NETWORKS AND HETEROGENEOUS MEDIA
©American Institute of Mathematical Sciences
Volume **10**, Number **1**, March **2015**

doi:10.3934/nhm.2015.10.71
pp. **71–85**

COMMUNITY DETECTION IN MULTIPLEX NETWORKS: A SEED-CENTRIC APPROACH

Manel Hmimida

DICEN-CNAM 292, rue St Martin
75141 PARIS CEDEX 03, & & LIPN - CNRS UMR 7030
Paris, France

Rushed Kanawati

PSC, UP13, LIPN, CNRS UMR 7030
Villetaneuse, France

Abstract. Multiplex network is an emergent model that has been lately proposed in order to cope with the complexity of real-world networks. A multiplex network is defined as a multi-layer interconnected graph. Each layer contains the same set of nodes but interconnected by different types of links. This rich representation model requires to redefine most of the existing network analysis algorithms. In this paper we focus on the central problem of community detection. Most of existing approaches consist on transforming the problem, in a way or another, to the classical setting of community detection in a monoplex network. In this work, we propose a new approach that consists on adapting a seed-centric algorithm to the multiplex case. The first experiments on heterogeneous bibliographical networks show the relevance of the approach compared to the existing algorithms.

1. **Introduction.**

Nodes in complex networks are generally arranged in tightly connected groups that are loosely connected one to each other. Such groups are called communities. Community members are generally admitted to share common properties. Hence, unfolding the community structure of a network could provide much insights about the overall structure of the network. The problem of community detection in complex networks has received much f attention in the last decade. Most of existing approaches are designed for simple static networks, where all edges are supposed to be of the same type [15, 29]. However, real networks are often:

- *Heterogeneous*: nodes and links may have different types.
- *Dynamic*: nodes and links may evolve with time.

The multiplex network model has been introduced lately in order to cope with the both heterogeneous and dynamic networks [5, 13, 8]. A multiplex network (a.k.a multi-slice, multi-relational, multi-layer network) is defined as an interconnected multi-layer network. Each layer contains exactly the same set of nodes. However, the nature of links vary from one layer to another. An exemple of a multiplex network is illustrated on figure 1. The figure shows a three-layer multiplex defined over a set of nodes, each node represents an author of an academic publication

2010 *Mathematics Subject Classification.* Primary: 58F15, 58F17; Secondary: 53C35.
Key words and phrases. Mux-Licod, complex network, multiplex networks, community detection, modularity, redundancy.

(dataset is extracted from the well known DBLP[1] bibliographical database). The first (resp. second, third) defines a co-authorship (resp. co-citation, co-venue) relationship between authors. Two authors (nodes) are linked in the first (resp. second, third) layer if they co-author a publication (resp. have been co-cited by a third author, have participated to the same venue).

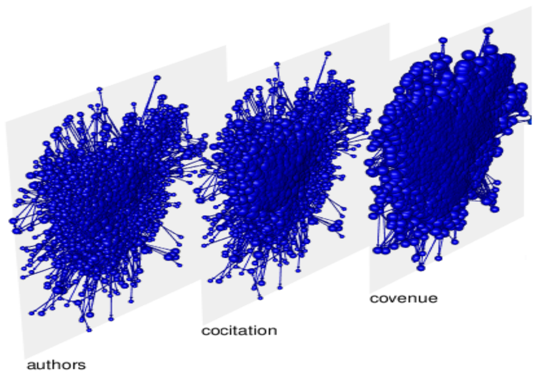

FIGURE 1. Multiplex bibliographical network. Nodes are authors, layers define the following three relationships: co-authorship, co-citation and co-venue

The multiplex model can readily copes with multi-relationnel networks (ex. one relation per layer), but also with dynamic networks (a layer corresponds to the network stat at a given time stamp). However, analyzing multiplex networks requiers to redefine most of basic metrics that have been previously defined in the context of monoplex network such as node's centralities [9, 3], clustering coefficient [11] and nodes similarity [13]. In this work, we study the problem of community detection in multiplex networks [4]. Few works have addressed this problem. Most of which are based transforming the problem, in a way or another, to the classical setting of community detection in a monoplex network [4, 28, 29]. Another approach consists on extending an existing algorithm to deal directly with multiplex networks. This is for instance the approche adopted in [22], where the modularity quality function has been extended to the multiplex case allowing to apply classical modularity-optimization approaches [7]. A multi-objective optimization approach has been proposed in [2]. In this work we propose to extend a seed-centric community detection approach to be applied to the multiplex network [30]. Seed-centric approaches are based on the idea of first selecting some special nodes in the network for which local communities are first computed. A global community structure of the network is then inferred from the set of obtained local communities [18]. These approaches are mainly based on local computations allowing hence to handle large-scale graphs. A quick survey of on seed-centric approches for community detection in monoplex networks is reported in [27].

The remainder of this paper is organized as follows. In section 2 we introduce basic definitions and used notations related to multiplex networks. In section 3, we provide a quick survey on main approaches for community detection in multiplex

[1]http://www.dblp.org

networks. Next in section 4, we describe the proposed approach, called *mux-Licod* which is an extension of the *Licod* algorithm provided in [30]. Experiments on real networks are reported and commented in section 5. Finally we conclude in section 6.

2. **Multiplex networks.**

2.1. **Notations.**

Formally, we define a multiplex network as a multi-layer graph:

$$G =< V, E_1, \dots, E_\alpha : E_k \subseteq V \times V \ \forall k \in \{1, \dots, \alpha\} > \tag{1}$$

where V is a set of nodes and E_k is a set of edges of type k. α denotes the number of layers in the multiplex. Next, we introduce some new notations that will be used further in this paper.

- $A^{[k]}$ is the adjacency matrix of layer k
- $n = |V|$ is the number of nodes in the multiplex.
- $m_k = |E_k|$ is the number of edges in layer k
- $\Gamma(v)^{[k]} = \{x \in V : (x, v) \in E_k\}$ denotes the neighbors of v in layer k
- $d_v^k = \parallel \Gamma(v)^{[k]} \parallel$ is the degree of node v in layer k

2.2. **Multiplex metrics.**

The multi-layer nature of a multiplex network poses the problem of redefining all basic network metrics [3]. We consider next the redefinition of three very basic network concepts used later in our proposed algorithm (see section 4). These are: node's degree, node's neighborhood, and shortest-path mesure between two nodes.

Multiplex node degree. The degree of a node i in a multiplex network composed of α layers can simply be defined as an aggregation of i's degrees in each of the layers. Formally, we have:

$$d_i^{multiplex} = \mathcal{F}(d_i^{[1]}, \dots, d_i^{[\alpha]}) \tag{2}$$

Where $\mathcal{F}$ is an aggregate function. Basic aggregate functions, such a the *mean, sum, . . . , etc*, do not allow distinguishing nodes having very different degree distributions across layers. For exemple, consider the case of a 3-layer multiplex and two nodes i, j such that i has one neighbor is each layer while j has three neighbors in one layer and none in each of the remaining two layers. Both nodes will have exactly the same multiplex degree if a simple aggregate function is applied. A more interesting option would be the application of an entropy-like aggregate function as proposed in [3]:

$$d_i^{multiplex} = -\sum_{k=1}^{\alpha} \frac{d_i^{[k]}}{d_i^{[tot]}} log\left(\frac{d_i^{[k]}}{d_i^{[tot]}}\right) \tag{3}$$

where $d_i^{[tot]} = \sum_{k=1}^{\alpha} d_i^{[k]}$ is the total degree of a node i in the multiplex network.

Following this definition, the degree of a node i would be null if all it is only connected in one layer. It would be maximized if the node is equally connected in each layer. This would be more suitable for computing degree centrality of a node where nodes connected across multiple layers of a multiplex would be more important than nodes connected in only one layer.

Neighborhood. In much a similar way to the multiplex degree definition, the set of neighbors of a node i in a multiplex would be defined as an aggregation of its neighbors sets in each layer. One restrictive aggregate function would be the set-intersection function: j is a neighbor of i if j is connected to i is each layer. On the opposite side, a very loose definition of node's neighborhood would be defined by using the set-union function: j is a neighbor of i if j is connected to i in at least one layer [19]. A trade-off between these two basic definitions is the one proposed in [8]: j is a neighbor of i if it is connected to i in at least m layers where $1 \leq m \leq \alpha$. This definition may suits multiplex composed of large number of layers. It poses also the problem of defining a threshold on the number of layers to consider.

We propose here another trade-off based on node's similarity. The principle is to select from the set of all neighbors of a node (across all layers), those which are most *similar* to the considered node. Topological similarity measures can be used for that purpose. More formally, let $\Gamma(i)^{total} = \{j : \exists k : j \in \Gamma(i)^{[k]}\}$ be the set of neighbors of node i across all layers. We define the multiplex neighbourhood of a node i as follows:

$$\Gamma_{\delta^{multiplex}(i)} = \{c \in \Gamma(i)^{total} :\sim (c, i) \geq \sigma\} \tag{4}$$

Where $\delta \in [0, 1]$ is a similarity threshold. Different similarity functions can be applied. An exemple would be the Jaccard similarity measure defined as follows:

$$sim_{jaccard}(i, j) = \frac{\Gamma(i)^{total} \cap \Gamma(j)^{total}}{\Gamma(i)^{total} \cup \Gamma(j)^{total}} \tag{5}$$

Shortest-path. Again, shortest-path length in a multiplex between two nodes i, j can be defined as an aggregation of shortest-path length between these nodes in each layer go the multiplex. Entropy-based aggregate function can also be applied. More generally, any dyadic topological measure X can be defined as an aggregate as follows:

$$X_{ent}(i, j) = -\sum_{k=1}^{\alpha} \frac{X(i, j)^{[k]}}{X_{total}} \log(\frac{X(i, j)^{[k]}}{X_{total}}) \tag{6}$$

where $X_{total} = \sum_{k=1}^{\alpha} X(i, j)^{[k]}$. The entropy based aggregate is more suitable for capturing the distribution of the measure value across all layers. A higher value indicates uniform distribution attribute value across the multiplex layers.

3. Community detection in multiplex networks.

We classify existing approaches for community detection in multiplex networks in two main classes:

1. *Approaches based on using existing monoplex community detection algorithms.* The principle is to transform the problem of community detection in a multiplex to the one of community detection is a monoplex network. One first approach consists on transforming a multiplex into a monomplex by applying some *layer aggregation* scheme [28]. A second approach consists on applying a community detection algorithm to each layer of the multiplex. Then apply an *ensemble-clustering* approach in order to combine all obtained partitions [6].
2. *Approaches based on extending existing algorithms to deal directly with multiplex networks.*

Next we detail the above identified approaches.

3.1. **Layer aggregation approaches.** Generally, the layer aggregation approach consists on transforming a multiplex network into a weighted graph $G =< V, E, W >$ where W is a weight matrix. Figure 2 illustrates the process of layer aggregation.

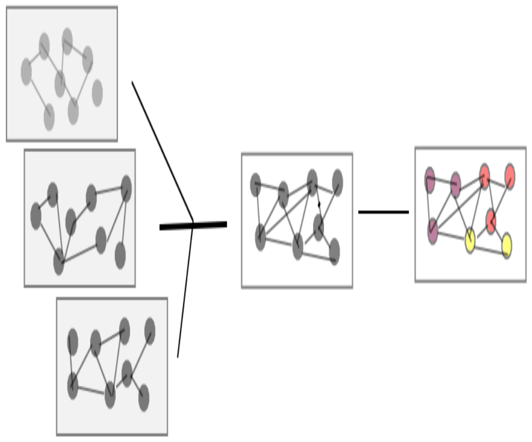

FIGURE 2. Layer aggregation based approach.

Different weights computations schemes can be applied. Examples are:

Binary weights: two nodes i, j are linked in the resulting simple graph if it exists at least one layer in the multiplex where these nodes are linked. Formally we have:

$$w_{ij} = \begin{cases} 1 & \text{if } \exists 1 \leq k \leq \alpha : (i,j) \in E_k \\ 0 & \text{else} \end{cases} \tag{7}$$

Frequency-based weighting: In [29], authors propose to weight a link (i,j) by the average of weights in all layers in the multiplex. Formally we have:

$$w_{ij} = \frac{1}{\alpha} \sum_{k=1}^{\alpha} A_{ij}^{[k]} \tag{8}$$

A similar weighting scheme is proposed in [4] where a link is weighted by its redundancy :

$$w_{ij} = \parallel \{d : A_{ij}^{[d]} \neq 0\} \parallel \tag{9}$$

Similarity-based weighting schema: The weight of a link (i,j) is given by the similarity of two nodes computed in the multiplex graph. In practice, temporal dyadic similarity measures used to compute similarity score of two nodes [25] (layers in the multiplex are considered as time stamps). In [4], authors propose to use the clustering coefficient before computing the weight of links in the aggregated simple graph.

Linear combinaison: In [10], authors propose to consider independantly the different layers of a multiplex. Therefore, the weight of a link in the resulting aggregated graph should take into account the difference of layers contributions. A linear combinaison schemas can then be applied:

$$A = \sum_{k=1}^{\alpha} w_k A^{[k]} \tag{10}$$

The weights w_k can also be learned based on user defined constraints on the clustering of some nodes into communities.

More recently, another transformation approach has been proposed [20]. It consists on mapping a multiplex to a 3-uniform hypergraph $H = (V, E)$ such as the node set in the hypergraph is $V = V \cup 1, ..., \alpha$ and $(u, v, i) \in E$ if $\exists l : A^l_{uv} \neq 0, u, v \in V, i \in 1, ..., \alpha$. Community detection algorithmes in hypergraphs can then be applied [24].

3.2. **Ensemble-clustering based approach.** This approach consists on first applying a community detection algorithm to each layer. Then an ensemble clustering approach is applied in order to combine all obtained partitions (see figure 3).

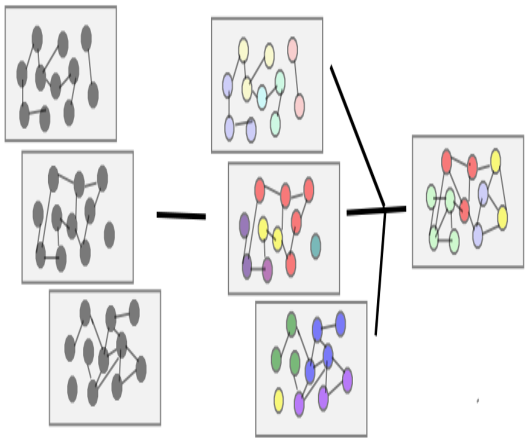

FIGURE 3. Partition Aggregation

A variety of ensemble clustering approaches can be applied [1]. One widely applied method is based on constructing a **consensus graph** out of the set of partitions to be combined [1]. The consensus graph G_{cons} is defined over the same set of nodes of the initial graph G. Two nodes $i, j \in V$ are linked in G_{cons} if there is at least one partition where both nodes are in a same cluster. Each link (i, j) is weighted by the frequency of instances that nodes i, j are placed in the same cluster. Notice that the obtained graph is not necessarily a connected one. Different approaches can be applied in order to compute the aggregated clustering out from the consensus graph:

- In [1], authors transform the graph into a complete one by adding missing links with a null weight, then nodes are finally partitioned into clusters using agglomerative hierarchical clustering with some linkage rule, or by using a classical graph partition method such as the Kernighan-Lin algorithm.
- in [12] a similar approach is applied but with enforcing that nodes in the same result clusters should be connected in the initial graph by a sufficiently short path.
- In [21] authors propose a simple but effective method that consists on pruning links in the obtained consensus graph whose weights (frequency) is under a given threshold $\lambda \in [0, 1]$. The set of obtained connected components is taken

to be the aggregated partition. The main problem of this approach is the problem of defining the value of the threshold λ to use.

3.3. **Extending monoplex approaches to the multiplex case.** In [29] a unified approach for community detection is proposed allowing to model different types of algorithms using matrix computations. This unified approach allows to identify different aggregation strategies in addition to the two basic ones: layer aggregation and partition ensemble clustering.

In [2], a multi-objective optimization approach is proposed. This idea is to first rank the layers of the multiplex. A classical community detection algorithm is applied on the first layer. Then for each subsequent layer a community detection algorithm that optimize two criteria at once is applied: optimizing the local modularity of the obtained partition, and maximizing the similarity with the partition found on the previous layer. The main problem of this approach is about defining a layer ranking function.

The leading role that modularity and its optimization have played in the context of communities detection in simple graphs has naturally motivated works to generalize the modularity in case of multiplex networks. A multiplex modularity is proposed is [23] as follows:

$$Q_{multiplex}(P) = \frac{1}{2\mu} \sum_{c \in P} \sum_{\substack{i,j \in c \\ k,l:1 \to \alpha}} \left(\left(A_{ij}^{[s]} - \lambda_k \frac{d_i^{[k]} d_j^{[k]}}{2m^{[k]}} \right) \delta_{kl} + \delta_{ij} C_{ij}^{kl} \right) \tag{11}$$

where μ is a normalization factor, and λ_k is a resolution factor as introduced for multi-resolution modularity [26]. Note that in our case, only links inter-layer are implicit connecting node i to itself in the others layers. Therefore we have: $C_{ij}^{kl} = 0 \; \forall i \neq j$. With this new modularity, classical approaches for modularity maximization can be applied directly to multiplex networks. However, there is now evidences showing that this new modularity does not suffer from the drawbacks of the classical modularity optimization approaches [16].

4. **The proposed approach.** In this section, we propose a new seed-centric approach, called *mux-Licod*, for community detection in multiplex networks. The proposed approach is an extension of the *Licod* algorithm designed for monoplex networks [30]. Seed-centric algorithms constitue an emerging trend in the area of community detection. The basic idea underlaying these approaches consists on identifying special nodes in the target network, called seeds, around which communities can then be identified. These algorithms are mainly based on performing local computations allowing to consider their application to large-scale graphs. Next, we give an informal description of the proposed algorithm. Implementation issues are discussed in 4.2. A quick discussion of the algorithm computational complexity is presented in 4.3.

4.1. **Algorithm description.** Algorithm 1 presents the general outlines of a typical seed-centric community detection algorithm [27]. Three main steps can be distinguished:

1. Seed computation.
2. Seed local community computation.
3. Community computation from all local communities calculated in step 2

Algorithm 1 General seed-centric community detection algorithm

```
Require: $G =< V, E >$ a connected graph,
1: $\mathcal{C} \leftarrow \emptyset$
2: $S \leftarrow$ compute_seeds(G)
3: for $s \in S$ do
4:     $C_s \leftarrow$ compute_local_com(s,G)
5:     $\mathcal{C} \leftarrow \mathcal{C} + C_s$
6: end for
7: return compute_community($\mathcal{C}$)
```

Following the idea of the original *Licod* algorithm we search to define seeds as leaders of communities: set of nodes that are plying central role inside the community. In order to find such *leaders*, we first search for all single *leader-nodes*. These are defined as nodes that have higher centrality than most of their direct neighbors. Once all *leader-nodes* are identified, we cluster these in function of their similarity in order to obtain the final *leaders*. Notice that *leaders* are then defined as a set of nodes not necessarily directly connected in the graph. Each node in the graph, then rank the set of leaders in function of some preference membership function. A complete rank of all leaders by each node is required. Then, since two directly connected nodes are likely to belong to the same community we apply a local preference merging algorithm that allows each node to update its membership preference vector in function of preferences of its *neighbors*. The procedure of local preference merging iterates until stabilization. Each node in the graph will then be affected to the community defined by the *leader*?ranked first in its preference vector.

4.2. **Implementation issues.** Algorithm 2 sketches the outlines of *mux-Licod*. In the current implementation we use degree centrality in order to identify *leader-nodes*. We use the formula 3 in order to compute the degree of a node in the multiplex network. The function `isLeaderNode()` (line 3 in 2) is then simply implemented by comparing the degree of each node with the degree of its direct *neighbors*. We apply the similarity-based neighborhood selection formal for computing the set of neighbors of a node in the multiplex network (see formula 4). The Jaccard similarity function given in formula 5 is used. A node is a *leader-node* if its degree centrality is greater than the degree centrality of most of its direct neighbors. Formally a node i is a *leader-node* if $\frac{|\{v \in \Gamma_{\delta}^{multiplex}(i) : d_i^{multiplex} \geq d_v^{multiplex}\}|}{|\Gamma_{\delta}^{multiplex}(i)|} \geq \sigma$. $\sigma \in [0, 1]$ is an algorithm parameter. *Leader-nodes* are clustered using the same Jaccard similarity function in order to compute the set of *leaders* (line 7 in algorithm 2). The idea is that *leader-nodes* having a relatively large number of common neighbors are grouped into one community.

As for the original *Licod* algorithm, the membership degree of node to a community is estimated using the value of the shortest path length linking the node to the *leader*. Here we apply formula 6 for computing the length of the shortest path between nodes in multiplex network. Different rank aggregation functions can be applied for merging the preferences community membership of neighboring nodes (line 16 in algorithm 2) . We apply here the *local kemeny* approach proposed in [14] as it was shown that this gives the best results in the context of the original *Licod* algorithm [30]. The preference merge process is iterated till convergence or for max number of iterations.

4.3. **Algorithm complexity.** The computational complexity of the proposed algorithm depends on the complexity of three basic steps of computing leaders, computing community membership degrees and then preference rank aggregation applied method. Using the degree centrality for leader-nodes identification requiers $O(m)$ steps where $m = \max_{i\in[1,\alpha]} |E_i|$ is the max number of edges in the layers of the multiplex. The most expensive step is one required for computing the preference of nodes membership to each identified community since this requiers computing shortest-paths lengths between each node and identified leaders-nodes. Applying classical shortest-path length compassion algorithm, the complexity is $O(m + |V|log(|V|)$. The preference merging step has a complexity of $O(l \times log(l))$ where $l = |C|$ is the number of identified leaders. Actually the local rank merging procedure is simply implemented using the quick-sort algorithm as detailed in [14]. However, the number of identified leaders is usually very small compared to the number of all nodes. To sum up, the *Mux-licod* algorithm, in this current implementation will have a computational complexity of $O(m + |V|log(|V|)$.

Algorithm 2 mux-LICOD algorithm

```
Require: $G =< V, E_1, \dots, E_\alpha >$ a connected Multiplex graph
1: $\mathcal{L} \leftarrow \emptyset$ {#set of leader-nodes}
2: for  $v \in V$ do
3:     if $isLeaderNode(v)$ then
4:         $\mathcal{L} \leftarrow \mathcal{L} \cup \{v\}$
5:     end if
6: end for
7: $\mathcal{C} \leftarrow cluster(\mathcal{L})$ {#Compute Leaders}
8: for $v \in V$ do
9:     for $c \in \mathcal{C}$ do
10:        $M[v].append(membership(v, c))$
11:    end for
12:    $P[v]$ = sortAndRank($M[v]$)
13: end for
14: repeat
15:    for $v \in V$ do
16:        $P[v] \leftarrow \mathbf{rankAggregate}_{\mathbf{x}\in\{\mathbf{v}\}\cap\Gamma^{\mathbf{multiplex}}(\mathbf{v})}\mathbf{P}[\mathbf{x}]$
17:    end for
18: until Stabilization of $P[v]$? $\forall v \in V$
19: $\mathcal{COM} \leftarrow \emptyset$
20: for $v \in V$ do
21:    $\mathcal{COM}.append(P[v][0])$
22: end for
23: return  $\mathcal{COM}$
```

5. **Experiments.**

5.1. **Settings.** In order to evaluate the proposed algorithm, we choose to compare its performances with basic approaches using layer-aggregation and ensemble-clustering approaches. To be as fair as possible we applied these two approaches with the original *Licod* algorithm. The same parameters are used for *Licod* and *mux-licod*, namely we set σ to 0.9 as this was the best value as shown in experiments reported in [30]. The neighbourhood threshold σ is set to the default value 0.5.

5.2. **Evaluation criteria.** To the best of our knowledge, there do not exist any multiplex networks with ground-truth partitions into communities. Thus, classical approaches for evaluating performances of community detection algorithme using *supervised* indices can not be applied. Only, unsupervised estimation of the quality of obtained communities can be used. One such metric is the *redundancy* proposed in [4]. The redundancy computes the average of redundant link of each intra-community in all multiplex layers. The intuition is that the link intra-community should be recurring in different layers. The computing of this indicator is as follows:

We denote by:

- P the set of couple (u, v) which are directly connected to at least one layer.
- $\bar{\bar{P}}$ the set of couple (u, v) which are directly connected in at least two layers.
- $P_c \subset P$ represents all links in the community c.
- $\bar{\bar{P}}_c \subset \bar{\bar{P}}$ the subset of $\bar{\bar{P}}$ and which are also in c.

The redundancy of the community c is given by:

$$\rho(c) = \sum_{(u,v)\in\bar{\bar{P}}_c} \frac{\| \{k : \exists A^{[k]}_{uv} \neq 0\} \|}{\alpha\times \| P_c \|} \tag{12}$$

The quality of a given multiplex partition is defined as follows:

$$\rho(\mathcal{P}) = \frac{1}{\| \mathcal{P} \|} \sum_{c\in\mathcal{P}} \rho(c) \tag{13}$$

We also, follow the proposition made in [2] fr evaluating the quality of obtained communities using the simple modularity criteria applied to each layer of the multiplex. The idea is to measure is taking into account different types of relationships between nodes can enhance the quality of ebonies communities with respect to one relationship. We also computed the modularity value of the obtained partition with respect to the aggregated graph using two different aggregation schemes: a simple layer-union and the one obtained by applying the Jacquard-based node similarity function (two nodes are liked in the aggregated network if one is in the neighborhood of the another as computed by the Jaccard similarity neighborhood function, see section 2.2).

5.3. **Datasets.** We evaluated the differsnt approaches n two different real work datasets. The first one is a 3-layer multiplex network extracted form the bibliographical database `DBLP`, while the second is a set of three 2-layer multiplex networks build from of a bibliographical sharing web service *Bibsonomy*[2]. Next we briefly describe these two datasets.

Dblp dataset. Dblp is bibliographical database referencing a huge amount of scientific papers mostly related to computer science. We extracted from the publicly available database, a subset corresponding to publications covering the time period 1980 to 1985. A 3-layer multiplex network is constructed out from this dataset: Nodes of the multiplex are authors. The first layer encodes a co-authorship relationship. The second one gives co-citation relationships beteen authors while te third layer gives co-venue relation between authors (participating to the same conference edition). The table 1 summarizes the information about networks built out of the `DBLP` dataset.

[2]http://www.bibsonomy.org

Layer	# Nodes	# Edges	Density
co-authoring	2809	5109	0.001295439
co-citation	2809	36187	0.000000780
co-venue	2809	251819	0.000000161

TABLE 1. Basic statistics about the 3-layer multiplex extracted from DBLP

Bibsonomy. Bibsonomy is a bibliographical reference sharing system. Users can tag references that can be available for browsing by other users. We use here the 2007 post-core at level 5, used for evaluation in [17]. The network is composed by three types of nodes: tags, resources and users. Out of this tripartite graph, we build three 2-layer multiplex networks defined respectively on each of the three sets: users, tags and ressources. First, we start by decomposing the tripartite graph into three bipartite subgraphs : User-Tag, Resource-Tag and User-Resource. Then, we make two projections of each bipartite graph. As result, we obtain six unipartite graphs that compose the three 2-layer networks. Figure 4 illustrate the applied construction process of the 2-layer multiplex defined over the tag nodes.

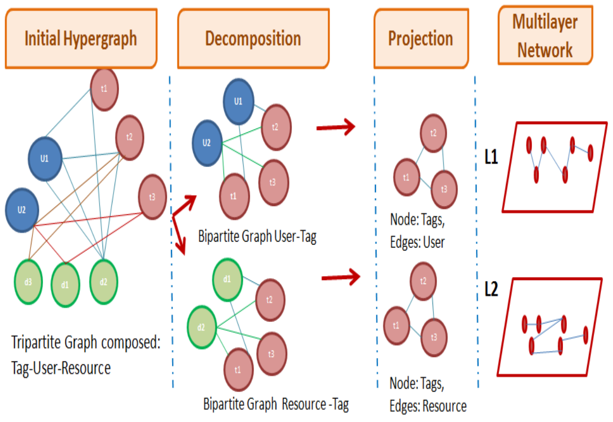

FIGURE 4. 2-layer tag multiplex network

Basic statistics about obtained multiples networks are summarized in table 5.3.

Multiplex network	Layers	# Nodes	# Edges	Density
User	*User based Resource*	116	901	0,135
	User based Tag	116	985	0,147
Tag	*Tag based Resource*	412	2496	0,0294
	Tag based User	412	1956	0,0231
Resource	*Resource based Tag*	361	2814	0,0433
	Resource based User	361	1685	0,0259

TABLE 2. Bibsonomy multiplex networks

5.4. **Results.** Table 5.4 (resp. 5.4) shows the number of communities obtained on the *DBLP* (resp. Bibsonomoy) multiplex networks when applying the three selected community detection algorithms.

Methods	**Dblp network**
Layer Aggregation	184
Ensemble Clustering	59
mux-LICOD	179

TABLE 3. Numbers of obtained communities in the Dblp multiplex network

Methods	**User network**	**Resource network**	**Tag network**
Layer Aggregation	13	48	42
Ensemble Clustering	21	86	88
mux-LICOD	24	70	46

TABLE 4. Numbers of obtained communities in the Bibsonomy multiplex networks

In figure 5 (resp. 6, 7, 8) we show the modularities of communities structures computed by the three different approaches with respect to each layer of the multiplex and with respect to aggregated networks for the `DBLP` (resp. `Bibsonomy`) dataset.

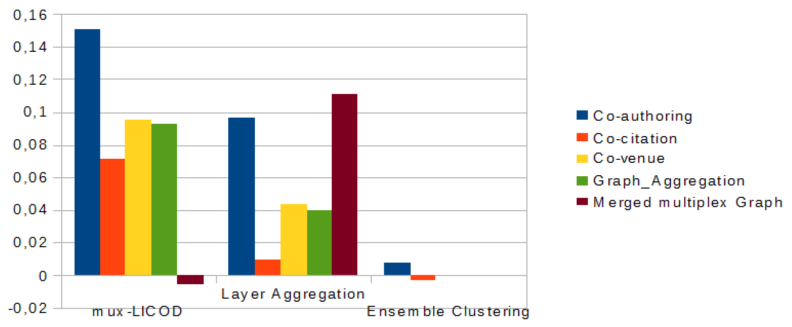

FIGURE 5. Modularities of the obtained community structure with respect to each layer and to aggregated networks - DBLP dataset

Results show that *mux-licod* provides better modularity values for almost all networks. This may suggest that both layer aggregation and ensemble-clustering approaches are not the adequate approaches for multiplex community detection. Actually, layer aggregation approaches leaders to a great loss of information about the heterogeneous nature of links. While ensemble-clustering approaches merge communities that are much different (since each is computed according to a different relation). The best way might be the one we propose based on analyzing all layers of the multiplex at once.

In table 5.4 (resp. 5.4) we show the values of redundancy index of obtained communities. Again, *muc-Licod* outperforms the other two approaches in almost all cases. These first results are very promising. However, these need also confirmation on other kind of datasets. The approach based on extending algorithms to deal with multiplex networks seems to outperform approaches based on transforming the multiplex community detection to the problem of monoplex community detection.

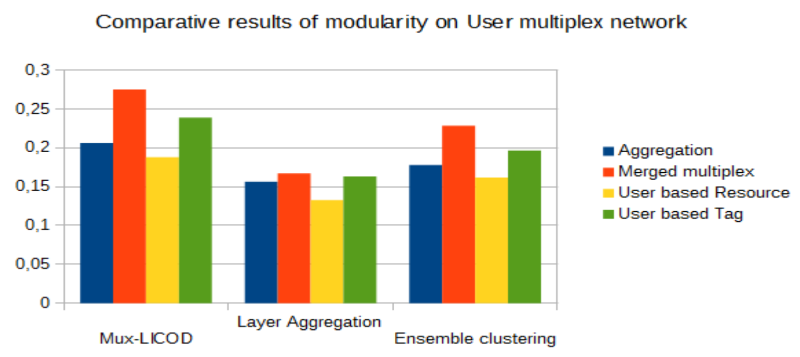

FIGURE 6. Modularities of the obtained community structure with respect to each layer and to aggregated networks - Bibsonomy dataset, User multiplex network

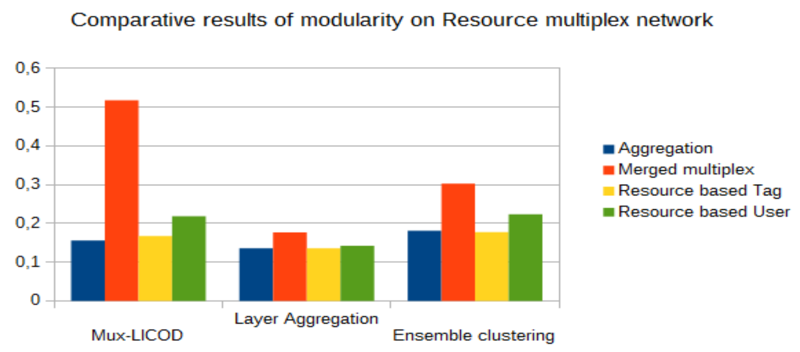

FIGURE 7. Modularities of the obtained community structure with respect to each layer and to aggregated networks - Bibsonomy dataset, Ressource multiplex network

Comparaisons of performances of *mux-Licod* with approaches from the same type (see section 3.3) are required to better evaluate the performances of the proposed algorithm.

Methods	**Redundancy**
Layer Aggregation	$0,0017$
Ensemble Clustering	$0,0346$
mux-LICOD	$\mathbf{0,4761}$

TABLE 5. Comparative results of redundancy on Dblp dataset

6. **Conclusion.** In this paper we presented a new approach of community detection based on seed-centric algorithm that takes into account different types of relationships between nodes in different layers of a multiplex network. Our approach called *mux-LICOD* is a direct generalization of the *Licod* algorithm, initially proposed to cope with monoplex networks. First experiments on real datasets show that the

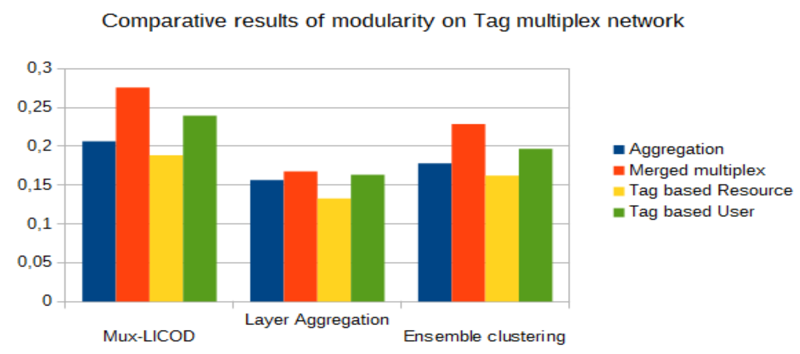

FIGURE 8. Modularities of the obtained community structure with respect to each layer and to aggregated networks - Bibsonomy dataset, Tag multiplex network

Methods	User network	Resource network	Tag network
Layer Aggregation	**0, 085**	0, 010	0, 026
Ensemble Clustering	0, 050	0, 059	0, 100
mux-LICOD	0, 069	**0, 484**	**0, 201**

TABLE 6. Comparative results of redundancy on Bibsonomy dataset

proposed approach yields better results than other classical approaches based on layer-aggregation or ensemble-clustering approaches.

We are working to extend experiments to include other state-of-the-art approaches, including using different algorithms when applying layer-aggregation and ensemble clustering approaches. Comparison with other approaches extending algorithms designed for monoplex networks such as the *generalized Louvain* algorithm are also scheduled [3].

Our proposed algorithm, *Mux-Licod* has many parameters. Effects of each parameter should also be carefully studied. The evaluation problem still to be an open problem, even for the monoplex networks. Task-driven evaluation approaches, such as proposed in [30] should also be considered.

REFERENCES

[1] S. Alexander and G. Joydeep, Cluster ensembles a knowledge reuse framework for combining multiple partitions, *The Journal of Machine Learning Research*, **3** (2003), 583–617.

[2] A. Amelio and C. Pizzuti, A cooperative evolutionary approach to learn communities in multilayer networks, in *Parallel Problem Solving from Nature–PPSN XIII*, Lecture Notes in Computer Science, 8672 Springer International Publishing, Switzerland, 2014, 222–232.

[3] F. Battiston, V. Nicosia and V. Latora, Structural measures for multiplex networks, *Physical Review E*, **89** (2014), 032804.

[4] M. Berlingerio, M. Coscia and F. Giannotti, Finding and characterizing communities in multidimensional networks, in *2011 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, IEEE, 2011, 490–494.

[5] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale and D. Pedreschi, Evolving networks: Eras and turning points, *Intell. Data Anal.*, **17** (2013), 27–48.

[6] M. Berlingerio, F. Pinelli and F. Calabrese, Abacus: frequent pattern mining-based community discovery in multidimensional networks, *Data Mining and Knowledge Discovery*, **27** (2013), 294–320.

[7] V. D. Blondel, J.-l. Guillaume and E. Lefebvre, Fast unfolding of communities in large networks, *Journal of Statistical Mechanics: Theory and Experiment*, **2008** (2008), P10008.
[8] P. Brodka and P. Kazienko, *Encyclopedia of Social Network Analysis and Mining, Ch. Multi-layered Social Networks,* Springer, 2014.
[9] P. Bródka, K. Skibicki, P. Kazienko and K. Musial, A degree centrality in multi-layered social network, in *2011 International Conference on Computational Aspects of Social Networks (CASoN)*, IEEE, 2011, 237–242.
[10] D. Cai, Z. Shao, X. He, X. Yan and J. Han, Mining hidden community in heterogeneous social networks, in *Proceedings of the 3rd International Workshop on Link Discovery*, ACM, 2005, 58–65.
[11] E. Cozzo, M. Kivelä, M. De Domenico, A. Solé, A. Arenas, S. Gómez, M. A. Porter and Y. Moreno, Clustering coefficients in multiplex networks, CoRR, arXiv:1307.6780, 2013.
[12] J. Dahlin and P. Svenson, Ensemble approaches for improving community detection methods, CoRR, arXiv:1309.0242, 2013.
[13] M. De Domenico, A. Solé, S. Gómez and A. Arenas, Random walks on multiplex networks, CoRR, arXiv:1306.0519, 2013.
[14] C. Dwork, R. Kumar, M. Naor and D. Sivakumar, Rank aggregation methods for the web, in *Proceedings of the 10th International Conference on World Wide Web*, ACM, 2001, 613–622.
[15] S. Fortunato, Community detection in graphs, *Physics Reports*, **486** (2010), 75–174.
[16] B. H. Good, Y.-A. de Montjoye and A. Clauset, Performance of modularity maximization in practical contexts, *Physical Review E*, **81** (2010), 046106, 19pp.
[17] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme and G. Stumme, Tag recommendations in social bookmarking systems, *AI Communications*, **21** (2008), 231–247.
[18] R. Kanawati, YASCA: An ensemble-based approach for community detection in complex networks, in *Computing and Combinatorics*, Lecture Notes in Computer Science, 8591, Springer International Publishing, Switzerland, 2014, 657–666.
[19] P. Kazienko, P. Brodka and K. Musial, Individual neighbourhood exploration in complex multi-layered social network, in *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Vol. 3*, IEEE, 2010, 5–8.
[20] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno and M. A. Porter, Multilayer networks, preprint, arXiv:1309.7233, 2013.
[21] S. Massoud, *Coeurs Stables de Communautés dans les Graphes de Terrain*, Ph.D thesis, 2012.
[22] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter and J.-P. Onnela, Community structure in time-dependent, multiscale, and multiplex networks, *Science*, **328** (2010), 876–878.
[23] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter and J.-P. Onnela, Community structure in time-dependent, multiscale, and multiplex networks, *Science*, **328** (2010), 876–878.
[24] T. Murata, Modularity for heterogeneous networks, in *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, ACM, 2010, 129–134.
[25] A. Potgieter, R. J. E. Cooke, K. A. April and I. O. Osunmakinde, Temporality in link prediction: Understanding social complexity, *Emergence: Complexity & Organization*, **11** (2009), 69–83.
[26] J. Reichardt and S. Bornholdt, Statistical mechanics of community detection, *Physical Review E*, **74** (2006), 016110, 14pp.
[27] K. Rushed, Seed-centric approaches for community detection in complex networks, in *Social Computing and Social Media*, Springer, 2014, 197–208.
[28] D. Suthers, J. Fusco, P. Schank, K.-H. Chu and M. Schlager, Discovery of community structures in a heterogeneous professional online network, in *2013 46th Hawaii International Conference on System Sciences (HICSS)*, IEEE, 2013, 3262–3271.
[29] L. Tang and H. Liu, Community detection and mining in social media, *Synthesis Lectures on Data Mining and Knowledge Discovery*, **2** (2010), 1–137.
[30] Y. Zied and K. Rushed, Licod: Leader-driven approach for community detection in complex networks, *Vietnam Journal of Computer Science*, (2014), p30.

Received July 2014; revised December 2014.

E-mail address: manel.hmimida@lipn.univ-paris13.fr
E-mail address: rushed.kanawati@lipn.univ-paris13.fr