Partial and Overlapping Community Detection in Multiplex Social Networks

Nazanin Afsarmanesh Tehrani[1,2] and Matteo Magnani[1,2(✉)]

[1] Gavagai, Stockholm, Sweden
nazanin.afsarmanesh@gavagai.se
[2] InfoLab, Department of Information Technology,
Uppsala University, Uppsala, Sweden
matteo.magnani@it.uu.se

Abstract. We extend the popular clique percolation method to multiplex networks. Our extension requires to rethink the basic concepts on which the original clique percolation algorithm is based, including cliques and clique adjacency, to handle the presence of multiple types of ties. We also provide an experimental characterization of the communities that our method can identify.

Keywords: Multiplex network · Community detection · Overlapping Partial · Clique · Clique percolation

1 Introduction

Community detection is one of the most popular social network analysis tasks, for which a large number of algorithms have been developed [1,2]. The number of existing methods is not only justified by the importance of this task, but also by the absence of a unique definition of what a community is: different algorithms are often designed to identify different types of communities, and it is thus practically important for a social network analyst to have a toolbox with alternative algorithms.

The clique percolation method [3] is based on the intuition that the presence of a community can be observed in a social network through the presence of cliques, that is, sets of actors who are all adjacent to each other. This method has a set of features that make it well-suited to the discovery of communities in social networks: (1) it allows to specify how much connectivity is necessary to recognize the presence of a community (minimum clique size k), (2) it allows the same actor to be present in multiple communities (overlapping), and (3) it does not force all actors to be part of a community (partial).

However, this approach is currently only defined for networks with one single type of tie, while in social networks individuals often interact with different groups of people, such as friends, colleagues, and family, and this determines multiple types of relationships, including multiple types of ties between the same

S. Staab et al. (Eds.): SocInfo 2018, LNCS 11186, pp. 15–28, 2018.
https://doi.org/10.1007/978-3-030-01159-8_2

pairs of actors. Therefore, in this paper we extend this approach to deal with multiplex networks, to add clique percolation and the special features of its communities to the multiplex network analysis toolbox.

To increase the expressiveness of models based on simple graphs, multiplex networks [4,5] and heterogeneous information networks [6] have been introduced, among other models, allowing vertices and edges to have different types and to be described by multiple attributes. A specific type of multiplex system, called *multiplex network*, is characterized by vertices that can be connected through multiple types of edges and has been used for almost one century in the field of social network analysis [7,8]. For what concerns community detection, people have developed several methods to find overlapping communities in simple graphs [9]. Clique based methods [3,10,11], fuzzy community detection algorithms [12,13] and link partitioning methods [14,15] are examples of overlapping clustering algorithms.

To the best of our knowledge, these two lines of research have been almost completely distinct so far: while we have methods for overlapping community detection on simple graphs [9], and we have partitioning community detection methods for multiplex networks [16], the problem of detecting overlapping communities in multiplex networks has only been addressed by three methods based on different definitions of community, including the one presented in this paper.

Some approaches convert the multiplex network to a simple graph [17–20], and then employ existing methods. However, this may result in information loss, because the clustering algorithm would not know whether a set of edges belongs to the same or to different edge types, potentially leading to the discovery of communities scattered across a large number of edge types and weak ties.

2 Preliminaries

Multiplex networks are graph-based data structures where the same pair of vertices can be connected by different types of edges.

Definition 1 (Multiplex network). *Given a set of vertices V and a set of edge types $\mathcal{L}$, a multiplex network is defined as a triple $M = (V, E, \mathcal{L})$ where $E \subseteq V \times V \times \mathcal{L}$.*

The objective of the method introduced in this paper is to identify a set of communities, also known as community structure. Here we define a community as a set of vertices combined with a set of edge types, to indicate that for example a group of actors can be part of a *friend and family* community without being part of a *work* or *sport* community.

Definition 2 (Community). *Given a multiplex network $M = (V, E, \mathcal{L})$, a community is defined as a set $C \subseteq 2^V \times 2^{\mathcal{L}}$.*

This definition of community allows both vertices and edge types to overlap across communities, and does not force all vertices and edge types to be included

in any community. This is a special case of the type of community that is used for example by the generalized Louvain [21] method, which is defined as a set of pairs $(v, l) \in V \times \mathcal{L}$. However, differently from the generalized Louvain method and in line with the multiplex community detection methods Abacus [22] and Infomap [23], our multiplex clique percolation approach may find communities that are overlapping on the same edge type[1] and also partial.

The clique percolation method was introduced by Palla et al. in 2005 [3]. For a given k, CPM builds up communities from k-cliques, that is, complete subgraphs in the network with k vertices. Two k-cliques are said to be adjacent if they share $k-1$ vertices. A *k-clique community* is defined as a maximal union of k-cliques that can be reached from each other through a series of adjacent k-cliques. In general, if the number of links is increased above some critical point, a giant community would appear that covers a vast part of the system. Therefore, k is chosen as the smallest value where no giant community appears. CPM allows overlapping communities in a natural way as a vertex can belong to multiple cliques. Figure 1 shows an example of how CPM works. Given an input graph, first maximal cliques are identified, then adjacent cliques are grouped together to form communities.

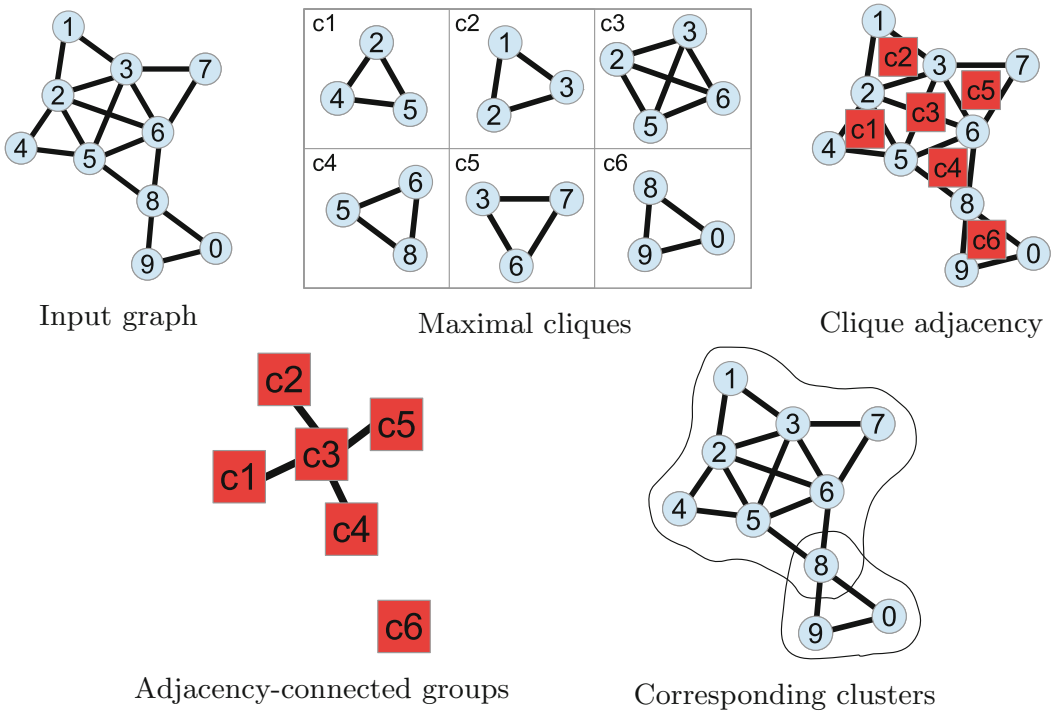

Fig. 1. A step-by-step view of the original clique percolation method

[1] Using the generalized Louvain method, every pair (v, l) is included in exactly one community.

3 Multiplex Clique Percolation

Our extended CPM algorithm for multiplex networks (CPM^M), of which we describe an implementation in the next section, follows the same main general steps of CPM. However, the concepts on which it is based must be extended to multiplex networks. In particular, we need to define what a clique on multiple edge types is, when two multiplex cliques can be considered adjacent, and how adjacent cliques should be grouped to build communities.

3.1 Cliques on Multiple Edge Types

While a clique on a simple graph is a well understood structure, defined as a set of vertices that are all adjacent to each other, the same concept can be extended in different ways for multiplex networks depending on how multiple edge types are allowed to contribute to the clique connectivity. Considering a specific set of edge types, we might require that a clique contains all the possible edges on all these edge types. In other words, a clique is formed by a combination of cliques on individual edge types. We refer to this type of cliques as AND-cliques.

Definition 3 (k-m-AND-clique). *Let L_{ij} be the set of edge types between vertices i and j. We define a k-m-AND-clique as a subgraph in the multiplex network with k vertices that includes a combination of at least m different k-cliques from m different edge types. In other words, a k-m-AND-clique is a subgraph with k vertices C where*

$$|\bigcap_{i,j \in C} L_{ij}| \geq m \tag{1}$$

Similar to the case of cliques on simple graphs, we can define a concept of maximality for cliques in multiplex networks, where neither k nor m can be increased.

3.2 Adjacency and Communities

When cliques may exist on different edge types, the concept of adjacency should also consider this aspect.

To illustrate why, consider a definition of adjacency where k-m-AND-cliques only need to share $k-1$ vertices to be considered adjacent. As shown in Fig. 2 (lhs) adjacent cliques do not necessarily share any edge types on all pairs and

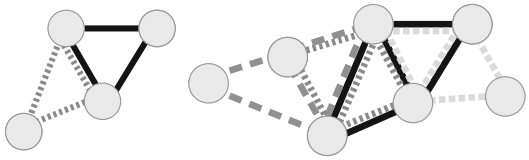

Fig. 2. Adjacent cliques

they might share edge types only on their common pairs of vertices. It is worth noting that more diversity among the edge types in external connections of adjacent cliques results in denser internal connectivity. In addition, cliques at distance one still have to share some edge types on some of their pairs of vertices, as in the figure, but when the distance between cliques becomes greater than one, as in Fig. 2 (rhs), they may end up having completely different edge labels. To enforce uniformity among edge types throughout the whole community, we thus need more constraints than what we can define at the level of clique adjacency.

Definition 4 (Multiplex clique-based community). *A multiplex clique-based $[(k-m)$-AND-clique$]_{(m',m'')}$ community is the maximal union of m'-adjacent k-m-AND-cliques where all cliques share at least m'' edge types on all of their pairs of vertices.*

Please notice that this is a very general definition, and in practice we can just use two parameters: k and $m(=m, m', m'')$.

3.3 Algorithm

In Fig. 3 we have sketched an algorithm to detect communities according to our definitions, where without loss of generality we will assume that $m = m' = m''$. The details of the algorithm, including pseudo-code, are given in the appendix [24], and the algorithm is available in the *multinet* library[2].

The algorithm is divided into three parts, as in the original method: finding cliques, which can be done using an extension of Bron–Kerbosch's algorithm, building the adjacency graph, and extracting communities.

In a simple graph, each clique is included in exactly one community, therefore, communities can be identified from a clique-clique overlap matrix (see [3] for the details). However, this statement is not necessarily true for k-m-AND-cliques and the corresponding communities. Because of the more complicated relations between cliques, instead of the overlap matrix used in the original method we generate an adjacency graph as in Fig. 3. In the graph we have indicated for each vertex the edge types where the corresponding clique is defined.

Two cliques can be included in at least one community if: (1) there exists a path between the corresponding vertices in the clique-adjacency graph, and (2) for all vertices in the path the corresponding cliques share at least m edge types on all of their pairs. Therefore, each community corresponds to a maximal tree in the clique-adjacency graph where condition (2) holds.

Figure 3 shows all the maximal trees from our clique-adjacency graph for $m \geq 1$. As we see, clique $c4$ can be included in three communities: C1, C2 and C3. No new clique can be added to these sets without reducing the value of m for which the cliques' constraint holds. As an example, community C4 satisfies the cliques' constraint for $m = 2$. Adding any adjacent clique to it, like c3, c5 and c6, the constraint would no longer hold for $m = 2$ because only one edge type would be common for both cliques. In Fig. 3 for each maximal tree we have

[2] https://CRAN.R-project.org/package=multinet.

indicated the edge types where the constraint is satisfied, and we also show all communities in this example for $m \geq 1$.

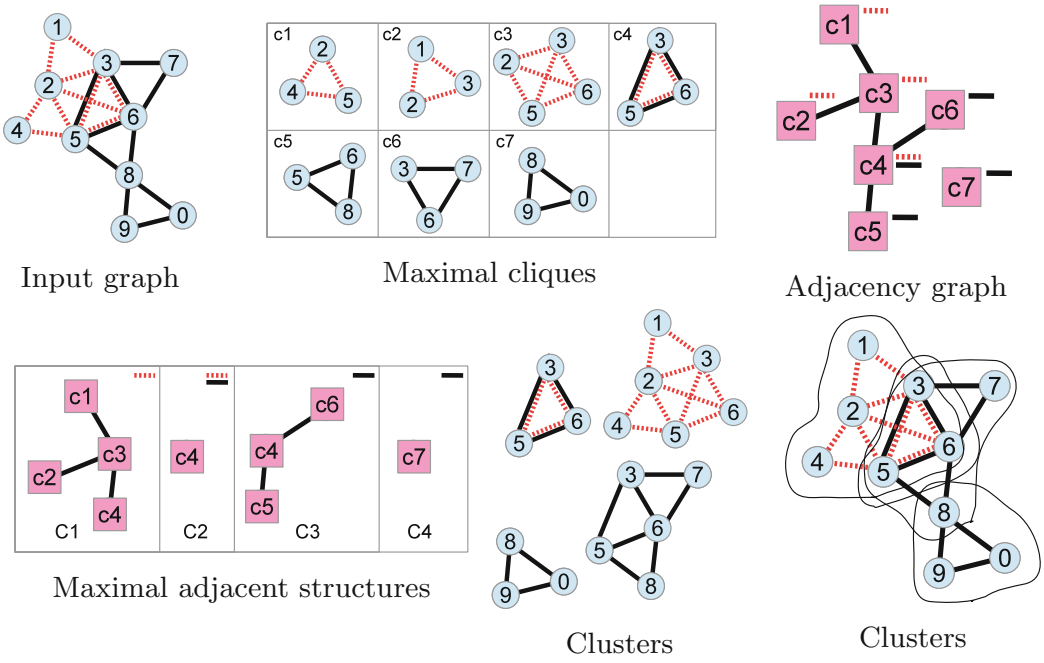

Fig. 3. A step-by-step view of our method

4 Experiments

4.1 Qualitative Analysis

Table 1 shows the result of our experiments on a real multiplex network of five edge types [27] coding five types of relationships inside a university department (Facebook, Work, Lunch, Co-authorship, Leisure) on 61 employees, with k = 3 and m = 2; we only report communities that share at least two edge types, not those that can be found on single edge types one by one. Our algorithm finds 26 communities where the size of communities vary from 3 to 12 vertices. The edge types Facebook, Work and Lunch which are denser than Leisure and Coauthor appear more frequently among the communities. As expected, some of the 61 vertices in the network are not included in any community.

It can be realized from this table that we can identify different types of overlapping vertices among communities in multiplex networks. If we consider the communities on the two edge types Lunch and Work, e.g. C3 and C5, the structure of communities are more or less similar to the case of single networks as the communities are well-separated with a limited number of overlapping vertices. This can be considered as a general rule for the case where the sets of contributing edge types are the same. On the other hand, we have two communities C21 and C22 where the sets of contributing edge types are not

exactly the same. These two communities have 5 overlapping vertices while C21 has only 6 vertices. This is in fact a consequence of what we experienced earlier, that is, cliques can be included in different communities on different combinations of edge types. In addition, we can also have hierarchical community structures: small communities with a larger number of contributing edge types inside larger communities with a smaller number of edge types, such as C16 and C17.

These overlapping structures identified by our method, with core communities built on many edge types and larger communities including more peripheral vertices on less edge types, are compatible with the type of communities often observed in online social networks, as studied by [28].

Table 1. Communities identified in a real data set for k = 3 and m = 2

#	Vertices	Edge types
C01	U107 U1 U29 U32	Facebook lunch
C02	U124 U109 U47	Facebook lunch
C03	U130 U134 U4	Lunch work
C04	U91 U65 U72	Lunch work leisure
C05	U4 U112 U68 U141	Lunch work
...		
C16	U59 U91 U110	Facebook lunch work leisure
C17	U59 U91 U110 U113 U138	Work leisure
...		
C21	U109 U18 U3 U54 U76 U79	Facebook lunch
C22	U109 U126 U3 U54 U76 U79 U90	Lunch leisure
C23	U106 U118 U41	Lunch work leisure
C24	U123 U59 U71 U91 U130 U47 + **6**	Facebook work
...		

4.2 A Quantitative Characterization of the Obtained Communities

Table 2 shows various statistics describing the communities found in a multiplex network about co-authorship in the arxiv repository on 13 research fields, corresponding to 13 edge types, using different parameters.

From the statistics we can see that the algorithm actually finds overlapping and partial communities. Notice that this is expected, but not necessary, as depending on the data and parameters not all the types of communities expected in theory necessarily exist or are found, a typical example being the generalized Louvain algorithm where for many values of the interlayer coupling parameter vertices are forced into the same community for all edge types.

We can also see a significant overlapping inside edge types (%over_et), indicating for example that the same individual can part of multiple communities inside the same research field.

Table 2. Quantitative characterization of the community structures identified by the algorithm. #c: number of communities. sc1: size of largest community. sc2/1: relative size of second largest community wrt largest community. %in_com: portion of vertices included in at least one community. %all_et portion of vertices included in the same community for all edge types. %over_v: portion of vertices included in more than one community on different edge types. %over_et: portion of vertices included in more than one community on the same edge type

k	m	#c	sc1	sc2/1	%in_com	%all_et	%over_v	%over_et
3	1	4278	1562	0.96	0.84	0.28	0.35	0.36
3	3	773	144	0.96	0.26	0.31	0.31	0.32
3	5	82	42	0.95	0.04	0.28	0.25	0.25
5	1	1320	70	0.86	0.36	0.47	0.22	0.20
5	3	180	56	0.80	0.10	0.39	0.17	0.17
5	5	9	42	0.95	0.01	0.24	0.20	0.13
7	1	248	60	0.93	0.11	0.60	0.10	0.09
7	3	33	56	0.80	0.03	0.44	0.11	0.08
7	5	1	40	0.00	0.00	0.12	0.00	0.00

The ratio sc2/1 shows that no single giant community is identified with the tested settings. In addition, the number of communities decreases when k increases and when m increases. This, in addition to the ability to extract denser communities, can also be exploited to reduce computation time.

5 Discussion

In this paper we have extended the CPM method to identify overlapping communities in multiplex networks. We have first focused on the formal definition of the method, discussing how to extend existing concepts to the multiplex context, defined an algorithm and studied its empirical behavior on real datasets. Some interesting aspects emerge from the formal definition of the method and from our experiments.

The attempt to keep communities homogeneous results in a phenomenon not visible when single graphs and the original method are used. While in CPM the same vertex can belong to multiple communities, in CPM^M whole cliques can belong to different communities, as demonstrated in our working example. This suggests that the type of overlapping produced using our approach enables the identification of some kind of hierarchical community structures.

As a final consideration, if some edge types are denser than others they may end up appearing more often in the found communities. A possible way to reduce the prominence of some edge types is to then remove these edge types and run the method again for decreased k and m, to find more communities on other edge types. However, evaluating this approach requires a more extensive qualitative analysis than the one fitting this paper, and we leave it to future work.

Acknowledgments. This work was partially funded by the European Union's Horizon 2020 research and innovation program under grant agreement No. 732027.

A Algorithm

The algorithm is divided into three parts, as in the original method: finding cliques, building the adjacency graph, and extracting communities.

A.1 Finding Cliques

Our algorithms starts by locating all maximal k-m-AND-cliques. Algorithm 1 is an extension of Bron–Kerbosch's algorithm. It is a recursive algorithm where the recursive step takes a clique A as input and returns all maximal k-m-AND-cliques containing A that can be constructed using vertices in B, with $k \geq \overline{k}$ and $m \geq \overline{m}$. In this way, given a multiplex network $M = (V, E, \mathcal{L})$, a call to find-cliques$(\{\}, V, \{\}, \overline{k}, \overline{m})$ with $\overline{k} > 1$ and $\overline{m} > 0$ returns all maximal cliques in M with $k \geq \overline{k}$ and $m \geq \overline{m}$.

The algorithm works by updating two sets: one containing vertices that can be used to extend the currently processed clique, and one to keep track of already examined cliques. More precisely, the parameter B is a set of vertices such that, for every vertex $n \in B$, $A \cup \{n\}$ is a previously unseen clique on at least $\overline{m}$ edge types. Whenever a vertex from B is used to extend the clique A, then B is updated by removing those vertices that are no longer connected to all vertices in the new clique A'. C is the same as B, but containing those vertices that have already been examined by the algorithm during some previous iteration, so that no duplicates are produced. Given a set of vertices A we notate the set of edge types where the vertices in A form a clique as $L(A)$, and the number of vertices in A as $S(A)$. Therefore, if $|L(A)| = 0$ then A is not a clique on any edge type. We also define $\max(\emptyset) = 0$.

As an example, assume we call find-cliques$(\{\}, \{0, 1, \ldots, 9\}, \{\}, 3, 1)$ on the network in Fig. 3. The algorithm would then start exploring one of the vertices in B, let us say 5. The new call will thus include in B only those vertices that can still form a clique on at least one edge type when joined with 5: find-cliques$(\{5\}, \{2, 3, 4, 6, 8\}, \{\}, 3, 1)$. Let us now assume that at the next iteration 3 is added to the current clique: find-cliques$(\{3, 5\}, \{2, 6\}, \{\}, 3, 1)$, and then 6: find-cliques$(\{3, 5, 6\}, \{2\}, \{\}, 3, 1)$. At this point $S(A)$ is 3, satisfying the minimum clique size, and $|L(\{3, 5, 6\})| = 2 > \max(\{|L(A \cup \{b\})| : b \in B\}) = \max(\{|L(\{2, 3, 5, 6\})|\}) = 1$. In fact, $\{3, 5, 6\}$ is a clique on two edge types, while $\{2, 3, 5, 6\}$ is a clique on only one edge type. Therefore, the current clique is returned as a maximal one (c4 in Fig. 3). At the next iteration, find-cliques$(\{2, 3, 5, 6\}, \{\}, \{\}, 3, 1)$ is called and clique c3 is returned. At some later point, the algorithm would call find-cliques$(\{2, 3, 5\}, \{\}, \{6\}, 3, 1)$, not returning any clique because $C = \{6\}$ indicates that this path has already been explored, and find-cliques$(\{4, 5\}, \{2\}, \{3\}, 3, 1)$, ultimately leading to the discovery of clique c1, and so on.

Algorithm 1. find-cliques$(A, B, C, \overline{k}, \overline{m})$ returns all maximal k-m-AND-cliques containing A that can be constructed using vertices in B, with $k \geq \overline{k}$ and $m \geq \overline{m}$

Input: A a clique
Input: B a set of vertices n such that $A \cup \{n\}$ is also a clique
Input: C a set of vertices n such that $A \cup \{n\}$ has already been processed by the algorithm before
Input: $\overline{k}$ minimum number of vertices to output a clique
Input: $\overline{m}$ minimum number of layers to output a clique
1: **if** $S(A) \geq \overline{k} \wedge \max(\{|L(A \cup \{b\})| : b \in B\}) < |L(A)| \wedge \max(\{|L(A \cup \{c\})| : c \in C\}) < |L(A)|$ **then**
2: OUTPUT A
3: **end if**
4: **for** $b \in B$ **do**
5: $A' = A \cup \{b\}$
6: $B = B \setminus \{b\}$
7: $B' = \{b' \in B : |L(A' \cup \{b'\})| \geq \overline{m}\}$
8: $C' = \{c' \in C : |L(A' \cup \{c'\})| \geq \overline{m}\}$
9: find-cliques$(A', B', C', \overline{k}, \overline{m})$
10: $C = C \cup \{b\}$
11: **end for**

A.2 Clique-Adjacency Graph

In a simple graph, each clique is included in exactly one community, therefore, communities can be identified from a clique-clique overlap matrix (see [3] for the details). However, this statement is not necessarily true for k-m-AND-cliques and the corresponding communities. Because of the more complicated relations between cliques, instead of the overlap matrix used in the original method we generate an adjacency graph as the one represented in Fig. 3, where each vertex of the graph corresponds to a maximal clique and an edge between two vertices indicates that the corresponding cliques share at least k vertices and at least m edge types on all of their pairs of vertices. In the graph we have indicated for each vertex the edge types where the corresponding clique is defined. In the following we refer to this graph as clique-adjacency graph.

A.3 From the Clique-Adjacency Graph to Communities

As previously mentioned, each clique can be included in different communities with different combinations of its adjacent cliques. Here our objective is using the adjacency graph and the information regarding the edge labels simultaneously to find communities in the Multiplex network. Two cliques can be included in at least one community if: (1) there exists a path between the corresponding vertices in the clique-adjacency graph, and (2) for all vertices in the path the corresponding cliques share at least m edge types on all of their pairs. We call the latter rule the *cliques' constraint*, and we call a tree *maximal* if no other adjacent clique can be added without reducing the maximal m for which the

cliques' constraint is satisfied. Therefore, each community in the original network corresponds to a maximal tree in the clique-adjacency graph where the cliques' constraint holds for all vertices in the tree. So the problem is equivalent to recognizing all such maximal trees in the graph.

Algorithm 3 takes a community A as input and returns all maximal communities containing A. In this algorithm, given a set of cliques (that is, a community) A we notate the set of edge types we are currently considering to find a maximal community as $\Lambda(A)$. Notice that $\Lambda(A) \subseteq L(A)$: we are going to run this algorithm multiple times starting from the same clique with different $\Lambda(A)$'s, that is, looking for communities in different sets of edge types. We also define $N(c)$ as the neighbors of c in the adjacency graph, as before.

Algorithm 3 consists of two phases. First, it finds a maximal community on $\Lambda(A)$ (lines 1–15). Then, it recursively does the same for all subsets of $\Lambda(A)$ for which larger communities can be found (lines 17–20). B contains the set of cliques that may be used to extend the community A.

To understand the algorithm we invite the reader to start inspecting lines 1–3, 7–8, and 13–15: this is just a simple depth-first visit of the adjacency matrix, finding the largest connected component containing A only traversing cliques that exist on all the edge types in $\Lambda(A)$ (line 3). This visit, that finds the maximal community containing A on $\Lambda(A)$, is extended in two ways. First, lines 4–6 make sure that we do not produce communities that have already been found starting from another clique: if we encounter an already processed clique containing all the edge types in $L(A)$, then this community must have been found while processing it. Second, lines 9–12 cover the case when during the visit we encounter a neighboring clique that only contains a subset L' of $\Lambda(A)$. This means that there is a community containing A that is maximal on L'. Consequently, for each L' we encounter during the process we will later call Algorithm 3 again to extract a maximal community containing A on those edge types (line 17–19). Line 16 makes sure that we do not process the same set L' more then once (notice that the elements of D are sets of edge types). The fact that the different L''s correspond to cliques that we have encountered while visiting the adjacency matrix guarantees that we are not running the algorithm for all the subsets of $L(A)$ but only for those for which a community exists.

Algorithm 2. find-communities(Cliques, $\overline{m}$)

Input: Cliques is the set of cliques produced by the previous step
Input: $\overline{m}$ is the minimum number of edge types to output a community
1: $C = \emptyset$
2: **for** $c \in$ Cliques **do**
3: $A = \{c\}$
4: $\Lambda(A) = L(\{c\})$
5: find-communities$(A, N(\{c\}), C, \emptyset, \overline{m})$
6: $C = C \cup \{c\}$
7: **end for**

Algorithm 2 uses Algorithm 3 to iterate over all cliques and finds all maximal communities containing the clique under examination for any $m \geq \overline{m}$. Therefore, at the end it outputs all maximal communities.

A.4 Time Complexity

Finding maximal cliques is an NP-hard problem, that is, one for which even small datasets can take too long to be processed. In practice, as for its simple-graph version, the execution time of the multiplex clique percolation algorithm depends on the input data and in particular on the number and size of cliques.

Table 3 shows the algorithm's execution time for real multiplex social networks of increasing size from an online repository (http://multilayer.it.uu.se), on a 2,4 GHz Intel Core i7 desktop computer with 16 GB RAM. We have repeated each execution 5 times, observing similar results (the mode is indicated in the table) and we have stopped the computation after 10 min. The drastic drop in execution time for the dkpol dataset (including follow, mention and retweet edge types from Twitter) when stricter constraints are used, and the execution times on the ff-tw-yt data, representing common users from three online social

Algorithm 3. find-communities$(A, B, C, D, \overline{m})$ returns all maximal communities containing A.

Input: A is a set of cliques (partial community)
Input: B is the set of cliques that can be used to extend the community A
Input: C contains the vertices already processed by the algorithm
Input: D is a set of sets of edge types, not to process the same set of edges again
Input: $\overline{m}$ is the minimum number of edge types to output a community

```
1: while B ≠ ∅ do
2:    c = B.pop()
3:    if Λ(A) ⊆ L({c}) then
4:       if c ∈ C then
5:          return
6:       end if
7:       A = A ∪ {c}
8:       B = B ∪ N(c) \ A
9:    else if |Λ(A) ∩ L({c})| ≥ m̄ then
10:      if L({c}) ∉ D then
11:         S = S ∪ {c}
12:      end if
13:   end if
14: end while
15: OUTPUT A
16: D = D ∪ Λ(A)
17: for L' ∈ {L(c) : c ∈ S} do
18:   Λ(A) = L'
19:   find-communities(A, S, C, D, m̄)
20: end for
```

Table 3. Execution time (s)

Data	#edges	$k = 3, m = 1$	$k = 4, m = 2$
aucs	620	0	0
dkpol	20226	>600	1
arxiv	59026	569	547
ff-tw-yt	74862	>600	106
dblp	222510	>600	>600

networks, highlight how the computation time strongly depends on the data structure and is not linearly dependent on the network size. At the same time, these experiments show the effect of the constraints on execution time.

References

1. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
2. Coscia, M., Giannotti, F., Pedreschi, D.: A classification for community discovery methods in complex networks. Stat. Anal. Data Min. **4**(5), 512–546 (2011)
3. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435**(7043), 814–818 (2005)
4. Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J.P., Moreno, Y., Porter, M.A.: Multiplex Networks. J. Complex Netw. **2**(3), 203–271 (2014)
5. Dickison, M.E., Magnani, M., Rossi, L.: Multiplex Social Networks. Cambridge University Press, Cambridge (2016)
6. Sun, Y., Han, J.: Mining Heterogeneous Information Networks: Principles and Methodologies. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers, San Rafael (2012)
7. Bott, H.: Observation of play activities in a nursery school. Genet. Psychol. Monogr. **4**, 44–88 (1928)
8. Moreno, J.L.: Who Shall Survive? A New Approach to the Problem of Human Interrelations. Nervous and Mental Disease Publishing Co., Washington, D. C. (1934)
9. Xie, J., Kelley, S., Szymanski, B.K.: overlapping community detection in networks: the state-of-the-art and comparative study. ACM Comput. Surv. (CSUR) **45**(4) (2013)
10. Kumpula, J.M., Kivelä, M., Kaski, K., Saramäki, J.: Sequential algorithm for fast clique percolation. Phys. Rev. **78**(2), 026109 (2008)
11. Yan, B., Gregory, S.: Detecting communities in networks by merging cliques. In: IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009, vol. 1, pp. 832–836. IEEE (2009)
12. Nepusz, T., Petróczi, A., Négyessy, L., Bazsó, F.: Fuzzy communities and the concept of bridgeness in complex networks. Phys. Rev. E **77**(1), 016107 (2008)
13. Zhang, S., Wang, R.-S., Zhang, X.-S.: Identification of overlapping community structure in complex networks using fuzzy c-means clustering. Phys. A Stat. Mech. Its Appl. **374**(1), 483–490 (2007)

14. Ahn, Y.-Y., Bagrow, J.P., Lehmann, S.: Link communities reveal multiscale complexity in networks. Nature **466**(7307), 761–764 (2010)
15. Evans, T.S., Lambiotte, R.: Line graphs, link partitions, and overlapping communities. Phys. Rev. E **80**(1), 016105 (2009)
16. Bothorel, C., Cruz, J.D., Magnani, M., Micenkova, B.: Clustering attributed graphs: models, measures and methods. Netw. Sci. **3**(3), 408–444 (2015)
17. Berlingerio, M., Coscia, M., Giannotti, F.: Finding and characterizing communities in multidimensional networks. In: International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 490–494 (2011)
18. Cai, D., Shao, Z., He, X., Yan, X., Han, J.: Community mining from multi-relational networks. In: Jorge, A.M., Torgo, L., Brazdil, P., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 445–452. Springer, Heidelberg (2005). https://doi.org/10.1007/11564126_44
19. Rodriguez, M.A., Shinavier, J.: Exposing multi-relational networks to single-relational network analysis algorithms. J. Inf. **4**(1), 29–41 (2010)
20. Tang, L., Wang, X., Liu, H.: Community detection via heterogeneous interaction analysis. Data Min. Knowl. Discov. **25**(1), 1–33 (2012)
21. Mucha, P.J., Richardson, T., Macon, K., Porter, M.A., Onnela, J.-P.: Community structure in time-dependent, multiscale, and multiplex networks. Science **328**(5980), 876–8 (2010)
22. Berlingerio, M., Pinelli, F., Calabrese, F.: ABACUS: frequent pAttern mining-BAsed Community discovery in mUltidimensional networkS. Data Min. Knowl. Discov. **27**(3), 294320 (2013)
23. De Domenico, M., Lancichinetti, A., Arenas, A., Rosvall, M.: Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. Phys. Rev. X **5**(1), 11027 (2015)
24. Afsarmanesh, N., Magnani, M.: Finding overlapping communities in multiplex networks. https://arxiv.org/abs/1602.03746
25. Magnani, M., Rossi, L.: The ML-model for multi-edge type social networks. In: International Conference on Social Network Analysis and Mining (ASONAM), pp. 5–12. IEEE Computer Society (2011)
26. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**(2), 026113 (2004)
27. Rossi, L., Magnani, M.: Towards effective visual analytics on multiplex and multilayer networks. Chaos Solitons Fractals **72**, 68–76 (2015)
28. Leskovec, J., Lang, K.J., Mahoney, M.W., Dasgupta, A.: Statistical properties of community structure in large social and information networks. In: Proceeding of the 17th International Conference on World Wide Web - WWW 2008, p. 695 (2008)