Detection of overlapping communities in dynamical social networks

Rémy Cazabet, Frédéric Amblard, Chihab Hanachi

Abstract—**Community detection on networks is a well-known problem encountered in many fields, for which the existing algorithms are inefficient 1) at capturing overlaps in-between communities, 2) at detecting communities having disparities in size and density 3) at taking into account the networks' dynamics. In this paper, we propose a new algorithm (iLCD) for community detection using a radically new approach. Taking into account the dynamics of the network, it is designed for the detection of strongly overlapping communities. We first explain the main principles underlying the iLCD algorithm, introducing the two notions of intrinsic communities and longitudinal detection, and detail the algorithm. Then, we illustrate its efficiency in the case of a citation network, and then compare it with existing most efficient algorithms using a standard generator of community-based networks, the LFR benchmark.**

Index Terms—**Community detection, dynamic networks, social network analysis**

I. Introduction

THE detection of communities within networks is one of the most interesting and complex problem in the field of networks studies. For a while it was related to the problem of graph partitioning, aiming at dividing a graph into an arbitrary given number of groups and minimizing the number of edges between those groups. The counterpart of this approach is that one has to know the number and the size of the clusters one wants to obtain, which reduce the efficiency of such a tool for the detection of communities in real settings. In such latter circumstances one would at least expect a solution to find the best partition of a network without knowing initially the precise characteristics of these partitions, in particular their number and respective sizes.

Community detection in its actual definition really began in 2002 with the Girvan & Newman algorithm [1]. Since then, a very large amount of algorithms have been proposed, sometimes with great improvements in time and efficiency. In [2] Fortunato confronted the best-known algorithms, proposing a benchmark (the LFR benchmark) that generates graphs with well-defined communities. Then, he ran the different algorithms on the generated graphs and compared the communities detected with the expected ones, known by construction. According to his results, two algorithms, Infomap [3] and the fast modularity optimization by Blondel et al. [4], are the best algorithms available until now. Looking at the results of Infomap, we can say that the problem of detection in "ideal" problems, as the ones produced by the LFR benchmark [6], is solved. But real-world networks have peculiar properties that could seriously reduce the efficiency of these algorithms.

But another interesting algorithm was proposed recently, the k-clique percolation method (CPM) [5]. Although it is less efficient on classic community detection compared with the two previous ones, it proposes one more thing: it can detect overlapping communities.

If we look at the case of community detection in social networks, we immediately notice that the question of overlapping communities is crucial. For example, thinking about your personal social network, you will naturally consider that you belong to several communities: for example your family, your co-workers, college friends, and so on and so forth. These communities can be identified by topological properties, for instance clustering, i.e. an important density of the connections in-between the peoples of these communities – the members of your family are very likely to know each other. But every person also belongs to a lot of other communities, with few but essential overlaps (at least yourself), in-between those communities. The same thing can be observed in scientific networks, often captured using citation networks. Whatever the granularity of analysis, research domains are identifiable, but their boundaries are quite difficult to characterize, due to the existence of such overlaps or to the existence of scientific communities at the border of the well identified domains or even at the crossroad in-between different domains (think about bioinformatics).

Another problem of community detection is related to the definition of community. To cite Fortunato: "The first problem in graph clustering is to look for a quantitative definition of community. No definition is universally accepted." [7]

Actually, most algorithms have their own definitions of what a

Manuscript received April 15, 2010. This work was partially supported by the Future and Emerging Technologies programme FP7-COSI-ICT of the European Commission through project QLectives (grant no.: 231200).

R. Cazabet is with the Institut de Recherche en Informatique de Toulouse (UMR CNRS 5505), Université Paul Sabatier, Toulouse 3, FRANCE (phone: +33 561 556 765; fax: +33 561 556 258; e-mail: remy.cazabet@irit.fr).

F. Amblard is with the Institut de Recherche en Informatique de Toulouse (UMR CNRS 5505, Université Toulouse 1 Capitole, FRANCE (e-mail: famblard@univ-tlse1.fr).

C.Hanachi is with the Institut de Recherche en Informatique de Toulouse (UMR CNRS 5505), Université Toulouse 1 Capitole, FRANCE (e-mail: hanachi@univ-tlse1.fr).

community is, and analysis of real-world networks too, highly dependent on the context and the phenomenon you are studying. In this paper, we will study communities from the social network point of view. We think that in most cases, social communities are very close to the traditional definition of community, i.e. dense regions of the network that are weakly connected to the rest of the network, but we will not try to optimize a global function, like the modularity, as we think that in order to detect meaningful communities we have to include a part of the complexity of the underlying phenomenon.
In section 2, we introduce and explicit two notions that provide a new vision over some aspects of this problem, the longitudinal detection and the intrinsic nature of communities.
Section 3 proposes a new algorithm for community detection based on these notions. Section four describes tests on this algorithm, both to compare it with other algorithms using the LFR benchmark, and to observe its results on real networks.

II. Two concepts to characterize social communities

In this section, we introduce two notions that we use in our algorithm. On the one hand, the intrinsic nature of communities deals with what makes a community distinct from the others. On the other hand, the longitudinal detection is a new dynamical way of approaching community detection. They have in common to make the distinction between overlapping communities easier.

A. Intrinsic nature of communities

We can oppose two definitions of community:
The first one, used by most of algorithms, is what we can call a "relative" definition of community. In this perspective, a community is determined not only by nodes and edges of this community but also by the remaining part of the network, its properties, topology, and so on, i.e. the topological environment of the community. If we have the same configurations of nodes and edges in a sparse graph or in a denser one, most detection algorithms will detect the community in one of the case and not in the other. Moreover, if the algorithm detected a big community at the first run, and if we run again the same algorithm on this community only, the algorithm will generally propose a new decomposition of the community, without recognizing that in this latter case, the whole network is only one community. Even if this could make sense to detect sub-communities, we think this is pretty rare in real networks for small communities to be exactly included in bigger ones. In this perspective, studying only an isolated part of a network and ignoring outgoing edges is not efficient.
We oppose this definition to what we call *intrinsic communities*. Let's take a simple example: for illustrative purpose, if we choose as a definition that "cliques (whatever their size) are always communities and a community must be a clique", we have a definition of intrinsic communities. This means that for one particular set of nodes and edges (let's take a 4-clique for instance), we can define absolutely whether or not it is a community, whatever its topological environment. However, this is not necessarily a minimal community: if we remove edges or nodes, depending on the chosen definition of a community, we could detect other communities inside of it (3-cliques). Following the same idea, if the 4-clique is included in a 5-clique, depending on the chosen definition, we could stand that the community is the 5-clique and ignore the smaller one or not. The point of choosing the 5-clique rather than the 4-clique in this case is only a simplification choice. The important thing is, as long as a clique exists in a network, it is possible to identify it. An advantage of the intrinsic property of communities is that they can be as overlapped or intricated as possible, they always will be detected as different ones. If you think of two k-cliques, even if they have k-1 nodes in common, each one of them is by definition different from the other. Therefore, the community is totally independent of its context or topological environment. A 5-clique is as much a 5-clique in a large sparse network than in a small dense one. A final advantage deals with the determination power. Using an algorithm searching intrinsic communities in a network, it can determine whether or not this network is a community (depending on the chosen definition for a community, of course). We have to insist that the definition proposed in this paragraph for communities (defined as k-cliques) is for illustrative purpose only, the important point being the notion of *intrinsic community.*
In the proposed algorithm, as we make longitudinal detections of communities, what will define intrinsically a community will not be defined only by its edges and nodes but also by a particular pattern of development.
The CPM (Clique Percolation Method) already detects intrinsic communities: as an example, if we detect in a large graph a large community and then removing all the nodes not included in this community, the algorithm will again detect a single community, being now the whole network.
We think that relative communities can be interesting, for example if we want to partition a large graph into large parts, for instance to identify the main scientific domains using the Google scholar network or to identify national communities on a web 2.0 social network. But the corresponding problem is more related to graph partitioning than to community detection, as it does not correspond to any social reality to consider all the people of a country as a community. On the other hand, to deal with social communities, that are frequently strongly overlapped and have an important disparity among them, we think that searching for intrinsic communities will be more efficient.

B. Longitudinal detection

Trying to detect communities in a graph, we traditionally only look at a particular state of a network, a snapshot of this network at a given time. But nearly all networks have not been created in this state, they do have an history, a past that explains how they are now. This dynamics can be used to analyze and/or explain the actual characteristics of the network. As an example, the power law for degree distribution observed in many networks can be explained by the preferential attachment model [8]. According to this model, we can use a dynamical hypothesis on the construction of the network to explain as well as to reproduce real networks. We think that the formation of communities can be studied in a similar way. In the past few years, some studies like the ones by Bilgin [9] or Leskovec [10] analyzed the evolution of networks in time. Some works have already been done to study the evolution of communities [12][13] but only by successively detecting communities on several snapshots of the graph at different time steps, and then looking for the differences. A community detection algorithm that would take into account the evolution of the network would enable to identify these dynamical communities more accurately.

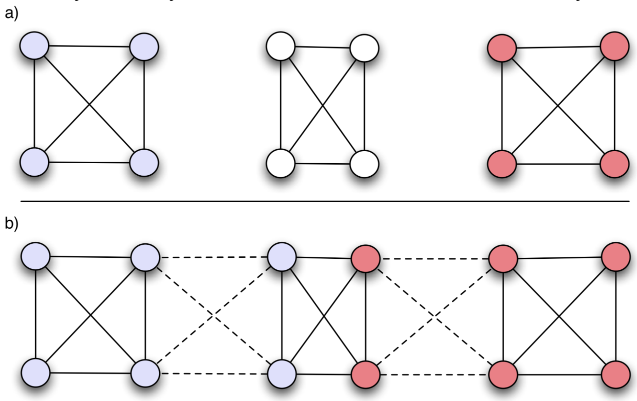


Figure 1 – in a), there are two communities (blue and red) and a set of already connected nodes (in white). In b), new edges have been created. A longitudinal detection can identify communities (for instance blue and red or blue, white and red), whereas for static algorithms, b) it remains a regular grid and no detection is possible.

Longitudinal detection can be more powerful too, because this analysis matches with a reality. As an example, think about a community as the one of the members of a sports club. At the beginning, few persons will have the intention to create this club. To do so, they will have to interact a lot with each other, forming a cluster of well-connected people. At this stage, it is possible to intuitively identify an embryonic community. Afterwards, new people joining the club will have to interact with some clubs' members, and, then, will probably be introduced to other members of the club in a kind of insertion dynamics. Though we can assume that the growth process of a community can be modeled by a kind of snowball effect, with initially a small community and new members joining gradually. This longitudinal vision can really be helpful when confronted to situations where two (or more) communities are close, with numerous links between them, but for side effects reasons. For example, the football team and the basketball teams of the same high school will have members that know well each other, because they are classmates, and they were probably even before entering their sports teams. If we look now starting with "basket-ball team" and "football team" as well-identified communities, and a pool of new students which are classmates, although they are well connected, by looking along time at who creates links with which sport team we will easily differentiate them (Figure 1)

Of course, this snowball effect must be limited, otherwise the entire network would belong to the older communities. In our model, it's the intrinsic nature of communities that will limit the detection to realistic communities only.
We have to rise the point of the availability of the longitudinal datasets needed to make such an analysis. The information concerning network dynamics does not exist for most datasets but some of them are quite naturally adapted and available, as it is the case of citation networks for example. From a technical point of view, this information could be easily obtained in web 2.0 social networks,

III. The Algorithm

A. *Description of the iLCD algorithm*

First, we need to define the input of the iLCD (intrinsic Longitudinal Community Detection) algorithm. Due to the longitudinal analysis, we will use a list of edges, ordered by their creation time, those edges could correspond to links creation among existing nodes or could also imply the creation of a new node. As some edges creations can be simultaneous (think about the publication of several articles in a given journal issue), we will use ordered sets of edges, where edges of a given set are created at the same time.
More formally, let's note $G=(V,E)$ the graph that is dynamically built and $C=<C_k>$ the set of communities that is dynamically built. Initially, G and C are empty. We then define E_{in} the set of edges in input as $E_{in} = <E_t>$ i.e. composed by ordered time-stamped sets of edges.
$E_t= <(u,v)>$ is the set of edges (u,v) created at time t
(u,v) being the edge linking nodes u and v.

```
For each time-stamped set $E_t$
    For each edge $(u,v)$ of the set $E_t$
        Add $(u,v)$ to $E$. If $u$ or $v$ is not in $V$, add it to $V$

        Determine the updates of existing communities. For each
        community $C_k$ to which $u$ (respectively $v$) belongs, try to
        integrate $v$ (resp. $u$) to $C_k$ (1)

    End for
    Update of previous communities
    If $u$ and $v$ do not already belong to the same community,
        Try to create a new community.(2)
    Merge similar communities.(3)
End for
```

We have to notice that, following our approach, each node can belong to one or more communities and that we add synchronously all links belonging to the same set E_t, calculating first the potential induced changes by all the links added and hereafter proceeding to the update. A limitation of our solution is that at a given time step t we can't add at the same time two new nodes and a link in-between. Taking the example of citation networks, it would correspond to the situation where two papers published at the same time reference each others.

1) *Update of existing communities*

We detail here the procedure concerning the addition or not of new nodes to one or several existing communities. In our approach, a node can be integrated into a given community if it satisfies two conditions. For each community, we compute (and update each time the community is modified) two values that will be used to characterize it. These are the estimation of the mean number of second neighbors (EMSN) and the estimation of the mean number of robust second neighbors (EMRSN). These values represent, for a node of the community,

- The mean number of neighbors inside the community it can access with a path of length 2 or less,
- the mean number of neighbors inside the community that can be accessed with a path of length 2 or less, by at least two different paths (therefore robust).

We do not try to compute the actual values of EMSN and EMRSN, but just an approximation of these values on a random network with the same number of nodes and edges than the considered community. Surprisingly, we could not find in the literature a formula to compute EMSN that take into account the redundancy in second neighbors, very strong in communities, though we calculate it as follows:
let d be the average degree of the nodes of the community.

$$d = \left\lfloor \frac{2l}{n} \right\rfloor$$

with
l : number of internal edges of the community
n = number of nodes of the community

We recursively calculate EMSN for a random network equivalent to the community, $f(d)$ with

$$f : [0;d] \to \Re$$

$$f(i) = \begin{cases} i = 0 \to d \\ i > 0 \to f(i-1) + (d-1) * \dfrac{n - f(i-1)}{n} \end{cases}$$

The idea is to successively count the number of neighbors encountered only once at rank two. Starting from a given node (*i=0*), we have on average *d* first neighbors. Then for each of these first neighbors, we could add on average *d - 1* new neighbors, (one of its neighbors being already counted). The probability of these neighbors to not have been counted yet corresponds to the number of nodes not already counted as neighbors $n - f(i-1)$ over the total number of nodes *n*.
The estimation of EMRSN is done in a similar way.

Then, the calculation of the two indices done, we will accept a new node if:

- The number of its neighbors at rank 2 in the community is greater than EMSN. It ensures that the community presents short paths in-between the nodes and corresponds to the quick spread of information inside communities.
- The number of its robust neighbors at rank 2 is greater than EMRSN. This ensures that the node is not connected to its second neighbors only through very few hubs. It corresponds to the property of robustness of communities. The removing of one edge or one node does not change drastically the community, or the information flow inside of it.

The main advantage of these adaptive thresholds is that they allow the existence in the same graph of big and small, dense and sparse communities, each having its own characteristic values we use to determine the belonging or not of new added nodes.

2) *Creation of a new community*

We now detail the procedure concerning the potential creation of a new community when new edges are added to the network at a given timestep. Each time a new edge is added to the network, we determine weather it enables to form a minimal community or not. The minimal community is a predefined pattern, which corresponds to the smallest community we want to detect. This pattern could be cliques of 3 or 4 nodes, but if we want to detect only bigger and denser communities, or if we are working on very dense networks, we can choose bigger cliques as the minimal pattern to detect.
Therefore, if this minimal pattern is detected, we create a new community. We then add to this community all the nodes composing the pattern. After that, we try to integrate to the community all the neighbors of every node of the pattern, simultaneously, as described in the preceding section.

3) *Merging similar communities*

Hereafter the addition of new nodes and edges to the network, after determining whether or not they join existing communities and whether or not they do create new ones, we merge the communities that are very close to each other. To proceed, at each step, we merge all communities that have more than a certain ratio of nodes in common. Actually, it is frequent that two communities initially detected as separated become more and more overlapped along time and become nearly identical. The definition of the threshold upon which we consider communities as identical is a though point, and depends on the results we want to obtain. If we want to conserve only drastically different communities, i.e. with a minimal overlap, we should set a low threshold (0.2,0.3). On the contrary, if we want to obtain more detailed results including more intricate communities, we should raise this threshold (0.7, 0.9).
Depending then on the retained threshold, when two communities are detected as similar, we simply remove the younger one. In most cases, this younger community is more

likely to be just an artifact caused by the simultaneous inclusion of a set of nodes into an older community. We could maybe obtain better results by merging the two communities, however in this first version of the algorithm, we tried to limit as much as possible the use of uncertain heuristics.

B. Parameters

This algorithm finally uses only two parameters, which do not play really on the detection process but rather on the granularity of the results wanted. These parameters are:
- The size of the minimal clique we want to detect **Mk**, which affect the creation of new communities. On small or sparse graphs, a value of 3 will be optimal, when a value of 4 will be more appropriate for bigger or denser graphs, to avoid detecting several times the same community.
-The threshold for community merging **Tm**, dealing with the removal of similar communities, plays on the overlap between two communities we tolerate before than to decide they correspond to a single one.

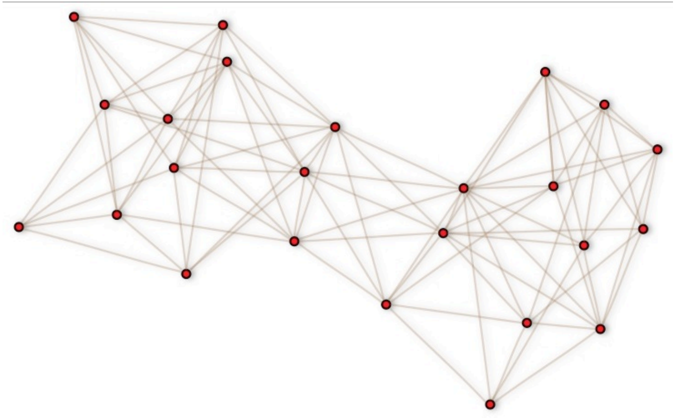


Figure 2 - CPM (k=3 or k=4) detect only one community in this network, while longitudinal detection detect two communities, with an overlap in the middle.

C. Intuitive explanation

Thinking about the meaning of the k-clique percolation method (CPM) for a social network, with k=4, we can interpret it as the following heuristic: a person belongs to a community if he has at least 3 relations inside this community and if each of these relations are linked together, moreover all of them might also have a relation with a fourth person in the community. We easily understand that it's very likely to well detect communities based of the properties of triangle closure and high clustering. However, it seems a little arbitrary and it also has limitations. First, it depends strongly on the mean degree of nodes. But the main problem is probably that the condition is not restrictive enough. A node connected only to a single 4-clique in a large community will be added to this community, when it does not make always sense. Moreover, when two communities have some high-connected nodes in common (a frequent case in real networks, think for example of two successive classes of students, with a certain number of students repeating a year), CPM will likely consider the two classes as one community, when it seems more relevant to divide in 2 or 3 overlapping communities (Figure 2).
If we try to explain our algorithm in a similar way, we can summarize it to two rules:
to be added to a community, a node must:

- Be able to access easily (in two step) to most of the community (at least as much as the others nodes of the community)
- Have a robust access to other nodes of the community (at least as much as the others nodes of the community).

That's why our algorithm might be more reliable, particularly in the case of important overlaps in-between communities. We can resume it by saying that CPM is able to detect individual overlap (random nodes of some communities belongs to other communities), but not communities having a set of nodes in common.

D. Complexity

It's nearly impossible to really determine the complexity of this algorithm, as it strongly depends on the size and number of communities, the number of cliques, parameters, and so on. However, we can give an insight into it:
The complexity will at least be proportional to the number of edges, as we have to do a computation each time we add a new edge. On the other hand, this calculation only depends on local information: second neighborhood, and communities involved in this neighborhood. We can ensure that the complexity does not grow exponentially, but it can rise strongly with high degrees and big communities.
In term of memory, we only need at each step to memorize the current network (edges and nodes), and the current communities.
Finally, we made some computations on graphs of different size. On the graphs generated by LFR benchmark used for comparison in this paper (up to 5000 nodes), the algorithm end up in a fistful of seconds.
The computation on a larger real network (the citation network of the "High Energy Physic – Theory" Arxiv section, from 1992 to 2003, composed of approximately 27.000 nodes and 350.000 edges) with our current implementation in JAVA requires less than 10 minutes on a personal computer.
On the same computer, CPM stops after 5 minutes of running due to memory overflow.

IV. Results

Like every community detection algorithm, this algorithm gives as a result a set of communities. However, it tries to detect communities at every level, which means that some communities can be included in others, some can be highly overlapping, some nodes can not belong to any of the communities, and so on. Depending on the aim of the application, it could be interesting to do some filtering, for example by removing the very small communities.

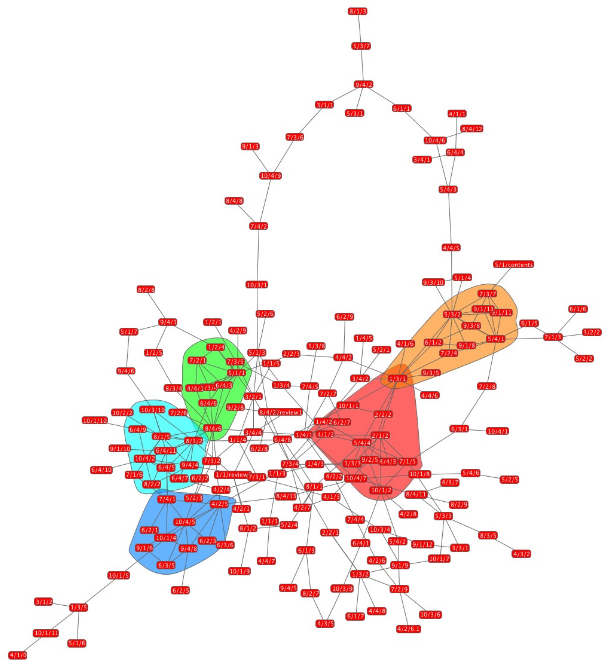


Figure 3 - Communities detected in the JASSS citation network with longitudinal detection

A. *JASSS citation network.*

The Journal of Artificial Societies and Social Simulation (JASSS) has already been studied in a perspective of community evolution [12] but using a static network analysis on two different periods. We decided to apply our own algorithm to the same case study enabling a qualitative comparison among results. The JASSS dataset considered here is composed of all articles published in JASSS between 1998 (date of the creation of the journal) and 2008 and citing at least one other article published in JASSS. Nodes are papers and edges are citations among papers, however we consider the graph as undirected, as link orientation does not play any role in the current version of our algorithm.

As we can see on (Figure 3), it is a sparse network, with a lot of nodes having only one or two edges with the other JASSS papers. The aim of the community detection on this graph is to find the most important subjects treated by several papers. At the exception of the k-clique percolation method, none of the traditional algorithms can be used to do that: they would try to do a complete partition of the network, when at the evidence a majority of nodes do not belong to any clear community (at least from the data we have). It is only by studying intrinsic communities that we can obtain interesting results.

In figure 3, we displayed all communities detected by longitudinal detection that have more than 3 nodes. The algorithm detects 5 communities. Visually, one can observe some overlaps in-between some communities. Moreover, one can notice that some of the communities, even without overlaps, are closer than others. By looking at the keywords of the publication of the corresponding nodes, we can easily attribute research themes to these communities, which are quite homogeneous from a qualitative point of view. The 3 communities that are on the left concern directly multi-agent simulation (*MAS and industrial district; replication of MAS; MAS and Role Playing Games*). The last two in the right part concern: *norms and reputations* (the bottom one) and *opinion dynamics* (upper one).

Even though the dataset and the techniques are very different from the ones used by Meyer and Lorscheid [12] (they were looking at co-citation networks rather than at citation networks), the main communities detected are quite similar. Moreover, communities are meaningful and, given the dataset, it seems difficult to obtain a better detection.

Moreover, our results can help to make a longitudinal analysis. For example, by looking at the norm & reputation community, we can follow its evolution (Figure 4): it was first detected with 3 nodes linked together in 2001. Then, one paper of 2002 extended the community by 3 nodes (itself and two older papers that were already weakly linked to the community). A new paper was added in 2004, and a final one in 2007.

We can also conduce a comparative analysis in time between communities. Compared to the norm & reputation community, the community about model replication was only detected in 2006, and is mainly composed of recent papers. Finally, one paper, which belongs to two communities, is added in one of them in 2002 and in another one in 2005. We have to remind that the date of inclusion of a paper within a community does not necessarily correspond to its publication date.

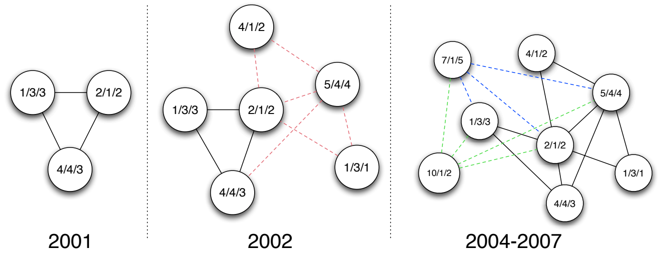
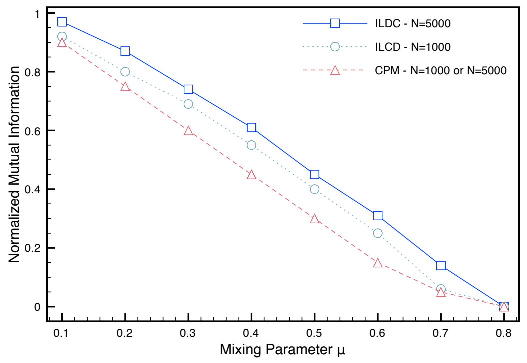


Figure 4 - Growth of the "norm & reputation" community.

A final interesting result on this example concerns the node (1/3/1). This paper, entitled "Understanding Complex Social Dynamics: A Plea For Cellular Automata Based Modelling", by Hegselmann and Flache [14], is a general article dealing with the use of Cellular automata for the simulation of social dynamics. By looking at the results provided by our algorithm, we can observe that this paper belongs concurrently to the periphery of both "norm & simulation" and "opinion dynamics" communities. It also belongs to a small community of three nodes about Cellular Automata approaches. We think that it is a good example of overlapping, where a traditional algorithm will encounter some troubles to do an assignation of this node to any particular community, knowing that this particular node has several edges with each one of these 3 communities.



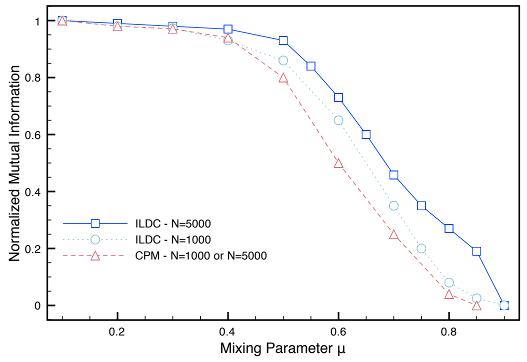


Figure 5 - Comparison between CPM and iLCD. Up: big communities. Down: small communities.

B. LFR benchmark

The confrontation of the efficiency of algorithms is essential to learn their strengths and weaknesses. Probably the best paper published recently on this issue was written by Lancichinatti et al. [6], and uses the LFR Benchmark to compare communities detected by several algorithms with the real communities known by construction. The idea is to generate graphs, making vary three parameters:

- The size of the network (from 1000 to 5000 nodes) (N)
- The size of the communities (either between 10 and 50 nodes or between 20 and 100 nodes) (C)
- The average ratio, for each node, between edges to its community and edges with the remaining of the network (µ).

Unfortunately, only one of the evaluated algorithms (CPM) is able to deal with overlapping communities. In the following, we will compare the results obtained by our algorithm (iLCD) and CPM, both with and without overlap. The two algorithms have parameters; we decided to fix them for all tests, as it is nearly impossible to adjust them efficiently for real graphs for which we do not know the solution to be found. For CPM, we chose the value that appeared to be the more efficient in the article by Lancichinatti, **k**=4. For longitudinal detection, we choose a merging threshold **Tm**=0,3 and a minimal clique **Mk**=4. **Tm** is not really sensible, tests have shown that adding or removing 0,15 does not affect significantly the results (it can be slightly better or worse depending on the cases).

Before presenting the results, we must underline that the LFR benchmark is far to generate networks that best apply to our algorithm. First, obviously, they do not have any longitudinal aspect. This is a crucial problem, and we will discuss in the analysis below that it is effectively causing biases. The other problem is that the LFR benchmark does not try to produce realistic networks in their structure, apart from ensuring a power law for the degree distribution. In real networks, connections outside of the communities are not random. There are communities with more links among them than others (see the results on the JASSS citation network), and the communities themselves have some properties, like core-periphery structure. However, despite these problems, results are quite convincing.

We began our tests with graphs without overlap. In Figure 5, we can see that our algorithm gives results that are comparable to the CPM. The bigger the graph, the better the results. While results with big communities could be improved, results for small communities are close to the ones of the best known algorithms.

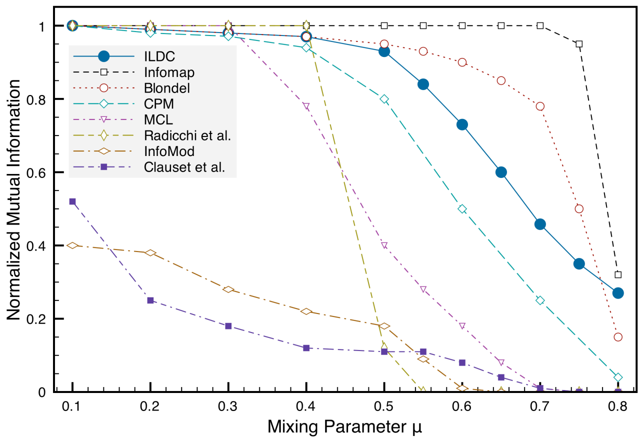


Figure 6 - Comparison of several community detection algorithms on the same case (iLCD with plain line).

When we look at community detection with overlapping communities as it has been done in [6], we can do the same observations: results are of the same order, though lightly better for iLCD and particularly for big graphs.

In Figure 6, we compared the efficiency of several algorithms on the case in which the communities are the more reliably defined: small communities and big graphs without overlap. For a given degree, a small community is more precisely defined than a bigger one because of a relatively higher clustering. In a big network, for constant size of communities, there are less multiple-edges between communities than in a smaller network.

Compared algorithms are the best known of the field, with

these denominations : CPM, Infomap, Blondel, MCL, Radicchi et al. InfoMod, Clauset et al.
We ended our tests by a comparison of the CPM and iLCD on a case of very important overlap: we generate networks with the LFR benchmark, with the constraint that every node belongs to 3 different communities (and has no edge outside of these communities). Therefore, each node has more edges outside than inside of its community. However, according to the previous tests, belonging to communities is still strong enough to be detected. This time, we do not make vary the edge repartition but only the density. Intuitively, the higher the density, the better defined are the communities. If the density is too low, communities will be too sparse to be identified (The more edges nodes have inside their communities, the easier it is to detect their belonging to these communities).
On Figure 7, we observe as predicted that with our algorithm, the denser the graph is, the better the detection. On the other hand, CPM is far more sensitive to the modifications of density. Above a threshold (for x around 20), communities began to have cliques in common, and then merge into one big community. We can modify the size of the cliques (k) of the CPM to overcome this problem, however, to study real graphs, it will be hard to choose the right value for k, as a too high value would fail to detect sparser communities. The problem stays unsolvable when the same network includes both sparse and dense communities, as it could be the case for real social networks. One of the advantages of our algorithm is that it is less sensitive to variations of the network's properties, an advantage to study real networks.

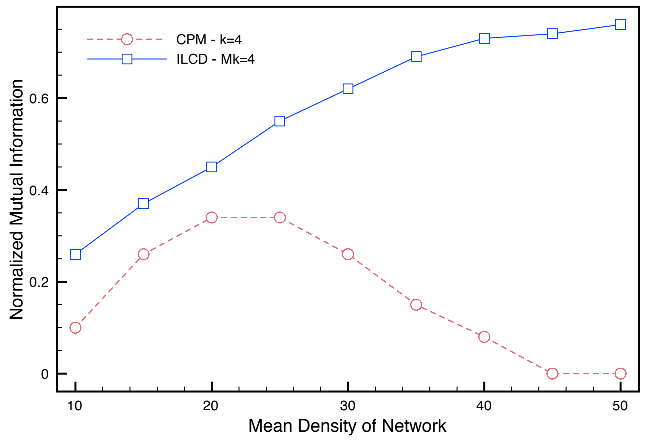


Figure 7 - Comparison of CPM and iLCD when varying the density of the network.

V. Conclusion

The aim of this paper is to propose a new algorithm able to deal with strong overlaps in-between communities. However there are already numerous algorithms that seem to yield good results for community detection on graphs without overlaps; on many real graphs, and especially on social networks, there are quite important overlaps preventing these algorithms to work correctly and to produce interesting results. The only algorithm thought to deal with this problem was CPM. The algorithm we described here is an alternative to CPM, using a completely different mechanism, giving results equals or better in most of the cases, less sensitive to parameters, and more adapted to the specificities of real-world networks and communities.
However, this algorithm is still a first step that needs to be improved. First, if it takes into account the growth of communities along time, it is not yet able to deal with the natural evolution of communities: shrinking, merging and splitting. Secondly, the tests done on the LFR benchmark have shown some weaknesses with big communities. Tests on more adapted datasets must be conducted.
One problem to test effectively the efficiency of our algorithm is the difficulty to find real graphs with tagged communities, and we think that generators like the LFR benchmark are not realistic enough to reveal the weaknesses and strengths of some of them.

Acknowledgment

The authors would like to thank Iris Lorscheid and Matthias Meyer for sharing their JASSS dataset as well as for interesting discussion on the subject.

References

[1] M. Girvan and M.E.J. Newman, "Community structure in social and biological networks", *PNAS*, vol.99, no. 12, pp. 7821–7826, 2002.
[2] A. Lancichinetti et S. Fortunato, "Community detection algorithms: A comparative analysis," *Physical Review E*, vol. 80, Nov. 2009, p. 056117.
[3] M. Rosvall et C.T. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks," *Proceedings of the National Academy of Sciences*, vol. 104, Mai. 2007, p. 7327-7331.
[4] V.D. Blondel, J. Guillaume, R. Lambiotte, et E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, 2008, p. P10008.
[5] G. Palla, I. Derenyi, I. Farkas, et T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, Juin. 2005, p. 814-818.
[6] A. Lancichinetti, S. Fortunato, et F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, vol. 78, Oct. 2008, p. 046110.
[7] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, Fév. 2010, p. 75-174.
[8] A. Barabási et R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, Oct. 1999, p. 509-512.
[9] C.C. Bilgin et B. Yener, "Dynamic Network Evolution: Models, Clustering, Anomaly Detection," Rensselaer Polytechnic Institute, Tech. Rep.,2008
[10] J. Leskovec, J. Kleinberg, et C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, Chicago, Illinois, USA: ACM, 2005, p. 177-187.
[11] P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, et J. Onnela, "Community Structure in Time-Dependent, Multiscale, and Multiplex Networks," Nov. 2009.
[12] I.L.A.K.G.T. Matthias Meyer, "The Development of Social Simulation as Reflected in the First Ten Years of JASSS: a Citation and Co-Citation Analysis," Oct. 2009.
[13] G. Palla, A. Barabasi, et T. Vicsek, "Quantifying social group evolution," *Nature*, vol. 446, Avr. 2007, p. 664-667.
[14] Andreas Flache et Rainer Hegselmann, "Understanding Complex Social Dynamics: a Plea for Cellular Automata Based Modelling," *Journal of Artificial Societies and Social Simulation*, vol. 1, 1998.