Approximation Algorithms for the Minimum (Sliding Window) Temporal Vertex Cover Problem

Sophia Heck

July 29, 2023

3725826

Master Thesis

at

Algorithm Engineering Group Heidelberg
Heidelberg University

Supervisor:
Univ.-Prof. PD. Dr. rer. nat. Christian Schulz

Co-Referee:
Assoc. Prof. Dr. Eleni Akrida

Acknowledgments

I like to thank Assoc. Prof. Eleni Akrida, who gave me the valuable insight and feedback throughout this thesis. My thanks also go to Prof. Christian Schulz, without whom this work would not have been possible. I also want thank Nina Klobas, who always had an open ear for my questions. Furthermore, I want to express my appreciation for the support of my family and friends throughout my Master's degree.

Hiermit versichere ich, dass ich die Arbeit selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und wörtlich oder inhaltlich aus fremden Werken Übernommenes als fremd kenntlich gemacht habe. Ferner versichere ich, dass die übermittelte elektronische Version in Inhalt und Wortlaut mit der gedruckten Version meiner Arbeit vollständig übereinstimmt. Ich bin einverstanden, dass diese elektronische Fassung universitätsintern anhand einer Plagiatssoftware auf Plagiate überprüft wird.

Heidelberg, July 29, 2023

Sophia Heck

Abstract

Modern real life networks are often highly dynamic. Temporal graphs represent these changes in the network configuration through discrete edge appearances on a fixed set of vertices. In a temporal graph all these changes are known in advance until a maximal timestep, the lifetime of the graph. The classical problem of *Vertex Cover* aims to find a set of vertices such that all edges are covered by one of their endpoints. This can be naturally extended in the time changing setting to the *Temporal Vertex Cover (TVC)* and *Sliding Window Temporal Vertex Cover (Δ-TVC)*. In the TVC every edge is covered once over the whole lifetime, while in the Δ-TVC every appearing edge is covered once in every window of Δ consecutive timesteps. Both extensions are known to be NP-hard. In this thesis, known (approximation) algorithms for these extensions are implemented and experimentally evaluated. In particular, we consider two approximation algorithms and one non-polynomial exact algorithm.

The approximations have approximation ratios bounded by the maximal degree d of any subgraph in at a certain timestep, leading to a ratio of d and $d-1$. Moreover, two new approximation algorithms for the restricted case of always star temporal graphs are presented, leading to a $2\Delta - 1$ and a $\Delta - 1$ approximation ratio, where Δ is the sliding window size. For the experiments a temporal graph generator for certain temporal graph classes and a framework for solving the (Δ-)TVC are introduced. The experiments verify the stated runtime and approximation ratios of the known algorithms and even provide some improvements made through the implementations. We show that on real-life instances the $d-1$-approximation outperforms the d-approximation. Further, we compare the computation of the Δ-TVC on restricted inputs of always star temporal graphs through the known approximation algorithms with the here presented new ones. We show that the new approximations outperform the known $d-1$-approximation in shorter runtime even in some cases where $\Delta > d$.

Zusammenfassung

Moderne reale Netzwerke sind oft sehr dynamisch. Temporale Graphen bieten die Möglichkeit zeitliche Veränderungen durch diskretes Auftreten der Kanten auf einer festen Menge von Knoten abzubilden. In einem temporalen Graphen sind alle diese Änderungen bis zu einem maximalen Zeitschritt, der Lebensdauer des Graphen, im Voraus bekannt. Das klassische Graphproblem der *Knotenüberdeckung (Vertex Cover)* zielt darauf ab, eine Menge von Konten zu finden, so dass alle Kanten durch einen ihrer Endpunkte abgedeckt sind. Dieses Problem lässt sich in einem zeitlich veränderlichen Umfeld auf das *Temporal Vertex Cover (TVC)* und das *Sliding Window Temporal Vertex Cover (Δ-TVC)* erweitern. Im TVC wird jede Kante einmal über die gesamte Lebensdauer abgedeckt, während im Δ-TVC jede auftretende Kante einmal in jedem Fenster von Δ aufeinanderfolgenden Zeitschritten abgedeckt wird. Beide Erweiterungen sind bekanntermaßen NP-hart. In dieser Arbeit werden bekannte (Approximations-)Algorithmen für diese Erweiterungen implementiert und experimentell evaluiert. Im Einzelnen betrachten wir zwei Approximationsalgorithmen und einen nicht-polynomialen exakten Algorithmus.

Die Approximationen haben Approximationsverhältnisse, die durch den maximalen Grad d eines beliebigen Subgraphen in einem bestimmten Zeitschritt begrenzt sind, was zu einem Verhältnis von d und $d - 1$ führt. Darüber hinaus werden zwei neue Approximationsalgorithmen für den eingeschränkten Fall von immer sternförmigen temporalen Graphen vorgestellt, die zu einem $2\Delta - 1$- und einem $\Delta - 1$-Approximationsverhältnis führen, wobei Δ die Größe des Sliding Windows ist. Für die Experimente werden ein temporaler Graphengenerator für bestimmte temporale Graphenklassen und ein Framework zur Lösung der (Δ-)TVC eingeführt. Die Experimente verifizieren die angegebenen Laufzeit- und Approximationsverhältnisse der bekannten Algorithmen und liefern sogar einige Verbesserungen, die durch die Implementierungen erreicht wurden. Wir zeigen, dass in realen Graphen die $d-1$-Approximation die d-Approximation übertrifft. Weiterhin vergleichen wir die Berechnung der Δ-TVC auf immer sternförmigen temporalen Graphen durch die bekannten Approximationsalgorithmen mit den hier vorgestellten neuen. Wir zeigen, dass die neuen Approximationen die bekannte $d - 1$-Approximation in kürzerer Laufzeit übertreffen, selbst in Fällen, in denen $\Delta > d$.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Temporal graphs provide the ability to model systems which change over time. Many real-life systems such as biological, social or technical ones are highly time-varying in their behavior, e.g. in the spread of diseases, communication between individuals or the transfer of data [33] [20]. Therefore, temporal graphs can be used to extract information or solve problems e.g. monitoring tasks [18]. However, classical known graph problems, e.i. graph coloring or vertex cover, need to be adapted first into this time varying setting. The study of temporal graphs has been discussed frequently in recent literature [4], [18], [31]. The time changes modeled through them are varying in the sense that they have a fixed set of vertices and appearing/disappearing edges over time in a discrete manner.
This thesis focuses on the Vertex Cover (VC) problem which searches for a set of vertices, such that all edges are covered by one of their endpoints. This is adapted to Temporal Vertex Cover (TVC) and Sliding Window Temporal Vertex Cover (SW-TVC) [4] to meet the edge changes. Similar to the classical (static) VC the adaptations aim to cover all underlying edges by one of their endpoints. While the classical VC provides a vertex set as solution, the temporal adaptations provide a set of vertex appearances consisting of vertices at certain timesteps [4]. In the TVC an edge is considered covered, if the solution set contains a vertex appearance consisting of one endpoint of the edge and a timesteps, where the edge appears. Hence, every edge is covered at one appearance during the whole lifetime of the graph. This might not be sufficient for applications where e.g. repeated monitoring is required. Therefore, in the SW-TVC provides a window of fixed size Δ and each appearing edge needs to be covered in every Δ consecutive timesteps.
A popular application example of the classical VC is to place guards in a museum, where edges represent corridors with artwork and guards are placed at the vertices, which represent junctions [43]. However, if the museum has changing exhibitions, e.i. not all corridors hold paintings all the time, the problem can not be modeled in the classical way. In this

case one can use temporal graphs to model the temporal behavior and Vertex Cover adaptations, TVC and SW-TVC, to provide a solution to this problem. Most likely we would want to use SW-TVC with $\Delta = 1$, as the artwork should be secured at all time. However, in other monitoring tasks, e.g. in sensor networks, where we know when sensors are supposed to send signals, we might want to check repeatedly if the signals work, without checking every single one. Then we can define a larger Δ or even use TVC for this. Since many social, transportation or biological real-life networks change over time the temporal setting is particularly interesting.
Both adaptations, TVC and SW-TVC, are known to be NP-complete [4], similar to the VC problem [17]. Therefore, the tools of approximation and input restriction are commonly used to provided solutions in a polynomial time [4] [18].

1.2 Our Contribution

We provide an overview and (experimental) analysis of algorithms for SW-TVC and improve its approximation for the special case of always star temporal graphs. Therefore, we look at known approximation algorithms and provide an initial implementation as they have never been implemented. Through experiments, we verify their stated approximation ratios on small instances, stated runtimes and test their capability to solve real-life instances. Further, we focus on the special class of always star temporal graphs in an effort to derive better approximation ratios and runtime.
For the analysis of approximation algorithms on specific graphs, we want to distinguish graph instances based on their class to provide valuable input for testing and benchmarking the algorithms. Since there are no datasets providing a variation of graphs from different temporal graph classes, we develop a random generator for a range of specific classes including, among others, arbitrary, always most degree d and always star.
Our main contribution is a framework for storing temporal graphs and computing the Δ-TVC. Secondly we present for the special case that the inputs are always star temporal graphs two new approximation approaches. The framework provides the possibility to compute the $\Delta-$TVC based on different algorithms known in literature and the two new always star approximation algorithms. The known algorithms are a d-approximation algorithm [4] and a $(d-1)$-approximation algorithm [18] for always at most degree d temporal graphs and an exact (non-polynomial) dynamic programming algorithm [18] for arbitrary temporal graphs. For the verification of the stated runtimes and approximation ratios we experiment with the implementations on arbitrary instances. We are using the exact solution computed by the dynamic programming algorithm [18] for the verification of the stated approximation ratios. These experiments are on small instances since the exact solution is only computable in non-polynomial time. Moreover, we test the performance on real-life instances of the SNAP-library [28]. In the current literature the d- and $(d-1)$-approximation algorithms for always at most degree d graphs are also the best known polynomial computable solution for always star temporal graphs. As we are focusing on

this special case, we present two new approaches to approximate the Δ-TVC on them, achieving a $(2\Delta - 1)$-approximation ratio in $\mathcal{O}(T)$ and a $(\Delta - 1)$-approximation ratio in $\mathcal{O}(Tm\Delta^2)$. We show through experiments that these algorithms provide better ratios in shorter running time. Through these deliverables we are answering two main questions: firstly, how (Δ-)TVC can be approximated efficiently, secondly, with focus on always star temporal graphs, how to achieve a better approximation of (Δ-)TVC on them.

1.3 Structure

This thesis is structured in six chapters. After the introduction, we present the Preliminaries in Chapter 2. In Chapter 3 we give an overview of the related work in the temporal graph field and the known approximation ratios for (Sliding Window) Temporal Vertex Cover. Chapter 4 provides our main contributions, the presentation of our temporal graph generator and the framework for the TVC computation. Further, we explain the implementation of the known algorithms and present the two new approaches for always star temporal graphs together with the proofs for running time and approximation ratio. The experimental verification of the known algorithms are shown in Chapter 5 as well as the experimental comparison of the star algorithms with the current best solution. We conclude in Chapter 6 by summarizing and discussing our results and giving an outline of possible further work.

CHAPTER 2

Fundamentals

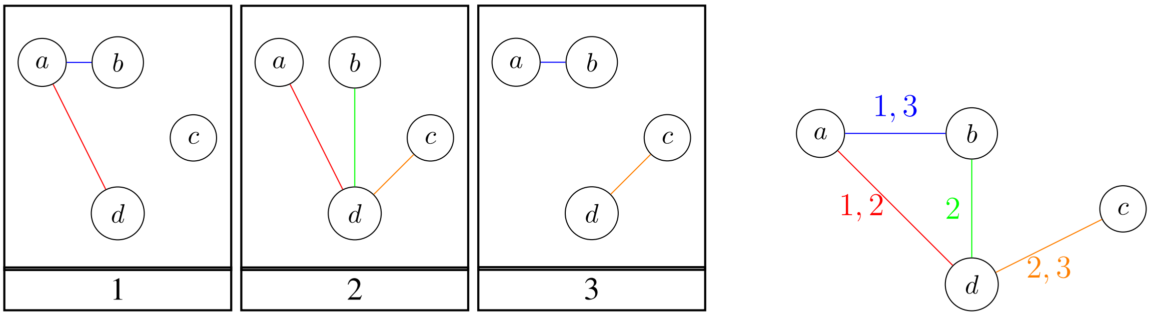
In this chapter, the basic definitions for (temporal) graphs, the problem definitions of (Δ-)TVC and approximation algorithms are presented. In addition, the notations used in this thesis are introduced.

2.1 Graph Preliminaries

A graph G is an abstract structure representing a set of objects together with their pairwise relationships. The objects are represented as a finite set of nodes V. Let n denote the total number of nodes. The relationships between them are represented by finite set of edges E. Let m denote the total number of edges. Hence, a graph $G = (V, E)$. An edge $e \in E$ consists of the connection of two vertices $u, w \in V$, we write $e = (u, w)$. In this thesis we only consider undirected graphs, meaning that edges have no direction, i.e. $e = (u, w) = (w, u)$. The nodes u and w are called the endpoints of e. The degree d of a node v is the number of edges connected to v. It is referred to as $d(v) = |\{(u, w) \in E | u = v \text{ or } w = v\}|$. We call two vertices u, w to be adjacent if $\{u, w\} \in E$. A common way to store such graphs is a $n \times n$ matrix $\mathcal{M}$, where the entry in row i and column j stores, whether vertices v_i and v_j are connected. In the undirected case this matrix is symmetrical. Another way of storing is an adjacency list of length n, with stores at index i the adjacent vertex indices of v_i.

2.2 Temporal Graph Preliminaries

In a temporal graph additionally to the above graph preliminaries the edges appear during a defined timespan in a discrete manner. This timespan is bounded by a maximal timestep T, the so-called lifetime of the graph. For better distinction, we can refer to graphs without the temporal component as static graphs.



(a) Subgraphs in each timestep

(b) Underlying graph with edge time labels

Figure 2.1: Visualizations of temporal graphs

Definition 1 (Temporal graphs)**.** *A temporal graph is a pair* (G, λ)*, where* $G = (V, E)$ *is an underlying (static) graph and* $\lambda : E \to 2^{\mathbb{N}}$ *is a time-labeling function which assigns to every edge of* G *a set of discrete-time labels.*

An edge $e \in E$ is called *active* at timestep t if it appears in that timestep, e.i. $t \in \lambda(e)$. To represent the temporal changes visually, there are two commonly used techniques, either by considering the subgraph of G at every timestep, see Figure 2.1a, or by visualizing the underlying structure of the graph and label each edge with the timesteps, in which it is active, see Figure 2.1b.

2.3 Temporal Vertex Cover

The studied problem in this thesis is the Temporal Vertex Cover, where one aims to cover each underlying edge by one of its endpoints through a set of vertices at certain timesteps. This problem is an adaptation of the classical (static) Vertex Cover Problem (VC). The VC aims to obtain a set of vertices such that at least one endpoint of every edge is included in the set. It can be applied in problems like civil and electrical engineering, protein sequencing or biochemistry [17]. In order to understand the temporal adjustment of the Vertex Cover problem, a definition of the classical problem is given first.

Definition 2 ((Static) Vertex Cover)**.** *Given an undirected graph* $G = (V, E)$ *and a parameter* $k \in \mathbb{N}$*,* $\exists$? *a subset* $V' \subseteq V$ *such that* $|V'| = k \land \forall \{u, v\} \in E : u \in V' \lor v \in V'$*.*

The minimum vertex cover problem aims to obtain the smallest possible number k of vertices in the cover. Since the minimum vertex cover problem can be reduced to Maximum Independent Set (MIS), which in turn can be reduced to the clique problem, it is NP-complete [17]. The MIS problem searches for a set of vertices such that no two vertices in the set are adjacent. If $\mathcal{I}$ is a MIS, then $(V - \mathcal{I})$ is a VC of the graph. Since if $u, w \in \mathcal{I}$, then $(v, w) \notin E$. Hence, $\forall\, e = (u, w) \in E$, $u \in (V - \mathcal{I})$ and/or $w \in (V - \mathcal{I})$. Every edge $e \in E$ is covered by a vertex $v \in (V - \mathcal{I})$, it is a VC.

Problems are considered as temporal if they provide a solution considering a temporal graph, e.i. all edge changes are known in advance. There are different ways of translating the Vertex Cover Problem into a temporal setting. One approach is to cover every appearing edge once over the whole lifetime of the graph. This problem is called Temporal Vertex Cover [4]. In the temporal versions of VC the solution consists of node appearances rather than nodes. A *node appearance* or *temporal vertex* (u, t) describes a node $u \in V$ at a certain timestep $t \in [0, T-1]$. An edge e is considered temporally covered by a vertex appearance (w, t), when w is an endpoint of e and e is active at t, e.i. $t \in \lambda(e)$. Hence, (w, t) covers all adjacent edges to w, which are active at t.

Definition 3 (Temporal Vertex Cover (TVC))**.** *A temporal vertex cover of (G, λ) is a temporal vertex subset S of (G, λ) such that every edge $e \in E(G)$ is temporally covered by at least one vertex appearance $(w, t) \in S$.*

For most applications, it may not be sufficient to cover each edge only once over the whole lifetime of the graph, e.g. if one considers monitoring tasks one wishes a regularly repeated manner of coverage. A commonly used way to face this is to consider a *Sliding Window* [4]. A window with specific size Δ considers a snapshot of the temporal graph covering Δ consecutive timesteps. It 'slides' over the lifetime, such that we have one window W_i starting in every timestep $i \in [0, T-\Delta-1]$, see Figure 2.2. The requirements are tightened by specifying that every edge should be temporally covered at least once in every window, in which the edge is active. A time window W_t is a set of time labels starting in t and covering all time steps until $t + \Delta - 1$. Let $E[W_t]$ denote the set of appearing edges in a time window, e.i. $E[W_t] = \{e \in E | \lambda(e) \cap W_t \neq \emptyset\}$. A vertex appearance (w, t) is called to be in a time window W_t if $t \in W_t$.

Definition 4 (Sliding Window Temporal Vertex Cover (SW-TVC))**.** *A sliding Δ-window temporal vertex cover of (G, λ) is a temporal vertex subset S of (G, λ) such that for every time window W_t and for every appearing edge $e \in E[W_t]$, e is temporally covered by a vertex appearance $(w, t) \in S$ in W_t.*

Through the parameterization with Δ, the sliding window model is more versatile and even includes the TVC as the special case $\Delta = T$, because this would be equivalent to considering the problem over the total lifetime T of a temporal graph. SW-TVC with window-size Δ is also referred to as Δ-TVC. An example of a 2-TVC is shown in Figure 2.2 on a temporal graph with lifetime 3. Since the edge (a, b) appears non-overlapping in both windows, node a needs to be included in two appearances ($t = 1$ and $t = 3$) to provide a valid cover.
The (minimum) SW-TVC problem is known to be NP-hard [4]. Techniques to find solutions for such problems in polynomial time are to approximate the solution or to restrict the inputs to a specific graph class in order to achieve better results. By using the properties of restricted inputs the solution quality and/or runtime can be improved. In terms of restriction there are two general approaches of adapting static graph classes into a temporal

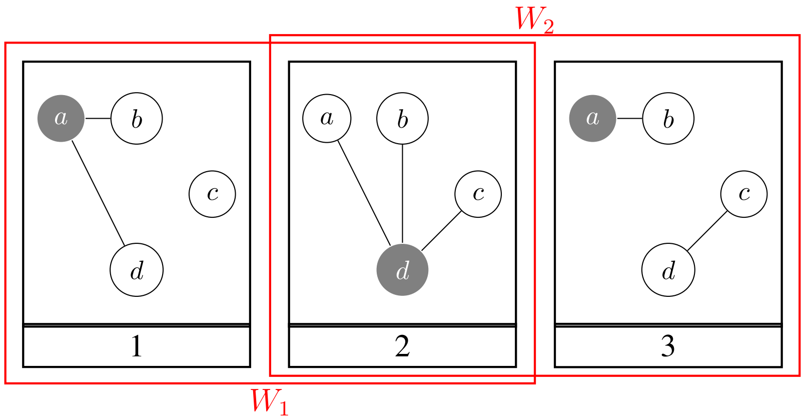


Figure 2.2: Sliding window temporal vertex cover

setting [4]. For a class $\mathcal{X}$ of static graphs a temporal graph (G, λ) is called $\mathcal{X}$ *temporal graph* or *underlying $\mathcal{X}$ temporal graph* if the underlying graph $G \in \mathcal{X}$, and it is called *always $\mathcal{X}$ temporal graph*, if each snapshot $G_i \in \mathcal{X}$ for every $i \in [T] = \{0, 1, \ldots, T-1\}$ [4]. In this thesis the focus is on always at most degree d temporal graphs, where every snapshot has at most degree d and always star temporal graphs, where every snapshot is a star graph, but the center can vary in each timestep.

2.4 Approximation Algorithms

Approximation algorithms are a tool to provide solutions to NP-hard problems in a polynomial runtime, where the solutions are guaranteed close to optimum. The deviation of the accuracy is bounded by the approximation ratio ρ.
For minimization algorithms, such as the Minimum TVC or SW-TVC, ρ is defined with respect to an objective function f. It refers to the ratio between the objective values of the provided and optimal solutions, which is not exceeded for any input I. Let the algorithm compute a solution $x(I)$, while $x^\star(I)$ is the optimal solution for that input, then ρ is defined as follows:

$$\rho = \sup_I \frac{f(x(I))}{f(x^\star(I))} .$$

For the Minimum TVC or SW-TVC the objective function f is the size of the computed cover.

CHAPTER 3

Related Work

This section gives an overview of the current state of research in temporal graphs and various researched problems on them. Then the focus is set on the temporal vertex cover problem (TVC) and its approximations. First, we summarize the general research on temporal graphs and temporal graph problems by describing the different notions of temporal behavior in graphs and the different extensions of the classical (static) path and non-path related problems into the temporal environment. With a focus on the temporal vertex cover considered in this thesis, we review the hardness proofs presented in the literature for the established temporal adaptations of it, the Temporal Vertex Cover and the Sliding Window Temporal Vertex Cover. Finally, we give an overview of the various existing exact and approximate algorithms for these extensions on arbitrary or specific graph classes.

3.1 Temporal Graphs

A graph is described as temporal when its structure changes over time. In contrast to static networks, temporal graphs offer the possibility to map systems with a changing topology over time, such as mobility, social or biological networks. In the literature temporal graphs appear under different names, which may refer to different underlying models.

The definition of temporal graphs applied here (see Def. 1) uses a popular model with fixed nodes and varying edges over time. Such a temporal graph consists of discrete subgraphs in which a subset of the underlying edges is active for each timestep during its lifetime. Besides *temporal* [18], [4], such graphs are also called *evolving* [14], [7], *dynamic* [16], [5], [19] or *time-varying* [38], [8], [42]. The description of these graphs are varying, with some using a sequence of the subgraphs [14], [7], others a time-labeling function assigning each edge the timesteps where it is active [4], [18], [8] or a collection of triplets (t, u, v) of two connected vertices u and v at a certain timestep t, a so-called link stream [42].

Some models for temporal graphs provide additional information regrading the dynamics, e.g. a model for path-related problems can contain latency descriptions (the required time to cross an edge) for each edge [8].
Temporal behavior in graphs is also found in the literature to describe models different from fixed vertices and discrete appearing edges. Leskovec et al. [27] define a time changing topology of the graph as a *graph over time*, where in each timestep new nodes are attached to the network and new edges created based on a defined network structure. These can be generated as spreading behaviors such as a Forest Fire Model [27] or recursive searches [40].
In the following course of the thesis every use of temporal graphs will refer to fixed nodes with varying edges, described over a time-labeling function as defined in Def. 1. Problems in this temporal setting are much studied. The focus in the literature is mainly on path-related problems. However, recently also non-path related problems have received more attention. A survey of related work on both types is presented below.

3.1.1 Path Related Temporal Graph Problems

In temporal graphs the feasibility of the path is affected by the time component as a path can only consist of edges appearing in an increasing (or at least non-decreasing) order. The impact of this restriction has been studied in various path or path-related problems.
In general there is a distinction between strict and non-strict paths referring to the amount of edges which can be crossed in a path in a single timestep [23], [15]. While in strict paths one can only cross one edge at a timestep, in non-strict paths one can cross multiple consecutive edges at the same timestamp.
For the classical shortest path problem, it is no longer sufficient to only consider the (weighted) shortest distance. There are different various criteria based on the temporal information of the graph to measure distance. They can be distinguished into four different types of the Temporal Shortest Paths: the earliest-arrival path, latest-departure path, the fastest path, and the shortest (distance) path [7], [44]. Wu et al. [44] propose four polynomial-time algorithms for them. Different algorithms under various considerations such as waiting time constraints have been studied since [21], [9], [29].
Path-related problems are much researched. The classical Traveling Salesperson Problem (TSP) is a well-known combinatorial optimization problem, which seeks to find the shortest possible tour that visits each city exactly once and returns to the starting city. In the context of temporal graphs, the Temporal TSP is a variant of the TSP that deals with time-dependent costs. Michail et al. [32] have shown that Temporal TSP with Costs one and two is APX-hard and have proposed a polynomial-time $(\frac{7}{4} + \epsilon)$-approximation algorithm.
The TSP problem is closely related to the graph exploration problem, while the first is an optimization problem aiming to minimize the total distance traveled, the latter is a process of visiting all nodes or edges in a systemic way. In the problem of temporal exploration an exploration visits every node and explores every edge (or the greatest possible number

of edges). In the case of star temporal graphs temporal exploration has been shown to be NP-complete, if every edge has at least 5 labels [3].

The problem of non-strict temporal exploration of a graph has been shown to be NP-hard on any underlying graph [12]. Even when the temporal diameter of the input graph is bounded by a constant c, it is NP-hard to approximate the problem with $\mathcal{O}(n^{1-\epsilon})$ or $\mathcal{O}(n^{\frac{1}{2}-\epsilon})$ ratios, when $c \geq 2$ [12], where the temporal diameter is the temporal extension to the classical diameter of a graph [2].

3.1.2 Non-path Related Temporal Graph Problems

Recently also non-path related temporal problems research has received more attention. This includes problems such as temporal graph coloring [31], Δ-cliques [42], temporal spanners [10] and the temporal vertex cover [4], [18] studied in this thesis. While for path-related problems the extension to the temporal setting is often naturally given through the consideration of feasible paths, i.e. edges in a correct chronological order, for non-path related problems the literature considers different techniques of adapting them into a temporal setting.

There are three commonly used ones [4], [18], [31], [19] [5], [42], [30]: the consideration of the problem over the whole lifetime, the sliding-window technique and the maintaining of a correct solution in every timestep. The first two have already been mentioned in Section 1 for the specific Vertex Cover problem, yielding TVC and SW-TVC problems, respectively.

The first intuitive adaptation used in the definition of TVC by Akrida et al. [4] considers a problem over the total lifetime of the graph. This is also found in other temporal graph problems such as the Temporal Graph Coloring by Mertzios et al. [31] by providing a union of vertex colors for each timestep, such that every vertex is colored properly at least once during the lifetime of the graph. Since these adaptations solve the problem over the entire lifetime, the relevance for applications may be reduced if the solution has to be queried frequently.

Therefore, these extensions are not considered as standalone in the literature, but as initial definitions followed by another technique, where the problem is considered over a temporal section of the graph, a time window, with a defined size $\Delta \in \mathbb{N}$. This technique, which has become more popular in the recent years, is called the *sliding window* technique. A solution is required for every time window of Δ consecutive time steps. It was introduced by Virad et al. [42], [41] to find contact patterns among high-school students by using temporal Δ-cliques. These are defined as a set of nodes and a time interval such that all pairs of nodes in this set interact at least once during each sub-interval of duration Δ. The authors also provide an exact algorithm to compute all maximal temporal Δ-cliques in $\mathcal{O}(2^n n^2 m^3 + 2^n n^3 m^2)$ time [42].

In sliding window temporal coloring [31], each vertex is assigned a color so that each appearing edge e must be colored correctly at least once during each time window of Δ successive time slots in which e is active. The extension is known to be NP-complete when considering more than one color, but it allows for an FPT (fixed-parameter tractable) algorithm parameterized by the number of vertices in $2^{\mathcal{O}(2^{n^2})}$ time [31].

Mertzios et al. [30] introduce the temporal matching problem, with the requirement that no edge appearance can be included, such that a vertex is matched twice in any Δ-window. The authors show the NP-completeness of Temporal Matching even if the underlying graph is a path by a reduction from Independent Set, and they propose an $\frac{\Delta}{2\Delta-1}$ approximation algorithm in $\mathcal{O}(Tm(\sqrt{n} + \Delta))$ time.
The sliding window extension and the consideration over the whole lifetime are closely related as the latter can be interpreted as the sliding window extension where $\Delta = T$.
The third extension technique is different from the previous adaptations, since the maintaining of a correct solution in every timestep does not consider the whole lifetime, but rather aims to minimize the update time while guaranteeing an optimal or good solution [6], [5].
Bhattacharya et al. [5] propose an algorithm for maintaining the vertex cover and the maximum matching during every time step. Such an adaptation provides a correct problem solution for every subgraph at a certain timestep, which may be required by some applications. For maximum matching the authors provide a data structure for maintaining a $(3 + \epsilon)$-approximation in $\mathcal{O}(\min(\frac{\sqrt{n}}{\epsilon}, \frac{m^{1/3}}{\epsilon^2}))$ amortized time per update [5]. For vertex cover they maintain a $(2 + \epsilon)$-approximation in $\mathcal{O}(\frac{logn}{\epsilon^2})$ amortized time per update [5].
If all changes over the lifetime are known in advance, solving a problem with this approach is equivalent to SW-TVC, where $\Delta = 1$. However, in this extension knowing the changes in the network structure is not necessarily a prerequisite and one only needs to receive edge updates in order to compute a new solution, since the aim of the extension is to maintain a solution.

3.2 Hardness of Temporal Vertex Cover

The classic/static Vertex Cover problem is known to be NP-hard [17]. In general this remains true for the adaptations TVC and SW-TVC [4]. This section gives an overview over known hardness proofs for the (SW-)TVC on special temporal graph classes.

For TVC Akrida et al. [4] have shown that the temporal adaptation TVC remains NP-complete even on the special case of star temporal graphs by reducing set cover to it. The set cover problem has a universe $U = \{1, 2, \ldots, n\}$ and a collection of $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of m subsets of $\mathcal{C}$ such that $\cup_{i=1}^{m} C_i = U$ as inputs. It searches for a subset $\mathcal{C}' \subset \mathcal{C}$ with the smallest cardinality such that $\cup_{\mathcal{C}_i \in \mathcal{C}'} C_i = U$. The authors take a general instance of set cover $(U, \mathcal{C})$ and construct an equivalent (underlying) star temporal graph (G, λ) for the computation of TVC. They set the lifetime $T = m$ and use $n + 1$ vertices. These split into the center vertex c and the leaves $v_1, \ldots, v_n$. The labeling function λ

assigns at each timestep $i \in [1, m]$ edges form the non-center nodes to the center according to $\mathcal{C}_i$, e.i. (c, v_j) is active $\forall\, j \in \mathcal{C}_i$. They prove, that $\mathcal{S}$ is a TVC on (G, λ) with $|S| \leq k$ iff there exist a set cover $\mathcal{C}'$ of $(U, \mathcal{C})$ with $\mathcal{C}' \leq k$.
In the case where the underlying graph G is a path or a cycle, i.e., when we consider a path/cycle temporal graph, Hamm et al. [18] show that TVC is solvable in polynomial time. For Δ-TVC, Akrida et al. [4] provide a polynomial time reduction from Δ-TVC to $(\Delta+1)$-TVC, proving that a $(\Delta + 1)$-TVC is at least as hard as Δ-TVC. As 1-TVC is equivalent to solving the vertex cover separately on T static graphs, it is at least as hard vertex cover. Since this is NP-hard on an arbitrary graph, Δ-TVC is NP-hard as well. Moreover, for any graph class $\mathcal{X}$ where VC is NP-hard, on always $\mathcal{X}$ temporal graphs Δ-TVC is also NP-hard.
The Exponential Time Hypothesis (ETH) states that there exists $\varepsilon < 1$ such that 3SAT cannot be solved in $\mathcal{O}(2^{\varepsilon n})$ time, where n is the number of variables in the input 3-CNF formula [22]. Assuming ETH, Akrida et al. [4] prove that there exists a constant ε such that SW-TVC cannot be solved in $f(T) \cdot 2^{\varepsilon n g(\Delta)}$ time for two (arbitrary) growing functions f and g. Further, the problem does not admit a Polynomial Time Approximation Scheme (PTAS) unless P=NP [4]. For the class of path/cycle temporal graphs Hamm et al. [18] propose a hardness proof for $\Delta \geq 2$, but show that a PTAS is admitted in this case.

3.3 Algorithms for Temporal Vertex Cover

The preceding section discussed that the problems of TVC and Δ-TVC are NP-hard, as well at the classical version of it. For the static vertex cover problem there exist several approximation algorithms [17] [1] [25]. These algorithms employ various techniques, including greedy, heuristic, memetic, or local search methods [17]. Additionally, Vertex Cover is closely related to other well-known problems, such as maximal matching. The maximal matching can be computed in $\mathcal{O}(m)$ time using a greedy algorithm and is known to provide a 2-approximation to both maximum matching and minimum vertex cover by using the endpoints of the maximal matching [5].
This section provides an overview of various exact and approximation algorithms from the literature that can be used to find solutions for TVC and Δ-TVC. The ideas and approximations for the algorithms are presented, but none of the algorithms are implemented and tested. The presented algorithms are categorized based on their temporal graph classes.

3.3.1 Arbitrary Graph Class

In the arbitrary temporal graph class no topological restrictions are made on the input. This section summarizes known algorithms from the literature on such inputs. Some of them can perform even better in terms of runtime on special graph classes, but also work in the general case.

An exact algorithm for SW-TVC provided by Akrida et al. [4] uses a dynamic programming approach with a runtime in $\mathcal{O}(T\Delta(n+m) \cdot 2^{n(\Delta+1)})$ [4]. This is asymptotically almost optimal, assuming ETH. The worst case runtime for this algorithm can be bounded for always star temporal graphs to $\mathcal{O}(T\Delta(n+m) \cdot 2^{\Delta})$ and for always C_k temporal graphs, where C_k the class of graphs having vertex cover number at most k, to $\mathcal{O}(T\Delta(n+m) \cdot n^{k(\Delta+1)})$.
Another approach by Hamm et al. [18] uses dynamic programming to solve the Δ-TVC in $\mathcal{O}(Tc^{\mathcal{O}|E(G)|)})$, where $c = min\{2^{d_\Delta}, \Delta\}$ and d_Δ the maximum Δ-window vertex degree. This algorithm also leads to a FPT algorithm for SW-TVC, which is single exponential in the number of edges running in $\mathcal{O}(Tc^{|E(G)|})$ where $c = min\{2^{\mathcal{O}|E(G)|)}, \Delta\}$. This approach also provides the possibility to only solve a partial graph input, which is used by the $d-1$ approximation for always degree at most d temporal graphs implemented in this thesis. The details of the algorithm and its implementation can be found in Section 4.3. < The literature also provides several ideas of approximation algorithms for SW-TVC on arbitrary graphs. Based on the idea that SW-TVC can be reduced to Set Cover, Akrida et al. [4] present two approximation algorithms which use set cover approximations. With Linear Programming, proposed by Vazirani [39], this leads to a $2k$-approximation for SW-TVC [4], where k is the maximum edge frequency ($k_{max} = \Delta$) defining the maximum appearance of the edge during an arbitrary window. Instead of Linear Programming it is also possible to use a greedy approach for set cover from Duh and Fürer[11] resulting in a $(\ln n + \ln \Delta + \frac{1}{2})$-approximation [4].

3.3.2 Always Degree at most d Temporal Graphs

For the temporal graph class where the degree at every snapshot is at most d, i.e. always degree at most d temporal graphs, two approximation algorithms for SW-TVC are provided. Akrida et al. [4] propose a d-approximation with runtime in $\mathcal{O}(mT)$ where m is the number of edges in the underlying graph G. The algorithm uses the idea to calculate SW-TVC on every possible single-edge temporal subgraph exactly in $\mathcal{O}(T)$ for every edge and take the union of the result.
Another approximation algorithm by Hamm et al. [18] is based on the approach to iteratively cover paths with two edges instead of single edges and chooses the middle vertex to be in the vertex cover. With that idea an overall approximation ratio of $(d-1)$ can be achieved. The runtime of this is in $\mathcal{O}(m^2T^2)$ where m is the number of edges in the underlying graph G.
Both of these algorithms are implemented in this thesis as they build the current state of the art for always star temporal graphs as well. The details of the functionality and implementation are described in Section 4.3.

3.3.3 Path/Cycle Temporal Graphs

Hamm et al. [18] show that TVC on instances with a path/cycle as their underlying graph is exactly solvable in polynomial time. Based on the idea of computing the Vertex Cover

in a (static) path graph in a greedy way, the proposed algorithm walks along the path from left to right and takes very second vertex. The running time is $\mathcal{O}(Tn)$.
For Δ-TVC on path/cycle temporal graphs Hamm et al. [18] propose a $(1 + \epsilon)$-approximation algorithm which runs in $\mathcal{O}(T(n + 1)^{\mathcal{O}\varepsilon^{-2})})$ time showing that the problem admits a PTAS.

CHAPTER 4

(SW-)TVC Approximation Algorithms

The methodology to answer the research question of how to approximate the (SW-)TVC efficiently in the general case and in the restricted case of always star temporal graphs consists of several steps and covers different impelmented components. A graph generator makes the generation of different temporal graph classes for the later experiments possible. The main component is the TVC-solver, which provides a framework to solve SW-TVC on an input graph with a given sliding window size and a certain solving algorithm. The framework covers several algorithms from the literature and the newly developed algorithms in the scope of this thesis for the restricted case of always star temporal graphs. The last component is a temporal graph visualizer for temporal graphs and the computed solutions on them for small instances. The interrelationships of the different components are shown in Figure 4.1.

The following subsections introduce the individual components and their implementations. Moreover, the last subsection presents new approximation algorithms for the restricted case of always star temporal graphs, whose implementation is also provided in the TVC solver.

4.1 Temporal Graph Visualizer

In the temporal graph analysis it is important to study the underlying structural properties and their temporal evolution. This manifests in the distinction between $\mathcal{X}$ temporal graphs and always $\mathcal{X}$ temporal graphs. Therefore, the temporal graph visualizer provides different visualization methods for temporal graphs and TVC, shown in Figure 4.2, based on the visualization methods stated in Section 1.

In Figure 4.2a the underlying network structure of a graph with edge labels according to λ are displayed, while in Figure 4.2b the same graph is split into the discrete timesteps. The temporal graph class can be seen in one of these visualizations, if it is a $\mathcal{X}$ temporal graph, $\mathcal{X}$ is underlying and therefore can be seen in the first display. If the graph is an always $\mathcal{X}$ temporal graph, such as the always star temporal graph in the Figure 4.2, the

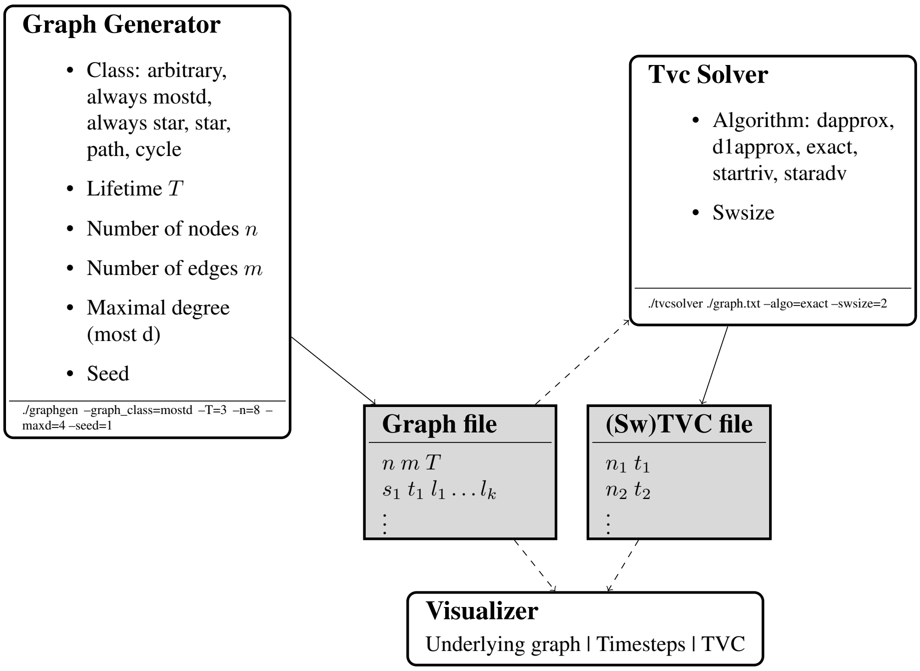


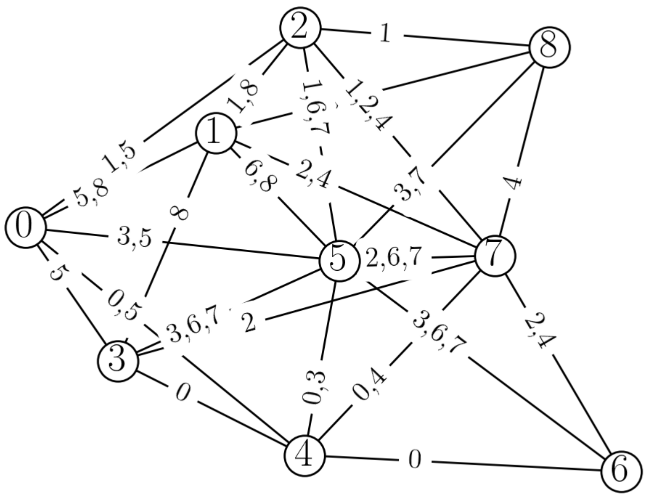
Figure 4.1: Interrelationships of the components

class displays itself in the evolutionary visualization. Moreover, this visualization provides the possibility to highlight a set of vertex appearances such as a TVC, as these consist of a vertex at a certain timestep. This is shown in Figure 4.2c.

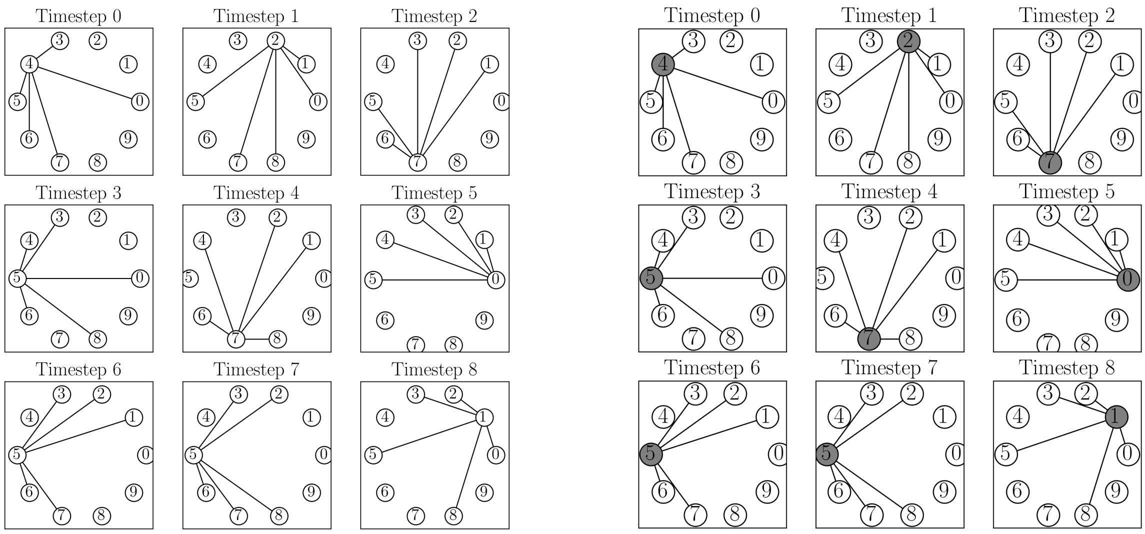
4.2 Temporal Graph Generation

Since this thesis aims to verify the complexity and compare the solution quality of different algorithm, we need various temporal graphs to test them. For classical graphs various algorithms are known in recent literature [37]. In this chapter we use these and extent them for the temporal graph generation. The temporal graphs must have different parameter ranges to test the behavior of the algorithms when one parameter, e.g., the number of edges, changes while the others remain constant. Moreover, a classification into to specific temporal graph classes is necessary to have a restricted input for the algorithms. In particular, the main restriction for the new approximation algorithms provided later is to always star temporal graphs. Besides this graph class, the generator includes the class of always at most degree d temporal graphs, to analyze the d-approximation and $d-1$-approximation algorithms provided by the literature on them. In terms of the underlying topologies of the graphs the generator can provide arbitrary and star temporal graphs, which can be used for runtime experiments.

For the required flexibility in the generated graphs, every generation should be configurable



(a) Visualization of the underlying graph with labels



(b) Visualization with separated timesteps

(c) Visualization with separated timesteps with the solution of 3-TVC marked in the timesteps

Figure 4.2: Visualizations of temporal graphs

over the number of nodes n and the lifetime T. To provide reproducible randomness every generator has a configurable random seed.

However, the main focus in this thesis is on the implementation and development of approximation algorithms and the presented generators serve the purpose of providing temporal graphs as input for their experimental analysis, not optimal generation of them.

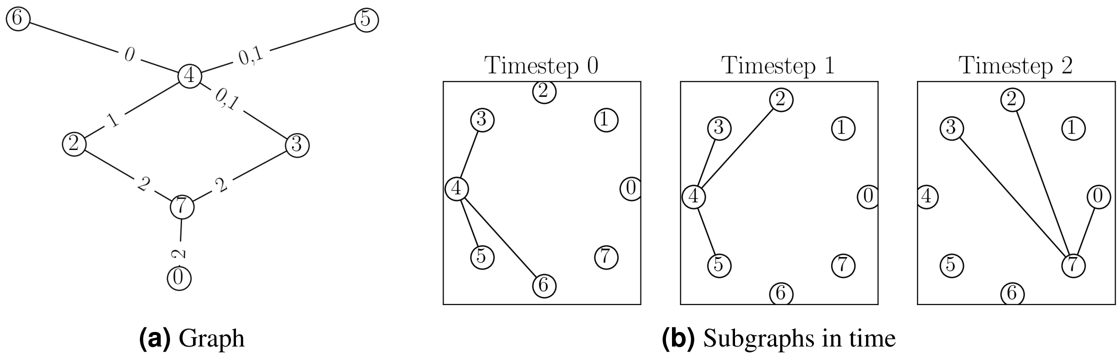


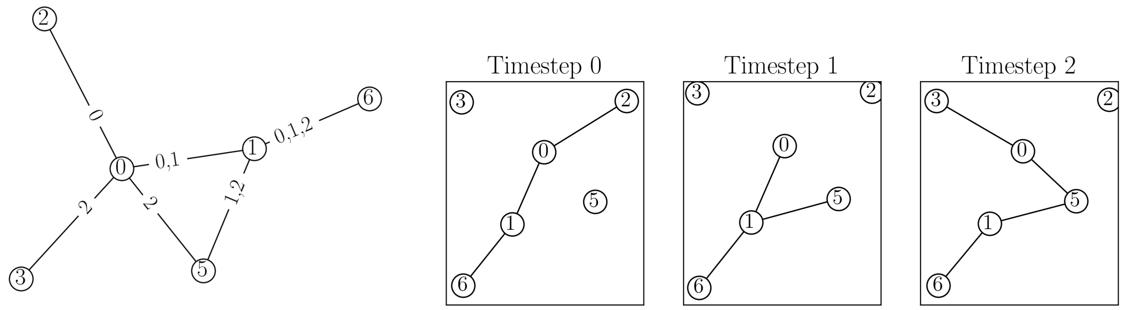
Figure 4.3: Always star temporal graphs

4.2.1 Always Star Temporal Graph Generation

In always star temporal graph topologies every subgraph at a timestep is a star. In addition to the configuration of the lifetime and node count, a requirement for the generation is to have a configurable maximal degree of the star, to enable a comparison between the algorithms for these graph classes and the always degree at most d graphs. The idea of the generation is to produce one star at each timestep t. To ensure that the configured maximal degree is reached in the beginning one timestep is chosen randomly, in which this degree must be reached. In every other timestep a random degree is chosen in $[0, d]$. Let the degree of the star at a timestep t be d_t. For the generation of the subgraph at t, $d_t + 1$ nodes get chosen randomly in $[0, n - 1]$. The first chosen node is the star center at t and edges are created to the remaining d chosen nodes. An example of a generated graph is shown in Figure 4.3.
The runtime for this generation is clearly in $\mathcal{O}(Tn)$, since the selection of $d + 1$ random nodes is in $\mathcal{O}(n)$ when calculating a permutation of all nodes and selecting the first $d + 1$ and the generation of maximal d edges in every timestep is clearly bounded by n as well.

4.2.2 Always Degree at most d Temporal Graph Generation

When generating temporal graphs with always at most degree d it is important to ensure that no degree exceeds the maximum degree allowed. However, to obtain a good graph for analysis, it is also desirable that the maximum degree is reached in at least one time step.
The generation is based on an $G(n, p)$ Erdős-Rényi model [34] in every timestep, but provides additional monitoring of the node degrees to never generate an edge, when the degree of one of the considered nodes already reached d. To ensure that the maximum degree is reached, the generator randomly selects a time step in which the maximum degree is enforced and the edge probability is adjusted accordingly. A generated always most d graph is shown in Figure 4.4. Moreover, the parameter p of the $G(n, p)$ is no input parameter, but rather chosen in dependence of d.



(a) Graph **(b)** Subgraphs in time

Figure 4.4: Always degree at most d temporal graphs

The runtime of the $G(n, p)$ model [34] is $\mathcal{O}(n + m)$, and we generate a separate subgraph at every timestep. Hence, the overall runtime of the generation is $\mathcal{O}((n + m)T)$. However, caused by the enforcement of the maximal degree in one timestep, this progress might have a large constant in this timestep and therefore take significantly longer than in the other timesteps.

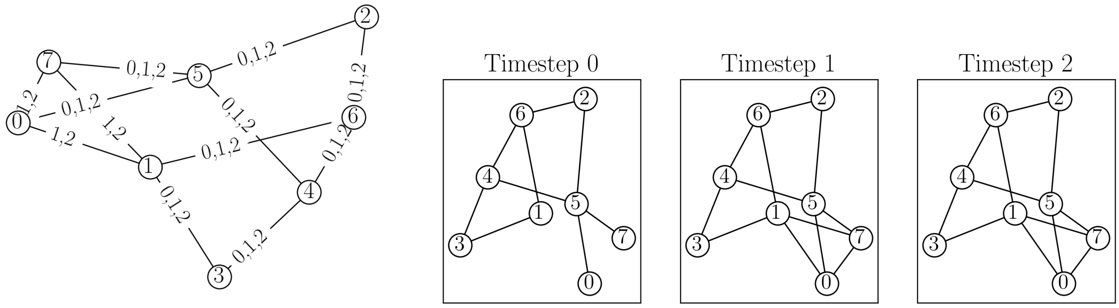
4.2.3 Underlying Topology Temporal Graph Generation

Temporal graphs with an underlying topology $\mathcal{X}$ refer to a topology in static graphs with the addition of having discrete time labels for each edge, describing when the edge is active. The basic idea to generate these, is to generate a static graph with this topology and add randomly discrete time labels to the edges.

For runtime experiments, we want to generate arbitrary temporal graphs with configurable number of egdes m. For arbitrary static graphs a well known model to generate $G(n, m)$ graphs is the Erdős-Rényi model [37]. For generating these static graphs we use the KaGen [37] library. After the generation we assign each edge a random size and fill the vector with time labels. An example of a generated graph is shown in Figure 4.5.

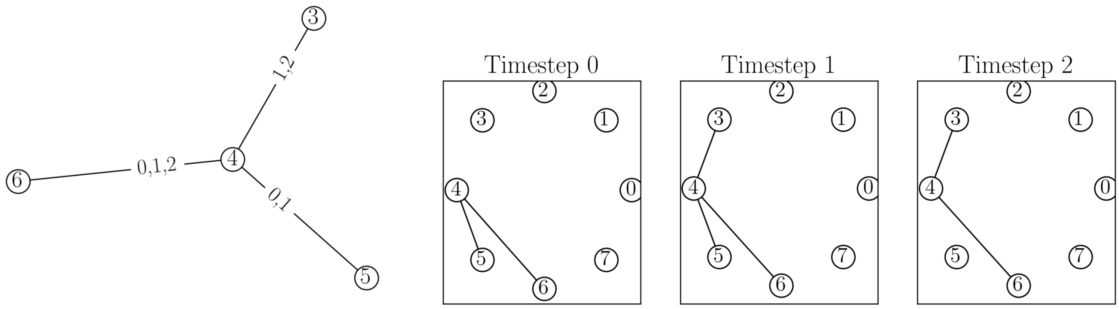
The underlying star generation of the static graph is created in the same way as for always star temporal graphs, see Figure 4.6, by randomly choosing $d+1$ nodes and connecting the first one with the others. Then each edge gets assigned a random number of time labels. In this model d can either be configured or is chosen randomly.

The runtime for this generator type is the runtime of the underlying graph with class $\mathcal{X}$ generator plus the insertion of labels for each edge. We implemented this by iterating over the edges. Then we permute a vector of all possible labels and chose a random amount of them. This ensures, that no label is picked twice. Since there can be at most T labels per edges, the runtime is in $\mathcal{O}_{Gen_{\mathcal{X}}} + \mathcal{O}(Tm)$.



(a) Graph **(b)** Subgraphs in time

Figure 4.5: Arbirtary temporal graphs



(a) Graph **(b)** Subgraphs in time

Figure 4.6: Star temporal graphs

4.3 Framework Design

The TVC-solver framework provides the possibility to compute SW-TVC on different input graphs with defined window size and algorithms to solve SW-TVC. Therefore, it holds a temporal graph data structure to store the input graph and implementations of several algorithms to solve SW-TVC, including an exact algorithm, d- and $d-1$-approximations and two always star approximation algorithms introduced in Section 4.4.

4.3.1 Temporal Graph Data Structure

The graph data structure holds an array of all the undirected edges, where an edge consists of two vertices and an array of time labels during the edge is active. The edges are stored in the vector $edges$. Besides this vector, the data structure provides two ways of accessing the edges. To provide the possibility to access all edges of a node, the data structure holds an adjacency list adj, in which the respective edge indices can be retrieved for each node. Every edge index is listed twice in this map, once at each endpoint, as the graph is undirected. Additionally, some algorithms require to access all edges of a particular timestep. To provide this, a vector of length T stores all edge indices active at a timestep. This can lead to edge indices being referred to at multiple timestep. As this is not needed in all algorithms, this vector is only initialized for algorithms using it. The data structure is visualized in Figure 4.7.

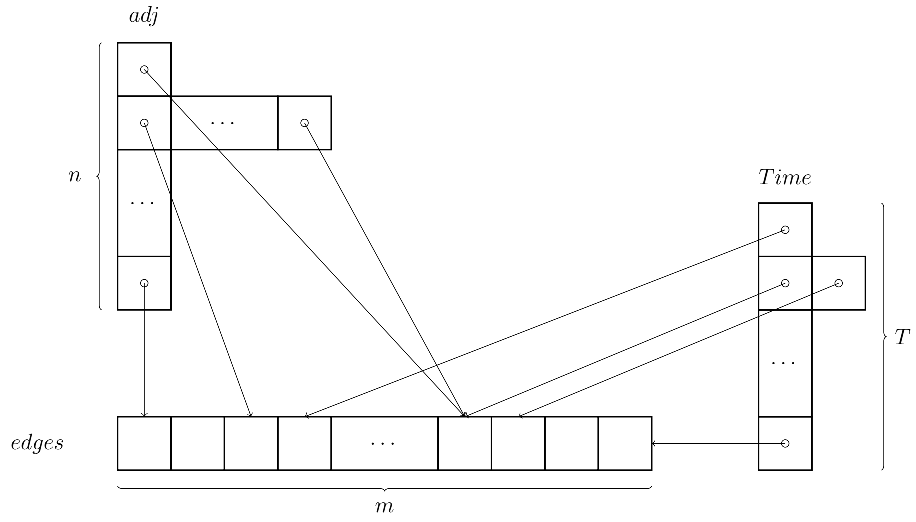


Figure 4.7: Temporal graph data structure

This provides the possibility to store temporal graphs efficiently, since the space complexity is in $\mathcal{O}(adj)+\mathcal{O}(edges)+\mathcal{O}(Time) = \mathcal{O}(n+2m)+\mathcal{O}((2+T)\cdot m)+\mathcal{O}(Tm) = \mathcal{O}(n+Tm)$.

4.3.2 Implementation of the d-Approximation Algorithm

The d-approximation algorithm for Δ-TVC [4], where d is the maximal degree appearing in any timestep, is based on the idea, that on a single edge graph TVC can be solved optimally in polynomial time. In particular the algorithm scans over all uncovered windows and selects one endpoint of the latest appearance of the single edge in the window to cover it. On a graph with more than one edge, this process is repeated individually for every edge and a union of these solutions builds an overall approximate solution. The runtime of the algorithm is in $\mathcal{O}(Tm)$. The approximation ratio is d, since in the worst case scenario for every vertex appearance included in the optimal solution all outgoing edges could be covered by the other endpoint in the solution calculated by the algorithm. Therefore, this solution is at most d times the optimal solution in size. The algorithm is implemented straightforward from the pseudocode in [4], which is presented below.

4.3.3 Implementation of an Exact Algorithm

Since SW-TVC is known to be NP-hard, an exact algorithm can only be a non-polynomial one. A dynamic-programming exact algorithm is provided by Hamm et al. [18], solving

Algorithm 1 SW-TVC on single-edge temporal graphs from [4]

Input: A temporal graph (G, λ) with lifetime T, where $G = (V, E)$, and a natural $\Delta \leq T$
Output: A temporal vertex cover $\mathcal{X}$ of (G, λ)

$\mathcal{X} := \emptyset$ $t = 1$ **while** $t \leq T - \Delta + 1$ **do**
  if $\exists r \in [t, t + \Delta - 1]$ *such that* $(u, v) \in E_r$ **then**
    choose maximum such r and add (u, r) to $\mathcal{X}$
    $t = r + 1$
  end
  else
    $t = t + 1$
  end
end
return $\mathcal{X}$

Algorithm 2 d-approximation of SW-TVC on always degree at most d temporal graphs from [4]

Input: A temporal graph (G, λ) with lifetime T, where $G = (V, E)$, and a natural $\Delta \leq T$
Output: A temporal vertex cover $\mathcal{X}$ of (G, λ)

for $i = 1$ *to* T **do**
  $\mathcal{X}_i := \emptyset$
end
foreach $e = (u, v) \in E$ **do**
  Compute the optimal solution $\mathcal{X}^e$ of the problem for $(G[\{u, v\}], \lambda)$ by Algorithm 1
  for $i = 1$ *to* T **do**
    $\mathcal{X}_i = \mathcal{X}_i \cup \mathcal{X}^e$
  end
end
return $\mathcal{X}$

SW-TVC in $\mathcal{O}(T\Delta^{\mathcal{O}(m)})$ time. The main idea is to split the problem into subinstances. These subinstances are the variation of SW-TVC called Partial Δ-TVC, which has the same requirements as SW-TVC but every edge has a range of uncovered windows, for which the solution should apply. Additionally, solution vertices (v, t) are only allowed in a defined time range described via start-/end-point in T [18]. The Partial Δ-TVC involves additionally to (G, λ) two functions $h : E(G) \to [T]$ and $l : E(G) \to [T]$ assigning each edge its highest and lowest uncovered windows. The aim is to find a solution to cover every edge given the range of uncovered windows of it. We developed Algorithm 3 based on the description in [18].
The exact algorithm uses a dynamic-programming table f, where every entry corresponds to exactly one such Partial Δ-TVC problem and stores the size of the optimal solution to it. The lowest uncovered window is modified to look at different instances. This is achieved by indexing the table with tuples $(t, x_1, \ldots, x_m)$, where $l'(e) = t + x_e$. The index is split into the number $t \in [0, T - \Delta + 1]$ of the starting window W_t and $x_i \in [0, \Delta]\ \forall\ i \in [m]$. A minimal sized solution is called witness.
In every recursion one optimal witness for one sub-instance $(t, x_1, \ldots, x_m)$ is found. If all edges e are covered in window W_t, $l'(e) > t$ and hence $x_e \neq 0$. In this case we move on to the next window, since

$$f(t, x_1, \ldots, x_m) = f(t + 1, x_1 - 1, \ldots, x_m - 1)$$

If there are still uncovered edges in the window W_t, there is at least one $x_i = 0$, which needs to be covered to get to the next recursion. If there are multiple $x_i = 0$ the algorithm always selects the lowest such i to cover next. To find an optimal witness in this step the authors use *edge configurations* and choose the best one to cover the edge.
An edge configuration γ of edge e_i consists of all adjacent edges at one endpoint and one timestep. Formally, if $e_i = (u, w)$, Then

$$\gamma(e_i, t)_v = \{e \in E(G) | v \in e_i \cap e, t \in \lambda(t)\}$$

is the edge configuration incident to e_i at t in endpoint v. The set of all edge configurations of e_i in endpoint t considers all timesteps, where e_i is active, and represented by

$$\gamma(e_t)_v = \{\gamma(e_i, t)_v | t \in \lambda(e_i)\}$$

Each of these configurations corresponds to one vertex appearance (v, t_i), where v is the endpoint the configuration is based on and t_i is the time, where this configuration appears in the time window. Hamm et al. [18] proved that it is sufficient to consider the latest appearance of an edge configuration, in case there are multiple. This vertex appearance covers e_i. Moreover, the edge configuration stores all edges, which are covered in case this corresponding vertex appearance is added to the solution. In case an edge configuration is chosen to cover e_i, we update all x_ks of the index according to

$$x'_k = \begin{cases} t_i - t, & \text{if } k = i \\ \max(x_k, t_i - t), & \text{if } k \in \gamma(e_i, t)_v \\ x_k, & \text{otherwise} \end{cases}$$

Algorithm 3 Exact computation of a (partial) SW-TVC on temporal graphs based on the description in [18]

Input: Compute a partial Δ-TVC for temporal graph (G, λ) with a dynamic programming table f, and index (t, x), and the highest uncovered window for each edge h
Output: The minimum size of the partial Δ-TVC (t, x, h)

Function `SolvePartialSWTVC(`$G, \lambda, \Delta, f, t, x, h$`)`**:**
  if $f(t, x) \neq \emptyset$ **then**
    return $f(t, x)[0]$
  end
  `// Case 1: All edges are covered in window` W_t
  if $\forall\, xi \in x : i \neq 0$ **then**
    `// Check if all edges are covered, trivial case`
    if $\forall\, i \in [0, |x| - 1] : x[i] > h[i]$ **then**
      $f(t, x) = (0, null, null)$
    end
    $x'[i] = x[i] - 1 \;\forall\, i \in [0, |x| - 1]$
    c = $SolvePartialSWTVC(f, t + 1, x', h)$
    $f(t, x) = (c, null, (t + 1, x'))$
    return c
  end
  `// Case 2: At least one edge is not covered in window` W_t
  i = smallest i, where $x[i] == 0$
  if *Edge i does not appear in W_t* **then**
    $x'[j] = x[j] \;\forall\, j \neq i \in [0, |x| - 1]$
    $x'[i] = x[i] + 1$
    c = $SolvePartialSWTVC(f, t, x', h)$
    $f(t, x) = (c, null, (t, x'))$
    return c
  end
  $best = (int_max, null, null)$
  foreach $ec \in$ *All latest configurations of i in W_t* **do**
    (v, ti) = Vertex appearance corresponding to ec
    $x'[i] = ti - t$
    $x'[j] = \max(x[j], ti - t) \;\forall\, j \in ec$
    $x'[k] = x[k] \;\forall\, k \neq i \notin ec$
    c = $SolvePartialSWTVC(f, t, x', h)$
    if $c + 1 < best[0]$ **then**
      $best = (c + 1, (v, ti), (t, x'))$
    end
  end
  $f(t, x) = best$
  return $best[0]$

Input: A dynamic programming table f with the solution, and the start index (t, x)
Output: A minimum size Δ-TVC for the given index
Function `ExtractSolution(`f, t, x`)`**:**
    $\mathcal{X} := \emptyset$
    repeat
        $(c, \mathbf{v}, \mathbf{i}) = f(t, x)$
        if $\mathbf{v} \neq null$ **then**
            $\mathcal{X} = \mathcal{X} \cup \{\mathbf{v}\}$
        end
        $(t, x) = \mathbf{i}$
    until $\mathbf{i} == null$;
    return $\mathcal{X}$
Input: A temporal graph (G, λ) with lifetime T, where $G = (V, E)$, and a natural $\Delta \leq T$
Output: A minimum size Δ-TVC
Function `ExactSWTCV(`G, Δ`)`**:**
    $m = |E|$
    Initialize f based on T, m, Δ
    $x[i] = 0 \; \forall \; i \in [0, m-1]$
    $h[i] = T - \Delta + 1 \; \forall \; i \in [0, m-1]$
    $SolvePartialSWTVC(G, \lambda, \Delta, f, 0, x, h)$
    return $ExtractSolution(f, 0, x)$

such that $f(t, x'_0, \ldots, x'_m)$ corresponds to the sub-problem similar to $f(t, x_0, \ldots, x_m)$ except that all edges are covered, which were covered if (v, t_i) of the edge configuration was added to the solution.
In the description of the algorithm by Hamm et al. [18] not all implementation details are provided. In particular, those regarding start and default cases, choosing the optimal solution and storing the witnesses of an entry.
We begin the process with the tuple $(0, 0, \ldots, 0)$, referring to the state, where all edges are uncovered in the first window, and a fixed $h(e) = T - \Delta + 1$, the last window, to calculate the Partial Δ-TVC equal to the Δ-TVC problem. The termination case is $l'(e_i) > h(e_i)\ \forall\ i \in [m]$, since no edge needs to be covered, therefore the solution is empty and can be returned.
To find the optimal solution for one $f(t, x_0, \ldots, x_m)$, we test all configurations of e_i in both endpoints and choose the vertex appearances based in the configuration resulting a minimal solution. The witness of $f(t, x_0, \ldots, x_m)$ is then the witness of $f(t, x'_0, \ldots, x'_m)$ + (v, t_i).
In terms of implementation our dynamic programming table f not only stores the size of the optimal solution, but also the new chosen vertex appearance to add to the witness and the index to the next considered entry, to provide the possibility to reconstruct the witness of this entry. By doing so the overall solution of $f(0, 0, \ldots, 0)$ can be easily reconstructed in the end.
Since the dynamic table has space complexity in $\mathcal{O}((\Delta+1)^m T)$ and not all entries are computed, we only store an indexed map instead to be able to calculate large temporal graphs.

4.3.4 Implementation of the $d-1$-Approximation Algorithm

While the d-approximation considers every edge in a separated manner and solves a single edge optimally, Hamm et al. [18] provide a $d-1$-approximation, based on the idea that not every edge should be considered separately, but as long as there are at least two connected uncovered edges they should be considered as a connected triangle. Such a triangle the authors call P_3. The idea is to split the graph in such P_3s and solve them optimally. Our developed pseudocode based on the description in [18] can be found in Algorithm 4.
The algorithm therefore processes the graph in two phases. The first phase runs while there is still an uncovered P_3 and the second phase if only single edges still need to be covered.
In the first phase, the authors select such an uncovered P_3 and build independent sub-instances of Partial Δ-TVC to solve them optimally. Therefore, we consider a set S of all timesteps, where this P_3 is uncovered. This set is split into independent subsets S_i. The independence is provided by a gap of at least $2\Delta - 1$ between the highest time label in S_{i-1} and the lowest time label in S_i, since then no two windows with the defined size Δ, one having a time labels in S_i and the other in S_{i+1}, overlap. On every of these S_i a Partial Δ-TVC is build and solved optimally with the exact algorithm in 4.3.3. The instance of the Partial Δ-TVC is provided by $(G[P_3], \lambda'_i(P_3), l_i(P_3), h_i(P_3))$. The graph is restricted to the three node and two edges of the P_3, while $\lambda'_i(P_3)$ assigns the two edges all their time labels in any window, where at least one time label is in S_i. Formally, the range between $[\min S_i - \Delta + 1, \max S_i + \Delta - 1]$ is considered. Similar, the lowest uncovered

Algorithm 4 $d-1$-approximation of SW-TVC on always degree at most d temporal graphs based on the description in [18]

Input: A temporal graph (G, λ) with lifetime T, where $G = (V, E)$, and a natural $\Delta \leq T$
Output: A temporal vertex cover $\mathcal{X}$ of (G, λ)

$\mathcal{X} := \emptyset$
Initialize $\mathcal{U}$ with all edge appearances
// Check if there is an uncovered P_3 in some time window W_i
foreach $e1 = (n1, v) \in E \; \forall \, n1 \in V$ **do**
    foreach $e2 = (u, n2) \in E \; \forall \, n2 \in V$ **do**
        $\mathcal{S} = \mathcal{U}(e1) \cap \mathcal{U}(e2)$
        if $\mathcal{S} \neq \emptyset$ **then**
            // Found a P_3
            SolveSubinstances($\mathcal{S}$, $\{e1, e2\}, \mathcal{U}$, $\mathcal{X}$)
        end
    end
end
// Check if there are any uncovered edges left in some time window W_i
foreach $e \in E$ **do**
    if $\mathcal{U}[e] \neq \emptyset$ **then**
        SolveSubinstances($\mathcal{U}[e]$, $\{e\}, \mathcal{U}$, $\mathcal{X}$)
    end
end
return $\mathcal{X}$

Input: The sub-problem, consisting of the uncovered appearances $\mathcal{S}$ of the edges in E, where $\mathcal{U}$ are all uncovered appearances, and $\mathcal{X}$ is the solution set
Output: No direct output, note that the function edits $\mathcal{U}$ and $\mathcal{X}$

Function SolveSubinstances($\mathcal{S}$, E, $\mathcal{U}$, $\mathcal{X}$)**:**
    Split $\mathcal{S}$ into subsets $\mathcal{S}_k$, such that $\max(\mathcal{S}_k) - min(\mathcal{S}_{k+1} \geq 2\Delta - 1 \; \forall \, k)$
    foreach $\mathcal{S}_k$ **do**
        $l_m in = min(\mathcal{S}_k) - \Delta + 1$ or 0 if first term is negative
        $l_m ax = max(\mathcal{S}_k) + \Delta - 1$
        $\lambda'_i(e) = \lambda(e)$ restricted to $[l_m in, l_m ax] \; \forall \, e \in E$
        $(G[V(E), E], \lambda'_i)$
        $x[e] = 0 \; \forall \, e \in E$
        $h[e] = \max(S_k) \; \forall \, e \in E$
        Initialize f based on T, m, Δ
        $SolvePartialSWTVC(G[V(E), E], \lambda'_i, f, l_m in, x, h)$ from Algorithm 3
        $\mathcal{X}_E = ExtractSolution(f, l_m in, x)$ from Algorithm 3
        $\mathcal{X} = \mathcal{X} \cup \mathcal{X}_E$
    end
    Remove all appearances of $e \in E$ in range $[min(\mathcal{S}_k), max(\mathcal{S}_k)]$

window for the edges is $l_i(P_3) = \min S_i - \Delta + 1$ and the highest uncovered window is $h_i(P_3) = \max S_i$. The union of the solutions provided by the exact algorithm are added to the overall solution.
In the second phase there are no P_3s left, but there still may be uncovered single edges. For each still uncovered edge e independent sub-instances are provided in the same manner as in the first phase. In the beginning the appearances of the single edge are split in independent subsets S_i. The Partial Δ-TVC $(G[e], \lambda_i'(e), l_i(e), h_i(e))$ is build on the graph restricted to the single edge e, where $\lambda_i'(e)$, $l_i(e)$ and $h_i(e)$ are restricted similar to the first phase except that only one edge is considered. These sub-instances are solved optimally again. The union of the solutions with the solution from the first phase build the overall solution.
The runtime of this algorithm is in $\mathcal{O}(T^2m^2)$, since the number of P_3s is bounded by $\mathcal{O}(Tm^2)$, the number of single edges by $\mathcal{O}(Tm)$, every restricted exact solution is computable in $\mathcal{O}(T)$. The approximation ratio $d-1$ is proved via the idea to break the temporal graph into sub-instances of two edge paths, where the middle vertex covers this optimal. Since at least two edge appearances can not be covered by fewer instances, the approximation ratio is at most $d-1$.
For the implementation the details of the detection of such uncovered P_3 and uncovered edges in general is not provided through the paper. For the consideration of uncovered edges, we store a mark for all uncovered edge appearances to use for the detections. Then the detection of P_3 can be handled via the center node, i.e. comparing uncovered edges of a node to check whether they have common appearances.

4.4 SW-TVC on Always Star Temporal Graphs

In this section, the problem is restricted to the special case of always star temporal graphs. First, we deduce that TVC remains NP-complete on this temporal graph class and then present two approximation algorithms to improve SW-TVC calculation. The current state of the art to solve them, would be using the always degree at most d algorithms. The algorithms devised and presented in this thesis can improve the solution quality as they use the main idea, that at most one (easily detectable) node in every timestep is included in the cover, i.e. the star center in that timestep. The first trivial algorithm includes the star center in every timestep and can approximate the Δ-TVC in $\mathcal{O}(T)$ time with a $2\Delta - 1$ approximation ratio. A more advanced technique is to check at a timestep if all edges can be covered by other instances in this window, resulting in a $\Delta - 1$- approximation, which is computable in $\mathcal{O}(Tm\Delta^2)$ time.

4.4.1 Hardness Conclusion

Akrida et al. [4] proved that TVC is NP-complete on star temporal graphs by showing that set cover is reducible to it. The problem in fact remains NP-hard on always star temporal graphs: star temporal graphs can be seen as the subclass of always star graphs, where the star center is the same in every timestep. Therefore, this already implies that solving it has to be at least as hard. Hence, solving TVC on always star temporal graphs is NP-complete. Therefore, the general SW-TVC is also NP-complete as TVC is the sub-problem of SW-TVC where $\Delta = T$. However, Akrida et al [4] provide an FPT algorithm parameterized by the sliding window size Δ, solving it optimally in $\mathcal{O}(T\Delta(n+m) \cdot 2^{\Delta})$. This thesis provides in the following Algorithms 5 and 6, which are polynomial-time exact algorithms for the cases $\Delta \leq 1$ and $\Delta \leq 2$ and approximation algorithms for higher Δ.

4.4.2 Trivial Algorithm

The trivial idea to solve (Δ-)TVC problem for always star classes is to include the star center in every time step, where at least one edge is active, in the cover. To detect the star center for timesteps with at least two active edges, we compare any two edges to identify the common vertex. In case only one edge is active in the timestep, both vertices are valid to be considered as the star center, since through its inclusion all edges in the timestep are covered. For an only edge $e = (v, w)$ at some timestep we use v as star center. This is realized in Algorithm 5.

Algorithm 5 Trivial always star algorithm

Input: A temporal graph (G, λ) with lifetime T, where $G = (V, E)$, and a natural $\Delta \leq T$
Output: A temporal vertex cover $\mathcal{X}$ of (G, λ)

$\mathcal{X} := \emptyset$
foreach t *in* T **do**
    if *there is an edge* $e = (u, w)$ *in* E_t **then**
        if E_t *has more than two edges* **then**
            $\mathcal{X} = \mathcal{X} \cap \{(center_t, t)\}$
        else
            $\mathcal{X} = \mathcal{X} \cap \{(u, t)\}$
        end
    end
end
return $\mathcal{X}$

Theorem 1. *The trivial always star algorithm approximates Δ-TVC on always star temporal graphs with $T \geq \Delta$ and $\Delta \geq 2$ with ratio $2\Delta - 1$ in $\mathcal{O}(T)$ time.*

Proof. To prove Theorem 1, we need to prove the running time and approximation ratio of Algorithm 5. The running time of the algorithm is in $\mathcal{O}(T)$, since we loop over all timesteps and the detection of the star center is in $\mathcal{O}(1)$, as we compare at most two edges. For the approximation ratio, we need to consider the worst case and compare the solution computed by Algorithm 5 with the optimal Δ-TVC. The worst case for the trivial algorithm is, when all edges are active in all time steps, because in that case our algorithm includes the (static) star center in every timestep, while only one coverage per window is required for the minimum TVC. Formally, let the size of the optimal Δ-TVC be $x^\star$ and the size of our solution be x. Then $x^\star = \lfloor \frac{T}{\Delta} \rfloor$ and $x = |\mathcal{X}| = T$. We need to show that the approximation ratio $\frac{x}{x^\star}$ is bounded:

$$\frac{T}{\lfloor \frac{T}{\Delta} \rfloor} \leq 2\Delta - 1 \tag{4.1}$$

To break this down, we consider a representation of the lifetime in terms of the window size: $T = c \cdot \Delta + d$, where $c, d \in \mathbb{N}$, $c \geq 1$ and $0 \leq d \leq \Delta - 1$. Then, we can derive the modulo classes $R_d \in \{R_0, \ldots, R_{\Delta-1}\}$ for the denominator of the equation. Since the most round-off is achieved in the $R_{\Delta-1}$ class, a value in this class maximizes the ratio $T/\lfloor \frac{T}{\Delta} \rfloor$. In this class T can be represented as $T = c \cdot \Delta + (\Delta - 1) = a \cdot \Delta - 1$ with $a = c+1, a > 1$ and hence $\lfloor \frac{a \cdot \Delta - 1}{\Delta} \rfloor = a - 1$. To derive the maximum value of the left-hand side in equation (4.1), we consider any value $T = a \cdot \Delta - 1$ and show that the next larger element of the class $R_{\Delta-1}$, represented as $T = (a+1) \cdot \Delta - 1$, does not lead to a larger approximation ratio.

$$\frac{a \cdot \Delta - 1}{a - 1} \geq \frac{(a+1) \cdot \Delta - 1}{a}$$

$$a \cdot \Delta - 1 \geq \frac{(a+1)(a-1) \cdot \Delta - a + 1}{a}$$

$$a \cdot \Delta - 1 \geq \frac{(a^2-1) \cdot \Delta - a + 1}{a}$$

$$a \cdot \Delta - 1 \geq (a - \frac{1}{a}) \cdot \Delta - 1 + \frac{1}{a}$$

$$a \cdot \Delta - 1 \geq a \cdot \Delta - 1 - \frac{\Delta - 1}{a}$$

$$0 \geq 1 - \Delta$$

Which clearly holds, since $\Delta \geq 2$ as stated in Theorem 1. Therefore, the T with the smallest valid value of a, which is $a = 2$, leads to the maximum value of $\frac{T}{\lfloor \frac{T}{\Delta} \rfloor}$.

$$\frac{2\Delta - 1}{\lfloor \frac{2\Delta - 1}{\Delta} \rfloor} = \frac{2\Delta - 1}{1} = 2\Delta - 1$$

Hence, equation (4.1) is true and Algorithm 5 has an approximation ratio of $2\Delta - 1$. □

For the case $\Delta = 1$, the algorithm provides the optimal solution. Which makes sense, since in the 1-TVC every snapshot is considered separately and the optimal solution consists of every star center similar to the computed solution by Algorithm 5.

4.4.3 More Advanced Algorithm

The second algorithm for always star temporal graphs is based on the idea of maintaining a table to monitor every sliding window and checking if all edges in a certain timestep t can be covered by other appearances in the window. In that case the star center of t does not need to be added to the cover for that window. However, it still may be needed to cover the edges in a later window. The pseudocode for the algorithm can be found in Algorithm 6. The monitoring table $\mathcal{C}$ holds all timesteps of a window and stores for all edges, whether they are active in it. To have minimal update costs in each iteration we keep a pointer for the first timestep in the window, which is the only one to be overwritten in every iteration. An additional vector $\mathcal{I}$ stores for each timestep in the current window, whether it is already included (2), available (1) or excluded (0) form the cover. The process is to iteratively go through all windows and check if any star center can be excluded from the cover. Therefore, we firstly update $\mathcal{C}$ and $\mathcal{I}$ based on the pointer to the first element and then iterate over the timesteps in the window. If a timestep is not already included in the cover and any of its edges can not be covered by other appearances, we add the star center of the timestep to the solution and mark it as included. Otherwise, we exclude it from the cover in that window. In that case we need to include all star center appearances in timestep $t + j$ to cover that step in the solution. Each j is chosen optimal (line 21) in the sense that we either choose the latest appearance to cover it or an earlier one, which is already included in the cover.

Theorem 2. *The advanced always star algorithm approximates Δ-TVC on always star temporal graphs with $T \geq \Delta$ and $\Delta \geq 2$ with ratio $\Delta - 1$ in $\mathcal{O}(Tm\Delta^2)$ time.*

Proof. The runtime of Algorithm 6 is in $\mathcal{O}(Tm\Delta^2)$, since the loops in lines 7 and 10 take time $\mathcal{O}(T\Delta)$. Checking if any edge is not covered by another star center (line 14) takes at most $\mathcal{O}(m\Delta)$, since we need to look at all edges in every timestep in the window. The loop in line 21 takes time at most $\mathcal{O}(m)$, since we store the cover candidates of every edge separately, to choose the optimal candidate. Hence, the overall runtime is in $\mathcal{O}(Tm\Delta^2)$.
The approximation ratio of the algorithm results from the fact that the algorithm excludes the first possible star center appearance which can be covered though others, even if several others could be excluded later if it was kept in the cover. Therefore, the worst case in terms of the approximation ratio arises on temporal topologies such as the one shown in Figure 4.8. In a general instance of the considered topology the lifetime T is a multiple of Δ. Let G_t denote the (static) subgraph of timestep t. Each subgraph repeats every Δ timesteps, i.e. $G_t = G_{t+\Delta}$. Moreover, the subgraphs $G_1, \ldots, G_{\Delta-1}$ are distinct in their edges and $E(G_0) = \bigcup_{i \in \Delta} E(G_i)$.
The green marked node appearances show the optimal solution, while the red marked ones are the solution computed by Algorithm 6. The optimal Δ-TVC contains the star center appearance of every G_i, where $i\%\Delta = 0$. Hence, its size is $\lceil \frac{T}{\Delta} \rceil$, since this is the amount of such G_i. The solution of Algorithm 6 on the other hand would exclude the star center appearances of these G_i if $\Delta \geq 2$, since they appear first and all their edges can be covered through the other star center appearances in each window, but would include the star center

Algorithm 6 More advanced always star algorithm

Input: A temporal graph (G, λ) with lifetime T, where $G = (V, E)$, and a natural $\Delta \leq T$
Output: A temporal vertex cover $\mathcal{X}$ of (G, λ)

$\mathcal{X} := \emptyset$
$\mathcal{C}[\Delta][m] := \langle\langle 0, \ldots, 0\rangle, \ldots, \langle 0, \ldots, 0\rangle\rangle$
$\mathcal{I}[\Delta] = \langle 1, \ldots, 1\rangle$
$first = 0$
Init $\mathcal{C}$ for timesteps $[0, \Delta - 2]$
$first = \Delta - 1$
foreach t *in* $[0, T - \Delta + 1]$ **do**
  Update $\mathcal{C}$ and $\mathcal{I}$ for timestep $t + \Delta - 1$ at $first$
  Update $first$
  foreach i *in* $[0, \Delta - 1]$ **do**
    $idx_i = (first + i)\%\Delta$
    if $\mathcal{I}[idx_i]$ *is already included in cover* **then**
      continue
    end
    if *Any edge* m *in* $\mathcal{C}[idx_i]$ *is not covered by another (not excluded) star center in* W_t **then**
      $ti = t + i$
      $\mathcal{X} = \mathcal{X} \cap \{(center_{ti}, ti)\}$
      $\mathcal{I}[idx_i] = 2$
    else
      $\mathcal{I}[idx_i] = 0$
      foreach *optimal* j *needed to cover an edge* m_i *in* i **do**
        $tj = t + j$
        $idx_j = (first + j)\%\Delta$
        $\mathcal{X} = \mathcal{X} \cap \{(center_{tj}, tj)\}$
        $\mathcal{I}[idx_j] = 2$
      end
    end
  end
end
return $\mathcal{X}$

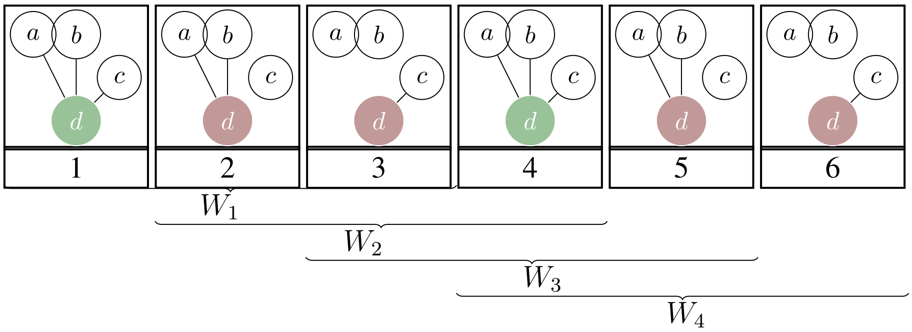


Figure 4.8: A worst case instance for the star-advance algorithm

of all other timesteps, since the subgraphs in the timesteps $\{j|i < j < i + \Delta \ \forall\, i\%\Delta = 0\}$ are distinct in every window. Hence, on instances with the considered topology our algorithm computes a solution of size $T - \lceil \frac{T}{\Delta} \rceil$ when $\Delta \geq 2$, what is stated in Theorem 2. By decomposing the lifetime over Δ, we get $T = a \cdot \Delta + b$, where $a, b \in \mathbb{N}_0^+$ and $b < \Delta$. To get to the ratio, we distinguish two cases, $b = 0$ and $b > 0$. In the first case, we consider $T = a \cdot \Delta$ ($b = 0$). The size of the optimal solution is

$$\left\lceil \frac{T}{\Delta} \right\rceil = \left\lceil \frac{a \cdot \Delta}{\Delta} \right\rceil = a$$

Therefore the approximation ratio is

$$\frac{T - \lceil \frac{T}{\Delta} \rceil}{\lceil \frac{T}{\Delta} \rceil} = \frac{a \cdot \Delta - a}{a} = \Delta - 1$$

For $b > 0$ we have $T = a \cdot \Delta + b$ and the size of the optimal solution is

$$\left\lceil \frac{T}{\Delta} \right\rceil = \left\lceil \frac{a \cdot \Delta + b}{\Delta} \right\rceil = a + 1$$

In this second case the ratio can be calculated as

$$\begin{aligned}
\frac{T - \lceil \frac{T}{\Delta} \rceil}{\lceil \frac{T}{\Delta} \rceil} &= \frac{a \cdot \Delta + 1 - (a+1)}{a+1} \\
&= \left(1 - \frac{1}{a+1}\right) \Delta + \frac{b}{a+1} - 1 \\
&= \Delta - 1 - \frac{\Delta - b}{a+1}
\end{aligned}$$

Since $b < \Delta$ the last subtrahend is always positive and the maximal ratio arise in the first case. Therefore, the approximation ratio of Algorithm 6 is $\Delta - 1$. $\square$

Similar to Algorithm 5 the more advanced Algorithm 6 computes the optimal solution for 1-TVC. The algorithm includes every appearing star center in that case, as every window has size 1 and no appearing edges can be covered by a star center from another timestep. This is also the optimal solution for 1-TVC. Further, the algorithm is also exact for 2-TVC, since Theorem 2 proves a ratio of 1 for $\Delta = 2$.

CHAPTER 5

Experimental Evaluation

This section provides the experimental results of the algorithms presented in Section 4. In particular, the d-approximation and $d-1$-approximation are verified regarding their stated runtime and approximation bounds. We then test their performance on real-life instances. Moreover, on restricted inputs of always star temporal graphs the two new approximation algorithms are tested against the current stateof the art being the approximations for always at most d approximation algorithms.

For the experiments we use slurm-jobs with 8 cores and 100GiB of RAM on an Ubuntu 20.04.5 LTS machine with linux kernel version 5.4.0-135, 112-core Intel(R) Xeon(R) Gold 6238R CPU running at 2.20GHz, and 512GiB main memory.

5.1 Runtime and Approximation Ratio Verification for d and $d-1$ Approximation Algorithms

The d-approximation algorithm states a runtime of $\mathcal{O}(Tm)$ and the $d-1$ approximation a runtime of $\mathcal{O}(T^2m^2)$. In this part we are running experiments increasing the number of edges and the lifetime of the input graphs to verify the stated bounds. Moreover, their solution size is then tested against the exact solver to show the approximation ratio. However, the approximation experiments are only run on small instances, since the exact algorithm runs in exponential time.

5.1.1 Runtime Experiments with Increasing Edge Number

The runtime of the stated approximations is dependent on the number of edges. In these first experiments this is to be checked. Therefore, we generate graphs with the arbitrary temporal graph generator with $n \in \{2\,048, 16\,384, 262\,144\}$, $T = 256$ and the number of edges is varied in $m \in$

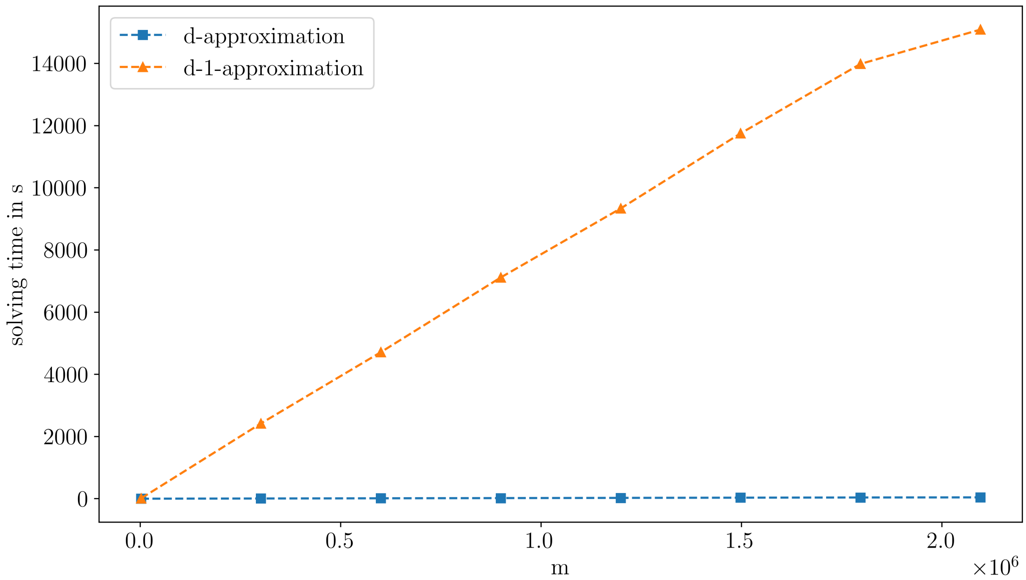


Figure 5.1: Runtime comparison in terms of m

$\{2\,048, 301\,202, 600\,356, 899\,510, 1\,198\,665, 1\,497\,819, 1\,796\,973, 2\,096\,128\}$. To reduce the effect of a generated temporal graph being randomly favored by one of the algorithms, we generate three instances of each configuration with different random seeds $s \in \{0, 3, 5\}$.

Figure 5.1 shows the experimental runtime of computing 16-TVC with the d and $d-1$ approximation algorithms averaged with the geometric mean over the instances. The d-approximation algorithm runs as claimed linear in terms of the edge number, while the $d-1$-approximation, which is stated to be quadratic to the edge number, appears to also be linear on the given instances.

The algorithm works over detecting and covering not yet covered two length paths, so called P_3s. The linear runtime can be explained through the way the detection of these P_3s is implemented, which happens over the center node, e.i. comparing uncovered edges of every node. Hence, only already connected edges are tested. When the graph has an arbitrary topology this results in the maximal underlying degree begin small compared to m.

More detailed insights into these results for both algorithms are shown in Figure 5.2 for the d-approximation and in Figure 5.3 for the $d-1$-approximation. Both clearly show the linear runtime increase in terms of m on the input graphs.

However, the worst case runtime of the $d-1$-approximation is reached on underlying star temporal graphs, since on these graphs every edge needs to be checked against every other, because they might form a P_3 connected in the single center node of a star temporal graph.

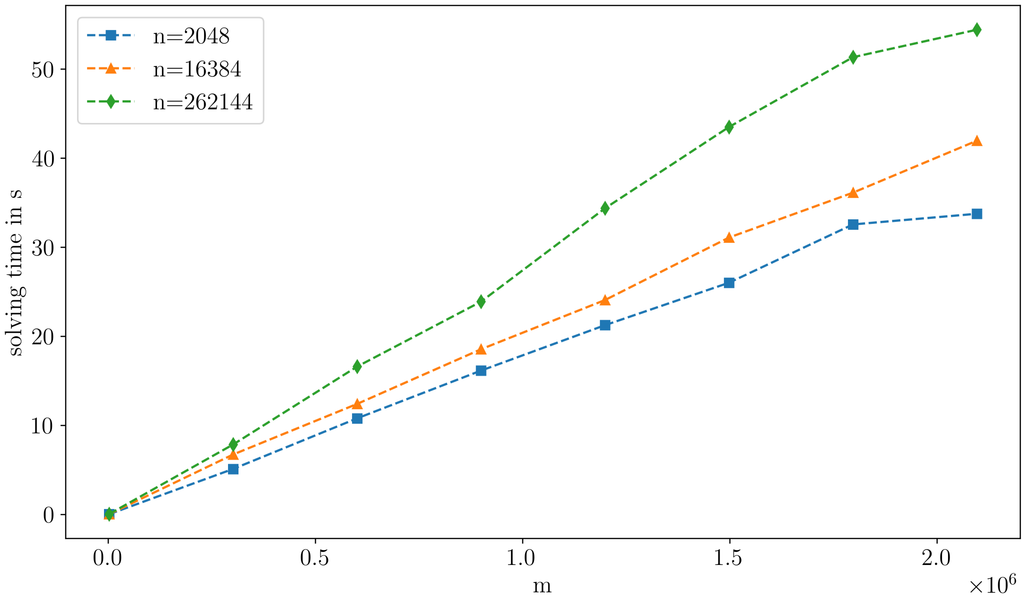


Figure 5.2: Runtime for the d-approximation in terms of m

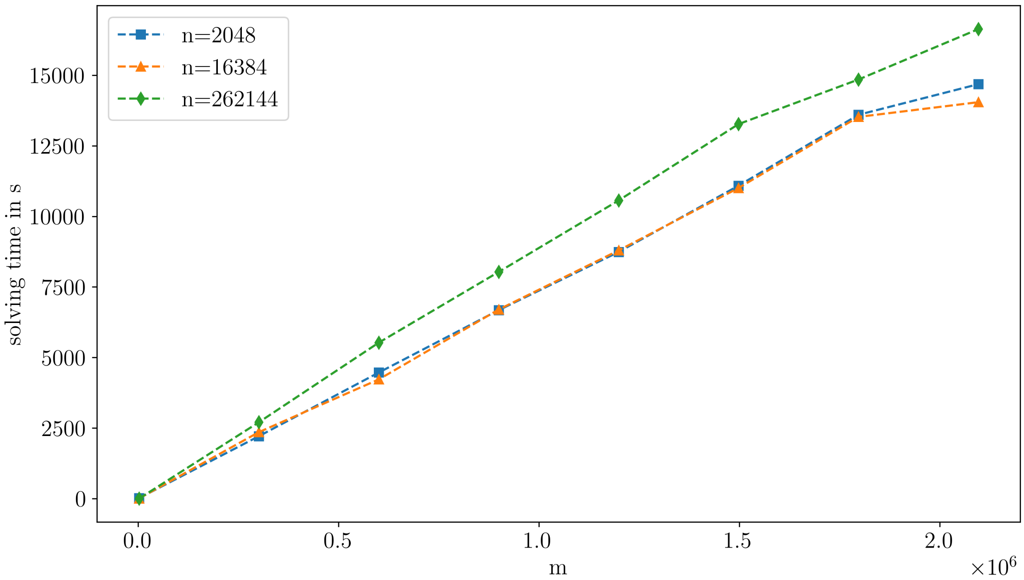


Figure 5.3: Runtime for the $d-1$-approximation in terms of m

To verify this assumption of the worst case runtime we generate underlying star temporal graphs with $n = 1048576$, $T = 128$ and the number of edges $m \in \{2\,048, 76\,653, 151\,259, 225\,865, 300\,470, 375\,076, 449\,682, 524\,288\}$, again with random seeds $s \in \{0, 3, 5\}$. The expected runtime is shown in Figure 5.4.

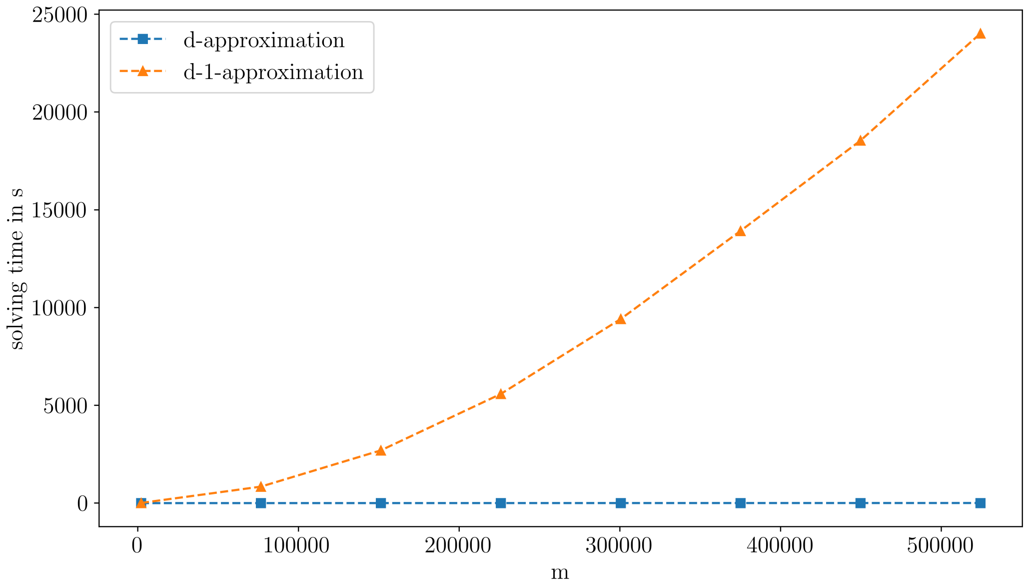


Figure 5.4: Runtime comparison in terms of m on underlying star graphs

5.1.2 Runtime Experiments with increasing Lifetime

Next to the number of edges the runtime of the stated approximations is also depended on the lifetime. To show this dependency, we generate different arbitrary temporal graphs with $n \in \{128, 2\,048, 16\,384\}$, $e \in \{1\,024, 8\,128\}$ and random seeds $s \in \{0, 3, 5\}$. Moreover, this time the runtime is varied for $T \in \{6, 2\,354, 4\,692, 7\,030, 9\,369, 11\,707, 14\,045\}$. The runtime results of computing 16-TVC with the d and $d-1$ approximation algorithms averaged with geometric mean are shown in Figure 5.5 and are as expected for the d-approximation linear to the runtime, while the $d-1$-approximation has quadratic dependency.

More detailed insights into the runtime performance of the algorithms are shown in Figure 5.6 for the d-approximation and in Figure 5.7 for the $d-1$-approximation. In these each data-point consists of three graphs with the same configuration except of the random seed $s \in \{0, 3, 5\}$. The behavior of both algorithms is as expected, showing the linear and quadratic dependency from the lifetime T.

Moreover, an interesting runtime result shows the variation of Δ, displayed in Figure 5.8. Therefore, we consider generated arbitrary temporal graph instances with $n = 2048$, $e \in \{1\,024, 8\,128\}$ and $T = 4692$. This time we vary the sliding window size $\Delta \in \{469, 938, 1\,407, 1\,876, 2\,346, 2\,815, 3\,284, 3\,753, 4\,222, 4\,692\}$.

The results state a dependence of Δ for both algorithms, showing its maximum around $\Delta = \frac{T}{2}$. This makes sense in for both algorithms, as in the d-approximation algorithm during the optimal solving a single edge, the number of windows to consider and the number of possible appearances in a window depend on Δ, and for the $d-1$-approximation algorithm

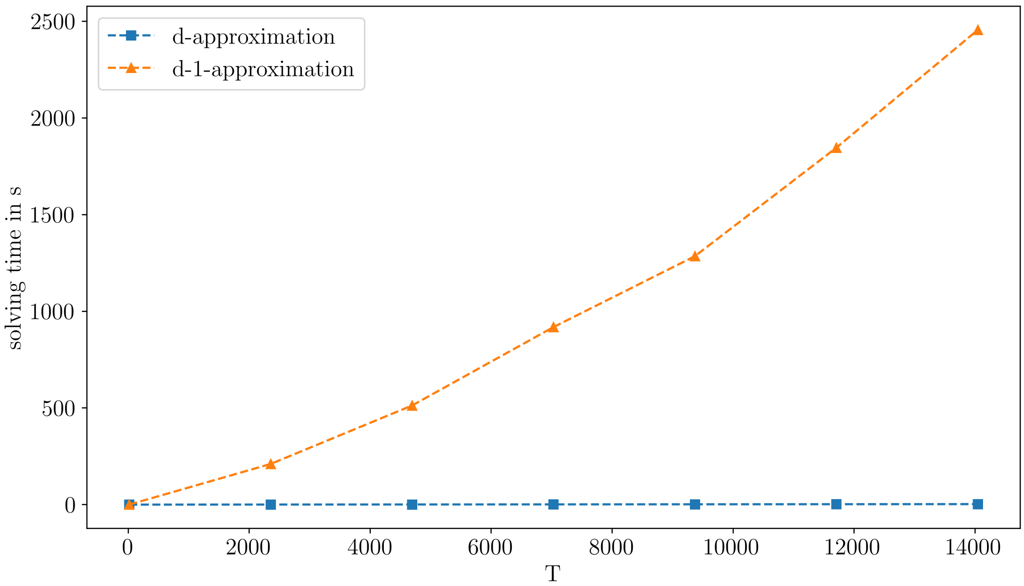


Figure 5.5: Runtime comparison in terms of T

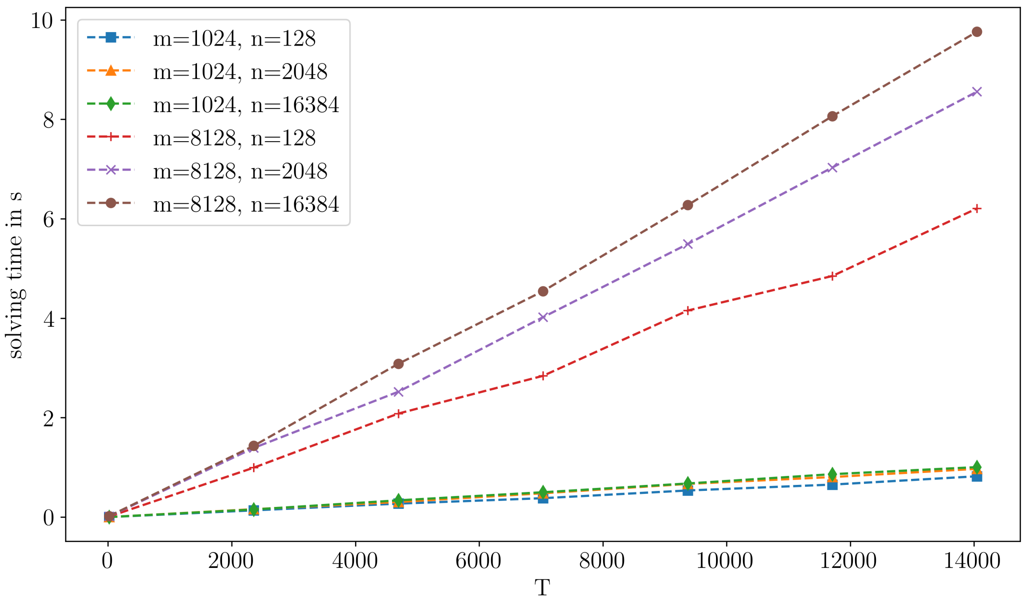


Figure 5.6: Runtime for the d-approximation in terms of T

the solving time of a sub-instance as well as the number of sub-instances depend on Δ. This is particularly interesting, since this shows an additional dependence of the parameter Δ not covered in the theoretical analysis.

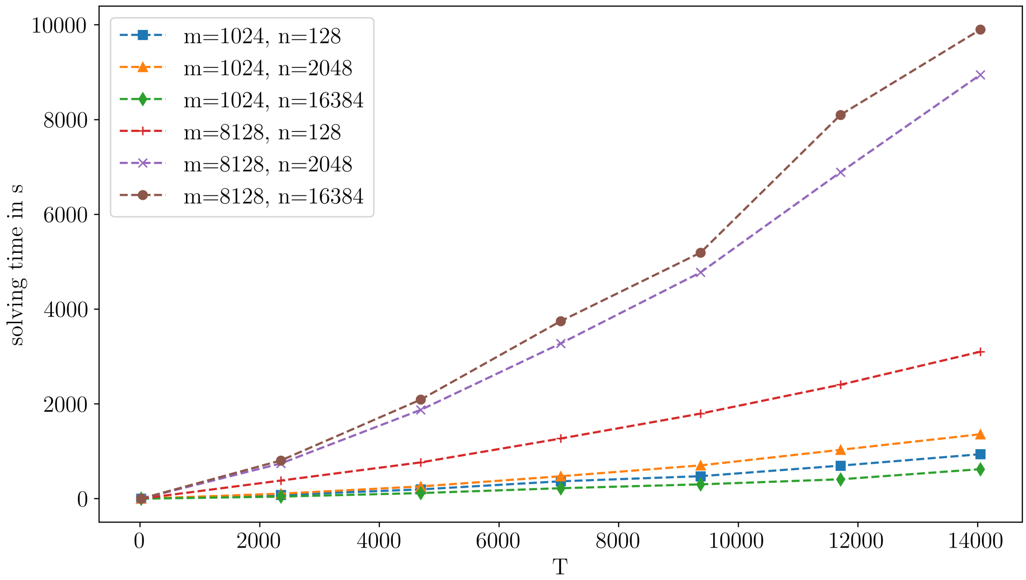


Figure 5.7: Runtime for the $d-1$-approximation in terms of T

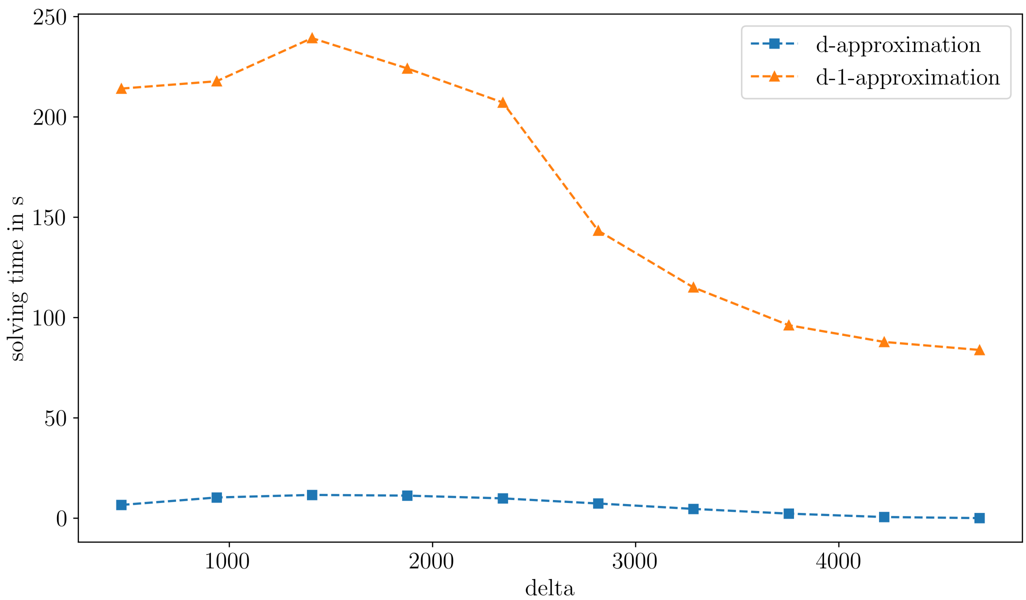


Figure 5.8: Runtime comparison in terms of Δ ($T = 4\,096$)

5.1.3 Approximation Ratio Experiments

Next to the runtime bounds, an aim of the thesis is also to verify the stated approximation ratios. Therefore, instances of always degree at most d temporal graphs are generated and solved by the approximation algorithms as well as the presented exact algorithm. By

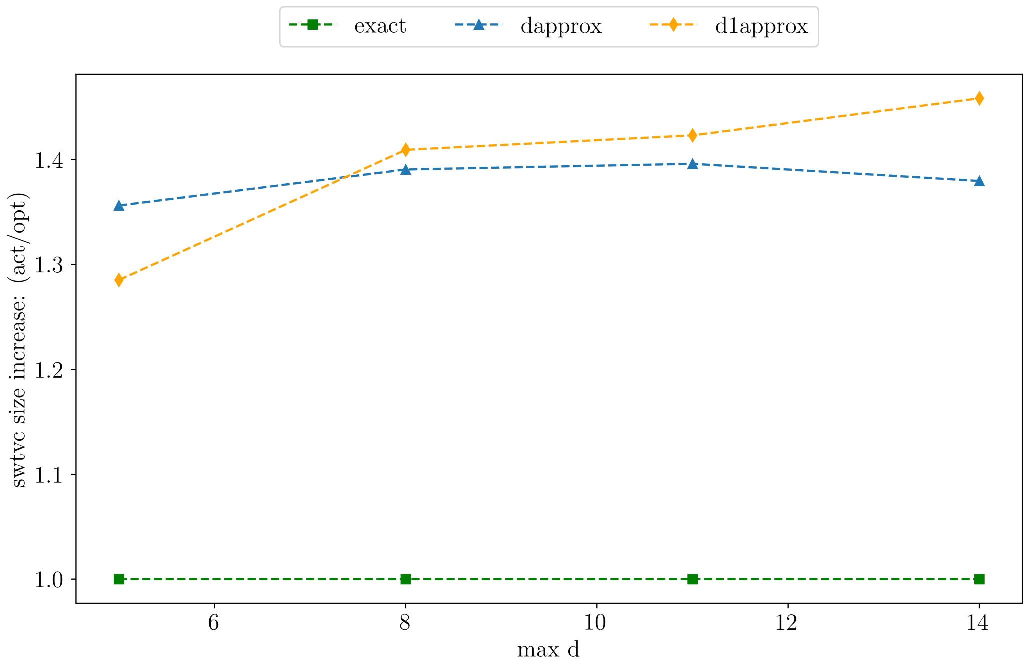


Figure 5.9: Approximation ratio comparison

dividing the computed solution by the optimal one, we receive the approximation ratio. This is tested on small instances, since the optimal solution is only computable in exponential runtime ($\mathcal{O}(T\Delta^{\mathcal{O}(m)})$). As input graphs, we generate always degree at most d graphs with $n = 16$, $T = 16$, and vary the maximal degree $d \in \{5, 8, 11, 14\}$.
The approximation ratios for 2-TVC are shown in Figure 5.9. To calculate these we divide the computed solutions by the optimal solution, generated by the exact algorithms. As clearly seen, both algorithms are far within the stated ratios. Surprisingly, the $d-1$ approximation performed worse than the d-approximation on these instances. This can be explained, through the functionality of the $d-1$ approximation, since it calculates an optimal solution for the P_3 in the area affected by all the P_3s ($[\min S_i - \Delta + 1, \max S_i + \Delta - 1]$), but the solution vertex appearances are only computed in the occurrence area ($[\min S_i, \max S_i]$). On larger graphs this effect is not as substantial. This can be seen in Figure 5.10, which are the same graphs used in the lifetime increasing experiments.

5.1.4 Experiments on Real-Life Data

As both most d approximation algorithms can solve arbitrary temporal graphs, we test their performance on reallife graphs. We use graphs from the SNAP library [27], which provides several social networks based on email communication [36], social media platforms [35], exchange web sites [36] [26] or hyperlink networks in form of connections between subreddits [24]. The details of the dataset are provided in Table 5.1.

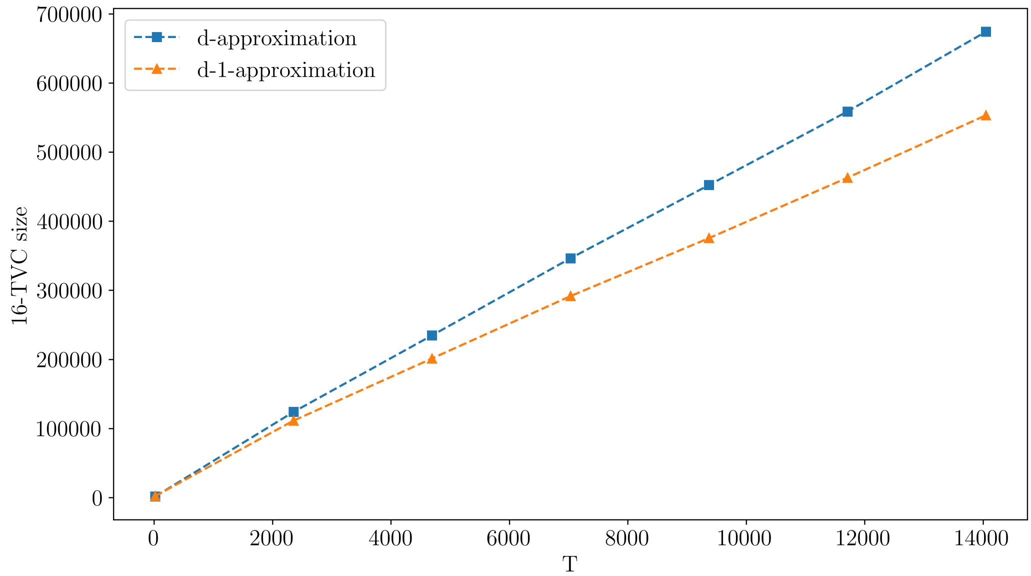


Figure 5.10: SW-TVC size comparison

As these come in different formats and use timestamps in date format or unix timestamp (seconds since the epoch), first a preprocessing is required to fit them in the uniform format used as input for the TVC-solver. In terms of ease, we consider hourly contacts. We remove the direction of the edges, self-loops and any possible other provided information. The edges represent then connection point in form of comments, links, etc. between the nodes.

Table 5.1: Reallife temporal graph dataset from the SNAP library [27]

Graph	T	$\|V\|$	$\|E\|$	Description
email-Eu-core-temporal	19 295	1 005	16 064	E-mails between users at a research institution
sx-askubuntu	62 732	515 280	455 691	Comments, questions, and answers on Ask Ubuntu
sx-mathoverflow	56 408	88 580	187 986	Comments, questions, and answers on Math Overflow
sx-superuser	66 560	567 315	714 570	Comments, questions, and answers on Super User
wiki-talk-temporal	55 690	1 140 149	2 787 967	Users editing talk pages on Wikipedia
CollegeMsg	4 649	1 899	13 838	Messages on a Facebook-like platform at UC-Irvine
soc-redditHyperlinks-body	29 184	27 862	137 808	Hyperlinks between subreddits on Reddit
soc-redditHyperlinks-title	29 184	43 694	234 777	Hyperlinks between subreddits on Reddit

Figure 5.11 and 5.12 show the results of a 64-TVC computation normalized by the d-approximation algorithm. We repeated the experiments three times and built the geometric mean. The detailed results can be found in Table A.1. As expected the $d-1$-approximation algorithm provides better solutions in all cases. Using improvement calculated as $\left(\frac{\sigma_B}{\sigma_A} - 1\right) * 100\%$ [13], where Algorithm A is compared with Algorithm B and σ_S is some objective, the $d-1$-approximation algorithm achieved an improvement of $11{,}58\%$ in solution size. Rather surprising, is that the $d-1$-approximation algorithm is also faster

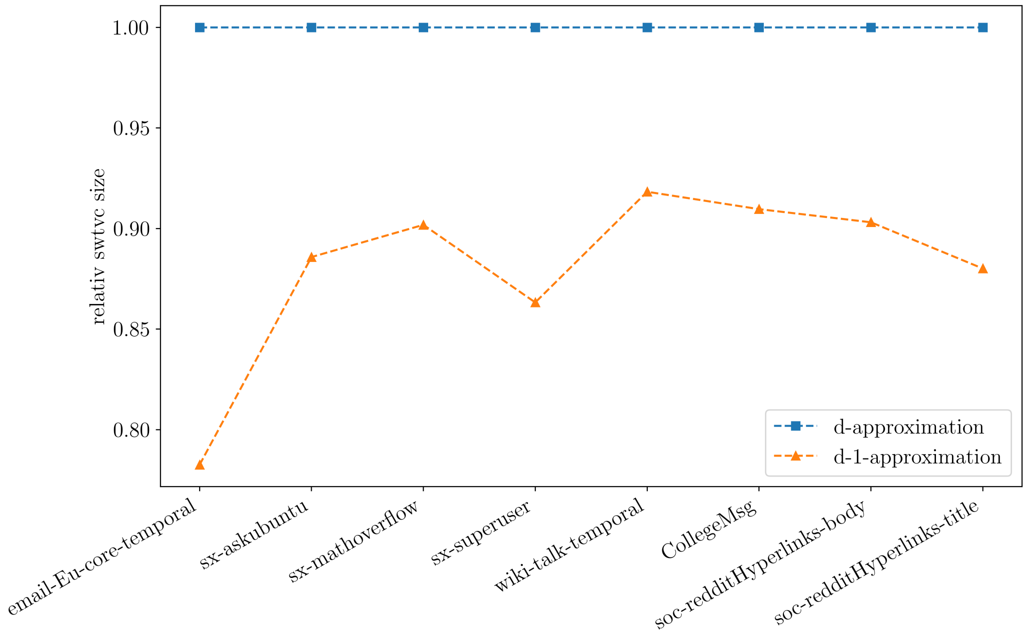


Figure 5.11: 64-TVC size comparison on real-life instances

than the d-approximation algorithm is most case, achieving a spectacular time improvement of 1 031,02%. This can be explained as all these graphs are rather sparse in terms of edge appearances. While the implementation of the $d-1$-approximation algorithm works on uncovered labels, the d-approximation algorithms iterates for every edge through the lifetime.

5.2 Experimental Evaluation of new Always Star Approximation Algorithms

The current state of the art provides the always degree at most d approximation algorithms to solve always star temporal graphs. In Section 4.4 two new algorithms for this restricted case are presented. This section provides an experimental evaluation of them against the current state.

5.2.1 Experiments under the Condition $\Delta < d$

Therefore, we generate always star temporal graphs with $n = 128$, $T = 64$ and the random seed $s \in \{0, 3\}$. To provide a comparison with the d and $d-1$-algorithms, we vary the maximum degree of the graphs $d \in \{10, 15, 20, 25, 30\}$. Moreover, to get better insight into the approximation ratios we also use (underlying) star temporal graphs, a subclass

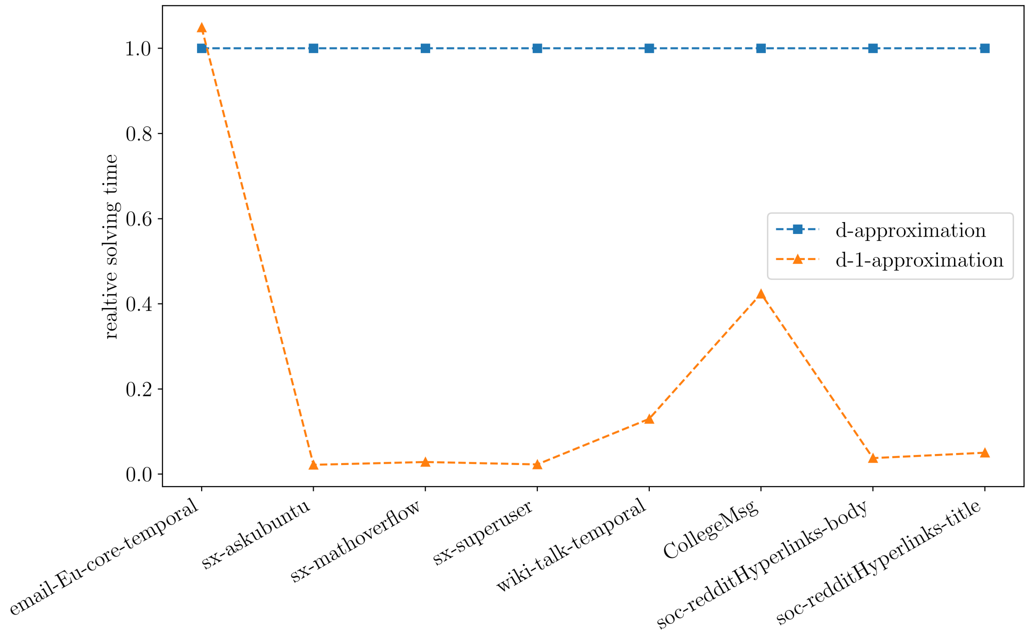


Figure 5.12: 64-TVC runtime comparison on real-life instances

of the always star temporal graphs. By doing so, we increase the inputs, where the new always star algorithms can not compute the optimal solution, since the worst cases of both algorithms also originate from this class. For the generation we use the same configuration as for the always star ones. In total, the graph dataset used consists of one half always star and one half star temporal graphs. The instances with their class, maximal degree, lifetime and number of nodes and edges are displayed in Table 5.2.

The results of the 3- and 4-TVC are displayed in Figure 5.13, where the solution size is normalized by the exact solution. This experiment shows that the star algorithms provide far better, even close to optimal solutions than the d-approximation algorithm in this scenario. As expected the $d-1$-approximation performances much better than the d approximation, as the $d-1$-approximation searches for uncovered triangles, leading to the detection of the star center.

To get insight into the overall performance, we also compare the running time to compute the solutions and normalized the results by the fastest algorithm. Figure 5.14 shows that the runtime of the $d-1$-approximation algorithm, is by far the largest ($\approx$ 29,977 ms per instances), while the other algorithms are much faster. The per instances runtimes are $\approx$ 0,056 ms for the star-trivial, $\approx$ 0,917 ms for the d-approximation and $\approx$ 1,457 for the star-advance approximation, which is completely within the expectations since the number of edges in the graphs increases with the increase of d, as shown in Table 5.2.

The details of the experiments can be found in Table A.2 and Table A.3. Overall, the both the star-trivial and the star-advance approximation provide better solutions in shorter

Table 5.2: Graph dataset 1 of always star temporal graphs

Class	Maximal degree d	Lifetime T	Number of Nodes $\|V\|$	Number of Edges $\|E\|$
star	10	64	128	564
star	10	64	128	548
star	10	64	128	569
star	15	64	128	796
star	15	64	128	784
star	15	64	128	808
star	20	64	128	1 024
star	20	64	128	981
star	20	64	128	1 024
star	25	64	128	1 237
star	25	64	128	1 173
star	25	64	128	1 222
star	30	64	128	1 443
star	30	64	128	1 350
star	30	64	128	1 423
ustar	10	64	128	10
ustar	10	64	128	10
ustar	10	64	128	10
ustar	15	64	128	15
ustar	15	64	128	15
ustar	15	64	128	15
ustar	20	64	128	20
ustar	20	64	128	20
ustar	20	64	128	20
ustar	25	64	128	25
ustar	25	64	128	25
ustar	25	64	128	25
ustar	30	64	128	30
ustar	30	64	128	30
ustar	30	64	128	30

runtime than the $d-1$-approximation. While the d-approximation is faster than the star-advance approximation, its computed solutions are not competetive.

5.2.2 Experiments under the Condition $\Delta > d$

In terms of analysis the d and $d-1$-approximations provide better worst case ratios as than star-trivial and the star-advance approximation, being $2\Delta - 1$ and $\Delta - 1$, when $\Delta > d$. But the worst case scenario especially for the star-advance approximation is very specific. We would argue that in most cases the star-advance algorithm still outperforms the most d approximations, as it is specifically designed for always star temporal graphs and includes at most one vertex in any timestep. Therefore, the following experiments test the algorithms in the case where $\Delta > d$.
The input graph dataset consists again of always star and star temporal graphs generated with $n = 128, T = 64$ and the random seed $s \in \{0, 3, 5\}$. The maximal degree is small $d \in \{3, 4, 5, 6, 7, 8\}$. Table 5.3 shows the generated temporal graph dataset with class, maximal degree, lifetime and number of nodes and edges.

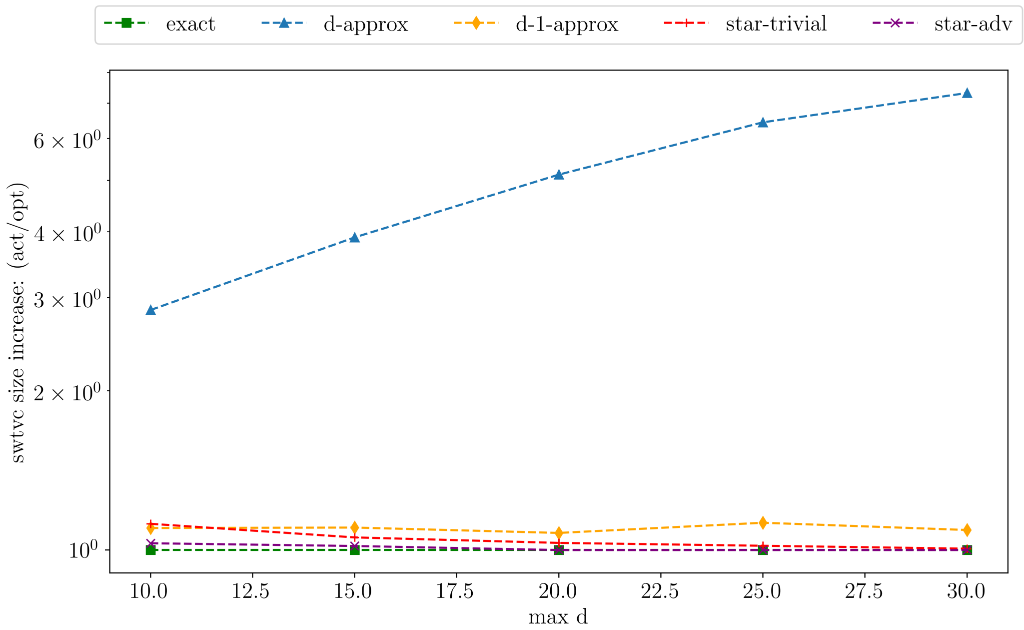


Figure 5.13: Approximation ratios on always star temporal graphs

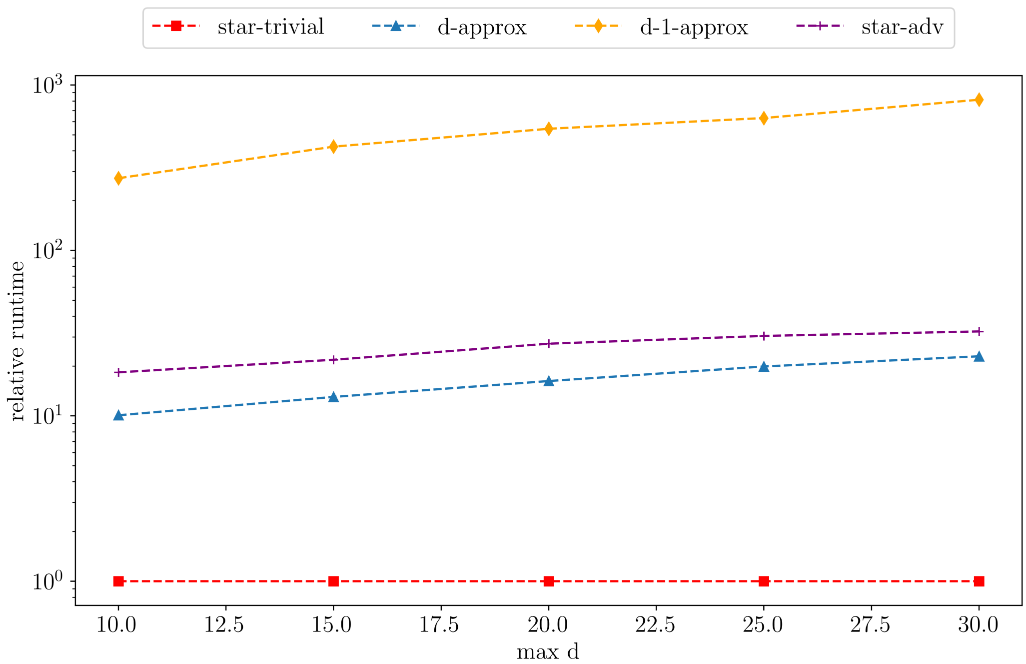


Figure 5.14: Runtime comparison on always star temporal graphs

Table 5.3: Graph dataset 2 of always star temporal graphs

Class	Maximal degree d	Lifetime T	Number of Nodes $\|V\|$	Number of Edges $\|E\|$
star	2	64	128	125
star	2	64	128	124
star	2	64	128	127
star	4	64	128	243
star	4	64	128	236
star	4	64	128	245
star	5	64	128	299
star	5	64	128	291
star	5	64	128	296
star	6	64	128	353
star	6	64	128	345
star	6	64	128	350
star	7	64	128	411
star	7	64	128	396
star	7	64	128	406
star	8	64	128	463
star	8	64	128	446
star	8	64	128	460
ustar	2	64	128	2
ustar	2	64	128	2
ustar	2	64	128	2
ustar	4	64	128	4
ustar	4	64	128	4
ustar	4	64	128	4
ustar	5	64	128	5
ustar	5	64	128	5
ustar	5	64	128	5
ustar	6	64	128	6
ustar	6	64	128	6
ustar	6	64	128	6
ustar	7	64	128	7
ustar	7	64	128	7
ustar	7	64	128	7
ustar	8	64	128	8
ustar	8	64	128	8
ustar	8	64	128	8

To ensure the discussed condition of $\Delta + 1 > d$ in any case, we compute 20-TVC. Figure 5.15 displays the normalized results, showing that our expectation that the star-advance algorithm computes the best results is true. The star-trivial approximation becomes better with increase of the maximal degree. This can be explained through the fact that in these cases especially for underlying stars the size of the optimal solution increases, while the computed size stays the same. On the one hand the increase of the degree of the underlying star, leads to a larger optimal solution, because there are more possible combinations of edges in a window. The solution of the star-trivial approximation on the other hand consists still of every star center and its size stays the same.

To complete these experimental results, Figure 5.16 shows the runtime for computing 20-TVC on these inputs normalized by the fastest algorithm. Similar to before, within the runtime expectations, the $d-1$-approximation algorithm is the slowest with $\approx 6{,}854$ ms

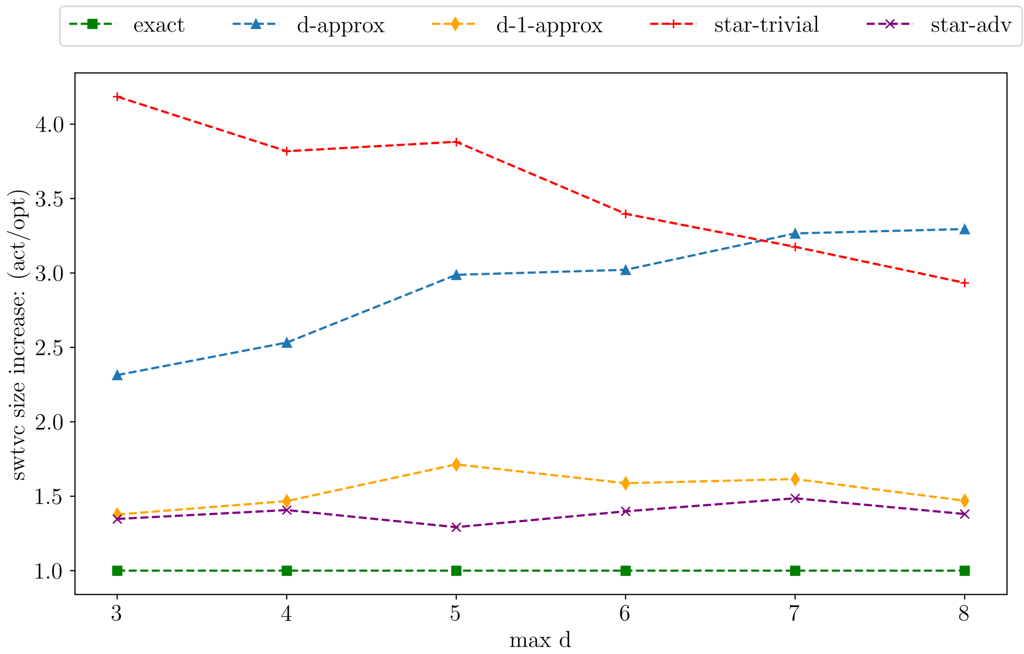


Figure 5.15: Approximation ratios on small degree always star temporal graphs

per instance. The star-advance approximation algorithm follows, with $\approx$ 2,278 ms, followed then by the d-approximation algorithm with $\approx$ 0,259 ms and then the star-trivial algorithm with $\approx$ 0,054 ms.
The detailed results are shown in Table A.4. It is clearly shown that the star star-advance approximation algorithm outperforms the $d-1$-approximation in almost all instances in shorter runtime.

5.2.3 Experiments on large Instances

The experiments in the previous subsection only work with small instances of always star temporal graphs as they provide the exact solution as reference, which is not computable in a reasonable time for larger ones. Therefore, we compare in the next experiment only the size of the approximations for SW-TVC.
We use always star and star temporal graphs as inputs, with $n \in \{1\,024, 4\,096\}$, $d \in \{5, 50, 100\}$ and the random seed $s \in \{0, 3, 5\}$. We increase our input over the lifetime $T \in \{2\,354, 4\,692, 7\,030, 9\,369, 11\,707, 14\,045\}$. The generated temporal graphs with class, maximal degree, lifetime and number of nodes and edges are shown in Table 5.4.

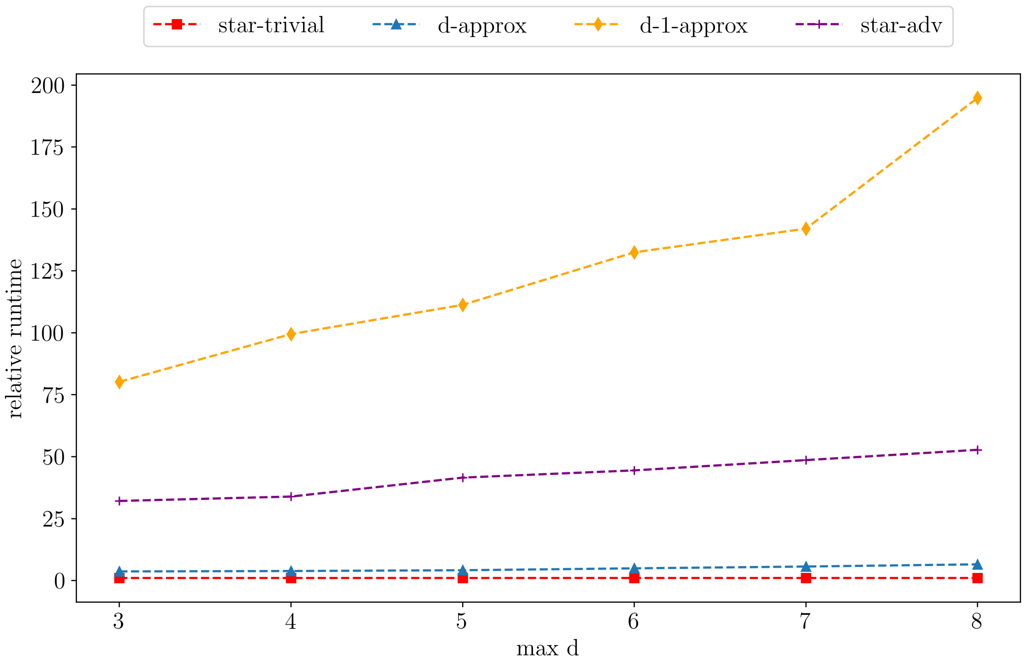


Figure 5.16: Runtime comparison on always star small degree temporal graphs

Table 5.4: Graph dataset 3 of always star temporal graphs

Class	Maximal degree d	T	$\|V\|$	$\|E\|$	Class	Maximal degree d	T	$\|V\|$	$\|E\|$
star	100	11 707	1 024	22 126	ustar	100	11 707	1 024	100
star	100	11 707	1 024	21 449	ustar	100	11 707	1 024	100
star	100	11 707	1 024	21 770	ustar	100	11 707	1 024	100
star	50	11 707	1 024	13 930	ustar	50	11 707	1 024	50
star	50	11 707	1 024	13 471	ustar	50	11 707	1 024	50
star	50	11 707	1 024	13 765	ustar	50	11 707	1 024	50
star	5	11 707	1 024	6 340	ustar	5	11 707	1 024	5
star	5	11 707	1 024	6 225	ustar	5	11 707	1 024	5
star	5	11 707	1 024	6 289	ustar	5	11 707	1 024	5
star	100	11 707	4 096	21 954	ustar	100	11 707	4 096	100
star	100	11 707	4 096	22 187	ustar	100	11 707	4 096	100
star	100	11 707	4 096	22 007	ustar	100	11 707	4 096	100
star	50	11 707	4 096	13 768	ustar	50	11 707	4 096	50
star	50	11 707	4 096	13 967	ustar	50	11 707	4 096	50
star	50	11 707	4 096	13 909	ustar	50	11 707	4 096	50
star	5	11 707	4 096	6 293	ustar	5	11 707	4 096	5
star	5	11 707	4 096	6 358	ustar	5	11 707	4 096	5
star	5	11 707	4 096	6 401	ustar	5	11 707	4 096	5
star	100	14 045	1 024	22 784	ustar	100	14 045	1 024	100
star	100	14 045	1 024	22 217	ustar	100	14 045	1 024	100
star	100	14 045	1 024	22 335	ustar	100	14 045	1 024	100
star	50	14 045	1 024	14 448	ustar	50	14 045	1 024	50
star	50	14 045	1 024	14 033	ustar	50	14 045	1 024	50
star	50	14 045	1 024	14 200	ustar	50	14 045	1 024	50
star	5	14 045	1 024	6 663	ustar	5	14 045	1 024	5
star	5	14 045	1 024	6 588	ustar	5	14 045	1 024	5
star	5	14 045	1 024	6 621	ustar	5	14 045	1 024	5
star	100	14 045	4 096	22 705	ustar	100	14 045	4 096	100
star	100	14 045	4 096	22 734	ustar	100	14 045	4 096	100
star	100	14 045	4 096	22 759	ustar	100	14 045	4 096	100
star	50	14 045	4 096	14 325	ustar	50	14 045	4 096	50
star	50	14 045	4 096	14 420	ustar	50	14 045	4 096	50
star	50	14 045	4 096	14 447	ustar	50	14 045	4 096	50
star	5	14 045	4 096	6 678	ustar	5	14 045	4 096	5
star	5	14 045	4 096	6 686	ustar	5	14 045	4 096	5

star	5	14 045	4 096	6 722	ustar	5	14 045	4 096	5
star	100	2 354	1 024	16 456	ustar	100	2 354	1 024	100
star	100	2 354	1 024	15 571	ustar	100	2 354	1 024	100
star	100	2 354	1 024	15 950	ustar	100	2 354	1 024	100
star	50	2 354	1 024	9 869	ustar	50	2 354	1 024	50
star	50	2 354	1 024	9 483	ustar	50	2 354	1 024	50
star	50	2 354	1 024	9 582	ustar	50	2 354	1 024	50
star	5	2 354	1 024	3 673	ustar	5	2 354	1 024	5
star	5	2 354	1 024	3 684	ustar	5	2 354	1 024	5
star	5	2 354	1 024	3 581	ustar	5	2 354	1 024	5
star	100	2 354	4 096	15 722	ustar	100	2 354	4 096	100
star	100	2 354	4 096	16 554	ustar	100	2 354	4 096	100
star	100	2 354	4 096	16 579	ustar	100	2 354	4 096	100
star	50	2 354	4 096	9 502	ustar	50	2 354	4 096	50
star	50	2 354	4 096	9 965	ustar	50	2 354	4 096	50
star	50	2 354	4 096	9 990	ustar	50	2 354	4 096	50
star	5	2 354	4 096	3 627	ustar	5	2 354	4 096	5
star	5	2 354	4 096	3 672	ustar	5	2 354	4 096	5
star	5	2 354	4 096	3 734	ustar	5	2 354	4 096	5
star	100	4 692	1 024	19 147	ustar	100	4 692	1 024	100
star	100	4 692	1 024	18 043	ustar	100	4 692	1 024	100
star	100	4 692	1 024	18 667	ustar	100	4 692	1 024	100
star	50	4 692	1 024	11 728	ustar	50	4 692	1 024	50
star	50	4 692	1 024	11 201	ustar	50	4 692	1 024	50
star	50	4 692	1 024	11 452	ustar	50	4 692	1 024	50
star	5	4 692	1 024	4 764	ustar	5	4 692	1 024	5
star	5	4 692	1 024	4 734	ustar	5	4 692	1 024	5
star	5	4 692	1 024	4 712	ustar	5	4 692	1 024	5
star	100	4 692	4 096	18 077	ustar	100	4 692	4 096	100
star	100	4 692	4 096	19 116	ustar	100	4 692	4 096	100
star	100	4 692	4 096	18 733	ustar	100	4 692	4 096	100
star	50	4 692	4 096	11 132	ustar	50	4 692	4 096	50
star	50	4 692	4 096	11 717	ustar	50	4 692	4 096	50
star	50	4 692	4 096	11 552	ustar	50	4 692	4 096	50
star	5	4 692	4 096	4 738	ustar	5	4 692	4 096	5
star	5	4 692	4 096	4 798	ustar	5	4 692	4 096	5
star	5	4 692	4 096	4 797	ustar	5	4 692	4 096	5
star	100	7 030	1 024	20 426	ustar	100	7 030	1 024	100
star	100	7 030	1 024	19 517	ustar	100	7 030	1 024	100
star	100	7 030	1 024	19 848	ustar	100	7 030	1 024	100
star	50	7 030	1 024	12 676	ustar	50	7 030	1 024	50
star	50	7 030	1 024	12 143	ustar	50	7 030	1 024	50
star	50	7 030	1 024	12 370	ustar	50	7 030	1 024	50
star	5	7 030	1 024	5 445	ustar	5	7 030	1 024	5
star	5	7 030	1 024	5 377	ustar	5	7 030	1 024	5
star	5	7 030	1 024	5 411	ustar	5	7 030	1 024	5
star	100	7 030	4 096	20 021	ustar	100	7 030	4 096	100
star	100	7 030	4 096	20 303	ustar	100	7 030	4 096	100
star	100	7 030	4 096	20 142	ustar	100	7 030	4 096	100
star	50	7 030	4 096	12 352	ustar	50	7 030	4 096	50
star	50	7 030	4 096	12 583	ustar	50	7 030	4 096	50
star	50	7 030	4 096	12 560	ustar	50	7 030	4 096	50
star	5	7 030	4 096	5 373	ustar	5	7 030	4 096	5
star	5	7 030	4 096	5 468	ustar	5	7 030	4 096	5
star	5	7 030	4 096	5 492	ustar	5	7 030	4 096	5
star	100	9 369	1 024	21 465	ustar	100	9 369	1 024	100
star	100	9 369	1 024	20 692	ustar	100	9 369	1 024	100
star	100	9 369	1 024	20 948	ustar	100	9 369	1 024	100
star	50	9 369	1 024	13 425	ustar	50	9 369	1 024	50
star	50	9 369	1 024	12 921	ustar	50	9 369	1 024	50
star	50	9 369	1 024	13 164	ustar	50	9 369	1 024	50
star	5	9 369	1 024	5 954	ustar	5	9 369	1 024	5
star	5	9 369	1 024	5 848	ustar	5	9 369	1 024	5
star	5	9 369	1 024	5 908	ustar	5	9 369	1 024	5
star	100	9 369	4 096	21 185	ustar	100	9 369	4 096	100
star	100	9 369	4 096	21 440	ustar	100	9 369	4 096	100
star	100	9 369	4 096	21 303	ustar	100	9 369	4 096	100
star	50	9 369	4 096	13 173	ustar	50	9 369	4 096	50
star	50	9 369	4 096	13 375	ustar	50	9 369	4 096	50
star	50	9 369	4 096	13 394	ustar	50	9 369	4 096	50
star	5	9 369	4 096	5 894	ustar	5	9 369	4 096	5
star	5	9 369	4 096	5 961	ustar	5	9 369	4 096	5
star	5	9 369	4 096	5 990	ustar	5	9 369	4 096	5

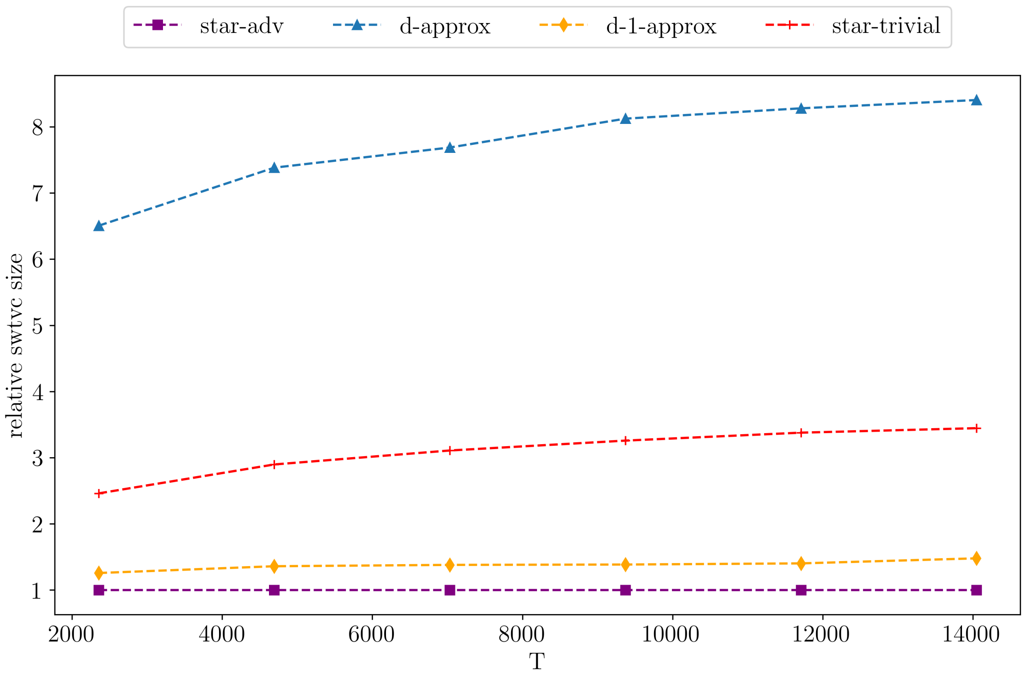


Figure 5.17: 16-TVC size comparison on large always star temporal graphs

On these inputs we compute the 16-TVC. Table A.5 shows the detailed results. The normalized sizes of the computed solutions are shown in Figure 5.17. They show the expected results of the d-approximation algorithm performing worst with an average 16-TVC size of 19 582, then the star-trivial approximation algorithm with 7 908. The $d-1$ approximation algorithm reaches an average size of 3 441, which is again surpassed by the star-advance approximation algorithm with 2 450. This is an improvement of 40.46% of the star-advance approximation compared to the $d-1$ approximation.
The runtimes normalized by the fastest algorithm of this experiment is shown in Figure 5.18. The star-trivial approximation algorithm runs on average 12,13ms per instances, followed by the d-approximation algorithm with 256,16ms and the star-advance approximation algorithm 3 199,48ms. The longest runtime per instance is needed by the $d-1$ approximation algorithm with on average 6 996,85ms per instances. Taking the improvement formula from above this leads to a time improvement of the star-advance approximation algorithm of 218.69%.

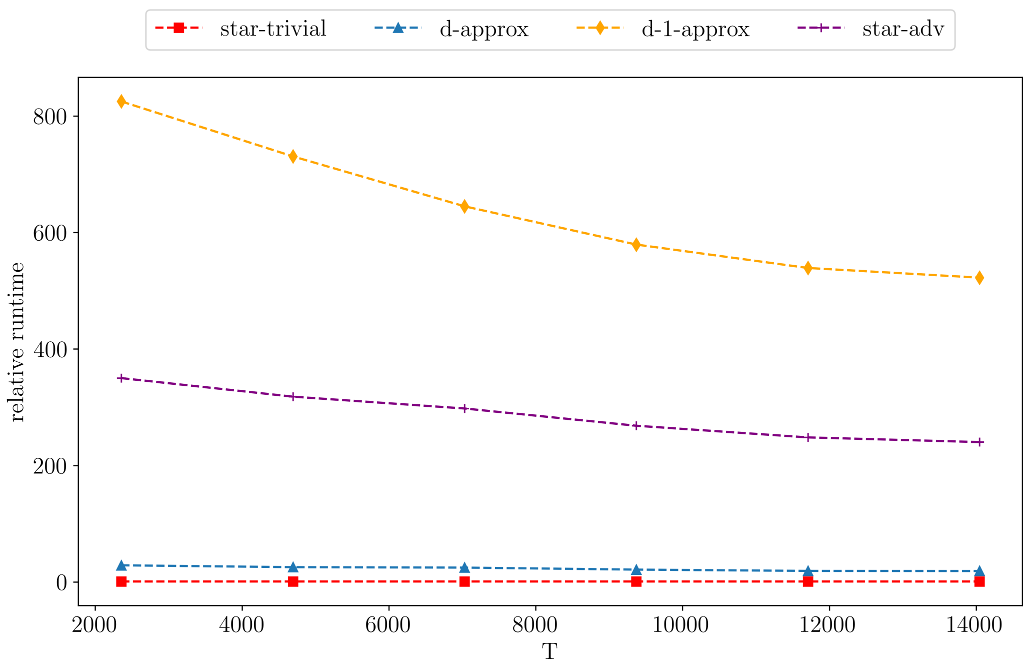


Figure 5.18: Runtime comparison on large always star temporal graphs

CHAPTER 6

Discussion

6.1 Evaluation

This section assesses the research questions formulated in Section 1 and how they are answered, and identifies the limitations which the thesis is subject to.

6.1.1 Improvement Through the new Always Star Approximation Algorithms

The experiments in the previous section have shown that the new star approximation algorithms perform better on always star temporal graphs because they use the known topology of this class of graphs. In particular, both star approximation algorithms outperform the current best known approximations, even in the case $\Delta + 1 > d$ the star-advance approximation algorithms outperforms the approximation algorithms with at most degree d for most temporal graphs.
The experiments where $\Delta < d$ (see Figure 5.13) clearly show that the provided star approximation algorithms always provide the best, even near-optimal solution. They also verify the expected behavior of the d-approximation computing worse in terms of solution size than the $d - 1$-approximation for these inputs.
In the experiment on small maximal degree temporal graphs with a large window size Δ (see Figure 5.15) the condition $\Delta + 1 > d$ holds, leading to the analytical fact, that $\Delta - 1$-approximation provided by the advance star algorithm and $2\Delta - 1$-approximation provided by the star-trivial algorithm might compute worse results than the d- or $d-1$-approximation. However, the experiments prove that assumption to be wrong for the star-advance approximation algorithm on most inputs, because its worse case is very specific.
Comparing the computation of 16-TVC on larger instances (see Figure 5.17) again clearly show the superiority of the star-advance algorithm with an 40.46% solution size improvement and 218.69% runtime improvement compared to the $d - 1$-approximation.

Overall, the experiments reveal significant improvement of the solution size of the (Δ)-TVC on always star temporal graphs achieved by the star-advance approximation algorithms compared to the know state of research.

6.1.2 Research Questions

The two questions answered in this thesis were how (Δ-)TVC can be approximated efficiently and how to achieve a better approximation of (Δ-)TVC on the restricted input of always star temporal graphs.
The overview of the state of research showed that (Δ-)TVC is NP-hard on arbitrary inputs. Moreover, several exact and approximation algorithms have been provided in the literature for arbitrary and specific temporal graph classes. To experimentally demonstrate the performance of these approximations, we choose one exact algorithm and two approximation algorithms, with d and $d-1$ approximation ratios for always degree at most d temporal graphs, for implementation in the TVC-solver framework. The advantage of these algorithms is, that they can run on every input graph, while the solution quality is bounded by the maximum degree d. Therefore, we were able to run them on real-life instances not categorized in any temporal graph class. Using these as inputs, which are sparse in their edge appearances, the $d-1$ approximation algorithm outperformed the d approximation algorithm with an 11,58$\%$ improvement in solution size and 1 031,02$\%$ in runtime. Overall, the experiments on these approximations gave a good overview on how (Δ-)TVC can be approximated efficiently. Especially when the inputs are sparse in their edge appearances, the $d-1$ approximation algorithm is a good solution. Since the focus in this thesis was on providing a first implementation and verify the stated complexity, we believe some additional engineering in the implementations is possible to make the computations faster in their actual runtime.
For the restricted input of always star temporal graphs we provide two algorithms using the known topology of the input graph class to derive better approximation ratios. They take advantage of the fact, that at most one vertex appearance, the star center, in any timestep should be included into the solution. By doing so the star-trivial approximation algorithm we derive achieves a $2\Delta-1$ ratio in $\mathcal{O}(T)$ and the star-advance approximation algorithm achieves a $\Delta-1$ ratio in $\mathcal{O}(Tm\Delta^2)$. They all known approximations in terms of solution quality, even if they are analytically not directly comparable, as they use different parameters in their worst case bounds. This was still the case on graphs where $\Delta > d$. In terms of runtime, the star-advance algorithm is much faster than the $d-1$ approximation and slower than the d-approximation, with regard to exact analyzed runtimes. Looking at the initial question of how to achieve better approximations for (Δ-)TVC on the restricted input of on always star temporal graphs, this can clearly be answered using the provided star-advance algorithm.
Which algorithm a practitioner uses depends heavily on the input data set. If it can be restricted to the class of always star temporal graphs and $\Delta >> d$ does not apply, this thesis showed that the star advance approximation algorithm is the best choice.

6.1.3 Limitations

The used datasets in the experiments were mostly artificially created by the temporal graph generator presented in Section 4.2. As the real-life temporal graph dataset lack a classification into temporal graph classes, especially the always star temporal graph class, we were only able to run the always at most d algorithms on real-life data.
In terms of the approximation ratio, the experiments were limited by the exact solver, whose runtime and space usage were both exponential. Therefore, the available resources are easily exhausted even by small input sizes. In terms of space the algorithm has the dynamic programming table and the dynamic programming recursive function calls, both increasing exponentially with the input. Through the use of an indexed map instead of a full table, the space used was reduced to only contain the actually computed values. Even with this improvement the algorithms aborts on most instances, because there is not enough memory available and can only compute solutions on very small instances.
For further research it could be interesting to look at other exact solvers as the one provided by Akrida et al. [4] as this is almost optimal in terms of runtime, assuming the ETH. This approach still uses dynamic programming, which could lead to a bottleneck for the space usage. As the exact solving is NP-hard in any case, there is no efficient algorithm for this, yet the considerations of other techniques and improvements could lead to the computation of exact solutions for slightly larger inputs, without the computation being aborted by exhausting the space resources.
The experiments ran on a server with slurm scheduler. Therefore, even if the setup for all the experiments was the same for all experiments (8 cores with 100GiB), the other running jobs on the server might have had an impact in the frequency of the CPUs and therefore the runtimes as well.

6.2 Conclusion

This section presents the key findings in this thesis and offers an outlook on further work in the field of temporal graphs.
The aims of the thesis were to explore the state of the art algorithms for extensions of the vertex cover problem into the temporal environment, to evaluate them experimentally and to make improvements for the restricted case of always star temporal graphs. These were fulfilled by giving a survey of the current state of research, providing a temporal graph generator and a framework in which (SW-)TVC can be solved, then presenting two new approximation algorithms, and finally performing several experiments with them.
The presented temporal graph generator enables the generation of temporal graphs with defined temporal graph classes, in particular arbitrary, always at most d-degree, always star and star temporal graphs.
A key achievement of this thesis is the TVC-solver framework, which provides implementations for solving SW-TVC covering five algorithms. When the silding window size Δ

is configured as T, then these algorithms also solve the TVC problem. The framework offers a first implementation for the d and $d-1$ approximation algorithms for always at most degree d temporal graphs presented in [4], [18] and a first implementation of the exact algorithm presented in [18]. Moreover, the framework implements of the two new approximation algorithms presented in this thesis.
This already leads to the second main achievement being the introduction of these two new approximation algorithms for computing Δ-TVC on the restricted case of always star temporal graphs. The first trivial approach includes every star center and leads to a $2\Delta - 1$ approximation ratio in $\mathcal{O}(T)$ runtime. The second approach is based on the more advanced idea, that a star center only needs to be included if there is any edge in the associated timestep not appearing again in any window which includes that timestep. This algorithm offers a $\Delta - 1$ approximation ratio in $\mathcal{O}(Tm\Delta^2)$ runtime.
In conclusion, we provide the implementations for a temporal graph generator and a TVC-solver framework with known algorithms as well as the proposed star-trivial and star-advance approximation algorithms.
The purpose of the experiments was to verify the runtime and approximation ratios stated for the known approximation algorithms and to test them against the proposed algorithms on always star temporal graphs.
For the first aim the runtimes and approximation ratios of the d and $d-1$ approximation algorithms were tested. Therefore, the first experiments were run with increasing edge numbers and lifetime, where the d-approximation stated a linear increase and the $d-1$ approximation a quadratic on in both parameters. While all stated bounds for the d-approximation held, the implementation of the $d-1$ approximation lead to a linear runtime increase over the edge number on arbitrary temporal graphs. Nevertheless, the worst runtime with respect to the number of edges could be shown on underlying star temporal graphs. The ratio verifications were only run on small instances since the exact algorithm runs in exponential time, but they showed the ratios on arbitrary graphs were well within the stated bounds. On real-life instances the $d-1$ approximation algorithm is found to be the best choice considering solution size as well as runtime.
The second part of the experimental evaluation was to test the new proposed approximation algorithms on always star temporal graphs against the best current state of the art being the d and $d-1$ approximation algorithms. The exact ratios again were only testable on small instances. The experiments clearly show that both proposed algorithms outperform the best known algorithm, the $d-1$ approximation, in shorter runtime.
Looking at the approximation ratios when computing a Δ-TVC in the case $\Delta > d$ indicates that one would get a better solution using the $d-1$ approximation algorithm. However, the experiments show that the star-advance algorithm outperforms in shorter runtime the $d-1$ approximation algorithm even in that case on all tested inputs. The star-trivial algorithm computes, as expected, a worse solution when considering very small d, as on underlying star temporal graphs this generates to close to static graphs, the worst case for this algorithm. The experiments on larger instances show that the star-advance algorithm leads to an 40.46% improvement in the solution size and 218.69% improvement

in the runtime compared to the $d-1$-approximation when computing the 16-TVC.
Overall, the experimental evaluation has shown the runtime and quality bounds for the known algorithms and confirmed an improvement in both runtime and ratio of the proposed star-advance algorithm against the $d-1$-approximation algorithm.

6.3 Further work

To build on the research presented in this thesis, there are several directions for further work. One possible area could be to make the temporal graph generation more efficient. As the focus of this thesis was primarily on the approximation algorithms and their implementation and analysis, there may be opportunities to optimise the graph generation process to allow for faster and perhaps parallel generation.
Another area of interest would be to implement other exact or approximation algorithms for (SW)-TVC in the presented TVC-solver framework on general or other restricted temporal graphs or to engineer the implemented ones, as we believe there still could be some improvements made to allow computations on larger graphs in shorter time. Furthermore, the framework could be extended by implementing algorithms for other temporal graph problems such as temporal matching or temporal coloring.
In terms of (SW)-TVC analysis, one could also consider restrictions to other classes of temporal graphs and to analyse these cases in terms of complexity and potential approximation ratios. By studying the problem for these constrained inputs, it may be possible to develop more efficient or even optimal algorithms to compute it.

Bibliography

[1] Faisal N. Abu-Khzam, Sebastian Lamm, Matthias Mnich, Alexander Noe, Christian Schulz, and Darren Strash. Recent Advances in Practical Data Reduction. In Hannah Bast, Claudius Korzen, Ulrich Meyer, and Manuel Penschuck, editors, *Algorithms for Big Data - DFG Priority Program 1736*, volume 13201 of *Lecture Notes in Computer Science*, pages 97–133. Springer, 2022.

[2] Eleni C. Akrida, Leszek Gasieniec, George B. Mertzios, and Paul G. Spirakis. Ephemeral networks with random availability of links: The case of fast networks. *J. Parallel Distributed Comput.*, 87:109–120, 2016.

[3] Eleni C. Akrida, George B. Mertzios, Paul G. Spirakis, and Christoforos L. Raptopoulos. The temporal explorer who returns to the base. *J. Comput. Syst. Sci.*, 120:179–193, 2021.

[4] Eleni C. Akrida, George B. Mertzios, Paul G. Spirakis, and Viktor Zamaraev. Temporal vertex cover with a sliding time window. *J. Comput. Syst. Sci.*, 107:108–123, 2020.

[5] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. Deterministic Fully Dynamic Data Structures for Vertex Cover and Matching. *SIAM J. Comput.*, 47(3):859–887, 2018.

[6] Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Xiaowei Wu. Dynamic Set Cover: Improved Amortized and Worst-Case Update Time. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2537–2549. SIAM, 2021.

[7] Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing Shortest, Fastest, and Foremost Journeys in Dynamic Networks. *Int. J. Found. Comput. Sci.*, 14(2):267–285, 2003.

[8] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *Int. J. Parallel Emergent Distributed Syst.*, 27(5):387–408, 2012.

[9] Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding Temporal Paths Under Waiting Time Constraints. *Algorithmica*, 83(9):2754–2802, 2021.

[10] Arnaud Casteigts, Joseph G. Peters, and Jason Schoeters. Temporal cliques admit sparse spanners. *J. Comput. Syst. Sci.*, 121:1–17, 2021.

[11] Rong-chii Duh and Martin Fürer. Approximation of \emphk-Set Cover by Semi-Local Optimization. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 256–264. ACM, 1997.

[12] Thomas Erlebach and Jakob T. Spooner. Non-strict Temporal Exploration. In Andrea Werneck Richa and Christian Scheideler, editors, *Structural Information and Communication Complexity - 27th International Colloquium, SIROCCO 2020, Paderborn, Germany, June 29 - July 1, 2020, Proceedings*, volume 12156 of *Lecture Notes in Computer Science*, pages 129–145. Springer, 2020.

[13] Marcelo Fonseca Faraj and Christian Schulz. Buffered Streaming Graph Partitioning. *ACM J. Exp. Algorithmics*, 27:1.10:1–1.10:26, 2022.

[14] Afonso Ferreira. Building a reference combinatorial model for MANETs. *IEEE Netw.*, 18(5):24–29, 2004.

[15] Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. Temporal graph classes: A view through temporal separators. *Theor. Comput. Sci.*, 806:197–218, 2020.

[16] George Giakkoupis, Thomas Sauerwald, and Alexandre Stauffer. Randomized Rumor Spreading in Dynamic Graphs. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 495–507. Springer, 2014.

[17] Jawad Haider and Muhammad Fayaz. A Smart Approximation Algorithm for Minimum Vertex Cover Problem based on Min-to-Min (MtM) Strategy. *International Journal of Advanced Computer Science and Applications*, 11(12), 2020. Publisher: The Science and Information Organization.

[18] Thekla Hamm, Nina Klobas, George B. Mertzios, and Paul G. Spirakis. The Complexity of Temporal Vertex Cover in Small-Degree Graphs. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on*

Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022, pages 10193–10201. AAAI Press, 2022.

[19] Kathrin Hanauer, Monika Henzinger, and Christian Schulz. Recent Advances in Fully Dynamic Graph Algorithms - A Quick Reference Guide. *ACM J. Exp. Algorithmics*, 27:1.11:1–1.11:45, 2022.

[20] Petter Holme. Temporal Networks. In Reda Alhajj and Jon G. Rokne, editors, *Encyclopedia of Social Network Analysis and Mining, 2nd Edition*. Springer, 2018.

[21] Wenyu Huo and Vassilis J. Tsotras. Efficient temporal shortest path queries on evolving social graphs. In Christian S. Jensen, Hua Lu, Torben Bach Pedersen, Christian Thomsen, and Kristian Torp, editors, *Conference on Scientific and Statistical Database Management, SSDBM '14, Aalborg, Denmark, June 30 - July 02, 2014*, pages 38:1–38:4. ACM, 2014.

[22] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[23] David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and Inference Problems for Temporal Networks. *J. Comput. Syst. Sci.*, 64(4):820–842, 2002.

[24] Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. Community interaction and conflict on the web. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 933–943. International World Wide Web Conferences Steering Committee, 2018.

[25] Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F. Werneck. Finding near-optimal independent sets at scale. *J. Heuristics*, 23(4):207–229, 2017.

[26] Jure Leskovec, Daniel P. Huttenlocher, and Jon M. Kleinberg. Governance in Social Media: A Case Study of the Wikipedia Promotion Process. In William W. Cohen and Samuel Gosling, editors, *Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010, Washington, DC, USA, May 23-26, 2010*. The AAAI Press, 2010.

[27] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2, 2007.

[28] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford Large Network Dataset Collection, June 2014.

[29] Tiantian Liu, Huan Li, Hua Lu, Muhammad Aamir Cheema, and Lidan Shou. Towards Crowd-aware Indoor Path Planning. *Proc. VLDB Endow.*, 14(8):1365–1377, 2021.

[30] George B. Mertzios, Hendrik Molter, Rolf Niedermeier, Viktor Zamaraev, and Philipp Zschoche. Computing maximum matchings in temporal graphs. *J. Comput. Syst. Sci.*, 137:1–19, 2023.

[31] George B. Mertzios, Hendrik Molter, and Viktor Zamaraev. Sliding window temporal graph coloring. *J. Comput. Syst. Sci.*, 120:97–115, 2021.

[32] Othon Michail and Paul G. Spirakis. Traveling salesman problems in temporal graphs. *Theor. Comput. Sci.*, 634:1–23, 2016.

[33] Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Commun. ACM*, 61(2):72, 2018.

[34] Joel C. Miller and Aric A. Hagberg. Efficient Generation of Networks with Given Expected Degrees. In Alan M. Frieze, Paul Horn, and Pawel Pralat, editors, *Algorithms and Models for the Web Graph - 8th International Workshop, WAW 2011, Atlanta, GA, USA, May 27-29, 2011. Proceedings*, volume 6732 of *Lecture Notes in Computer Science*, pages 115–126. Springer, 2011.

[35] Pietro Panzarasa, Tore Opsahl, and Kathleen M. Carley. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *J. Assoc. Inf. Sci. Technol.*, 60(5):911–932, 2009.

[36] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. Motifs in Temporal Networks. In Maarten de Rijke, Milad Shokouhi, Andrew Tomkins, and Min Zhang, editors, *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, pages 601–610. ACM, 2017.

[37] Manuel Penschuck, Ulrik Brandes, Michael Hamann, Sebastian Lamm, Ulrich Meyer, Ilya Safro, Peter Sanders, and Christian Schulz. Recent Advances in Scalable Network Generation. *CoRR*, abs/2003.00736, 2020. arXiv: 2003.00736.

[38] J. Tang, S. Scellato, M. Musolesi, C. Mascolo, and V. Latora. Small-world behavior in time-varying graphs. *Phys. Rev. E*, 81(5):055101, May 2010. Publisher: American Physical Society.

[39] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.

[40] A. Vazquez. Disordered networks generated by recursive searches. *Europhysics Letters*, 54(4):430, May 2001.

[41] Jordan Viard, Matthieu Latapy, and Clémence Magnien. Revealing contact patterns among high-school students using maximal cliques in link streams. In Jian Pei, Fabrizio Silvestri, and Jie Tang, editors, *Proceedings of the 2015 IEEE/ACM Interna-*

tional Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015, Paris, France, August 25 - 28, 2015, pages 1517–1522. ACM, 2015.

[42] Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *Theor. Comput. Sci.*, 609:245–252, 2016.

[43] Martin Weigt and Alexander K. Hartmann. Number of Guards Needed by a Museum: A Phase Transition in Vertex Covering of Random Graphs. *Phys. Rev. Lett.*, 84(26):6118–6121, June 2000. Publisher: American Physical Society.

[44] Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. Path Problems in Temporal Graphs. *Proc. VLDB Endow.*, 7(9):721–732, 2014.

APPENDIX A

Further Results

This section presents further details of the experiments above. Note that the artificial graphs are described in the form $graphclass\_maxd\_T\_n\_m\_seed$.

A.1 Details of the Experiments on Real-Life Instances

Table A.1: Results for the 64-TVC on real-life temporal graphs

Graph	dapprox		d1approx	
	|Δ-TVC|	t	|Δ-TVC|	t
CollegeMsg	21 649	19 493,071	**19 693**	**9 119,800**
email-Eu-core-temporal-Dept1	25 230	**7 249,439**	**19 605**	12 152,959
email-Eu-core-temporal-Dept2	19 848	**5 731,778**	**17 211**	9 905,366
email-Eu-core-temporal-Dept3	7 308	5 461,386	**5 916**	**3 557,490**
email-Eu-core-temporal-Dept4	16 716	**4 738,784**	**13 157**	11 805,710
soc-redditHyperlinks-body	268 512	884 924,502	**242 497**	**28 868,536**
soc-redditHyperlinks-title	521 559	1 389 602,348	**459 034**	**71 972,082**
sx-mathoverflow-a2q	99 904	1 000 302,961	**92 882**	**9 142,004**
sx-mathoverflow-c2a	113 990	803 290,062	**106 658**	**17 392,584**
sx-mathoverflow-c2q	121 238	994 914,564	**110 552**	**16 404,972**

A.2 Details of the Experiments under the Condition $\Delta < d$

Table A.2: Results for the 3-TVC on small degree always star temporal graphs

Graph	Exact		dapprox		d1approx		startriv		staradv	
	|Δ-TVC|	t	|Δ-TVC|	t	|Δ-TVC|	t	|Δ-TVC|	t	|Δ-TVC|	t
star_10_64_128_564_0	**64**	24 630,741	307	1,464	**64**	18,052	**64**	**0,055**	**64**	3,407
star_10_64_128_548_3	**64**	24 347,696	325	1,378	**64**	17,049	**64**	**0,055**	**64**	3,498
star_15_64_128_796_0	**64**	55 813,187	440	2,694	65	34,736	**64**	**0,061**	**64**	4,542
star_15_64_128_784_3	**64**	57 717,567	478	1,968	71	33,852	**64**	**0,055**	**64**	4,709
star_20_64_128_1024_0	**64**	98 117,401	593	2,695	**64**	33,238	**64**	**0,056**	**64**	6,074
star_20_64_128_981_3	**64**	91 840,151	626	1,980	**64**	53,555	**64**	**0,043**	**64**	6,014
star_25_64_128_1237_0	**64**	162 988,901	750	3,213	65	32,419	**64**	**0,056**	**64**	7,548
star_25_64_128_1173_3	**64**	155 141,819	788	3,108	67	43,529	**64**	**0,056**	**64**	6,886
star_2_64_128_125_0	**64**	857,431	100	0,333	**64**	4,433	**64**	**0,054**	**64**	0,936
star_2_64_128_124_3	**64**	850,592	104	0,324	**64**	3,261	**64**	**0,054**	**64**	0,837
star_30_64_128_1443_0	**64**	227 029,399	906	3,167	**64**	45,736	**64**	**0,056**	**64**	8,675
star_30_64_128_1350_3	**64**	200 734,248	954	3,665	**64**	58,075	**64**	**0,057**	**64**	7,905
star_5_64_128_299_0	**64**	6 143,851	181	0,743	68	8,747	**64**	**0,058**	**64**	1,943
star_5_64_128_291_3	**64**	5 680,281	189	0,748	72	11,059	**64**	**0,055**	**64**	1,843
ustar_10_64_128_10_0	**50**	303,007	126	0,180	63	16,797	63	**0,052**	51	0,240
ustar_10_64_128_10_3	**56**	105,587	61	0,378	62	12,324	63	**0,072**	58	0,369
ustar_15_64_128_15_0	**58**	1 302,143	161	0,271	69	17,410	63	**0,055**	59	0,283
ustar_15_64_128_15_3	**60**	417,719	101	0,240	62	15,305	63	**0,052**	62	0,276
ustar_20_64_128_20_0	**61**	5 210,725	212	0,333	68	19,754	63	**0,054**	**61**	0,312
ustar_20_64_128_20_3	**62**	2 775,284	143	0,306	69	15,558	63	**0,054**	**62**	0,303
ustar_25_64_128_25_0	**63**	40 181,146	279	0,426	80	39,615	**63**	**0,061**	**63**	0,367
ustar_25_64_128_25_3	**63**	10 569,073	183	0,391	74	24,468	**63**	**0,054**	**63**	0,352
ustar_2_64_128_2_0	**28**	3,247	44	0,077	32	3,604	55	**0,041**	30	0,144
ustar_2_64_128_2_3	**24**	1,709	25	0,065	26	1,152	36	**0,028**	26	0,121
ustar_30_64_128_30_0	**63**	127 914,023	316	0,515	80	34,036	**63**	**0,055**	**63**	0,379
ustar_30_64_128_30_3	**63**	27 637,473	193	0,463	72	42,580	**63**	**0,054**	**63**	0,372
ustar_5_64_128_5_0	**34**	42,953	98	0,203	46	12,405	63	**0,052**	36	0,216
ustar_5_64_128_5_3	**46**	19,421	56	0,112	57	4,359	62	**0,051**	51	0,188

Table A.3: Results for the 4-TVC on small degree always star temporal graphs

Graph	Exact		dapprox		d1approx		startriv		staradv	
	|Δ-TVC|	t	|Δ-TVC|	t	|Δ-TVC|	t	|Δ-TVC|	t	|Δ-TVC|	t
star_10_64_128_564_0	**64**	26 360,635	307	1,542	**64**	18,032	**64**	**0,055**	**64**	3,333
star_10_64_128_548_3	**64**	26 005,341	324	1,438	**64**	16,357	**64**	**0,055**	**64**	3,366
star_15_64_128_796_0	**64**	58 346,683	440	2,766	65	37,479	**64**	**0,080**	**64**	6,330
star_15_64_128_784_3	**64**	62 872,995	477	2,076	71	39,789	**64**	**0,054**	**64**	4,743
star_20_64_128_1024_0	**64**	108 204,005	593	2,732	**64**	38,201	**64**	**0,058**	**64**	6,032
star_20_64_128_981_3	**64**	105 096,569	625	2,608	**64**	53,957	**64**	**0,054**	**64**	6,052
star_25_64_128_1237_0	**64**	180 665,710	749	3,354	65	33,481	**64**	**0,057**	**64**	7,443
star_25_64_128_1173_3	**64**	188 687,488	786	3,210	67	45,411	**64**	**0,056**	**64**	6,934
star_2_64_128_125_0	**64**	815,347	100	0,342	**64**	2,293	**64**	**0,055**	**64**	0,848
star_2_64_128_124_3	**64**	830,142	104	0,338	**64**	3,364	**64**	**0,053**	**64**	0,864
star_30_64_128_1443_0	**64**	280 464,185	905	3,972	**64**	42,995	**64**	**0,060**	**64**	8,538
star_30_64_128_1350_3	**64**	282 755,102	951	3,811	**64**	57,302	**64**	**0,056**	**64**	8,000
star_5_64_128_299_0	**64**	5 868,629	181	0,760	68	8,670	**64**	**0,057**	**64**	1,711
star_5_64_128_291_3	**64**	5 648,038	189	0,766	72	11,952	**64**	**0,055**	**64**	1,890
ustar_10_64_128_10_0	**47**	907,205	108	0,165	59	18,761	63	**0,055**	52	0,341
ustar_10_64_128_10_3	**48**	368,975	55	0,301	59	12,452	63	**0,075**	52	0,447
ustar_15_64_128_15_0	**54**	5 672,509	140	0,241	67	16,201	63	**0,056**	57	0,364
ustar_15_64_128_15_3	**54**	2 822,163	91	0,209	61	16,702	63	**0,052**	56	0,342
ustar_20_64_128_20_0	**57**	36 735,923	186	0,288	72	24,180	63	**0,054**	**57**	0,379
ustar_20_64_128_20_3	**57**	28 419,462	127	0,268	66	17,366	63	**0,052**	**57**	0,400
ustar_25_64_128_25_0	**58**	600 265,517	237	0,392	75	46,645	63	**0,054**	**58**	0,459
ustar_25_64_128_25_3	**59**	179 791,849	160	0,342	70	25,585	63	**0,055**	**59**	0,453
ustar_2_64_128_2_0	**21**	4,409	34	0,069	23	4,048	55	**0,041**	25	0,201
ustar_2_64_128_2_3	**21**	1,639	22	0,058	25	1,492	36	**0,040**	23	0,146
ustar_30_64_128_30_0	**62**	3 883 609,293	268	0,422	77	39,002	63	**0,052**	**62**	0,461
ustar_30_64_128_30_3	**61**	771 898,714	170	0,385	68	47,026	63	**0,054**	**61**	0,357
ustar_5_64_128_5_0	**28**	87,201	77	0,116	42	14,377	63	**0,051**	32	0,306
ustar_5_64_128_5_3	**39**	30,748	48	0,101	50	4,409	62	**0,051**	43	0,238

A.3 Details of the Experiments under the Condition $\Delta > d$

Table A.4: Results for the 20-TVC on small degree always star temporal graphs

	Exact		dapprox		d1approx		startriv		staradv	
Graph	$\|\Delta$-TVC$\|$	t	$\|\Delta$-TVC$\|$	t	$\|\Delta$-TVC$\|$	t	$\|\Delta$-TVC$\|$	t	$\|\Delta$-TVC$\|$	t
star_2_64_128_125_0	**64**	1 154,255	100	0,480	**64**	4,780	**64**	**0,057**	**64**	0,874
star_2_64_128_124_3	**64**	1 019,370	104	0,464	**64**	3,909	**64**	**0,056**	**64**	0,856
star_2_64_128_127_5	**64**	679,698	91	0,439	**64**	3,234	**64**	**0,054**	**64**	0,857
star_3_64_128_184_0	**64**	13 416,341	127	0,649	73	6,050	**64**	**0,052**	**64**	1,595
star_3_64_128_181_3	**64**	10 404,411	135	0,638	78	5,208	**64**	**0,054**	**64**	1,158
star_3_64_128_188_5	**64**	3 219,050	116	0,704	70	5,514	**64**	**0,054**	**64**	1,190
star_4_64_128_243_0	**64**	120 337,030	153	0,843	**64**	6,483	**64**	**0,070**	**64**	1,511
star_4_64_128_236_3	**64**	52 325,829	165	0,818	**64**	6,106	**64**	**0,053**	**64**	1,433
star_4_64_128_245_5	**64**	13 280,033	137	0,877	**64**	5,688	**64**	**0,056**	**64**	1,552
star_5_64_128_299_0	**64**	247 481,594	180	1,004	68	6,628	**64**	**0,056**	**64**	1,770
star_5_64_128_291_3	**64**	151 024,671	186	1,082	72	7,844	**64**	**0,054**	**64**	1,799
star_5_64_128_296_5	**64**	61 910,675	162	0,994	68	12,446	**64**	**0,078**	**64**	2,515
star_6_64_128_353_0	**64**	858 697,847	201	1,247	**64**	9,652	**64**	**0,058**	**64**	2,019
star_6_64_128_345_3	**64**	599 432,356	210	1,177	**64**	10,446	**64**	**0,056**	**64**	2,012
star_6_64_128_350_5	**64**	222 513,955	190	1,164	**64**	8,658	**64**	**0,054**	**64**	2,031
star_7_64_128_411_0	**64**	1 263 262,993	229	1,376	67	8,807	**64**	**0,056**	**64**	2,351
star_7_64_128_396_3	**64**	840 424,921	235	1,354	72	10,769	**64**	**0,055**	**64**	2,282
star_7_64_128_406_5	**64**	590 604,531	215	1,367	65	10,658	**64**	**0,055**	**64**	2,371
star_8_64_128_463_0	**64**	1 417 327,784	252	1,565	**64**	13,330	**64**	**0,057**	**64**	2,778
star_8_64_128_446_3	**64**	3 197 551,518	260	1,539	**64**	15,443	**64**	**0,054**	**64**	2,563
star_8_64_128_460_5	**64**	1 053 271,975	239	1,604	**64**	17,808	**64**	**0,057**	**64**	2,726
ustar_2_64_128_2_0	**3**	5,091	6	**0,039**	6	3,902	55	0,040	**3**	1,470
ustar_2_64_128_2_3	**4**	2,462	6	0,042	6	2,278	36	**0,027**	7	0,825
ustar_2_64_128_2_5	**3**	1,845	6	**0,042**	**3**	1,611	61	0,047	9	1,713
ustar_3_64_128_3_0	**3**	95,053	9	**0,046**	**3**	4,698	59	0,048	**3**	2,116
ustar_3_64_128_3_3	**4**	45,910	9	0,046	9	2,693	52	**0,031**	8	1,167
ustar_3_64_128_3_5	**3**	5,558	9	**0,046**	6	1,366	63	0,050	9	2,287
ustar_4_64_128_4_0	**4**	786,300	12	0,050	10	6,151	59	**0,047**	7	2,188
ustar_4_64_128_4_3	**6**	733,235	10	0,048	12	2,679	60	**0,044**	8	1,559
ustar_4_64_128_4_5	**3**	105,900	12	**0,051**	6	5,954	63	0,053	10	3,016
ustar_5_64_128_5_0	**4**	8 617,290	15	0,054	12	4,077	63	**0,051**	7	2,997
ustar_5_64_128_5_3	**6**	10 541,954	11	0,054	15	5,315	62	**0,052**	8	2,225
ustar_5_64_128_5_5	**3**	2 912,667	15	0,056	8	4,575	63	**0,054**	6	3,256
ustar_6_64_128_6_0	**5**	49 055,531	17	0,059	15	4,519	63	**0,052**	15	2,936
ustar_6_64_128_6_3	**8**	18 058,933	13	0,058	19	4,854	62	**0,051**	**8**	2,090
ustar_6_64_128_6_5	**4**	79 407,595	18	0,063	9	7,231	63	**0,055**	10	3,909
ustar_7_64_128_7_0	**5**	453 595,579	20	0,084	17	4,713	63	**0,055**	18	3,287
ustar_7_64_128_7_3	**8**	100 928,573	15	0,062	21	4,626	62	**0,053**	9	2,158
ustar_7_64_128_7_5	**6**	408 463,550	22	0,066	10	9,940	63	**0,054**	16	3,904
ustar_8_64_128_8_0	**7**	4 774 784,390	21	0,119	20	7,518	63	**0,055**	16	3,095
ustar_8_64_128_8_3	**8**	1 005 514,269	16	0,066	18	7,532	63	**0,054**	10	2,511
ustar_8_64_128_8_5	**7**	6 158 481,487	25	0,073	11	7,411	63	**0,054**	17	4,029

A.4 Details of the Experiments on larger Always Star Temporal Graphs

Table A.5: Results for the 16-TVC on larger always star temporal graphs

	dapprox		d1approx		startriv		staradv	
Graph	$\|\Delta$-TVC$\|$	t	$\|\Delta$-TVC$\|$	t	$\|\Delta$-TVC$\|$	t	$\|\Delta$-TVC$\|$	t
star_100_11707_1024_22126_0	500 921	11 757,671	13 559	46 374,379	**11 707**	**11,554**	**11 707**	19 821,947
star_100_11707_1024_21449_3	485 381	11 404,086	13 509	47 745,287	**11 707**	**11,737**	**11 707**	19 234,884
star_100_11707_1024_21770_5	493 042	11 590,530	13 573	47 700,282	**11 707**	**12,036**	**11 707**	20 068,499
star_50_11707_1024_13930_0	267 916	7 173,214	12 796	21 274,967	**11 707**	**12,659**	**11 707**	12 730,092
star_50_11707_1024_13471_3	259 908	6 868,859	12 817	21 130,757	**11 707**	**11,209**	**11 707**	12 200,148
star_50_11707_1024_13765_5	264 158	7 395,377	12 872	21 490,555	**11 707**	**12,384**	**11 707**	12 895,693

star_5_11707_1024_6340_0	33 635	3 179,968	12 875	1 774,420	**11 707**	**9,218**	**11 707**	5 704,968
star_5_11707_1024_6225_3	33 120	3 071,300	12 778	1 852,149	**11 707**	**9,222**	**11 707**	5 591,945
star_5_11707_1024_6289_5	33 473	3 123,525	12 888	1 745,913	**11 707**	**9,234**	**11 707**	5 631,434
star_100_11707_4096_21954_0	488 334	11 530,723	13 629	48 212,073	**11 707**	**11,568**	**11 707**	20 297,328
star_100_11707_4096_22187_3	490 971	11 671,493	13 547	46 348,544	**11 707**	**12,328**	**11 707**	19 861,291
star_100_11707_4096_22007_5	486 798	11 360,290	13 583	47 681,916	**11 707**	**12,880**	**11 707**	19 618,950
star_50_11707_4096_13768_0	261 494	7 116,953	12 877	20 988,240	**11 707**	**12,057**	**11 707**	12 520,851
star_50_11707_4096_13967_3	262 926	7 102,937	12 867	21 525,114	**11 707**	**11,178**	**11 707**	12 790,532
star_50_11707_4096_13909_5	260 629	7 308,158	12 878	21 847,768	**11 707**	**11,389**	**11 707**	12 746,835
star_5_11707_4096_6293_0	33 327	3 152,780	12 832	1 748,769	**11 707**	**9,403**	**11 707**	5 765,765
star_5_11707_4096_6358_3	33 439	3 290,806	12 804	1 794,649	**11 707**	**9,680**	**11 707**	5 896,473
star_5_11707_4096_6401_5	33 302	3 205,310	12 768	1 778,918	**11 707**	**9,306**	**11 707**	5 727,456
star_100_14045_1024_22784_0	599 043	14 569,003	16 266	57 329,785	**14 045**	**15,548**	**14 045**	25 305,339
star_100_14045_1024_22217_3	582 401	13 928,273	16 225	57 011,371	**14 045**	**14,219**	**14 045**	23 927,530
star_100_14045_1024_22335_5	589 981	14 161,514	16 259	57 155,197	**14 045**	**14,084**	**14 045**	24 454,375
star_50_14045_1024_14448_0	320 532	8 820,032	15 374	25 795,469	**14 045**	**14,779**	**14 045**	15 747,823
star_50_14045_1024_14033_3	311 649	8 589,266	15 381	25 447,548	**14 045**	**14,366**	**14 045**	14 876,722
star_50_14045_1024_14200_5	316 222	8 980,295	15 453	26 078,356	**14 045**	**13,831**	**14 045**	15 681,821
star_5_14045_1024_6663_0	40 336	4 171,156	15 461	2 176,461	**14 045**	**11,163**	**14 045**	7 164,989
star_5_14045_1024_6588_3	39 647	4 272,750	15 329	2 226,976	**14 045**	**11,142**	**14 045**	7 322,980
star_5_14045_1024_6621_5	40 013	4 009,637	15 418	2 158,981	**14 045**	**11,265**	**14 045**	7 210,725
star_100_14045_4096_22705_0	584 077	14 430,714	16 298	58 871,562	**14 045**	**15,043**	**14 045**	24 422,580
star_100_14045_4096_22734_3	587 859	14 785,812	16 282	57 490,557	**14 045**	**15,383**	**14 045**	25 665,256
star_100_14045_4096_22759_5	579 465	14 476,052	16 266	57 351,081	**14 045**	**14,437**	**14 045**	24 503,708
star_50_14045_4096_14325_0	312 876	9 039,127	15 423	26 363,451	**14 045**	**14,167**	**14 045**	15 315,741
star_50_14045_4096_14420_3	314 805	8 853,127	15 442	25 924,275	**14 045**	**13,967**	**14 045**	15 689,124
star_50_14045_4096_14447_5	310 600	9 078,838	15 429	26 023,150	**14 045**	**13,535**	**14 045**	16 013,614
star_5_14045_4096_6678_0	39 916	4 178,478	15 356	2 180,308	**14 045**	**10,882**	**14 045**	7 223,188
star_5_14045_4096_6686_3	40 057	3 972,368	15 389	2 257,906	**14 045**	**10,839**	**14 045**	7 114,840
star_5_14045_4096_6722_5	39 784	4 158,938	15 290	2 134,444	**14 045**	**10,965**	**14 045**	7 354,068
star_100_2354_1024_16456_0	102 829	1 781,804	2 713	9 670,128	**2 354**	**2,020**	**2 354**	3 010,554
star_100_2354_1024_15571_3	99 151	1 777,638	2 706	9 903,041	**2 354**	**1,992**	**2 354**	2 763,978
star_100_2354_1024_15950_5	97 343	1 708,911	2 759	9 819,751	**2 354**	**2,007**	**2 354**	3 013,584
star_50_2354_1024_9869_0	54 743	1 042,883	2 562	4 295,936	**2 354**	**1,969**	**2 354**	1 805,254
star_50_2354_1024_9483_3	52 816	977,546	2 563	4 381,554	**2 354**	**1,890**	**2 354**	1 717,415
star_50_2354_1024_9582_5	52 212	1 015,710	2 602	4 282,722	**2 354**	**1,892**	**2 354**	1 791,758
star_5_2354_1024_3673_0	6 862	385,758	2 612	358,598	**2 354**	**1,899**	**2 354**	688,203
star_5_2354_1024_3684_3	6 666	386,385	2 584	384,316	**2 354**	**1,835**	**2 354**	677,394
star_5_2354_1024_3581_5	6 735	362,193	2 598	365,645	**2 354**	**1,902**	**2 354**	648,349
star_100_2354_4096_15722_0	98 528	1 642,638	2 699	9 464,577	**2 354**	**1,954**	**2 354**	2 834,355
star_100_2354_4096_16554_3	101 592	1 843,739	2 710	9 737,382	**2 354**	**1,960**	**2 354**	2 996,988
star_100_2354_4096_16579_5	98 749	1 711,114	2 766	9 489,266	**2 354**	**2,043**	**2 354**	3 012,117
star_50_2354_4096_9502_0	52 746	1 070,132	2 575	4 358,206	**2 354**	**1,947**	**2 354**	1 769,241
star_50_2354_4096_9965_3	54 096	1 029,706	2 597	4 313,696	**2 354**	**1,923**	**2 354**	1 775,190
star_50_2354_4096_9990_5	52 890	1 018,817	2 622	4 315,972	**2 354**	**1,903**	**2 354**	1 827,569
star_5_2354_4096_3627_0	6 696	369,347	2 560	365,837	**2 354**	**1,871**	**2 354**	659,811
star_5_2354_4096_3672_3	6 853	373,436	2 570	366,613	**2 354**	**1,929**	**2 354**	658,646
star_5_2354_4096_3734_5	6 755	386,099	2 578	358,136	**2 354**	**1,791**	**2 354**	703,070
star_100_4692_1024_19147_0	201 762	3 938,203	5 390	18 776,287	**4 692**	**3,902**	**4 692**	6 855,237
star_100_4692_1024_18043_3	198 696	3 776,812	5 404	18 970,826	**4 692**	**4,344**	**4 692**	6 421,203
star_100_4692_1024_18667_5	197 593	3 836,926	5 455	19 537,807	**4 692**	**3,961**	**4 692**	6 879,251
star_50_4692_1024_11728_0	107 664	2 638,712	5 083	8 295,084	**4 692**	**3,898**	**4 692**	4 130,379
star_50_4692_1024_11201_3	105 878	2 343,003	5 125	8 517,608	**4 692**	**3,767**	**4 692**	4 050,515
star_50_4692_1024_11452_5	105 967	2 340,158	5 169	8 713,020	**4 692**	**3,873**	**4 692**	4 121,413
star_5_4692_1024_4764_0	13 514	1 055,863	5 165	689,678	**4 692**	**3,956**	**4 692**	1 700,000
star_5_4692_1024_4734_3	13 309	1 025,434	5 141	689,425	**4 692**	**3,933**	**4 692**	1 763,140
star_5_4692_1024_4712_5	13 478	930,577	5 156	712,920	**4 692**	**3,766**	**4 692**	1 770,965
star_100_4692_4096_18077_0	195 644	3 813,686	5 444	18 505,426	**4 692**	**3,959**	**4 692**	6 478,127
star_100_4692_4096_19116_3	201 824	3 937,803	5 382	18 946,264	**4 692**	**4,070**	**4 692**	7 003,442
star_100_4692_4096_18733_5	194 361	3 905,430	5 487	19 313,325	**4 692**	**4,254**	**4 692**	6 900,983
star_50_4692_4096_11132_0	104 868	2 272,529	5 154	8 351,121	**4 692**	**3,756**	**4 692**	3 990,558
star_50_4692_4096_11717_3	107 686	2 394,353	5 148	8 594,615	**4 692**	**3,854**	**4 692**	4 297,348
star_50_4692_4096_11552_5	104 379	2 733,495	5 186	8 718,171	**4 692**	**3,921**	**4 692**	4 164,974
star_5_4692_4096_4738_0	13 385	952,287	5 121	701,128	**4 692**	**3,739**	**4 692**	1 713,703
star_5_4692_4096_4798_3	13 564	1 067,044	5 134	731,389	**4 692**	**3,826**	**4 692**	1 759,727
star_5_4692_4096_4797_5	13 401	924,211	5 132	726,043	**4 692**	**3,705**	**4 692**	1 741,391
star_100_7030_1024_20426_0	303 655	6 278,545	8 107	28 316,814	**7 030**	**6,278**	**7 030**	11 106,534
star_100_7030_1024_19517_3	291 581	6 635,713	8 078	27 844,636	**7 030**	**6,989**	**7 030**	10 855,845
star_100_7030_1024_19848_5	298 971	6 076,525	8 143	27 663,215	**7 030**	**5,823**	**7 030**	10 691,886
star_50_7030_1024_12676_0	161 958	3 925,702	7 642	12 664,173	**7 030**	**5,989**	**7 030**	6 893,937

star_50_7030_1024_12143_3	155 945	3 717,791	7 704	12 963,919	**7 030**	**5,948**	**7 030**	6 468,956
star_50_7030_1024_12370_5	160 063	3 744,237	7 756	13 073,418	**7 030**	**5,635**	**7 030**	6 733,403
star_5_7030_1024_5445_0	20 218	1 677,491	7 731	1 054,603	**7 030**	**5,748**	**7 030**	2 970,120
star_5_7030_1024_5377_3	19 885	1 662,078	7 669	1 038,797	**7 030**	**5,406**	**7 030**	2 952,925
star_5_7030_1024_5411_5	20 225	1 634,294	7 757	1 075,554	**7 030**	**5,538**	**7 030**	2 961,485
star_100_7030_4096_20021_0	293 315	6 215,703	8 173	26 574,309	**7 030**	**6,188**	**7 030**	10 975,874
star_100_7030_4096_20303_3	297 247	6 207,147	8 103	28 857,511	**7 030**	**5,984**	**7 030**	10 938,785
star_100_7030_4096_20142_5	290 152	6 230,430	8 212	29 386,717	**7 030**	**6,325**	**7 030**	10 755,033
star_50_7030_4096_12352_0	157 111	3 900,416	7 710	13 160,286	**7 030**	**5,616**	**7 030**	6 649,863
star_50_7030_4096_12583_3	159 108	3 855,095	7 717	12 580,172	**7 030**	**5,607**	**7 030**	6 749,396
star_50_7030_4096_12560_5	155 555	4 630,327	7 774	13 459,074	**7 030**	**6,075**	**7 030**	6 949,141
star_5_7030_4096_5373_0	20 012	1 607,005	7 695	1 064,980	**7 030**	**5,441**	**7 030**	2 915,401
star_5_7030_4096_5468_3	20 136	1 800,084	7 676	1 032,711	**7 030**	**5,392**	**7 030**	2 926,502
star_5_7030_4096_5492_5	19 951	1 835,451	7 651	1 073,902	**7 030**	**5,540**	**7 030**	2 967,702
star_100_9369_1024_21465_0	403 842	9 653,248	10 845	37 719,498	**9 369**	**9,191**	**9 369**	15 873,637
star_100_9369_1024_20692_3	389 110	8 630,456	10 806	37 426,241	**9 369**	**9,720**	**9 369**	14 776,156
star_100_9369_1024_20948_5	397 767	8 708,228	10 862	37 461,994	**9 369**	**9,534**	**9 369**	15 085,356
star_50_9369_1024_13425_0	215 576	5 504,672	10 216	16 977,443	**9 369**	**9,091**	**9 369**	9 653,507
star_50_9369_1024_12921_3	208 145	5 405,966	10 249	17 273,177	**9 369**	**9,113**	**9 369**	9 401,788
star_50_9369_1024_13164_5	213 002	5 457,557	10 306	17 797,152	**9 369**	**9,533**	**9 369**	9 510,210
star_5_9369_1024_5954_0	26 920	2 657,104	10 305	1 469,272	**9 369**	**7,623**	**9 369**	4 293,960
star_5_9369_1024_5848_3	26 533	2 401,356	10 214	1 458,734	**9 369**	**7,409**	**9 369**	4 283,171
star_5_9369_1024_5908_5	26 947	2 433,127	10 332	1 388,696	**9 369**	**7,329**	**9 369**	4 315,038
star_100_9369_4096_21185_0	392 868	8 925,401	10 920	38 065,854	**9 369**	**9,123**	**9 369**	14 942,135
star_100_9369_4096_21440_3	394 344	8 870,725	10 809	37 580,647	**9 369**	**9,991**	**9 369**	15 384,308
star_100_9369_4096_21303_5	389 410	8 846,750	10 901	38 274,586	**9 369**	**9,998**	**9 369**	15 416,756
star_50_9369_4096_13173_0	210 328	5 423,646	10 322	17 194,751	**9 369**	**9,017**	**9 369**	9 493,572
star_50_9369_4096_13375_3	211 083	5 753,605	10 276	16 889,491	**9 369**	**8,749**	**9 369**	9 685,126
star_50_9369_4096_13394_5	208 567	5 655,034	10 357	17 438,441	**9 369**	**9,203**	**9 369**	9 958,550
star_5_9369_4096_5894_0	26 795	2 396,164	10 280	1 397,112	**9 369**	**7,541**	**9 369**	4 249,876
star_5_9369_4096_5961_3	26 764	2 415,739	10 242	1 423,002	**9 369**	**7,556**	**9 369**	4 450,545
star_5_9369_4096_5990_5	26 657	2 394,540	10 205	1 380,103	**9 369**	**8,032**	**9 369**	4 331,863
ustar_100_11707_1024_100_0	35 409	67,347	2 801	19 177,590	11 672	**32,740**	**1 291**	6 131,939
ustar_100_11707_1024_100_3	7 240	54,974	2 530	17 930,730	11 705	**36,007**	**1 294**	6 229,118
ustar_100_11707_1024_100_5	38 428	65,272	2 991	19 232,955	11 706	**41,833**	**1 326**	6 729,020
ustar_50_11707_1024_50_0	16 582	35,606	1 971	9 699,211	11 662	**32,835**	**1 058**	2 991,117
ustar_50_11707_1024_50_3	5 485	**23,170**	2 118	10 746,668	11 703	33,187	**1 046**	3 226,722
ustar_50_11707_1024_50_5	20 839	**33,656**	2 230	11 221,738	11 704	39,072	**1 009**	3 763,876
ustar_5_11707_1024_5_0	1 934	**5,944**	774	2 291,969	10 723	32,689	**719**	409,244
ustar_5_11707_1024_5_3	1 046	**5,582**	726	1 316,726	10 223	23,268	**675**	323,700
ustar_5_11707_1024_5_5	2 450	**6,010**	771	3 607,897	11 700	36,853	**763**	503,463
ustar_100_11707_4096_100_0	7 472	57,119	3 562	28 756,604	11 667	**38,430**	**1 351**	6 914,350
ustar_100_11707_4096_100_3	6 301	55,732	2 540	16 236,386	11 699	**35,904**	**1 312**	6 003,413
ustar_100_11707_4096_100_5	4 005	52,583	2 814	21 892,437	11 706	**39,191**	**1 314**	6 507,262
ustar_50_11707_4096_50_0	6 428	**30,341**	2 867	17 104,686	11 358	34,117	**1 067**	3 267,243
ustar_50_11707_4096_50_3	4 461	**28,943**	2 078	9 609,171	11 694	35,453	**1 061**	2 948,323
ustar_50_11707_4096_50_5	3 705	**28,165**	2 143	13 718,278	11 674	33,038	**1 040**	2 971,590
ustar_5_11707_4096_5_0	**643**	**5,634**	670	2 264,421	9 855	27,794	658	382,155
ustar_5_11707_4096_5_3	1 745	**5,798**	1 034	3 954,429	11 012	24,095	**732**	400,351
ustar_5_11707_4096_5_5	1 350	**5,876**	1 356	6 556,056	11 629	38,576	**771**	470,869
ustar_100_14045_1024_100_0	42 845	76,422	3 733	35 300,881	14 044	**53,637**	**1 464**	7 662,910
ustar_100_14045_1024_100_3	7 126	63,400	3 740	22 993,125	14 043	**51,054**	**1 449**	7 312,494
ustar_100_14045_1024_100_5	44 416	80,318	3 039	19 000,937	14 001	**45,034**	**1 455**	7 676,640
ustar_50_14045_1024_50_0	21 786	**39,347**	3 575	21 691,109	13 880	50,172	**1 182**	4 107,398
ustar_50_14045_1024_50_3	4 635	**33,467**	2 405	11 913,710	14 043	46,760	**1 192**	3 376,279
ustar_50_14045_1024_50_5	21 724	**38,825**	2 237	10 522,597	13 990	48,079	**1 202**	3 857,094
ustar_5_14045_1024_5_0	2 271	**6,891**	1 162	2 853,075	13 828	46,816	**914**	487,256
ustar_5_14045_1024_5_3	498	**6,538**	465	384,578	5 999	8,394	**418**	199,699
ustar_5_14045_1024_5_5	2 058	**6,831**	1 020	2 584,628	10 540	32,470	**705**	437,956
ustar_100_14045_4096_100_0	7 359	66,177	3 348	21 522,534	14 043	**51,173**	**1 471**	7 683,996
ustar_100_14045_4096_100_3	7 017	64,632	3 216	20 331,752	14 038	**53,530**	**1 456**	7 539,955
ustar_100_14045_4096_100_5	6 253	64,283	3 564	31 151,148	13 996	**51,020**	**1 499**	7 648,056
ustar_50_14045_4096_50_0	6 358	**34,167**	2 916	15 109,704	13 855	51,151	**1 176**	4 063,052
ustar_50_14045_4096_50_3	4 980	**33,125**	2 702	12 252,796	14 038	45,359	**1 190**	3 666,720
ustar_50_14045_4096_50_5	5 218	**34,780**	3 032	21 069,261	13 981	51,194	**1 225**	3 715,837
ustar_5_14045_4096_5_0	1 805	**7,316**	1 214	4 039,073	13 046	37,863	**860**	554,344
ustar_5_14045_4096_5_3	2 684	**7,413**	1 383	6 770,807	13 467	49,044	**885**	601,050
ustar_5_14045_4096_5_5	1 539	**6,696**	1 386	12 391,540	13 144	44,503	**861**	510,968
ustar_100_2354_1024_100_0	8 209	13,212	1 094	5 063,488	2 350	**2,816**	**639**	1 345,236
ustar_100_2354_1024_100_3	1 569	11,322	996	4 125,710	2 353	**2,875**	**631**	1 287,012

ustar_100_2354_1024_100_5	7 024	13,212	1 111	3 407,977	2 353	**2,770**	**690**	1 145,688
ustar_50_2354_1024_50_0	4 029	6,977	694	1 743,094	2 279	**2,629**	**437**	710,460
ustar_50_2354_1024_50_3	1 319	6,180	689	2 580,170	2 353	**2,885**	**428**	651,380
ustar_50_2354_1024_50_5	3 540	7,022	752	2 150,314	2 353	**2,824**	**463**	558,998
ustar_5_2354_1024_5_0	297	**1,270**	**153**	205,928	1 950	2,025	157	67,742
ustar_5_2354_1024_5_3	238	**1,260**	192	167,656	1 792	2,063	**150**	71,651
ustar_5_2354_1024_5_5	339	**1,852**	**158**	149,042	1 916	1,911	163	69,318
ustar_100_2354_4096_100_0	1 652	11,174	991	3 607,103	2 353	**2,930**	**672**	1 143,752
ustar_100_2354_4096_100_3	1 714	11,426	976	3 394,842	2 353	**2,846**	**655**	1 233,398
ustar_100_2354_4096_100_5	1 426	11,333	1 014	4 358,043	2 353	**2,840**	**678**	1 279,970
ustar_50_2354_4096_50_0	1 424	6,244	625	2 320,766	2 353	**2,850**	**427**	607,809
ustar_50_2354_4096_50_3	1 217	6,230	644	1 855,090	2 298	**2,997**	**474**	655,045
ustar_50_2354_4096_50_5	1 076	6,062	755	2 421,178	2 350	**2,773**	**468**	654,143
ustar_5_2354_4096_5_0	195	**1,371**	205	463,551	1 753	1,985	**152**	81,597
ustar_5_2354_4096_5_3	397	**1,431**	225	344,852	2 238	2,675	**177**	77,078
ustar_5_2354_4096_5_5	228	**0,946**	238	643,905	2 110	1,718	**174**	109,501
ustar_100_4692_1024_100_0	12 778	26,178	1 464	6 341,132	4 691	**7,661**	**844**	2 206,666
ustar_100_4692_1024_100_3	2 596	23,117	1 432	6 018,463	4 691	**7,988**	**832**	2 575,368
ustar_100_4692_1024_100_5	14 264	26,468	1 795	6 566,760	4 691	**8,146**	**857**	2 348,928
ustar_50_4692_1024_50_0	6 289	13,266	995	3 854,085	4 691	**7,090**	**606**	1 160,019
ustar_50_4692_1024_50_3	1 784	11,521	1 034	3 123,691	4 691	**7,878**	**574**	1 321,274
ustar_50_4692_1024_50_5	6 854	13,725	1 297	3 657,873	4 691	**7,783**	**567**	1 165,423
ustar_5_4692_1024_5_0	429	**2,333**	312	299,681	3 977	4,876	**290**	107,053
ustar_5_4692_1024_5_3	499	**2,351**	**327**	307,835	4 517	5,220	332	139,675
ustar_5_4692_1024_5_5	668	**2,433**	**288**	559,835	3 904	5,534	289	138,179
ustar_100_4692_4096_100_0	3 619	22,816	1 464	9 833,122	4 650	**8,047**	**893**	2 445,113
ustar_100_4692_4096_100_3	3 053	21,962	1 629	10 229,905	4 690	**8,851**	**839**	2 931,863
ustar_100_4692_4096_100_5	2 205	21,310	1 606	8 980,153	4 691	**9,044**	**868**	2 670,985
ustar_50_4692_4096_50_0	2 683	11,896	974	5 803,003	4 634	**8,730**	**630**	1 307,196
ustar_50_4692_4096_50_3	2 459	12,248	1 321	8 039,746	4 677	**8,729**	**594**	1 558,858
ustar_50_4692_4096_50_5	2 129	11,802	1 222	5 738,743	4 691	**8,790**	**581**	1 476,267
ustar_5_4692_4096_5_0	538	**2,617**	369	1 686,684	3 252	5,067	**241**	141,627
ustar_5_4692_4096_5_3	899	**2,574**	631	2 202,088	4 432	7,610	**318**	217,026
ustar_5_4692_4096_5_5	489	**2,591**	436	1 681,287	3 824	6,727	**272**	178,145
ustar_100_7030_1024_100_0	23 690	41,310	2 414	18 632,435	7 002	**16,021**	**1 051**	4 083,797
ustar_100_7030_1024_100_3	3 245	31,660	1 996	8 072,090	7 029	**16,494**	**1 033**	3 276,506
ustar_100_7030_1024_100_5	22 490	39,974	2 061	14 210,834	7 022	**14,499**	**1 030**	3 987,761
ustar_50_7030_1024_50_0	11 247	19,683	1 813	10 353,227	6 914	**15,662**	**747**	1 958,017
ustar_50_7030_1024_50_3	2 454	17,014	1 470	4 396,982	7 028	**15,985**	**743**	1 631,887
ustar_50_7030_1024_50_5	10 959	20,267	1 603	7 283,285	7 019	**14,150**	**759**	1 963,048
ustar_5_7030_1024_5_0	864	**3,637**	482	739,316	6 436	12,915	**439**	192,091
ustar_5_7030_1024_5_3	285	**3,308**	287	422,028	3 714	6,086	**272**	124,280
ustar_5_7030_1024_5_5	810	**3,576**	339	653,848	4 280	7,674	**312**	178,238
ustar_100_7030_4096_100_0	4 611	32,750	1 807	10 237,909	7 029	**11,609**	**939**	4 234,822
ustar_100_7030_4096_100_3	4 604	32,647	1 978	14 521,770	6 961	**15,764**	**1 020**	3 791,835
ustar_100_7030_4096_100_5	3 062	31,328	2 132	13 322,537	7 029	**14,399**	**1 000**	3 994,468
ustar_50_7030_4096_50_0	3 723	17,603	1 354	5 606,070	7 029	**14,261**	**755**	2 009,259
ustar_50_7030_4096_50_3	3 112	17,255	1 426	9 138,213	6 945	**15,718**	**749**	1 968,541
ustar_50_7030_4096_50_5	2 905	17,518	1 595	8 399,080	6 740	**14,012**	**745**	1 903,728
ustar_5_7030_4096_5_0	395	**3,505**	381	542,011	4 861	6,617	**341**	144,307
ustar_5_7030_4096_5_3	927	**3,927**	542	648,743	6 627	10,990	**460**	210,376
ustar_5_7030_4096_5_5	1 061	**3,972**	724	4 005,550	5 599	12,761	**401**	225,516
ustar_100_9369_1024_100_0	25 800	52,215	2 344	13 437,311	9 215	**21,132**	**1 155**	4 498,207
ustar_100_9369_1024_100_3	7 259	44,499	2 202	17 416,668	9 367	**25,125**	**1 186**	5 573,948
ustar_100_9369_1024_100_5	28 205	51,753	2 602	17 192,247	9 143	**21,380**	**1 181**	4 838,071
ustar_50_9369_1024_50_0	12 818	27,285	1 662	8 476,812	9 070	**22,291**	**853**	2 385,345
ustar_50_9369_1024_50_3	4 955	**23,310**	1 763	10 119,711	9 364	26,902	**873**	2 778,250
ustar_50_9369_1024_50_5	14 956	26,118	1 931	10 984,717	9 123	**21,036**	**881**	2 665,844
ustar_5_9369_1024_5_0	932	**4,461**	**322**	798,964	4 697	9,465	330	214,731
ustar_5_9369_1024_5_3	977	**4,444**	549	2 132,913	7 984	21,228	**548**	348,549
ustar_5_9369_1024_5_5	1 630	**4,525**	564	1 846,000	7 633	20,345	**517**	354,646
ustar_100_9369_4096_100_0	6 384	44,580	2 364	19 153,019	9 328	**24,063**	**1 186**	5 447,590
ustar_100_9369_4096_100_3	5 378	44,360	2 529	17 565,642	9 222	**22,302**	**1 162**	4 828,836
ustar_100_9369_4096_100_5	3 500	42,885	2 875	18 355,866	9 368	**26,967**	**1 161**	4 414,953
ustar_50_9369_4096_50_0	4 335	**23,570**	2 161	11 647,938	9 318	26,739	**937**	2 830,459
ustar_50_9369_4096_50_3	4 064	23,477	1 886	10 950,661	9 208	**22,785**	**875**	2 617,345
ustar_50_9369_4096_50_5	3 369	**22,914**	2 249	11 812,362	9 366	23,698	**904**	2 306,804
ustar_5_9369_4096_5_0	479	**4,667**	543	1 061,132	6 462	13,363	**440**	247,107
ustar_5_9369_4096_5_3	865	**4,772**	635	1 531,394	8 842	25,485	**605**	342,821
ustar_5_9369_4096_5_5	1 043	**4,897**	695	1 720,500	7 396	18,707	**512**	294,662