Acta Cybernetica **20** (2011) 35–52.

Dynamic Communities and their Detection

András Bóta*, Miklós Krész†, and András Pluhár‡

Abstract

Overlapping community detection has already become an interesting problem in data mining and also a useful technique in applications. This underlines the importance of following the lifetime of communities in real graphs. Palla et al. developed a promising method, and analyzed community evolution on two large databases [23]. We have followed their footsteps in analyzing large real-world databases and found, that the framework they use to describe the dynamics of communities is insufficient for our data. The method used by Palla et al. is also dependent on a very special community detection algorithm, the clique percolation method, and on its monotonic nature. In this paper we propose an extension of the basic community events described in [23] and a method capable of handling communities found a non-monotonic community detection algorithm. We also report on findings that came from the tests on real social graphs.

Keywords: graph mining, network analysis, community, community detection, dynamic communities

1 Introduction

The analysis of adaptive networks is considered to be a traditional research field, which in recent years, has received a new impulse, thanks to the variety of available test databases [3, 20]. These graphs are so large in some cases, that only the fastest, near linear time algorithms have a chance of tackling the given tasks. One of the approaches to network analysis, community detection has received a lot of attention both from the point of theory and applications [1, 6, 8, 9, 10, 12, 15, 19, 22, 24, 25]. The definition of communities centers around dense subgraphs of the network. In traditional community detection, we are looking for disjoint subsets of vertices, that are connected to each other more closely, that to the rest of the graph. [1] In overlapping community detection, the subsets are not disjoint. Hereafter by the

*University of Szeged, E-mail: bandras@inf.u-szeged.hu
†University of Szeged, E-mail: kresz@jgypk.u-szeged.hu
‡University of Szeged, E-mail: pluhar@inf.u-szeged.hu

[1] Note that non-overlapping communities, or *clusters* had been researched even earlier. The clustering methods have huge literature, we refer to those notions only when they are significant in the case of overlapping communities.

DOI: 10.14232/actacyb.20.1.2011.4

notion of communities we will mean overlapping communities, otherwise we will use either clusters or non-overlapping communities. We will also take several basic definitions from graph theory [4].

However, communities are not static, they can change or even disappear in time; this might have caused the split of a club in the famous experiment of Zachary [27].

The dynamics of these networks is usually represented as a series of graphs $\{G_i\}_{i\in\mathcal{T}}$, where $\mathcal{T} = \{1,\ldots,\ell\}$ represents a discrete time set. The number of time instances usually depends on two things: the length of the period, in which we observe the network, and the resolution, which governs, that in a given time period, how many "snapshots" do we take from the network. Although one might try to consider the whole history of the graph series in one step, for large graphs this is not feasible. Both Asur et al. [2] and Palla et al. [23] have chosen the path of following the entities from G_i to G_{i+1}, and reconstructing the whole process out of these. Note that Asur et al. deals with clusters, while Palla et al. follows the lifetime of (overlapping) communities. They both took a normative approach to communities: they have decided upon the possible events that might happen to a community. In other words, they have simply listed the basic events of a hypothetical classification. We reproduce some of the relevant work of Palla et al. as follows.

1.1 Basic events

The basic events described in [23] are the following:

- Birth, when a new community emerges without predecessor.
- Death, when a community disappears without successor.
- Merging, when several communities join together to form a new community.
- Splitting, when a community splits into several new communities.
- Growth, when a community gains new members.
- Contraction, when a community loses members.

Assuming the above described events, the problem is reduced to the following. Given the graphs G_1 and G_2, describing the starting and destination graphs, compute the set of communities in both, let those be $\mathcal{K}_1$ and $\mathcal{K}_2$. Then find a relation $\mathcal{R}$ on $\mathcal{K}_1 \times \mathcal{K}_2$ in an "obvious way"; if a $C_1 \in \mathcal{K}_1$ is in no relation, then it is a *death.* If $C_1 \in \mathcal{K}_1$ is in relation with $C_1^2,\ldots,C_\ell^2 \in \mathcal{K}_2$ and $C_i^2 \subset C_1$ for $i \in [1,\ldots,\ell]$, then C_1 *splits into* $C_1^2,\ldots,C_\ell^2$. Similarly, if $C_2 \in \mathcal{K}_2$ is in no relation, then it is a *birth.* If $C_2 \in \mathcal{K}_2$ is in relation with $C_1^1,\ldots,C_\ell^1 \in \mathcal{K}_1$ and $C_i^1 \subset C_2$ for $i \in [1,\ldots,\ell]$, then $C_1^1,\ldots,C_\ell^1$ *merged into* C_2. Finally a related pair (C_1,C_2), $C_i \in \mathcal{K}_i$ for $i \in \{1,2\}$, should mean a *growth (contraction)* if $C_1 \subset C_2$ ($C_1 \supset C_2$).

1.2 Implementation

Computing the relation $\mathcal{R}$, assuming that the basic events cover all possible cases, is not hard theoretically. In practice this is different, since for large graphs a naive approach of considering all pairs $(C_1, C_2) \in \mathcal{K}_1 \times \mathcal{K}_2$ is too costly. The way out of this quadratic complexity is an idea due to I. Derényi [23]. They assume that the communities are always given by the Clique percolation method (CPM) [1, 22], which is *monotonic*. More exactly, if H and G are graphs on the same vertex set, $E(H) \subset E(G)$ and $C \in \mathcal{K}_H$ then there exists a $C' \in \mathcal{K}_U$, such that $C \subset C'$, where $\mathcal{K}_H$ and $\mathcal{K}_G$ are the sets of communities of H and G, respectively.

Then the *union graph* U is formed such that $V(U) := V(G_1) \cup V(G_2)$, $E(U) := E(G_1) \cup E(G_2)$ and $\mathcal{K}_U$ is computed. Now instead of going through all (C_1, C_2) pairs from $\mathcal{K}_1 \times \mathcal{K}_2$, one needs to check only those pairs for which there is a $C' \in \mathcal{K}_U$, such that $(C_1, C_2) \subset C'$.

1.3 Results of Palla et al.

By applying this method, they get a convincing results on the dynamics of communities. The most significant results are: the larger a community the older it is. The expected lifetime of a community increases with its size. A small community is more stable if it does not change its members, while it is the opposite for large communities. Let us note that the methodology is crucial, for the definitions, time scale and database the reader should consult the paper [23].

2 Problems

We have conducted a similar research on two large social networks described in the results chapter, meaning we analyzed the changes in the community structure of the corresponding networks. We tried to adapt the methodology of [23] with little success. To find communities, we used CFinder[2] for the CPM, resulting in the computational issues described in subsection 2.2. Then we decided on using the implementation of the N^{++} method[3] developed in [9], which was finally able to handle our networks. We have also found, that the framework they have used is insufficient to describe the community dynamics in our database. One of the reasons for this could be the difference between our database and the one they have used. This will be discussed in the results section. Another reason might be the difference between the community detection algorithms. This will be discussed in section 2.3.

2.1 Basic events

In our experience, the description of basic events is not complete. The deviations from the description have at least three main causes. (i) The time scale is too large,

[2]The software was downloaded from the page http://angel.elte.hu/cfinder/

[3]We would like to express our thanks for obtaining free access to the appropriate softwares.

and G_1 differs significantly from G_2. This is unavoidable, since in several cases the measurement intervals are given. (ii) A community might become extinct not by splitting, but by leaving behind a number of overlapping communities on the same vertex set[4]; we will discuss this in depth in subsection 3.1. (iii) In dense real graphs, it happens frequently that a set of overlapping communities change into another set of communities, and the relations cannot be easily mapped.

2.2 Computational issues

For large dense graphs the CPM is time inefficient. Deciding the proper value of the parameter k is also problematic. The other problem is, that for some benchmark graphs, e.g. the Zachary, the CPM performs poorly. It is natural to try out other community detection methods, since those might give better predictions in applications [8, 10, 19].

2.3 Implementation

The method relies on the monotonicity of CPM. In general, communities do not behave this way, they might split when adding edges to a graph, or merge when deleting edges. This phenomenon makes it harder to map the communities $\mathcal{K}_1$ and $\mathcal{K}_2$.

3 Solutions

To motivate our solution, we analyze a small problem in depth. It shows that the introduction of new classes for community events are unavoidable.

3.1 Motivation

The most obvious way to deal with the problem mentioned in subsection 2.1 (i) is an artificial refinement of the time scale. That is, we fix the list of the changes that happened between G_i and G_{i+1}, and insert new graphs into the series, such that each new graph differs from the previous one in only one item. However, we will see that changing this order changes both the number and types of appearing community events, which implies, that the artificial refinement of the time scale should be avoided in practice. More importantly, this example will show us, in correspondence with problem (ii), that the seven basic events defined above are inadequate when dealing with differences larger than one.

We illustrate the above mentioned problems with Zachary's karate club network [27]. Five edges were removed from the network one by one, in two different se-

[4]Let A, B, C and D be cliques. Add a few edges between among those, such that $A \cap B$ and $C \cup D$ are the two resulting communities. Deleting two and adding two edges, $A \cap C$ and $B \cap D$ will be the new communities, which does not fit in the scheme.

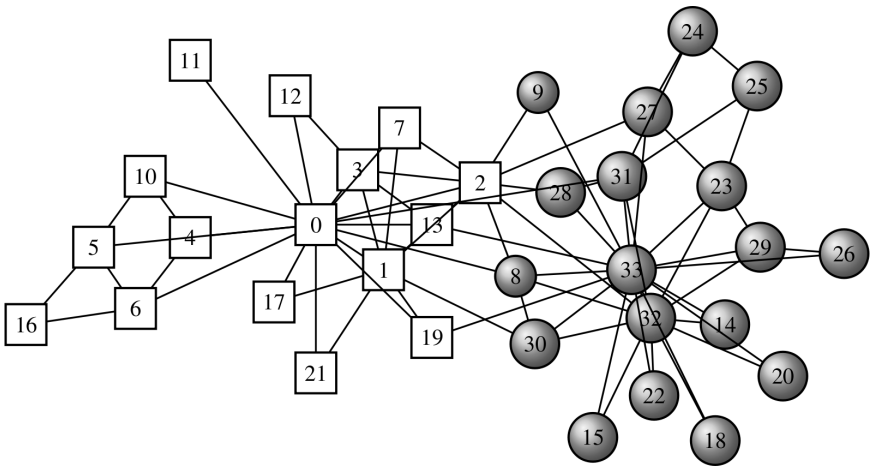

Figure 1: Zachary's karate club network. The boxes and naughts indicate the vertices corresponding to the split that has appeared during the original experiment.

Original	a	b	c	d	Final
7	7	7	7	7	**13**
9	9	9	9	9	**13**
10	10	10	10_a	10_a	10_a
			10_b	10_{ba}	**13**
				10_{bb}	**13**

Table 1: The changes in the community structure with the first edge removal sequence. Communities 1 through 6, 8, 11 and 12 do not change, and are omitted.

quences[5]. The N^{++} algorithm was used for the purpose of detecting communities; the important parts of the outputs are summarized in Tables 1 and 2.

In the first sequence, edges a, b, c, d and e are removed in this order. After removing the first two edges, nothing changes. In the next step however, community 10 containing the nodes $\{0, 1, 2, 3, 7, 13, 19\}$ splits into communities $10_a : \{0, 1, 3, 7\}$ and $10_b : \{0, 1, 2, 13, 19\}$. In the next step, 10_b splits further into $10_{ba} : \{1, 2, 19\}$ and $10_{bb} : \{0, 1, 2, 13\}$. After removing the fifth edge from the network, several communities merge together, namely $7 : \{0, 1, 17, 21\}$, $9 : \{0, 2, 8, 32\}$, $10_{ba} : \{1, 2, 19\}$ and $10_{bb} : \{0, 1, 2, 13\}$. Out of these a new community, 13, is born: $\{0, 1, 2, 8, 13, 17, 19, 21, 32\}$.

Thus far, the basic community events defined in [23] were sufficient for the description. What happens however, if we compare the communities from the original graph with the communities of the final graph? Ignoring those communities

[5]The edges $a = (0, 13)$, $b = (0, 19)$, $c = (2, 3)$, $d = (2, 7)$ and $e = (3, 19)$.

Original	c	a	e	d	Final
7	7	7	7	7	**13**
9	9	9	9	9	**13**
10	10	10	10_a	10_c	**13**
			10_b	10_d	10_e

Table 2: Changes of the communities in the second sequence. Communities 1 through 6, 8, 11 and 12 do not change, and are omitted.

that did not change, we have number $7, 9$ and 10 in the original graph, and 10_a and 13 in the modified graph. It would seem that community 10 is involved in a contraction event resulting in 10_a. Communities 7 and 9 merge into the new community 13, but 13 also acquires some nodes from community 10, meaning it is also a growth event. This gets further complicated considering the results of the one by one edge removal : It seems that the first event, that took place was a split event. One way to solve this problem is to extend the number of community events used by the algorithm with more complex events that involve multiple basic ones. Before deciding which combinations should be used, let us examine the second edge removal sequence.

In the second sequence, edges c, a, e, d and b are removed in this order. Despite the different sequence, the first few steps are the same until community 10 splits apart. The resulting communities are different however, with 10_a being $\{1, 2, 13\}$ and $10_b : \{0, 1, 2, 3, 7, 19\}$. In the next step, 10_a is involved in a growth event, stealing a node from 10_b, which is in a contraction event, resulting in $10_c : \{1, 2, 13, 19\}$ and $10_d : \{0, 1, 2, 3, 7\}$. It is obvious that the two events are connected, but the original framework does not allow such complex situations.

In the final step, community 10_d loses another node resulting in $10_e : \{0, 1, 3, 7\}$ which corresponds to 10_a from the previous example. A merge event occurs at the same time, joining communities number $7, 9$ and 10_c. It is important to note, that 10_c is not identical to any community in the previous example, not even 10_b.

This example has two important consequences. First, if one tries to refine the time scale by creating an artificial series of graphs by adding or removing edges one by one, the result depends on the order of edges, rendering this approach meaningless. In the rest of the paper, we will not use this naive idea for solving the problem at hand.

The second consequence is closely tied to the first one. It appears from the example above, that the basic events used to describe the changes of communities are satisfactory only if we change the graph one edge at a time. In all the other cases, more complex events, combinations of the basic events, are required.

3.2 Extending the number of community events

Using all possible combinations of the basic events is not necessary, as it would over-complicate the results of the algorithm. The first four events listed below are straightforward, each one is a combination of two basic events. The obscure event represents combinations of possibly more than two events, including merge-split, merge-split-merge, grow-split-merge, and so on. This means, that this event could be divided further into other events, but using the test data available we have found, that these five additional events are sufficient for describing community dynamics. With them, the changes in community structure can be identified reasonably well without unnecessary complications of the algorithm.

Tables 3 and 4 list the number of different community events found. A more detailed analysis of the results will be given in the results section. In the first one, the number of newly introduced merge and split variants are more numerous than their original counterparts. The number of obscure cases is somewhat higher, but still has the same magnitude. In table 4 the newly introduced events are far less numerous, but still relevant. This justifies the introduction of these events, and also implies, that further dividing the obscure cases would result in categories containing a very few number of events.

- Grow-merge, several communities join together, and also absorb some additional members.
- Contraction-merge, several communities join together, but loose some members in the process.
- Grow-split, a community splits into several communities, but these communities absorb additional members.
- Contraction-split, a community splits into several communities, and these communities also loose members.
- Obscure case. Multiple communities are involved from both $\mathcal{K}_1$ and $\mathcal{K}_2$, with their members reordering.

With respect to the above framework, we redefine the concepts of the split and merge events. Given the community sets $\mathcal{K}_1$ and $\mathcal{K}_2$, and a community $C_1 \in \mathcal{K}_1$ which is in relation with communities $C_1^2, \ldots, C_\ell^2 \in \mathcal{K}_2$, we define the split event if $|C_1| = |C_1^2 \cup C_2^2 \cup \cdots \cup C_\ell^2|$. If $|C_1| < |C_1^2 \cup C_2^2 \cup \cdots \cup C_\ell^2|$, then we define the grow-split event, and finally if $|C_1| > |C_1^2 \cup C_2^2 \cup \cdots \cup C_\ell^2|$, we define the contraction-split event.

The definition of the merge event is symmetrical: given the community sets $\mathcal{K}_1$ and $\mathcal{K}_2$, and communities $C_1^1, \ldots, C_\ell^1 \in \mathcal{K}_1$ which are in relation with a community $C_2 \in \mathcal{K}_2$, if $|C_1^1 \cup C_2^1 \cup \cdots \cup C_\ell^1| = |C_2|$, then we define the merge event. If $|C_1^1 \cup C_2^1 \cup \cdots \cup C_\ell^1| < |C_2|$, then we define the grow-merge event, and finally if $|C_1^1 \cup C_2^1 \cup \cdots \cup C_\ell^1| > |C_2|$, we define the contraction-merge event.

The obscure case is less strictly defined. If there are multiple communities from $\mathcal{K}_1$ and $\mathcal{K}_2$ connected by a relation, without further analysis, we classify this relation as an obscure event.

4 Details of the algorithm

The method described in this section follows the idea of I. Derényi [23], with a few important modifications. We will adopt the concept of the union graph U, and use it to solve the originally quadratic problem described in subsection 1.2 in almost the same way as in [23]. Our additional work centers around two important modifications of the original method.

The first one is abandoning the requirement of monotonicity. This means, that we can no longer assign only one community from $\mathcal{K}_U$ to any community from the original graphs. The solution of this problem is described in the subsection 4.4.

The second modification incorporates the detection of the extended community events. The solution to this is fairly straightforward, and will be discussed in subsection 4.5.

We will also describe an optional modification of the original idea of [23]. Instead of using U, we will use I, the *intersection* graph of G_1 and G_2. That is $V(I) := V(G_1) \cap V(G_2)$, $E(I) := E(G_1) \cap E(G_2)$, and $\mathcal{K}_I$ represents the communities of the intersection graph I. [6] We will describe this approach in 4.3.

4.1 Overview

The input of the algorithm consists of three community sets $\mathcal{K}_1, \mathcal{K}_2$ and $\mathcal{K}_U$, where $\mathcal{K}_U$ is the community set of U.

The output of the algorithm is a relation $\mathcal{R}$ on $\mathcal{K}_1 \times \mathcal{K}_2$, corresponding to community events described in [23] and subsection 3.2.

The algorithm can be divided into two phases. The first phase creates a relation $\mathcal{R}_1$ on $\mathcal{K}_1 \times \mathcal{K}_U$ and $\mathcal{R}_2$ on $\mathcal{K}_2 \times \mathcal{K}_U$. The second phase combines $\mathcal{R}_1$ and $\mathcal{R}_2$ into $\mathcal{R}$. Because of the non-monotonic nature of the community detection algorithm, a preprocessing step is required before executing the second phase.

4.2 First phase

In the first phase we search for relations among $C_i^1 \in \mathcal{K}_1$ for all i and $C_\ell^u \in \mathcal{K}_U$ for all ℓ. We are looking for two types of relations. The first type is the *exact match:* $C_i^1 = C_\ell^u$. The second is a *contain match:* $C_i^1 \subset C_\ell^u$. Because of the non-monotonic nature of the community detection algorithm $C_\ell^u \subset C_i^1$ might also occur. This case also counts as a contain match.

For the purpose of finding these relations, we iterate over the elements of $\mathcal{K}_1$ and compare each C_i^1 to every element of $\mathcal{K}_U$. If we find a contain match, we create

[6] The community sets can be created by any, possibly non-monotonic, community detection algorithm.

a relation $r_1(C_i^1, C_\ell^u)$. If we find an exact match, we also create $r_1(C_i^1, C_\ell^u)$, but we ignore C_ℓ^u in the subsequent computations. This step is repeated for $\mathcal{K}_2$ and $\mathcal{K}_U$.

4.3 Intersection approach

Since a community detection algorithm is not necessarily monotonic, there might be elements of $\mathcal{K}_1$ or $\mathcal{K}_2$ that are not in relation with any element of $\mathcal{K}_U$, these were counted as "deaths" before. The intersection approach tries to solve this problem by replacing $\mathcal{K}_U$ with $\mathcal{K}_I$. The only difference in the implementation is, that in the first phase $\mathcal{K}_1$ (and $\mathcal{K}_2$ later) is replaced with $\mathcal{K}_I$, and $\mathcal{K}_U$ is replaced with $\mathcal{K}_1$ (and $\mathcal{K}_2$ later). After the first phase has finished, the algorithm continues after inverting the relations r_1 and r_2.

The intersection method provides almost the same results as the union approach with a few exceptions, that will be noted in the results chapter. The size of the intersection graph I is smaller than or equal (in the special case when $G_1 = G_2$) to the size of the union graph U. From the computational point of view, this implies that the running time of the community detection algorithm should be lower on I. Other than this, the use of the intersection approach is completely optional.

4.4 Preprocessing

As we noted before, there may be more than one element of $\mathcal{K}_U$, that is in relation r_1 with a given C_i^1. The same holds for C_j^2 and the relation r_2. To solve this, we put a new, fictitious community C_a^u to $\mathcal{K}_u$, set $r_1(C_i^1, C_a^u)$, and delete all former relations containing C_i^1. The same is done for C_j^2. If C_i^1 and C_j^2 were in relation with the same elements of $\mathcal{K}_u$, then the same C_a^u is used.

4.5 Second phase

In this phase we run through the elements of $\mathcal{K}_u$. For each element $C_\ell^u \in \mathcal{K}_U$, let the elements $\mathcal{C}^1 := \{C_1^1, \ldots, C_i^1\} \subset \mathcal{K}_1$ and $\mathcal{C}^2 := \{C_1^2, \ldots, C_j^2\} \subset \mathcal{K}_2$ be in relation with C_ℓ^u according to r_1 and r_2 respectively. Let $\cup\mathcal{H}$ be the $\cup_{H\in\mathcal{H}} H$ for any set $\mathcal{H}$.

Depending on i and j, and the sizes of the communities involved, we create the relation $r(C_{i'}^1, C_{j'}^2)$ for every i' and j', and we assign community events to these relations.

- If $\mathcal{C}^1 = \emptyset$ and $|\mathcal{C}^2| > 0$, then $r(\emptyset, C_{j'}^2)$ is a *birth* event for every j'.
- If $|\mathcal{C}^1| > 0$ and $\mathcal{C}^2 = \emptyset$, then $r(C_{i'}^1, \emptyset)$ is a *death* event for every i'.
- If $|\mathcal{C}^1| = 1$ and $|\mathcal{C}^2| = 1$, that is $\mathcal{C}^1 = \{C_1^1\}$ and $\mathcal{C}^1 = \{C_1^2\}$, then
  - If $|C_1^1| = |C_1^2|$, $r(C_1^1, C_1^2)$ is an *exact match.*
  - If $|C_1^1| > |C_1^2|$, $r(C_1^1, C_1^2)$ is a *contraction* event.
  - If $|C_1^1| < |C_1^2|$, $r(C_1^1, C_1^2)$ is a *growth* event.

- If $|\mathcal{C}^1| = 1$ and $|\mathcal{C}^2| > 1$, then
  - If $|C_i^1| = |\cup \mathcal{C}^2|$, $r(C_i^1, C_{j'}^2)$ is a *split* event for every j'.
  - If $|C_i^1| < |\cup \mathcal{C}^2|$, $r(C_i^1, C_{j'}^2)$ is a *grow-split* event for every j'.
  - If $|C_i^1| > |\cup \mathcal{C}^2|$, $r(C_i^1, C_{j'}^2)$ is a *contraction-split* event for every j'.
- If $|\mathcal{C}^1| > 1$ and $|\mathcal{C}^2| = 1$, then
  - If $|\cup \mathcal{C}^1| = |C_j^2|$, $r(C_{i'}^1, C_j^2)$ is a *merge* event for every i'.
  - If $|\cup \mathcal{C}^1| < |C_j^2|$, $r(C_{i'}^1, C_j^2)$ is a *grow-merge* event for every i'.
  - If $|\cup \mathcal{C}^1| > |C_j^2|$, $r(C_{i'}^1, C_j^2)$ is a *contraction-merge* event for every i'.
- If $|\mathcal{C}^1| > 1$ and $|\mathcal{C}^2| > 1$ then we introduce an obscure event $r(C_{i'}^1, C_{j'}^2)$ for every i' and j'.

4.6 Time complexity

The time complexity of the algorithm will be addressed both from the theoretical and empirical point of view. In the first phase, we compare each community from $\mathcal{K}_1$ to communities from $\mathcal{K}_U$. In the worst case, no exact matches are found resulting in an $O(n * m)$ complexity, where n is the size of $|\mathcal{K}_1|$ and m is the size of $|\mathcal{K}_U|$. An exact match always reduces the number of further computations. Therefore when we find an exact match $r_1(C^1, C^u)$ ($r_2(C^2, C^u)$), we can ignore all other relations involving C^1, C^2 and C^u.

The obscure events are catchier. Note, that the sizes of $\mathcal{K}$'s might be exponential in $|V(G_1)|$, while a complex event could involve any number of communities on both sides, which would result in doubly-exponential running time. Fortunately, usually we have less communities than vertices, and the number of obscure events are small, consisting of only a few sets. So in practice the obscure events have only a negligible effect on the (actual) running time.

In the second phase, if each element from $\mathcal{K}_1$ and $\mathcal{K}_2$ were connected with every element from $\mathcal{K}_U$, we would obtain a worst case complexity of $O(n*m*k)$, $k = |\mathcal{K}_2|$. In practice, a community from $\mathcal{K}_U$ corresponds to only a few communities from $\mathcal{K}_1$ and $\mathcal{K}_2$. This reduces the actual running time of this phase to $O(c * m)$, where c is a small constant.

5 Results

To test our algorithm, besides the small examples like Zachary's graph, we have used two large test databases. One came from an international bank [8], while the other one is a large social network. The bank graph is based on a transaction database, and consists of roughly 80000 nodes and 270000 edges, and we were provided with three time instances taken in a six month period. The edges of the

Events	12u	23u	13u
Deaths	6586	6481	9901
Births	6729	6477	9971
Unchanged	4888	5062	2529
Growths	2185	2084	1629
Contractions	1985	2123	1569
Splits	83	91	45
Grow-splits	154	133	172
Cont-splits	144	149	111
Merges	200	184	117
Grow-merges	356	315	256
Cont-merges	350	326	418
Obscure	531	595	594

Table 3: Results on the bank database. The columns show the community events in the following order: first and the second time instances, the second and third time instances, finally the first and the third instances.

social graph were also determined as the output of a data mining project, and the graph contains roughly one million nodes and 1.5 million edges. Here four time instances were taken in a four month period. Due to the size and structure of these networks, the CFinder fails to provide communities, so the communities were listed by the N^{++} method only.

5.1 Observations

Table 3 shows the results for the bank dataset for different time intervals. Notice, that the number of death and birth events are very high, about 40% of the communities die. A possible explanations for this is, that the time lapse between the instances is relatively long (three months), and in the last case, where the first and last instances are compared, this is extended to six months. During this long time period, the community structure of a network changes significantly.

Another explanation might be based on problem (iii), referred in 2.1. If a dense community changes significantly, it is hard to know what had really happened to it, while the algorithm classifies it as a death event.

The number of unchanged, growing and contracting communities has the same magnitude, which dominates the splitting and merging events.[7]

It indicates a certain dynamic equilibrium that the number of growth and contract events, and the death and birth events are balanced. The number of pure split/merge events are less than the newly introduced grow/contraction split/merge

[7]Since the first types of events involve the exact match events described in 4.2, this explains the low running time.

Events	12u	23u	34u	14u
Deaths	6847	7812	7931	23519
Births	6112	4119	3782	14934
Unchanged	59985	59247	56031	38272
Growths	2999	2161	1963	4343
Contractions	3468	3629	3458	6753
Splits	457	454	383	538
Grow-splits	83	55	37	32
Cont-splits	115	83	102	266
Merges	919	820	785	1030
Grow-merges	232	180	178	438
Cont-merges	142	137	93	79
Obscure	90	92	59	66

Table 4: Results for the social network database. The columns show the community events in the following order: first and the second time instances, the second and third time instances, the third and fourth time instances, and finally the first and the fourth instances.

events. This justifies the introduction of these events, and underlines the complications in community dynamics.

The running time of our community matching algorithm was 8 seconds, while the search for communities took 5 minutes. The used computer configuration was a machine of two cores, with each processor running at 2.0 GHz and supplied with 2 gigabytes of memory.

Table 4 shows the matching results on the social network. In contrast to our previous results, the number of deaths and births are significantly lower, and the number of unchanged communities dominates all other events. The lapse between the time instances is short (one month), but in the last case, the first and last instances are compared resulting in a four months interval. Even in this case, the number of unchanged communities is much larger than the number of deaths, so one concludes that the social graph is more stable than the graph of the bank dataset. It is important to note, that the number of birth events are significantly lower than the number of death events, and the number of split-like events do not balance this by generating more communities. This indicates that the number of communities of the network is decreasing. Indeed, the network has lost around 12.5 percent of its communities in the observed period.

As in the previous case, the number of growth and death events are balanced. Here the number of pure split and merge events outnumber the newly introduced grow/contraction and split/merge events. This also points to the fact, that this network is more stable than the previous.

Even though this network is larger than the first, the running time of the matching algorithm is about the same due to the very high number of unchanged commu-

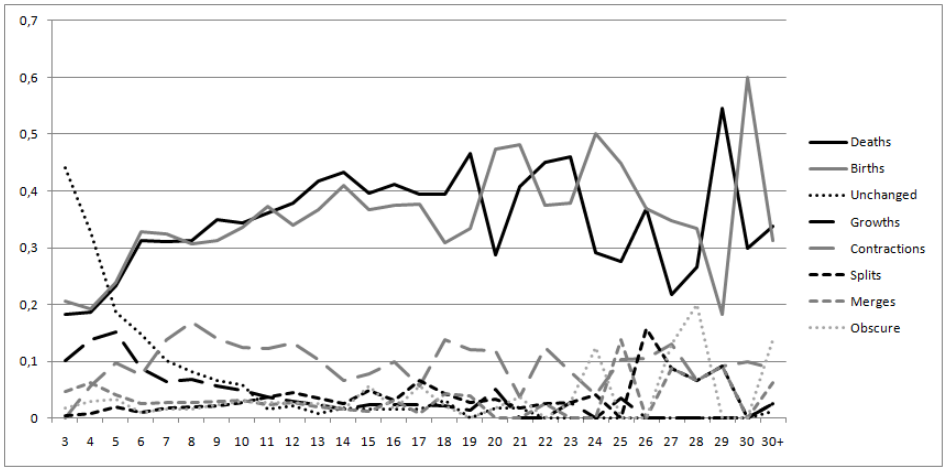

Figure 2: The percentage of community events compared to the community sizes for the bank dataset. The results were generated with the union graph approach.

nities. Note that the N^{++} needed about 60 seconds for finding the communities, our community matching algorithm took 14 seconds. We have used the same machine as before.

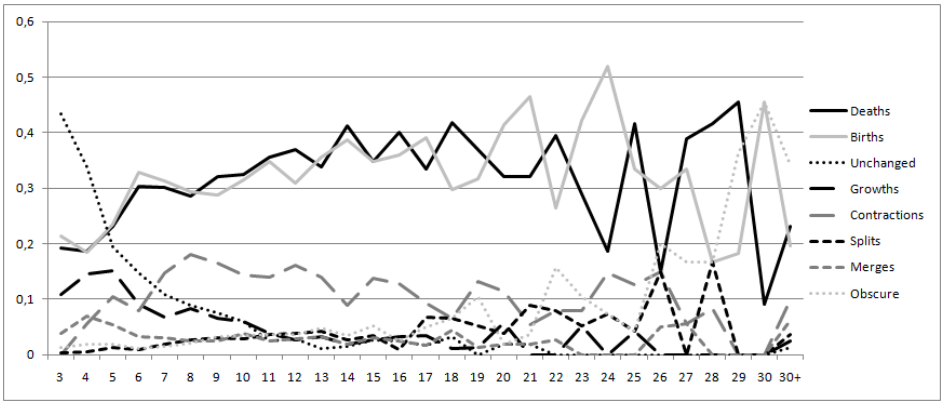

Figure 3: The percentage of community events compared to the community sizes for the bank dataset. The results were generated with the intersection graph approach.

5.2 Community evolution

Following the footsteps of Palla et al. [23], we compared the sizes and lifetimes of communities. Figure 2 displays our findings on the bank dataset. As we have concluded in the previous section, the community structure of this dataset changes very rapidly. For almost all sizes of communities the number of deaths outweigh all other community events. The only exceptions are very small communities of sizes three and four. It should also be noted, that there is a small spike of the obscure cases for the large communities. Aside from these, it can be said, that the size of a community does not relate to the community event it is involved in. Most importantly, the findings do not confirm the results of [23] that the expected lifetime of a community is a monotone function of its size. For a decisive result, further studies are needed.

If we take the intersection graph approach displayed on Figure 3, the results are almost the same, except for the largest communities. These cases are classified as obscure cases in contrast to declaring them dead as before.

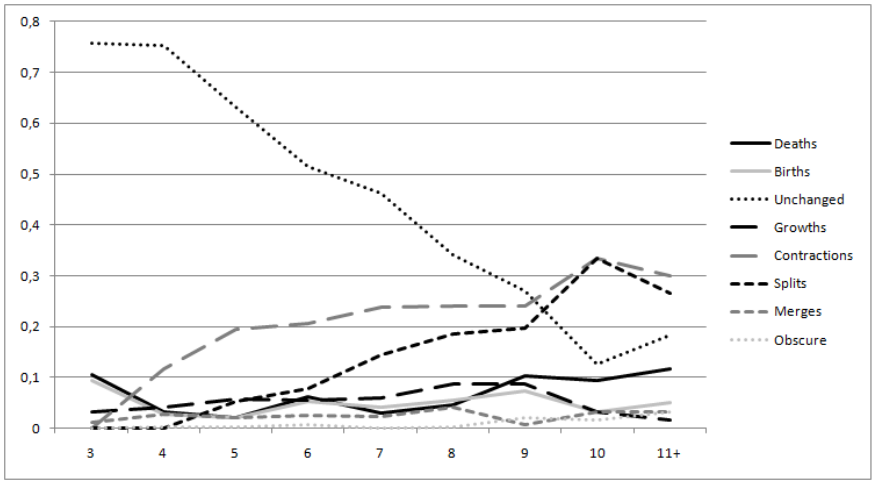

Figure 4: The percentage of community events compared to the community sizes for the social graph dataset with one month difference. The results were generated with the union graph approach.

The social network shows very different behavior. Figure 4 shows results for a one month time lapse. The percentage of death events is very small and constant compared to the community sizes, which is in contrast to the previous dataset. The number of unchanged communities is very high for small sizes, and decreases monotonically. This behavior was also reported in [23].

For large communities the contraction and split events dominate. The number of these events increases monotonically. The number of growth and merge events is small and independent of the sizes, which is somewhat surprising considering their

symmetrical counterparts. The number of all other events is small and independent of the community sizes. These results also indicate a more stable nature for this dataset.

Figure 5 shows the results for the four month interval. It is clear that in a four month interval, the community structure of a network can change dramatically, yet the difference between the results for the bank dataset and this one is clear. The number of deaths is significantly lower, but still relevant, and except for the smallest communities, independent of size. The number of birth events is lower than the number of death events, this matches with our observations in the previous chapter. The number of unchanged events is high for small communities, and it decreases somewhat faster, than in the previous network. The number of split events is increasing much faster than before, and contraction events are dominant even for medium sized communities. It should also be noted that small communities aside, the number of contraction events is independent from the community sizes. The number of all the other cases is small and constant.

The findings above reveal a strange behavior: while this network is more stable than the banking network, in some sense it is steadily loosing communities. The banking network on the other hand changes more dynamically, but it is in a state of equilibrium.

For the social network, the intersection method provides almost the same results, with the percentage of death events being slightly lower.

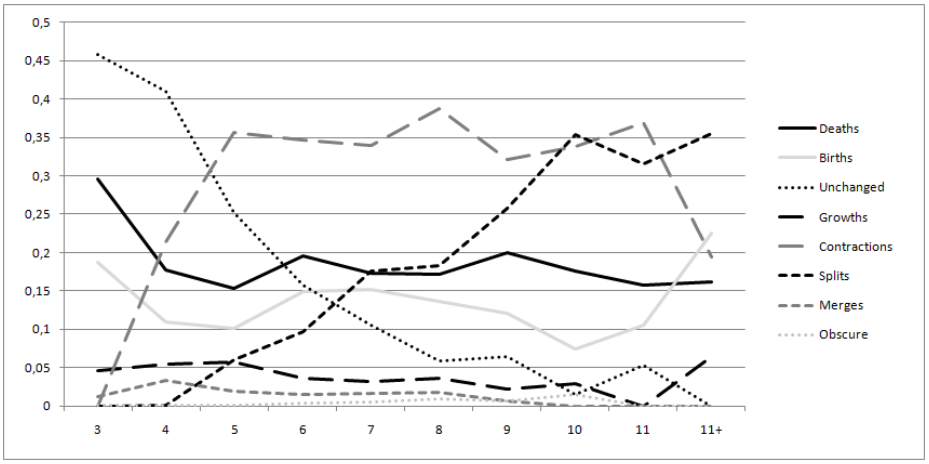

Figure 5: The percentage of community events compared to the community sizes for the social graph dataset with four months difference. The results were generated with the union graph approach.

6 Conclusion

Based on the earlier results of [23, 2] we have developed a new algorithm and methodology for following the life cycle of communities in dynamic graphs. The methodology successfully extends the incomplete notions used in [23], and results in an algorithm that works based on general community detection methods. The algorithm is very fast, it solves large community matching problems in seconds.

We have worked with two large datasets with fixed observation periods. Our findings indicate, that there is a significant difference between the behavior of the community structures and dynamics of these datasets. One of the networks is more stable, but looses communities steadily, the other network is more dynamical, but maintains its community number. We have also examined the changes in community structure in relation with the sizes of the involved communities. Our findings confirm some of the claims of [23], however not all of them.

Acknowledgment

The authors were supported by the Project named "TÁMOP-4.2.1/B-09/1/KONV-2010-0005 - Creating the Center of Excellence at the University of Szeged" also supported by the European Union and co-financed by the European Regional Fund.

The second author was partially supported by HSC-DAAD Hungarian-German Research Exchange Program (project P-MÖB/837) and Gyula Juhász Faculty of Education, University of Szeged (project CS-001/2010).

The third author was partially supported by the grant OTKA K76099 and by the grant **TÁMOP-4.2.2/08/1/2008-0008**.

References

[1] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, T. Vicsek CFinder: Locating cliques and overlapping modules in biological networks. *Bioinformatics* **22**, 1021–1023 (2006).

[2] S. Asur, S. Parthasarathy and D. Ucar, An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Transactions on Knowledge Discovery from Data* Volume **3**, Issue 4, November 2009.

[3] R. Albert, A. L. Barabási Statistical mechanics of complex networks. *Reviews of Modern Physics*, **74** 2002.

[4] B. Bollobás, *Modern Graph Theory,* Springer, New York (1998).

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebre, Fast unfolding of community hierarchies in large networks. arXiv (2008): 0803.0476.

[6] B. Bollobás and O. Riordan, Clique percolation. *Random Structures Algorithms* **35** (2009), no. 3, 294–322.

[7] D. Braha, Y. Bar-Yam, Time-Dependent Complex Networks: Dynamic Centrality, Dynamic Motifs, and Cycles of Social Interaction, Adaptive Networks, chapter 3, 2009.

[8] A. Csernenszky, Gy. Kovács, M. Krész, A. Pluhár and T. Tóth, Parameter optimization of infection models. *Proc. of the Challenges for Analysis of the Economy, the Business, and Social Progress International Scientific Conference,* Szeged, November 19–21, 617–623, 2009.

[9] L. Csizmadia, Recognizing communities in social graphs, MSc thesis, University of Szeged, 2003. (in Hungarian)

[10] L. Csizmadia, A. Bóta and A. Pluhár, Community detection and its use in Real Graphs. *Proceedings of the 13th International Multiconference INFORMATION SOCIETY - IS* 2010, 393-396.

[11] T. S. Evans and R. Lambiote, Edge Partitions and Overlapping Communities in Complex Networks. arXiv:0912.4389v1, 2009.

[12] S. Fortunato, Community Detection in graphs. Physics Reports Volume 486, Issues 3-5, February 2010, Pages 75-174.

[13] M. Granovetter, The Strength of Weak Ties. *American Journal of Sociology* **78(6)** 1360–1380, 1973.

[14] M. Granovetter, Threshold models of collective behavior. *American Journal of Sociology* **83(6)** 1420–1443, 1978.

[15] E. Griechisch and A. Pluhár, Community Detection by using the Extended Modularity. $(CS)^2$ - Conference of PhD Students in Computer Science, Szeged, 2010.

[16] J. Kleinberg, An Impossibility Theorem for Clustering. *Advances in Neural Information Processing Systems (NIPS)* **15**, 2002.

[17] J. M. Kumpula, M. Kivela, K. Kaski and J. Saramaki, A sequential algorithm for fast clique percolation. *Phys. Rev. E* **79**, 026109 (2008).

[18] S. Milgram, The Small World Problem. *Psychology Today*, **1**(1), 60–67, 1967.

[19] T. Népusz, A. Petróczi, L. Négyessy and F. Bazsó, Fuzzy communities and the concept of bridgeness in complex networks. *Phys. Rev. E* **77**, 016107 (2008).

[20] M.E.J. Newman, The structure and function of complex networks. *SIAM Rev.* **45**, 167–256, 2003.

[21] M. E. J. Newman, Detecting community structure in networks. *Eur. Phys. J. B* **38**, 321–330 (2004).

[22] G. Palla, I. Derényi, I. Farkas and T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435**, 814 (2005).

[23] G. Palla, A.-L. Barabási and T. Vicsek, Quantifying social group evolution. *Nature* **446**, 664–667 (2007).

[24] A. Pluhár On the clusters of social graphs based on telephone log-files. (in Hungarian) Research Report 2001.

[25] A. Pluhár Determination of communities in social graphs by fast algorithms. (in Hungarian) Research Report 2002.

[26] G. Thilo, S. Hiroki, (Eds.) *Adaptive Networks.* Springer Verlag, New York, 2009.

[27] W. W. Zachary, An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* **33**, 452–473 (1977).