Discovering dynamic communities in interaction networks

Polina Rozenshtein, Nikolaj Tatti, and Aristides Gionis

Helsinki Institute for Information Technology, Aalto University
firstname.lastname@aalto.fi

Abstract. Online social networks are often defined by considering interactions over large time intervals, e.g., consider pairs of individuals who have called each other at least once in a mobilie-operator network, or users who have made a conversation in a social-media site. Although such a definition can be valuable in many graph-mining tasks, it suffers from a severe limitation: it neglects the precise time that the interaction between network nodes occurs.

In this paper we study *interaction networks*, where one considers not only the social-network topology, but also the exact time that nodes interact. In an interaction network an edge is associated with a time stamp, and multiple edges may occur for the same pair of nodes. Consequently, interaction networks offer a more fine-grained representation that can be used to reveal otherwise hidden dynamic phenomena in the network.

We consider the problem of discovering communities in interaction networks, which are dense and whose edges occur in short time intervals. Such communities represent groups of individuals who interact with each other in some specific time instances, for example, a group of employees who work on a project and whose interaction intensifies before certain project milestones. We prove that the problem we define is **NP**-hard, and we provide effective algorithms by adapting techniques used to find dense subgraphs. We perform extensive evaluation of the proposed methods on synthetic and real datasets, which demonstrates the validity of our concepts and the good performance of our algorithms.

Keywords: community detection, graph mining, social-network analysis, dynamic graphs, time-evolving networks, interaction networks

1 Introduction

Searching for communities in social networks is one of the most well-studied problems in social-network analysis. A number of different methods has been proposed, employing a diverse set of algorithmic tools, such as, agglomerative approaches, min-cut formulations, random walks, spectral methods, and more. Somewhat in contrast to this line of work, it has been observed that large networks are characterized by the lack of clear and well-defined communities [13,21].

The lack of well-defined communities can be contributed to the high degree of interconnectivity, and the existence of overlapping communities. The phenomenon of overlapping communities is aggravated by the fact that community-detection methods typically ignore the time of interaction between network nodes, for instance, the same type of link can be used to represent friends in a hobby club and work colleagues.

On the other hand, as the amount of available data increases in volume and richness, it becomes possible to analyze not only the underlying topology but also the exact time of interactions. Analysis of such interaction events can reveal much more information about the structure and dynamics of the communities in the network. To be more concrete, consider the following examples.

Example 1: A group of researchers across many different European institutions are working on a large project. The members of the group go along with their everyday lives and other tasks, often unrelated to the project. However, once every few weeks or months, before deadlines of deliverables or project meetings, there is a lot of interaction among the group members.

Example 2: A group of twitter users is interested in technology products, in particular smartphones, and they are very active in blogging reviews and commenting the posts of each other. Their interaction is sparse, but it sustains over a long time, and it intensifies significantly after the release of a new product.

The main point of these two examples is that the communities are *not* isolated. Their members interact with each other, but they also interact with others outside the community. If one ignores the interaction dynamics and considers only the static social-network topology, the communities will be hidden and it will be impossible to discover them. It is only when considering the interaction time instances among the community members that it becomes possible to identify them: in both of the above examples, many interactions occur among the community members, but in a number of relatively short time intervals.

In this paper we formalize the idea exemplified above. We consider *interaction networks* for which we assume that all interaction events between the network nodes are known. Examples of such interaction networks include *call graphs* of telecommunications, *email communication networks*, *mention and commenting networks* in social media, *collaboration networks*, and more. Thus, interaction-network datasets are already abundant in many application domains.

In the context of interaction networks, we study the problem of discovering communities that are dense and whose edges occur in short time intervals. We prove that the problem we define is **NP**-hard, even though that the corresponding problem on static graphs is polynomially-time solvable. For the problem we define, we provide algorithms inspired by the literature of finding dense subgraphs. Our experiments demonstrate the effectiveness of the proposed algorithms, as well as the validity of our hypothesis. Namely, that it is possible to find communities that satisfy the requirements we set: dense interactions that occur within a number of short time intervals.

2 Preliminaries and notation

An *interaction network* $G = (V, E)$ consists of a set of n nodes V and a set of m time-stamped interactions E between pairs of nodes

$$E = \{(u_i, v_i, t_i)\}, \text{ with } i = 1, \ldots, m, \text{ such that } u_i, v_i \in V \text{ and } t_i \in \mathbb{R}.$$

We consider that interactions are *undirected*. More than one interaction may take place between a pair of nodes, with different time stamps. Conversely, more than one interaction may take place at the same time, between different nodes.

For an interaction network $G = (V, E)$ we associate the set of edges $\pi(E)$ to be the pairs of nodes for which there is at least one interaction (one may think of π as "projecting" the edges of the interaction network along the time axis)

$$\pi(E) = \{(u, v) \in V \times V \mid (u, v, t) \in E \text{ for some } t\}.$$

Given an interaction network $G = (V, E)$, the network $\pi(G) = (V, \pi(E))$ is a standard graph with no time stamps on its edges. We refer to $\pi(G)$ as the *topology network* of G or as the *underlying network* of G.

Given an interaction network $G = (V, E)$ and a subset of nodes $W \subseteq V$, we define the *induced interaction network* $G(W) = (W, E(W))$, such that $E(W)$ consists of the interactions whose both end-points are contained in W,

$$E(W) = \{(u, v, t) \in E \mid u, v \in W\}.$$

We also consider time intervals $[s, f]$, where $s \in \mathbb{R}$ is the start point and $f \in \mathbb{R}$ is the end-point of the interval. We define the *span* of an interval to be its time duration, i.e., $\text{span}(T) = f - s$.

We define a *time-interval set* $\mathcal{T}$ to be a collection of non-overlapping time intervals, $\mathcal{T} = (T_1, \ldots, T_k)$. The span of $\mathcal{T}$ is the sum of individual spans,

$$\text{span}(\mathcal{T}) = \sum_{i=1}^{k} \text{span}(T_i).$$

Given an interaction network $G = (V, E)$ and a time interval $T = [s, f]$ we define the *spliced interaction network* $G(T) = (V, E(T))$, where $E(T)$ are the interactions that occur in T,

$$E(T) = \{(u, v, t) \in E \mid s \leq t \leq f\}.$$

The above notion can be extended in a straightforward manner, so as to define the spliced interaction network with respect to a set of time intervals $\mathcal{T} = (T_1, \ldots, T_k)$. This is achieved by collecting edges from individual time intervals, that is, $G(\mathcal{T}) = (V, E(\mathcal{T}))$, where $E(\mathcal{T}) = \bigcup_{i=1}^{k} E(T_i)$.

The concepts of *induced interaction network* and *spliced interaction network* provide two different ways to select subsets of interaction networks; one is based on subsets of nodes and the other is based on time intervals. The definition

of dynamic communities, which is the central concept of our paper, relies on these two subset-selection strategies. In particular, for an interaction network $G = (V, E)$, a subset of nodes W, and a set of time intervals $\mathcal{T}$, we define a dynamic community $G(W, \mathcal{T})$ as the subgraph that consists of the nodes in W and the set of interactions among the nodes in W that occur within $\mathcal{T}$. In more formal terms, $G(W, \mathcal{T})$ is defined to be the spliced interaction network $H(\mathcal{T})$, where H is the induced interaction network $G(W)$.

To measure the quality of a dynamic community we rely on the notion of *density*. We recall the definition of density as defined for static graphs, e.g., for the topology network $\pi(G) = (V, \pi(E))$ of an interaction network $G = (V, E)$. We also review the *densest-subgraph problem* for static graphs.

Given a static graph $H = (V, F)$, i.e., the edges F do not have time stamps, the *density* of the graph $d(H)$ is twice the ratio of edges and the vertices,

$$d(H) = \frac{2\,|F|}{|V|}.$$

Problem 1 (Densest subgraph). Given a static graph $H = (V, F)$, find a subset of vertices W that maximizes the density $d(H(W))$.

Unlike the problem of finding the largest clique, which is **NP**-hard, finding the densest subgraph is polynomially-time solvable. Furthermore, there is a linear-time factor-2 approximation algorithm [2,8]. The algorithm deletes iteratively a vertex with the lowest degree, obtaining a sequence of subgraphs. Among those subgraphs the algorithm returns the one with the highest density.

3 Dense communities in interaction networks

Given an interaction network $G = (V, E)$ we aim to find a set of nodes W and a set of time intervals $\mathcal{T}$, such that the subgraph $G(W)$ is relatively dense within $\mathcal{T}$. To ensure that the time span of the subgraph $G(W)$ is short, we impose two types of constraints on the time-interval set $\mathcal{T}$: (i) constraints on the number of intervals of $\mathcal{T}$, and (ii) constraints on the total length of $\mathcal{T}$. We discuss these two constraints shortly. For the problem of finding dense dynamic communities, which we provide below, we also assume a *quality score* $q(W, \mathcal{T}; G)$ that measures the density of the community $G(W, \mathcal{T})$ in the interaction network G.

Problem 2. Assume that we are given a *quality score* $q(W, \mathcal{T}; G)$ that measures the quality of the community defined by nodes W and time interval span $\mathcal{T}$ in the interaction network G. Assume also we are given a budget K on the number of time intervals, and a budget B on the total time span. Our goal is to find a set of nodes W and a set of time intervals $\mathcal{T}$ that maximize

$$q(W, \mathcal{T}; G)\,, \text{ such that } |\mathcal{T}| \leq K \text{ and } \text{span}(T) \leq B.$$

The first constraint states that we can have at most K intervals while the second constraint requires that the total duration is at most B. Both constraints are

required: assuming that the quality score increases with the time span, if we drop the second constraint, then we can always choose the whole time span. Such a solution, however, does not capture the intuition of dynamic communities that we aim to discover. On the other hand, if we drop the first constraint, then we can pick individual edges by setting a time interval of duration 0 around each individual edge. Namely, the constraint on the number of intervals is necessary to impose time-continuity on the solutions found.

Regarding the score function used to assess the quality of a community, our proposed measure is the density of the topology network, after restricting to node set W and time-interval set $\mathcal{T}$

$$q(W, \mathcal{T}; G) = d(\pi(G(W, \mathcal{T}))) \,,$$

that is, we count twice the number of interactions that occur between nodes of W within time intervals $\mathcal{T}$, and divide this number by $|W|$.

3.1 Complexity

We proceed to establish the complexity of the problem of finding a dense dynamic community in interaction networks (Problem 2).

Proposition 1. *The decision version of Problem 2 is **NP**-complete.*

Proof. We are given an interaction network G, budgets K, B, and a threshold σ, and we need to answer whether there is a node set W and a time-interval set $\mathcal{T}$, which satisfy the two budget constraints, and for which $q(W, \mathcal{T}; G) \geq \sigma$.

The problem is clearly in **NP**. To prove the hardness, we obtain a reduction from VertexCover. An instance of VertexCover specifies a graph H and budget ℓ, and asks whether there is a set $V' \subseteq V$, such that $|V'| \leq \ell$ and each edge of the graph is adjacent to at least one of the nodes of V'.

Consider graph $H = (U, F)$ with n nodes and m edges, and budget ℓ. Let us define an interaction network $G = (V, E)$. The node set V consists of U and $n+1$ additional auxiliary nodes, and the set of edges E is defined as follows: First we consider $n + 1$ distinct time points $t_0, \ldots, t_n$. At t_0 we consider interactions between all the auxiliary nodes, and between auxiliary nodes and each $v \in U$. We arbitrarily order the nodes in U and let v_i be the i-th node. At time t_i we connect v_i with all its neighbors in H.

Assume that there exists a solution W and $\mathcal{T}$, for Problem 2, with budgets $K = \ell + 1$ and $B = 0$. We claim that W will contain all nodes and $\mathcal{T}$ will contain t_0 and the time points corresponding to the vertex cover of H.

Let us first prove that $W = V$ and $(t_0, t_0) \in \mathcal{T}$. Assume otherwise. Then, since the remaining time intervals have only edges between U, there must be at most $n(n-1)/2$ edges, yielding density at most $n - 1$. Let us replace one of the selected time intervals with t_0 and reset W to be auxiliary nodes. This solution gives us a density of n, which is a contradiction.

Now we have established that t_0 is a part of $\mathcal{T}$. A straightforward calculation shows that it is always beneficial to add auxiliary nodes to W, if they are not

part of a solution. Once this is shown, we can show further that adding any missing nodes from U also improves the density. Consequently, $W = V$.

Set $\sigma = 2(n(n+1)/2 + n(n+1) + m)/(2n+1)$. The first two terms in the numerator correspond to the edges at t_0. The remaining m edges must come from the remaining time intervals. This is only possible if and only if the time intervals contain all edges from H, that is, the corresponding nodes cover every edge, which completes the reduction. □

4 Algorithms for discovering communities

In this section we present the algorithm we propose for Problem 2. Since the problem is **NP**-hard, we propose an iterative method, which improves the solution by optimizing each one of the two components, the node set W and the time-interval set $\mathcal{T}$, in an alternating fashion, while keeping the other fixed.

Both of the objectives of our alternating optimization method give rise to interesting computational problems. One problem reduces to finding the densest subgraph, and the other is related to coverage, and it is even **NP**-hard. Next we formalize the two problems of our alternating optimization method.

Problem 3. Consider an interactive network $G = (V, E)$. Consider the problem of finding a dense dynamic community, with budgets K and B, and quality score q. Assume that a set of nodes W is provided as input. Find a time-interval set $\mathcal{T}$ that maximizes

$$q(W, \mathcal{T}; G)\,, \text{ such that } |\mathcal{T}| \leq K \text{ and } \text{span}(T) \leq B.$$

Problem 4. Consider the problem of finding a dense dynamic community on an interactive network $G = (V, E)$ with quality score q. Assume that a time-interval set $\mathcal{T}$ is given as input. Find a set of nodes W that maximizes $q(W, \mathcal{T}; G)$.

The proposed algorithm starts from an initial time interval set $\mathcal{T}_0$, and obtains a solution $(W, \mathcal{T})$ by iteratively solving the two problems defined above until convergence. Pseudocode of the method is given in Algorithm 1. As one may expect the iterative algorithm does not provide a guarantee for the quality of the solution that it returns. However, as it is stated by the following proposition, whose proof is straightforward, it has the desirable property that both of the alternating optimization problems return the correct component of the solution if they obtain as input the other component correctly.

Proposition 2. *Let $(W, \mathcal{T})$ be a solution to Problem 2 for a given interaction network G. Then (i) $\mathcal{T}$ is a solution to Problem 3 given G and W, and (ii) W is a solution to Problem 4 given G and $\mathcal{T}$.*

In the next two sections, 4.1 and 4.2, we present in detail our solution for the two subproblems of the iterative algorithm. In Section 4.3 we discuss the initialization of the algorithm.

Algorithm 1: Iterative algorithm for finding a dense dynamic community

```
T_0 ← initial sets of time intervals;
i ← 0;
while (convergence; i++) do
    W_{i+1} ← solution to Problem 4 given T_i;
    T_{i+1} ← solution to Problem 3 given W_{i+1};
return (W_i, T_i);
```

4.1 Finding an optimal set of nodes

We start with Problem 4 where the goal is to find an optimal set of nodes W given a set of time intervals $\mathcal{T}$. Assume that we are given a set of time intervals $\mathcal{T}$ and let $H = \pi(G(\mathcal{T}))$ be the topology network for the interactions that occur within $\mathcal{T}$ (i.e., the topology network of the interaction network spliced by $\mathcal{T}$). Note that

$$q(W, \mathcal{T}; G) = d(H(W)) \,.$$

Consequently, finding the optimal set of nodes is equivalent to the densest-subgraph problem (Problem 1) on the (static) graph H. It follows that finding the optimal set of nodes W, given time-interval set $\mathcal{T}$, can be done in polynomial time. In our implementation, we use the linear-time algorithm of Charikar [8], which, as outlined in Section 2, offers a factor-2 approximation guarantee.

4.2 Finding an optimal set of time intervals

We now present our solutions for the second subproblem of the iterative algorithm, namely, finding an optimal set of time intervals for a given set of nodes. Unfortunately, even if this is a subproblem of the general community-discovery problem, it remains **NP**-hard. The proof of this claim is a simplified version of the proof of Proposition 1.

We view the problem of finding optimal time intervals as an instance of a *maximum-coverage with multiple budgets* (MCMB) problem.

*Problem 5 (*MCMB). Given a ground set $U = \{u_1, \ldots, u_m\}$ with weighted elements $w(u_i)$, a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_k\}$, p cost functions c_i mapping each subset of $\mathcal{S}$ to a positive number, and n budget parameters B_i, find a subset $\mathcal{P} \subseteq \mathcal{S}$ maximizing

$$\sum_{u \in X} w(u), \text{ such that } X = \bigcup_{S \in \mathcal{P}} S, \text{ and } \sum_{S \in \mathcal{P}} c_i(S) \leq B_i, \text{ for all } i = 1, \ldots, p.$$

When $p = 1$, the problem is the standard budgeted maximum coverage. The problem is still **NP**-hard but there exists an approximation algorithm by Khuller et al. that achieves $(1 - 1/e)$ approximation ratio [17]. However this algorithm requires to enumerate all 3-subset collections, making it infeasible in practice.

The optimization problem can be also viewed as an instance of maximizing submodular function under multiple linear constraints. Kulik et al. presented a polynomial algorithm that achieves $(1-\epsilon)(1-1/e)$ approximation ratio [18]. Unfortunately, this algorithm is not practical even for modest ϵ.

To see how finding a set of time intervals is related to maximum coverage, consider as ground set the set of edges $\pi(E(\mathcal{T}))$ (interactions that occur in $\mathcal{T}$ without the time stamps), and for each time interval $T \in \mathcal{T}$ create a subset S_T containing all edges whose corresponding interactions occur in T. There are two cost functions $c_1(T) = 1$ and $c_2(T) = \text{span}(T)$. The first budget constraint enforces the number of allowed time intervals to stay below K, while the second budget enforces the time-span constraint.

Thus, we need to solve the MCMB problem, defined above, with two budget constraints. We propose two solutions, both of which are inspired by the standard greedy approach for maximum coverage. The difference between the two proposed approaches is on how they try to satisfy the budget constraints. The first approach incorporates both budget constraints on the greedy step, while the second approach sets a parameter that controls the amount of violation of one constraint, and optimizes this parameter with binary search.

The standard greedy approach for maximum coverage is to select the set that has the best ratio of newly covered elements with respect to its cost. Motivated by this idea, we suggest the following greedy approach. Given a currently selected set of time intervals, say $\mathcal{T}$, we find the interval R that has the best ratio

$$\frac{q(W, \mathcal{T} \cup R, G) - q(W, \mathcal{T}, G)}{\max(x, y)}, \text{ where } x = \frac{1}{K - |\mathcal{T}|} \text{ and } y = \frac{\text{span}(R)}{B - \text{span}(\mathcal{T})}.$$

The numerator in the ratio is the number of new edges that can be covered with the new interval R. The denumerator is the maximum of two quantities, x and y, representing the two constraints on number of time intervals and time span, respectively. Both x and y are normalized so that they are equal to 1 if adding R will cap the corresponding constraint. By taking the maximum of the ratios we consider the constraint that is closer to be capped and penalize the ratio accordingly. The algorithm stops when one of the two constraints gets violated. We will refer to this approach as GREEDY.

Our second approach is based on the following observation. Assume that we are given a number α and consider optimizing

$$q(W, \mathcal{T}, G) - \alpha \cdot \text{span}(\mathcal{T}), \text{ such that } |\mathcal{T}| \leq K. \tag{1}$$

Note that we do not enforce any budget on the time span. If we set $\alpha = 0$, then the solution will contain the whole time. On the other hand, if we set α to be large, $\mathcal{T}$ will be just singular points. In fact, as it is shown in the following proposition, the time span of the optimal solution decreases as α increases.

Proposition 3. *Consider α_1 and α_2 with $\alpha_1 < \alpha_2$. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the solutions of Equation (1) for α_1 and α_2, respectively. Then* $\text{span}(\mathcal{T}_1) \geq \text{span}(\mathcal{T}_2)$.

Proof. Define $\beta_i = \text{span}(\mathcal{T}_i)$ and $d_i = q(W, \mathcal{T}_i, G)$. Due to optimality, we have

$$d_1 - \alpha_1\beta_1 \geq d_2 - \alpha_1\beta_2 \text{ and } d_2 - \alpha_2\beta_2 \geq d_1 - \alpha_2\beta_1.$$

By rearranging the terms we obtain $\alpha_1\beta_2 - \alpha_1\beta_1 \geq d_2 - d_1 \geq \alpha_2\beta_2 - \alpha_2\beta_1$. Rearranging the left and the right side gives us $(\alpha_1 - \alpha_2)(\beta_2 - \beta_1) \geq 0$. Since $\alpha_1 < \alpha_2$, we must have $\beta_1 \geq \beta_2$, which proves the proposition. □

Ideally, if we can solve Equation (1) optimally, we can use binary search to find the smallest α such that the time span of the solution does not exceed the budget. As we do not have an exact solver for Equation (1), we apply a greedy approach where in each step we find a single time interval that maximizes the score function. We then apply a binary search to find α that produces a feasible solution. We refer to this algorithm as BINARY.

4.3 Initialization

The quality of the solution discovered by the iterative algorithm depends on the set of time intervals $\mathcal{T}_0$ used as initial seed. Consider an optimal solution $(W, \mathcal{T})$, with $\mathcal{T} = (T_1, \ldots, T_K)$, which achieves density d^*. It follows that there is one single time interval $T \in \mathcal{T}$, for which the optimal set of nodes W has density at least d^*/K on $\pi(G(T))$. This observation motivates us to limit ourselves to consider only time interval sets of size 1. Assuming large computational power, one could test every possible time interval as a seed, consequently run the iterative algorithm, and return the best solution found out of all runs. There are $\mathcal{O}(m^2)$ such intervals, which is polynomial.

When running the algorithm $\mathcal{O}(m^2)$ times is expensive, we can select J random intervals, run the iterative algorithm for each of those random intervals, and return the best solution found out of all runs. In our experiments we evaluate the effect of the number of random seeds J to the quality of the solution found.

5 Experimental evaluation

To evaluate the proposed methods we use several datasets: synthetic and real-world social communication networks. We describe our datasets in detail below.

Synthetic data. We simulate activity on a network with a planted community. Different parameters for the planted community and the background noise are used, and the objective is to measure how the algorithms behave with respect to those parameters. The background network G is an Erdős-Rényi random graph, with expected degree being one of the model parameters. We plant a dense subgraph G', whose expected degree is a second model parameter. The length of whole time interval T is $|T| = 1000$ time units, while the interactions for the edges of G' can be covered by $k = 3$ arbitrary planted time intervals with total length of $|T'| = 100$ time units (10 times shorter than $|T|$). We randomly assign edges of G to time instances in T and edges of G' to time instances in T'.

Table 1: Characteristics of the two families of synthetic datasets. Planted community in Synthetic1 is a 5-clique. Planted community in Synthetic2 is an 8-node subgraph.

Name	$\|V\|$	Exp[$\|E\|$]	community avg degree	background avg degree
Synthetic1	100	200	4	1 – 6
Synthetic2	100	200	2 – 7	4

Table 2: Basic characteristics of real-world datasets. $|V|$: number of nodes; $|\pi(E)|$: number of edges of the topology network; $|E|$: number of interactions; $d(\pi(G))$: density of the whole topology network; $d^*(\pi(G))$: density of densest subgraph of the topology network.

Name	$\|V\|$	$\|\pi(E)\|$	$\|E\|$	$d(\pi(G))$	$d^*(\pi(G))$
Tumblr	1980	2454	7645	2.479	7.0
Students	883	2246	9865	5.087	11.292
Enron	1143	2019	6245	3.533	14.387

We test the ability of our algorithms to discover the planted communities in two settings. In the first setting (dataset family Synthetic1) we fix the planted subgraph and we vary the average degree of the background network. The objective is to test the robustness against background noise. In the second setting (dataset family Synthetic2) we fix the average degree of the background network and we vary the density of the planted subgraph. The characteristics of the synthetic datasets are given in Table 1.

Real-world data. We use three datasets. The characteristics of these datasets are summarized in Table 2.

Tumblr: This is a subset of that Memetracker dataset,[1] which contains only quoting between Tumblr users. The subset covers three months: 02.2009–04.2009.

Students:[2] This dataset logs the activity in a student online community at University of California, Irvine. Nodes represent students and edges represent messages with ignored directions. We used a subset of the dataset that covers four months of communication from 2004-06-28 to 2004-10-26.

Enron:[3] This is the well-known dataset that contains the email communication of the senior management in a large company. It spans over 20 years from 1980.

Discovering hidden structure. We test the ability of our algorithms to detect the planted communities for different levels of background and in-community average degrees. We quantify the quality of our algorithms by measuring *precision* and *recall*, with respect to the ground-truth communities. We also report the

[1] `http://snap.stanford.edu/data/memetracker9.html`
[2] `http://toreopsahl.com/datasets/#online_social_network`
[3] `http://www.cs.cmu.edu/~./enron/`

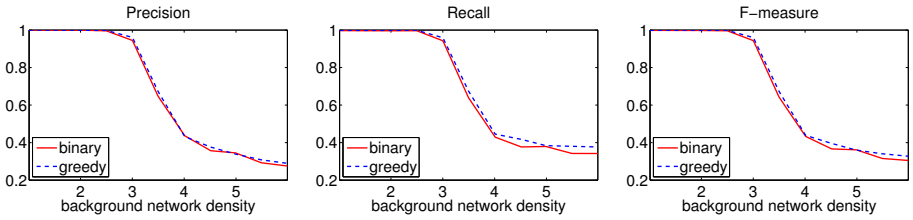


Fig. 1: Precision, recall and F-measure on Synthetic1, as a function of the background-network density. The planted community is a 5-clique.

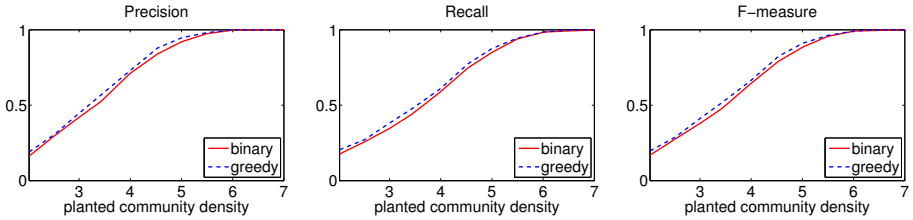


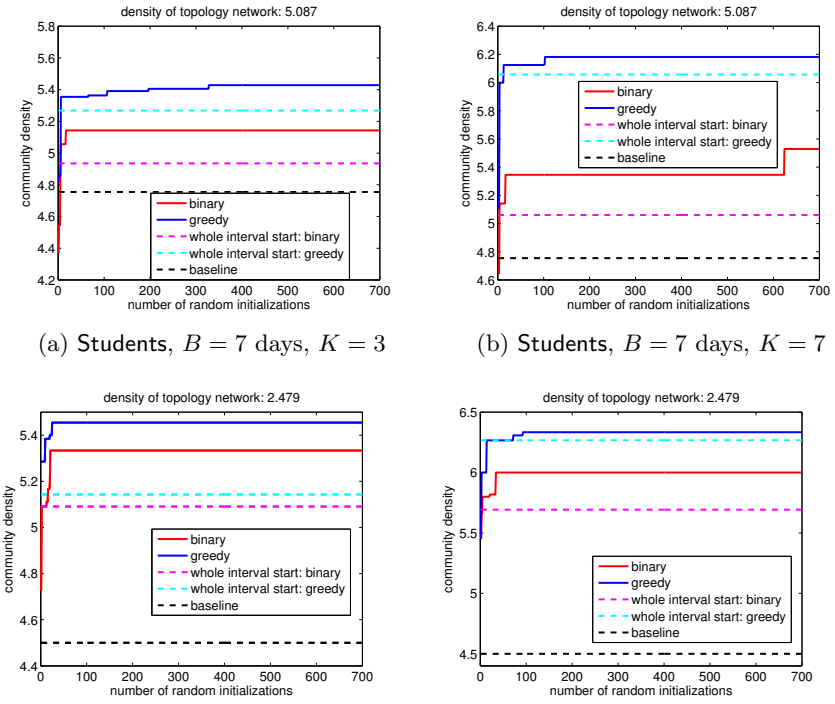
Fig. 2: Precision, recall and F-measure on Synthetic2, as a function of the density of a planted community of 8 nodes. The background-network density is set to 4.

F-measure, the harmonic mean of precision and recall. Results reported below are averages over $J = 1000$ independent runs.

Precision, recall and F-measure results for the two families of synthetic datasets are shown in Figures 1 and 2, respectively. Recall that datasets Synthetic1, contain a community based on a 5-clique. Both algorithms are able to discover this community correctly when the average degree of the underlying graph is smaller then the average degree of the planted community. Even when the community density is equal to the background-network density (around 4), the algorithms tend to keep high precision and recall. Precision and recall regrade at the same rate, indicating that with increase of background-network density the algorithms retrieve less nodes of planted community and more noisy nodes. Nevertheless, the measures do not drop very low, implying that the 5-clique spread over $k = 3$ short time intervals is distinguishable even within a dense background network.

The results on the second family of datasets (Synthetic2), shown in Figure 2, are similar. Both algorithms perform well when the background-network density is smaller than the planted-community density.

Effect of random seeds. Both of our algorithms are instances of Algorithm 1. In the experiments shown above we initialize the interval seed $\mathcal{T}_0$ with the whole time interval T spanned by the dataset. Starting from $\mathcal{T}_0 = \{T\}$ ensures that the subgraph we discover belongs to some dense structure in the topology network. However, if such a dense structure occurs in a scattered manner, the initialization



(a) Students, $B = 7$ days, $K = 3$ (b) Students, $B = 7$ days, $K = 7$

(c) Tumblr, $B = 7$ days, $K = 3$ (d) Tumblr, $B = 7$ days, $K = 7$

Fig. 3: Effect of random initializations on real-world datasets.

$\mathcal{T}_0 = \{T\}$ may mislead. To overcome this problem and avoid dense structures that cannot be covered in the given time budget, we initialize Algorithm 1 with many random time intervals, and return the best solution found.

The improvement of performing random initializations is shown in Figure 3. The experiments are shown for Tumblr and Students. The figures show the best density discovered by our algorithms, when J independent random runs are performed. As expected, random initializations improve the performance of the algorithms. The most significant improvement is obtained for the Student dataset. We also experiment with a baseline that finds the densest subgraph over *all* possible intervals that satisfy the time budget B (no iterative process is followed). We see that our algorithms perform significantly better than this baseline.

Discovered communities. Table 3 reports the densities of the communities discovered by our algorithms in the real-world datasets. We use $J = 200$ random initializations. We compare our algorithms with the same baseline as before: the densest subgraph over all intervals that satisfy the time budget B.

Overall, we observe that Greedy and Binary perform equally well, while in some settings Binary yields denser communities than Greedy.

Table 3: Densities of discovered subgraphs. The second column contains the number of allowed sets K and the column "budget" contains the time span budget B. For Tumblr and Students, B_1, B_2 and B_3 are equal to 1, 3 and 7 days, respectively. For Enron, B_1, B_2 and B_3 are 10, 30 and 120 days, respectively.

		budget = B_1			budget = B_2			budget = B_3		
Name	K	Binary	Greedy	Base	Binary	Greedy	Base	Binary	Greedy	Base
Tumblr	1	3.818	3.818	3.866	4.0	4.0	4.0	4.5	4.5	4.5
	2	4.0	4.0	3.866	4.571	4.571	4.0	5.111	5.111	4.5
	3	4.6	4.285	3.866	5.2	5.2	4.0	5.5	5.4	4.5
	4	4.909	4.8	3.866	5.384	5.25	4.0	5.857	5.666	4.5
	5	5.166	5.111	3.866	5.666	5.5	4.0	6.0	5.866	4.5
	7	5.5	5.333	3.866	6.0	5.714	4.0	6.333	6.333	4.5
	10	6.181	5.818	3.866	6.428	6.181	4.0	6.8	6.666	4.5
Students	1	2.947	3.384	3.428	3.76	3.764	3.84	4.545	4.647	4.755
	2	3.5	3.3	3.428	4.32	4.133	3.84	5.225	5.125	4.755
	3	4.2	3.846	3.428	4.384	4.444	3.84	5.304	5.312	4.755
	4	4.0	4.0	3.428	4.545	4.615	3.84	5.642	5.368	4.755
	5	4.363	4.363	3.428	4.933	4.941	3.84	5.939	5.642	4.755
	7	4.625	4.545	3.428	5.210	5.185	3.84	6.108	6.0	4.755
	10	4.956	4.888	3.428	5.666	5.485	3.84	6.5	6.307	4.755
Enron	1	6.7272	6.7272	6.727	8.8	8.8	8.8	11.909	11.909	11.9
	2	8.875	8.4705	6.727	9.2222	9.625	8.8	13.047	11.913	11.9
	3	10.470	10.0	6.727	10.555	11.176	8.8	13.307	12.8	11.9
	4	11.058	10.736	6.727	11.904	12.2	8.8	13.642	13.047	11.9
	5	11.473	11.4	6.727	12.25	12.16	8.8	13.714	13.238	11.90
	7	12.370	12.16	6.727	12.666	13.0	8.8	13.931	13.857	11.9
	10	13.285	13.185	6.727	13.357	13.571	8.8	14.074	14.0	11.9

For fixed value of the time budget B, the density of the discovered community increases with K. For small values of K (1 to 3), the density of the communities discovered by our algorithm is equal, or in some cases slightly smaller, than the density of the communities discovered by the baseline. This behavior is expected, as the brute-force baseline tests all possible intervals, while our algorithms use only some random intervals for initialization. However, as the value of K increases, the algorithms take advantage of the provided flexibility to use many intervals effectively; for $K > 3$ both algorithms always outperform the baseline.

Furthermore, as we can see by contrasting Tables 2 and 3, the discovered communities are almost as dense as the densest subgraphs on the whole topology network, even though the time budget is significantly smaller than the time span of the dataset. For example, the densest subgraph of the over 20-year-large Enron dataset has average degree 14.387, while we were able to discover a subgraph with average degree 13.285 in a budget of 10 days, spanning 10 time intervals.

6 Related work

Community detection is one of the most studied problems in social-network analysis. A lot of research has been devoted to the case of static graphs, and the typical setting is to partitioning a graph into disjoint communities [9, 12, 26, 30]; a thorough survey on such methods has been compiled by Fortunato [10].

Typically the term "dynamic graphs" refers to the model where edges are added or deleted. In this setting, once an edge is inserted in the graph it stays "alive" until the current time or until it is deleted. For example, this setting is used to model the process in which individuals establish friendship connections in a social network. On the contrary, our model intends to capture the continuous interaction between individuals. In the dynamic-graph setting, researchers have studied how networks evolve with respect to the arrival of new nodes and edges [19, 20, 31], the process of how groups and communities are formed [4], as well as methods for mining rules for graph evolution [5].

With respect to community detection in time-evolving graphs, the prominent line of work is to consider different graph snapshots, find communities in each snapshot separately (or by incorporating information from previous snapshots), and then establish correspondences among the communities in consecutive snapshots, so that it is possible to study how communities *appear*, *disappear*, *split*, *merge*, or *evolve*. A number of research papers follows this framework [3, 14, 22, 24, 29]. Similar recent works apply concepts of Laplacian dynamics [23] and frequent pattern mining [6] to ensure coherence and sufficiency of communities found in sequence of graph snapshots.

Many dynamic-graph studies are dedicated to the event-detection problem. The comprehensive tutorial by Akoglu and Faloutsos covers recent research on this topic.[4] The majority of the works focuses on how to compare different graph snapshots, and it aims to detect those snapshots that the graph structure changes significantly. The research tools developed in this area include novel metrics for graph similarity [25] and graph distance—see the survey of Gao et al. [11] and recent paper [28]—as well as extending scan-statistics methods for graphs [27], while a number of papers relies on matrix-decomposition methods [1, 16].

To our knowledge, the approach that is best aligned with our problem setting, is presented by Bogdanov et al., for the problem of mining heavy subgraphs in time-evolving networks [7]. Yet, the two approaches are conceptually very distinct. First, the approach of Bogdanov et al. is still based on network snapshots, and thus sensitive to boundary quantization effects. Second, their concept of heavy subgraphs is based on edge weights, and their discovery problem maps to *prize collecting Steiner tree*, as opposed to a density-based objective.

Hu et al. propose a framework for mining frequent coherent dense subgraphs across a sequence of biological networks [15]. Their core concept is to construct a second-order graph, which represents co-activity of edges in the initial graph. As with the previous papers, Hu et al. work with network snapshots, which is quite a different model than the one we consider in this paper.

[4] `http://www.cs.stonybrook.edu/~leman/icdm12/`

In summary, in contrast to the existing work, in this paper we introduce a new point of view in the area of dynamic graphs, namely, we incorporate in our analysis point-wise interactions between the network nodes.

7 Concluding remarks

In this paper we considered the problem of finding dense dynamic communities in interaction networks, which are networks that contain time-stamped information regarding all the interactions among the network nodes. We formulated the community-discovery problem by asking to find a dense subgraph whose edges occur in short time intervals. We proved that the problem is **NP**-hard, and we provided effective algorithms inspired by methods for finding dense subgraphs.

Our paper is a step towards a more refined analysis of social networks, in which interaction information is taken into account and it is used to provide a more accurate description of communities and their dynamics in the network.

Our work opens many possibilities for future research. First we would like to extend the problem definition in order to discover many dense dynamic communities. This can be potentially achieved by asking to *cover* all (or a large fraction of) the interactions of the network with dense dynamic communities.

Second, we would like to incorporate additional information in our approach. As an example, think that the "smartphone community" discussed in the introduction, may use certain specialized vocabulary, brand names, or hashtags, which can provide additional clues for discovering the community. Our framework uses only time stamps of interactions; complementing our methods with additional information can potentially improve the quality of the results greatly.

Acknowledgements. This work was supported by Academy of Finland grant 118653 (ALGODAN)

References

1. L. Akoglu and C. Faloutsos. Event detection in time series of mobile communication graphs. In *Army Science Conference*, 2010.
2. Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2), 2000.
3. S. Asur, S. Parthasarathy, and D. Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. *TKDD*, 3(4), 2009.
4. L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD*, 2006.
5. M. Berlingerio, F. Bonchi, B. Bringmann, and A. Gionis. Mining graph evolution rules. In *ECML PKDD*, 2009.
6. M. Berlingerio, F. Pinelli, and F. Calabrese. Abacus: frequent pattern mining-based community discovery in multidimensional networks. *Data Mining and Knowledge Discovery*, 27(3):294–320, 2013.
7. P. Bogdanov, M. Mongiovì, and A. K. Singh. Mining heavy subgraphs in time-evolving networks. In *ICDM*, 2011.

8. M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, 2000.
9. G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *KDD*, 2000.
10. S. Fortunato. Community detection in graphs. *Physics Reports*, 486, 2010.
11. X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1), 2010.
12. M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99, 2002.
13. D. F. Gleich and C. Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *KDD*, 2012.
14. D. Greene, D. Doyle, and P. Cunningham. Tracking the evolution of communities in dynamic social networks. In *ASONAM*, 2010.
15. H. Hu, X. Yan, Y. Huang, J. Han, and X. J. Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 2005.
16. T. Ide and H. Kashima. Eigenspace-based anomaly detection in computer systems. In *KDD*, 2004.
17. S. Khuller, A. Moss, and J. S. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1), 1999.
18. A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA*, 2009.
19. R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *KDD*, 2006.
20. J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *KDD*, 2008.
21. J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In *WWW*, 2010.
22. Y. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. Tseng. Facetnet: A framework for analyzing communities and their evolutions in dynamic networks. In *WWW*, 2008.
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(5980):876–878, 2010.
24. G. Palla, A. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 446, 2007.
25. P. Papadimitriou, A. Dasdan, and H. Garcia-Molina. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications*, 1(1), 2010.
26. P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms Applications*, 10(2), 2006.
27. C. E. Priebe, J. M. Conroy, D. J. Marchette, and Y. Park. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, 11(3), 2005.
28. K. Sricharan and K. Das. Localizing anomalous changes in time-evolving graphs.
29. J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *KDD*, 2007.
30. S. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, 2000.
31. R. Zhou, C. Liu, J. X. Yu, W. Liang, and Y. Zhang. Efficient truss maintenance in evolving networks. *arXiv preprint arXiv:1402.2807*, 2014.