family, work together or have common interests). Instead of assuming that a community is a set of nodes with many links between them, they consider a community to be a set of closely interrelated links.

They define a similarity measure as the Jaccard coefficient. This measure is computed on the sets of neighbors of each edges sharing one node (i.e. only adjacent edges). The formula used is:

$$S(e_{ik}, e_{jk}) = \frac{|n_+(i) \cap n_+(j)|}{|n_+(i) \cup n_+(j)|}$$

where e_{ik} is an edge between nodes i and k and $n_+(i)$ is the set of neighbors of node i. It is important to note that the approach can be used with an arbitrary similarity function for the edges. Furthermore, even if with this formula weights and multipartite structures are not considered, authors claim that it is possible to extend the approach in order to obtain these features.

Authors then build a dendrogram with a classical hierarchical clustering approach using the defined similarity measure. In the dendrogram each leaf is a link from the original network and branches represent link communities. In the identified hierarchical structure, links occupy unique communities whereas nodes naturally occupy multiple communities, owing to their links. Thus the extracted network structure is both hierarchical and overlapping. The dendrogram is then cut optimizing the partition density objective function [155].

Authors confront their approach with three state-of-the-art algorithms (clique percolation [3], fast greedy modularity [19] and Infomap [40], each of them reviewed in this survey) using four different evaluation metrics (overlap and community coverage, and overlap and community quality) obtaining on average better results.

11. No Definition

There exist in literature a number of frameworks for community discovery which use a very trivial definition of community or they have no definition at all. These methods often start from the assumption that there are some desirable features for the community not provided by many algorithms. They define some preprocessing and/or post-processing operations and then apply them to a number of different known other methods which do not extract communities with the desired features. In this way they improve the results.

Basically, the meta definition adopted is the following:

Meta Definition 9 (Community). *Communities in a complex network are sets which present a number of particular features regardless why their nodes are grouped together.*

The works which present a proper community definition are, for instance, the evolutionary clustering [9] or the CONGA algorithm [29], already presented in this survey.

By presenting their desired common features for the sets in the form of an independent community definition, these methods are not included in this category.

Instead we focus on two methods: the first one is an hybrid framework who combine bayesian and non-bayesian approaches [46], the second one is a method that relies on a custom definition of community given by the analyst and then performs a multidimensional community discovery identifying the noisy relations inside the network [47].

11.1. Hybrid [46]*

For this framework, authors start from the point that overlapping communities are a more precise description for the multiplicity of node's links, compared to non-overlapping approaches. If a node's links cannot be explained by a single membership, then the community discovery problem has to be solved in an overlapping formulation. On the other hand, if a node's links can be explained almost equally well by a number of single and mixed memberships, hard clustering may make a simpler assignment. The conclusion is that a combination of an overlapping community discoverer that takes as input already hard defined community by a non overlapping methods should have better performances.

From these assumptions HFCD framework is built. It is made of three parts: the **Bayesian core**, the **hint** source procedure and the **coalescing strategies**.

The **Bayesian core** is the overlapping community discovery algorithm that collects the hint from the other non overlapping method and outputs the final communities partition. In this work authors use a Latent Dirichlet Allocation on Graphs [156] as their core method. The reason of the choice is the non parametric and robust nature of the method on small perturbations. Authors claim that it is possible to choose any other Bayesian model based on Latent Dirichlet Allocation [157].

The Bayesian core needs some **hints** in order to perform the community discovery procedure. These hints are provided by any other non overlapping community detection algorithm, called by the author "hard clustering". In the article, authors use two different hard clustering techniques: modularity [19] and Cross Associations [55] (here reviewed in its evolution as Context-specific Cluster Tree [18]).

The most important contribution of this approach is creating a procedure solving the problem of how to incorporate the hints into the core model. This is done by the **coalescing strategies**. Authors propose three different strategies: *attribute*, *seeds* and *prior*. In the *attribute* strategy the community to which a node belongs is recorded as one attribute of the node itself. The LDA algorithm is then extended and it will consider the values of the community attribute as input for the community detection. In *seeds* the community extracted by the hard clustering method are used as the initial configuration of the communities in the first step of LDA process. This information is then completely forgotten in the subsequent LDA loops. *Prior* is a strategy similar to seeds: the communities are used as initial configuration, but in this case the old community affiliations are retained and the final result is a sort of average among the old hard communities and the new overlapping structure. In order to make the inference procedure both from attributes and for the initial configuration, authors use the Gibbs sampling technique [158].

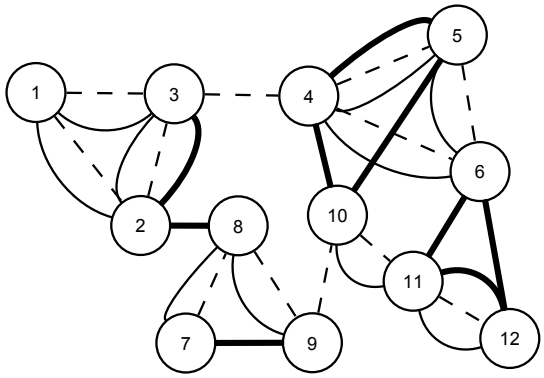


Figure 14: A multidimensional network. Solid, dashed and tick lines represent edges in three different dimensions.

11.2. Multi-relational Regression [47]

This algorithm aims mainly to discover hidden multidimensional communities. Authors call "relation" a dimension, i.e. a criterion to connect entities. They define some relation networks, group them together and create a kind of social network, calling it multi-relational social network or heterogeneous social network, another name for a multidimensional network. The basic assumption is the following: each relation (explicit or implicit) plays different roles in different tasks.

For instance consider the multidimensional network depicted in Figure 14. Authors suppose that an analyst might want to specify that nodes 8, 9, 10 and 11 belong to the same community. Then the three dimensions (represented by solid, dashed and tick edges) have different importance in reflecting the user information need. In particular the tick dimension can be considered noise, and the most important dimension is obviously the dashed dimension. The community discovery process should take this situation into account in order to provide an output close to the user information needs.

To this end, authors represent each relation with a weighted matrix. Each element in the matrix reflects the relation strength between the two corresponding entities. This matrix is then mined depending on a user example (or information need): the user submits a query defining the desired community structure. From this structure, the algorithm reconstruct the possible hidden relation, combining with linear techniques the single relation graphs, and then perform the community discovery on the resulting hidden graph.

The hidden relation is tackled as a prediction problem: once the combination coefficients of the desired entities and the desired relations are computed, the hidden relation strength between any object pair can be predicted. This is a regression problem that can be solved with a number of techniques [159]. For a discussion of the issues in this solution based on unconstrained linear regression please see [160]. The exact regression used is the Ridge Regression.

12. Related Works

In the last decade, several reviews of community discovery methods have been published in literature. To the best of our knowledge, the most important works are [161, 162, 163, 164, 165, 166].

Fortunato and Castellano [164], hugely extended by Fortunato in [6], have published the most recent and probably the most comprehensive review on the community discovery problem. They consider various definitions of community (local, global and vertex similarity), some features of the communities to be extracted and different categories for tackling the problem. The amount of algorithms and references considered in this review is impressive. We believe that a novel review on this topic is needed because authors analyze the main techniques of each method for community detection, but they do not build an organization of community definitions. Thus, they do not consider what is the main contribution of the present work: the creation of a community definition based classification of the state of the art algorithms. Not having our "inverted index" of community definitions, this review cannot be used by a researcher with his/hers own definition of what is a community in order to find the set of methods most close to his/hers problem. The aim of this review is to talk to people interested in build a new community detection algorithm, not to people who want to use the methods present in literature. Further, their work does not include some more advanced features and definitions of community present in literature, such as multidimensionality or an influence spread formulation of the problem.

Also Porter et al. [165] and Schaeffer [166] provide a very recent review of community discovery methods. In this last work it is also introduced the problem of a comprehensive meta definition of community in a graph. However, they suffer of the same problem previously exposed: although authors begin to provide different definitions of community, they do not create the classification of the community discovery algorithm based on it.

Newman [161] provides a pioneering work, in which organizes the historical approaches to the community discovery in complex networks following their traditional fields of application. Newman presents the most important classical approaches in computer science and sociology, enumerating algorithms such as spectral bisection [50] or hierarchical clustering [167]. The paper then is devoted to the review of the novel physic approaches to the community discovery problem. Such methods are the known edge betweenness [4] and modularity [80] approaches. This paper is very useful in order to have an historical view of the approaches to the problem, but it records few works and it cannot consider all the algorithms and categories of methods developed from its publication.

Chakrabarti and Faloutsos [162] present a complete survey of many aspects in the task of graph mining. In their review one important chapter is dedicated to community detection concepts, techniques and tools. Authors introduce the basic concepts of the classical notion of community structure based on edge density, along with other key concepts such as transitivity, edge betweenness and resilience. However, this survey is not explicitly devoted to the community discovery problem. Thus the survey is limited to the description of the existing methods and does not investigate further the possibility of different definitions of community or more complex analysis setting.

Danon et al. [163] test an impressive number of different community discovery algorithms. Their work is based on the comparison of the time complexity and performances of the considered methods. Further, they define an heuristic to evaluate the results of each algorithm and compare also their performances. However, this work is more focused on a practical comparison of the methods, and not on a truly classification, both in the community definition and in the feature considered for the input network, of the existing techniques present in literature.

Recently, some authors also proposed a benchmark graph, useful to test the community discovery algorithms [168].

13. Conclusions

In this work we have presented a survey on community discovery algorithms. Aim of this survey was to create a manual for the community discovery problem, useful to answer the question: "Given what is for the analyst a community, which community detection algorithm should he/she use?". This is a sort of "inverted index" in respect to the classical approach of the community discovery reviews, which are meant for an analyst already inside the community discovery problem.

In order to do so, we have firstly tackled the main problem in this field of research: the lack of an universal accepted definition of what is a community. As pointed out by Fortunato [6], this lack of a theoretical framework has some important consequences not only in the community detection task itself (if we do not agree on the meaning of "community" how can we extract a community from the network?) but also in other aspects. One of these aspects is, for instance, the evaluation of an algorithm w.r.t. the results from another approach with a different community definition.

We have proposed a meta definition of community, and on this basis we have built a novel classification of the

community discovery methods based on the relationships of each community definition with the general meta definition. The reviewed approaches are clustered under general categories such as internal density, community structure definition and so on. This classification is a proposed answer to the problems pointed out by the review of Fortunato. Each main method is then presented in details along with its relationships with other algorithms.

A crucial open problem we identify is to deeply study the overlap among the definitions of community. As pointed out in Section 3.1 there are several complex connections among different definitions and different algorithms. It is worthwhile to create an accurate graph representation of this overlap, in which the nodes are the algorithms connected if they share part of their community definition, some features of the input/output, some quality functions or a search space exploration approach. This multidimensional complex network can be studied in order to have a more clear and detailed view on the community discovery problem.

Another contribution of this paper is the inclusion of novel important features of a graph partition algorithm, not considered by other reviews. The definition of different features is critical because, as one can expect, there is no "perfect method" in general, but methods able to consider multidimensionality or not, algorithms treating overlapping communities or not, and so on. Including novel features like the multidimensionality is important because they add a fundamental analytical power to better describe real world phenomena. Moreover, an approach is not better if it has a longer list of supported features: in some cases a specialized method can achieve better performances of a general one. To this aim, we have designed the Table 2 as a useful tool in order to check the features of all algorithms, which helps the analyst in finding the desired algorithm also at the level of the features and not only at the level of the underlying comunity definition, presented previously.

To define and predict what will be the most important features in future is another open problem we leave as a future work. There is interest especially in multidimensionality, seen as a feature considered as part of the solution of the problem and not only as an input to be preprocessed. In other words, we want not only to consider the multidimensionality as an input, but also to extract truly multidimensional communities. Another interesting feature could be the presence of both hierarchical and overlapping organization of the community structure at the same time, since these two features were recently seen no more as exclusive features [45].

Acknowledgements. We gratefully acknowledge Sune Lehmann for useful discussions.

14. Bibliography

[1] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee, "Self-organization and identification of web communities," *IEEE Computer*, vol. 35, pp. 66–71, 2002.

[2] R. Guimera and L. A. N. Amaral, "Functional cartography of complex metabolic networks," *Nature*, vol. 433, no. 7028, pp. 895–900, 2005.

[3] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, pp. 814–818, June 2005.

[4] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *PROC.NATL.ACAD.SCI.USA*, vol. 99, p. 7821, 2002.

[5] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," 2002.

[6] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75 – 174, 2010.

[7] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Community evolution in dynamic multi-mode networks," in *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 677–685, ACM, 2008.

[8] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Discovering leaders from community actions," in *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, (New York, NY, USA), pp. 499–508, ACM, 2008.

[9] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 554–560, ACM, 2006.

[10] B. Long, X. Wu, Z. M. Zhang, and P. S. Yu, "Unsupervised learning on k-partite graphs," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 317–326, ACM, 2006.

[11] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 817–826, ACM, 2009.

[12] L. Tang, X. Wang, and H. Liu, "Uncovering groups via heterogeneous interaction analysis," in *ICDM*, IEEE, 2009.

[13] A. Banerjee, S. Basu, and S. Merugu, "Multi-way clustering on relation graphs.," in *SDM*, SIAM, 2007.

[14] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda, "Learning systems of concepts with an infinite relational model," in *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, pp. 381–388, AAAI Press, 2006.

[15] L. Friedland and D. Jensen, "Finding tribes: identifying close-knit individuals from employment patterns," in *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 290–299, ACM, 2007.

[16] D. Chakrabarti, "Autopart: parameter-free graph partitioning and outlier detection," in *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, (New York, NY, USA), pp. 112–124, Springer-Verlag New York, Inc., 2004.

[17] J. Ferlez, C. Faloutsos, J. Leskovec, D. Mladenic, and M. Grobelnik, "Monitoring network evolution using mdl," *Data Engineering, International Conference on*, vol. 0, pp. 1328–1330, 2008.

[18] S. Papadimitriou, J. Sun, C. Faloutsos, and P. S. Yu, "Hierarchical, parameter-free community discovery," in *ECML PKDD '08: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, (Berlin, Heidelberg), pp. 170–187, Springer-Verlag, 2008.

[19] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, p. 066111, 2004.

[20] E. A. Leicht and M. E. J. Newman, "Community structure in directed networks," *Physical Review Letters*, vol. 100,

	Name	Overlap	Dir	Weight	Dyn	NoPar	MDim	Incr	Multip	Complexity	BESn	BESm	Year	Ref
Feature Distance	Evolutionary*				✓			✓		$\mathcal{O}(n^2)$	5k	?	2006	[9]
	MSN-BD			✓					✓	$\mathcal{O}(n^2ck)$	6k	3M	2006	[10]
	SocDim	✓		✓			✓			$\mathcal{O}(n^2 \log n)*$	80k	6M	2009	[11]
	PMM			✓			✓			$\mathcal{O}(mn^2)$	15k	27M	2009	[12]
	MRGC		✓		✓		✓		✓	$\mathcal{O}(mD)$	40k	?	2007	[13]
	Infinite Relational					✓	✓			$\mathcal{O}(n^{2c}D)$	160	?	2006	[14]
	Find-Tribes				✓				✓	$\mathcal{O}(mnK^2)$	26k	100k	2007	[15]
	AutoPart		✓			✓		✓		$\mathcal{O}(mk^2)$	75k	500k	2004	[16]
	Timefall				✓	✓			✓	$\mathcal{O}(mk)$	7.5M	53M	2008	[17]
	Context-specific Cluster Tree					✓			✓	$\mathcal{O}(mk)$	37k	367k	2008	[18]
Internal Density	Modularity			✓						$\mathcal{O}(mk \log n)$	400k	2.5M	2004	[19]
	Directed modularity		✓	✓						$\mathcal{O}(n^2 \log n)$	50	?	2008	[20]
	External Optimization		✓	✓						$\mathcal{O}(n^2 \log n)$	27k	?	2005	[21]
	Local modularity			✓				✓		$\mathcal{O}(n^2)$	400k	2.5M	2005	[22]
	Modularity Unfolding			✓						$\mathcal{O}(mk)$	118M	1B	2008	[23]
	Multislice modularity		✓	✓	✓		✓		✓	$\mathcal{O}(mkD)$	2k	?	2010	[24]
	MetaFac				✓		✓			$\mathcal{O}(mnD)$	?	2M	2009	[25]
	Variational Bayes		✓			✓				$\mathcal{O}(mk)$	115	613	2008	[26]
	$LA \rightarrow IS^2*$	✓	✓							$\mathcal{O}(mk+n)$	16k	?	2005	[27]
	Local Density		✓			✓		✓		$\mathcal{O}(nK \log n)$	108k	330k	2005	[28]
Bridge	Edge Betweenness		✓							$\mathcal{O}(m^2n)$	271	1k	2002	[4]
	CONGO*	✓								$\mathcal{O}(n \log n)$	30k	116k	2008	[29]
	L-Shell	✓						✓		$\mathcal{O}(n^3)$	77	254	2005	[30]
	Internal-External Degree	✓								$\mathcal{O}(n^2 \log n)$	775k	4.7M	2009	[31]
Diffusion	Label Propagation			✓		✓		✓		$\mathcal{O}(m+n)$	374k	30M	2007	[32]
	Node Colouring				✓				✓	$\mathcal{O}(ntk^2)$	2k	?	2007	[33]
	Kirchhoff	✓		✓						$\mathcal{O}(m+n)$	115	613	2004	[34]
	Communication Dynamic	✓	✓		✓			✓		$\mathcal{O}(mnt)$	160k	530k	2008	[35]
	GuruMine		✓		✓					$\mathcal{O}(TAn^2)$	217k	212k	2008	[8]
	DegreeDiscountIC		✓							$\mathcal{O}(k \log n + m)$	37k	230k	2009	[36]
	MMSB	✓	✓							$\mathcal{O}(nk)$	871	2k	2007	[37]
Close	Walktrap									$\mathcal{O}(mn^2)$	160k	1.8M	2006	[38]
	DOCS	✓								?	325k	1M	2009	[39]
	Infomap		✓	✓						$\mathcal{O}(m \log^2 n)$	6k	6M	2008	[40]
Structure	K-Clique	✓								$\mathcal{O}(m^{\frac{\ln m}{10}})$	20k	127k	2005	[3]
	S-Plexes Enumeration									$\mathcal{O}(mn)$	?	?	2009	[41]
	Bi-Clique	✓							✓	$\mathcal{O}(m^2)$	200k	500k	2008	[42]
	EAGLE	✓	✓	✓						$\mathcal{O}(3^{\frac{n}{3}})$	16k	31k	2009	[43]
Link	Link modularity	✓		✓					✓	$\mathcal{O}(2mk \log n)$	20k	127k	2009	[44]
	Link Jaccard*	✓		✓					✓	$\mathcal{O}(n\bar{K}^2)$	885k	5.5M	2010	[45]
NoD	Hybrid*	✓	✓	✓		✓				$\mathcal{O}(nk\bar{K})$	325k	1.5M	2010	[46]
	Multi-relational Regression			✓			✓			?	?	?	2005	[47]

Table 3: Resume of the community discovery methods.

p. 118703, 2008.

[21] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Physical Review E*, vol. 72, pp. 027104+, Aug 2005.

[22] A. Clauset, "Finding local community structure in networks," *Physical Review E*, vol. 72, p. 026132, 2005.

[23] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J.STAT.MECH.*, p. P10008, 2008.

[24] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J. Onnela, "Community Structure in Time-Dependent, Multiscale, and Multiplex Networks," *ArXiv e-prints*, Nov. 2009.

[25] Y.-R. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, and A. Kelliher, "Metafac: community discovery via relational hypergraph factorization," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 527–536, ACM, 2009.

[26] J. M. Hofman and C. H. Wiggins, "A bayesian approach to network modularity," *Physical Review Letters*, vol. 100, p. 258701, 2008.

[27] J. Baumes, M. Goldberg, and M. Magdon-ismail, "Efficient identification of overlapping communities," in *In IEEE International Conference on Intelligence and Security Informatics (ISI*, pp. 27–36, 2005.

[28] S. E. Schaeffer, "Stochastic local clustering for massive graphs," in *Proceedings of the Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-05)*, vol. 3518 of *Lecture Notes in Computer Science*, pp. 354–360, Springer-Verlag GmbH, 2005.

[29] S. Gregory, "A fast algorithm to find overlapping communities in networks," in *ECML PKDD '08: Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, (Berlin, Heidelberg), pp. 408–423, Springer-Verlag, 2008.

[30] J. Bagrow and E. Bollt, "A local method for detecting communities," *Physical Review E*, vol. 72, p. 046108, 2005.

[31] A. Lancichinetti, S. Fortunato, and J. Kertesz, "Detecting the overlapping and hierarchical community structure of complex networks," *New Journal of Physics*, vol. 11, p. 033015, 2009.

[32] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, p. 036106, 2007.

[33] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe, "A framework for community identification in dynamic social networks," in *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 717–726, ACM, 2007.

[34] F. Wu and B. A. Huberman, "Finding communities in linear time: a physics approach," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 38, no. 2, pp. 331–338, 2004.

[35] M. Goldberg, S. Kelley, M. Magdon-Ismail, K. Mertsalov, and W. A. Wallace, "Communication dynamics of blog networks," in *The 2nd SNA-KDD Workshop '08 (SNA-KDD'08)*, August 2008.

[36] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 199–208, ACM, 2009.

[37] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *JOURNAL OF MACHINE LEARNING RESEARCH*, vol. 9, p. 1981, 2007.

[38] P. Pons and M. Latapy, "Computing communities in large networks using random walks," *Journal of Graph Algorithms and Applications*, 2006.

[39] F. Wei, W. Qian, C. Wang, and A. Zhou, "Detecting overlapping community structures in networks," *World Wide Web*, vol. 12, no. 2, pp. 235–261, 2009.

[40] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Science*, vol. 105, pp. 1118–1123, Jan. 2008.

[41] C. Komusiewicz, F. Hüffner, H. Moser, and R. Niedermeier, "Isolation concepts for efficiently enumerating dense subgraphs," *Theor. Comput. Sci.*, vol. 410, no. 38-40, pp. 3640–3654, 2009.

[42] S. Lehmann, M. Schwartz, and L. K. Hansen, "Bi-clique communities," *PHYS.REV.*, vol. 78, p. 016108, 2008.

[43] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, "Detect overlapping and hierarchical community structure in networks," *PHYSICA A*, vol. 388, p. 1706, 2009.

[44] T. S. Evans and R. Lambiotte, "Line graphs, link partitions and overlapping communities," *Physical Review E*, vol. 80, p. 016105, 2009.

[45] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multi-scale complexity in networks," *Nature*, 2010.

[46] T. Eliassi-Rad, K. Henderson, S. Papadimitriou, and C. Faloutsos, "A hybrid community discovery framework for complex networks," in *SIAM Conference on Data Mining*, 2010.

[47] D. Cai, Z. Shao, X. He, X. Yan, and J. Han, "Community mining from multi-relational networks," in *Proceedings of the 2005 European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05)*, (Porto, Portugal), 2005.

[48] J. Rissanen, "Modelling by the shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[49] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell system technical journal*, vol. 49, no. 1, pp. 291–307, 1970.

[50] A. Pothen, H. D. Simon, and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM J. Matrix Anal. Appl.*, vol. 11, no. 3, pp. 430–452, 1990.

[51] R. Lambiotte, J. Delvenne, and M. Barahona, "Laplacian Dynamics and Multiscale Modular Structure in Networks," *ArXiv e-prints*, Dec. 2008.

[52] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," vol. 74, pp. 016110–+, July 2006.

[53] L. Kaufman and P. J. Rousseeuw, "Finding groups in data: An introduction to cluster analysis," John Wiley, 1990.

[54] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 89–98, ACM, 2003.

[55] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos, "Fully automatic cross-associations," in *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 79–88, ACM, 2004.

[56] B. Long, Z. M. Zhang, X. Wú, and P. S. Yu, "Spectral clustering for multi-type relational data," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*, (New York, NY, USA), pp. 585–592, ACM, 2006.

[57] S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: A survey," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 1, no. 1, pp. 24–45, 2004.

[58] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, August 1991.

[59] P. Grunwald, "A tutorial introduction to the minimum description length principle," 2004.

[60] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Mach. Learn.*, vol. 2, pp. 139–172, September 1987.

[61] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pp. 81–92, VLDB Endowment, 2003.

[62] H. Koga, T. Ishibashi, and T. Watanabe, "Fast agglomerative hierarchical clustering algorithm using locality-sensitive hash-

ing," *Knowl. Inf. Syst.*, vol. 12, no. 1, pp. 25–53, 2007.

[63] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Facetnet: a framework for analyzing communities and their evolutions in dynamic networks," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*, (New York, NY, USA), pp. 685–694, ACM, 2008.

[64] M.-S. Kim and J. Han, "A particle-and-density based evolutionary clustering method for dynamic networks," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 622–633, 2009.

[65] B. Gao, T.-Y. Liu, X. Zheng, Q.-S. Cheng, and W.-Y. Ma, "Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering," in *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, (New York, NY, USA), pp. 41–50, ACM, 2005.

[66] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh, "Clustering with bregman divergences," in *Journal of Machine Learning Research*, pp. 234–245, 2004.

[67] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A generalized maximum entropy approach to bregman co-clustering and matrix approximation," in *In KDD*, pp. 509–514, 2004.

[68] H. Cho, I. Dhillon, Y. Guan, and S. Sra, "Minimum sum squared residue co-clustering of gene expression data," 2004.

[69] M. Mcpherson, L. S. Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[70] L. Tang, S. Rajan, and V. K. Narayanan, "Large scale multi-label classification via metalabeler," in *WWW '09: Proceedings of the 18th international conference on World wide web*, (New York, NY, USA), pp. 211–220, ACM, 2009.

[71] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," 2004.

[72] L. Tang and H. Liu, "Scalable learning of collective behavior based on sparse social dimensions," in *CIKM*, 2009.

[73] L. Tang and H. Liu, "Uncovering cross-dimension group structures in multi-dimensional networks," in *SDM workshop on Analysis of Dynamic Networks*, 2009.

[74] G. H. Golub and C. F. V. Loan, "Matrix computations," (Baltimore, MD, USA), 1989.

[75] J. Kettenring, "Canonical analysis of several sets of variables," *Biometrika*, vol. 58, no. 3, pp. 433–451, 1971.

[76] J. Pitman, "Combinatorial stochastic processes," 2002.

[77] S. Papadimitriou, A. Gionis, P. Tsaparas, R. A. Visnen, H. Mannila, and C. Faloutsos, "Parameter-free spatial data mining using mdl," *Data Mining, IEEE International Conference on*, vol. 0, pp. 346–353, 2005.

[78] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu, T. J. Watson, and R. Ctr, "A framework for projected clustering of high dimensional data streams," in *In Proc. of VLDB*, pp. 852–863, 2004.

[79] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu, "Graphscope: parameter-free mining of large time-evolving graphs," in *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 687–696, ACM, 2007.

[80] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, p. 026113, 2004.

[81] B. H. Good, Y.-A. de Montjoye, and A. Clauset, "The performance of modularity maximization in practical contexts," 2009.

[82] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, p. 036104, 2006.

[83] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, p. 066133, 2004.

[84] M. E. J. Newman, "Modularity and community structure in networks," *PROC.NATL.ACAD.SCI.USA*, vol. 103, p. 8577, 2006.

[85] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri, "Extending the definition of modularity to directed graphs with overlapping communities," *J.STAT.MECH.*, p. P03024, 2009.

[86] P. Bak and K. Sneppen, "Punctuated equilibrium and criticality in a simple model of evolution," *Phys. Rev. Lett.*, vol. 71, pp. 4083–4086, Dec 1993.

[87] S. Boettcher and A. G. Percus, "Optimization with extremal dynamics," *Phys. Rev. Lett.*, vol. 86, pp. 5211–5214, Jun 2001.

[88] A. Arenas, J. Duch, A. Fernandez, and S. Gomez, "Size reduction of complex networks preserving modularity," *New Journal of Physics*, vol. 9, p. 176, 2007.

[89] R. Guimera and L. A. Amaral, "Cartography of complex networks: modules and universal roles," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, February 2005.

[90] L. C. Freeman, "Centered graphs and the construction of ego networks," *Mathematical Social Sciences*, no. 3, pp. 291–304, 1982.

[91] K. Wakita and T. Tsurumi, "Finding community structure in mega-scale social networks: [extended abstract]," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*, (New York, NY, USA), pp. 1275–1276, ACM, 2007.

[92] P. Boldi, M. Santini, and S. Vigna, "A large time-aware web graph," *SIGIR Forum*, vol. 42, no. 2, pp. 33–38, 2008.

[93] M. L. Wallace, Y. Gingras, and R. Duhon, "A new approach for detecting scientific specialties from raw cocitation networks," *J. Am. Soc. Inf. Sci. Technol.*, vol. 60, no. 2, pp. 240–246, 2009.

[94] M. J. Barber, "Modularity and community detection in bipartite networks," *arXiv*, vol. 76, pp. 066102–+, Dec. 2007.

[95] S. Gomez, P. Jensen, and A. Arenas, "Analysis of community structure in networks of correlated data," *ArXiv e-prints*, Dec. 2008.

[96] V. A. Traag and J. Bruggeman, "Community detection in networks with positive and negative links," *arXiv*, vol. 80, pp. 036115–+, Sept. 2009.

[97] M. N. L. Narasimhan, "Principles of continuum mechanics," John Wiley and Sons, New York, 1993.

[98] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, pp. 455–500, September 2009.

[99] L. D. Lathauwer and A. de Baynast, "Blind deconvolution of ds-cdma signals by means of decomposition in rank-(1, l, l) terms," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1562–1571, 2008.

[100] A. N. Langville and W. J. Stewart, "A kronecker product approximate preconditioner for sans," 2003.

[101] J. Sun, S. Papadimitriou, and P. Yu, "Window-based tensor analysis on high-dimensional and multi-aspect streams," pp. 1076 –1080, dec. 2006.

[102] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: dynamic tensor analysis," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 374–383, ACM, 2006.

[103] B. Bader, R. Harshman, and T. Kolda, "Temporal analysis of semantic graphs using asalsan," pp. 33 –42, oct. 2007.

[104] E. Acar, D. M. Dunlavy, and T. G. Kolda, "Link prediction on evolving data using matrix and tensor factorizations," in *LDMTA'09: Proceeding of the ICDM'09 Workshop on Large Scale Data Mining Theory and Applications*, pp. 262–269, IEEE Computer Society Press, December 2009.

[105] Y. Chi, S. Zhu, Y. Gong, and Y. Zhang, "Probabilistic polyadic factorization and its application to personalized recommendation," in *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, (New York, NY, USA), pp. 941–950, ACM, 2008.

[106] T. G. Kolda and J. Sun, "Scalable tensor decompositions for multi-aspect data mining," in *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, (Washington, DC, USA), pp. 363–372, IEEE Computer Soci-

ety, 2008.
[107] M. B. Hastings, "Community detection as an inference problem," *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 74, no. 3, p. 035102, 2006.
[108] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," pp. 105–161, 1999.
[109] J. Baumes, M. K. Goldberg, M. Magdon-Ismail, and W. A. Wallace, "Discovering hidden groups in communication networks," in *ISI*, pp. 378–389, 2004.
[110] J. Baumes, M. K. Goldberg, M. S. Krishnamoorthy, M. Magdon-Ismail, and N. Preston, "Finding communities by clustering a graph into overlapping subgraphs," in *IADIS AC*, pp. 97–104, 2005.
[111] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," 1998.
[112] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science, Number 4598*, vol. 220, 4598, pp. 671–680, 1983.
[113] R. A. Hanneman and M. Riddle, "Introduction to social network methods," 2005.
[114] D. R. White, F. Harary, M. Sobel, and M. Becker, "The cohesiveness of blocks in social networks: Node connectivity and conditional density," 2001.
[115] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, pp. 35–41, March 1977.
[116] Q. Yang and S. Lonardi, "A parallel edge-betweenness clustering tool for protein-protein interaction networks," *Int. J. Data Min. Bioinformatics*, vol. 1, no. 3, pp. 241–247, 2007.
[117] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, pp. 2658–2663, March 2004.
[118] I. Vragović and E. Louis, "Network community structure and loop coefficient method," *Phys. Rev. E*, vol. 74, p. 016105, Jul 2006.
[119] S. Gregory, "An algorithm to find overlapping community structure in networks," in *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)*, pp. 91–102, Springer-Verlag, September 2007.
[120] S. Gregory, "Finding overlapping communities using disjoint community detection algorithms," in *Complex Networks: CompleNet 2009*, pp. 47–61, Springer-Verlag, May 2009.
[121] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, p. 167, 2003.
[122] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *ACM Trans. Web*, vol. 1, no. 1, p. 5, 2007.
[123] S. Fortunato, V. Latora, and M. Marchiori, "A method to find community structures based on information centrality," 2004.
[124] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group formation in large social networks: membership, growth, and evolution," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 44–54, ACM, 2006.
[125] S. Gregory, "Finding overlapping communities in networks by label propagation," 2009.
[126] T. Y. Berger-Wolf and J. Saia, "A framework for analysis of dynamic social networks," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 523–528, ACM, 2006.
[127] C. Tantipathananandh and T. Berger-Wolf, "Constant-factor approximation algorithms for identifying dynamic communities," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 827–836, ACM, 2009.
[128] N. A. Alves, "Unveiling community structures in weighted networks," *Physical Review E*, vol. 76, p. 036101, 2007.
[129] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, p. 509, 1999.
[130] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek, "Evolution of the social network of scientific collaborations," *PHYSICA A*, vol. 311, p. 3, 2002.
[131] G. Kossinets and D. J. Watts, "Empirical analysis of an evolving social network," *Science*, vol. 311, no. 5757, pp. 88–90, 2006.
[132] N. Agarwal, H. Liu, L. Tang, and P. S. Yu, "Identifying the influential bloggers in a community," in *WSDM '08: Proceedings of the international conference on Web search and web data mining*, (New York, NY, USA), pp. 207–218, ACM, 2008.
[133] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 420–429, ACM, 2007.
[134] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, ACM, 2003.
[135] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 207–216, ACM, 1993.
[136] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, "Examiner: Optimized level-wise frequent pattern mining with monotone constraints," *Data Mining, IEEE International Conference on*, vol. 0, p. 11, 2003.
[137] A. Goyal, B.-W. On, F. Bonchi, and L. V. S. Lakshmanan, "Gurumine: A pattern mining system for discovering leaders and tribes," *Data Engineering, International Conference on*, vol. 0, pp. 1471–1474, 2009.
[138] Y. W. Teh, D. Newman, and M. Welling, "A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation," in *Advances in Neural Information Processing Systems*, vol. 19, 2007.
[139] T. Minka and J. Lafferty, "Expectation-propagation for the generative aspect model," in *In Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pp. 352–359, Morgan Kaufmann, 2002.
[140] E. A. Erosheva and S. E. Fienberg, "Bayesian mixed membership models for soft clustering and classification," in *Classification - The ubiquitous challenge*, Springer Berlin Heidelberg, 2005.
[141] F. Fouss, A. Pirotte, J. michel Renders, and M. Saerens, "A novel way of computing dissimilarities between nodes of a graph, with application to collaborative filtering," 2004.
[142] H. Zhou and R. Lipowsky, "Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities," in *International Conference on Computational Science*, pp. 1062–1069, 2004.
[143] F. Wei, C. Wang, L. Ma, and A. Zhou, "Detecting overlapping community structures in networks with global partition and local expansion," *Progress in WWW Research and Development*, 2008.
[144] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, (New York, NY, USA), pp. 81–90, ACM, 2004.
[145] M. Berlingerio, F. Bonchi, B. Bringmann, and A. Gionis, "Mining graph evolution rules," in *ECML/PKDD (1)*, pp. 115–130, 2009.
[146] S. Nijssen and J. N. Kok, "A quickstart in frequent structure mining can make a difference," in *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 647–652, ACM, 2004.

[147] M. Kuramochi and G. Karypis, "Finding frequent patterns in a large sparse graph*," *Data Min. Knowl. Discov.*, vol. 11, no. 3, pp. 243–271, 2005.
[148] K. Saito and T. Yamada, "Extracting communities from complex networks by the k-dense method," in *ICDMW '06: Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops*, (Washington, DC, USA), pp. 300–304, IEEE Computer Society, 2006.
[149] H. Ito, K. Iwama, and T. Osumi, "Linear-time enumeration of isolated cliques," in *ESA*, pp. 119–130, 2005.
[150] H. Ito and K. Iwama, "Enumeration of isolated cliques and pseudo-cliques," in *ACM Transactions on Algorithms*, 2008.
[151] B. Balasundaram, S. Butenko, I. Hicks, and S. Sachdeva, "Clique relaxations in social network analysis: The maximum k-plex problem," in *Operations Research*, 2009.
[152] N. Nishimura, P. Ragde, and D. M. Thilikos, "Fast fixed-parameter tractable algorithms for nontrivial generalizations of vertex cover," *Discrete Appl. Math.*, vol. 152, no. 1-3, pp. 229–245, 2005.
[153] T. Uno, M. Kiyomi, and H. Arimura, "Lcm ver.3: collaboration of array, bitmap and prefix tree for frequent itemset mining," in *OSDM '05: Proceedings of the 1st international workshop on open source data mining*, (New York, NY, USA), pp. 77–86, ACM, 2005.
[154] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, no. 9, pp. 575–577, 1973.
[155] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proceedings of the National Academy of Science*, vol. 104, pp. 36–41, Jan. 2007.
[156] K. Henderson and T. Eliassi-Rad, "Applying latent dirichlet allocation to group discovery in large graphs," in *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, (New York, NY, USA), pp. 1456–1461, ACM, 2009.
[157] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.
[158] T. Griffiths, "Gibbs sampling in the generative model of latent dirichlet allocation," tech. rep., Stanford University, 2002.
[159] A. Bjorck, "Numerical methods for least squares problems," (Philadelphia), SIAM, 1996.
[160] T. Hastie, R. Tibshirani, and J. H. Friedman, "The elements of statistical learning," Springer, 2003.
[161] M. Newman, "Detecting community structure in networks," *Eur. Phys. J. B*, vol. 38, pp. 321–330, mar 2004.
[162] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Comput. Surv.*, vol. 38, no. 1, p. 2, 2006.
[163] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas, "Comparing community structure identification," 2005.
[164] S. Fortunato and C. Castellano, "Community structure in graphs," 2007.
[165] M. A. Porter, J.-P. Onnela, and P. J. Mucha, "Communities in networks," *NOTICES OF THE AMERICAN MATHEMATICAL SOCIETY*, vol. 56, p. 1082, 2009.
[166] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27 – 64, 2007.
[167] J. P. Scott, *Social Network Analysis: A Handbook*. SAGE Publications, 2000.
[168] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, vol. 78, p. 046110, 2008.