Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Polina Rozenshtein

Discovering dynamic communities in interaction networks

Master's Thesis
Espoo, August 11, 2013

Supervisor: Professor Aristides Gionis, Aalto University

Advisors: Prof. Aristides Gionis
Nikolaj Tatti, D.Sc.

Aalto University
School of Science

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF MASTER'S THESIS

Author:	Polina Rozenshtein		
Title: Discovering dynamic communities in interaction networks			
Date:	August 11, 2013	**Pages:**	64
Major:	Information and Computer Science	**Code:**	T-110
Supervisor:	Professor Aristides Gionis		
Advisors:	Prof. Aristides Gionis Nikolaj Tatti, D.Sc.		

Very often online social networks are defined by aggregating information regarding the interaction between the nodes of the network. For example, a call graph is defined by considering an edge for each pair of individuals who have called each other at least once — or at least k times. Similarly, an implicit social network in a social-media site is defined by considering an edge for each pair of users who have interacted in some way, e.g., have made a conversation, commented to each other's content, etc. Despite the fact that this type of definitions have been used to obtain a lot of insights regarding the structure of social networks, it is obvious that they suffer from a severe limitation: they neglect the precise time that the interaction between network nodes occurs.

In this thesis we propose to study *interaction networks*, where one considers not only the underlying topology of the social network, but also the exact time instances 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.

In the context of interaction networks, we study the problem of discovering communities, which are dense in terms of the underlying network structure, 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
Language:	English

Acknowledgements

Working on my thesis I had a great opportunity to be a part of Data Mining group at ICS department. It was my pleasure to work in that outstanding research team of bright people.

I highly appreciate the opportunity to be supervised by the head of Data Mining Group Prof. Aristides Gionis. This thesis was made possible only by his supportive guiding and effective discussions.

I am also truly thankful to my advisor Dr. Nikolaj Tatti for his enormous help and patience. The work in this thesis could not be that smooth and exciting without his high interest and participation.

I would like to thank my friends from Macadamia program, and my friends, who did not forget me even staying at a long distance. Thank you Ilya, Anya and Lena. I especially appreciate priceless and endless chats with my best friend Roman. Being full of joy and fun these conversations lightened up long nights of my hard studies. All the best to you, aniue.

Finally, I would like to express my deep gratitude to my parents Inna and Yuriy, grandparents Olga and Nikolay, brother Egor, and, of course, to my beloved husband Boris and cat Yuki for their love, support, and patience.

Espoo, August 11, 2014

Polina Rozenshtein

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction to community detection

Modern computer science contributes significantly to the process of revealing and understanding hidden structures and dependencies in the complex systems that compound the world. One fundamental model to represent and analyze structural dependences and interactions between objects is a *graph* or a *network*. Graph representation can naturally be applied to systems of objects that share some type of dyadic relationships, dependencies, correlations, or interactions: the objects are mapped to the nodes and the relationships between the objects – to the edges between the corresponding nodes of the graph. Although sometimes the terms *graph* and *network* have distinct meaning, and define undirected and directed graph models respectively, in this work, we use these terms as synonyms. If needed, the directionality is specified explicitly.

Well-known examples of graph models come from various social networks, where the nodes correspond to people, while the edges represent established relationships, such as friendship, kinship, collaboration, employment, or a club or party membership, etc. Other straightforward examples can be found in traffic networks, such as electric power grids, road systems, or the Internet. Chemical and biological systems with various complicated dependencies, such as food chains or protein-protein interactions, are additional instances of real-world networks.

One of the most important structural features of the graph model is a community or a cluster — a group of nodes that induces a dense subgraph, and has only a few edges that connect the group members to other nodes of the graph. This phenomenon is common in real-world networks: roads are dense in cities and sparse in the countryside; people form groups by interests, age, place of residence, or work, and they are unlikely to communicate with people that have other interests and living in distant regions.

Originally, community detection problem was formulated to be applied to social networks; even the word *community* refers to the social context. Social networks tend to have multiple overlapping communities. Understanding these hidden structures and dependencies has applications in sociology, psychology, marketing, and resource management [23, 33, 45].

Another important domain of applications of community detection can be found in a biological setting. A typical example of biological networks is a protein-protein interaction network. Community detection methods group proteins that have similar functions, and those that are involved in the same processes. This grouping leads to classification of the proteins, and an understanding of their roles [59]. The role of genes in a gene-expression network, or enzymes in a metabolic network can be studied in a similar way [11, 55].

Additionally, community detection provides a compact hierarchical representation of a graph that can be visualized and compressed [20, 53]. This application is important for real-world networks, such as the World Wide Web or virtual Facebook-like social networks that contain millions of nodes.

The list of applications of community detection is not limited by these examples, and a more extensive overview can be found in a survey on community detection by S. Fortunato [23].

1.2 Community detection in interaction networks

While networks emerge in different areas of science, and in general have similar structures and properties, in this work, we focus on a specific type of networks – the social networks. Social networks represent people as nodes, and various social interactions as edges. The interaction (e.g. personal communication, phone call, e-mail) can occur multiple times, and at different moments in the recorded network history. In order to enhance the analysis, the dynamic nature of the interaction network should be incorporated into the graph model, for example, by multiple time-annotated edges. Modeling and analysis of the temporal component of networks is the focus of this thesis.

Searching for communities in social networks is one of the most well-studied problems in graph mining. A large number of different methods have been proposed, employing a diverse set of algorithmic tools, such as agglomerative approaches, min-cut formulations, random walks, spectral methods, and more. On the other hand, it has been observed that large networks are characterized by a lack of clear and well-defined communities at a large scale [26, 41].

The lack of well-defined communities can be attributed to the high degree of interconnectivity in social and communication networks, as reported by numerous studies on small-world phenomena [18, 35]: individuals are connected to each other via short chains, and as the size of a community grows, it is very likely to have a large number of links going out of the community. Let us consider an example:

Example 1: The employees of a large company are interconnected. Besides professional interests, they may have diverse hobbies, and belong to some clubs. As the company is large, a large number of employees may have the same hobbies. Consequently, the border between the *club* community and the *company* community vanishes. Moreover, the social aspect and information propagation may lead to further integration of these communities. Through members that belong to both communities, people learn about the other community, and may join. The probability of that action increases with the number of cross-border edges [30, 58].

As we are becoming proficient in storing and analyzing data of increasing volume and richness, it is feasible to consider network data at fine granularity, and to analyze not only the underlying topology, but also the exact time instances of interactions. Analysis of such interaction events can reveal much more information about the structure and dynamics of the communities in the network. In *Example 1*, the *company* and the *club* communities can be distinguished based on the fact that interaction within the *company* community happens during work days, while free-time activities are concentrated on weekends.

The example of two strongly overlapping *company* and *club* communities is very straightforward. In practice, semantic communities in communication networks can be hidden by numerous unrelated communications, which are active at various time points.

To be more concrete, consider the following examples:

Example 2: A group of researchers across many different European institutions are working on a European project. The members of the group go about their everyday lives and other tasks, often unrelated to the European project. However, once every few weeks or months, say, before deadlines of deliverables or project meetings, there is a lot of interaction among the group members, and interactions may occur in various combinations that depend on the sub-project composition and the roles of the people in the project.

Example 3: 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 examples is that the communities we describe are *not* isolated. Their members interact with each other, but they also interact with many other people outside those communities. If one ignores the interaction dynamics and considers only the static topology of the underlying social network, the communities will be hidden and it will be virtually impossible to discover them. It is only when considering the exact time instances of the interactions among the community members that it becomes possible to identify them. In both of the examples above, many interactions occur among the community members, but in a number of relatively short time intervals.

1.3 Focus of this thesis

In this work we formalize the idea exemplified in the previous section. 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 the corresponding problem on static graphs is polynomially-time solvable. For the dynamic density problem we define, we provide effective algorithms inspired by the literature on 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.

The thesis is organized as follows: First, in Section 2 we discuss the related work, briefly covering the evolution of dynamic community detection problem, starting from the case of pure static graph models, and moving towards extensive use of the time component of interaction network models. In Section 3.1 we present our notation and some preliminaries, which we need to formally define the problem. The problem definition is provided in Section 3.3. Next, Section 3.4 contains a complexity analysis of the formulated problem. In Section 4 we present our algorithms, which we evaluate using synthetic and real-world datasets in Section 5. Finally, Section 6 consists of conclusions, and lists directions for future work.

1.4 Acknowledgments and additional references

Part of this thesis will appear as a paper in *The European Conference on Machine Learning and Principles of Knowledge Discovery in Databases, 2014* [52]. The algorithms were developed jointly with Nikolaj Tatti and Aristides Gionis, while the proof of **NP**-hardness in Proposition 1 was given by Nikolaj Tatti. We would like to thank Kiran Garimella, who has kindly provided us with his collection of Twitter users activity in Helsinki area (Twitter dataset in experiments Section 5). The implementations of the proposed algorithms used for testing and evaluation can be found on ICS Data Mining Group page[1].

[1]`http://research.ics.aalto.fi/dmg/software.shtml`

Chapter 2

Related work

A comprehensive survey on different aspects of dynamic network analysis has been recently done by Aggarwal and Subbian [2]. In this work we focus on community detection problem in dynamic network, and here we briefly cover the models and methods that were proposed for community detection in different settings. We start with static community detection, as it is the most basic and well-studies problem formulation, and then cover existing dynamic models with increase of complexity.

2.1 Static community detection

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 partition a graph into disjoint communities [22, 25, 49, 57]; a thorough survey on such methods has been compiled by S. Fortunato [23].

2.2 Dynamic community detection

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. In the dynamic-graph setting, researchers have studied how networks evolve with respect to the arrival of new nodes and edges.

For instance, Kumar et al. [38] have studied evolution of certain types of frequent structures in a social network: singletons, small isolated communities, and giant connected components. They examined and modeled how these structures grow, merge, and exchange the nodes. Another work in that direction [40] studied evolution of dynamic network at "microscopic level" through modeling the process of node arriving and edge establishing. They showed that studying of billions of individual nodes provides a comprehensive understanding of global network structure evolution.

A work by Backstrom et al. [8] addressed the questions of how groups and communities are formed. They thoroughly explored the influence of various structural features on community evolution. Another work [9] proposed to derive graph-evolution rules from frequent patterns adopting a classical association rules framework.

2.3 Snapshot-based community detection

With respect to community detection in time-evolving graphs, a prominent line of work is to consider different graph snapshots, find communities in each snapshot separately (or/and incorporate information from previous snapshots), and then establish correspondences among the communities in consecutive snapshots. This approach makes it possible to study how the communities *appear*, *disappear*, *split*, *merge*, or *evolve*. From a high-level point of view, these typical methods first detect the communities in static graphs induced by one/each time stamp, and then adjust solutions on consequent time stamps to provide consistency. A number of research papers follows this framework [7, 28, 42, 47, 56]. An illustrative example is evolving clustering, proposed by Chakrabarti et al. [15]. This approach provides clustering on adjacent time stamps with respect to *consistency* and *smoothness*. In order to ensure *smoothness*, the clustering on each next time stamp should be as close as possible to the clustering on the previous time stamp. However, the clustering should reflect the data structure accurately to provide *consistency*. Thus, the result clustering is a trade-off between accuracy of structure detection and consistency with historical records.

Similar recent works apply concepts of Laplacian dynamics [46] and frequent pattern mining [10] to ensure coherence and sufficiency of communities found in a sequence of graph snapshots. Another example of snapshot-based approach is based on state-of-the-art spectral clustering extended by consistency constraints [17] or adaptive forgetting factor [62]. A comprehensive study of evolutionary clustering can be found in a survey by M. Spiliopoulou [54].

Many dynamic graph studies are dedicated to event-detection problem, which is intrinsically related to community detection: a common approach of event detection is identification of graph sub-structure (e.g. communities), tracking, and change detection of some key substructures. An extensive tutorial by Akoglu and Faloutsos covers recent research on this topic.[1] The majority of the works focuses on how to compare different graph snapshots, and aims to detect those snapshots where the graph structure changes significantly. The research tools developed in this area include novel metrics for graph similarity [48] and graph distance, (see the survey of Gao et al. [24]), as well as extending scan-statistics methods for graphs [50], while a number of papers relies on matrix-decomposition methods [5, 32].

2.4 Interaction-based community detection

Evolution of the network can be fast and highly dynamic (e.g., Twitter or Facebook). In that case, a snapshot-based model leads to significant loss of information. The interaction-based model works with a stream of communication, and is more informative, but challenging. An interaction stream can be viewed as a sequence of time stamps, where each snapshot has zero time length, and contains only one edge. Not many problems were formulated and addressed under of that model.

Several recent works are dedicated to the *clustering* problem in interaction-based model. Due to a large volume of data to be processed, the devised algorithms need to be memory- and time-efficient. To address the problem of scalability a range of sampling, sketching, and approximation techniques were proposed [4, 64]. Another natural direction is sliding window and other approaches inherited from snapshot-based model [61]. Furthermore, the clustering problem in fast-evolving networks is similar to dense pattern mining, as cluster is a dense structure. A min-hash method can be used to detect tightly-connected groups of nodes in the stream [3].

To our knowledge, the approach that is best aligned with interaction-based community detection problem setting, is presented by Bogdanov et al., for the problem of mining heavy subgraphs in time-evolving networks [13]. The approach of Bogdanov et al. is still based on network snapshots, and forming snapshots from edge time stamps is sensitive to boundary quantization effects. Their concept of heavy subgraphs is based on edge weights, and their discovery problem maps to *prize collecting Steiner tree.*

In a similar direction as the previous paper, Hu et al. propose a framework for mining frequent coherent dense subgraphs across a sequence of biological networks [31]. Their core concept is to construct a second-order graph, which

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

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 work.

In contrast to the existing research, in this work 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. In this new setting we study the problem of finding dense subgraphs with short time support.

Chapter 3

Dense community in interaction network

3.1 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

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

A small example of such network can be found on Figure 3.1. The set of nodes consists of $n = 5$ vertices $V = \{A, B, C, D, E\}$. Their activity is contained in $m = 7$ time stamps, and depicted by solid lines $E = \{(B, D, 1), (B, C, 2), (A, C, 4), (C, D, 5), (D, E, 7), (A, C, 7), (C, E, 10)\}$. We will illustrate the model properties and definitions hereinafter by that toy example.

We consider that interactions are *undirected*. More than one interaction may take place between a pair of nodes, with different time stamps. (E.g. interaction between nodes A and C occurs twice: as $t = 4$ and $t = 7$.) Conversely, more than one interaction may take place at the same time, between different nodes. (Interactions (D, E) and (A, C) occur simultaneously at the time moment $t = 7$.)

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 in the time period of monitoring (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

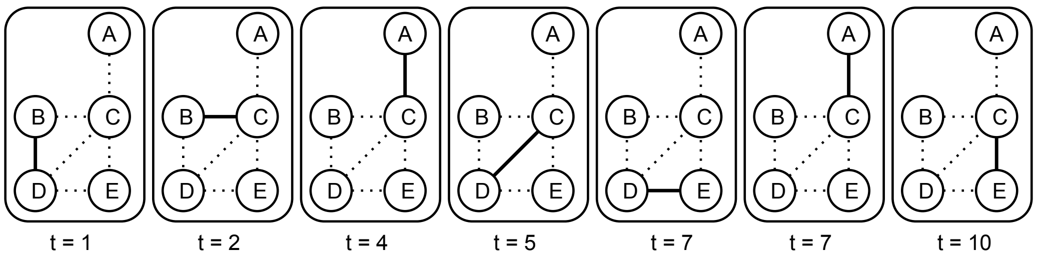


Figure 3.1: A toy example of interaction dynamic network, consisted of $n = 5$ nodes, $m = 7$ time stamps, and spanning $T = 9$ time units.

the *topology network* of G or as the *underlying network* of G. In the dataset on Figure 3.1 $\pi(G)$ is a graph depicted by dotted lines.

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., $\text{span}(T) = f - s$.

For example of *induced interaction network* let us consider $W = \{A, B, C\}$. $E(W)$ is a subset of E, which contains only interactions between these nodes and has a form of: $E(W) = \{(B, C, 2), (A, C, 4), (A, C, 7)\}$. $E(W)$ belongs to the interval which starts at $t = 2$, ends at $f = 7$, and spans 6 time units. Note that $E(W)$ does not induce a continuous subset of time stamps in E, and contains activity of the same edge as many times, as it occurs in E.

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

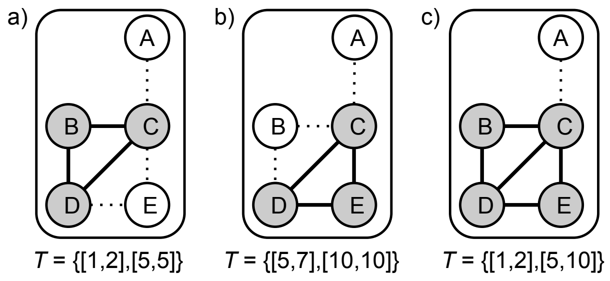

a)
A
B
C
D
E
$\mathcal{T}$ = {[1,2],[5,5]}
b)
A
B
C
D
E
$\mathcal{T}$ = {[5,7],[10,10]}
c)
A
B
C
D
E
$\mathcal{T}$ = {[1,2],[5,10]}

Figure 3.2: Examples of communities in the toy dataset from Figure 3.1.

$\mathcal{T} = (T_1, \ldots, T_K)$. This is achieved by collecting edges from individual time intervals, that is, $G(\mathcal{T}) = (V, E(\mathcal{T}))$, where $E(\mathcal{T}) = \bigcup_{i=1}^{K} E(T_i)$.

An example of *spliced interaction network* can be constructed as a subset of time stamps in Figure 3.1. Let $\mathcal{T} = \{[4, 7]\}$. Then $E(\mathcal{T}) = \{(A, C, 4), (C, D, 5), (D, E, 7), (A, C, 7)\}$. Note that activity of an edge (A, C) is included twice, as these nodes interacted twice during $\mathcal{T}$.

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 this work, 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 $G(W)$.

A "projection" $\pi(G(W, \mathcal{T}))$ of dynamic community $G(W, \mathcal{T})$ along time axis results in a static subgraph of the underlying network. Three examples of projected *dynamic communities* for the toy dataset from Figure 3.1 can be found in Figure 3.2. The community C_a shown in Figure 3.2(a) is defined by $G(\{B, C, D\}, \{[1, 2], [5, 5]\})$, the community C_b from Figure 3.2(b) – by $G(\{C, D, E\}, \{[5, 7], [10, 10]\})$, the community C_c from Figure 3.2(c) – by $G(\{B, C, D, E\}, \{[1, 2], [5, 10]\})$.

3.2 Densest subgraph problem

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 its average degree calculated as 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 [27]. Furthermore, there is a linear-time factor-2 approximation algorithm [6, 16]. The algorithm deletes iteratively a vertex with the lowest degree, obtaining a sequence of subgraphs. Among those subgraphs the algorithm selects the one with the highest density.

Let us measure density for communities in Figure 3.2. Communities C_a and C_b are 3-cliques and have average degree of 2, while $d(C_c) = 2.4$. However, if we restrict $|\mathcal{T}| = K = 2$ for every community, then $\text{span}(\mathcal{T}_a) \leq 1$, $\text{span}(\mathcal{T}_b) \leq 2$ and $\text{span}(\mathcal{T}_c) \leq 6$. Community C_a is strictly preferable to community C_b in the sense of time span minimization and density maximization. In general, these two objectives are in contradiction, and it is intuitively clear that a denser subgraph (such as C_c) is scattered over a longer time period. Furthermore, the densest subgraph can be always found in the network, spliced by the whole time interval $[0, T]$. On the other hand, if we increase the allowed size of $|\mathcal{T}|$ to $K = 3$, then we can cover community C_c with nodes $W_c = \{B, C, D, E\}$ by time intervals $\mathcal{T}_c = \{[1, 2], [5, 7], [10, 10]\}$, and $\text{span}(\mathcal{T}_c) \leq 3$, which is reasonably small in the scale of our toy example. Thus, by introducing and varying constraints values we may obtain more meaningful results. The observations, made on that small example, lead to the problem formulated in the next section.

3.3 Problem formulation

Our goal is to discover communities that are dense and whose interactions occur in several relatively short time intervals. 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 $G(W, \mathcal{T})$ of 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 $\mathcal{T}$ on 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*

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

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 $|W|$.

3.4 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. $\square$

Chapter 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*

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

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

Algorithm 1: Iterative algorithm for finding a dense dynamic community

```
1 $\mathcal{T}_0 \leftarrow$ initial sets of time intervals;
2 $i \leftarrow 0$;
3 while (convergence; $i$++) do
4     $W_{i+1} \leftarrow$ solution to Problem 4 given $\mathcal{T}_i$;
5     $\mathcal{T}_{i+1} \leftarrow$ solution to Problem 3 given $W_{i+1}$;
6 return $(W_i, \mathcal{T}_i)$;
```

the quality of the solution that it returns. However, as it is stated by the following proposition, 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 an optimal solution to Problem 2 for a given interaction network G. Then (i) $\mathcal{T}$ is an optimal solution to Problem 3 given G and W, and (ii) W is an optimal solution to Problem 4 given G and $\mathcal{T}$.*

Proof. (i) The proof comes straightforward as Problems 2 and 3 have the same (up to fixed component W) objective functions and sets of constraints.

Let $(W, \mathcal{T})$ be an optimal solution to Problem 2 for a given interaction network G. Suppose $\mathcal{T}$ is not an optimal solution to Problem 3 given G and W. That means that there exits $\mathcal{T}^*$, such that $q(W, \mathcal{T}^*; G) > q(W, \mathcal{T}; G)$ with $|\mathcal{T}| \leq K$ and $\text{span}(T) \leq B$. That contradicts to our assumption that $(W, \mathcal{T})$ is an optimal solution to Problem 2. Consequently, $\mathcal{T}$ is an optimal solution for Problem 3 given W and G.

(ii) Analogous to (i). □

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.

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 [16], which, as outlined in Section 3.1, 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 $U = \{u_1, \dots, u_m\}$ with weighted elements $w(u_i)$, a collection of subsets $\mathcal{S} = \{S_1, \dots, S_k\}$, 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*

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

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 [34]. 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 [37]. 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.

4.2.1 Greedy approach

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, 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

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

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 greedy heuristic approach as GH. Time complexity of GH is $O(Kn^2)$ (where n is number of time stamps) as we perform brute-force to find each next interval, and can retrieve maximum K intervals.

4.2.2 Binary search approach

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

$$q(W, \mathcal{T}, G) - \alpha \cdot \operatorname{span}(\mathcal{T})\,, \text{ such that } |\mathcal{T}| \leq K. \tag{4.1}$$

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 (4.1) for α_1 and α_2, respectively. Then* $\operatorname{span}(\mathcal{T}_1) \geq \operatorname{span}(\mathcal{T}_2)$.

Proof. Define $\beta_i = \operatorname{span}(\mathcal{T}_i)$ 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 (4.1) 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 (4.1), 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 BS. Note that time complexity of BS equals to $O(Kn^2)$ as well as GH (n stands for number of time stamps), while constant factor depends on number of iteration in binary search.

4.2.3 Dynamic programming approach

Another approach to solution of Problem 3 is based on dynamic programming. As it was shown above, we cannot solve Problem 3 optimally in polynomial time, as it is instance of MCMB. However, we can approximate the cost function of Problem 3, and devise an exact polynomial algorithm for the modified problem. Based on definitions from Chapter 3.1 and properties of set union cardinality we can construct the following upper bound for coverage cost function $q(W, \mathcal{T}; G)$:

$$q(W, \mathcal{T}, G) = d(\pi(G(W, \mathcal{T}))) = \frac{2\left|\pi(G(W, \mathcal{T}))\right|}{|W|} = \frac{2\left|\cup_{i=1}^{K}\pi(G(W, T_i))\right|}{|W|} \leq$$
$$\frac{2\sum_{i=1}^{K}\left|\pi(G(W, T_i))\right|}{|W|} = \sum_{i=1}^{K} d(\pi(G(W, T_i))) = \sum_{i=1}^{K} q(W, T_i, G) = \tilde{q}(W, \mathcal{T}, G) \tag{4.2}$$

The corresponding modified problem has a shape:

Problem 6. *Given budgets K and B, and quality score $\tilde{q}$. With fixed set of nodes W find a time interval set $\mathcal{T}$ that maximizes*

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

Proposition 4. *Optimal solution of Problem 6 is a K-factor approximation of solution of Problem 3.*

Proof. First, note that any feasible solution for Problem 3 is feasible for Problem 6 and vise versa.

Given optimal solution $\mathcal{T}^*$ for Problem 3. Due to Equation 4.2 $q(W, \mathcal{T}^*; G) \leq \tilde{q}(W, \mathcal{T}^*; G)$ and $\tilde{q}(W, \mathcal{T}^*; G) \leq \tilde{q}\left(W, \tilde{\mathcal{T}}^*; G\right)$, where $\tilde{\mathcal{T}}^*$ is optimal interval set for Problem 6.

By construction $\tilde{q}(W, \mathcal{T}, G)$ counts each edge of $\pi(G(W, \mathcal{T}))$ at most K times, while in the original cost function $q(W, \mathcal{T}, G)$ each edge entries exactly once. Namely,

$$q(W, \mathcal{T}, G) \leq \tilde{q}(W, \mathcal{T}, G) = \sum_{i=1}^{K} q(W, T_i, G) \leq K q(W, \mathcal{T}, G)$$

Thus, $\tilde{q}\left(W, \tilde{\mathcal{T}}^*; G\right) \leq K q\left(W, \tilde{\mathcal{T}}^*; G\right)$.

Considering the inequalities altogether, we obtain:

$$q(W, \mathcal{T}^*; G) \leq \tilde{q}(W, \mathcal{T}^*; G) \leq \tilde{q}\left(W, \tilde{\mathcal{T}}^*; G\right) \leq K q\left(W, \tilde{\mathcal{T}}^*; G\right)$$

In addition, $\mathcal{T}^*$ is an optimal solution for Problem 3:

$$\frac{1}{K} q(W, \mathcal{T}^*; G) \leq q\left(W, \tilde{\mathcal{T}}^*; G\right) \leq q(W, \mathcal{T}^*; G)$$

Therefore, optimal solution of Problem 6 is K-factor approximation for Problem 6. $\square$

The Problem 6 can be solved optimally by dynamic programming designed as follows.

With fixed W the edges $E(W)$ can be viewed as $n = |E(W)|$ points (tuples) (u_i, v_i) in 1-dimensional space, ordered by time component t_i in non-descending order. Given budgets K and B, we cover points by K intervals of total length of B. In the frames of Problem 3 we aim on coverage as many *distinct* points as possible, and maximize the cost $q(W, \mathcal{T}; G)$. In the setting of Problem 6, we maximize cost function $\tilde{q}(W, \mathcal{T}; G) = \sum_{T_i \in T} q(W, T_i, G)$. The modified cost function can be calculated for each interval independently.

That property makes it possible to construct a subproblem of dynamic programming.

We define a sub-problem as finding the maximal cost coverage of the first i edges by m intervals in budget b. Note that time budget b should be discretized in order to ensure optimal coverage by dynamic programming. By the nature of Problem 3 time can be discretized in respect of the smallest reasonable time unit, based on granularity of time stamps. Thus, we can record the corresponding maximal cost element $D[i, m, b]$ into $D \in R^{(n+1)\times(K+1)\times(B+1)}$ tensor.

Consider coverage of i points by m intervals in budget b. The optimal cost coverage will contain optimal coverage of some first points $j_s < i$ by $m-1$ intervals in some budget $r_s \leq b$, and coverage of points $j_t, .., i$ $(j_s < j_t)$ by one interval in budget $r \in [b - j_t, b]$. The points with indexes $j : j_s < j < j_t$ will be uncovered. That is, to find an optimal coverage cost $D(i, m, b)$ we need to try all possible splits of the budget $B \geq r_s + r_t$, and all possible end j_s and start j_t points for intervals $m-1$ and m. We do not need to explore splits into more intervals as $D(i_s, m_t - 1, r_s)$ contains the optimal cost coverage of i_s points by $m-1$ interval, thus we cannot obtain better results.

These observations define the subproblem for dynamic programming optimization, and lead to the recurrence equation:

$$D(i, m, b) = \max_{j_s, j_t, r_s, r_t} D(j_s, m-1, r_s) + q(W, [j_t, i], G)$$

where max is over $j_s \in [m-1, i-1]$, $j_t \in [m-1, i-1]$, $r_s \in [0, b]$, $r_t \in [0, b-j_t]$ and $span([j_t, i]) \leq r_t$ with $i \in [1, n], m \in [1, K], b \in [0, B]$. The initial values of D are set to 0, if $i = 0$ or $m = 0$. As points can be covered in zero budget, the other D entries for $b = 0$ should be calculated using recurrence equation.

To backtrack and find the optimal coverage we need an axillary matrix $B \in R^{(n+1)\times(K+1)\times(B+1)}$. An entry of B is a triple: the values of j_s – the end point of the interval $m-1$, j_t – the start point of the interval m, and the budget b_s that was spent on j_s points.

Proposition 5. *Dynamic programming for Problem 6 can be done in time* $O(n^4B^2K)$ *and space* $O(nKB)$*.*

Proof. The size of tensor D is $O(nKB)$. For each entry $D(i, m, b)$ we need to explore submatrix $D(0 : i-1, m-1, 0 : b)$, which is of the size $O(nB)$ and find a start of a new interval. Overall it takes $O(n^2B)$. The cost $q(W, [j_t, i], G)$ and $span([j_t, i])$ can be calculated in $O(n)$ and $O(1)$ respectively. Thus, we need $O(n^4B^2K)$ to fill the cost tensor D. As we store indexes j_s, j_t and budgets r_s in B, backtracking and recovering the coverage can be done in $O(K)$. Overall complexity of dynamic programming is $O(n^4B^2K)$.

As for space complexity, we need to maintain tensor D of a size $O(nKB)$ and tensor B of a size $O(3nKB)$. The space complexity is $O(nKB)$. □

Below we refer to dynamic programing approach as DP.

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.

If 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.

Chapter 5

Experimental evaluation

5.1 Datasets

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

5.1.1 Synthetic data

We simulate activity on a network with a planted dense dynamic 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. For every edge in G, we choose uniformly randomly a time stamp, when the edge is active. Whole time interval of the simulation T. The interactions for the edges of G' occur in some short time period T', with $|T'| \ll |T|$. We set T' to be 10 times shorter than $|T|$. Some basic characteristics of the artificial datasets can be found in Table 5.1. Below we describe the synthetic datasets in details.

Basic Synthetic datasets

We start by testing the ability of our algorithms to discover the planted communities with two families of artificial datasets Synthetic1 and Synthetic2. These datasets span time interval T of $|T| = 1000$ time units. The interactions of the edges of G' are arbitrary planted to be covered by $K = 3$ time intervals with total length of $|T'| = 100$ time units. We assign edges of G

Table 5.1: Characteristics of the synthetic dataset families. Planted community in the datasets of type 1 (Synthetic1, SmallSynthetic1, MultiSynthetic1) is a 5-clique. Planted community in the datasets of type 2 (Synthetic2, SmallSynthetic2, MultiSynthetic2) is an 8-node subgraph. $|V|$ – number of nodes in the dataset; $|\pi(E)|$ – expected number of edges in the underlying network; $d(H)$ – expected density of the planted community; $d(G)$ – expected density of the background network; $|T|$ – length of the whole time interval (in time units); B – time budget required to cover the community activity (in time units); K – number of continuous time intervals that contain community activity; $|C|$ – number of planted communities.

Name	$\|V\|$	$\|\pi(E)\|$	$d(H)$	$d(G)$	$\|T\|$	B	K	$\|C\|$
Synthetic1	100	200	4	1 – 6	1000	100	3	1
Synthetic2	100	200	2 – 7	4	1000	100	3	1
SmallSynthetic1	50	100	4	1 – 6	100	10	3	1
SmallSynthetic2	50	100	2 – 7	4	100	10	3	1
MultiSynthetic1	100	200	4	1 – 6	1000	100	3	3
MultiSynthetic2	100	200	2 – 7	4	1000	100	3	3

uniformly randomly to time instances in T, and edges of G' to time instances in T'.

In the first setting (dataset family Synthetic1) we fix the planted subgraph (a clique) and vary the average degree of the background network. The objective is to test the robustness against background noise. In the second case (dataset family Synthetic2) the average degree of the background network is fixed, while the density of the planted subgraph changes.

Small Synthetic datasets

Being limited by the high time complexity of the DP approach, we generated smaller versions of the dataset families Synthetic1 and Synthetic2 to test and compare performance of all three algorithms. SmallSynthetic1 and SmallSynthetic2 have whole interval T equal to $|T| = 100$ time units, and planted community is constructed to be covered by one time interval with $|T'| = 10$ time units. Similarly to Synthetic1 and Synthetic2, family SmallSynthetic1 is designed to test robustness of clique discovery with varying noise in the background network, while in SmallSynthetic2 we fix background noise and change density of the planted community.

Synthetic datasets with several planted communities
The last two artificial datasets are constructed to examine if the proposed approaches can detect several communities. MultiSynthetic1 and MultiSynthetic2 have the same characteristics as Synthetic1 and Synthetic2, but contain $n = 3$ non-overlapping planted communities of the same expected density. The activity of each community is scattered arbitrary among $K = 3$ intervals T' (individual for each community) and $|T'| = 100$.

5.1.2 Real-world data

We use five datasets of real-world Internet communication. The characteristics of these datasets are summarized in Table 5.2.

Facebook [60]: That dataset is a 3-month-long subset of activity in a New Orleans regional community on Facebook. The data contain anonymized list of wall posts (interactions) with recorded time stamps. The subset covers time period from 9.05.06 to 20.08.06.

Twitter: The dataset tracks activity of Twitter users in the Helsinki region. We consider records that contain mentions of other users as interactions. The span of the used subset is 08.2010 – 10.2010.

Tumblr: This is a subset of the 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 2004-06-27 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.

To test the performance of DP on the real-world networks we created a smaller subsets of the Facebook, Students and Enron datasets. Their characteristics also can be found in Table 5.2.

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

Table 5.2: Basic characteristics of the real-world datasets. $|V|$: number of nodes; $|\pi(E)|$: number of edges of the topology network; $|E|$: number of interactions; $|T|$: time span of the dataset (in days); $d(\pi(G))$: density of the whole topology network. Denote the densest found by Charikar subgraph of the topology network as H. $d(H)$: density of the densest subgraph of the topology network; $|V(H)|$: number of nodes in the densest subgraph of the topology network.

Name	$\|V\|$	$\|\pi(E)\|$	$\|E\|$	$\|T\|$	$d(\pi(G))$	$d(H)$	$\|V(H)\|$
Facebook	4117	5143	10000	104	2.498	5.292	198
Twitter	4605	6006	11868	93	2.608	10.119	67
Tumblr	1980	2454	7645	89	2.479	7.0	18
Students	889	2267	9837	120	5.100	11.292	99
Enron	1143	2019	6245	8080	3.533	14.387	31
FacebookSmall	198	114	150	2	1.151	2.0	3
StudentsSmall	105	83	150	6	1.580	1.809	21
EnronSmall	105	120	150	95	2.285	3.354	31

5.2 Discovering hidden structure

5.2.1 Planted communities

We test the ability of our algorithms to detect the planted communities for different levels of background noise and 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 F-measure, the harmonic mean of precision and recall. Results reported below are averages over $J = 1000$ independent runs.

One planted community

Precision, recall and F-measure results for the two basic families of synthetic datasets Synthetic1 and Synthetic2 are shown in Figures 5.1 and 5.2, 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 than 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

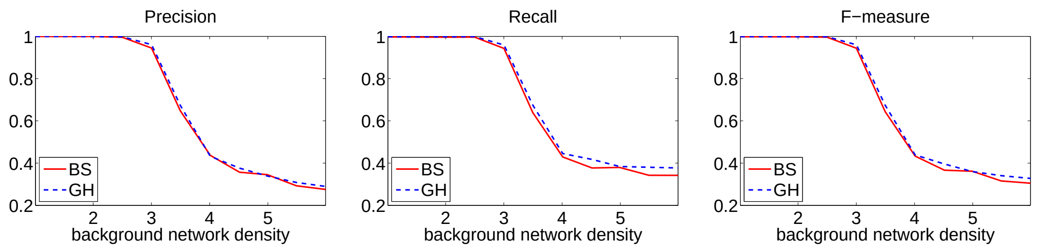


Figure 5.1: Precision, recall and F-measure on Synthetic1, as a function of the background network density. The planted community is a 5-clique scattered over $K = 3$ intervals.

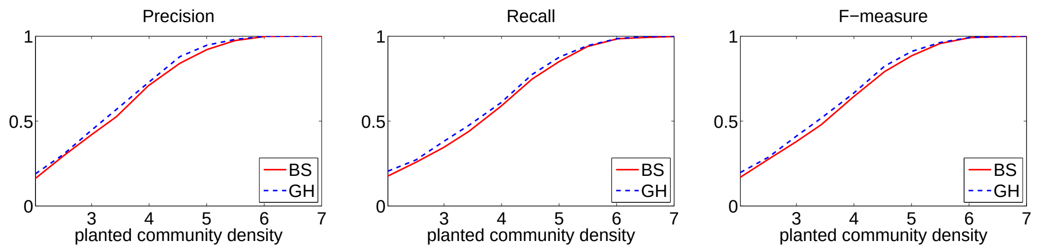


Figure 5.2: Precision, recall and F-measure on Synthetic2, as a function of the density of a planted community of 8 nodes scattered over $K = 3$ intervals. The background network density is set to 4.

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 5.2, are similar: both algorithms perform well when the background network density is smaller than the planted-community density.

The results of all three approaches on the small datasets SmallSynthetic1 and SmallSynthetic2 are presented in Figures 5.3 and 5.4. We generate artificial datasets such that every edge of the planted community appears once. In that case the objective function for DP is equal to the objective that GH and BS optimize. Recall that DP considers the same edges, covered by different time intervals as duplicates, while GH and BS count every covered edge exactly once. Thus, in the case when edges appear with low frequency

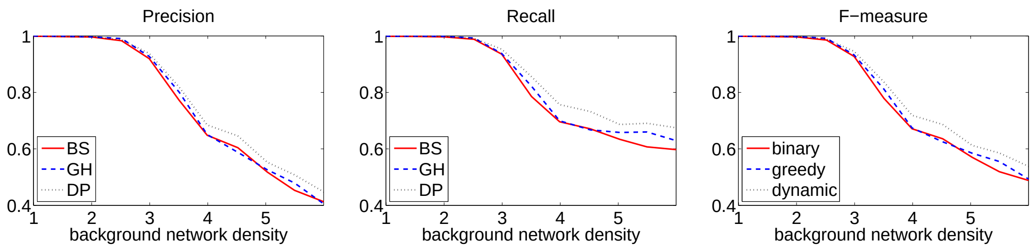


Figure 5.3: Precision, recall and F-measure on SmallSynthetic1, as a function of the background network density. The planted community is a 5-clique scattered over $K = 3$ intervals.

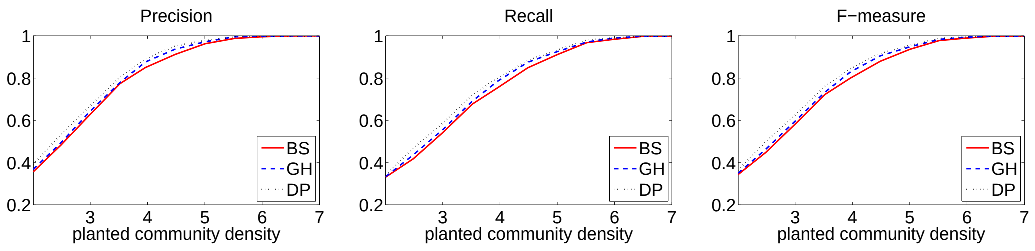


Figure 5.4: Precision, recall and F-measure on SmallSynthetic2, as a function of the density of a planted community of 8 nodes scattered over $K = 3$ intervals. The background network density is set to 4.

DP is expected to perform similar to the other approaches. According to Figures 5.3 and 5.4 DP tends to outperform GH and BS on our synthetic datasets SmallSynthetic1 and SmallSynthetic2. Nevertheless, the difference in the measured performance is not significant.

Several planted communities

Usually the real-world interactive networks contain more than one community. To test the behavior of our approaches in that setting we utilize MultiSynthetic1 and MultiSynthetic2 families of datasets. These datasets contain $|C| = 3$ planted non-overlapping in members communities. Aimed to retrieve all of them we run the algorithms 3 times. After each run we removed edges of the found community from the dataset. The results of that experiments obtained by GH and BS are shown in Figures 5.5 and 5.6. To calculate precision, recall and corresponding F-measure we found the best F-measure match for each of three retrieved communities among the ground-truth communities, and calculate precision and recall with respect to that match. We chose a median (among three retrieved communities) F-measure value to

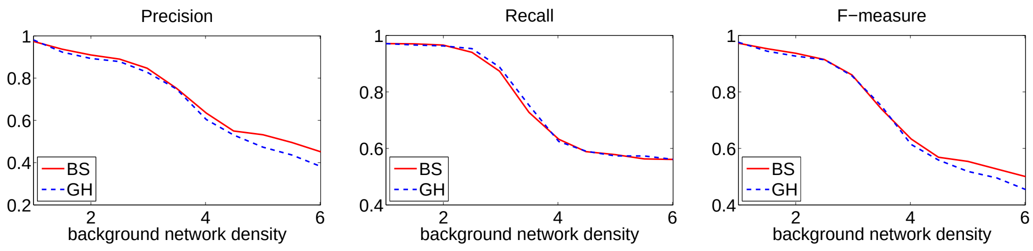


Figure 5.5: Precision, recall and F-measure on MultiSynthetic1, as a function of the background network density. The planted community is a 5-clique scattered over $K = 3$ intervals. Number of planted communities $|C| = 3$.

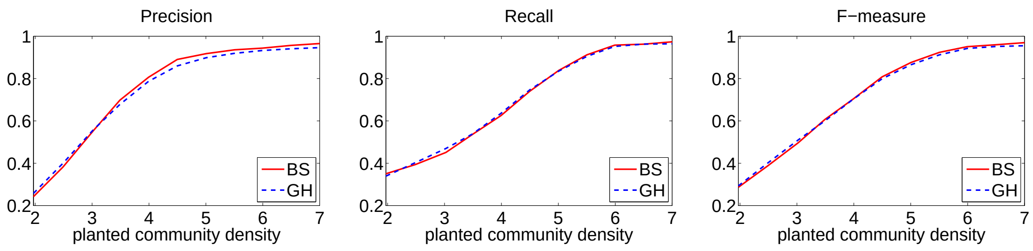


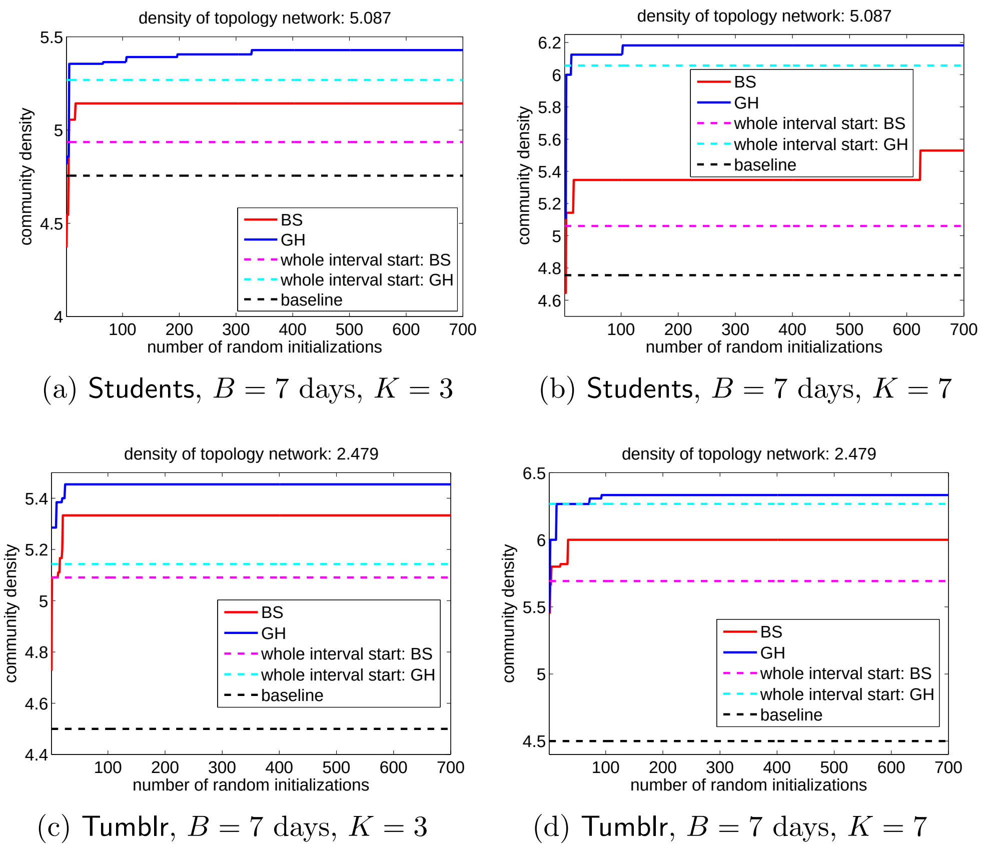
Figure 5.6: Precision, recall and F-measure on MultiSynthetic2, as a function of the density of a planted community of 8 nodes scattered over $K = 3$ intervals. Number of planted communities $|C| = 3$. The background network density is set to 4.

show on the plots.

The resulting Figures 5.5 and 5.6 demonstrate that in the realistic several communities setting the GH and BS approaches have the same tendencies as in the case of one planted community datasets. The performance stays high until the noise level becomes significant, and the average degree of the planted community is close to the average degree of the underlying network.

5.2.2 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 $\mathcal{T}_0 = \{T\}$ may mislead. To overcome this problem and avoid dense structures that cannot be covered in the given time budget, we initialize Algorithm 1



(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, $K = 7$

Figure 5.7: Effect of random initializations on the real-world datasets.

with many random time intervals, and return the best solution found.

The improvement of performing random initializations is shown in Figure 5.7. 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.

5.2.3 Discovered communities

Table 5.3 reports the densities of the communities discovered by our algorithms in the real-world datasets (for results on larger range of parameters K and B please see Table A.1 in Appendix A). We use $J = 200$ random initial-

Table 5.3: Densities of discovered subgraphs. B – time budget in days.

			Community density			Community size		
Dataset	B	K	GH	BS	Base	GH	BS	Base
Facebook	1	1	2.4	2.4	2.4	5	5	5
		5	3.666	3.666	2.4	6	6	5
		10	3.75	3.75	2.4	8	8	5
	7	1	3	3	3	6	6	6
		5	3.875	4	3	16	9	6
		10	4.285	4.47	3	14	17	6
Twitter	1	1	4	4	4	6	6	6
		5	5.111	5.333	4	9	9	6
		10	6.4	6.4	4	10	10	6
	7	1	4.666	4.666	4.666	9	9	9
		5	6	6.222	4.666	14	9	9
		10	6.923	7.2	4.666	13	15	9
Tumblr	1	1	3.866	3.866	3.866	30	30	30
		5	5.111	5.25	3.866	9	8	30
		10	5.818	6.181	3.866	11	11	30
	7	1	4.5	4.5	4.5	8	8	8
		5	5.888	6	4.5	18	11	8
		10	6.714	6.8	4.5	14	15	8
Students	1	1	3.411	3.333	3.428	17	15	21
		5	4.666	4.625	3.428	9	16	21
		10	5.5	5.625	3.428	16	16	21
	7	1	4.755	4.697	4.755	45	43	45
		5	5.826	6	4.755	46	25	45
		10	6.764	7.121	4.755	34	41	45
Enron	1	1	6.181	6.181	6.181	11	11	11
		5	10	10.37	6.181	17	16	11
		10	12.2	12.38	6.181	20	21	11
	7	1	6.363	6.363	6.363	11	11	11
		5	11.26	11.23	6.363	19	26	11
		10	13.07	13.07	6.363	28	28	11

izations. Here and in later experiments 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 GH and BS perform equally well, while in some settings BS yields denser communities than GH.

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 close to 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 5.2 and 5.3, 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-long Enron dataset has average degree 14.387, while we were able to discover a subgraph with average degree 13.07 in a budget of 7 days, spanning 10 time intervals.

A typical retrieved dense community has a size of about 10 to 20 nodes, which is reasonably small for dense social communities, such as friends, group-mates or co-workers. Moreover, this value is independent from the parameter selection in the tested range, and is similar for all the datasets. We believe that a small size is a general property of the dynamic communities, successfully captured by the considered problem formulation. On the contrary, the communities found as the densest subgraphs of the underlying network (Table 5.2) have large size (up to 200 nodes) and large variance.

Community discovery by DP approach

The characteristic of communities, found by all three considered algorithms on small subsets of real datasets can be found in Table 5.4. As we remember from the tests on artificial datasets SmallSynthetic1 and SmallSynthetic2 (Figures 5.3 and 5.4), DP was able to outperform GH and BS. However, in the synthetic dataset the expected number of each edge occurrence was equal to one, which does not hold for real datasets. That should lead to poor performance of the DP in comparison to the two other approaches. Nonetheless, Table 5.4 reports that the results of DP are close to densities obtained by GH and BS, although DP never outperforms them. In spite of promising results on the artificial and real datasets, DP is impractical for

Table 5.4: Densities of discovered subgraphs on the small real-world datasets. B – time budget in hours.

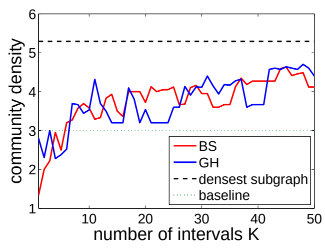
			Community density				Community size			
Dataset	B	K	GH	BS	DP	Base	GH	BS	DP	Base
EnronSmall	1	1	3.4	3.4	3.4	3.4	10	10	10	10
		3	3.4	3.4	3.3	3.4	10	10	20	10
	3	1	3.4	3.4	3.4	3.4	10	10	10	10
		3	3.4	3.4	3.3	3.4	10	10	20	10
StudentsSmall	1	1	1.857	1.857	1.857	1.866	14	14	14	15
		3	1.866	1.857	1.6	1.866	15	14	5	15
	3	1	1.9	1.9	1.9	1.857	20	20	20	14
		3	1.857	1.882	1.333	1.857	14	17	3	14
FacebookSmall	1	1	1.5	1.5	1.142	1.5	4	4	14	4
		3	2	2	1	1.5	3	3	2	4
	3	1	1.5	1.5	1.333	1.5	4	4	3	4
		3	2	2	1	1.5	3	3	2	4

analysis of large social networks due to its high complexity. Thus, we limit the experimental analysis of DP performance by Table 5.4, and further focus on the GH and BS.

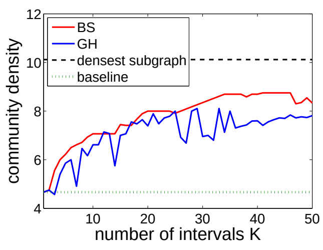
5.2.4 Effect of parameter selection

Effect on the number of intervals K

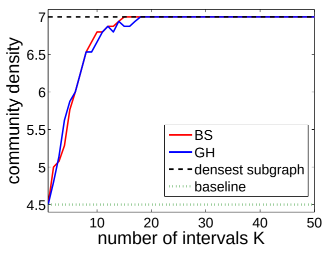
We tested how the choice of number of intervals K affects the density of discovered communities. The resulting densities for all datasets with fixed time budget B and K ranges from 1 to 50 are shown in Figure 5.8. We run GH and BS starting from the whole time interval T (without random initialization), and compare the resulting density with the density of the subgraph obtained by the brute-force baseline, and the density of the densest subgraph of the underling network. As expected, the density of the discovered community grows with increasing of the number of intervals K. With significantly large critical K the algorithms are able to cover the densest subgraph of the underlying network (which is an upper bound for dynamic community density in the considered problem formulation). The critical value of K depends on the dataset and is related to the number of edges in the densest subgraph. For



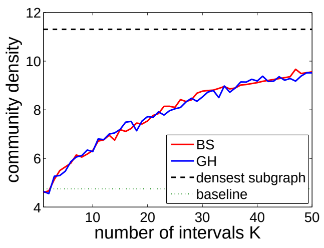
(a) Facebook



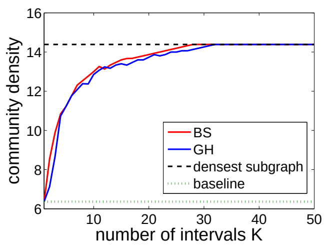
(b) Twitter



(c) Tumblr



(d) Students



(e) Enron

Figure 5.8: Effect of parameter K. Time budget B fixed to $B = 7$ days. The initial interval equals to the whole interval T (no random initialization).

example, for Tumblr dataset the densest subgraph consists of 18 nodes with density 7 (Table 5.2), hence, it contains 63 edges. For Enron, the number of edges in the densest subgraph is 223, while for Twitter it is 339. None of these communities cannot be guaranteed to be covered by $K = 50$ intervals, however, for Tumblr and Enron it was possible to cover the whole densest subgraph in significantly smaller number of time intervals. Additionally, the density retrieved by our algorithm is significantly higher than the baseline.

Effect of time budget B

The impact of different time budget B is shown in Figures 5.9. We fixed $K = 7$ and started from initial interval $T_0 = T$. The density of the retrieved community grows with increasing of the time budget. The rate of increase is rather slow and tends to saturation, which illustrates the importance of the parameter K.

5.2.5 Retrieved intervals

In this section we present various experimental results, related to the time intervals that cover found communities. All the results in that section were obtained without use of random initialization.

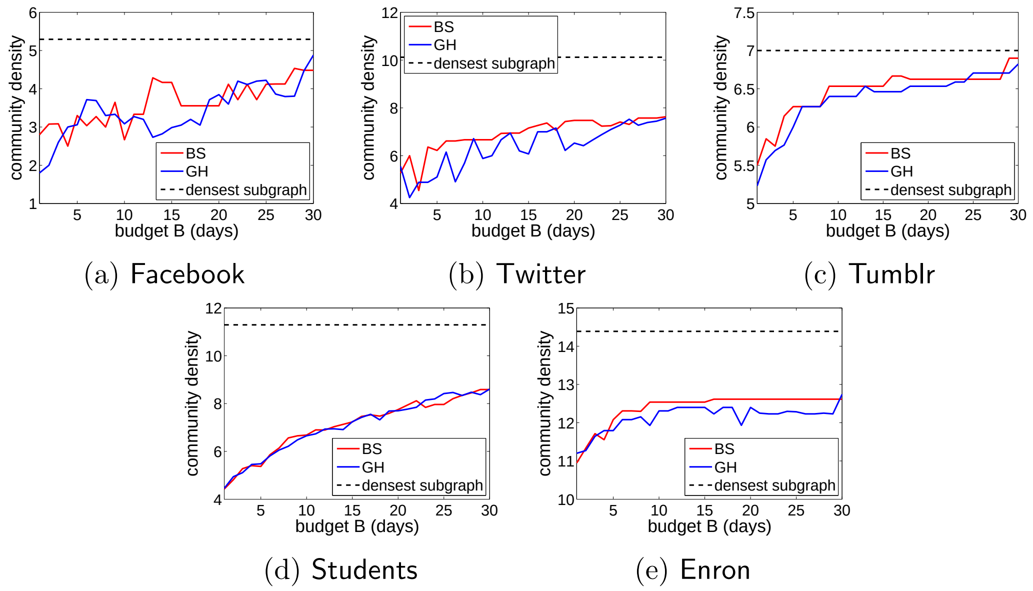


Figure 5.9: Effect of parameter B. Number of intervals K fixed to $K = 7$. The initial interval equals to the whole interval T (no random initialization).

Role of using multiple intervals

As it is shown in Table 5.3, GH and BS are able to find community of significant density, comparable to the densest subgraph of the underlying network. Table 5.5 illustrates that the found communities fit the time budget only in virtue of concept of several intervals K. Column *spent* B reports total length of the retrieved time intervals. Column *span* indicates minimal length of some single time interval that covers all the edges of the retrieved community subgraph. According to Table 5.5, the coverage of a retrieved dense community by one time interval requires a large time budget (up to several months) that is comparable to the whole time span of the dataset. Moreover, *span* is typically much larger than the time budget we spent constructing the K intervals.

Characteristics of the intervals

Table 5.5 presents characteristics of the captured time intervals for different datasets, $K \in \{3, 5, 10\}$ and $B = 3$ days. The first observation is that both GH and BS use the time budget quite tightly, however, GH typically tends to utilize B more extensive than BS. Nonetheless, BS achieves slightly higher densities than GH (see Table 5.3). From the columns *median* $|T_i|$,*min* $|T_i|$ and *max* $|T_i|$ one can see that budget B is spread rather evenly between

Table 5.5: Total time span of the found communities. B – time budget (days); spent B – actually utilized time budget (days); span – length (in days) of the minimal length time interval, needed to cover the retrieved community in one time interval.

			GH			BS		
Dataset	B	K	density	spent B	span	density	used B	span
Facebook	1	3	3	0.39	9.09	3.11	0.79	18.1
		7	3.77	0.72	20.2	3.71	0.72	29.5
	7	3	3.75	4.44	20.2	3.75	3.57	20.2
		7	4.23	5.61	73	4.28	4.9	90.1
Twitter	1	3	3.66	0.41	32.3	4.57	0.46	51.7
		7	5.6	0.69	32.7	6	0.92	72.4
	7	3	5.33	6.99	32.7	5.63	5.84	79.3
		7	6.53	6.49	79.5	6.66	5.09	79.3
Tumblr	1	3	4.33	0.87	33.4	4.66	0.77	56.8
		7	5.45	0.82	49.4	5.63	0.9	49.4
	7	3	5.45	6.76	44.9	5.5	5.92	37.3
		7	6.3	6.99	82.1	6.36	5.92	60.4
Students	1	3	4	0.75	8.51	4.25	0.54	8.61
		7	5.33	0.99	29.5	5.2	0.92	55.3
	7	3	5.42	6.91	25.3	5.42	6.66	32.2
		7	6.13	6.56	76.1	6.57	6.96	73.4
Enron	1	3	8.76	0.83	69.4	8.76	0.83	69.4
		7	11.2	0.98	320	11.2	0.88	158
	7	3	9.6	4.32	69.4	9.89	6.45	158
		7	12	6.64	320	12.3	5.35	320

the intervals. However, BS outputs intervals with more diverse time length than GH: its maximum is typically higher, and minimum is lower. With increasing of K the maximal length time interval, found by GH, shrinks, while the maximal length interval output by BS remains the same. GH is designed to spread the time budget among the intervals in balance. On the contrary, BS greedily maximizes the gain of each new interval, and checks budget constraints on B only a posteriori in the loop of binary search. Thus, on selecting the next interval BS tends to spent a large fraction of the time budget B, and the last retrieved intervals have length that is close to zero.

Table 5.6: Time length characteristics of the retrieved intervals. Time budget B fixed to 3 days; K – number of intervals; spent B – actually used time budget (days); median $|T_i|$ – median length of the retried K intervals; min $|T_i|$ and max $|T_i|$ – minimal and maximal time length of the found K intervals.

		spent B		median $\|T_i\|$		min $\|T_i\|$		max $\|T_i\|$	
Dataset	K	GH	BS	GH	BS	GH	BS	GH	BS
Facebook	3	2.99	2.1	0.96	0.82	0.71	0.01	1.32	1.26
	5	2.85	2.26	0.61	0.15	0.01	0	1.23	1.26
	10	2.95	2.84	0.13	0.08	0	0	1.67	1.42
Twitter	3	2.88	0.46	1.09	0	0.45	0	1.32	0.45
	5	2.52	2.11	0.45	0.07	0.07	0	1.19	1.7
	10	2.91	1.25	0.24	0	0	0	0.85	0.45
Tumblr	3	2.94	2.78	0.99	0.99	0.79	0.79	1.15	0.99
	5	2.88	2.81	0.62	0.79	0.3	0	0.7	0.99
	10	2.8	2.97	0.23	0.08	0.06	0	0.53	0.99
Students	3	2.99	2.69	1.02	0.08	0.68	2.31	1.28	2.61
	5	2.51	2.81	0.5	0.1	0.03	0.03	1.08	2.04
	10	2.96	2.81	0.3	0.12	0.18	0	0.4	1.24
Enron	3	2.28	2.26	0.69	0.69	0.64	0.64	0.94	0.92
	5	2.97	2.96	0.64	0.69	0.11	0	0.92	0.92
	10	2.56	2.6	0.11	0.09	0	0	0.82	0.7

The next Table 5.7 reflects how the density of the resulting community is spread among the intervals. Each time interval on average covers a sub-community of a small density, while the union of the intervals results in a denser structure. The sparsest sub-community has degree close to 1, the densest sub-community is neither comparable to the union of all K intervals. However, due to the uneven time length of the retrieved intervals (Table 5.6), the densest sub-community constructed by BS is generally denser than the one constructed by GH.

To show how scattered the retrieved time intervals are, we collected experimental results related to the length of the time gaps between K found intervals. According to Table 5.8, the solutions for all datasets with different parameters contain significant gaps between the output time intervals. The change of gaps size with incrementing of K is dataset specific, and, as we believe, reflects the structure and behavior of the dynamic communities. In

Table 5.7: Density of the sub-communities of the retrieved intervals. Time budget B fixed to 3 days; K – number of intervals; total density – density of the found community in the union of K intervals; avg $d(T_i)$ – average density of the sub-community, covered by one of K time intervals; min $d(T_i)$ and max $d(T_i)$ – minimal and maximal density of the sub-community, covered by one of K retrieved intervals.

		total density		avg $d(T_i)$		min $d(T_i)$		max $d(T_i)$	
Dataset	K	GH	BS	GH	BS	GH	BS	GH	BS
Facebook	3	3.33	3.66	1.53	1.74	1	1.33	2	2.4
	5	3.75	3.75	1.45	1.52	1	1	2	2.33
	10	4	4.15	1.27	1.29	1	1	1.5	1.71
Twitter	3	4.66	4.57	2.66	2.16	2	1	4	4
	5	5.8	5.77	2.11	1.92	1.6	1	3.71	4
	10	6.4	6.4	1.61	1.6	1.2	1	3.71	3.71
Tumblr	3	5.27	5.23	2.41	2.52	1.8	2	3.2	3.33
	5	5.63	5.71	2.05	2.06	1.42	1.33	3	3.38
	10	6.18	6.57	1.85	1.83	1	1	2.88	3.33
Students	3	4.43	4.72	2.52	2.14	2.23	1.33	3.04	3.4
	5	5	5.4	1.6	1.7	1.14	1	2	2.66
	10	6.08	6.13	1.7	1.63	1.53	1.11	2	2.42
Enron	3	9.6	9.6	4.14	4.14	3.5	3.5	5.09	5.09
	5	10.8	10.8	3.44	3.47	2	1.77	4.6	5.09
	10	12.7	12.7	2.62	2.9	1.6	1.71	4.54	5.16

some cases, such as in Facebook, the gaps grow when more time intervals are allowed. Maximal time gap increases, and the retrieved communities become more scattered in time. In the case of Twitter and Tubmlr the average gap size decreases, along with the maximal and minimal gap sizes, thus we can conclude that the community activity in these datasets is spread more evenly.

Each node of the discovered community appears on average in a small number of time intervals, as it can be seen from Table 5.9. Thus, each of these time intervals contributes equally, and there is no redundancy. Moreover, there are typically some nodes that are covered by exactly one interval. Similar properties are illustrated by Table 5.10, which shows how often the same community edge appears in the output intervals. That table emphasizes the role of each interval, as on average the community edges (the interactions)

Table 5.8: Length characteristics of the time gaps between the retrieved intervals. Time budget B fixed to 3 days; K – number of the intervals; avg $\bar{T}_i$ – average length of the time gaps between K retried intervals; min $\bar{T}_i$ and max $\bar{T}_i$ – minimal and maximal length of the time gaps between K found intervals.

		avg $\bar{T}_i$		min $\bar{T}_i$		max $\bar{T}_i$	
Dataset	K	GH	BS	GH	BS	GH	BS
Facebook	3	5.1	5.17	1.17	1.3	9.04	9.04
	5	4.35	4.5	1.3	0.23	9.04	9.04
	10	9.68	9.7	0.56	0.86	44.6	44.3
Twitter	3	38.7	43.8	3.69	36.3	73.7	51.3
	5	19.4	21.4	1.24	7.27	42.4	42.4
	10	9.48	9.66	0.72	0.16	34.6	42.4
Tumblr	3	31.6	31.7	20.7	20.9	42.5	42.6
	5	17.2	17.1	5.22	5.19	43.3	37.3
	10	8.86	8.83	1.86	0.55	16.4	34.1
Students	3	3.46	5.76	1.51	4.82	5.4	6.71
	5	6.96	6.68	1.7	5.17	14.6	9.23
	10	6.7	8.1	1.86	0.81	14.1	21.4
Enron	3	33.5	33.5	20.5	20.5	46.6	46.6
	5	39	38.8	4.82	20.5	83.7	61
	10	35.3	35.3	5.01	6.77	161	74.4

occur in not more than 1.5 time intervals. The maximal number of intervals that contain the same edge is usually smaller than the number of intervals.

The examples of captured community and covering time intervals can be found of Figures 5.10 and 5.11. Each subplot corresponds to one time interval with start and end points written in its title. The active edges of the time interval are marked red. These figures illustrate the properties discussed above: dense community is composed by small components scattered among the time intervals of various time length.

Table 5.9: Frequency of the community nodes in the retrieved time intervals. Time budget B fixed to 3 days; K – number of the intervals; avg Fr_n – average number of the retrieved intervals K where the same community node appears; min Fr_n and max Fr_n – minimal and maximal number of retrieved intervals K where the same community node appears.

		avg Fr_n		min Fr_n		max Fr_n	
Dataset	K	GH	BS	GH	BS	GH	BS
Facebook	3	2	2	1	1	3	3
	5	2.37	2.37	1	2	3	3
	10	3.27	3.15	2	2	4	4
Twitter	3	1.77	1.71	1	1	3	2
	5	3	2.55	2	1	4	4
	10	4.6	3.9	2	2	7	6
Tumblr	3	2.54	2.38	2	1	3	3
	5	3.54	2.78	2	2	5	5
	10	5.45	4.21	1	2	9	8
Students	3	1.96	2.09	1	1	3	3
	5	3.5	3	3	2	5	4
	10	4.44	3.66	1	2	7	7
Enron	3	2.46	2.46	1	1	3	3
	5	3.58	3.38	1	1	5	5
	10	5.5	5.03	1	1	10	9

5.2.6 Several communities

The social networks contain more than one community. Our algorithms were able to discover several communities on the synthetic dataset (recall Figures 5.5 and 5.6). On the real datasets we used the same strategy, as on the synthetic datasets: with fixed parameters K and B we run our algorithms n times to obtain n communities. Before every next run we removed the edges of all previously found communities from the dataset. The results can be found in Figures 5.12,5.13,5.14 and 5.15. We compare properties of the discovered communities with corresponding properties of the densest subgraphs, found in the underlying network, using similar strategy: we remove edges of once detected the densest subgraph from the networks, and iterate

Table 5.10: Frequency of the community edges in the retrieved time intervals. Time budget B fixed to 3 days; K – number of the intervals; avg Fr_e – average number of the retrieved intervals K where the same community edge appears; min Fr_e and max Fr_e – minimal and maximal number of the retrieved intervals K where the same community edge appears.

		avg Fr_e		min Fr_e		max Fr_e	
Dataset	K	GH	BS	GH	BS	GH	BS
Facebook	3	1	1	1	1	1	1
	5	1	1.06	1	1	1	2
	10	1.09	1.11	1	1	3	3
Twitter	3	1.04	1	1	1	2	1
	5	1.13	1.03	1	1	2	2
	10	1.25	1.12	1	1	3	3
Tumblr	3	1.17	1.17	1	1	3	3
	5	1.32	1.17	1	1	4	3
	10	1.7	1.34	1	1	5	6
Students	3	1.14	1.03	1	1	3	2
	5	1.12	1.03	1	1	3	2
	10	1.25	1.05	1	1	4	2
Enron	3	1.05	1.05	1	1	2	2
	5	1.13	1.1	1	1	2	2
	10	1.17	1.16	1	1	3	3

finding the densest subgraphs. We stop when the set of the remaining edges is empty.

As it is shown in Figure 5.12, retrieving communities based on the topology network without time component results in several highly dense communities. However, the density of each new found community decreases (especially dramatically for Facebook dataset). On the contrary,the proposed algorithms produce more communities of significant density. Moreover, the size of the communities (Figure 5.13), found in the topology network is unreasonably large, while our algorithms keep output communities of size around 10–20 nodes (we have to use lin-log scale to visualize the resulting community sizes on the same figure with output of GH and BS). Furthermore, Figure 5.14 depicts that the time span of a single interval needed to cover the under-

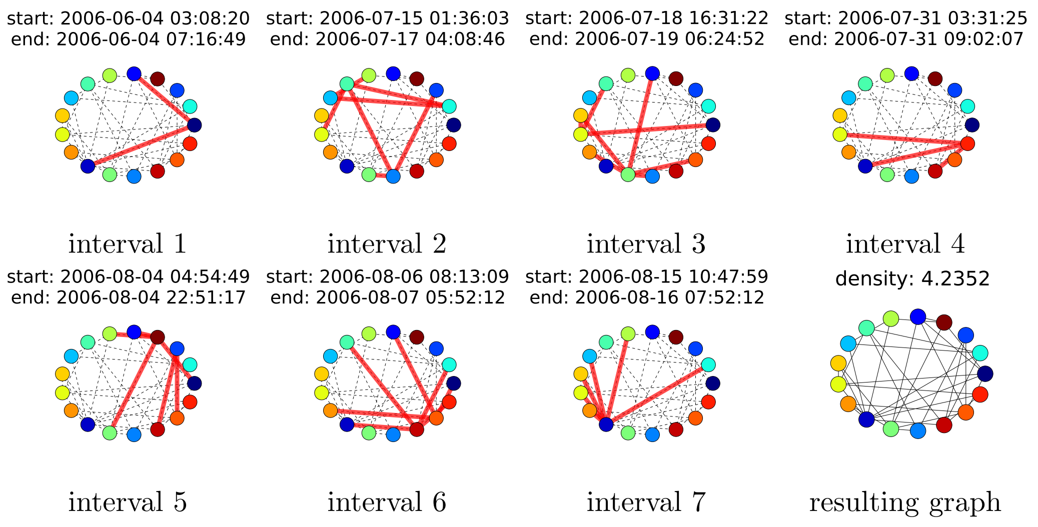


Figure 5.10: Example of community found by GH. Facebook dataset, $K = 7$ and $B = 7$ days.

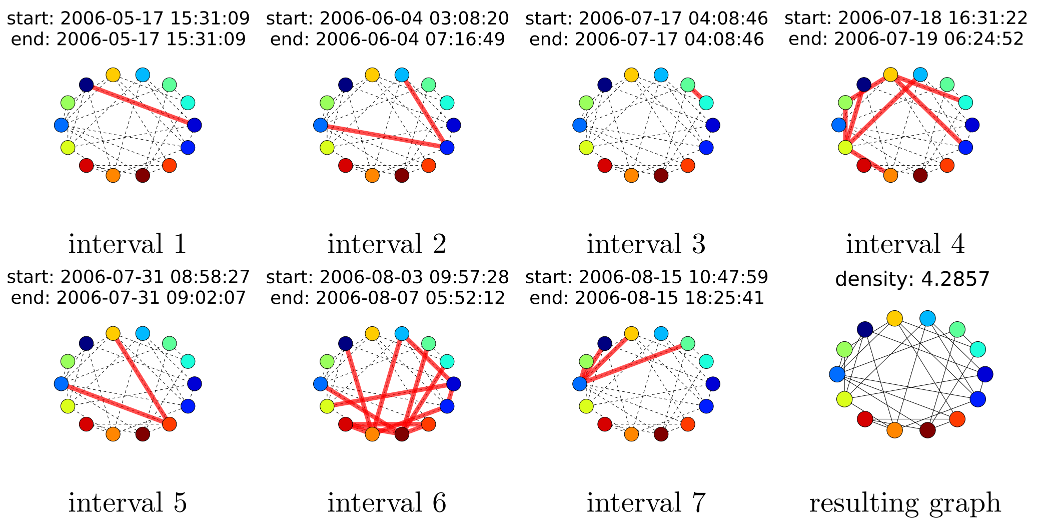
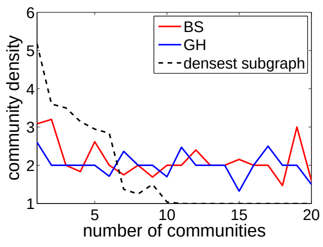
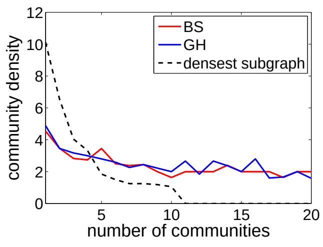


Figure 5.11: Example of community found by BS. Facebook dataset, $K = 7$ and $B = 7$ days.

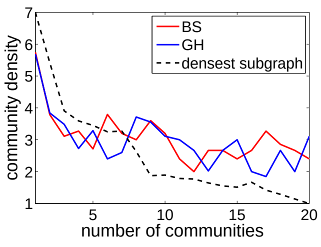
lying network densest subgraphs is large (is comparable to the time span of the whole dataset.) Thus, to find exact intervals, where the discovered community is active, one needs further processing, which is incorporated into our algorithms.



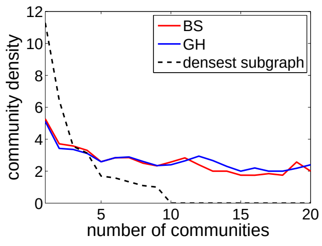
(a) Facebook.



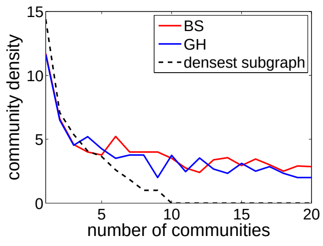
(b) Twitter



(c) Tumblr

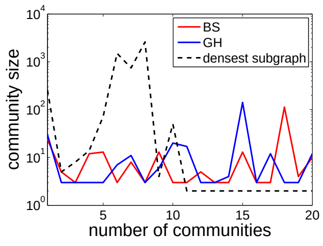


(d) Students

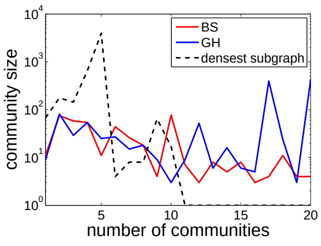


(e) Enron

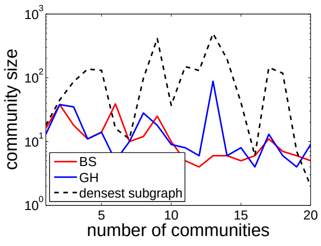
Figure 5.12: Density of multiple discovered communities. Comparison of several found communities with dense subgraphs of topology network. $K = 7$, $B = 3$ days.



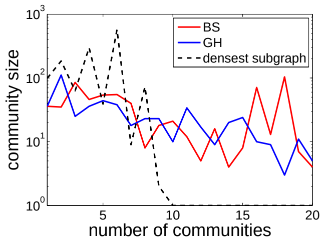
(a) Facebook



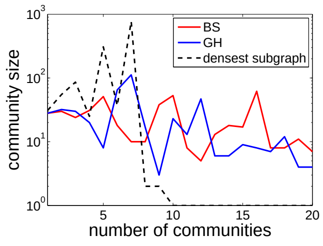
(b) Twitter



(c) Tumblr

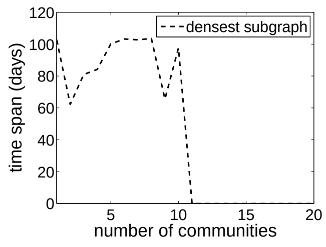


(d) Students

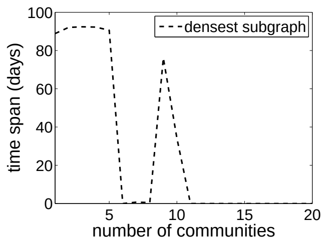


(e) Enron

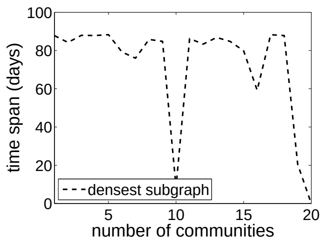
Figure 5.13: Size of multiple discovered communities. Comparison of several found communities with dense subgraphs of topology network. $K = 7$, $B = 3$ days.



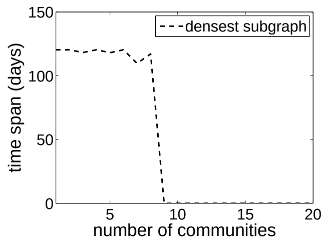
(a) Facebook



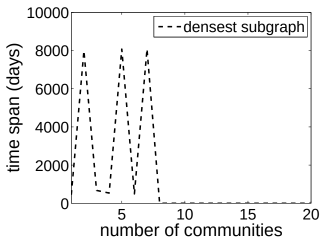
(b) Twitter



(c) Tumblr

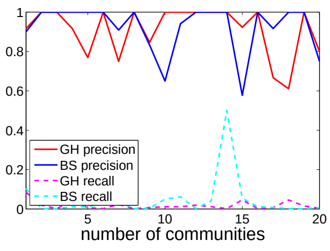


(d) Students

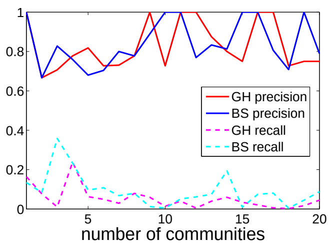


(e) Enron

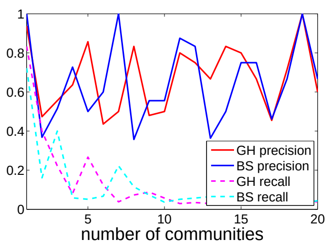
Figure 5.14: Time span of multiple communities discovered on the underlying network.



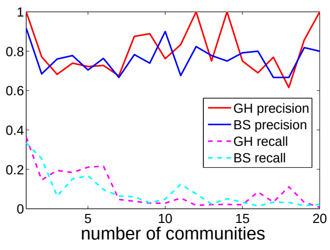
(a) Facebook



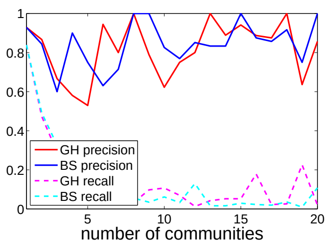
(b) Twitter



(c) Tumblr



(d) Students



(e) Enron

Figure 5.15: Relations between multiple output communities and dense subgraphs of topology network. $K = 7$, $B = 3$ days.

It would be expected that communities found by GH and BS are subsets of communities, captured on the underlying network. To test this hypothesis, we compare every retrieved community C_i, $i \in \{1, .., n\}$ to every underlying network densest subgraph U and find the best match. For a retrieved community C_i we define the best match community $U(C_i) \in U = \{U_1, .., U_n\}$ as $U(C_i) = \arg\max_{U_j \in U} P(C_i, U_j)$. Where set function P has a meaning of precision with U_j as a ground-truth set. Using the best matching underlying network community $U(C_i)$ we calculate precision $I_P(C_i) = P(C_i, U(C_i))$ and recall $I_R(C_i) = R(C_i, U(C_i))$ for each retrieved community. The resulting plots are shown in Figures 5.15. In these plots we refer to I_P as precision and to I_R as recall. High precision I_P and low recall I_R for all $n = 20$ found communities show that the underlying network densest subgraphs are supersets of the communities, obtained by GH and BS. That observation confirm our hypothesis that dynamic communities are hidden in the dense subgraphs of underlying network.

5.2.7 Twitter community example

The Twitter dataset contains texts of the tweets. To illustrate the retrieved community we use a set of hash-tags that appear in the community communication. First of all, we note that hash-tags from the retrieved interactions are typically the most frequent tags in the found community during the recorded history. That means that our algorithms are able to capture the most important activity of the community. For example, using GH algorithm with $K = 7$ intervals, $B = 3$ days we found community of 8 Twitter users with density $= 6.0$. There are some organizations such as aaltoes[4] and AaltoGarage[5] among them. Thus, there is no surprise that retrieved intervals contain the following hash-tags: summerofstartups, startup, entrepreneur, slush10, aaltoes, me310, vc, churchillclub. Considering all activity in that 8-nodes community we can see that all of these tags are on the top of frequent tags, and represent the community interests well. The list of top frequent hash-tags (with corresponding number of occurrence) in the community: aaltoes: 52, startup: 12, vc: 11, summerofstartups: 11, entrepreneur: 7, startups: 4, web: 3, slush10: 2, skype: 2, funrank: 2, africa: 2, mobile: 2, demoday: 2, design: 2, linkedin: 2, aalto: 2.

On the other hand, a community found as the densest subgraph of the underlying network consists of 67 nodes with density of 10.119. That is a rather large community. The list of hash-tags used in their communication

[4]`http://aaltoes.com/`
[5]`http://www.aaltovg.com/`

is of size 112, and contains a less focused set of hash-tags on the top: aaltoes: 80, summerofstartups: 28, startup: 18, vc: 13, ff: 11, fb: 8, elonmerkki: 7, entrepreneur: 7, slush10: 6, newtwitter: 5, 2010mvv: 4, garage48: 4, facebook: 4, web: 4, startups: 4, smss2010: 4, mvv2010: 3, angrybirds: 3, fail: 3, spotonloc: 3, fif2010: 3, baltic: 3, africa: 3, mobile: 3, demoday: 3, helsinki: 3, gov20: 3, e20: 3, failcon: 2, bacon: 2, mindtrek: 2, skype: 2, funrank: 2, nxcfi: 2, blog: 2, yc: 2, hankenes: 2, design: 2, education: 2, linkedin: 2, nokia: 2, n8: 2, aalto: 2.

From that list we can see that tags with the highest frequency are covered by communication of communities retrieved by our algorithm. Coupling that observation with Figure 5.15, we can conclude that the community found by our algorithm is a subset of the densest subgraph of the underlying network, and can be viewed as its semantic dynamic core.

Chapter 6

Conclusions

In this work 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 work 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.

This work opens many possibilities for future research. First, 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 hash-tags, 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.

Second, we would like to improve scalability of the proposed algorithms to process the big data datasets that come from Twitter or Facebook.

Furthermore, it would be interesting to compare empirically performance of our approaches with snapshot based techniques on different granularity.

In addition, the proposed approaches may be modified to find ego-centric dynamic communities, and used for node classification and role-discovery in the communication network.

Other directions of feature work are related to utilizing other concepts of dense structure, considering frequency and statistical properties of the interactions.

Bibliography

[1] Adomavicius, G., and Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on 17*, 6 (2005), 734–749.

[2] Aggarwal, C., and Subbian, K. Evolutionary network analysis: A survey.

[3] Aggarwal, C. C., Li, Y., Yu, P. S., and Jin, R. On dense pattern mining in graph streams. *Proceedings of the VLDB Endowment 3*, 1-2 (2010), 975–984.

[4] Aggarwal, C. C., Zhao, Y., and Philip, S. Y. On clustering graph streams. In *SDM* (2010), SIAM, pp. 478–489.

[5] Akoglu, L., and Faloutsos, C. Event detection in time series of mobile communication graphs. In *Army Science Conference* (2010).

[6] Asahiro, Y., Iwama, K., Tamaki, H., and Tokuyama, T. Greedily finding a dense subgraph. *Journal of Algorithms 34*, 2 (2000).

[7] Asur, S., Parthasarathy, S., and Ucar, D. An event-based framework for characterizing the evolutionary behavior of interaction graphs. *TKDD 3*, 4 (2009).

[8] Backstrom, L., Huttenlocher, D. P., Kleinberg, J. M., and Lan, X. Group formation in large social networks: membership, growth, and evolution. In *KDD* (2006).

[9] Berlingerio, M., Bonchi, F., Bringmann, B., and Gionis, A. Mining graph evolution rules. In *ECML PKDD* (2009).

[10] Berlingerio, M., Pinelli, F., and Calabrese, F. Abacus: frequent pattern mining-based community discovery in multidimensional

networks. *Data Mining and Knowledge Discovery 27*, 3 (2013), 294–320.

[11] Beyer, A., Thomason, P., Li, X., Scott, J., and Fisher, J. Mechanistic insights into metabolic disturbance during type-2 diabetes and obesity using qualitative networks. In *Transactions on Computational Systems Biology XII*. Springer, 2010, pp. 146–162.

[12] Bilgin, C. C., and Yener, B. Dynamic network evolution: Models, clustering, anomaly detection. *IEEE Networks* (2006).

[13] Bogdanov, P., Mongiovì, M., and Singh, A. K. Mining heavy subgraphs in time-evolving networks. In *ICDM* (2011).

[14] Chakrabarti, D., Faloutsos, C., and McGlohon, M. Graph mining: Laws and generators. In *Managing and Mining Graph Data*. Springer, 2010, pp. 69–123.

[15] Chakrabarti, D., Kumar, R., and Tomkins, A. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), ACM, pp. 554–560.

[16] Charikar, M. Greedy approximation algorithms for finding dense components in a graph. In *APPROX* (2000).

[17] Chi, Y., Song, X., Zhou, D., Hino, K., and Tseng, B. L. On evolutionary spectral clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD) 3*, 4 (2009), 17.

[18] Collins, J. J., and Chow, C. C. It's a small world. *Nature 393*, 6684 (1998).

[19] Csermely, P. Creative elements: network-based predictions of active centres in proteins and cellular and social networks. *Trends in biochemical sciences 33*, 12 (2008), 569–576.

[20] Cui, W., Zhou, H., Qu, H., Wong, P. C., and Li, X. Geometry-based edge clustering for graph visualization. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (2008), 1277–1284.

[21] Dhillon, I. S. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (2001), ACM, pp. 269–274.

[22] Flake, G. W., Lawrence, S., and Giles, C. L. Efficient identification of web communities. In *KDD* (2000).

[23] Fortunato, S. Community detection in graphs. *Physics Reports 486* (2010).

[24] Gao, X., Xiao, B., Tao, D., and Li, X. A survey of graph edit distance. *Pattern Analysis and applications 13*, 1 (2010).

[25] Girvan, M., and Newman, M. E. J. Community structure in social and biological networks. *PNAS 99* (2002).

[26] Gleich, D. F., and Seshadhri, C. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *KDD* (2012).

[27] Goldberg, A. V. *Finding a maximum density subgraph.* University of California Berkeley, CA, 1984.

[28] Greene, D., Doyle, D., and Cunningham, P. Tracking the evolution of communities in dynamic social networks. In *ASONAM* (2010).

[29] Guimera, R., and Amaral, L. A. N. Functional cartography of complex metabolic networks. *Nature 433*, 7028 (2005), 895–900.

[30] Hopcroft, J., Khan, O., Kulis, B., and Selman, B. Tracking evolving communities in large linked networks. *Proceedings of the national academy of sciences of the United States of America 101*, Suppl 1 (2004), 5249–5253.

[31] Hu, H., Yan, X., Huang, Y., Han, J., and Zhou, X. J. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics* (2005).

[32] Ide, T., and Kashima, H. Eigenspace-based anomaly detection in computer systems. In *KDD* (2004).

[33] Java, A., Song, X., Finin, T., and Tseng, B. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis* (2007), ACM, pp. 56–65.

[34] Khuller, S., Moss, A., and Naor, J. S. The budgeted maximum coverage problem. *Information Processing Letters 70*, 1 (1999).

[35] Kleinberg, J. Navigation in a small world. *Nature 406* (2000).

[36] Krishnamurthy, B., and Wang, J. On network-aware clustering of web clients. In *ACM SIGCOMM Computer Communication Review* (2000), vol. 30, ACM, pp. 97–110.

[37] Kulik, A., Shachnai, H., and Tamir, T. Maximizing submodular set functions subject to multiple linear constraints. In *SODA* (2009).

[38] Kumar, R., Novak, J., and Tomkins, A. Structure and evolution of online social networks. In *KDD* (2006).

[39] Leicht, E. A., and Newman, M. E. Community structure in directed networks. *Physical review letters 100*, 11 (2008), 118703.

[40] Leskovec, J., Backstrom, L., Kumar, R., and Tomkins, A. Microscopic evolution of social networks. In *KDD* (2008).

[41] Leskovec, J., Lang, K. J., and Mahoney, M. W. Empirical comparison of algorithms for network community detection. In *WWW* (2010).

[42] Lin, Y., Chi, Y., Zhu, S., Sundaram, H., and Tseng, B. Facetnet: A framework for analyzing communities and their evolutions in dynamic networks. In *WWW* (2008).

[43] Linden, G., Smith, B., and York, J. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE 7*, 1 (2003), 76–80.

[44] McGlohon, M., Akoglu, L., and Faloutsos, C. Statistical properties of social networks. In *Social Network Data Analytics*. Springer, 2011, pp. 17–42.

[45] Mislove, A., Viswanath, B., Gummadi, K. P., and Druschel, P. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining* (2010), ACM, pp. 251–260.

[46] Mucha, P. J., Richardson, T., Macon, K., Porter, M. A., and Onnela, J.-P. Community structure in time-dependent, multiscale, and multiplex networks. *Science 328*, 5980 (2010), 876–878.

[47] Palla, G., Barabási, A., and Vicsek, T. Quantifying social group evolution. *Nature 446* (2007).

[48] Papadimitriou, P., Dasdan, A., and Garcia-Molina, H. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications 1*, 1 (2010).

[49] Pons, P., and Latapy, M. Computing communities in large networks using random walks. *Journal of Graph Algorithms Applications 10*, 2 (2006).

[50] Priebe, C. E., Conroy, J. M., Marchette, D. J., and Park, Y. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory 11*, 3 (2005).

[51] Reddy, P. K., Kitsuregawa, M., Sreekanth, P., and Rao, S. S. A graph based approach to extract a neighborhood customer community for collaborative filtering. In *Databases in Networked Information Systems*. Springer, 2002, pp. 188–200.

[52] Rozenshtein, P., Tatti, N., and Gionis, A. Discovering dynamic communities in interaction networks. In *ECML PKDD* (2014).

[53] Schaeffer, S. E. Graph clustering. *Computer Science Review 1*, 1 (2007), 27–64.

[54] Spiliopoulou, M. Evolution in social networks: A survey. In *Social network data analytics*. Springer, 2011, pp. 149–175.

[55] Stuart, J. M., Segal, E., Koller, D., and Kim, S. K. A gene-coexpression network for global discovery of conserved genetic modules. *science 302*, 5643 (2003), 249–255.

[56] Sun, J., Faloutsos, C., Papadimitriou, S., and Yu, P. S. Graphscope: parameter-free mining of large time-evolving graphs. In *KDD* (2007).

[57] van Dongen, S. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, 2000.

[58] Van Vlaenderen, H. Community development research: Merging communities of practice. *Community Development Journal 39*, 2 (2004), 135–143.

[59] Vázquez, A., Flammini, A., Maritan, A., and Vespignani, A. Modeling of protein interaction networks. *Complexus 1*, 1 (2002), 38–44.

[60] Viswanath, B., Mislove, A., Cha, M., and Gummadi, K. P. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)* (August 2009).

[61] Wang, C.-D., Lai, J.-H., and Yu, P. Dynamic community detection in weighted graph streams. In *Proc. of SDM* (2013), SIAM, pp. 151–161.

[62] Xu, K. S., Kliger, M., and Hero, A. Evolutionary spectral clustering with adaptive forgetting factor. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on* (2010), IEEE, pp. 2174–2177.

[63] Zachary, W. W. A survey of approaches and issues in machine-aided translation systems. *Computers and the Humanities 13*, 1 (1979), 17–28.

[64] Zhao, Y., and Philip, S. Y. On graph stream clustering with side information. In *Proceedings of the seventh SIAM International Conference on Data Mining* (2013), SIAM, pp. 139–150.

Appendix A

Main results extended table

Table A.1: Extended version of Table 5.3. Densities of discovered subgraphs. B – time budget in days.

			Community density			Community size		
Dataset	B	K	GH	BS	BASE	GH	BS	BASE
Facebook	1	1	2.4	2.4	2.4	5	5	5
		3	3	3.111	2.4	6	9	5
		5	3.666	3.666	2.4	6	6	5
		7	3.777	3.714	2.4	9	7	5
		10	3.75	3.75	2.4	8	8	5
	3	1	2.75	2.75	2.75	8	8	8
		3	3.333	3.666	2.75	6	6	8
		5	3.75	3.75	2.75	8	8	8
		7	3.866	3.875	2.75	15	16	8
		10	4	4.153	2.75	11	13	8
	7	1	3	3	3	6	6	6
		3	3.75	3.75	3	8	8	6
		5	3.875	4	3	16	9	6
		7	4.235	4.285	3	17	14	6
		10	4.285	4.47	3	14	17	6
Twitter	1	1	4	4	4	6	6	6
		3	3.666	4.571	4	6	7	6
		5	5.111	5.333	4	9	9	6
		7	5.6	6	4	10	8	6
		10	6.4	6.4	4	10	10	6

	3	1	4.444	4.444	4.444	9	9	9
		3	4.666	4.571	4.444	9	7	9
		5	5.8	5.777	4.444	10	9	9
		7	6	5.818	4.444	8	11	9
		10	6.4	6.4	4.444	10	10	9
	7	1	4.666	4.666	4.666	9	9	9
		3	5.333	5.636	4.666	9	11	9
		5	6	6.222	4.666	14	9	9
		7	6.533	6.666	4.666	15	12	9
		10	6.923	7.2	4.666	13	15	9
Tumblr	1	1	3.866	3.866	3.866	30	30	30
		3	4.333	4.666	3.866	6	9	30
		5	5.111	5.25	3.866	9	8	30
		7	5.454	5.636	3.866	11	11	30
		10	5.818	6.181	3.866	11	11	30
	3	1	4	4	4	8	8	8
		3	5.272	5.23	4	11	13	8
		5	5.636	5.714	4	11	14	8
		7	5.818	6.133	4	11	15	8
		10	6.181	6.571	4	11	14	8
	7	1	4.5	4.5	4.5	8	8	8
		3	5.454	5.5	4.5	11	12	8
		5	5.888	6	4.5	18	11	8
		7	6.307	6.363	4.5	13	11	8
		10	6.714	6.8	4.5	14	15	8
Students	1	1	3.411	3.333	3.428	17	15	21
		3	4	4.25	3.428	14	8	21
		5	4.666	4.625	3.428	9	16	21
		7	5.333	5.2	3.428	9	15	21
		10	5.5	5.625	3.428	16	16	21
	3	1	3.84	3.8	3.84	25	30	25
		3	4.437	4.727	3.84	32	11	25
		5	5	5.4	3.84	10	10	25
		7	5.333	5.47	3.84	9	34	25
		10	6.08	6.133	3.84	25	30	25
	7	1	4.755	4.697	4.755	45	43	45
		3	5.428	5.428	4.755	28	42	45
		5	5.826	6	4.755	46	25	45

		7	6.137	6.578	4.755	29	38	45
		10	6.764	7.121	4.755	34	41	45
Enron	1	1	6.181	6.181	6.181	11	11	11
		3	8.769	8.769	6.181	13	13	11
		5	10	10.37	6.181	17	16	11
		7	11.2	11.2	6.181	20	20	11
		10	12.2	12.38	6.181	20	21	11
	3	1	6.181	6.181	6.181	11	11	11
		3	9.6	9.6	6.181	15	15	11
		5	10.82	10.88	6.181	17	18	11
		7	11.68	11.71	6.181	19	28	11
		10	12.72	12.76	6.181	22	26	11
	7	1	6.363	6.363	6.363	11	11	11
		3	9.6	9.894	6.363	15	19	11
		5	11.26	11.23	6.363	19	26	11
		7	12.08	12.3	6.363	25	26	11
		10	13.07	13.07	6.363	28	28	11