Online Density Bursting Subgraph Detection from Temporal Graphs

by

Yanyan Zhang

M.Eng., University of Chinese Academy of Sciences, 2014
B.Eng., Shandong University, 2011

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Science

SIMON FRASER UNIVERSITY
Summer 2018

Approval

Name:	**Yanyan Zhang**
Degree:	**Master of Science (Computer Science)**
Title:	**Online Density Bursting Subgraph Detection from Temporal Graphs**
Examining Committee:	**Chair:** Qianping Gu Professor School of Computing Science
	Jian Pei Senior Supervisor Professor School of Computing Science
	Jiannan Wang Supervisor Assistant Professor School of Computing Science
	Martin Ester Internal Examiner Professor School of Computing Science
Date Defended:	**May 7, 2018**

Abstract

Given a temporal weighted graph that consists of a potentially endless stream of updates, we are interested in finding density bursting subgraphs (DBS), where a DBS is a subgraph that accumulates its density at the fastest speed. Online DBS detection enjoys many novel applications. At the same time, it is challenging since the time duration of a DBS can be arbitrarily long but a limited size storage can buffer only up to a certain number of updates. To tackle this problem, we observe the critical decomposability of DBSs and show that a DBS with a large time duration can be decomposed into a set of indecomposable DBSs with equal or larger burstiness. We further prove that the time duration of an indecomposable DBS is upper bounded and propose an efficient method $TopkDBSOL$ to detect indecomposable DBSs in an online manner. Extensive experiments demonstrate the effectiveness, efficiency, and scalability of $TopkDBSOL$ in detecting significant DBSs from temporal graphs.

Keywords: dense subgraph, online, temporal graph, bursting subgraph

Dedication

To my family.

Acknowledgements

I owe my deepest gratitude to my senior supervisor, Dr. Jian Pei, for his continuous guidance, enthusiam and encouragement throughout my studies, and throughout the process of researching and writing this thesis. His rigorous attitude for science, extensive knowledge on the statistics and computing science, and broad vision for the specialized area influenced me a lot and led me to think deeply on my studies, research, and even the future careering development.

I am deeply grateful to Dr. Jiannan Wang for being my supervisor and offering me helpful suggestions on this thesis as well as how to be a good data researcher.

My sincere gratitude is also given to Dr. Martin Ester, the internal examiner of my supervisory committee, for his kind help and insightful comments.

I also owe my great gratitude to my lab mates for their kind help and encouragement. A particular acknowledgement gose to Juhua Hu, Xiangbo Mao, Yu Yang, Chuancong Gao, Xiao Meng, Zicun Cong, Zijin Zhao, Xia Hu, Yajie Zhou.

Last but not least, I want to express my sincere gratitude to my husband, Dr. Lingyang Chu, for his love and company. Being my closest family as well as best friend, he gives me continuous support throughout my studies and research. My warmest gratitude would give to my parents, Chuanfang Zhang and Chuanqin Ma, for giving birth to me and offering me their unconditional love and support. Also, I would like to thank my dear sister Meng Zhang, for her love and support, as well as her company with our parents.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we first introduce the motivation of this thesis, then illustrate the key ideas and major contributions of our work. In the end, we introduce the structure of the thesis.

1.1 Motivation

General Eric Shinseki said, "If you don't like change, you're going to like irrelevance even less". Finding the most and fastest changing parts is a central task in analyzing temporal data. For example, in a series of snapshots of a social network, analysts are often interested in the density bursting subgraphs where a group of people gain intra-group connection density at the fastest speed. More specifically, in a stream of snapshots of a business collaboration network, where each vertex is a person or a company and the weight of an edge represents the collaboration strength between two parties in the time duration of a snapshot, a density bursting subgraph is a group of parties whose collaboration strengths in between increase dramatically fast. Such a density bursting subgraph may indicate a new business consortium is forming, for example, due to new business opportunities like ICO.

As another concrete example, the taxi trips in a city naturally form a temporal network, where each vertex is a location in the city and the weight of an edge is the number of taxi trips between two locations during a specific time period. In such a network, a density bursting subgraph indeed reveals a burst of taxi trips among a group of locations. Our case study in Figure 6.5 and Section 6.4 give two examples of such bursts of taxi trips, which reveal interesting travel patterns of people on weekdays and weekends, respectively.

We can also form a network where each vertex is a keyword and the weight of an edge is the frequency two keywords co-occurring in an article in a specific year. Then, a bursting subgraph in a series of snapshots of the network indeed suggests a topic that gains fast growth in a period. Our experimental results (Table 6.4) give 6 such example bursting topics and their corresponding periods.

If one wishes, the list of possible applications of finding density bursting subgraphs can easily keep growing. Surprisingly, although finding density bursting subgraphs is interesting

and has many applications, this problem has not been touched systematically in literature. As reviewed in Chapter 2, the existing works on finding dense subgraphs [7, 1, 30, 33, 35] only focus on density but do not consider the speed of density changes. The previous works on dense temporal subgraph detection [37, 27, 31, 2, 3, 6, 32] maintain dense subgraphs against incremental or streaming updates, but again do not account the change speed of density. Since a dense subgraph may slowly accumulate a large density in a long time, it may not necessarily be a density bursting subgraph. Therefore, existing dense (temporal) subgraph detection methods cannot be straightforwardly extended to detect density bursting subgraphs.

As will be investigated in Section 4.2, Chapter 4, a closer look finds out that the problem of finding density bursting subgraphs is far from trivial. Due to the nature of weighted graphs, a burst can last for a long and potentially indefinite time. A static method has to buffer all the snapshots involved in a burst, thus cannot handle a large number of updates. This leaves us no choice but to design an efficient online algorithm for density bursting subgraph detection.

1.2 Major Idea and Contributions

In this thesis, we tackle the novel problem of online density bursting subgraph detection. Specifically, given a stream of snapshots of a temporal graph, a *density bursting subgraph* (DBS) is a subgraph that accumulates its density at the fastest speed during a time interval. We measure the speed of density accumulation by *burstiness*, which is the ratio between the density gain of a subgraph and the time to accumulate the density. We make the following contributions.

First, we consider the static Top-k DBS Finding (TDF) version of the problem, that is, all snapshots of a temporal graph are available and we find the set of DBSs with the top-k largest burstiness. We model the static TDF problem as a mixed integer programming problem and show that it is NP-hard. We also propose a baseline method $SlideWin$ to find a good solution to the TDF problem by iteratively solving a constrained quadratic programming (CQP) problem and a maximum density segment (MDS) problem.

Second, considering the general DBS detection problem, we show that the time duration of a DBS can be arbitrarily long, and thus a straightforward extension of $SlideWin$ or alike does not work. To tackle the problem systematically, we follow a principled approach – we try to identify the "atomic" components of DBSs. Critically, we observe that a DBS with a large time duration can be decomposed into a set of indecomposable DBSs with equal or larger burstiness. Most importantly, we show that the time duration of an indecomposable DBS is upper bounded, which makes it possible to detect indecomposable DBSs in an online manner.

Last, we formulate the online Top-k DBS Finding (OTDF) problem, which is to detect top-k indecomposable DBSs online. The OTDF problem is also NP-hard. We develop an efficient algorithm $TopkDBSOL$, which achieves a 2-approximation of the top-1 indecomposable DBS. Our extensive experiments and an interesting case study clearly show that our method is effective and efficient in finding DBSs from large temporal graphs.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows. We review the related works in Chapter 2, and formulate the static Top-k DBS Finding problem (TDF) in Chapter 3. In Chapter 4, we investigate how to find a single DBS and present the proposed static baseline method $SlideWin$. We also discuss the major challenges of online DBS detection in this chapter. In Chapter 5, we explore the decomposition properties of DBS, present the online Top-k DBS Finding (OTDF) problem and develop $TopkDBSOL$. We report a systematic empirical study in Chapter 6 and conclude the thesis in Chapter 7.

Chapter 2

Related Work

Online density bursting subgraph detection is a novel task that has not been touched in literature. Nevertheless, as one of the most widely used structures to model the relationship between entities, graphs, as well as detecting dense subgraphs from (temporal) graphs, have been studied extensively in literature. Our work is highly relevant to dense subgraph detection, dense temporal subgraph detection, and Maximum Density Segment (MDS) Problem, which works as a building block in our method. Therefore, we review these three subjects in this chapter.

2.1 Dense Subgraph Detection

Detecting dense subgraphs from static graphs is a well investigated task [7, 1, 30, 33, 35]. Our work is most related to the cohesiveness based methods [8, 29, 23, 24, 11], which measure the density of a subgraph by cohesiveness, that is, a quadratic function $\mathbf{x}^\top A\mathbf{x}$ of a subgraph embedding $\mathbf{x} \in \triangle^n$, where $\triangle^n = \{\mathbf{x} \mid \sum_i \mathbf{x}_i = 1, \mathbf{x}_i \geq 0\}$ represents standard n-dimensional simplex.

For unweighted graphs, Motzkin *et al.* [28] proved that maximizing the cohesiveness is equivalent to finding the maximum clique in a graph. For weighted graphs, Pavan *et al.* [29] proposed Dominent Set (DS), which detects clique-like dense subgraphs by applying an optimization method named replicator dynamics to find local maximum points of cohesiveness. However, the $O(n^2)$ time complexity of replicator dynamics limits the scalability of DS. To further improve the efficiency of DS, Bulò *et al.* [8] proposed a more efficient optimization method named Infection Immunization Dynamics (IID) to find local maximum points of cohesiveness. IID is a population game dynamics motivated by the analogy with infection and immunization processes within a population of "game players". The time complexity of IID is $O(n)$. Since most dense subgraphs exist in local regions, Liu *et al.* [24, 23] proposed the shrinking and expansion algorithm (SEA) to efficiently search local maximum points of cohesiveness in small local subgraphs and prevent unnecessary time and space cost.

All these cohesiveness based methods efficiently find clique-like dense subgraphs that are well demonstrated to be stable and robust to noise [26, 12, 24, 10, 39, 14]. However, they cannot process the temporal information inherent in a potentially endless stream of snapshots.

2.2 Dense Temporal Subgraph Detection

Detecting dense temporal subgraphs from temporal graphs has attracted much attention in recent years [37, 27, 31, 2, 3, 6, 32, 38].

Bogdanov *et al.* [6] proposed MEDEN to detect dense temporal subgraphs with a large sum of edge weights. Ma *et al.* [27] proposed FIDES, which is three orders of magnitudes faster than MEDEN, for the same problem. The above research works detect a dense subgraph by finding the connected temporal subgraph with the largest sum of positive edge weights. As a result, these algorithms tend to detect a very large connected subgraph in a graph without negative edge weights, which does not have much value in practice.

Yang *et al.* [37] used γ-quasi-clique to find the set of most diversified γ-dense subgraphs that can cover the original temporal subgraph as much as possible. Boden *et al.* [5] employed γ-quasi-clique to find vertices densely connected by edges with similar labels. These methods buffer temporal graphs in a static manner, and have to compute from scratch when new updates arrive. Thus, they cannot process a temporal graph with an endless stream of updates.

To handle streaming updates, Aggarwal *et al.* [2] proposed a probabilistic model to mine dense structural patterns by summarizing a graph stream. Angel *et al.* [3] incrementally computed dense subgraphs by maintaining a small number of sparse subgraphs. Epasto *et al.* [17] maintained a temporal subgraph with near-optimal average degree. Bhattacharya *et al.* [4] maintained a temporal subgraph with the largest average degree. These methods find subgraphs with large density instead of considering how fast a temporal subgraph accumulates density. Since a subgraph can slowly accumulate a large density during an arbitrarily long time, a subgraph with large density is not necessarily a density bursting subgraph. As a result, the methods mentioned above cannot accurately find density bursting subgraphs.

Shin *et al.* [31] proposed a dense subtensor detection method named *DenseAlert*, which can be extended to maintain the top-1 temporal subgraph that has the largest average degree [18] within a time window. However, since a subgraph with a large average degree usually has a low edge density [33], the temporal subgraph maintained by *DenseAlert* usually has a small edge density, and is substantially different from the top-k DBSs detected by our method.

2.3 Maximum Density Segment Problem

Given a sequence of numbers $Q = \langle q_1, \ldots, q_n \rangle$ and a positive integer $L \leq n$, the Maximum Density Segment (MDS) problem [9] is to select a subsequence of Q with length at least L, such that the average sum of the selected numbers is maximized. Clearly, if L = 1, then a trivial solution with the largest element in the array will be returned.

The MDS problem is a fundamental problem that has many linear time solutions. Huang *et al.* [20] solved it in $O(nL)$ time, and prove that the size of an optimal solution is at most $2L - 1$. Lin *et al.* [22] proposed a right-skew method with time complexity $O(n \log L)$. Kim *et al.* [21] gave a slope interpretation on the original MDS problem, and solved it in $O(n)$ time by finding the line segment with the maximum slope. Goldwasser *et al.* [19] applied locally optimal segments to solve it in $O(n)$ time. Chung *et al.* [15] extended Goldwasser's method [19] to a linear online method. Curtis *et al.* [16] proposed an online method based on sliding window.

In our work, we employ MDS methods as a building block. However, as proved in Section 3.3, finding dense burst subgraphs in temporal graphs is an NP-hard problem, which is dramatically different from the polynomial-time solvable MDS problem.

Chapter 3

Problem Definition

In this chapter, we first introduce several essential notions, then formalize the top-k density bursting subgraph finding problem and investigate its computational complexity. Table 3.1 summarizes some frequently used notations.

3.1 Temporal Graph and Temporal Subgraph

In this thesis, we in general consider weighted graphs where each edge carries a weight. A **temporal graph**, denoted by $\mathcal{G}(t_0, t_c) = \langle G(t_0), G(t_1), \ldots, G(t_c)\rangle$, is a sequence of snapshots that arrive at different times between an initial time t_0 and a current time t_c. In a temporal graph, new snapshots may introduce new vertices and edges. However, in most real world scenario, the number of vertices in a temporal graph is not infinite. Therefore, we assume that all snapshots share the same set of vertices V, which is large enough to cover all the vertices in the temporal graph. The changes over time would only happen on edges. We further assume that t_0 is the time when $\mathcal{G}(t_0, t_c)$ is initialized as an empty graph, that is, $G(t_0)$ contains only a set of vertices but no edges.

The **snapshot** that arrive at time $t_h \in \{t_1, \ldots, t_c\}$ is a static graph denoted by $G(t_h) = (V, A(t_h))$, where t_h is the arrival time, and $A(t_h)$ is the affinity matrix that defines the edge weights between vertices.

Denote by $n = |V|$ the number of vertices in V. For each snapshot $G(t_h)$, we represent the affinity matrix $A(t_h)$ by an n-by-n non-negative matrix, where the entry $A_{ij}(t_h)$ at the i-th row and the j-th column of $A(t_h)$ is the edge weight between the i-th vertex $v_i \in V$ and the j-th vertex $v_j \in V$. There is an edge between v_i and v_j if and only if $A_{ij}(t_h) > 0$, and there is no edge connection between vertices in different snapshots.

A **time interval**, denoted by $T = (t_b, t_e] = \{t_{b+1}, t_{b+2}, \ldots, t_e\}$, is the set of the time points between **begin time** t_b and **end time** t_e, excluding t_b. The **duration** of $T = (t_b, t_e]$ is $t_e - t_b$.

An **accumulated graph** during $T = (t_b, t_e]$ is a static graph denoted by $G(t_{b+1}, t_e) = (V, A(t_{b+1}, t_e))$, where $A(t_{b+1}, t_e) = \sum_{h=b+1}^{e} A(t_h)$ is the accumulated affinity matrix. Denote

Table 3.1: Frequently used notations.

Notation	Description
$\mathbf{x} \in \triangle^n$	The indicator vector to represent a set of vertices.
$(\mathbf{x}^*, T^*)$	A Density Bursting Subgraph (DBS) in $\mathcal{G}(t_0, t_c)$.
$(\hat{\mathbf{x}}, \hat{T})$	An indecomposable DBS candidate in $\mathcal{G}(t_0, t_c)$.
t_s	$t_s = prev(prev(t_c))$ is the begin time of $\mathcal{G}(t_s, t_c)$.
$\epsilon_k(t_{c-1})$	The k-th largest burstiness of all ODBSCs in $\mathcal{D}(t_{c-1})$.
$V(t_{b+1}, t_e)$	The set of vertices connected by at least one edge in the accumulated graph $G(t_{b+1}, t_e)$. (see Section 3.1)
$\mathcal{P}_{\mathbf{x}^*}$	$\mathcal{P}_{\mathbf{x}^*} = \{(t_h, s_{\mathbf{x}^*}(t_h)) \mid t_h \in \{t_0, \ldots, t_c\}\}$ the set of points induced by $\mathbf{x}^*$ (see Section 4.1.2).
$\mathbf{u}(v_i) \in \triangle^n$	The indicator vector where only the i-th entry is 1 and the other entries are 0's. (see Section 4.2)

by $A_{ij}(t_{b+1}, t_e)$ the element at the i-th row and the j-th column of $A(t_{b+1}, t_e)$, we write the set of vertices that are connected by at least one edge in $G(t_{b+1}, t_e)$ as $V(t_{b+1}, t_e) = \{v_i \in V \mid \exists A_{ij}(t_{b+1}, t_e) > 0\}$.

Denote by $\mathcal{G}(t_{b+1}, t_e) = \langle G(t_{b+1}), \ldots, G(t_e)\rangle$ the sequence of snapshots that arrive during $T = (t_b, t_e]$. A **temporal subgraph** is a sequence of subgraphs that are induced by a set of weighted vertices $S \subseteq V$ on each of the snapshots in $\mathcal{G}(t_{b+1}, t_e)$. Each vertex v_i in S is assigned a positive weight $\mathbf{x}_i$ that indicates the importance of v_i in S. The weights of all vertices in $V \setminus S$ are set to 0's. In this way, we can induce S by an n-dimensional vector $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\top$, such that $V_{\mathbf{x}} = S = \{v_i \in V \mid \mathbf{x}_i > 0\}$. Following the conventional dense subgraph detection settings [24], we enforce $\mathbf{x}$ to be in the standard simplex, that is $\mathbf{x} \in \triangle^n = \{\mathbf{x} \mid \sum_i \mathbf{x}_i = 1, \mathbf{x}_i \geq 0\}$. For the rest of the thesis, we write a temporal subgraph as a tuple $(\mathbf{x}, T)$.

The **duration** of a temporal subgraph $(\mathbf{x}, T)$ is exactly the duration of T. For any vertex $v_i \in V$, if $v_i \in V_{\mathbf{x}}$, we say v_i is **contained** by $(\mathbf{x}, T)$ and write $v_i \in (\mathbf{x}, T)$.

Next, we introduce the notion of density bursting subgraph.

3.2 Density Bursting Subgraph

For a temporal subgraph $(\mathbf{x}, T)$ and a time $t_h \in T$, denote by $G_{\mathbf{x}}(t_h)$ the subgraph induced by $V_{\mathbf{x}}$ from the snapshot $G(t_h)$. We measure the density of $G_{\mathbf{x}}(t_h)$ by the following **cohesiveness** [29].

$$q_{\mathbf{x}}(t_h) = \mathbf{x}^\top A(t_h)\mathbf{x} \tag{3.1}$$

The **burstiness** of $(\mathbf{x}, T)$ measures how fast it accumulates cohesiveness during time interval $T = (t_b, t_e]$, that is,

$$g(\mathbf{x}, T) = \frac{\sum_{h=b+1}^{e} q_{\mathbf{x}}(t_h)}{t_e - t_b} = \frac{\mathbf{x}^\top A(t_{b+1}, t_e)\mathbf{x}}{t_e - t_b} \tag{3.2}$$

Next, we define **Density Bursting Subgraph** (DBS).

Definition 3.1. *Given a temporal graph $\mathcal{G}(t_0, t_c)$ and a minimum duration threshold θ, a density bursting subgraph, denoted by $(\mathbf{x}^*, T^*)$ where $T^* = (t_{b^*}, t_{e^*}]$, is a temporal subgraph in $\mathcal{G}(t_0, t_c)$, such that*

1. *$\mathbf{x}^* \in \triangle^n$ is a local maximum point of $g(\mathbf{x}, T^*)$;*
2. *T^* is a global maximum point of $g(\mathbf{x}^*, T)$; and*
3. *$t_{e^*} - t_{b^*} \geq \theta$.*

Denote by $G_{\mathbf{x}^*}(t_{b^*+1}, t_{e^*})$ the subgraph induced by $V_{\mathbf{x}^*}$ from the accumulated graph $G(t_{b^*+1}, t_{e^*})$. As illustrated later in Section 4.1.1, condition (1) of Definition 3.1 requires $G_{\mathbf{x}^*}(t_{b^*+1}, t_{e^*})$ to be a dense subgraph in $G(t_{b^*+1}, t_{e^*})$ [29, 25, 8].

Given $\mathbf{x}^*$, condition (2) of Definition 3.1 requires T^* to be the optimal time interval, such that the temporal subgraph $(\mathbf{x}^*, T^*)$ achieves the largest burstiness.

In condition (3) of Definition 3.1, the minimum duration threshold θ effectively prevents trivial temporal subgraphs that have extremely small duration. According to Equation 3.2, a trivial temporal subgraph with extremely small duration can have an extremely large burstiness even if its cohesiveness is very small. More often than not, such trivial temporal subgraphs consist of a single edge in snapshots with very small duration, which are not of much interest in real world applications.

Interestingly, by changing the meaning of $A(t_h)$, the DBS defined in Definition 3.1 can be easily extended to model the following interesting patterns in temporal graphs.

1. When $A(t_h)$ represents the absolute difference matrix between the affinity matrices of neighboring snapshots, a DBS $(\mathbf{x}, T)$ models the subgraph $\mathbf{x}$ whose absolute internal edge weights changes the most during time interval T.
2. When each entry of $A(t_h)$ represents the relative change between the corresponding edge weights of neighboring snapshots, a DBS $(\mathbf{x}, T)$ models the subgraph $\mathbf{x}$ whose internal edge weights have the largest relative change during time interval T.

3.3 Top-k DBS Finding Problem

Now we introduce the Top-k DBS Finding problem and prove that it is NP-hard.

Definition 3.2. *Given a temporal graph $\mathcal{G}(t_0, t_c)$, a minimum duration threshold θ, and a positive integer k, **the problem of Top-k DBS Finding** (TDF for short) is to compute the set of DBSs in $\mathcal{G}(t_0, t_c)$ that have the top-k largest burstiness.*

Theorem 3.1. *The TDF problem is NP-hard.*

Proof. We only need to prove that the Top-1 (i.e., $k = 1$) DBS Finding problem is NP-hard. Consider an arbitrary unweighted and undirected graph G, whose affinity matrix is A. The entries of A are either 0 or 1. We create an instance of the TDF problem by constructing a temporal graph $\mathcal{G}(t_0, t_c) = \langle G(t_0), \ldots, G(t_c) \rangle$, such that $t_c - t_0 = \theta = 1$, $A(t_c) = A$ and the affinity matrices of all the snapshots in $\mathcal{G}(t_0, t_{c-1})$ are matrices of all 0's. Since $t_c - t_0 = \theta = 1$, it follows Definition 3.1 that $T^* = (t_0, t_c]$ is the only optimal time interval for any DBS in $\mathcal{G}(t_0, t_c)$. Thus, the problem of maximizing $g(\mathbf{x}, T^*)$ s.t. $\mathbf{x} \in \triangle^n$ can be reduced to the Top-1 DBS Finding problem. Since $g(\mathbf{x}, T^*) = \mathbf{x}^\top A\mathbf{x}$, we are actually maximizing $\mathbf{x}^\top A\mathbf{x}$ s.t. $\mathbf{x} \in \triangle^n$, which is NP-hard [28]. □

Given the definition and the hardness of the TDF problem, in the next chapter, we will first introduce how to find a single DBS by modeling the TDF problem as a mixed integer programming problem. Then we will propose a static baseline method, namely *SlideWin*, to find good solutions for the TDF problem.

Chapter 4

Finding a Single DBS and a Static Baseline Method

In this chapter, we first introduce how to find a single DBS. Then we present a static sliding window method to find a good solution to the TDF problem. At last, we present a challenge in designing online methods to solve the TDF problem.

4.1 Finding a Single DBS

In this section, we introduce how to find a single DBS by solving a Mixed Integer Programming (MIP) problem.

According to Definition 3.1, a DBS $(\mathbf{x}^*, T^*)$ is a local maximum point of the following MIP problem.

$$\begin{aligned} &\underset{(\mathbf{x},T)}{\operatorname{argmax}}\, g(\mathbf{x}, T) \\ &s.t.\, \mathbf{x} \in \triangle^n, T = (t_b, t_e], t_e - t_b \geq \theta \end{aligned} \tag{4.1}$$

To find a single DBS, we find a local maximum point of the MIP problem by iteratively updating $\mathbf{x}$ and T to monotonously increase $g(\mathbf{x}, T)$. Next, we illustrate how to update $\mathbf{x}$ and T by solving a Constrained Quadratic Programming (CQP) problem [29] and a Maximum Density Segment (MDS) problem [9], respectively.

4.1.1 The CQP problem

Given a time interval $T = (t_b, t_e]$ such that $t_e - t_b \geq \theta$, by plugging Equation 3.2 into Equation 4.1 and omitting the constant factor $t_e - t_b$, we transform the MIP problem into the following **CQP problem** [29].

$$\begin{aligned} &\underset{\mathbf{x}}{\operatorname{argmax}}\, \mathbf{x}^\top A(t_{b+1}, t_e)\mathbf{x} \\ &s.t.\, \mathbf{x} \in \triangle^n \end{aligned} \tag{4.2}$$

Algorithm 1: $FindDBS(\mathcal{G}(t_0, t_c), \theta, (\mathbf{x}, T))$

Input: $\mathcal{G}(t_0, t_c)$, θ, $(\mathbf{x}, T)$ an initial temporal subgraph.
Output: $(\mathbf{x}^*, T^*)$ a DBS in $\mathcal{G}(t_0, t_c)$.

1: **repeat**
2: Solve the CQP problem by IID [8, 11]: $\mathbf{x} \leftarrow \underset{\mathbf{x}}{\operatorname{argmax}}\, \mathbf{x}^\top A(t_{b+1}, t_e)\mathbf{x}, s.t.\, \mathbf{x} \in \triangle^n$.
3: Solve the MDS problem by MDSD [15]: $T \leftarrow \underset{T}{\operatorname{argmax}}\, \frac{s_{\mathbf{x}}(t_e) - s_{\mathbf{x}}(t_b)}{t_e - t_b}, s.t.\, t_e - t_b \geq \theta$.
4: **until** The value of $g(\mathbf{x}, T)$ does not increase.
5: **return** $(\mathbf{x}^*, T^*) = (\mathbf{x}, T)$.

According to the previous works of dense subgraph detection [29, 25, 8], a local maximum point $\mathbf{x}^*$ of the CQP problem induces a dense subgraph $G_{\mathbf{x}^*}(t_{b+1}, t_e)$ in $G(t_{b+1}, t_e)$.

We can efficiently find a local maximum point of the CQP problem by the Infection Immunization Dynamics (IID) method [8]. The time complexity of IID is $O(\lambda n)$, where λ is the number of iterations of IID, and n is the volume of V. Since $\mathbf{x} \in \triangle^n$ is usually very sparse, we can efficiently solve the CQP problem using a small sub-matrix of $A(t_{b+1}, t_e)$ [11].

4.1.2 The MDS problem

Given $\mathbf{x} \in \triangle^n$, denote by $s_{\mathbf{x}}(t_r) = \sum_{h=0}^{r} q_{\mathbf{x}}(t_h)$ the sum of cohesiveness from t_0 to t_r. We rewrite the burstiness in Equation 3.2 as $g(\mathbf{x}, T) = \frac{s_{\mathbf{x}}(t_e) - s_{\mathbf{x}}(t_b)}{t_e - t_b}$, which is exactly the **slope** between two **points**, $(t_b, s_{\mathbf{x}}(t_b))$ and $(t_e, s_{\mathbf{x}}(t_e))$, in the 2-dimensoinal Cartesian coordinate system.

Using the above slope representation of burstiness, we transform the MIP problem into the following **MDS problem** [9].

$$\begin{aligned} &\underset{T}{\operatorname{argmax}}\, \frac{s_{\mathbf{x}}(t_e) - s_{\mathbf{x}}(t_b)}{t_e - t_b} \\ &s.t.\, t_e - t_b \geq \theta \end{aligned} \tag{4.3}$$

Denote by $\mathcal{P}_{\mathbf{x}} = \{(t_h, s_{\mathbf{x}}(t_h)) \mid t_h \in \{t_0, \ldots, t_c\}\}$ the set of points induced by $\mathbf{x}$. Solving the MDS problem is equivalent to finding $T^* = (t_{b^*}, t_{e^*}]$ such that $t_{e^*} - t_{b^*} \geq \theta$, and the points $(t_{b^*}, s_{\mathbf{x}}(t_{b^*}))$ and $(t_{e^*}, s_{\mathbf{x}}(t_{e^*}))$ in $\mathcal{P}_{\mathbf{x}}$ achieve the global maximum slope.

We can efficiently compute T^* by the MDS Detection (MDSD) method proposed by Chung *et al.* [15]. The time complexity of MDSD is $O(c + 1)$, where $c + 1$ the number of snapshots in $\mathcal{G}(t_0, t_c)$.

We summarize the $FindDBS$ method in Algorithm 1, which starts from an initial temporal subgraph $(\mathbf{x}, T)$ and finds a DBS by iteratively solving the CQP problem and the MDS problem.

Algorithm 2: $SlideWin(\mathcal{G}(t_0, t_c), \theta, k)$

Input: $\mathcal{G}(t_0, t_c)$, θ, k a positive integer.
Output: $\mathcal{K}(t_c)$ the set of top-k DBSs in $\mathcal{G}(t_0, t_c)$.

```
1: D ← ∅.
2: for each t_e ∈ {t_1, ..., t_c} do
3:    for each t_b ∈ {t_0, ..., t_{e-1}} such that t_e − t_b ≥ θ do
4:       for each v_i ∈ V(t_{b+1}, t_e) do
5:          (x*, T*) ← FindDBS(G(t_0, t_c), θ, (u(v_i), (t_b, t_e])).
6:          D ← D ∪ (x*, T*).
7:       end for
8:    end for
9: end for
10: K(t_c) ← TOPK(D).
11: return K(t_c).
```

4.2 A Static Baseline and a Challenge for Online Solution

The TDF problem requires to find multiple local maximum points of the non-concave MIP problem in Equation 4.1.

Normally, to find a set of good solutions to the non-concave MIP problem, we first find multiple DBSs by running $FindDBS$ multiple times with different initializations, then return the set of DBSs that have the top-k largest burstiness. Following this idea, we propose the static baseline method $SlideWin$ in Algorithm 2, where $\mathcal{D}$ stores all detected DBSs, and $TOPK(\mathcal{D})$ returns the top-k DBSs in $\mathcal{D}$.

To obtain as many different initializations as possible, we initialize T by every time interval $(t_b, t_e]$ such that $t_e - t_b \geq \theta$. We initialize $\mathbf{x}$ in the same way as IID [8]. That is, for each $v_i \in V(t_{b+1}, t_e)$, we set $\mathbf{x} = \mathbf{u}(v_i)$, where $\mathbf{u}(v_i) \in \triangle^n$ is an n-dimensional vector such that only the i-th entry is 1 and all the other entries are 0's. Initializing $\mathbf{x}$ in this way keeps the size of $V_{\mathbf{x}}$ small and improves the efficiency of IID [8].

The major drawback of $SlideWin$ is the requirement to buffer all the snapshots in $\mathcal{G}(t_0, t_c)$. Since $\mathcal{G}(t_0, t_c)$ may be a long or even endless stream of snapshots, buffering all the snapshots in $\mathcal{G}(t_0, t_c)$ is infeasible. As a result, we are much more interested in an online method that maintains the top-k DBSs in real time using a limited buffer of snapshots. However, designing an online method for the TDF problem is still difficult due to the following challenge.

For an online method, the number of snapshots to buffer is lower bounded by the maximum duration of DBSs, which, as shown in Theorem 4.1, can be as large as $t_c - t_0$ in a temporal graph $\mathcal{G}(t_0, t_c)$.

Theorem 4.1. *There exists a temporal graph $\mathcal{G}(t_0, t_c)$ such that the maximum duration of a DBS in $\mathcal{G}(t_0, t_c)$ is $t_c - t_0$.*

Proof. We prove by constructing a temporal graph $\mathcal{G}(t_0, t_c)$ such that $t_c - t_0 \geq \theta$, every snapshot in $\mathcal{G}(t_1, t_c)$ has exactly the same non-empty affinity matrix, and $\forall i \in \{1, \ldots, c-1\}$, $t_i - t_{i-1} = t_{i+1} - t_i$. By Definition 3.1, for any local maximum point $\mathbf{x}^* \in \triangle^n$ of $q_{\mathbf{x}}(t_1)$, $(\mathbf{x}^*, (t_0, t_c])$ is a DBS in $\mathcal{G}(t_0, t_c)$. The theorem follows. □

Theorem 4.1 presents a big challenge in designing online methods for the TDF problem. To find a DBS with duration $t_c - t_0$, we have to buffer all the snapshots in $\mathcal{G}(t_0, t_c)$, which, unfortunately, may be a long or even endless stream of snapshots.

In the next chapter, we will first introduce the decomposition property of DBSs, which opens the door to efficient online DBS detection. Then we will demonstrate our online Top-k DBS detection method.

Chapter 5

DBS Decomposition and Online Top-k DBS Detection

In this chapter, we first introduce an important decomposition property of DBSs, then present our online Top-k DBS detection method.

5.1 DBS Decomposition

In this section, we present a critical observation: a long DBS can be easily decomposed into a set of shorter DBSs that have the same or larger burstiness. We also show that the durations of the shorter DBSs are upper bounded, which makes it possible to design a highly efficient online algorithm for the TDF problem.

Given a DBS $(\mathbf{x}^*, (t_{b^*}, t_{e^*}])$, if $\exists t_h \in (t_{b^*}, t_{e^*}]$ such that $t_{b^*} + \theta \leq t_h \leq t_{e^*} - \theta$, then $(\mathbf{x}^*, (t_{b^*}, t_{e^*}])$ is said to be **decomposable** at time t_h. Otherwise, $(\mathbf{x}^*, (t_{b^*}, t_{e^*}])$ is **indecomposable**.

If a DBS $(\mathbf{x}^*, (t_{b^*}, t_{e^*}])$ is decomposable at time t_h, we say the two temporal subgraphs $(\mathbf{x}^*, (t_{b^*}, t_h])$ and $(\mathbf{x}^*, (t_h, t_{e^*}])$ are the **components** of $(\mathbf{x}^*, (t_{b^*}, t_{e^*}])$ at time t_h.

Denote by $prev(t_e) = \max\{t_h \mid t_h \leq t_e - \theta\}$ the time of the last snapshot that arrives no later than $t_e - \theta$, by $next(t_b) = \min\{t_h \mid t_h \geq t_b + \theta\}$ the time of the first snapshot that arrives no earlier than $t_b + \theta$, and by $\mathcal{T}(t_b, t_e) = \{t_h \mid next(t_b) \leq t_h \leq prev(t_e)\}$ the set of times between $next(t_b)$ and $prev(t_e)$. Clearly, $(\mathbf{x}^*, (t_{b^*}, t_{e^*}])$ is indecomposable if and only if $\mathcal{T}(t_{b^*}, t_{e^*}) = \emptyset$. If $\mathcal{T}(t_{b^*}, t_{e^*}) \neq \emptyset$, then $(\mathbf{x}^*, (t_{b^*}, t_{e^*}])$ is decomposable at any time $t_h \in \mathcal{T}(t_{b^*}, t_{e^*})$.

Next, we present the important observation that a decomposable DBS has exactly the same burstiness as its components.

Theorem 5.1. *For a decomposable DBS $(\mathbf{x}^*, T^*)$ where $T^* = (t_{b^*}, t_{e^*}]$, any component of $(\mathbf{x}^*, T^*)$ at time $t_h \in \mathcal{T}(t_{b^*}, t_{e^*})$ has the same burstiness as $(\mathbf{x}^*, T^*)$.*

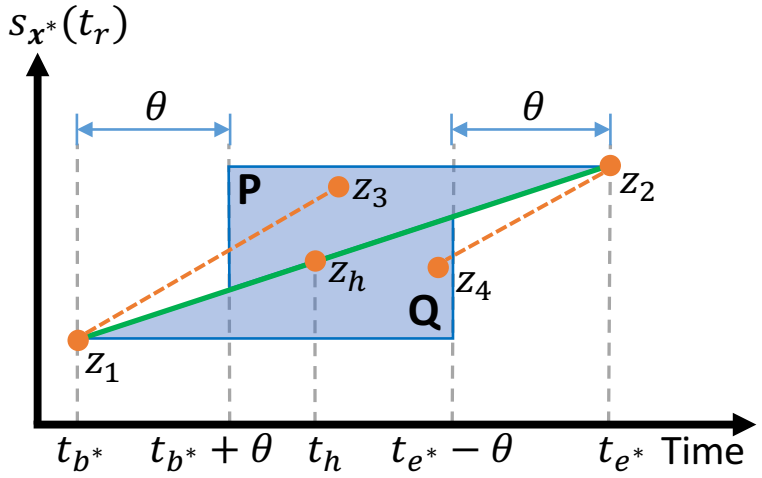


Figure 5.1: The slope representation of a decomposable DBS $(\mathbf{x}^*, T^*)$, where $T^* = (t_{b^*}, t_{e^*}]$ and $\mathcal{T}(t_{b^*}, t_{e^*}) \neq \emptyset$. P and Q are the two blue triangle regions, respectively.

Proof. Consider the slope representation of $(\mathbf{x}^*, T^*)$ in Figure 5.1. We first prove by contradiction that the triangle regions P and Q do not contain any point in $\mathcal{P}_{\mathbf{x}^*}$. Assume P contains $z_3 \in \mathcal{P}_{\mathbf{x}^*}$. Then, the slope between z_1 and z_3 is larger than the slope between z_1 and z_2. Thus, T^* is not the global maximum ponit of the MDS problem. This contradicts with the condition that $(\mathbf{x}^*, T^*)$ is a DBS. Similarly, Q does not contain any point $z_4 \in \mathcal{P}_{\mathbf{x}^*}$.

Second, we prove that $(\mathbf{x}^*, T^*)$ and its components have the same burstiness. Since $(\mathbf{x}^*, T^*)$ is decomposable, we know $\mathcal{T}(t_{b^*}, t_{e^*}) \neq \emptyset$ and $(\mathbf{x}^*, T^*)$ is decomposable at any time $t_h \in \mathcal{T}(t_{b^*}, t_{e^*})$. Recall that $z_h \in \mathcal{P}_{\mathbf{x}^*}$ cannot be contained by P or Q, since $\forall t_h \in \mathcal{T}(t_{b^*}, t_{e^*})$, $t_{b^*} + \theta \leq t_h \leq t_{e^*} - \theta$, z_h must reside on the segment between z_1 and z_2. By the slope representation of burstiness, $g(\mathbf{x}^*, (t_{b^*}, t_h]) = g(\mathbf{x}^*, (t_h, t_{e^*}]) = g(\mathbf{x}^*, T^*)$. □

According to Theorem 5.1, a decomposable DBS $(\mathbf{x}^*, T^*)$ can be decomposed into two components, $(\mathbf{x}^*, (t_{b^*}, t_h])$ and $(\mathbf{x}^*, (t_h, t_{e^*}])$, that have the same burstiness as $(\mathbf{x}^*, T^*)$. However, since $\mathbf{x}^*$ may not be a local maximum point of $g(\mathbf{x}, (t_{b^*}, t_h])$ or $g(\mathbf{x}, (t_h, t_{e^*}])$, the components of $(\mathbf{x}^*, T^*)$ may not be DBSs.

If a component $(\mathbf{x}', T')$ of a decomposable DBS $(\mathbf{x}^*, T^*)$ is not a DBS, we can further increase the burstiness of $(\mathbf{x}', T')$ by feeding $(\mathbf{x}', T')$ as the initial temporal subgraph into *FindDBS*. The output of *FindDBS* is a new DBS $(\mathbf{x}^*_{new}, T^*_{new})$ such that $g(\mathbf{x}^*_{new}, T^*_{new}) > g(\mathbf{x}', T') = g(\mathbf{x}^*, T^*)$. If $(\mathbf{x}^*_{new}, T^*_{new})$ is also decomposable, we keep decomposing it and updating its components by *FindDBS*. Eventually, we can decompose a decomposable DBS $(\mathbf{x}^*, T^*)$ into a set of indecomposable DBSs with the same or even larger burstiness.

Interestingly, the duration of any indecomposable DBS $(\mathbf{x}^*, (t_{b^*}, t_{e^*}])$ is upper bounded by $t_{e^*} - prev(prev(t_{e^*}))$, because $\mathcal{T}(t_{b^*}, t_{e^*}) = \emptyset$ if and only if $t_{b^*} > prev(prev(t_{e^*}))$. This makes it possible to detect indecomposable DBSs in an online manner without buffering all snapshots. Another reason to find indecomposable DBSs is that most DBSs with large

burstiness are indecomposable in practice. We will analyze this phenomenon using the experiments in Table 6.2 and Section 6.1.

5.2 Online Top-k DBS Detection

Enabled by the observations in Section 5.1, in this chapter, we define the Online Top-k DBS Finding (OTDF) problem and develop an efficient online DBS detection method named $TopkDBSOL$.

5.2.1 The OTDF problem

Denote by $\mathcal{M}(t_c)$ the set of indecomposable DBSs in $\mathcal{G}(t_0, t_c)$. We define the OTDF problem as follows.

Definition 5.1. *Given $\mathcal{M}(t_{c-1})$ and a sequence of buffered snapshots $\mathcal{G}(t_s, t_c)$, **the problem of Online Top-k Density Bursting Subgraph Finding** (OTDF for short) is to compute the top-k indecomposable DBSs in $\mathcal{M}(t_c)$.*

The OTDF problem is NP-hard following the same reduction in the proof of Theorem 3.1.

Recall that the duration of any indecomposable DBS ending at time t_c is upper bounded by $t_c - prev(prev(t_c))$, we set

$$t_s = prev(prev(t_c)),$$

so that any indecomposable DBS in $\mathcal{M}(t_c) \setminus \mathcal{M}(t_{c-1})$ is an indecomposable DBS in $\mathcal{G}(t_s, t_c)$.

However, an indecomposable DBS $(\mathbf{x}^*, T^*)$ in $\mathcal{G}(t_s, t_c)$ may not be an indecomposable DBS in $\mathcal{M}(t_c) \setminus \mathcal{M}(t_{c-1})$, because, without the snapshots in $\mathcal{G}(t_0, t_{s-1})$, we cannot verify whether or not T^* is the global maximum point of $g(\mathbf{x}^*, T)$ in $\mathcal{G}(t_0, t_c)$. As a result, we cannot compute $\mathcal{M}(t_c)$ directly.

To tackle this problem, we compute a superset of $\mathcal{M}(t_c)$ by finding the set of indecomposable DBS candidates. Here, an **indecomposable DBS candidate** is a temporal subgraph $(\hat{\mathbf{x}}, \hat{T})$ that has a begin time $t_{\hat{b}}$, and is an indecomposable DBS in $\mathcal{G}(t_{\hat{b}}, t_c)$. Since any indecomposable DBS is an indecomposable DBS candidate, the set of indecomposable DBS candidates, denoted by $\mathcal{D}(t_c) = \{(\hat{\mathbf{x}}, \hat{T}) \mid \hat{T} = (t_{\hat{b}}, t_{\hat{e}}] \wedge t_{\hat{e}} \leq t_c\}$, is a super set of $\mathcal{M}(t_c)$.

As illustrated in the rest of this section, given $\mathcal{D}(t_{c-1})$ and $\mathcal{G}(t_s, t_c)$, we can efficiently find a good solution to the OTDF problem in three steps: (1) find **new indecomposable DBS candidates** (**NDBSC**) in $\mathcal{D}(t_c) \setminus \mathcal{D}(t_{c-1})$; (2) update **old indecomposable DBS candidates** (**ODBSC**) in $\mathcal{D}(t_{c-1})$; and (3) compute the top-k indecomposable DBS candidates in $\mathcal{D}(t_c)$ as the final result.

5.2.2 Finding the Set of NDBSCs

Algorithm 3: $FindNDBSC(\mathcal{G}(t_s, t_c), \theta, \mathcal{D}(t_{c-1}), k)$

Input: $\mathcal{G}(t_s, t_c), \theta, \mathcal{D}(t_{c-1}), k$.
Output: $\mathcal{N}(t_c)$ the set of NDBSCs.

$\mathcal{N}(t_c) \leftarrow \emptyset$, and compute $\epsilon_k(t_{c-1})$.
for each $t_b \in \{t_s, \ldots, t_{c-1}\}$ such that $t_c - t_b \geq \theta$ **do**
  $T \leftarrow (t_b, t_c]$.
  for each $v_i \in V(t_{b+1}, t_c)$ **do**
    if $\alpha(v_i, T) \geq \epsilon_k(t_{c-1})$ **then**
      $\mathbf{x} \leftarrow \mathbf{o}(v_i, T)$.
      $(\hat{\mathbf{x}}, \hat{T}) \leftarrow FindDBS(\mathcal{G}(t_s, t_c), \theta, (\mathbf{x}, T))$.
      if $(\hat{\mathbf{x}}, \hat{T}) \notin \mathcal{D}(t_{c-1})$ **then**
        $\mathcal{N}(t_c) \leftarrow \mathcal{N}(t_c) \cup (\hat{\mathbf{x}}, \hat{T})$.
      end if
    end if
  end for
end for
return $\mathcal{N}(t_c)$.

According to the definition of NDBSC, any indecomposable DBS in $\mathcal{G}(t_s, t_c)$ is an NDBSC if it is not contained in $\mathcal{D}(t_{c-1})$. A straightforward way to find indecomposable DBSs in $\mathcal{G}(t_s, t_c)$ is to call $SlideWin$. However, $SlideWin$ is inefficient because it calls $FindDBS$ for every vertex $v_i \in V(t_{b+1}, t_e)$.

Interestingly, we observed that a large proportion of the vertices in $V(t_{b+1}, t_e)$ are not contained in any top-k indecomposable DBS. Inspired by this observation, we first derive an upper bound for the burstiness of any indecomposable DBS that contains a vertex v_i, then we propose a **smart initialization** heuristic, which applies the upper bound to filter out a large proportion of vertices that are not contained in any top-k indecomposable DBS. This significantly reduces the number of calls of $FindDBS$, and achieves a speedup of two orders of magnitudes in our experiments.

Next, we show that $\alpha(v_i, T) = \frac{\max_j A_{ij}(t_{b+1}, t_e)}{t_e - t_b}$ is an upper bound of the burstiness of any indecomposable DBS that contains a vertex v_i during a time interval $T = (t_b, t_e]$.

Theorem 5.2. *For any indecomposable DBS* $(\mathbf{x}^*, T^*)$*, if* $v_i \in (\mathbf{x}^*, T^*)$*, then* $g(\mathbf{x}^*, T^*) \leq \alpha(v_i, T^*)$.

Proof. Since $(\mathbf{x}^*, T^*)$ is an indecomposable DBS, $\mathbf{x}^*$ is a local maximum point of $\mathbf{x}^\top A(t_{b^*+1}, t_{e^*})\mathbf{x}$. Since $v_i \in (\mathbf{x}^*, T^*)$, $v_i \in V_{\mathbf{x}^*}$. According to Liu *et al.* [25], since $v_i \in V_{\mathbf{x}^*}$ and $\mathbf{x}^*$ is a local maximum point, $(\mathbf{x}^*)^\top A(t_{b^*+1}, t_{e^*})\mathbf{x}^* = \sum_j \mathbf{x}^*_j A_{ij}(t_{b^*+1}, t_{e^*})$. Since $\mathbf{x}^* \in \triangle^n$, $\sum_j \mathbf{x}^*_j A_{ij}(t_{b^*+1}, t_{e^*}) \leq \max_j A_{ij}(t_{b^*+1}, t_{e^*})$. In sum, $g(\mathbf{x}^*, T^*) = \frac{(\mathbf{x}^*)^\top A(t_{b^*+1}, t_{e^*})\mathbf{x}^*}{t_{e^*} - t_{b^*}} = \frac{\sum_j \mathbf{x}^*_j A_{ij}(t_{b^*+1}, t_{e^*})}{t_{e^*} - t_{b^*}} \leq \alpha(v_i, T^*)$. □

Now, we introduce the smart initialization method.

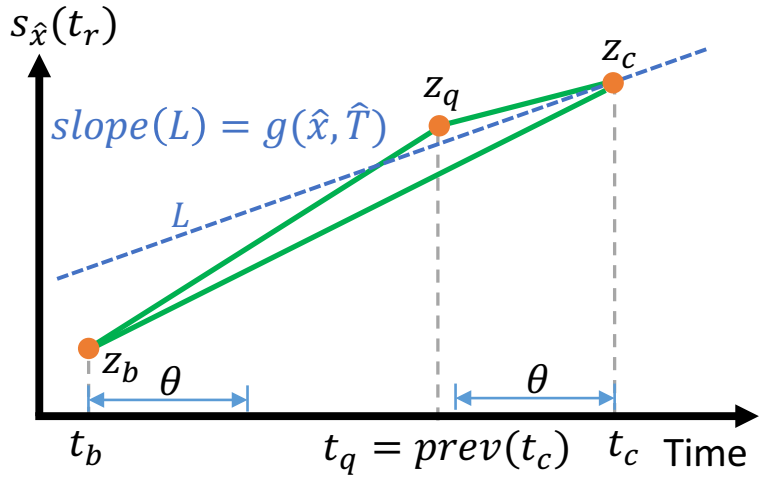


Figure 5.2: The slope representation to prove Theorem 5.3. L is an auxiliary line that crosses z_c. The slope of L is $g(\hat{x}, \hat{T})$.

Denote by $\epsilon_k(t_{c-1})$ the k-th largest burstiness of all ODBSCs in $\mathcal{D}(t_{c-1})$. By Theorem 5.2, for any time interval T, if $\alpha(v_i, T) < \epsilon_k(t_{c-1})$, then v_i is not contained by any indecomposable DBS $(\mathbf{x}^*, T^*)$ such that $T^* = T$ and $g(\mathbf{x}^*, T^*) \geq \epsilon_k(t_{c-1})$.

Since we are only interested in the top-k indecomposable DBSs whose burstiness is larger than $\epsilon_k(t_{c-1})$, we skip each vertex v_i such that $\alpha(v_i, T) < \epsilon_k(t_{c-1})$. If $\alpha(v_i, T) \geq \epsilon_k(t_{c-1})$, we initialize $FindDBS$ by $\mathbf{x} = \mathbf{o}(v_i, T) = 0.5\mathbf{u}(v_i) + 0.5\mathbf{u}(v_j)$, where $v_j = \text{argmax}_{v_j} A_{ij}(t_{b+1}, t_e)$ is the nearest neighbour of v_i in $G(t_{b+1}, t_e)$. As to be shown in Theorem 5.4, by setting $\mathbf{x} = \mathbf{o}(v_i, T)$, we achieve a 2-approximation of the top-1 indecomposable DBS.

Algorithm 3 summarizes $FindNDBSC$, which efficiently finds a set of NDBSCs. The smart initialization is performed in steps 5-6.

5.2.3 Updating the Set of ODBSCs

When a new snapshot arrives at time t_c, the ODBSCs in $\mathcal{D}(t_{c-1})$ need to be updated because a ODBSC in $\mathcal{D}(t_{c-1})$ may not be necessarily an indecomposable DBS candidate in $\mathcal{D}(t_c)$.

Denote by $(\hat{\mathbf{x}}, \hat{T})$, $\hat{T} = (t_{\hat{b}}, t_{\hat{e}}]$ an ODBSC in $\mathcal{D}(t_{c-1})$. If $t_{\hat{b}} \geq t_s$, we can directly update $(\hat{\mathbf{x}}, \hat{T})$ by calling $FindDBS(\mathcal{G}(t_s, t_c), \theta, (\hat{\mathbf{x}}, \hat{T}))$. However, when $t_{\hat{b}} < t_s$, we cannot update $(\hat{\mathbf{x}}, \hat{T})$ by calling $FindDBS(\mathcal{G}(t_s, t_c), \theta, (\hat{\mathbf{x}}, \hat{T}))$, because the snapshots before time t_s are not buffered in $\mathcal{G}(t_s, t_c)$. For such ODBSCs, we show in Theorem 5.3 a necessary and sufficient condition to verify whether it is contained in $\mathcal{D}(t_c)$.

Theorem 5.3. *For any ODBSC $(\hat{\mathbf{x}}, \hat{T}) \in \mathcal{D}(t_{c-1})$ where $\hat{T} = (t_{\hat{b}}, t_{\hat{e}}]$ and $t_{\hat{b}} < t_s$, $(\hat{\mathbf{x}}, \hat{T}) \in \mathcal{D}(t_c)$ if and only if $\nexists t_h \in (t_s, prev(t_c)]$ such that $g(\hat{\mathbf{x}}, (t_h, t_c]) > g(\hat{\mathbf{x}}, \hat{T})$.*

Proof. (Direction only-if) Suppose $\exists t_h \in (t_s, prev(t_c)]$ such that $g(\hat{\mathbf{x}}, (t_h, t_c]) > g(\hat{\mathbf{x}}, \hat{T})$. Since $t_{\hat{b}} < t_s < t_h$, $(\hat{\mathbf{x}}, \hat{T})$ is not an indecomposable DBS in $\mathcal{G}(t_{\hat{b}}, t_c)$. Thus, $(\hat{\mathbf{x}}, \hat{T}) \notin \mathcal{D}(t_c)$.

Algorithm 4: $UpdateODBSC(\mathcal{G}(t_s, t_c), \theta, \mathcal{D}(t_{c-1}))$

Input: $\mathcal{G}(t_s, t_c)$, θ, $\mathcal{D}(t_{c-1})$.
Output: $\mathcal{U}(t_c)$ the set of updated ODBSCs.

$\mathcal{U}(t_c) \leftarrow \emptyset$.
for each $(\hat{\mathbf{x}}, \hat{T}) \in \mathcal{D}(t_{c-1})$ **do**
  if $t_{\hat{b}} \geq t_s$ **then**
    $\mathcal{U}(t_c) \leftarrow \mathcal{U}(t_c) \cup FindDBS(\mathcal{G}(t_s, t_c), \theta, (\hat{\mathbf{x}}, \hat{T}))$.
  else if $\exists t_h \in (t_s, prev(t_c)] : g(\hat{\mathbf{x}}, (t_h, t_c]) > g(\hat{\mathbf{x}}, \hat{T})$ **then**
    $\mathcal{U}(t_c) \leftarrow \mathcal{U}(t_c) \cup FindDBS(\mathcal{G}(t_s, t_c), \theta, (\hat{\mathbf{x}}, (t_h, t_c]))$.
  else
    $\mathcal{U}(t_c) \leftarrow \mathcal{U}(t_c) \cup (\hat{\mathbf{x}}, \hat{T})$.
  end if
end for
return $\mathcal{U}(t_c)$.

Algorithm 5: $TopkDBSOL(\mathcal{G}(t_s, t_c), \theta, \mathcal{D}(t_{c-1}), k)$

Input: $\mathcal{G}(t_s, t_c)$, θ, $\mathcal{D}(t_{c-1})$, k.
Output: $\mathcal{D}(t_c)$ and $TOPK(\mathcal{D}(t_c))$.

$\mathcal{N}(t_c) \leftarrow FindNDBSC(\mathcal{G}(t_s, t_c), \theta, \mathcal{D}(t_{c-1}), k)$.
$\mathcal{U}(t_c) \leftarrow UpdateODBSC(\mathcal{G}(t_s, t_c), \theta, \mathcal{D}(t_{c-1}))$.
$\mathcal{D}(t_c) \leftarrow \mathcal{N}(t_c) \cup \mathcal{U}(t_c)$.
return $\mathcal{D}(t_c)$ and $TOPK(\mathcal{D}(t_c))$.

(Direction if) Suppose $\exists t_b \in (t_{\hat{b}}, t_s]$ such that $g(\hat{\mathbf{x}}, (t_b, t_c]) > g(\hat{\mathbf{x}}, \hat{T})$. Then, as shown in Figure 5.2, z_b is under L. Since $\nexists t_h \in (t_s, prev(t_c)]$ such that $g(\hat{\mathbf{x}}, (t_h, t_c]) > g(\hat{\mathbf{x}}, \hat{T})$, hence for $t_q = prev(t_c)$ in Figure 5.2, z_q is above or on L. This means that the slope of the segment (z_b, z_q) is larger than the slope of L, thus $g(\hat{\mathbf{x}}, (t_b, t_q]) > g(\hat{\mathbf{x}}, \hat{T})$. Since $t_{\hat{b}} < t_b$, $t_q \leq t_{c-1}$ and $t_q - t_b \geq t_q - t_s \geq \theta$, it follows $g(\hat{\mathbf{x}}, (t_b, t_q]) > g(\hat{\mathbf{x}}, \hat{T})$ that $(\hat{\mathbf{x}}, \hat{T}) \notin \mathcal{D}(t_{c-1})$. This contradicts with the condition $(\hat{\mathbf{x}}, \hat{T}) \in \mathcal{D}(t_{c-1})$. Therefore, $\nexists t_b \in (t_{\hat{b}}, t_s]$ such that $g(\hat{\mathbf{x}}, (t_b, t_c]) > g(\hat{\mathbf{x}}, \hat{T})$.

Since $\nexists t_h \in (t_s, prev(t_c)]$ such that $g(\hat{\mathbf{x}}, (t_h, t_c]) > g(\hat{\mathbf{x}}, \hat{T})$, we know $\nexists t_b \in (t_{\hat{b}}, prev(t_c)]$ such that $g(\hat{\mathbf{x}}, (t_b, t_c]) > g(\hat{\mathbf{x}}, \hat{T})$. It follows $(\hat{\mathbf{x}}, \hat{T}) \in \mathcal{D}(t_{c-1})$ that $(\hat{\mathbf{x}}, \hat{T})$ is an indecomposable DBS in $\mathcal{G}(t_{\hat{b}}, t_c)$. Therefore, $(\hat{\mathbf{x}}, \hat{T}) \in \mathcal{D}(t_c)$. □

Theorem 5.3 provides a straightforward way to verify whether a ODBSC $(\hat{\mathbf{x}}, \hat{T})$, whose begin time $t_{\hat{b}}$ is smaller than t_s, is contained in $\mathcal{D}(t_c)$. If $(\hat{\mathbf{x}}, \hat{T}) \notin \mathcal{D}(t_c)$, it follows Theorem 5.3 that $\exists t_h \in (t_s, prev(t_c)]$ such that $g(\hat{\mathbf{x}}, (t_h, t_c]) > g(\hat{\mathbf{x}}, \hat{T})$, then we update $(\hat{\mathbf{x}}, \hat{T})$ by calling $FindDBS(\mathcal{G}(t_s, t_c), \theta, (\hat{\mathbf{x}}, (t_h, t_c]))$; if $(\hat{\mathbf{x}}, \hat{T}) \in \mathcal{D}(t_c)$, we keep it in $\mathcal{D}(t_c)$.

We summarize $UpdateODBSC$ and $TopkDBSOL$ in Algorithm 4 and Algorithm 5, respectively.

Next, we show that $TopkDBSOL$ achieves a 2-approximation of the top-1 indecomposable DBS in $\mathcal{G}(t_0, t_c)$.

Theorem 5.4. *$TopkDBSOL$ produces a 2-approximation of the top-1 indecomposable DBS in $\mathcal{G}(t_0, t_c)$.*

Proof. Denote by $(\tilde{\mathbf{x}}, \tilde{T})$ the real top-1 indecomposable DBS in $\mathcal{G}(t_0, t_c)$, where $\tilde{T} = (t_{\tilde{b}}, t_{\tilde{e}}]$. By Theorem 5.2, $\forall v_i \in (\tilde{\mathbf{x}}, \tilde{T})$, $g(\tilde{\mathbf{x}}, \tilde{T}) \leq \alpha(v_i, \tilde{T})$. Denote by $\epsilon_k(t_{\tilde{e}-1})$ the k-th largest burstiness of all indecomposable DBSs in $\mathcal{D}(t_{\tilde{e}-1})$. Since $(\tilde{\mathbf{x}}, \tilde{T})$ is the top-1 indecomposable DBS, $\alpha(v_i, \tilde{T}) \geq g(\tilde{\mathbf{x}}, \tilde{T}) \geq \epsilon_k(t_{\tilde{e}-1})$. Thus, for $t_c = t_{\tilde{e}}$, step 7 of $FindNDBSC$ calls $FindDBS$ with an initialization $(\mathbf{o}(v_i, \tilde{T}), \tilde{T})$ to find a NDBSC $(\hat{\mathbf{x}}, \hat{T}) \in \mathcal{D}(t_{\tilde{e}})$. Since $g(\mathbf{o}(v_i, \tilde{T}), \tilde{T}) \leq g(\hat{\mathbf{x}}, \hat{T}) \leq g(\tilde{\mathbf{x}}, \tilde{T}) \leq \alpha(v_i, \tilde{T})$ and $g(\mathbf{o}(v_i, \tilde{T}), \tilde{T}) = \frac{0.5*0.5*\max_j A_{ij}(t_{\tilde{b}+1}, t_{\tilde{e}}) + 0.5*0.5*\max_j A_{ij}(t_{\tilde{b}+1}, t_{\tilde{e}})}{t_{\tilde{e}} - t_{\tilde{b}}}$ $= 0.5 * \alpha(v_i, \tilde{T})$, $(\hat{\mathbf{x}}, \hat{T})$ is a 2-approximation of $(\tilde{\mathbf{x}}, \tilde{T})$. For $t_c > t_{\tilde{e}}$, if $(\hat{\mathbf{x}}, \hat{T}) \notin \mathcal{D}(t_c)$, $UpdateODBSC$ updates $(\hat{\mathbf{x}}, \hat{T})$ to increase its burstiness. The updated $(\hat{\mathbf{x}}, \hat{T})$ is still a 2-approximation of $(\tilde{\mathbf{x}}, \tilde{T})$. □

In the next chapter, we demonstrate the effectiveness and efficiency of our proposed online method by extensive experiments.

Chapter 6

Experiment

In this chapter, we evaluate the performance of the proposed methods *SlideWin* (**SW**) and *TopkDBSOL* (**OL**), and compare them with the state-of-the-art method *DenseAlert* (**DA**) [31]. To evaluate the effect of the smart initialization in Section 5.2.2, we also implement an algorithm named $TopkDBSOL^{nsi}$ ($\mathbf{OL^{nsi}}$) by disabling the smart initialization of OL. The Java code for DA is provided by its authors [31]. Our algorithms are implemented in C++. All experiments are conducted on a PC with Core-i7-3370 CPU (3.40 GHz), 16GB main memory, and a 5400 rpm hard drive running Windows 7 OS.

We report the **running time** (**RT**) of an algorithm. We also report **edge density burstiness** (**EDB**), the speed that a detected temporal subgraph accumulates edge density [13, 33, 23]. Specifically, denote by $(S, (t_b, t_e])$ a detected temporal subgraph that is induced by the set of vertices S from the snapshots of $\mathcal{G}(t_{b+1}, t_e)$. The **edge density** of the subgraph induced by S from $G(t_h)$ is computed by Eq. 6.1.

$$ED_S(t_h) = \frac{\sum_{v_i \in S} \sum_{v_j \in S} A_{ij}(t_h)}{|S|(|S|-1)} \tag{6.1}$$

, where $|S|$ is the volume of S. The **EDB** of $(S, (t_b, t_e])$ is computed by Eq. 6.2.

$$\frac{1}{t_e - t_b} \sum_{h=b+1}^{e} ED_S(t_h) \tag{6.2}$$

We employ EDB to evaluate the burstiness of detected temporal subgraphs, because the output of DA cannot be used to directly compute cohesiveness [29], and edge density is a well recognized evaluation metric for graph density [13, 33, 23].

We use the following five public real world data sets. Detailed information of the data sets is shown in Table 6.1.

DBLP Coauthorship (DBLP) Data Set [34]. This data set is the co-authorship network in DBLP. Each vertex is an author. Each edge represents the co-authorship between two authors. The edge weight is the number of coauthored publications.

Table 6.1: Detailed description of data sets.

Data Set	# Vertices	# Edges	# Snapshots	Time Granularity
DBLP	1,282,461	7,354,929	45	1 Snapshot / YEAR
ENRON	87,273	920,478	2,222	1 Snapshot / DAY
FBWP	46,952	585,932	1,591	1 Snapshot / DAY
TAXI-1	362	88,547	96	1 Snapshot / 15 MIN.
TAXI-2	362	70,202	96	1 Snapshot / 15 MIN.
KAN	33,967	142,068	20	1 Snapshot / YEAR

ENRON Data Set [34]. This data set is the email communication network of Enron. Each vertex is an employee. Each edge represents the email communication between two employees. The edge weight is the number of emails sent between two users.

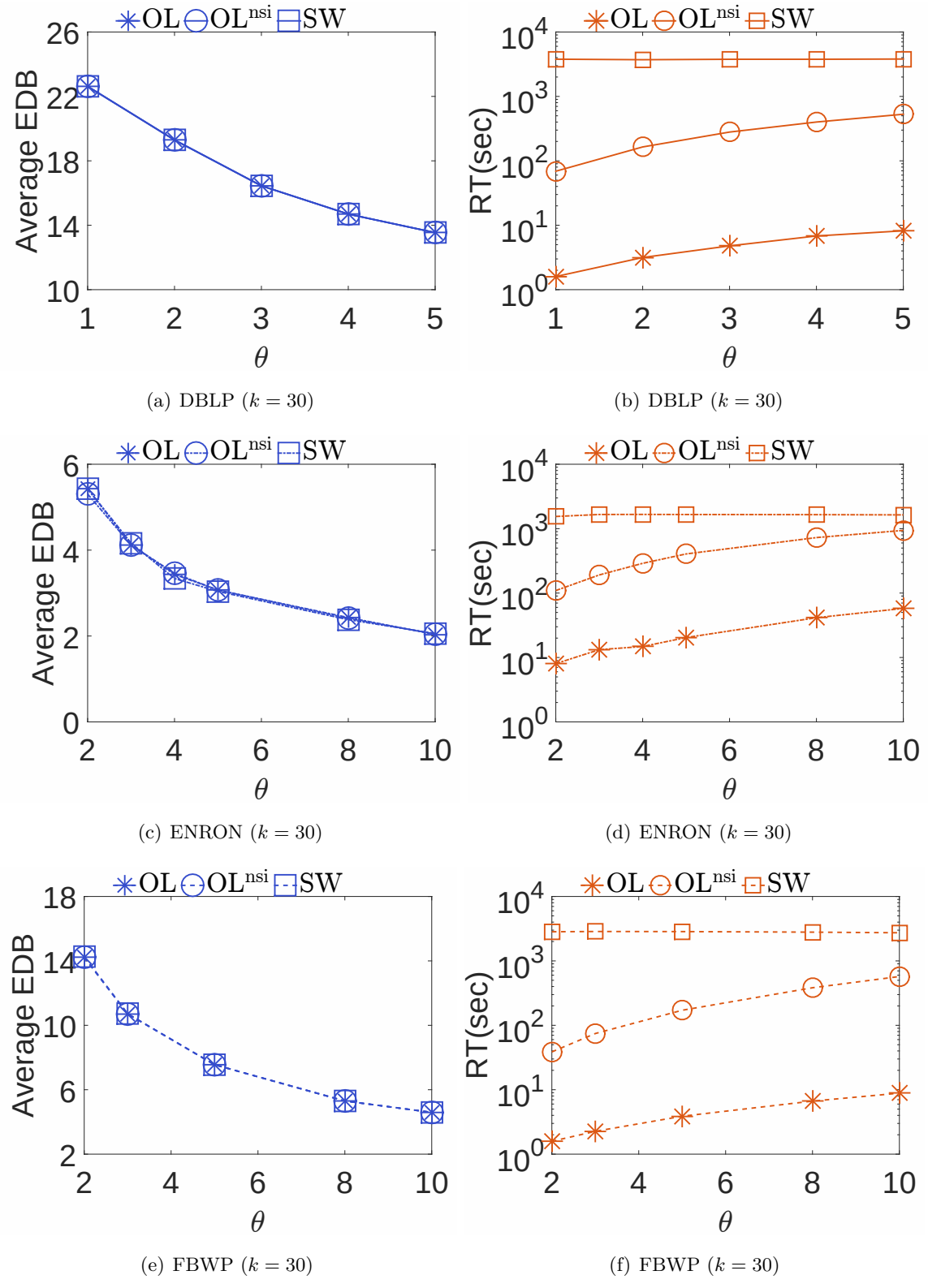
Facebook Wall Posts (FBWP) Data Set [34]. This data set is the wall post network of Facebook. Each vertex is a user. Each edge represents the wall post activity between two users. The edge weight is the number of wall posts.

TAXI-1 and TAXI-2 Data Sets. These data sets are constructed using the taxi trips in July 2017 in the city of Chicago[1]. We uniformly partition the city into 362 blocks, each of which corresponds to a vertex in the temporal graph. The period of one day is uniformly divided into 96 snapshots, each of which corresponds to a time interval of 15 minutes. The taxi trips during the same time interval of all days are grouped into the same snapshot. For each snapshot, the edge weight between two vertices is the number of taxi trips between two locations. TAXI-1 and TAXI-2 consist of the taxi trips of weekdays and weekends, respectively.

Keyword Association Network (KAN) Data Set. This data set is a keyword association network [3] extracted from the DBLP-Citation-network V10 data set (`https://static.aminer.org/lab-datasets/citation/dblp.v10.zip`). We use the abstracts of the papers published in some well established data mining venues, such as KDD, ICDM, SDM, PKDD, PAKDD, TKDE and TKDD. Each vertex is a keyword. Each edge represents the co-occurrence of two keywords in the same abstract. The edge weight is the number of co-occurring abstracts. In this data set, a DBS corresponds to a set of keywords that describe an emerging hot research topic.

Since SW runs too slow on the full data sets of DBLP, ENRON and FBWP, we sample a small data set from each full data set for the experiments of SW. For DBLP, we first sample $10,000$ vertices from the accumulated graph of all snapshots by breath first search, and then use the temporal subgraph induced by the sampled vertices from each snapshot to form a sample of a snapshot. For ENRON and FBWP, we use the complete set of vertices, and sample two temporal graphs with continuous durations of 30 days and 50 days, respectively.

[1] `https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew`



(a) DBLP ($k = 30$)

(b) DBLP ($k = 30$)

(c) ENRON ($k = 30$)

(d) ENRON ($k = 30$)

(e) FBWP ($k = 30$)

(f) FBWP ($k = 30$)

Figure 6.1: The effect of parameters θ. Figures (a), (c) and (e) show the Average Edge Density Burstiness (EDB) of detected DBSs by all methods. Figures (b), (d) and (f) show the Running Time (RT) of all methods.

Table 6.2: The numbers of indecomposable DBSs (#Indec) and decomposable DBSs (#Dec) detected by SW. We rank all DBSs in descending order of burstiness, and *Rank* is the rank of the decomposable DBS with the largest burstiness.

	DBLP			ENRON			FBWP		
θ	2	3	5	2	5	10	2	5	10
#Indec	4,985	3,915	2,684	9,074	6,197	4,355	35,788	29,988	23,806
#Dec	51	6	0	14	0	0	47	0	0
Rank	613	402	N/A	244	N/A	N/A	310	N/A	N/A

6.1 Effects of Parameters

In this section, we analyze the effects of parameters θ and k on the small data sets sampled from DBLP, ENRON and FBWP.

6.1.1 Effect of θ

We show the effects of parameter θ on Running Time (RT) and Average Edge Density Burstiness (EDB) in Figure 6.1.

Figures 6.1(a), 6.1(c) and 6.1(e) show the effect of θ on Average EDB. That is, the Average EDB decreases when θ increases. This is because most co-authors in DBLP do not stay highly productive together for a long time, the email frequencies between colleagues in ENRON do not stay high for long, and most social events in FBWP only heat up at the fastest speed for a short time.

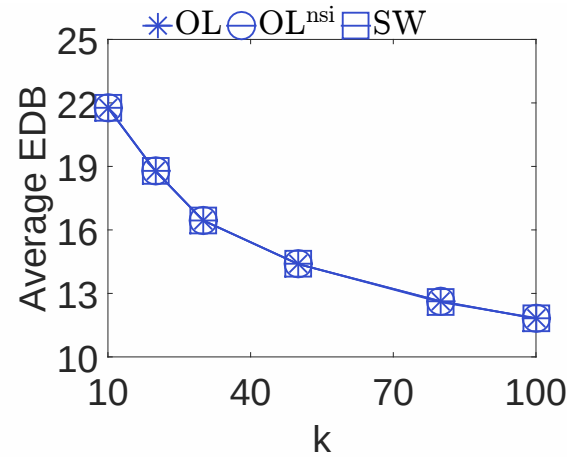
When θ increases, the Running Time of OL and OL^{nsi} increases, because a large θ increases the number of buffered snapshots in $\mathcal{G}(t_s, t_c)$, which increases the running time of $FindDBS$ in Algorithm 3 and Algorithm 4. While the Running Time of SW stays stable, because increasing θ increases the time cost of $FindDBS$, however, reduces the number of times to call $FindDBS$ in Algorithm 2. This is clearly shown in Figures 6.1(b), 6.1(d), 6.1(f).

Table 6.2 shows the effect of θ on the numbers of indecomposable DBS and decomposable DBS. Both #Indec and #Dec decrease when θ increases, because θ is the minimum duration threshold of DBS, a larger θ rules out more DBSs whose duration is smaller than θ.

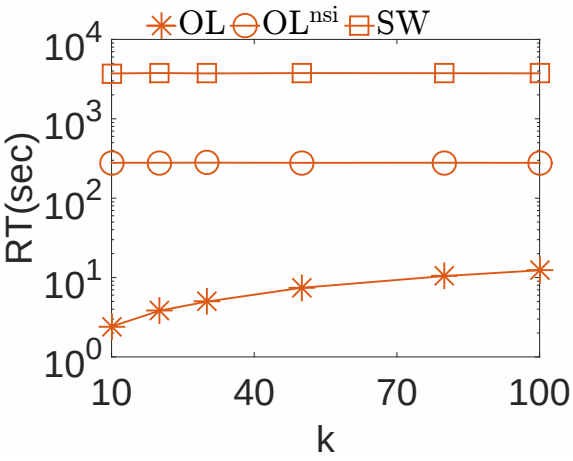
We can also see from Table 6.2 that #Dec is less than 1% of all detected DBSs, and the decomposable DBS with the largest burstiness has a very low *Rank*. These results verify our observation in Section 5.1: regarding the necessary condition (Theorem 5.1) for a DBS to be decomposable, most DBSs with large burstiness are indecomposable in practice.

6.1.2 Effect of k

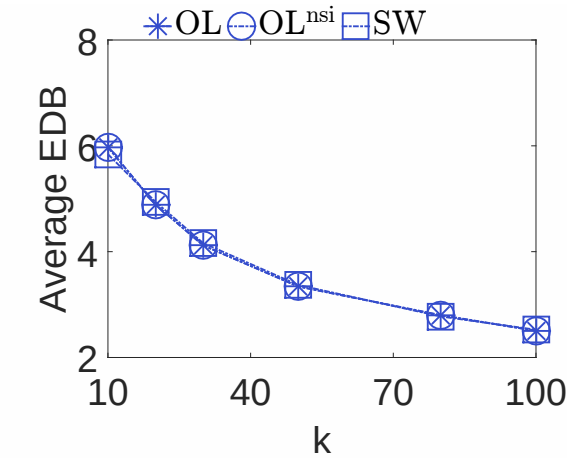
In this subsection, we will analyze the effect of k on Running Time (RT) and Average Edge Density Burstiness (EDB), as shown in Figure 6.2.



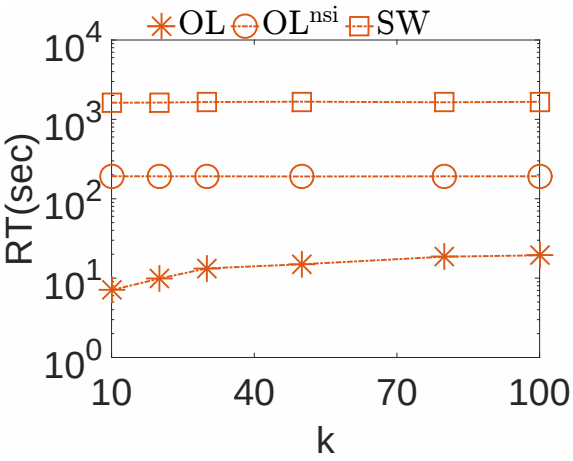
(a) DBLP ($\theta = 3$)



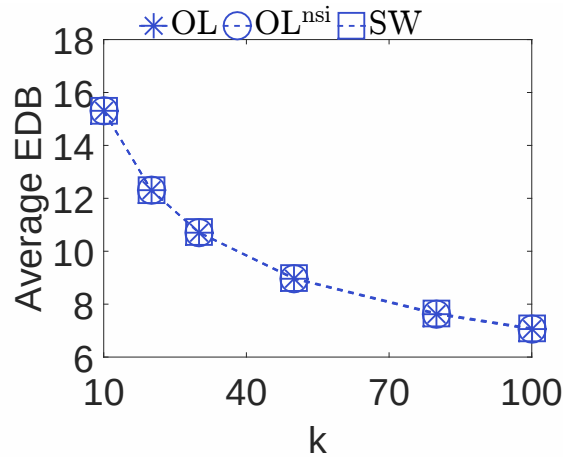
(b) DBLP ($\theta = 3$)



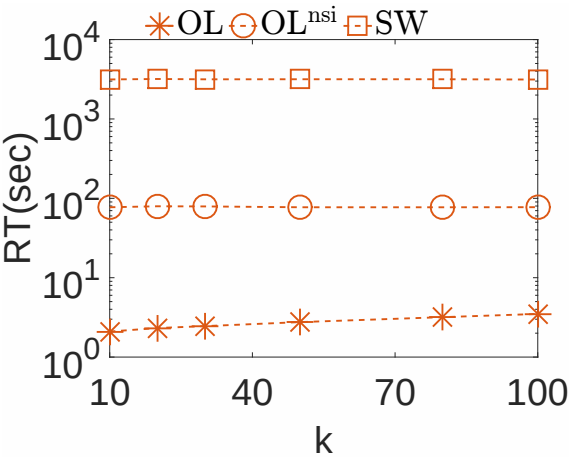
(c) ENRON ($\theta = 3$)



(d) ENRON ($\theta = 3$)



(e) FBWP ($\theta = 3$)



(f) FBWP ($\theta = 3$)

Figure 6.2: The effects of parameter k. Figures (a), (c) and (e) show the Average Edge Density Burstiness (EDB) of detected DBSs by all methods. Figures (b), (d) and (f) show the Running Time (RT) of all methods.

Table 6.3: The numbers of vertices (#V) and edges (#E) of the data sets sampled from DBLP, ENRON and FBWP.

ID	DBLP		ENRON		FBWP	
	#V ($\times 10^3$)	#E ($\times 10^5$)	#V ($\times 10^3$)	#E ($\times 10^5$)	#V ($\times 10^3$)	#E ($\times 10^5$)
1	20.0	2.5	1.0	0.5	2.0	0.2
2	100.0	11.6	5.0	2.3	6.0	0.9
3	300.0	30.8	9.0	4.8	10.0	1.8
4	500.0	46.0	15.0	6.3	20.0	3.8
5	1282.5	73.5	87.3	9.2	47.0	5.9

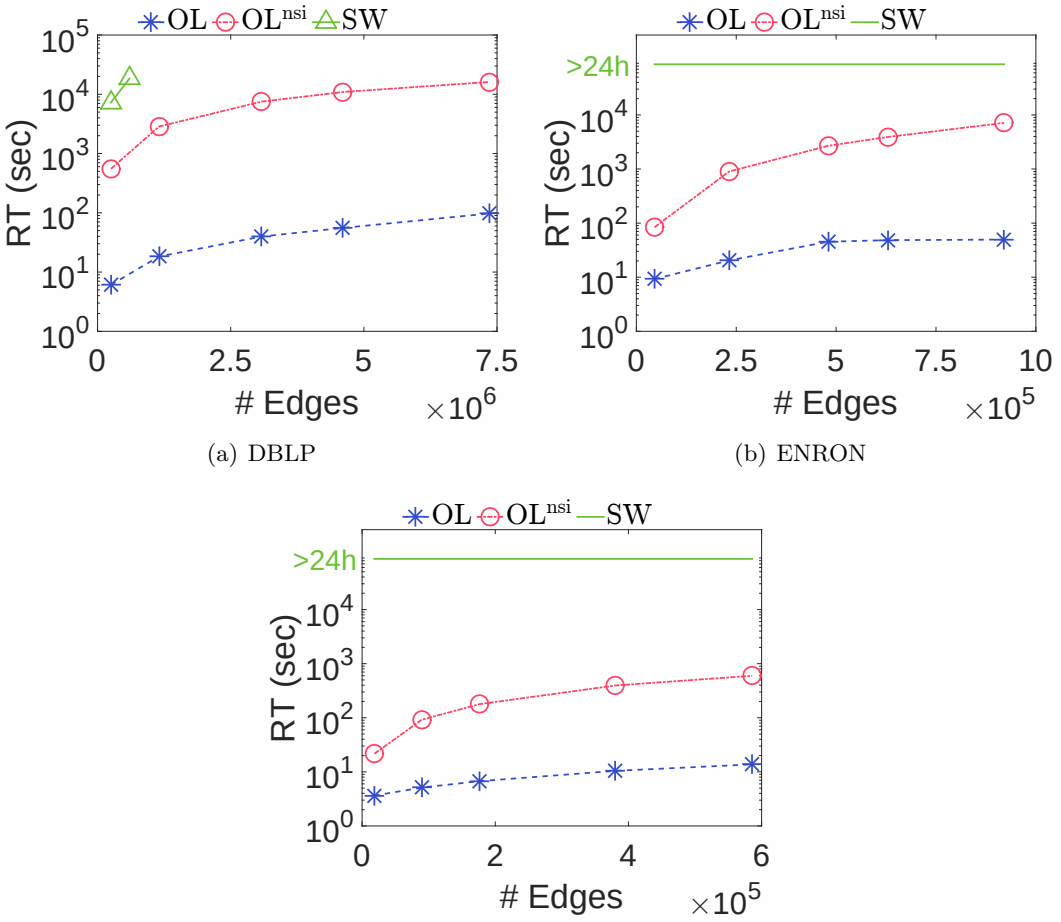
Since the detected DBSs are ranked in burstiness, the Average EDB decreases when k increases, which is consistent with the curves in Figures 6.2(a), 6.2(c), 6.2(e). The efficiency of OL^{nsi} and SW are irrelevant to k, thus their Running Time stays stable. However, the Running Time of OL increases, because a larger k decreases the value of $\epsilon_k(t_{c-1})$, which increases the number of times to call $FindDBS$ in Algorithm 3. This is clearly shown in Figures 6.2(b), 6.2(d) and 6.2(f).

As shown in Figure 6.1 and Figure 6.2, for all values of θ and k, the Average EDB of OL and OL^{nsi} are highly comparable with the brute force baseline SW. OL is dramatically faster than OL^{nsi} and SW by orders of magnitudes. These results demonstrate the effectiveness and efficiency of OL in detecting high quality DBSs with large burstiness.

6.2 Scalability Analysis

We compare the scalability of OL, OL^{nsi} and SW on DBLP, ENRON and FBWP. To obtain a series of samples of different sizes, for each data set, we sample four temporal graphs as follows. First, we start a Breadth First Search (BFS) from a randomly picked vertex on the accumulated graph of all snapshots. Second, let S be the set of all vertices visited by the BFS, we use S to induce a subgraph from each of the snapshots of the original temporal graph. Last, we use the sequence of induced subgraphs as a sampled temporal graph. The numbers of vertices and edges of the sampled temporal graphs are listed in Table 6.3, where the 5-th data set is simply the complete data set.

Figure 6.3 shows the RT performance. We do not report the RT of SW on ENRON and FBWP, because SW cannot finish in 24 hours due to its quadratic time complexity with respect to the number of snapshots. Since the smart initialization effectively reduces the number of calls of $FindDBS$ in Algorithm 3, OL always completes in less than 100 seconds, which is 2 orders of magnitudes faster than OL^{nsi} and is at least 4 orders of magnitudes faster than SW.



(a) DBLP

(b) ENRON

(c) FBWP

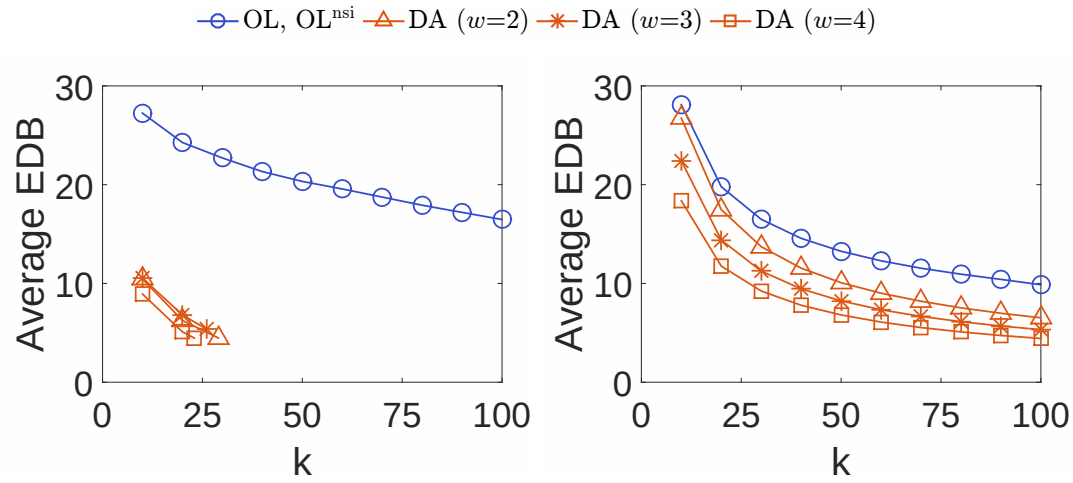
Figure 6.3: The RT of OL, OL^{nsi} and SW on the temporal graphs sampled from DBLP, ENRON and FBWP. The parameters are $\theta = 3, k = 30$.

6.3 Comparison with DenseAlert

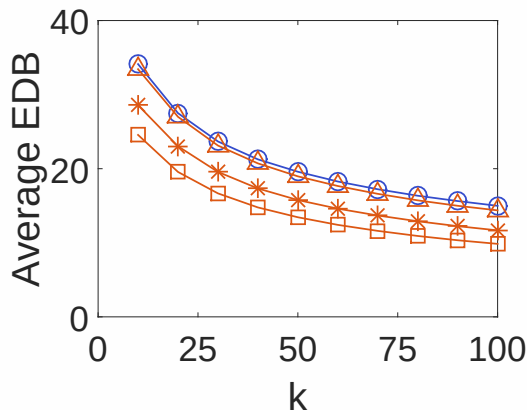
We compare the Average EDB of OL, OL^{nsi} and DA on the full datasets of DBLP, ENRON and FBWP. The Average EDB of OL and OL^{nsi} are identical, which demonstrates that the smart initialization in Algorithm 3 does not affect the Average EDB of OL.

In Figure 6.4(a), OL achieves a much higher Average EDB than DA on DBLP. This is because DA maintains the dense temporal subgraph with the largest average degree [18], which is typically a large graph with a small edge density [33].

Since DA only maintains the top-1 dense temporal subgraph, it often detects duplicate temporal subgraphs. The number of detected dense temporal subgraphs is limited by the number of snapshots. For example, in Figure 6.4(a), DA detects 45 dense temporal subgraphs from the 45 snapshots of DBLP, but in every case with respect to a fixed value of w, only less than 30 are not duplicates.



(a) DBLP (b) ENRON



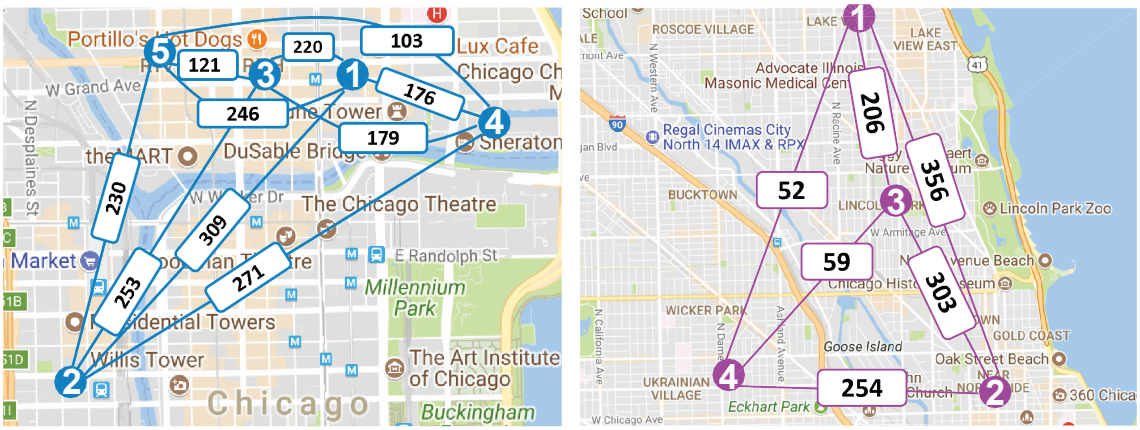
(c) FBWP

Figure 6.4: The Average EDB of OL, OL^{nsi} and DA. We set $\theta = 3$ for OL, OL^{nsi}. Since DA uses a fixed size sliding window, we set the window size w of DA as $w = \{2, 3, 4\}$, such that the sliding window contains 3, 4 and 5 snapshots, respectively.

In Figures 6.4(b)-(c), the Average EDB of OL is still higher than DA on ENRON and FBWP. However, the advantage of OL is not as significant as on DBLP, because the temporal networks of ENRON and FBWP are much sparser than DBLP. In sparse temporal networks, DA is forced to find small temporal subgraphs with heavily weighted edges, thus it achieves a closer Average EDB to OL. However, since DA focuses on optimizing average degree, its Average EDB is always inferior to OL.

6.4 Case Study

In this section, we will show the practical effectiveness of our method by two cases on the datasets TAXI-1, TAXI-2 and KAN, respectively.



(a) Weekdays (17:30-18:45)

(b) Weekends (22:45-23:45)

Figure 6.5: The bursting taxi trips between a set of locations in the city of Chicago in US. (a) shows the number of taxi trips between a set of locations during 17:30-18:45 on TAXI-1. (b) shows the number of taxi trips between a set of locations during 22:45-23:45 on TAXI-2.

6.4.1 Discovering Travel Patterns from TAXI-1 and TAXI-2

We show some interesting patterns of taxi trips discovered by finding the most significant DBSs from TAXI-1 and TAXI-2.

Figure 6.5(a) shows the bursting taxi trips between a set of locations during 17:30-18:45 on weekdays. The location 2 is surrounded by office buildings, the locations 1, 3, 4 and 5 are surrounded by shopping centres and restaurants. The number of taxi trips between these locations burst during 17:30-18:45 on weekdays, because people often go shopping and dine out after a day's busy work.

Figure 6.5(b) shows an even more interesting pattern of bursting taxi trips during 22:45-23:45 on weekends. *Where are people going at midnight?* The answer lies in the locations 1, 3 and 4, where the neighbourhoods have many mid-night pubs and restaurants that open late at night. *Who are these people?* We investigate the neighbourhood of location 2 and find that it is a college town surrounded by the Moody Bible Institute and the Loyola University Chicago. Most of the taxi trips are related to location 2. Obviously, midnight parties at pubs are one of the favourite weekend entertainments of young students. The pubs and restaurants start to close at 22:30, and many people go home before 00:00, therefore, the number of taxi trips bursts during 22:45-23:45.

In this simple case study, finding DBSs from the temporal network of taxi trips discovers interesting travel patterns that provides useful insights into people's behavior patterns. Such patterns can be used to, for example, improve taxis dispatch plans and develop better public traffic designs.

Table 6.4: The durations and topics of detected DBSs.

ID	Keywords	Years
1	deep learning, image patch, feature representation	2012-2015
2	multiple type, link prediction, heterogeneous information network	2011-2014
3	visual word, local feature, inverted index	2010-2013
4	domain adaptation, conditional probability, semg signal	2010-2013
5	graph laplacian, cluster label, feature selection algorithm, spectral feature, unsupervised feature selection	2012-2015
6	loss function, machine learning, wide range, vast amount	2011-2015

6.4.2 Finding Emerging Research Topics from KAN

In this subsection, we show some interesting research topics discovered by finding DBSs from the KAN dataset.

Each row of Table 6.4 shows a set of keywords, which describe a research topic that emerges at the fastest speed during the corresponding years. To validate the duration of the emerging topics, we crawled from AMINER (`https://aminer.org/`) the annual popularity of each keyword, and computed the popularity of each topic by the product of the popularities of all related keywords. Figure 6.6 shows the normalized popularities of the detected topics in each year.

As shown in Figure 6.6, the durations of all topics in Table 6.4 accurately highlight the time when their popularities rise fast.

In summary, by finding DBSs in the keyword association network of KAN, the proposed method effectively detects hot research topics, as well as the time interval when their popularity rise fast.

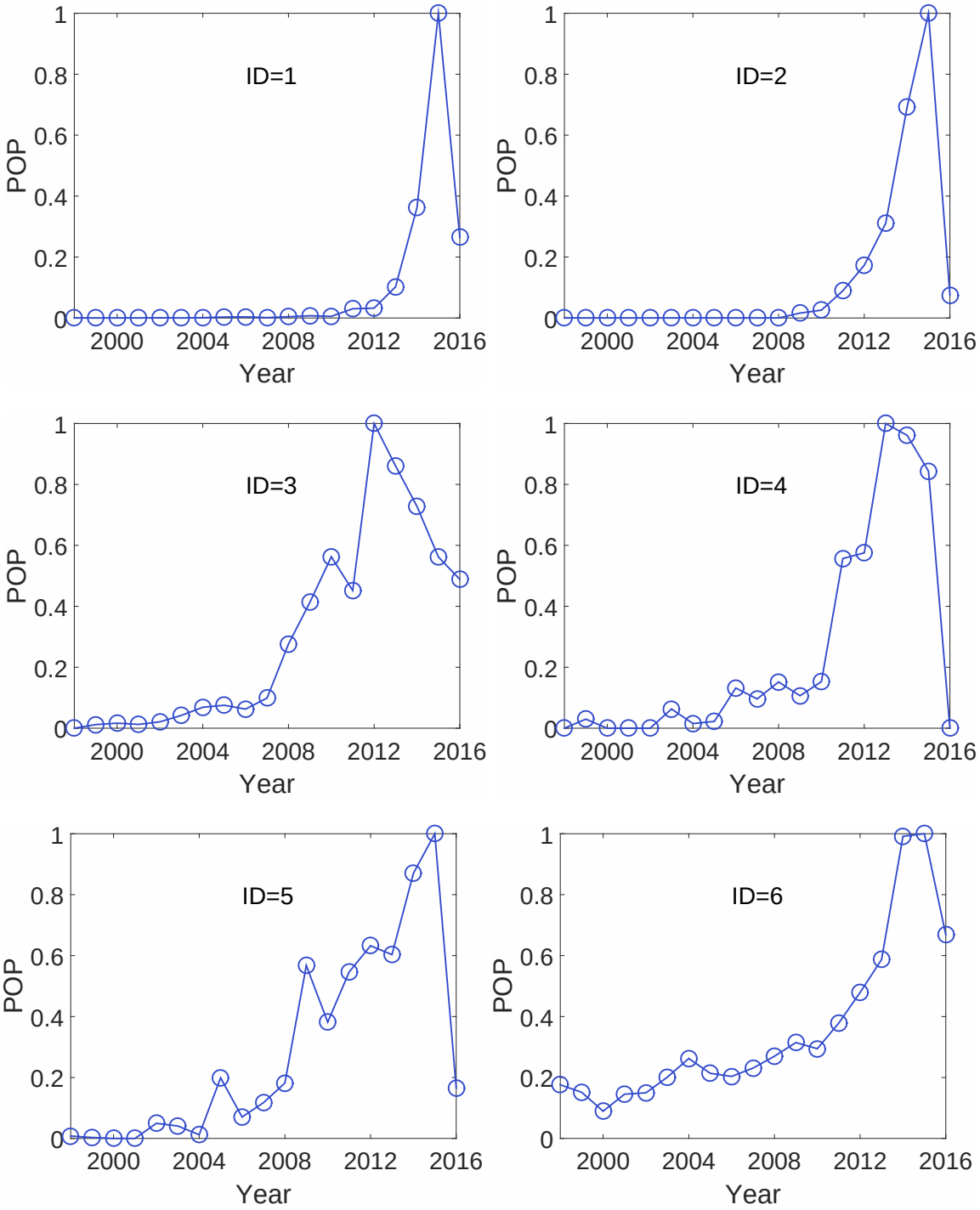


Figure 6.6: The normalized popularity (POP) of detected topics. x-axis is the time line of years. ID corresponds to Table 6.4.

Chapter 7

Conclusions

Many graphs in real world applications are temporal graphs that consist of a potentially endless stream of updates. Detecting DBSs from temporal graphs is a useful task that enjoys many interesting applications.

In this thesis, we tackle the novel problem of finding top-k DBSs from temporal graphs. We formulate the top-k DBSs finding problem as a MIP problem and, as a baseline, solve it by $SlideWin$. By investigating the decomposition property of DBSs, we further design $TopkDBSOL$ to find the set of top-k indecomposable DBSs in an online manner. Extensive experiments show that $TopkDBSOL$ finds a comparably good solution as $SlideWin$, and improves the efficiency of $SlideWin$ by orders of magnitudes in finding DBSs from large temporal graphs. The cases of Chicago taxi and keyword association network show that detecting DBSs discovers meaningful patterns in real world scenarios.

The method proposed in this thesis is the beginning of our DBS detection framework. In the future, we will further enrich our toolkit of DBS detection towards the following interesting tasks.

- *Recovering decomposable DBSs by aggregating indecomposable DBSs.* In this thesis, we show the infeasibility of directly finding decomposable DBSs in an online manner, and also prove that a decomposable DBS can be decomposed into a set of indecomposable DBSs with equal or larger burstiness. The proposed $TopkDBSOL$ focuses on detecting indecomposable DBSs, however, since some indecomposable DBSs are components of decomposable DBSs, how to recover decomposable DBSs by aggregating detected indecomposable DBSs is an interesting task that worth further investigation. As a naive example, we can recover a decomposable DBS by concatenating the detected indecomposable DBSs that have exactly the same burstiness, and are induced by the same set of vertices in successive time durations.

- *Finding DBSs from signed temporal networks.* Opinion based temporal social networks, such as Slashdot[1] and Epinions[2], consist of cohesive and opponent relationships. These networks are naturally modeled as signed temporal networks, whose edge weights between vertices are either positive (i.e., cohesive relationship) or negative (i.e., opponent relationship). Finding top-k DBSs from signed temporal networks finds emerging communities of coherent opinions, which is potentially useful in monitoring the trend of social opinions. However, it is also a challenging task to extend $TopkDBSOL$ to find DBSs from signed temporal networks, because the cohesiveness [29] in Equation 3.1 is not well defined due to the existence of negatively weighted edges.

- *Finding DBSs from temporal hypergraphs.* A temporal hypergraph, where each hyperedge indicates the relationship among multiple vertices at a specific time, is powerful in describing the complex dynamic relationships among multiple objects. For example, temporal hypergraph is superior for modeling multi-user relationships in social networks [36], such as multi-player online gaming networks and email networks where the carbon copying operation often involves multiple users. Detecting DBSs from such temporal hypergraphs discovers emerging communities of users that are strongly connected by hyperedges. However, there is no straightforward way to extend $TopkDBSOL$ to find DBSs from temporal hypergraphs, because our current definition and properties of DBSs do not hold on temporal hypergraphs.

[1] `https://snap.stanford.edu/data/soc-Slashdot0811.html`

[2] `https://snap.stanford.edu/data/soc-Epinions1.html`

Bibliography

[1] James Abello, Mauricio Resende, and Sandra Sudarsky. Massive quasi-clique detection. *Theoretical Informatics*, pages 598–612, 2002.

[2] Charu C Aggarwal, Yao Li, Philip S Yu, and Ruoming Jin. On dense pattern mining in graph streams. *PVLDB*, 3(1-2):975–984, 2010.

[3] A. Angel, N. Sarkas, N. Koudas, and D. Srivastava. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *PVLDB*, 5(6):574–585, 2012.

[4] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. Tsourakakis. Space-and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *ACM symposium on Theory of Computing*, pages 173–182, 2015.

[5] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl. Mining coherent subgraphs in multi-layer graphs with edge labels. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 1258–1266, 2012.

[6] Petko Bogdanov, Misael Mongiovì, and Ambuj K Singh. Mining heavy subgraphs in time-evolving networks. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 81–90, 2011.

[7] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *ACM Communications*, 16(9):575–577, 1973.

[8] Samuel Rota Bulò and Immanuel M Bomze. Infection and immunization: a new class of evolutionary game dynamics. *Games and Economic Behavior*, 71(1):193–211, 2011.

[9] K. Chao. Maximum-density segment. In *Encyclopedia of Algorithms*, pages 1–99. 2008.

[10] Tianlong. Chen, Shuqiang Jiang, Lingyang Chu, and Qingming Huang. Detection and location of near-duplicate video sub-clips by finding dense subgraphs. In *MM*, pages 1173–1176, 2011.

[11] Lingyang Chu, Shuhui Wang, Siyuan Liu, Qingming Huang, and Jian Pei. Alid: scalable dominant cluster detection. *PVLDB*, 8(8):826–837, 2015.

[12] Lingyang Chu, Shuhui Wang, Yanyan Zhang, Shuqiang Jiang, and Q. Huang. Graph-density-based visual word vocabulary for image retrieval. In *IEEE International Conference on Multimedia and Expo, ICME 2014, Chengdu, China, July 14-18, 2014*, pages 1–6, 2014.

[13] Lingyang Chu, Zhefeng Wang, Jian Pei, Jiannan Wang, Zijin Zhao, and Enhong Chen. Finding gangs in war from signed networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1505–1514, 2016.

[14] Lingyang Chu, Yanyan Zhang, Guorong Li, Shuhui Wang, Weigang Zhang, and Qingming Huang. Effective multimodality fusion framework for cross-media topic detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(3):556–569, 2016.

[15] Kai-min Chung and Hsueh-I Lu. An optimal algorithm for the maximum-density segment problem. *SIAM Journal on Computing*, 34(2):373–387, 2004.

[16] Sharon Curtis and Shin-Cheng Mu. Calculating a linear-time solution to the densest-segment problem. *Journal of Functional Programming*, 25, 2015.

[17] Alessandro Epasto, Silvio Lattanzi, and Mauro Sozio. Efficient densest subgraph computation in evolving graphs. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 300–310, 2015.

[18] AV Goldberg. Finding a maximum density subgraph. 1984.

[19] Michael H Goldwasser, Ming-Yang Kao, and Hsueh-I Lu. Linear-time algorithms for computing maximum-density sequence segments with bioinformatics applications. *J. Comput. Syst. Sci.*, 70(2):128–144, 2005.

[20] Xiaoqui Huang. An algorithm for identifying regions of a dna sequence that satisfy a content requirement. *Bioinformatics*, 10(3):219–225, 1994.

[21] Sung Kwon Kim. Linear-time algorithm for finding a maximum-density segment of a sequence. *Inf. Process. Lett.*, 86(6):339–342, 2003.

[22] Yaw-Ling Lin, Tao Jiang, and Kun-Mao Chao. Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis. *J. Comput. Syst. Sci.*, 65(3):570–586, 2002.

[23] H. Liu, L. J. Latecki, and S. Yan. Fast detection of dense subgraphs with iterative shrinking and expansion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(9):2131–2142, 2013.

[24] Hairong Liu and Shuicheng Yan. Common visual pattern discovery via spatially coherent correspondences. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 1609–1616, 2010.

[25] Hairong Liu and Shuicheng Yan. Robust graph mode seeking by graph shift. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 671–678, 2010.

[26] Hairong Liu and Shuicheng Yan. Efficient structure detection via random consensus graph. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 574–581, 2012.

[27] Shuai Ma, Renjun Hu, Luoshu Wang, Xuelian Lin, and Jinpeng Huai. Fast computation of dense temporal subgraphs. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 361–372, 2017.

[28] Theodore S Motzkin and Ernst G Straus. Maxima for graphs and a new proof of a theorem of turán. *Canad. J. Math*, 17(4):533–540, 1965.

[29] Massimiliano Pavan and Marcello Pelillo. Dominant sets and pairwise clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(1):167–172, 2007.

[30] Jian Pei, Daxin Jiang, and Aidong Zhang. On mining cross-graph quasi-cliques. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 228–238, 2005.

[31] K. Shin, B. Hooi, J. Kim, and C. Faloutsos. Densealert: Incremental dense-subtensor detection in tensor streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 1057–1066, 2017.

[32] Kijung Shin, Bryan Hooi, Jisu Kim, and Christos Faloutsos. D-cube: Dense-block detection in terabyte-scale tensors. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, pages 681–689, 2017.

[33] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 104–112, 2013.

[34] UKL. Konect. `http://konect.uni-koblenz.de/`, 2018.

[35] Jia Wang and James Cheng. Truss decomposition in massive networks. *PVLDB*, 5(9):812–823, 2012.

[36] Wenyin Yang, Guojun Wang, Md Zakirul Alam Bhuiyan, and Kim-Kwang Raymond Choo. Hypergraph partitioning for social networks based on information entropy modularity. *Journal of Network and Computer Applications*, 86:59–71, 2017.

[37] Yi Yang, Da Yan, Huanhuan Wu, James Cheng, Shuigeng Zhou, and John CS Lui. Diversified temporal subgraph pattern mining. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1965–1974, 2016.

[38] Yu Yang, Lingyang Chu, Yanyan Zhang, Zhefeng Wang, Jian Pei, and Enhong Chen. Mining density contrast subgraphs. *arXiv preprint arXiv:1802.06775*, 2018.

[39] Yanyan Zhang, Guorong Li, Lingyang Chu, Shuhui Wang, Weigang Zhang, and Qingming Huang. Cross-media topic detection: A multi-modality fusion framework. In *Proceedings of the 2013 IEEE International Conference on Multimedia and Expo, ICME 2013, San Jose, CA, USA, July 15-19, 2013*, pages 1–6, 2013.