Technical University of Berlin
Electrical Engineering and Computer Science
Institute of Software Engineering and Theoretical Computer Science
Algorithmics and Computational Complexity (AKT)

Towards Generalized Algorithmic Concepts for Temporal Betweenness Centrality

Maciej Rymar

Thesis submitted in fulfillment of the requirements for the degree
"Bachelor of Science" (B. Sc.) in the field of Computer Science

October 2020

Supervisor and first reviewer: Prof. Dr. Rolf Niedermeier
Second reviewer: Prof. Dr. Markus Brill
Co-Supervisors: Dr. Hendrik Molter and Dr. André Nichterlein

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Die selbstständige und eigenhändige Ausfertigung versichert an Eides statt

Berlin, den _____ _____

Datum Unterschrift

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

The independent and unaided completion of the thesis is affirmed by affidavit:

Berlin, _____ _____

Date Signature

Zusammenfassung

Im Feld der Netzwerkanalyse ist es oft erwünscht, die relative Wichtigkeit eines Knoten im Netzwerk zu quantifizieren. Ein Maß, das entwickelt wurde um dieses Problem anzugehen, ist *Betweenness Centrality*. Grob gesagt, ist dieses Maß die normalisierte Anzahl der kürzesten Pfade, die durch ein Knoten im Graphen gehen. Seit der ersten formellen Betrachtung im Jahr 1977, blieb es weiterhin relevant und viel studiert. Ein sich gerade herausbildendes Forschungsgebiet ist die Analyse der dynamischen Netzwerke. Ein Konzept, das für diesen Zweck genutzt wird, ist das der *temporalen Graphen* – Graphen, in denen Kanten mit der Zeit erscheinen und wieder verschwinden können. Im Gegensatz zu *dynamischen Graphen*, wird im Model der temporalen Graphen angenommen, dass man a priori ein totales Wissen über die Änderungen im Graphen hat. Daher liegt der Fokus nicht auf der Aufrechterhaltung von einigen relevanten Statistiken wenn der Graph sich ändert, sondern auf dem *Ausnutzen* der temporalen Dimension, um eine spezifische Einsicht zu gewinnen. Aktuell wird viel Mühe in die Analyse von temporalen Graphen investiert. In dieser Arbeit reflektieren wir frühere Ergebnisse über Betweenness Centrality, auch im temporalen Model, um dann diese Ergebnisse weiter auszubauen. Wir entwickeln ein Framework, das das Problem der Berechnung der Betweenness Centrality für mehrere Varianten des Maßes in einer einheitlichen Weise anzugehen erlaubt. Schlussendlich nutzen wir das Framework, um Algorithmen zu entwickeln, die einige noch nicht studierte Varianten von Betweenness Centrality berechnen.

Abstract

In network analysis, it is often desirable to quantify the relative importance of vertices that comprise the network. One measure devised to tackle this problem is *betweenness centrality*. Roughly speaking, it is the normalized number of shortest paths going through a vertex of a graph. First considered formally in 1977, it has stayed relevant and well-studied ever since. Furthermore, an emerging field of study is the analysis of dynamic networks. One concept used for this purpose is that of *temporal graphs*—graphs in which edges can appear and disappear over time. In contrast to *dynamic graphs*, in the temporal graph model we assume we have full knowledge about the changes in the graph a priori. Therefore, the focus does not lie on "maintaining" some relevant statistics as the graph changes, but rather on *exploiting* the temporal dimension to gain a unique insight into the structure of the network. Currently, a significant amount of effort is spent into the analysis of temporal graphs. In this work, we reflect on previous results about betweenness centrality—also in the temporal setting—and proceed to build up on them, creating a framework that allows us to tackle the task of computing betweenness centrality on a temporal graph for a variety of different variants of the measure, all in a unified manner. Using the framework, we devise algorithms for computing variants of betweenness centrality that have not been studied before.

Contents

Chapter 1

Introduction

In this chapter, we give a brief description and background on the main topics handled in the thesis.

1.1 Motivation

Graphs are possibly the most successful and most studied structures in all of computer science. Kickstarted in 1736 by none other than Leonhard Euler [Eul41], graph theory has remained relevant ever since. However, this thesis will tackle problems in the temporal graph setting, which is a natural extension of the concept of a graph. Nevertheless, since it may not be as familiar to the reader as traditional graph theory likely is, we shall also give a short introduction to the concept.

In this work we put ourselves in the field of network analysis [Bar+16; Sco88; WF94]. Networks have been used to model a wide variety of situations. Social networks can be used to model disease spread, while transportation networks are vital in urban planning.

However, we often see that in real life, networks and their structure evolve over time. For example, in a transportation network, connections (edges) between stations (vertices) appear and disappear as different trains, trams, or buses arrive and depart. Similarly, in a social network we see that people meet different people at different times. (For more cases in which the temporal aspect is important, see Holme and Saramäki [HS12].) Sometimes, such temporal dependencies can be either safely ignored or lend themselves to being modeled by a simple graph.

Other times, however, it may be desirable to model the dynamic aspect of a network more directly [Kos09; Mic16]. The concept that emerged for handling such cases is that of *temporal graphs* [HS12; Kos09; LVM18; Mic16]. Such graphs differ from the standard concept of simple graphs in that an explicit time component is introduced. Namely, edges within a graph are now allowed to appear and disappear as time passes. We can interpret an edge between two vertices u and v appearing at time t as "there is a connection running between u and v at time t" (in the case of transportation networks) or "u and v meet at time t" (in the case of social networks). Note that in this model, we assume that we have full knowledge of the temporal structure of the network a priori. Depending on the use case, this can be a reasonable assumption, e.g., there may be a fixed timetable (for a transportation network), or extrapolating historical data may yield

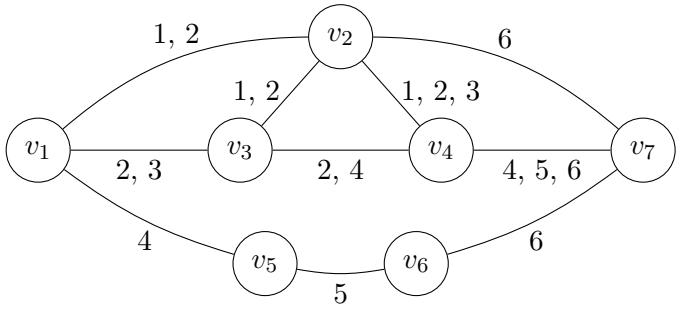


Figure 1.1: An example of a temporal graph. Nodes are connected with edges that appear at different times, e.g. v_1 and v_2 are connected at times 2 and 3, while v_5 and v_6 are connected only at time 5.

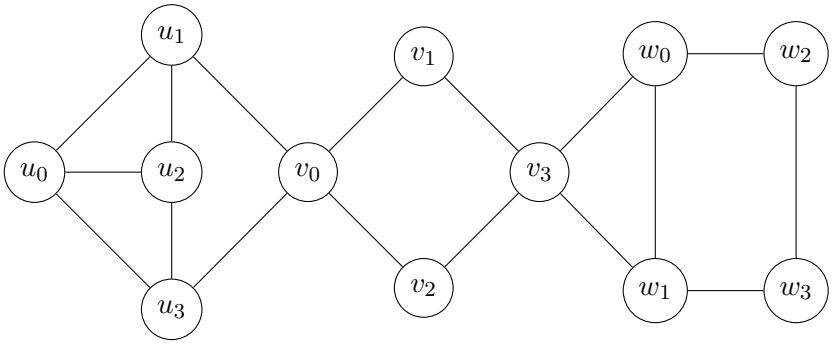
a good approximation. It is also worth noting that, as we shall see when discussing the concept more formally, temporal graphs can be interpreted as generalizations of simple (or *static*) graphs. For an example of a temporal graph, see Figure 1.1.

Moreover, whether a transportation-, social- or a flow network is considered, in many cases a deeper understanding of the underlying structure is desired. One important part of said structure is the connectivity of the graph and how it is affected by the individual vertices therein.

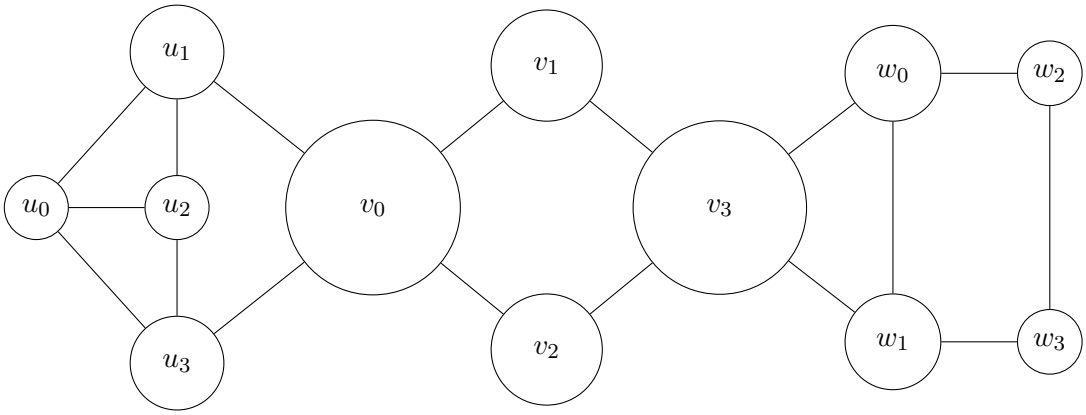
Many ways of approaching the subject of vertex importance have been devised, depending on one's particular needs. One important measure that has found widespread use in the static case is *betweenness centrality* [Fre77]. Roughly speaking, the betweenness centrality of a vertex measures how many shortest paths go through that vertex. Such vertices will often function as "hubs" of the network as a lot of traffic will go through them. For an example (static) graph and the betweenness centrality of the nodes within, see Figure 1.2. Due to its importance in the static case, many researchers have also extended the idea to the temporal case [ARFG17; Buß+20; Tan+10; Tsa+20; WM16].

Moreover, notice that measuring path length is not the only available way of determining the optimality of a path, which turns out to be even more true in the temporal case. Often, what measure of optimality is used depends on the context. Consider a situation in which authorities are trying to limit the spread of an infectious disease. They may then want to analyze the public transportation network. In this case, the measure of path optimality that would be the most relevant is the "duration" of the path—an average person will naturally take the fastest path from point A to point B. Hence, the authorities may decide to enact additional safety measures on the most central (and hence most crowded) stations, where the centrality is calculated based on fastest paths.

We may, however, be analyzing the situation from a different angle. For example, instead of a transportation network, consider analyzing the big picture via a social network in which the vertices correspond to people and edges correspond to a meeting of two people at a given point in time. A path in such a graph would then correspond to a "chain of spreading": person A meets person B and (possibly) infects them, then person B at a later point in time spreads the infection to person C, and so on. Hence, in such a network it would be important to identify people on a lot of "paths," who are



(a) Example (static) graph G.



(b) Same graph G with the size of each vertex scaled according to its betweenness centrality.

Figure 1.2: Example for betweenness centrality. The middle vertices (the v_i's) have the highest value of the measure as they connect the left- and the right-hand side of the graph (that is, they connect the u_i's with the w_i's). Note that v_1 and v_2 both have high betweenness centrality even though they both have a low degree and neither of them is a cut vertex.

both likely to get the disease *and* to spread it afterwards.

However, not every meeting may carry an equal risk. A person may show symptoms and decide against further contact with others or may recover from the disease, in both cases making it impossible for them to further spread the illness. Hence, we may only be interested in paths where the time between each pair of meetings in a "chain of spreading" is small enough as to make it probable that the disease can be transmitted. We would then be interested in determining the "central nodes" of our network, that is, people who are at the highest risk of spreading the disease.

We can therefore see that while the concept of using the number of "optimal" paths going through a vertex as a measure of centrality is the same in both cases, the ideas of optimality are very different. Hence, the algorithms devised for one measure of path optimality may not work for another. Case in point: our first example with the transportation network roughly corresponds to counting "fastest" paths. We shall successfully tackle this problem under a slight, but natural restriction—indeed, we provide a polynomial-time algorithm that solves it. On the other hand, for the second example,

the one where the "waiting time" has an upper limit, it has recently been shown that even checking *whether any such path exists* is already NP-Hard [Cas+20].

Therefore, our goal in this work is to provide a framework that allows us to tackle the task of computing the betweenness centrality measure for a variety of different path optimality concepts in a systematic way. Indeed, we exhibit a property of a set of paths that leads to a simplification of the problem, possibly allowing us to compute betweenness centrality more efficiently. We then proceed to use the framework to solve a few previously open questions.

1.2 Related work

All of the big concepts tackled in this work have been studied before. For general treatment on network analysis, see for example the works of Scott [Sco88], Wasserman and Faust [WF94], or Barabási et al. [Bar+16]. Although not called that at the time, early mentions of betweenness centrality trace back to 1971, in notes by Anthonisse [Ant71]. A more thorough treatment has later been given by Freeman [Fre77]. Since then, much effort has been put into the study of the measure from different angles.

A big hurdle to the widespread use of betweenness on real-life graph instances had been the difficulty of computing it in larger graphs. That situation has been changed somewhat with the publication of the seminal[1] work of Brandes [Bra01], where a new algorithm was presented. If we use n and m to respectively denote the number of vertices and edges in a graph, we can say that the new algorithm improved the time complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(nm)$ and $\mathcal{O}(nm + n^2 \log n)$ for unweighted and weighted graphs, respectively. Just as importantly, it reduced the space complexity of the then contemporary algorithms from $\mathcal{O}(n^2)$ to $\mathcal{O}(n + m)$, making the algorithm significantly more feasible to be run on larger sparse graphs. Moreover, a parallel algorithm for solving the problem has been devised by Jamour, Skiadopoulos, and Kalnis [JSK18].

White and Borgatti [WB94] study the concept in directed graphs—something relevant to temporal graphs since, as we shall see, even if all edges of a temporal graph are symmetric, temporal paths have an inherent direction dictated by increasing timestamps; a vertex u can be connected to v, but not vice versa.

The difficulty of computing betweenness centrality in spite of its high degree of usefulness in network analysis has motivated researchers in the area to turn their attention to approximate algorithms—with some success, see for example the works of Bader et al. [Bad+07], Bergamini and Meyerhenke [BM15], or Riondato and Upfal [RU16]. At the same time, it has been shown that there are subcubic reductions between the all-pairs shortest paths (APSP) and the betweenness centrality problems [AGW15]. This means that a truly subcubic algorithm[2] for computing the betweenness centrality of all vertices in a graph is unlikely to exist.

Another aspect of the topic tackled by research in the area is that of computing and maintaining the betweenness centrality in dynamic graphs, i.e., graphs, whose edges and/or vertices are added or removed. Such incremental algorithms can be found, for example, in the works of Pontecorvi and Ramachandran [PR15] or Kas, Carley, and

[1] At the time of writing this thesis, Brandes' paper had been cited more than 4000 times.

[2] An algorithm with a running time $\mathcal{O}(n^{3-\varepsilon})$, for some $\varepsilon > 0$.

Carley [KCC14]. Note that dynamic graphs, while a related concept, differ from temporal graphs—in a temporal graph, a path between two vertices can use edges that appear at different times, whereas in a dynamic graph all the edges in a path must exist at the same time.

Temporal graphs are also a well-established concept. For a thorough introduction to the subject, see Holme and Saramäki [HS12], Holme [Hol15], or Latapy, Viard, and Magnien [LVM18]. The canonical concepts of temporal path optimality, that we should also analyze in one of the chapters, have been presented by Wu et al. [Wu+14]. For another concept of path (optimality), see the works of Casteigts et al. [Cas+20] and Himmel et al. [Him+19]. The idea of extending the concept of betweenness centrality to temporal graphs has also appeared in multiple previous sources [ARFG17; Buß+20; Tan+10; Tsa+20; WM16].

Of particular interest is the work of Tsalouchidou et al. [Tsa+20], who consider an arbitrary linear combination of a path's length and duration as an optimality criterion and compute the temporal betweenness centrality with respect to such paths. This is very close to the problem we shall study in Section 4.3 and in fact, while formally showing it is outside the scope of this work, we conjecture that our results from that section also generalize those of Tsalouchidou et al. [Tsa+20], as will be discussed in the conclusions at the end of the thesis.

Finally, this thesis is most directly related to the paper of Buß et al. [Buß+20], who analyzed some combinations of optimality criteria, but using lexicographical tie-breaking instead of linear combinations—Section 4.3 generalizes the results from that paper to cover a wider range of possible combinations of criteria. Moreover, the framework we develop in Chapter 3 can be seen as a generalization of the approach used to derive the results of Buß et al. [Buß+20], that is, we work directly with the temporal graphs themselves. In contrast, the aforementioned paper of Tsalouchidou et al. [Tsa+20], as well as many others dealing with temporal graphs, deals with a "static expansion" of the temporal graph, i.e., a static graph that captures some relevant properties of the temporal graph. Such expansion can, however, lead to using more space than is necessary and it may also take additional time if one needs to first compute the static expansion from the original temporal graph. Hence, it is desirable to also devise approaches dealing with a temporal graph in its direct representation.

1.3 Our contribution

The concept of temporal betweenness centrality has already been studied by Buß et al. [Buß+20] (and others, see the previous section), where some variants have been proven computationally intractable, while a polynomial-time algorithm has been provided for several different ones. In this work, we notice a commonality between the results of Brandes [Bra01] and those of Buß et al. [Buß+20] and try to use it to generalize their insights, allowing us to solve the problem of computation of betweenness centrality on more of the possible variants of the measure.

We start by describing the computation of betweenness centrality in more general terms. We observe that, in essence, each concept of path optimality defines some subset $\mathcal{P}$ of all possible temporal paths in a temporal graph $\mathcal{G}$, e.g. the set of all shortest

paths between the vertices in the graph, or the set of all fastest paths in the graph. Then, to compute the betweenness centrality measure for some vertex v of our graph, we need to somehow count the paths in $\mathcal{P}$ that go through v and those that do not.

Hence, our approach is to first generalize the notion of path optimality to arbitrary path sets $\mathcal{P}$ and then provide a framework for simplifying the computation of the betweenness values for a class of path sets that have a particular property.

Finally, we use the framework to simplify some proofs of results that have already been shown previously, as well as solve open problems, including computing betweenness centrality for combinations of the canonical path optimality measures under lexicographic tie-breaking—some of the variants thereof have been analyzed by Buß et al. [Buß+20], but, to the best of our knowledge, many of the others have first been analyzed in this thesis.

Note that while our work focuses on temporal graphs, most of the notions and theorems described translate directly to static graphs, which can be seen as a special case of temporal graphs. For example, the seminal result of Brandes [Bra01] can be seen as a special case of the result derived in Section 4.3.

1.4 Organization of the work

In Chapter 1 we informally introduced the subject matter of this thesis. Chapter 2 provides definitions of (temporal) graphs and other basic concepts used throughout the work. The main contribution is split between the following two chapters. In Chapter 3, we generalize the concept of temporal betweenness centrality and then develop a framework for solving the problem of computing betweenness centrality in temporal graphs. Then, in Chapter 4, we proceed to use the framework on some practical examples. In particular, we show how the results of Buß et al. [Buß+20] can be derived within our framework. We then proceed to extend the results of Buß et al. [Buß+20] to further concepts of path optimality. Finally, in Chapter 5 we summarize our results and describe potential avenues for future research.

Chapter 2

Preliminaries

In this chapter, we define basic terms used throughout the work.

We write $\mathbb{N}$ for the set of non-negative integers, that is, $\mathbb{N} = \{0, 1, 2, \ldots\}$. Furthermore, let $n \in \mathbb{N}$. Then we use $[n]$ to denote the set $\{1, \ldots, n\}$. For example, we have $[0] = \emptyset$ and $[3] = \{1, 2, 3\}$.

We shall use the following Iversonian notation (as described by Graham, Knuth, and Patashnik [GKP94]): let $P(x)$ be any logical statement, whose truth may depend on the value of x. Then we denote by $[P(x)]_{\mathbb{1}}$ a quantity that is 1 precisely when $P(x)$ is true and 0 otherwise. For an example, we can write the n-th Harmonic number $H_n = \frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n}$ as[1]

$$H_n = \sum_k \frac{1}{k} [1 \leq k \leq n]_{\mathbb{1}} ,$$

or similarly, if we want to describe the sum of reciprocals of primes up to n, then we would write

$$\sum_p \frac{1}{p} [p \text{ is prime}]_{\mathbb{1}} [p \leq n]_{\mathbb{1}} .$$

Furthermore, in this work, we will also briefly consider binary relations. One property of those will be important to us, namely acyclicity (see, for example, Alcantud et al. [Alc+18]).

Definition 2.1 (Acyclicity)**.** Let R be a binary relation. We call R *acyclic* if its transitive closure is irreflexive.

2.1 Static and temporal graphs

Graph theory. For basic graph theory we shall mostly follow the notation of Diestel [Die17]. Let $G = (V, E)$ denote an undirected graph, where V denotes the set of vertices and $E \subseteq \binom{V}{2} = \{\{v, w\} \mid v, w \in V, v \neq w\}$ denotes the set of edges. We shall generally use n to refer to the number of vertices in the graph and m to refer to the number of edges. Formally, we will have $n \coloneqq |V|$ and $m \coloneqq |E|$.

[1] Note that by convention, when $[\cdot]_{\mathbb{1}}$ is zero, it "overpowers" undefined terms, so for example, in the equation below we have for the term with $k = 0$ that $\frac{1}{0} \cdot [1 \leq 0 \leq n]_{\mathbb{1}} = 0$.

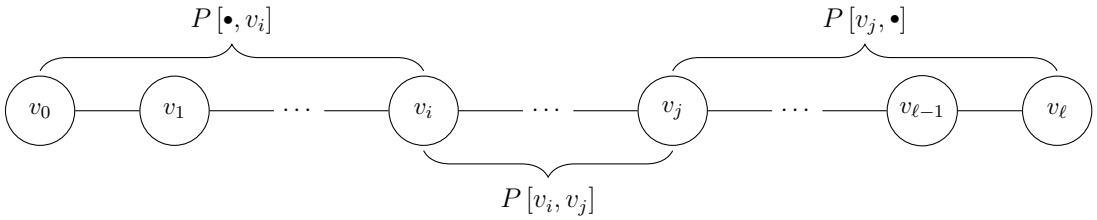


Figure 2.1: Subpaths of the path $P = (v_0, \ldots, v_\ell)$.

We denote by $N_G(v)$ the neighborhood of the vertex v, that is, the set $N_G(v) \coloneqq \{w \mid \{v, w\} \in E\}$. If the graph G is clear from context, then we can simply write $N(v)$.

Let $P = (v_0, v_1, \ldots, v_\ell)$ be a path. For $0 \leq i \leq j \leq \ell$, we shall write:

$$P[v_i, v_j] \coloneqq (v_i, v_{i+1}, \ldots, v_j)$$
$$P[\bullet, v_i] \coloneqq (v_0, v_1, \ldots, v_i)$$
$$P[v_j, \bullet] \coloneqq (v_j, v_{j+1}, \ldots, v_\ell)$$

We shall denote the concatenation of two paths with the $\oplus$ operator: the concatenation of two paths $P[x, y]$ and the path $Q[y, z]$ for appropriate $x, y, z \in V$ will be written as $P[x, y] \oplus Q[y, z]$. Note that, in general, $P[x, y] \oplus Q[y, z]$ does not have to be a path since it may be that y is not the only vertex that P and Q share.

Temporal graphs. We now present the notion of temporal graphs, that is, graphs whose connectivity evolves over time. For a more comprehensive introduction to the topic, see Holme and Saramäki [HS12] or Latapy, Viard, and Magnien [LVM18].

Let $\mathcal{G} = (V, \mathcal{E}, T)$ denote an *(undirected) temporal graph*, where V denotes the set of vertices, $T \in \mathbb{N}$ denotes the *timespan* of the graph, and $\mathcal{E} \subseteq \binom{V}{2} \times [T] = \{(\{v, w\}, t) \mid v, w \in V, v \neq w, t \in [T]\}$ the set of *temporal edges*. For a $v \in V$ and $t \in [T]$ we shall call (v, t) a *vertex appearance*. For a temporal edge $(\{v, w\}, t)$ or a vertex appearance (v, t) we shall call t the *timestamp* of the edge (vertex appearance). Whenever we want to stress the difference between temporal graphs and the standard, non-temporal graphs, we shall call the latter *static graphs*.

Let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph. We shall call $G_\downarrow = (V, E)$, with $E \coloneqq \{\{v, w\} \mid \exists t \in [T] : (\{v, w\}, t) \in \mathcal{E}\}$, the *underlying graph* of $\mathcal{G}$. Intuitively, $G_\downarrow$ corresponds to the graph $\mathcal{G}$ with the timestamp information removed.

For temporal graphs, we define several notions of "neighborhood," one for vertices and two for vertex appearances. The neighborhood of a vertex $v \in V$, denoted $N_\mathcal{G}(v)$ is directly related to the static case: $N_\mathcal{G}(v) \coloneqq N_{G_\downarrow}(v)$. For a vertex appearance $(v, t) \in V \times [T]$, the temporal neighborhood is the set of directly reachable vertex appearances, i.e., $N_\mathcal{G}(v, t) = \{(w, t') \mid (\{v, w\}, t') \in \mathcal{E}, t' \geq t\}$. Sometimes we may, however, be interested in the *strict* temporal neighborhood, which is defined analogously: $N^>_\mathcal{G}(v, t) = \{(w, t') \mid (\{v, w\}, t') \in \mathcal{E}, t' > t\}$. If the temporal graph $\mathcal{G}$ is clear from the context, then we can omit the subscript and write $N(v), N(v, t)$, or $N^>(v, t)$.

Similarly as in the static case, we introduce generic variables that represent the size of the graph. We will generally use n to be the number of vertices in our graph. For

the edges, in the temporal case we have two quantities to consider. We use M to refer to the number of temporal edges in the graph, while we let m refer to the number of edges of the underlying graph. Formally, we have $n \coloneqq |V|$ for the number of vertices and $M \coloneqq |\mathcal{E}|$ and $m \coloneqq |E|$ for the variables referring to the edges.

Walks and paths. Let $\mathcal{G}$ be a temporal graph. We now define walks and paths on $\mathcal{G}$.

Definition 2.2 (Temporal Walk)**.** A *temporal walk* W is an alternating sequence of vertices and timestamps $W = (v_0, t_1, v_1, \ldots, t_{\ell-1}, v_{\ell-1}, t_\ell, v_\ell)$, such that for any $0 \leq i < \ell$, we have $(\{v_i, v_{i+1}\}, t_{i+1}) \in \mathcal{E}$ and for any $i \leq j \leq \ell$ we also have $t_i \leq t_j$. A temporal walk is called *strict* if additionally the timestamps are strictly monotonically increasing, i.e., we have $t_i < t_j$ for all $1 \leq i < j \leq \ell$, and *non-strict* otherwise. A walk has multiple parameters describing it. We call ℓ the *length* of the walk. We call t_1 and t_ℓ the *departure-* and *arrival times* of the walk, respectively. The *duration* of a walk is the difference between its arrival and departure times, i.e. $t_\ell - t_1$. In the context of walks, we also call the temporal edge $(\{v_i, v_{i+1}\}, t_{i+1})$ a *transition* from v_i to v_{i+1} at time t_{i+1} and denote it by $v_i \overset{t_{i+1}}{\to} v_{i+1}$.

Note that we allow walks of length zero between a vertex and itself, however such walks do not have a well-defined departure- and arrival time (and hence also no well-defined duration).

If $W = (v_0, t_1, v_1, \ldots, t_{\ell-1}, v_{\ell-1}, t_\ell, v_\ell)$ is a walk, then we shall also write $v_0 \overset{t_1}{\to} v_1 \overset{t_2}{\to} \cdots \overset{t_{\ell-1}}{\to} v_{\ell-1} \overset{t_\ell}{\to} v_\ell$ to denote it and its transitions. We also write $(v_i \overset{t_{i+1}}{\to} v_{i+1}) \in W$ to express that the walk "uses" the transition $v_i \overset{t_{i+1}}{\to} v_{i+1}$, i.e., that W has the subsequence (v_i, t_{i+1}, v_{i+1}). Alternatively, for two vertex appearances (u, t_u) and (v, t_v) on the walk, we can also write $(u, t_u) \overset{t_u}{\to} (v, t_v) \in W$.

Moreover, a walk is said to *go through* the vertex appearance (v, t) if there exists some vertex u such that $(u \overset{t}{\to} v) \in W$. We can then also write $(v, t) \in W$.

Next, we define temporal paths as special cases of temporal walks.

Definition 2.3 (Temporal Path)**.** A *temporal path* P is a walk in which no vertex appears more than once in the corresponding alternating sequence. As with walks, we say that a path is *strict* if the corresponding walk is strict, and *non-strict* otherwise. Analogously, we define the length, departure time, arrival time and duration of the path as the respective value of the underlying walk.

Similarly to the static case, for a temporal path $P = v_0 \overset{t_1}{\to} v_1 \overset{t_2}{\to} \ldots \overset{t_{\ell-1}}{\to} v_{\ell-1} \overset{t_\ell}{\to} v_\ell$ and $0 \leq i \leq j \leq \ell$, we shall write

$$
\begin{aligned}
P[v_i, v_j] &\coloneqq v_i \overset{t_{i+1}}{\to} v_{i+1} \overset{t_{i+2}}{\to} \ldots \overset{t_j}{\to} v_j \\
P[\bullet, v_i] &\coloneqq v_0 \overset{t_1}{\to} v_1 \overset{t_2}{\to} \ldots \overset{t_i}{\to} v_i \\
P[v_j, \bullet] &\coloneqq v_j \overset{t_{j+1}}{\to} v_{j+1} \overset{t_{j+2}}{\to} \ldots \overset{t_\ell}{\to} v_\ell
\end{aligned}
$$

and write $P[x, y] \oplus Q[y, z]$ when concatenating the paths $P[x, y]$ and $Q[y, z]$ for appropriate $x, y, z \in V$ such that the concatenation forms a valid walk, i.e., P arrives at y

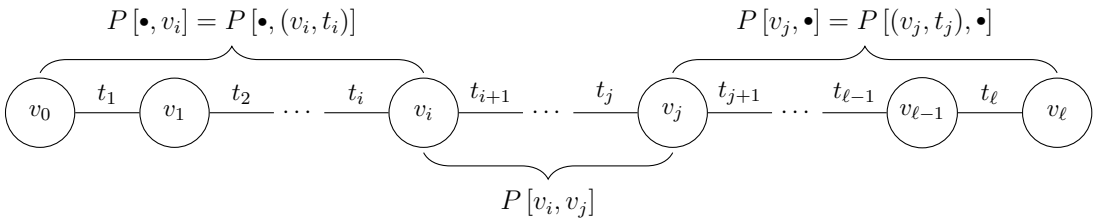


Figure 2.2: Subpaths of the temporal path P.

no later than Q leaves y. (Or, in the strict case, P arrives at y before Q leaves it.) If we want to be more explicit about the specific vertex appearance that is a part of the temporal path, then we can write $P[\bullet, (v, t)]$ instead of $P[\bullet, v]$, etc. See Figure 2.2 for reference.

Optimal paths in temporal graphs. Unlike in static graphs, in temporal graphs there are several different natural ways of defining the optimality of a path between two vertices $s, z \in V$.

- The first possibility is to consider the path length. This measure is the same as in the static case, namely it is concerned with the number of edges. A path is *shortest* if it has the least amount of edges among all s-z-paths.
- We may also be interested in paths which arrive as early as possible to our destination. Such paths are called *foremost* paths, that is, those paths, whose arrival time is the smallest among all s-z-paths.
- Finally, we may want to consider the paths of least duration. We call those paths, i.e., paths with the smallest duration among all s-z-paths, *fastest* paths.

Furthermore, we shall also speak about optimal paths with respect to a particular *vertex appearance* (v, t). Every path arriving at v must arrive at precisely one time t[2]. Hence, we can "partition" the set of paths arriving at v according to their arrival time. Such partitioning will prove useful in showing some results in Chapter 3. We define those optimal paths to an appearance analogously to optimal paths to vertices, the difference being that we then only consider paths which arrive at *exactly* time t. For example, a path is a shortest s-(v, t)-path if it is shortest among all s-v-paths that end *exactly* at the appearance (v, t).

As an example, consider Figure 2.3. There exists exactly one optimal s-z-path for each of the three optimality criteria. The top path is shortest, because it consists of only two edges. However it is not foremost because it arrives later than the middle path and is not fastest because it has a larger duration than the bottom path. Similarly, the middle path is foremost as it is the only path that arrives to z at time 5. Finally, the bottom path is the only fastest path, with a duration of 2. Also note that while the middle path is not a shortest s-z-path, it *is* a shortest path to $(z, 5)$, as no shorter path that arrives to z at exactly time 5 exists (indeed, there exists *no* other path that arrives to z at exactly time 5).

[2]Ignoring the path from v to itself that does not have well-defined arrival time.

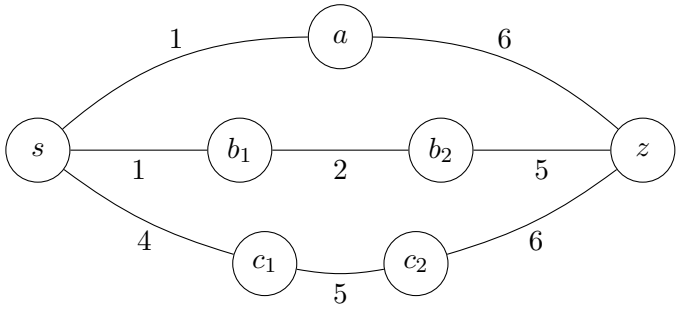


Figure 2.3: An example for different optimality concepts in a temporal graph. From top to bottom: shortest, foremost and fastest.

2.2 Betweenness centrality in (temporal) graphs

For the sake of completeness we shall formally introduce the basic idea of betweenness centrality. In this section we shall provide the typical definitions as seen in the literature. In the later chapters, we will, however, find it convenient to write it out in a different, but equivalent way.

As was already described in the introduction, betweenness centrality is the normalized sum of the number of shortest paths going through the vertex.

Definition 2.4. Let $G = (V, E)$ be an arbitrary graph and let $s, v, z \in V$ be arbitrary vertices. We then denote by

- σ_{sz} the number of shortest s-z-paths;
- $\sigma_{sz}(v)$ the number of shortest s-z-paths that go through v.

With that in mind we can define betweenness centrality.

Definition 2.5. Let $G = (V, E)$ be an arbitrary connected graph. Then, for every $v \in V$ we define

$$C_B(v) = \sum_{s \neq v \neq z} \frac{\sigma_{sz}(v)}{\sigma_{sz}}$$

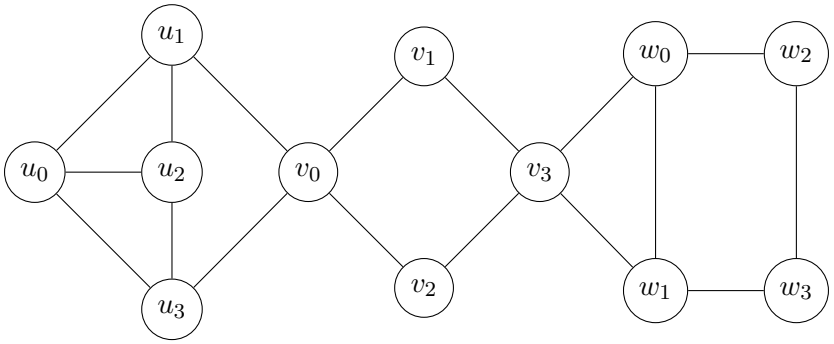
as the *betweenness centrality* of v.

For an example, consider Figure 2.4. The graph has already appeared in Chapter 1 and has only been reproduced here for convenience.

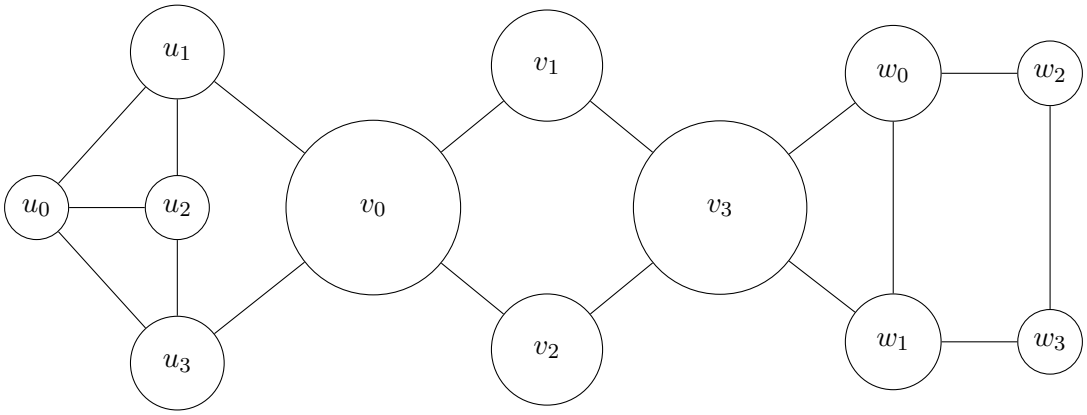
In the temporal case, we define the betweenness analogously. We let $\sigma^\star_{sz}$ be the number of optimal temporal s-z-paths (using some notion of optimality, e.g. shortest or fastest) and $\sigma^\star_{sz}(v)$ be the number of optimal temporal s-z-paths going through v. We then have

$$C_B(v) = \sum_{\substack{s \neq v \neq z \\ \sigma_{sz} > 0}} \frac{\sigma^\star_{sz}(v)}{\sigma^\star_{sz}}.$$

Note that, unlike in the static case, we may have $\sigma^\star_{sz} \neq \sigma^\star_{zs}$: see Figure 2.3 for an example. Even though all of our temporal edges are symmetric, there exist three different s-z-paths, but no z-s-path.



(a) Example (static) graph G.



(b) Same graph G with the size of each vertex scaled according to its betweenness centrality.

Figure 2.4: Example for betweenness centrality. The middle vertices (the v_i's) have the highest value of the measure as they connect the left- and the right-hand side of the graph (that is, they connect the u_i's with the w_i's). Note that v_1 and v_2 both have high betweenness centrality even though they both have a low degree and neither of them is a cut vertex.

2.3 Counting complexity: #P-hardness

As it is mentioned a few times throughout the thesis, we would like to give a brief and mostly informal presentation of the concept of counting problems and the complexity class #P, first introduced by Valiant [Val79b].

Many decision problems are of the form "does there exist a solution to the problem P?". Counting problems naturally extend the idea by asking, "how many solutions do there exist to the problem P?". For example, a decision problem may be, "are the vertices s and z in the graph connected?", that is, "does there exist an s-z-path?", while the corresponding counting problem could be, "how many s-z-paths are there?". Similarly, the SAT problem asks whether a satisfying variable assignment to a Boolean formula exists, while its counting version asks *how many* such assignments there are.

Just like for decision problems, different complexity classes of counting problems have been considered. We shall, however, limit ourselves to the complexity class #P. Formally speaking, if we choose $\{0, 1\}$ as our input alphabet, then $\#P$ is a subset of

functions of the type $\{0,1\}^* \to \mathbb{N}$.

Definition 2.6. (Arora and Barak [AB09]) #P is the set of all functions f of the type $f : \{0,1\}^* \to \mathbb{N}$ such that there exists a polynomial-time non-deterministic Turing machine M such that, for every $x \in \{0,1\}^*$, we have that $f(x)$ is equal to the number of paths from the initial configuration of M with input x to an accepting configuration.

Intuitively speaking, this class is roughly the counting problem equivalent of the well-known decision problem class of NP. Hence, for many problems in #P, polynomial-time algorithms are unlikely to exist. Just like in the case of decision problems, some problems can be considered "at least as hard as all other problems in #P," that is, all other problems in #P can be reduced to those (using an appropriate concept of reduction). Such problems are called #P-hard.

Naturally, the counting version of the problem is at least as hard as the decision version: we can always check whether the number of solutions to a problem is positive. In some cases, if the decision problem is already computationally intractable, that fact may be of little importance. However, sometimes the counting version may be significantly harder. For example, while it is easy to determine whether two vertices in a graph are connected, counting the number of paths between two vertices is #P-hard [Val79a].

Chapter 3

A framework for betweenness computation

In this chapter, we shall tackle the main problem of this work, namely that of computing temporal betweenness centrality in a general path optimality setting. In Section 3.1, we define notions necessary for developing our framework. We then present our main result about computation of the generalized betweenness centrality in Section 3.2. This generalization will allow us to approach the task of computing the betweenness centrality with respect to different possible optimality concepts in a unified manner, as we shall see in Chapter 4.

See Table 3.1 (on page 24) for an overview of the notation used throughout the chapter.

3.1 Generalized path optimality

We start by defining a general notion of betweenness centrality with respect to some arbitrary set of temporal paths $\mathcal{P}$. To this end, we shall generalize the concepts as described in the static case by Brandes [Bra01], and in the temporal case by Buß et al. [Buß+20].

First, it will be convenient to expand the set of possible timestamps in order to let the starting point of a path have a well-defined appearance associated with it.

Definition 3.1 (Extended timespan set)**.** Let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph. We write $\mathcal{T} \coloneqq \{0\} \cup [T]$ to denote the *extended timespan set*.

Since no temporal edge can have a timestamp 0 associated with it, it can serve as a sentinel value that we will associate with the beginning of a temporal path. That is, if we have a path $P = s \xrightarrow{t_1} v_1 \xrightarrow{t_2} \cdots \xrightarrow{t_\ell} z$, then we will associate the appearance $(s, 0)$ with s. Note, however, that for the purpose of computation of optimality criteria, the definitions from Chapter 2 still apply: the duration of P is still equal to $t_\ell - t_1$.

We can now define variables counting the number of paths between vertices.

Definition 3.2 (Path counting)**.** Let $\mathcal{G}$ be a temporal graph and $\mathcal{P}$ be a subset of its temporal paths. Let $s, v, z \in V$ and $t \in \mathcal{T}$. We write:

Table 3.1: A short overview of all the symbols defined and used throughout the chapter. Note that while omitted in the descriptions for the purpose of avoiding excessive redundancy, all the symbols refer only to paths in $\mathcal{P}$, as signified by their superscripts.

Variable	Meaning
$\sigma_{sz}^{\mathcal{P}}$	number of s-z-paths
$\sigma_{sz}^{\mathcal{P}}(v)$	number of s-z-paths going through v
$\sigma_{sz}^{\mathcal{P}}(v,t)$	number of s-z-paths going through (v,t)
$\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}}$	number of unique prefixes to (v,t) on s-z-paths
$\delta_{sz}^{\mathcal{P}}(v)$	pair dependency of s and z on v
$\delta_{sz}^{\mathcal{P}}(v,t)$	temporal pair dependency of s and z on (v,t)
$\delta_{sv}^{\mathcal{P}}(v,t)$	appearance dependency of s on (v,t)
$\delta_{s\bullet}^{\mathcal{P}}(v)$	cumulative dependency of s on v
$\delta_{s\bullet}^{\mathcal{P}}(v,t)$	temporal cumulative dependency of s on (v,t)
$\delta_{sz}^{\mathcal{P}}(v,t,(\{v,w\},t'))$	edge dependency
$C_B^{\mathcal{P}}(v)$	betweenness centrality of v
$\hat{C}_B^{\mathcal{P}}(v)$	total betweenness centrality of v
$\mathrm{Pre}_s^{\mathcal{P}}(w,t')$	set of predecessors of (w,t') on paths starting in s
$\mathrm{Succ}_s^{\mathcal{P}}(v,t)$	set of successors of (v,t) on paths starting in s

- $\sigma_{sz}^{\mathcal{P}}$ for the number of s-z-paths in $\mathcal{P}$ that start in s and end in z. We have that $\sigma_{ss}^{\mathcal{P}} = [\mathcal{P} \text{ contains the trivial zero-length path from } s \text{ to } s]_{\mathbb{1}}$;

- $\sigma_{sz}^{\mathcal{P}}(v)$ for the number of s-z-paths in $\mathcal{P}$ that go through the vertex v. Again, for the edge cases we have $\sigma_{sz}^{\mathcal{P}}(z) = \sigma_{sz}^{\mathcal{P}}(s) = \sigma_{sz}^{\mathcal{P}}$ and $\sigma_{ss}^{\mathcal{P}}(s) = \sigma_{ss}^{\mathcal{P}}$;

- $\sigma_{sz}^{\mathcal{P}}(v,t)$ for the number of s-z-paths in $\mathcal{P}$ that go through the vertex appearance (v,t). Note that since the first vertex in a path has a special appearance $(s,0)$, we have $\sigma_{sz}^{\mathcal{P}}(s,0) = \sigma_{sz}^{\mathcal{P}}(s) = \sigma_{sz}^{\mathcal{P}}$ and $\sigma_{sz}^{\mathcal{P}}(s,t') = 0$ for all $t' \in [T]$.

We can use these path counts to now define the notions of dependency of vertices on other vertices.

Definition 3.3 (Pair dependency, cumulative dependency)**.** Let $\mathcal{G}$ be a temporal graph

and $\mathcal{P}$ a subset of its temporal paths. We define

$$\delta_{sz}^{\mathcal{P}}(v) \coloneqq \begin{cases} 0, & \text{if } \sigma_{sz}^{\mathcal{P}} = 0, \\ \frac{\sigma_{sz}^{\mathcal{P}}(v)}{\sigma_{sz}^{\mathcal{P}}}, & \text{otherwise;} \end{cases}$$
$$\delta_{s\bullet}^{\mathcal{P}}(v) \coloneqq \sum_{z \in V} \delta_{sz}^{\mathcal{P}}(v)$$

as the *pair dependency* of s and z on v and the *cumulative dependency* of s on v, respectively.

In other words, $\delta_{sz}^{\mathcal{P}}(v)$ is the fraction of s-z paths that go through v. Intuitively, the higher this fraction is, the more important v is to the connectivity of s and z in the graph. Furthermore, $\delta_{s\bullet}^{\mathcal{P}}(v)$ is the cumulative dependency of s on v for all possible destinations. The natural extension of the idea would be to then ask how do *all* other vertices depend on v for their connectivity? The graph measure that catches such dependency on v in the whole graph is, of course, the betweenness centrality.

Definition 3.4 (Betweenness centrality)**.** Let $\mathcal{G}$ be a temporal graph and $\mathcal{P}$ be a subset of its temporal paths. Then, for any vertex $v \in V$, let

$$C_B^{\mathcal{P}}(v) \coloneqq \sum_{s \neq v \neq z} \delta_{sz}^{\mathcal{P}}(v)$$

be the *betweenness centrality* of v (with respect to $\mathcal{P}$). If the set of paths $\mathcal{P}$ in question is clear from the context, then we shall simply write $C_B(v)$.

We note that there are two main differences between the definition above and the one in Chapter 2: first, the definition above uses the path counts $\sigma_{sz}^{\mathcal{P}}$ indirectly, via the pair dependencies $\delta_{sz}^{\mathcal{P}}(v)$. This additional layer of abstraction allows us to define betweenness with a simple formula, avoiding the need of adding more conditions to the sum in Definition 3.4 or restricting our attention to (strongly) connected graphs. The value of the measure itself, however, remains unchanged. The second, and more important, difference is that we now allow the path set to be arbitrary and not just one of few possible choices.

With betweenness centrality defined, we can now formally state the main problem studied in this work.

$\mathcal{P}$-Betweenness-Centrality
Input: A temporal graph $\mathcal{G}$.
Task: Compute $C_B^{\mathcal{P}}(v)$ for every $v \in V$.

The generalization above is based on the standard concept of betweenness centrality. However, we shall find that slightly relaxing the condition in the sum of Definition 3.4 yields a highly related notion of "total" betweenness that includes all vertex pairs in the definition that is more convenient to use in our case.

Definition 3.5 (Total betweenness centrality)**.** Let $\mathcal{G}$ be a temporal graph and $\mathcal{P}$ a subset of its temporal paths. Then, for any vertex $v \in V$, we define

$$\hat{C}_B^{\mathcal{P}}(v) \coloneqq \sum_{s,z \in V} \delta_{sz}^{\mathcal{P}}(v)$$

to be the *total betweenness centrality* of v (with respect to $\mathcal{P}$). Again, if the set of paths $\mathcal{P}$ in question is clear from the context, then we shall simply write $\hat{C}_B(v)$.

The main reason behind using total betweenness centrality instead of the standard betweenness is that it simplifies some of our proofs as it works well with our definition of cumulative dependency:

Observation 3.6. *For any vertex $v \in \mathcal{G}$ we have*

$$\hat{C}_B^{\mathcal{P}}(v) = \sum_{s \in V} \delta_{s\bullet}^{\mathcal{P}}(v).$$

As mentioned before, $\hat{C}_B(v)$ and C_B are highly related. The following lemma formally expresses this relationship. Both the statement and the proof have already essentially appeared in the work of Buß et al. [Buß+20], but we reproduce it for the purpose of completeness and because the notation in this thesis is slightly different.

Lemma 3.7 (Lemma 2.8 in Buß et al. [Buß+20])**.** *For any vertex $v \in V$ it holds that*

$$C_B(v) = \hat{C}_B(v) - \sum_{w \in V} \left(\left[\sigma_{vw}^{\mathcal{P}} > 0\right]_{\mathbb{1}} + \left[\sigma_{wv}^{\mathcal{P}} > 0\right]_{\mathbb{1}} \right) + \left[\sigma_{vv}^{\mathcal{P}} > 0\right]_{\mathbb{1}}. \tag{3.1}$$

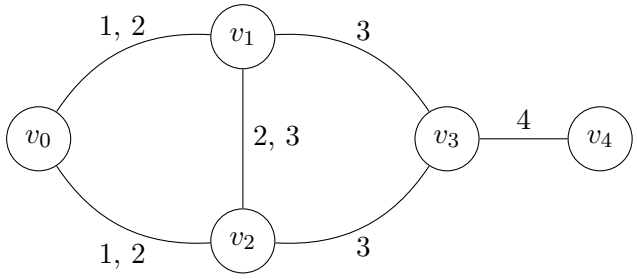
Proof. We can simply expand the sum to get the additional summands not present in the definition of C_B:

$$\begin{aligned}
\hat{C}_B(v) &= \sum_{s,z \in V} \delta_{sz}^{\mathcal{P}}(v) \\
&= \sum_{s \neq v \neq z} \delta_{sz}^{\mathcal{P}}(v) + \sum_{z \in V} \delta_{vz}^{\mathcal{P}}(v) + \sum_{s \in V} \delta_{sv}^{\mathcal{P}}(v) - \delta_{vv}^{\mathcal{P}}(v) \\
&= C_B(v) + \sum_{w \in V} \left(\delta_{vw}^{\mathcal{P}}(v) + \delta_{wv}^{\mathcal{P}}(v) \right) - \left[\sigma_{vv}^{\mathcal{P}} > 0\right]_{\mathbb{1}} \\
&= C_B(v) + \sum_{w \in V} \left(\left[\sigma_{vw}^{\mathcal{P}} > 0\right]_{\mathbb{1}} + \left[\sigma_{wv}^{\mathcal{P}} > 0\right]_{\mathbb{1}} \right) - \left[\sigma_{vv}^{\mathcal{P}} > 0\right]_{\mathbb{1}},
\end{aligned}$$

from which the result immediately follows. □

In other words, to relate the two quantities we only need to know for each pair of vertices whether there is any path in $\mathcal{P}$ connecting them. As an example, consider the graph in Figure 3.1a. In Figure 3.1b we list the appropriate values of betweenness and total betweenness with respect to the set of shortest (non-strict) paths. For example, the vertex v_0 is connected to every other vertex (including itself), but only v_1 and v_2 are connected to it. We therefore have $\sum_{w \in V} \left(\left[\sigma_{v_0 w}^{\mathcal{P}} > 0\right]_{\mathbb{1}} + \left[\sigma_{w v_0}^{\mathcal{P}} > 0\right]_{\mathbb{1}} \right) - 1 = 7$ and so $\hat{C}_B(v_0) = C_B(v_0) + 7 = 7$.

Furthermore, to work with temporal graphs more easily, we shall make a straightforward generalization of Definition 3.3 to vertex appearances.



(a) Temporal graph $\mathcal{G}$.

Vertex	C_B	$\hat{C}_B$
v_0	0	7
v_1	1	9
v_2	1	9
v_3	3	11
v_4	0	6

(b) The C_B and $\hat{C}_B$ values for $\mathcal{G}$.

Figure 3.1: Betweenness centrality vs. total betweenness centrality for the set of shortest non-strict paths.

Definition 3.8 (Temporal pair dependency, temporal cumulative dependency)**.** Let $\mathcal{G}$ be a temporal graph and $\mathcal{P}$ be a subset of its paths. Let

$$\delta^{\mathcal{P}}_{sz}(v,t) \coloneqq \begin{cases} 0, & \text{if } \sigma^{\mathcal{P}}_{sz} = 0 \\ \frac{\sigma^{\mathcal{P}}_{sz}(v,t)}{\sigma^{\mathcal{P}}_{sz}}, & \text{otherwise} \end{cases}$$
$$\delta^{\mathcal{P}}_{s\bullet}(v,t) \coloneqq \sum_{z \in V} \delta^{\mathcal{P}}_{sz}(v,t)$$

be the *temporal pair dependency* of s and z on (v,t) and the *temporal cumulative dependency* of s on (v,t), respectively. Additionally, we shall also call the special case of $\delta^{\mathcal{P}}_{sv}(v,t)$ the *appearance dependency* of s on (v,t).

Note that the source vertex of any temporal path has the special appearance $(s,0)$ associated with it, so for it the only timestamp $t \in \mathcal{T}$ where the values of $\delta^{\mathcal{P}}_{sz}(s,t)$ and $\delta^{\mathcal{P}}_{s\bullet}(s,t)$ can be non-zero is $t = 0$ (cf. Definition 3.2).

Since every path has a unique arrival time, there is a simple relation between dependencies on vertices and dependencies on vertex appearances.

Observation 3.9. *For any vertex $v \in V$ we have that*

$$\delta^{\mathcal{P}}_{sz}(v) = \sum_{t \in \mathcal{T}} \delta^{\mathcal{P}}_{sz}(v,t), \text{ and}$$
$$\delta^{\mathcal{P}}_{s\bullet}(v) = \sum_{t \in \mathcal{T}} \delta^{\mathcal{P}}_{s\bullet}(v,t).$$

In our recursive formula for the computation of cumulative dependencies that we shall derive later in this chapter, we will need to consider the successors (or dually, the predecessors) of each vertex appearance on paths in $\mathcal{P}$.

Definition 3.10 (Direct predecessor set, direct successor set)**.** Fix a source vertex $s \in V$. Let $\mathcal{P}_s \subseteq \mathcal{P}$ be the set of paths in $\mathcal{P}$ that start in s. Now let $(w,t') \in V \times \mathcal{T}$ be any vertex appearance. Then $\mathrm{Pre}^{\mathcal{P}}_s(w,t')$ is the set of all direct predecessors of (w,t') on paths in $\mathcal{P}_s$. Formally,

$$\mathrm{Pre}^{\mathcal{P}}_s(w,t') \coloneqq \left\{ (v,t) \in V \times \mathcal{T} \mid \exists P \in \mathcal{P}_s \colon (v,t) \xrightarrow{t'} (w,t') \in P \right\}.$$

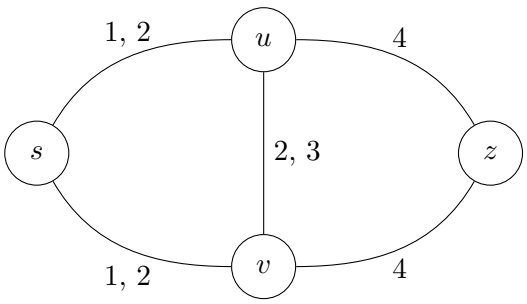


Figure 3.2: An example of a non-acyclic path set: $(u, 2)$ is a predecessor of $(v, 2)$ on the $s \xrightarrow{2} u \xrightarrow{2} v \xrightarrow{4} z$ foremost s-z-path, but $(v, 2)$ is also a predecessor of $(u, 2)$—on the $s \xrightarrow{2} v \xrightarrow{2} u \xrightarrow{4} z$ foremost s-z-path.

Analogously, we define the set $\mathrm{Succ}_s^{\mathcal{P}}(v, t)$ of successors of a vertex appearance (v, t) as the "inverse" of the predecessor relation. Formally,

$$\mathrm{Succ}_s^{\mathcal{P}}(v, t) \coloneqq \left\{(w, t') \mid (v, t) \in \mathrm{Pre}_s^{\mathcal{P}}(w, t')\right\}.$$

Note that the predecessor sets naturally induce a relation R on vertex appearances: we have $(v, t)R(w, t')$ if and only if $(v, t) \in \mathrm{Pre}_s^{\mathcal{P}}(w, t')$. One property of such induced relations that proves crucial in devising our algorithms for the $\mathcal{P}$-Betweenness-Centrality problem is that of acyclicity. Note that this induced relation is not concerned with vertices, but rather only with *vertex appearances*. This makes the property of acyclicity quite weak, and hence easy to fulfill. For example, if we only consider strict paths, then all possible path sets will induce acyclic relations: we can then only have that (v, t) is a predecessor of (w, t') if we have $t < t'$. However, in that case we clearly cannot have a path on which (w, t') is a predecessor of (v, t). This observation, although obvious, has some important consequences: for one of the examples that will be described in detail in Chapter 4, namely that of prefix-foremost paths, this is the crucial property differentiating the strict and non-strict case, allowing us to solve $\mathcal{P}$-Betweenness-Centrality in $\mathcal{O}(nM \log M + n^2)$ time in the strict case, while the non-strict variant is #P-hard.

An example of an acyclic relation for non-strict paths would be the relation induced by the set of all shortest paths starting at some source vertex s of a graph. On the other hand, if we take $\mathcal{P}$ to be the set of foremost paths starting at some source vertex s, then the relation will not necessarily be acyclic, since two appearances can be predecessors of each other on such paths. Such situation is pictured in the graph in Figure 3.2.

3.2 Computation of betweenness centrality

With the basic definitions out of the way, in this section we set out to prove our main result, a general dependency accumulation formula along with its implication about the algorithmic complexity of $\mathcal{P}$-Betweenness-Centrality.

The main insight of Brandes [Bra01] was that for the set of shortest paths, the cumulative dependencies $\delta_{s\bullet}^{\mathcal{P}}$ can be computed recursively. The same principle was used

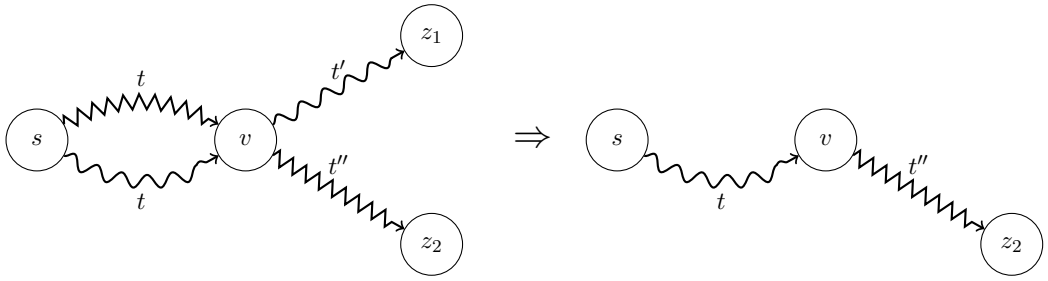


Figure 3.3: Prefix-compatibility condition. On the left we see two paths in $\mathcal{P}$, both starting in s and going through a common vertex appearance (v,t). The path P is the wavy path, while Q is the path with a zigzag pattern. On the right we see what prefix-compatibility stipulates: the path $P[\bullet,v] \oplus Q[v,\bullet]$ must also be in $\mathcal{P}$.

to achieve the results of Buß et al. [Buß+20]. Our goal is to now generalize these insights to yield a more universal result for multiple different path optimality concepts. This will allow us to approach computing the betweenness centrality with respect to an arbitrary concept of optimality (that has a certain property described below) in a unified manner.

First, we will need to count path prefixes which may not necessarily themselves be paths in $\mathcal{P}$ (but can be extended into paths in $\mathcal{P}$).

Definition 3.11 (Path prefix counting)**.** Fix some source $s \in V$ and a vertex appearance $(v,t) \in V \times \mathcal{T}$. We define $\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}$ to be the number of unique prefixes of paths starting in s and going through (v,t). Formally, let $\mathcal{P}_{s(v,t)} \subseteq \mathcal{P}$ be the set of all paths in $\mathcal{P}$ that start in s and go through (v,t). We now have $\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)} \coloneqq \left|\{P[\bullet,(v,t)] \mid P \in \mathcal{P}_{s(v,t)}\}\right|$.

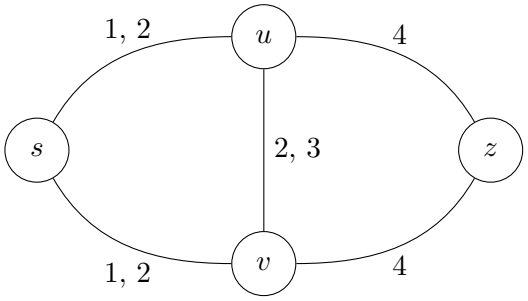
Note that since the source vertex s has a special appearance $(s,0)$ associated with it, the definition above makes sense for all possible appearances $(v,t) \in V \times \mathcal{T}$; we will have $\widetilde{\sigma}^{\mathcal{P}}_{s(s,0)} = 1$[1] and $\widetilde{\sigma}^{\mathcal{P}}_{s(s,t)} = 0$ for all $t \in [T]$. Of course, for any appearance $(v,0)$ with $v \neq s$ we will have $\mathcal{P}_{s(v,0)} = \emptyset$ and therefore also $\widetilde{\sigma}^{\mathcal{P}}_{s(v,0)} = 0$.

With the necessary definitions out of the way, we can move towards the main result of this section, that is, the generalization of the recursive formula for the cumulative dependencies for more general path sets than that of Brandes [Bra01]. The crucial property of a path set that we shall use in our framework is that of *prefix-compatibility*.

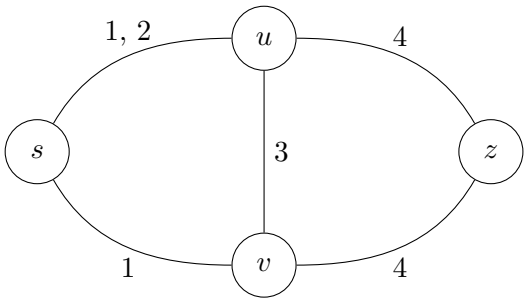
Definition 3.12 (Prefix-compatibility)**.** Let $\mathcal{P}$ be a subset of paths of $\mathcal{G}$. Then $\mathcal{P}$ is said to be *prefix-compatible* if, for an arbitrary vertex $s \in V$ and an arbitrary vertex appearance (v,t), we have for all pairs of paths $P,Q \in \mathcal{P}$ that both start in s and go through (v,t) that $P[\bullet,v] \oplus Q[v,\bullet] \in \mathcal{P}$.

The situation is pictured in Figure 3.3. Intuitively, it only matters that we got to the appearance (v,t) on *some* path in $\mathcal{P}$, and not on *which* path precisely. Hence, we can take parts of any two paths in $\mathcal{P}$ with a common vertex appearance and splice them together at that vertex to get another path in $\mathcal{P}$. Note that many natural path

[1] Again, there could be an edge case if $\mathcal{P}$ does not contain the trivial paths from a vertex to itself. In that case, if there are also no other paths starting in s, then we will have $\widetilde{\sigma}^{\mathcal{P}}_{s(s,0)} = 0$. Indeed, in the example described in Section 4.1 of the next chapter it will be the case for some "unimportant" vertices.



(a) A temporal graph for which the set of foremost paths is *not* prefix-compatible: $P = (s \xrightarrow{1} u \xrightarrow{2} v \xrightarrow{4} z)$ and $Q = (s \xrightarrow{2} v \xrightarrow{3} u \xrightarrow{4} z)$ are both foremost s-z-paths. However, $P[\bullet, (v,2)] \oplus Q[(v,2), \bullet]$ is not a path (and in particular, not a foremost path).



(b) A temporal graph for which the set of foremost paths *is* prefix-compatible. This is easy to see, as the only vertex appearance to which we have multiple paths is $(s, 1)$.

Figure 3.4: Prefix-compatibility of foremost paths. Note that both examples work the same in both the strict- as well as the non-strict case.

optimality concepts will lead to prefix-compatible path sets. For example, the set of all shortest paths is prefix-compatible.

Naturally, some other sets do not have this property in general, e.g. the set of all foremost paths: see Figure 3.4a. Among all foremost s-z-paths there are a few that go through $(v, 2)$. Two of them are $P = (s \xrightarrow{1} u \xrightarrow{2} v \xrightarrow{4} z)$ and $Q = (s \xrightarrow{2} v \xrightarrow{3} u \xrightarrow{4} z)$. However, the walk $P[\bullet, (v,2)] \oplus Q[(v,2), \bullet]$ is not a foremost s-z-path.

That being said, sets which are not prefix-compatible in general may still be prefix-compatible on a specific graph, in which case the results below will still apply. An example of a graph for which the set of foremost paths is prefix-compatible is provided in Figure 3.4b.

Note also that an important part of the definition is that it only cares about pairs of paths that start *in the same vertex* and puts no restrictions on any other pairs of paths.

An important property of all prefix-compatible sets is that the relation induced by their predecessor sets for some arbitrary source $s \in V$ is always acyclic. We will use this property when designing an algorithm computing betweenness centrality for prefix-compatible path sets.

Lemma 3.13. *Let $\mathcal{P}$ be a prefix-compatible set of paths and $\mathcal{P}_s \subseteq \mathcal{P}$ be the set of all paths in $\mathcal{P}$ starting in some source vertex $s \in V$. Then the relation induced by the corresponding predecessor sets $\mathrm{Pre}_s^{\mathcal{P}}$ is acyclic.*

Proof. Assume for the purpose of contradiction that there exist two vertex appearances (v, t) and (w, t') which violate the condition of acyclicity. That means that there exists a sequence of paths $P_1, \ldots, P_n \in \mathcal{P}_s$ such that we have a valid walk $P_1[(v,t), (v_1,t_1)] \oplus P_2[(v_1,t_1), (v_2,t_2)] \oplus \ldots \oplus P_n[(v_{n-1}, t_{n-1}), (w,t')]$ and similarly, there also exists a sequence of paths $Q_1, \ldots, Q_m \in \mathcal{P}_s$ such that we have a valid walk $Q_1[(w,t'), (w_1,t'_1)] \oplus Q_2[(w_1,t'_1), (w_2,t'_2)] \oplus \ldots \oplus Q_m\left[(w_{m-1}, t'_{m-1}), (v,t)\right]$.

Prefix-compatibility applied on (v_1, t_1) implies that $(P_1[\bullet, (v_1, t_1)] \oplus P_2[(v_1, t_1), \bullet]) \in \mathcal{P}_s$. We can keep applying prefix-compatibility to conclude that for $R = P_1[\bullet, (v_1, t_1)] \oplus P_2[(v_1, t_1), (v_2, t_2)] \dots \oplus P_n[(v_{n-1}, t_{n-1}), \bullet]$, we have $R \in \mathcal{P}_s$. Analogously, we can show that $S = \big(Q_1[\bullet, (w_1, t'_1)] \oplus Q_2[(w_1, t'_1), (w_2, t'_2)] \dots \oplus Q_m\big[(w_{m-1}, t'_{m-1}), \bullet\big]\big) \in \mathcal{P}_s$. However, we can now apply prefix-compatibility on (w, t') to conclude that we also have $(R[\bullet, (w, t')] \oplus S[(w, t'), \bullet]) \in \mathcal{P}_s$. Yet (v, t) appears both in R as well as S, so $R[\bullet, (w, t')] \oplus S[(w, t'), \bullet]$ is not a path, contradicting the fact that it is in $\mathcal{P}_s$. □

We can finally move on towards the lemma that underpins the main result of this chapter. The proof is similar to Theorem 6 by Brandes [Bra01] and analogous to Lemma 4.9 by Buß et al. [Buß+20]. Our contribution lies in the generalization of the proof to a wider range of path sets. As we shall see, such a generalization will prove to be of many uses in Chapter 4, where we shall solve $\mathcal{P}$-Betweenness-Centrality for multiple different path sets $\mathcal{P}$.

We start by making one last definition, namely we define a special notion of edge dependency that we will use in our proof.

Definition 3.14 (Edge dependency)**.** We write $\delta^{\mathcal{P}}_{sz}(v, t, (\{v, w\}, t'))$ to denote the fraction of s-z-paths in $\mathcal{P}$ that go through the appearance (v, t) and use the temporal edge $(\{v, w\}, t')$.

Using this definition we can now state and prove a lemma which is the core of our proof of the result below.

Lemma 3.15. *Let $\mathcal{G}$ be a temporal graph. Fix some source vertex $s \in V$. Let $\mathcal{P}$ be a prefix-compatible subset of the temporal paths in $\mathcal{G}$. If $\delta^{\mathcal{P}}_{sz}(v, t, (\{v, w\}, t'))$ is positive, then the following holds:*

$$\delta^{\mathcal{P}}_{sz}(v, t, (\{v, w\}, t')) = \frac{\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}}{\widetilde{\sigma}^{\mathcal{P}}_{s(w,t')}} \cdot \frac{\sigma^{\mathcal{P}}_{sz}(w, t')}{\sigma^{\mathcal{P}}_{sz}}.$$

Proof. Let $\mathcal{P}_s \subseteq \mathcal{P}$ be the set of paths in $\mathcal{P}$ that start in s. Let $P \in \mathcal{P}_s$, be an arbitrary s-z-path that goes through (w, t'). By prefix-compatibility, for any path $Q \in \mathcal{P}_s$ that also goes through (w, t'), we have that $Q[\bullet, (w, t')] \oplus P[(w, t'), \bullet]$ is in $\mathcal{P}$. Therefore we can combine an arbitrary prefix that ends in (w, t') with an arbitrary suffix that starts in (w, t') to get a valid s-z-path.

Let $R \in \mathcal{P}_s$, be an arbitrary path that goes through (v, t). Finally, let $S \in \mathcal{P}_s$ be any s-z path in $\mathcal{P}_s$ that goes through both (v, t) and (w, t'), making the direct transition $v \xrightarrow{t'} w$. Then, by prefix-compatibility, $R[\bullet, (v, t)] \oplus S[(v, t), \bullet] = R[\bullet, (v, t)] \oplus (v \xrightarrow{t'} w) \oplus S[(w, t'), \bullet]$ is in $\mathcal{P}_s$. Similarly, we can apply prefix-compatibility again, to get that $R[\bullet, (v, t)] \oplus (v \xrightarrow{t'} w) \oplus P[(w, t'), \bullet]$ is in $\mathcal{P}$. Hence we can combine an arbitrary prefix which ends in (v, t) with an arbitrary suffix which starts at (w, t') to get a valid s-z-path in $\mathcal{P}$ (see Figure 3.5). With all of that in mind, we can now make the following argument:

Of the $\widetilde{\sigma}^{\mathcal{P}}_{s(w,t')}$ many path prefixes which end in (w, t'), exactly $\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}$ go through (v, t) and use the transition $v \xrightarrow{t'} w$. Now, there are also $\frac{\sigma^{\mathcal{P}}_{sz}(w,t')}{\widetilde{\sigma}^{\mathcal{P}}_{s(w,t')}}$ many unique path suffixes starting from the appearance (w, t') and going to the vertex z. Therefore, there are $\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}$.

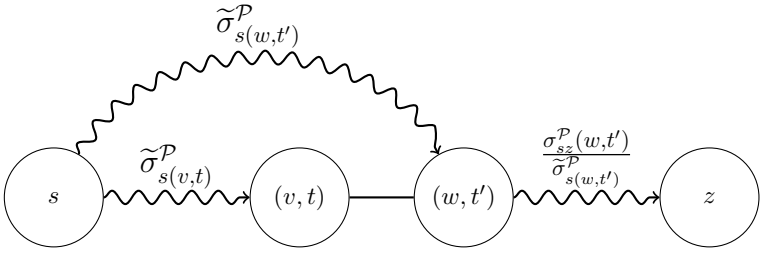


Figure 3.5: Lemma 3.15: combining arbitrary suffix to (v,t) with an arbitrary suffix from (w,t').

$\frac{\sigma^{\mathcal{P}}_{sz}(w,t')}{\widetilde{\sigma}^{\mathcal{P}}_{s(w,t')}}$ many s-z-paths that go through (v,t) and use the transition $v \xrightarrow{t'} w$. Finally, we divide by the total number of s-z-paths to get the expression in the statement of the lemma. □

The lemma above allows us to prove the following dependency accumulation formula for arbitrary prefix-compatible path sets.

Lemma 3.16 (General dependency accumulation)**.** *Let $\mathcal{G}$ be a temporal graph and $\mathcal{P}$ be a prefix-compatible subset of its temporal paths. Fix a source $s \in V$. Then the following temporal dependency accumulation formula holds.*

$$\delta^{\mathcal{P}}_{s\bullet}(v,t) = \delta^{\mathcal{P}}_{sv}(v,t) + \sum_{(w,t') \in \mathrm{Succ}^{\mathcal{P}}_{s}(v,t)} \frac{\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}}{\widetilde{\sigma}^{\mathcal{P}}_{s(w,t')}} \cdot \delta^{\mathcal{P}}_{s\bullet}(w,t') \tag{3.2}$$

Proof. We start by expanding the sum in the formula defining $\delta^{\mathcal{P}}_{s\bullet}(v,t)$ (see Definition 3.8), considering its summands a bit more precisely.

$$\begin{aligned} \delta^{\mathcal{P}}_{s\bullet}(v,t) &= \sum_{z \in V} \delta^{\mathcal{P}}_{sz}(v,t) \\ &= \delta^{\mathcal{P}}_{sz}(v,t) + \sum_{z \in V} \sum_{(w,t') \in \mathrm{Succ}^{\mathcal{P}}_{s}(v,t)} \delta^{\mathcal{P}}_{sz}(v,t,(\{v,w\},t)) \end{aligned} \tag{3.3}$$

Where $\delta^{\mathcal{P}}_{sz}(v,t,(\{v,w\},t))$ is as defined in Definition 3.14. Since (v,t) is not a direct predecessor on any s-v path, we need to pull it out of the sum. Conversely, for any $z \in V \setminus \{v\}$, the vertex v may be contained at most once in any s-z-path. Hence the inner sum of Equation (3.3) precisely captures $\delta^{\mathcal{P}}_{sz}(v,t)$.

We can now use Lemma 3.15 to simplify the summation formula:

$$\begin{aligned}
&\sum_{z\in V} \sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{P}}(v,t)} \delta_{sz}^{\mathcal{P}}(v,t,(\{v,w\},t)) \\
\overset{\text{Lemma 3.15}}{=} &\sum_{z\in V} \sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{P}}(v,t)} \frac{\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}}}{\widetilde{\sigma}_{s(w,t')}^{\mathcal{P}}} \cdot \frac{\sigma_{sz}^{\mathcal{P}}(w,t')}{\sigma_{sz}^{\mathcal{P}_s}} \\
= &\sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{P}}(v,t)} \sum_{z\in V} \frac{\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}}}{\widetilde{\sigma}_{s(w,t')}^{\mathcal{P}}} \cdot \frac{\sigma_{sz}^{\mathcal{P}}(w,t')}{\sigma_{sz}^{\mathcal{P}_s}} \\
= &\sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{P}}(v,t)} \frac{\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}}}{\widetilde{\sigma}_{s(w,t')}^{\mathcal{P}}} \sum_{z\in V} \frac{\sigma_{sz}^{\mathcal{P}}(w,t')}{\sigma_{sz}^{\mathcal{P}_s}} \\
\overset{\text{Definition 3.8}}{=} &\sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{P}}(v,t)} \frac{\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}}}{\widetilde{\sigma}_{s(w,t')}^{\mathcal{P}}} \cdot \delta_{s\bullet}^{\mathcal{P}}(w,t'),
\end{aligned}$$

from which the result immediately follows. □

Note that Equation (3.2) can often be transformed into a simpler form. For example, in the case of shortest paths $\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}}$ will simply be the number of shortest paths from s to (v,t), so we will have $\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}} = \sigma_{s(v,t)}^{\mathcal{P}}$. On the face of it, the simplification may look minor, but it implies that the process of computing the betweenness centrality with respect to the set of shortest paths will be simple (see Section 4.3). Indeed, in the next chapter we shall provide an example where an even greater simplification of Equation (3.2) is exhibited (see Section 4.2).

Lemma 3.16 allows us to finally prove the main theorem of this section by devising an algorithm which uses Equation (3.2) to solve $\mathcal{P}$-Betweenness-Centrality for any prefix-compatible set $\mathcal{P}$. To be able to prove a stronger result than would otherwise be possible, we first observe that because of the structure of Equation (3.2), we can immediately identify some appearances as "useless."

Definition 3.17. Let $\mathcal{G}$ be a temporal graph. Fix a source $s \in V$. We call a vertex appearance $(v,t) \in V \times \mathcal{T}$ *relevant* if either the appearance dependency $\delta_{sv}^{\mathcal{P}}(v,t)$ or the path prefix count $\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}}$ is nonzero.

Observe that by Equation (3.2), for any irrelevant vertex appearance (v,t), the cumulative dependency of s on (v,t), that is, $\delta_{s\bullet}^{\mathcal{P}}(v,t)$, is equal to zero. Hence it cannot contribute anything either to the other cumulative dependencies in the recursion, or to the value of betweenness (see Observation 3.6 and Observation 3.9). It shall soon become clear why the definition above allows us to prove a stronger result than would otherwise be possible.

Theorem 3.1 (General betweenness computation)**.** *Let $\mathcal{G}$ be a temporal graph and $\mathcal{P}$ a prefix-compatible subset of its temporal paths. Assume that for any source vertex $s \in V$ a set of vertex appearances R that includes all relevant vertex appearances, as well as*

- *the appearance dependencies $\delta_{sv}^{\mathcal{P}}(v,t)$;*

- *the prefix counts* $\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}$*;*
- *the predecessor sets* $\mathrm{Pre}^{\mathcal{P}}_s(v,t)$

for all $(v,t) \in R$ *can be computed in* $\mathcal{O}(f(n,M,T))$ *time. Then the betweenness values* $C^{\mathcal{P}}_B(v)$ *can be computed for all vertices* $v \in V$ *in* $\mathcal{O}(nf(n,M,T) + n^3T^2)$ *time.*

Proof. Consider Algorithm 3.1. We assume that the function Count-Prefixes that computes the necessary values in time $f(n,M,T)$ is given to us. We first prove the correctness of the algorithm before analyzing its running time.

Correctness: The general idea of the algorithm is to use Lemma 3.16 to implicitly compute the total betweenness centrality and use Lemma 3.7 to recover the $C^{\mathcal{P}}_B$ values. As Equation (3.1) of Lemma 3.7 has the constant summand $+1$, in Line 2 we initialize the betweenness values to 1.

The next step is to compute the cumulative dependencies $\delta^{\mathcal{P}}_{s\bullet}(v,t)$ for each source vertex s and appropriately update the betweenness values. We do this in the loop starting on Line 3. We first initialize the array holding the cumulative dependencies on Line 4. Note that the vertex appearances without any adjacent edges are certainly unreachable and hence irrelevant, so we do not have to consider them. We then compute the appropriate values for $\delta^{\mathcal{P}}_{sv}(v,t)$, $\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}$, and $\mathrm{Pre}^{\mathcal{P}}_s(v,t)$ using the auxiliary function that we assume we are given.

Finally, in the loop starting on Line 7 we compute the cumulative dependencies using the recursive formula of Lemma 3.16. We proceed in reverse topological order, that is, we start with vertices that have no successors (and hence for which the equation in Lemma 3.16 is trivial to evaluate) as our base case and then proceed backwards. This is possible as prefix-compatible sets always lead to acyclic predecessor graphs (see Lemma 3.13).

Finally, on Line 12 we apply a "connectivity correction." This term corresponds to the $\sum_{w \in V} \left[\sigma^{\mathcal{P}}_{vw} > 0\right]_{\mathbb{1}}$ part of Equation (3.1) in Lemma 3.7. Notice that we do not have to handle the term with $\left[\sigma^{\mathcal{P}}_{wv} > 0\right]_{\mathbb{1}}$ as on Line 11 we only ever add terms which correspond the sum of Equation (3.2), and never the $\delta^{\mathcal{P}}_{sv}(v,t)$ terms. (We do, however, add those terms to the $\delta^{\mathcal{P}}_{s\bullet}(v,t)$ values which then propagate into the C_B array, which is why we need the correction on Line 12.)

Running time: We can easily see that for the loop on Line 4 we need $\mathcal{O}(M) = \mathcal{O}(n^2T)$ time overall. Line 6 takes $\mathcal{O}(f(n,M,T))$ time by assumption. Before the execution of the for-loop on Line 7, we may first need to compute the topological order on the predecessor graph. This graph has at most $\mathcal{O}(nT)$ vertices and hence $\mathcal{O}((nT)^2)$ edges, so the topological ordering can be computed in $\mathcal{O}(n^2T^2)$ time. We then note that the for-loop on Line 7 goes over each edge of the predecessor graph exactly once, so once again we get an $\mathcal{O}(n^2T^2)$ bound for the overall execution time. Finally, Line 12 runs in $\mathcal{O}(nT)$ time (or in constant time, if we do some additional bookkeeping in the loop of Line 11).

Overall, we get $\mathcal{O}\left(n \cdot \left(f(n,M,T) + n^2T^2\right)\right) = \mathcal{O}\left(nf(n,M,T) + n^3T^2\right)$ for the running time of the whole algorithm. □

The significance of the theorem is as follows: let $\mathcal{P}$ be some path set of interest with respect to which we want to compute the betweenness, e.g. the set of shortest paths (we

Algorithm 3.1 General betweenness algorithm, see Theorem 3.1. It uses an auxiliary function COUNT-PREFIXES that returns a (super-) set of relevant vertex appearances R, as well as the values of $\delta^{\mathcal{P}}_{sv}(v,t), \widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}, \mathrm{Pre}^{\mathcal{P}}_{s}(v,t)$ for each of those appearances

Input: A temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$.
Output: Betweenness $C_B(v)$ of all vertices $v \in V(\mathcal{G})$.

for $v \in V$ **do**
  $C_B[v] \leftarrow 1$ ▷ Initialize to 1 per Equation (3.1)
for $s \in V$ **do**
  for $(\{u,v\},t) \in \mathcal{E}$ **do**
    $\delta^{\mathcal{P}}_{s\bullet}[v,t] \leftarrow 0$ ▷ Reset the array
  $R, \delta^{\mathcal{P}}_{sv}(v,t), \widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}, \mathrm{Pre}^{\mathcal{P}}_{s}(v,t) \leftarrow$ COUNT-PREFIXES$(\mathcal{G}, s)$
  for $(w,t') \in R$ in topological order determined by $\mathrm{Pre}^{\mathcal{P}}_{s}$ **do**
    $\delta^{\mathcal{P}}_{s\bullet}(w,t') \leftarrow \delta^{\mathcal{P}}_{s\bullet}(w,t') + \delta^{\mathcal{P}}_{sw}(w,t')$ ▷ Appearance dependency on (w,t')
    for $(v,t) \in \mathrm{Pre}^{\mathcal{P}}_{s}(w,t')$ **do**
      $\delta^{\mathcal{P}}_{s\bullet}[v,t] \leftarrow \delta^{\mathcal{P}}_{s\bullet}[v,t] + \frac{\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}}{\widetilde{\sigma}^{\mathcal{P}}_{s(w,t')}} \cdot \delta^{\mathcal{P}}_{s\bullet}[w,t']$ ▷ Sum of Equation (3.2)
      $C_B[v] \leftarrow C_B[v] + \frac{\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}}{\widetilde{\sigma}^{\mathcal{P}}_{s(w,t')}} \cdot \delta^{\mathcal{P}}_{s\bullet}[w,t']$
  $C_B[s] \leftarrow C_B[s] - |\{v \mid \exists t \in [T] : \delta^{\mathcal{P}}_{sv}(v,t) > 0\}|$ ▷ Connectivity correction
return C_B

shall see more examples in Chapter 4). Then, if $\mathcal{P}$ is prefix-compatible, we can reduce solving $\mathcal{P}$-BETWEENNESS-CENTRALITY to computing the predecessor sets and the $\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}$ values by supplying the function COUNT-PREFIXES. For example, Buß et al. [Buß+20] have effectively shown that for the set of shortest and the set of shortest-foremost paths we have $f = \mathcal{O}(n^2T^2)$, which using Theorem 3.1 yields $\mathcal{O}(n^3T^2)$ running time.

Furthermore, we can now see why the definition of "relevant" vertices is important: otherwise, the output of COUNT-PREFIXES would always have to be $\mathcal{O}(nT)$ in size. This would, in turn, imply that our predecessor graph has $\mathcal{O}(nT)$ vertices and so the running time provided by Theorem 3.1 would also have a lower bound of $\Omega(n^3T^2)$. However, this bound can be unnecessarily tight. Indeed, in one of the examples that we shall see in Chapter 4, we show a case where our predecessor graph has only $\mathcal{O}(n)$ vertices and $\mathcal{O}(M)$ edges. Hence, to be able to use Algorithm 3.1 in its fullest, we only require COUNT-PREFIXES to consider the relevant appearances.

The considerations above lead us to a slightly stronger result. Indeed, the corollary below merely quantifies the logic of our proof of Theorem 3.1 in a somewhat more precise manner. We only separate those two statements to simplify the notation in Theorem 3.1.

Corollary 3.18. *Let $\mathcal{G}$ be a temporal graph and $\mathcal{P}$ a prefix-compatible subset of its temporal paths. Assume that for any source vertex $s \in V$ a set of vertex appearances R that includes all relevant vertex appearances R, as well as*

- *the appearance dependencies $\delta^{\mathcal{P}}_{sv}(v,t)$;*
- *the prefix counts $\widetilde{\sigma}^{\mathcal{P}}_{s(v,t)}$;*

- *the predecessor sets* $\mathrm{Pre}_s^{\mathcal{P}}(v,t)$

for all $(v,t) \in R$ *can be computed in* $\mathcal{O}(f(n,M,T))$ *time, possibly amortized across all* n *vertices*[2]*. If the set* R *has size bounded by* $\mathcal{O}(g(n,M,T))$ *and the predecessor graph on* R *induced by* $\mathrm{Pre}_s^{\mathcal{P}}$ *has* $\mathcal{O}(h(n,M,T))$ *edges, then the betweenness values* $C_B^{\mathcal{P}}(v)$ *can be computed for all vertices* $v \in V$ *in* $\mathcal{O}\left(n\left[M + f(n,M,T) + g(n,M,T) + h(n,M,T)\right]\right)$ *time.* □

Naturally, we always have $g = \mathcal{O}(nT)$ and $h = \mathcal{O}(n^2T^2)$, which is the property we used in our proof of Theorem 3.1. Note that the theorems also apply to static graphs (which can be regarded as temporal graphs with $T = 1$ and non-strict paths). For such graphs we have $g = \mathcal{O}(n)$ and $h = \mathcal{O}(m)$. Therefore, Corollary 3.18 yields $\mathcal{O}(nm + nf(n,M,T))$ for any static graph, regardless of the path optimality concept used (as long as it is prefix-compatible). In particular, Brandes [Bra01] showed that for the standard betweenness centrality we have $f = \mathcal{O}(m)$ yielding the well-known $\mathcal{O}(nm)$ complexity for computing betweenness centrality in a static graph.

[2] That is, the computation for all n vertices can be carried out in $\mathcal{O}(nf(n,M,T))$ overall time.

Chapter 4

A case study of framework applications

In the previous chapter, we presented a property of path sets that leads to the existence of a recursive formula for calculating the cumulative dependencies, which in turn has some algorithmic implications about $\mathcal{P}$-Betweenness-Centrality. In this chapter, we use the framework on some examples of increasing complexity in order to show how it can be used to compute betweenness centrality with respect to a variety of different path sets. In Section 4.1, we use of Theorem 3.1 to solve a simple problem yet one that, to the best of our knowledge, has not been analyzed before. In Section 4.2, we show a case that has already appeared in literature, allowing us to contextualize our framework and show how it fits together with the previous results. Finally, in Section 4.3 we use the framework to show a more complicated and more general result. Namely, extending the results of Buß et al. [Buß+20], we present a polynomial-time algorithm that computes the betweenness centrality with respect to any combination of the canonical optimality criteria (shortest, fastest, foremost) with lexicographic tie-braking, except for the combinations that have previously been shown to be #P-hard.

4.1 Subset betweenness

Consider the following scenario: a nation state has a set of critical hospitals offering specialty care that are spread all throughout country and are connected with a rail network. Naturally, there are also many train stations on the paths between the hospitals, or even stations which are off to the side and do not lie on any paths connecting the hospitals themselves. Now, for patient safety reasons, the government wants to analyze the connectivity of those hospitals to determine what the critical (intermediate) train stations are. Hence, betweenness centrality is a natural measure to consider. However, notice that the situation is different than the standard scenario in which betweenness centrality is used. We have two categories of vertices in our graph representing the rail network: one, the hospitals, and two, all the other intermediate stations that the trains go through. Yet we only care about the connectivity of the former and not the latter—of course, the internal structure of the intermediate vertices does affect the connectivity of the hospitals, but we are only interested in this indirect effect, while the

direct connectivity of the intermediate stations with each other is immaterial for us.

The scenario above directly leads to a different measure of betweenness. In all generality, we may have two special sets of vertices $S, Z \subseteq V$, and are only interested in shortest paths going from the set S to the set Z.[1] We may have $S = Z$ in which case we effectively choose a special set of "terminal" vertices and the rest become "intermediate" vertices whose connectedness interests us only insofar as it affects the connectivity of the terminal vertices—exactly the scenario described above. Moreover, in particular we can have $S = Z = V$, in which case all vertices are the "important vertices," so we simply get shortest betweenness. Hence, the *subset betweenness* as described above can be seen as a generalization of the standard shortest betweenness.

To formalize the problem, let $\mathcal{G}$ be a temporal graph, $S, Z \subseteq V$ be subsets of its vertices, and $\mathcal{P}^{(\mathrm{SZ})}$ be the set of shortest S-Z paths, that is,

$$\mathcal{P}^{(\mathrm{SZ})} \coloneqq \{P \mid P \text{ is a shortest } s\text{-}z\text{-path for some } s \in S \text{ and } z \in Z\}.$$

Then the *subset betweenness problem* is simply the $\mathcal{P}^{(\mathrm{SZ})}$-Betweenness-Centrality problem.

It is easy to see that $\mathcal{P}^{(\mathrm{SZ})}$ is prefix-compatible. Moreover, we can also see that it is easy to compute both the predecessor sets $\mathrm{Pre}_s^{\mathcal{P}^{(\mathrm{SZ})}}$ and the prefix counts $\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}^{(\mathrm{SZ})}}$, since they are both exactly the same as in the case of shortest betweenness (because prefixes of shortest S-Z paths are themselves shortest paths). This computation can therefore be done with an augmented BFS, in the same manner as described in Buß et al. [Buß+20], in time $f(n, M, T) = \mathcal{O}(n^2T^2)$.

Putting it together, we can now use Theorem 3.1.

Proposition 4.1. *$\mathcal{P}^{(\mathrm{SZ})}$-Betweenness-Centrality can be solved in $\mathcal{O}(n^3T^2)$ time.* □

Notably, we can also see that for any vertex $s \in S$ and any vertex $v \in V \setminus (S \cup Z)$ we will have that $\delta_{sv}^{\mathcal{P}^{(\mathrm{SZ})}}(v, t) = 0$, so by Lemma 3.16 we see that $\hat{C}_B(v) > 0$ if and only if v participates in some shortest S-Z paths, as expected from our intuitive description at the beginning of this section.

Finally, it is also easy to see that the observations of this section can be generalized in the following way:

Proposition 4.2. *Let $\mathcal{G}$ be a temporal graph, $\mathcal{P}$ be a prefix-compatible subset of its paths, and $S, Z \subseteq V$ be arbitrary subsets of its vertices. Then the corresponding path subset*

$$\mathcal{P}^{(\mathrm{SZ})} \coloneqq \{P \in \mathcal{P} \mid P \text{ is an } s\text{-}z\text{-path for some } s \in S \text{ and } z \in Z\}$$

is also prefix-compatible. □

[1] Note that we have chosen shortest paths for simplicity of presentation. However, one can easily see that all the results shown later in this chapter would analogously generalize to their respective variant of subset betweenness.

4.2 Prefix-foremost betweenness

In this section, we analyze the case of prefix-foremost paths. This path optimality concept has already been studied by Buß et al. [Buß+20], so our focus shall lie mostly on showing how our results interact with those of Buß et al. [Buß+20]. First, we apply Lemma 3.16 to the case of prefix-foremost paths to recover a result from the paper and to show how greatly Equation (3.2) of Lemma 3.16 can simplify in some cases. Second, we show how the framework of Theorem 3.1 can be used on the case of prefix-foremost paths.

The concept of foremost paths is probably one of the most natural concepts of path optimality in temporal graphs. Hence, they have immediately appeared as an important target of study. However, as was proven by Buß et al. [Buß+20], the $\mathcal{P}$-Betweenness-Centrality problem with respect to the set of foremost paths is #P-hard, both in the case of non-strict, as well as the case of strict paths. In the following, we shall therefore try to work around this computational difficulty. However, we shall only focus on the strict case, as it turns out that the approach we describe below is insufficient in the non-strict case, still leading to a #P-hard problem (for a proof of that last assertion, see the paper of Buß et al. [Buß+20]).

One of the possible remedies to the difficulty of dealing with foremost paths involves making the observation that on any such foremost path, the only restriction is arrival time at the target vertex. Thus, on the way there we can make multiple detours and, in some cases, move around the graph almost arbitrarily. We would therefore like to place an additional requirement on the paths to limit such pathological behavior. Namely, we add the condition that all the *prefixes* of the paths also be foremost paths. This proves to be a powerful definition as, intuitively speaking, it forces paths to be "efficient enough" as to allow the corresponding betweenness problem to be computationally tractable.

Definition 4.3 (Prefix-foremost path (-set))**.** Let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph and $s, z \in V$ a pair of vertices. An s-z-path P is *prefix-foremost* if, for any $v \in V$ on P, we have that $P[\bullet, v]$ is a foremost path. We denote the set of all *strict* prefix-foremost paths in $\mathcal{G}$ by $\mathcal{P}^{(\mathrm{pfm})}$.

It was shown by Wu et al. [Wu+16] that if there exists any (strict) s-z-path, then there exists a (strict) prefix-foremost s-z-path.

Observation 4.4 (Prefix-foremost properties)**.** *Fix a source $s \in V$. For any vertex $v \in V$ that s is connected to, define by t_v the foremost arrival time from s to v. From the definition of the set of prefix-foremost paths $\mathcal{P}^{(\mathrm{pfm})}$ we can easily see the following:*

- *$\mathcal{P}^{(\mathrm{pfm})}$ is prefix-compatible.*
- *$\widetilde{\sigma}^{\mathcal{P}^{(\mathrm{pfm})}}_{s(v,t_v)} = \sigma^{\mathcal{P}^{(\mathrm{pfm})}}_{sv}$ and $\widetilde{\sigma}^{\mathcal{P}^{(\mathrm{pfm})}}_{s(v,t)} = 0$ for any $t \in \mathcal{T} \setminus \{t_v\}$.*
- *All paths starting in s and going through v use the same appearance (v, t_v).*
- *Since at most one appearance of v is ever used on paths in $\mathcal{P}^{(\mathrm{pfm})}$, we can simply write $\mathrm{Succ}^{\mathcal{P}^{(\mathrm{pfm})}}_s(v) := \mathrm{Succ}^{\mathcal{P}^{(\mathrm{pfm})}}_s(v, t_v)$, as well as $w \in \mathrm{Succ}^{\mathcal{P}^{(\mathrm{pfm})}}_s(v)$ instead of $(w, t_w) \in \mathrm{Succ}^{\mathcal{P}^{(\mathrm{pfm})}}_s(v)$.*

With all of that in mind we can show how Equation (3.2) simplifies in the case of $\mathcal{P}^{(\text{pfm})}$ and recover Lemma 4.13 of Buß et al. [Buß+20].

Proposition 4.5 (Lemma 4.13 in Buß et al. [Buß+20])**.** *Let $\mathcal{P}^{(\text{pfm})}$ be the set of strict prefix-foremost paths in a temporal graph $\mathcal{G}$. Let $s \in V$ be a source and $v \in V$ a vertex such that s is connected to v. Then the following holds:*

$$\delta_{s\bullet}^{\mathcal{P}^{(\text{pfm})}}(v) = 1 + \sum_{w \in \text{Succ}_s^{\mathcal{P}^{(\text{pfm})}}(v)} \frac{\sigma_{sv}^{\mathcal{P}^{(\text{pfm})}}}{\sigma_{sw}^{\mathcal{P}^{(\text{pfm})}}} \cdot \delta_{s\bullet}^{\mathcal{P}^{(\text{pfm})}}(w).$$

Proof. The proof is mostly an exercise in notational rewriting. Since $\mathcal{P}^{(\text{pfm})}$ is prefix-compatible, we can use Lemma 3.16 to get:

$$\begin{aligned}
\delta_{s\bullet}^{\mathcal{P}^{(\text{pfm})}}(v) &\overset{\textit{Observation 3.9}}{=} \sum_{t \in \mathcal{T}} \delta_{s\bullet}^{\mathcal{P}^{(\text{pfm})}}(v,t) \\
&\overset{\textit{Lemma 3.16}}{=} \sum_{t \in \mathcal{T}} \left(\delta_{sv}^{\mathcal{P}^{(\text{pfm})}}(v,t) + \sum_{(w,t') \in \text{Succ}_s^{\mathcal{P}^{(\text{pfm})}}(v,t)} \frac{\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}^{(\text{pfm})}}}{\widetilde{\sigma}_{s(w,t')}^{\mathcal{P}^{(\text{pfm})}}} \cdot \delta_{s\bullet}^{\mathcal{P}^{(\text{pfm})}}(w,t') \right) \\
&= \sum_{t \in \mathcal{T}} \delta_{sv}^{\mathcal{P}^{(\text{pfm})}}(v,t) + \sum_{t \in \mathcal{T}} \sum_{(w,t') \in \text{Succ}_s^{\mathcal{P}^{(\text{pfm})}}(v,t)} \frac{\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}^{(\text{pfm})}}}{\widetilde{\sigma}_{s(w,t')}^{\mathcal{P}^{(\text{pfm})}}} \cdot \delta_{s\bullet}^{\mathcal{P}^{(\text{pfm})}}(w,t') \\
&\overset{\textit{Observation 3.9}}{=} 1 + \sum_{t \in \mathcal{T}} \sum_{(w,t') \in \text{Succ}_s^{\mathcal{P}^{(\text{pfm})}}(v,t)} \frac{\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}^{(\text{pfm})}}}{\widetilde{\sigma}_{s(w,t')}^{\mathcal{P}^{(\text{pfm})}}} \cdot \delta_{s\bullet}^{\mathcal{P}^{(\text{pfm})}}(w,t') \\
&\overset{\textit{Observation 4.4}}{=} 1 + \sum_{w \in \text{Succ}_s^{\mathcal{P}^{(\text{pfm})}}(v)} \frac{\widetilde{\sigma}_{s(v,t_v)}^{\mathcal{P}^{(\text{pfm})}}}{\widetilde{\sigma}_{s(w,t_w)}^{\mathcal{P}^{(\text{pfm})}}} \cdot \delta_{s\bullet}^{\mathcal{P}^{(\text{pfm})}}(w,t_w) \\
&\overset{\textit{Observation 4.4}}{=} 1 + \sum_{w \in \text{Succ}_s^{\mathcal{P}^{(\text{pfm})}}(w)} \frac{\sigma_{sv}^{\mathcal{P}^{(\text{pfm})}}}{\sigma_{sw}^{\mathcal{P}^{(\text{pfm})}}} \cdot \delta_{s\bullet}^{\mathcal{P}}(w),
\end{aligned}$$

since $\sum_{t \in \mathcal{T}} \delta_{sv}^{\mathcal{P}^{(\text{pfm})}}(v,t) = \delta_{sv}^{\mathcal{P}^{(\text{pfm})}}(v) = 1$ for any vertex v that s is connected to. □

For the second part of this section, we showcase a use of the framework of Theorem 3.1 (or, more precisely, Corollary 3.18) on the case of prefix-foremost paths. To do so, we need to provide a Count-Prefixes function for the case of prefix-foremost paths. Algorithm 4.1 does just that. Note that the algorithm is effectively a part of Algorithm 2 by Buß et al. [Buß+20]. Moreover, Algorithm 4.2 does the job of "translating" the output of Algorithm 4.1 into the "interface" of Corollary 3.18. Using those algorithms together, we recover another result from the paper:

Proposition 4.6 (See Table 1 in Buß et al. [Buß+20])**.** *The $\mathcal{P}^{(\text{pfm})}$-Betweenness-Centrality problem can be solved in $\mathcal{O}(nM \log M + n^2)$ time.*

Proof. Consider Algorithm 4.1. Its correctness follows from Observation 4.4. Since $\mathcal{P}^{(\text{pfm})}$ is clearly prefix-compatible, we can now use the Corollary 3.18 for the running time.

Algorithm 4.1 COUNT-PREFIXES$(\mathcal{G}, s)$ for the strict prefix-foremost case, effectively a part of Algorithm 2 in Buß et al. [Buß+20]

Input: A temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$ and a source vertex $s \in V$.

Output: Set of relevant appearances R and $\delta_{sv}^{\mathcal{P}^{(\mathrm{pfm})}}(v,t)$, $\widetilde{\sigma}_{s(v,t_v)}^{\mathcal{P}^{(\mathrm{pfm})}}$, and $\mathrm{Pre}_s^{\mathcal{P}^{(\mathrm{pfm})}}(v,t)$ for those appearances.

1: **for** $v \in V$ **do** ▷ Initialization
2: $\quad \sigma[v] \leftarrow 0$
3: $\quad t_{\min}[v] \leftarrow -1$
4: $\quad P[v] \leftarrow \emptyset$
5: $Q \leftarrow$ empty priority queue of edges, with their timestamps as keys
6: $Q.\mathrm{enqueueAll}(\{s \xrightarrow{t} v \mid s \xrightarrow{t} v \in \mathcal{E}\})$
7: **while** Q not empty **do**
8: $\quad v \xrightarrow{t} w \leftarrow Q.\mathrm{extract\text{-}min}()$
9: $\quad$ **if** $t_{\min}[w] = -1$ **then** ▷ First and foremost arrival in w
10: $\quad\quad t_{\min}[w] \leftarrow t$
11: $\quad\quad Q.\mathrm{enqueueAll}(\{w \xrightarrow{t'} x \mid w \xrightarrow{t'} x \in \mathcal{E}, t < t'\})$
12: $\quad$ **if** $t_{\min}[w] = t$ **then** ▷ Foremost arrival in w
13: $\quad\quad \sigma[w] \leftarrow \sigma[w] + \sigma[v]$
14: $\quad\quad P[w] \leftarrow P[w] \cup v$
return TRANSFORM$(\sigma, t_{\min}, P)$

Since each edge can only be added at most once to the priority queue (on Line 11), we have $f(n, M, T) = \mathcal{O}(M \log M)$. Furthermore, there is at most one relevant vertex appearance per vertex in $\mathcal{G}$ (see Observation 4.4) and so the predecessor graph obviously cannot have more than M edges. We therefore get $g(n, M, T) = \mathcal{O}(n)$ and $h(n, M, T) = \mathcal{O}(M)$.

Overall, by Corollary 3.18, we get a running time of $\mathcal{O}(n\,[M + M \log M + n + M]) = \mathcal{O}(nM \log M + n^2)$. □

As a final remark, notice that the first part of this section, the proof of Proposition 4.5 was not necessary at all in our consideration of $\mathcal{P}^{(\mathrm{pfm})}$-BETWEENNESS-CENTRALITY—it was only presented to show another connection between this work and the paper of Buß et al. [Buß+20]. Indeed, Proposition 4.5 is already "contained" in Corollary 3.18—this is the power of the generalization made in Chapter 3. For the purpose of using Corollary 3.18, we only needed Observation 4.4 (that was necessary to devise Algorithm 4.1).

4.3 Combinations of optimality criteria

There are three canonical concepts of path optimality in temporal graphs that are the most studied: shortest, foremost, and fastest [Hol15; HS12; Wu+14]. Note that some other optimality concepts have appeared previously, that we shall, however, not study here, for example, *latest-departure* paths [Wu+16] (which for our purposes are computationally similar to foremost paths), or paths with waiting-time constraints (where the time spent on a vertex in a path is bounded) [Cas+20; Him+19].

Algorithm 4.2 An auxiliary function to translate the output of Algorithm 4.1 into the framework of Corollary 3.18

function TRANSFORM(σ, $t_{\min}$, P)
  $R \leftarrow \emptyset$
  for $v \in V$ **do**
    $R \leftarrow R \cup \{(v, t_{\min}[v])\}$
    $\delta_{sv}^{\mathcal{P}^{(\mathrm{pfm})}}(v, t_{\min}[v]) \leftarrow 1$
    $\widetilde{\sigma}_{s(v,t_v)}^{\mathcal{P}^{(\mathrm{pfm})}} \leftarrow \sigma[v]$
    $\mathrm{Pre}_s^{\mathcal{P}^{(\mathrm{pfm})}}(v, t_{\min}[v]) \leftarrow P[v]$
  return $R, \delta_{sv}^{\mathcal{P}^{(\mathrm{pfm})}}(v,t), \widetilde{\sigma}_{s(v,t_v)}^{\mathcal{P}^{(\mathrm{pfm})}}, \mathrm{Pre}_s^{\mathcal{P}^{(\mathrm{pfm})}}(v,t)$

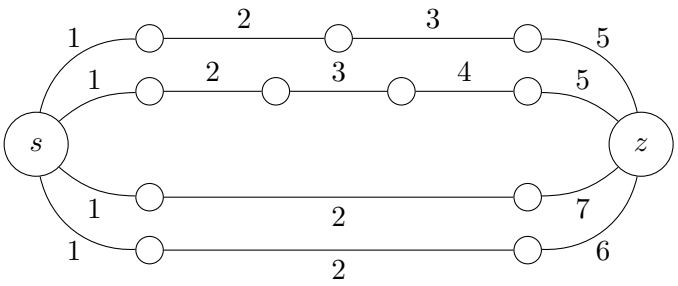


Figure 4.1: Optimality criteria: the order in which they are combined matters. The top path is the only shortest-fastest path, while the bottom one is the only fastest-shortest path.

The aforementioned canonical path optimality concepts all correspond to minimizing a certain quantity: path length, arrival time, duration. However, sometimes minimizing only one of the quantities leads to undesirable results: consider a public rail transport network. We might want to spend as little time as possible traveling, so we decide to use duration as our metric. That being said, some fastest paths may be very long in terms of the amount of edges, that is, we would have to switch trains many times on our journey. This is obviously highly inconvenient, increases the risk of boarding the wrong train, and so on. Hence, all else being equal, we would like to make as few changes as possible on our fastest path. More formally speaking, we would like to combine some optimality measures to obtain new ones. Towards that end, we shall construct new measures via lexicographic tie-breaking. For example, a path is shortest-fastest if it is the shortest path among all fastest paths.

Observe that the order matters: a shortest-fastest path will, in general, not be a fastest-shortest path. For example, consider the graph in Figure 4.1. The upper two paths are the only fastest paths. However, only the very top one is shortest-fastest, as it has fewer edges than the upper-middle one. Similarly, the lower two paths are the only shortest paths. However, only the bottom path is fastest-shortest, as the lower-middle one has a larger duration. Hence we see that shortest-fastest and fastest-shortest paths can be of a different duration or length.

Note that adding the same criterion twice has no effect: shortest-fastest-shortest

Table 4.1: Tractability of different variants of $\mathcal{P}$-Betweenness-Centrality, as shown by Buß et al. [Buß+20]. For example, the first entry in the third column refers to shortest-foremost betweenness, which was shown to be solvable in polynomial time. Note that the diagonal of the table also corresponds to the optimality concepts which are not combinations of multiple different ones. For example, shortest-shortest paths are the same as shortest paths. Hence, the table implicitly contains all possible ways of combining at most two path optimality concepts.

	Shortest	Fastest	Foremost
Shortest	P	?	P
Fastest	?	#P-hard	#P-hard
Foremost	?	#P-hard	#P-hard

paths are exactly the same paths as fastest-shortest paths, since the latter already all have the same length by definition. Therefore, it only makes sense to combine criteria at most twice (using each of shortest, fastest and foremost at most once), as doing it three or more times would yield a redundant measure.

This concept has already been studied by Buß et al. [Buß+20], where a few variants have been proven #P-hard, while a polynomial-time algorithm has been provided for some of the others. The results from the paper have been summarized in Table 4.1. To the best of our knowledge, no algorithm has been provided for the variants not handled in the paper. In this section, we shall use the framework of Theorem 3.1 to fill in those remaining entries in the table as well as solve the problem for all remaining non-trivial combinations of criteria, that is, those combining three measures of optimality, of which there are $3! = 6$ (since the ones which contain one of shortest, fastest, or foremost more than once are equivalent to one of the entries in the table). Note that our results cover both the strict as well as the non-strict case. Indeed, all of our proofs work in both of those cases, so we will not specify whether strict or non-strict paths are meant in the statements of lemmas and theorems.

We start by formalizing the notion of putting different optimality criteria together in a way which allows us to cover all possible combinations in a unified manner. In order to do that, we shall first define $c = (c_1, c_2, c_3)$, a "criterion function" that returns a tuple of numbers that we shall then compare lexicographically. For example, $(3, 5, 5) < (4, 2, 1)$ and $(2, 2, 2) < (2, 2, 3)$. Each entry of the tuple, c_i, will be one of path length, duration and arrival time.

Definition 4.7 (Criterion function)**.** We call a *criterion function* a member of the following family of functions:

$$\begin{aligned} c : \quad & \mathbb{N}^3 \to \mathbb{N}^3 \\ & (\ell, d, t) \mapsto (c_1, c_2, c_3) \end{aligned}$$

where for each c_i we have $c_i \in \{\ell, d, t, 1\}$.

By convention we shall always treat the first argument as the path-length, the second as duration, and the third as the arrival time. Indeed, for convenience we also overload the function for paths.

Definition 4.8. For a path set $\mathcal{P}$ we overload a criterion function c to a function $\mathcal{P} \to \mathbb{N}^3$. For a path $P \in \mathcal{P}$ of length ℓ, duration d, and arrival time t, we write $c(P) \coloneqq c(\ell, d, t)$.

Note that we allow the elements of the output tuple to be a constant, allowing us to formalize the unrestricted path set with $c(\ell, d, t) = (1, 1, 1)$. Also note that in some cases we can encode a combination of the canonical optimality criteria as a criterion function in multiple ways. For example, for fastest paths we could choose $c(\ell, d, t) = (d, 1, 1)$, but since fastest paths are the same as fastest-fastest-fastest paths, we could also take $c(\ell, d, t) = (d, d, d)$; if we want to consider shortest-foremost paths, then one possible choice would be $c(\ell, d, t) = (t, \ell, \ell)$, and another could be $c(\ell, d, t) = (t, 1, \ell)$.

We shall adopt one more convention when dealing with criterion functions. For $v \in V$ and $t \in \mathcal{T}$, if the source vertex and the respective values of distance, duration, and arrival of the paths in question are clear from the context, then we shall simply write $c(v)$ or $c(v, t)$, instead of directly referring to those values of our vertex or vertex appearance in consideration. This convention will prove useful when writing down a generalized algorithm later in this section.

Observe that each criterion function naturally determines the set of optimal paths between any two vertices (or vertex appearances).

Definition 4.9 (c-optimal paths)**.** Let c be a criterion function and $\mathcal{P}$ an arbitrary set of paths in a temporal graph $\mathcal{G}$. Finally, let $s \in V$ be an arbitrary source vertex. Then for every vertex $v \in V$ we call an s-v-path $P \in \mathcal{P}$ *c-optimal* if for every other s-v-path $P' \in \mathcal{P}$ we have $c(P) \leq c(P')$. Similarly, for any vertex appearance (v, t), we call an s-(v, t)-path c-optimal if it has a minimal value of c among all s-(v, t)-paths.

Taken further, we see that every criterion function c induces a whole path set of c-optimal paths for every pair of vertices.

Definition 4.10 (Induced path sets)**.** Let c be a criterion function. Let $\mathcal{P}$ be any set of paths in a temporal graph $\mathcal{G}$. Let $\mathcal{P}' \subseteq \mathcal{P}$ be a set of paths such that for every pair of vertices $s, z \in V$ we have that $\mathcal{P}'$ contains all c-optimal s-z-paths. Formally, for any $s, z \in V$, let $\mathcal{P}_{sz} \subseteq \mathcal{P}$ represent the set of all s-z-paths in $\mathcal{P}$. Then

$$\mathcal{P}' \coloneqq \bigcup_{s,z \in V} \{P \in \mathcal{P}_{sz} \mid \forall P' \in \mathcal{P}_{sz} \colon c(P) \leq c(P')\}.$$

We call $\mathcal{P}'$ the *induced path set (of c)*.

With the concept of criterion functions defined, we can now turn our attention towards the main objective of this section, namely providing a general solution to the $\mathcal{P}$-Betweenness-Centrality problem with respect to any set of paths induced by a criterion function. We start by showing the following simple lemma.

Lemma 4.11 (Walk-to-path construction)**.** *Let W be an s-z walk, for some $s, z \in V$. Then there exists a path P with length and arrival time no larger than-, and departure time no smaller than that of W. Additionally, if W is not a path, then P is shorter than W.*

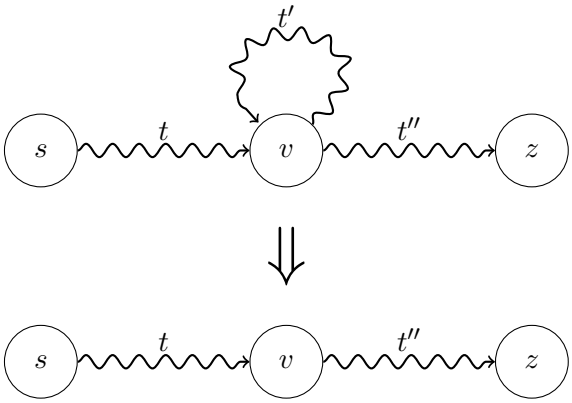


Figure 4.2: Walk-to-path construction

Proof. If W is a path, then the statement is trivial. Otherwise, there exists some vertex visited by W more than once. Let $v \in V$ be the first such vertex. Let (v, t) be the first appearance of v in W and let (v, t') be the last appearance of v in W.

Then the walk $W' = W\,[\bullet, (v, t)] \oplus W\,[(v, t'), \bullet]$, illustrated on Figure 4.2, is a path. Since it is a concatenation of a prefix and a suffix of W, it cannot have a worse length, departure, or arrival time. Finally, since W' visits (v, t) only once while W does so at least twice, W' is also shorter than W. □

We shall now restrict our attention to those criterion functions, whose output tuple contains path length as (at least) one of its entries. That is, we will exclude the unrestricted path set as well as the ones based only on an optimal combination of duration and arrival time. We know that computing the betweenness with respect to any such optimality criterion that does not include path length is #P-hard [Buß+20].[2] We shall now prove that $\mathcal{P}$-Betweenness-Centrality with respect to all remaining optimality concepts can be solved in polynomial time. We first turn our attention to a few simple lemmas that underpin the main result of this section.

Lemma 4.12. *Let $\mathcal{P}$ be a subset of temporal paths in $\mathcal{G}$ induced by some criterion function c. Consider now the set $\mathcal{P}^{(\mathrm{fa})} \subseteq \mathcal{P}$ of paths in $\mathcal{P}$ that for each pair of vertices $s, z \in V$ contains the set of fastest s-z-paths among those in $\mathcal{P}$.*

Let $P \in \mathcal{P}^{(\mathrm{fa})}$ be an arbitrary path. Let (v, t) be any appearance in P. Let Q be any other path in $\mathcal{P}^{(\mathrm{fa})}$ starting in the same source vertex $s \in V$ and going through (v, t). Then Q has the same departure time as P.

Proof. Let t_P and t_Q be the departure times of P and Q, respectively. Assume for the purpose of contradiction that $t_P \neq t_Q$. In particular, assume without loss of generality that $t_P < t_Q$.

Observe that the walk $R = Q\,[\bullet, (v, t)] \oplus P\,[(v, t), \bullet]$ is faster than P. If it is not a path, then, by Lemma 4.11, we can make it into a path R', without making it worse with

[2] Note that the case of fastest-foremost and foremost-fastest paths is not explicitly handled by Buß et al. [Buß+20], however the same proof as that for the case of foremost and for the case of fastest paths, using the exact same reduction, would also work on foremost-fastest and fastest-foremost paths.

respect to c. This path is faster than P. Hence, the only reason why it is not in $\mathcal{P}^{(\mathrm{fa})}$ can be that it is not in $\mathcal{P}$. However, it arrives no later than P. Therefore we can only have that $R' \notin \mathcal{P}$ if R', in spite of being faster, is longer than P. This implies that even if duration is one of the criteria in c, length is of "higher priority."

The fact that R' is longer than P implies that $Q[\bullet, (v,t)]$ is longer than $P[\bullet, (v,t)]$. However, then the walk $S = P[\bullet, (v,t)] \oplus Q[(v,t), \bullet]$ has the same arrival time as Q but is shorter. Again, by Lemma 4.11, it can be made into a path S'. This path may be slower than Q, but as we have already established, length has a higher priority in $\mathcal{P}$. Hence, Q cannot be optimal.

We have reached a contradiction, therefore we must have $t_P = t_Q$. □

This lemma has the following obvious corollary that we will need in our proof of the main result.

Corollary 4.13. *Let $\mathcal{P}^{(\mathrm{fa})}$ be as in Lemma 4.12. Let $\mathcal{P}' \subseteq \mathcal{P}^{(\mathrm{fa})}$ be an arbitrary subset of $\mathcal{P}^{(\mathrm{fa})}$. Then the property described in Lemma 4.12, i.e., that paths starting from the same source and going through the same appearance have the same departure time, still holds.* □

The corresponding lemma about shortest paths is also true.

Lemma 4.14. *Let $\mathcal{P}$ be a subset of temporal paths in $\mathcal{G}$ induced by some criterion function c. Consider now the set $\mathcal{P}^{(\mathrm{sh})} \subseteq \mathcal{P}$ of paths in $\mathcal{P}$ that for each pair of vertices $s, z \in V$ contains the set of shortest s-z-paths among those in $\mathcal{P}$.*

Let $P \in \mathcal{P}^{(\mathrm{sh})}$ be arbitrary. Let (v,t) be any appearance in P. Let Q be any other path in $\mathcal{P}^{(\mathrm{sh})}$ starting in the same source vertex $s \in V$ and going through (v,t). Then $P[\bullet, (v,t)]$ and $Q[\bullet, (v,t)]$ have the same length.

Proof. Assume for the purpose of contradiction that the two subpaths have a different length. Without loss of generality, assume that $P[\bullet, (v,t)]$ is shorter than $Q[\bullet, (v,t)]$. Then the walk $R = P[\bullet, (v,t)] \oplus Q[(v,t), \bullet]$ is shorter than Q and if it is not a path, then it can easily be transformed into a path R' (by Lemma 4.11).

We can use Corollary 4.13 to conclude that, if duration is among the optimality criteria, then P and Q have the same departure time. That means that R' has its duration (if relevant) and arrival time no worse than Q and, as mentioned before, is shorter than Q, contradicting the optimality of Q. □

We also have the corresponding corollary.

Corollary 4.15. *Let $\mathcal{P}^{(\mathrm{sh})}$ be as in Lemma 4.14. Let $\mathcal{P}' \subseteq \mathcal{P}^{(\mathrm{sh})}$ be an arbitrary subset of $\mathcal{P}^{(\mathrm{sh})}$. Then the property described in Lemma 4.14, i.e., that prefixes of paths starting from the same source and going through the same appearance have the same length, still holds.* □

Note that the condition imposed by the lemmas above, namely that of $\mathcal{P}$ being induced by a criterion function is only sufficient, but not necessary for the lemmas to hold. For example, consider the set of "slowest" paths, i.e., paths with maximal duration. This set is not induced by a criterion function, but, as can easily be checked, both of the lemmas and their corollaries will still hold. At the same time, the lemmas do *not*

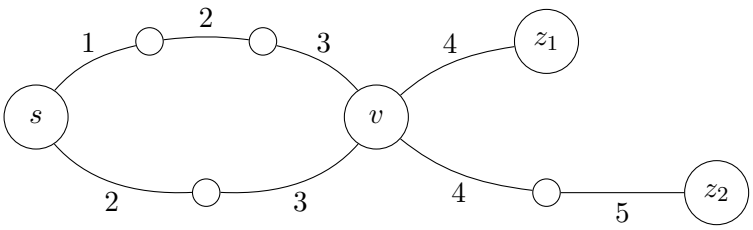


Figure 4.3: Lemmas 4.12 and 4.14 do not hold for arbitrary sets $\mathcal{P}$: the s-v-prefixes of optimal paths in $\mathcal{P}^{(\mathrm{mod}\,2)}$ from s to z_1 and to z_2 have different lengths and departure times.

hold for arbitrary sets $\mathcal{P}$, as might be one's first intuition: for example, consider the set $\mathcal{P}^{(\mathrm{mod}\,2)}$, the set where the only restriction is that for any pair of vertices $s, z \in V$, paths of even length are strictly better than those with an odd length (we can think of those as being shortest paths where their length is measured modulo two). Consider now the graph in Figure 4.3. There is only one optimal path between s and any other vertex. In particular, there is only one optimal path between s and each of z_1 and z_2, call them P and P', respectively, each one going through $(v, 3)$. However, we can see that the paths $P\,[\bullet, (v,3)]$ and $P'\,[\bullet, (v,3)]$ have different lengths and departure times.

Now that we have shown the two lemmas as well as their respective corollaries, we can prove the following theorem.

Theorem 4.1. *Let c be a criterion function such that at least one entry of its output tuple is path length. Then the corresponding induced set $\mathcal{P}^{(\text{sh-gen})}$ of optimal paths with respect to c is prefix-compatible.*

Proof. Let $P \in \mathcal{P}^{(\text{sh-gen})}$ be an arbitrary path in $\mathcal{P}^{(\text{sh-gen})}$. Let (v, t) be an arbitrary vertex appearance in P. Now let $Q \in \mathcal{P}^{(\text{sh-gen})}$ be an arbitrary optimal path starting in the same source vertex s as P and going through (v, t). Our goal is to prove that $R = Q\,[\bullet, (v,t)] \oplus P\,[(v,t), \bullet]$ is a member of $\mathcal{P}^{(\text{sh-gen})}$.

Consider the walk R. There are two possible reasons for why it may not be a an optimal path:

***R* is not a path.** This means that there is some vertex w that is both in $Q\,[\bullet, (v,t)]$ and in $P\,[(v,t), \bullet]$. Let (w, t') be the first such appearance in $Q\,[\bullet, (v,t)]$. We can then say that $Q\,[\bullet, (w,t')] \oplus P\,[(w,t'), \bullet]$ is a valid s-z-path such that, if duration is one of the optimality criteria, has the same departure time as P (by Corollary 4.13) and therefore the same duration. It also has an arrival time at most that of P. However, it is shorter than P (by Corollary 4.15), contradicting its optimality. We can therefore conclude that R is a path.

***R* is not optimal.** We will now assume that R is a path. It arrives at the same time as P. Furthermore, by Corollary 4.15, it has the same length as P. If duration is one of the criteria, then by Corollary 4.13 R has the same departure time as both P, and so also the same duration as P. Hence R is an optimal s-z-path.

We have shown that neither of the two cases is possible and so we can see that R is indeed an optimal path and is in $\mathcal{P}^{(\text{sh-gen})}$. We can now conclude that the set $\mathcal{P}^{(\text{sh-gen})}$ is prefix-compatible. □

Let us now consider solving the $\mathcal{P}^{(\text{sh-gen})}$-Betweenness-Centrality problem. Since we have proved that the set is prefix-compatible, we can use the framework of Theorem 3.1. We need to compute the numbers of unique prefixes of optimal paths through some appearance (v, t). Towards that end, we make the following observations, which follow from Corollaries 4.13 and 4.15, respectively.

Observation 4.16. *If duration is among the optimality criteria, then for all optimal paths $P \in \mathcal{P}^{(\text{sh-gen})}$ starting from some source $s \in V$ and going through some appearance $(v, t) \in V \times \mathcal{T}$, we have that $P[\bullet, (v, t)]$ is a fastest path to that particular appearance among paths in $\mathcal{P}^{(\text{sh-gen})}$.*

Since for $\mathcal{P}^{(\text{sh-gen})}$ we must have length among the optimality criteria by definition, for the next observation the condition in the beginning is unnecessary.

Observation 4.17. *For all optimal paths $P \in \mathcal{P}^{(\text{sh-gen})}$ starting from some source $s \in V$ and going through some appearance $(v, t) \in V \times \mathcal{T}$, we have that $P[\bullet, (v, t)]$ is a shortest path to that particular appearance among paths in $\mathcal{P}^{(\text{sh-gen})}$.*

All paths to an appearance (v, t) arrive at the same time t by definition, so we also trivially have the matching observation for the last relevant optimality criterion.

Observation 4.18. *For all optimal paths $P \in \mathcal{P}^{(\text{sh-gen})}$ starting from some source $s \in V$ and going through some appearance $(v, t) \in V \times \mathcal{T}$, we have that $P[\bullet, (v, t)]$ is a foremost path to that particular appearance among paths in $\mathcal{P}^{(\text{sh-gen})}$.*

Taken together, those observations mean that all prefixes of optimal paths are themselves optimal, with respect to vertex appearances.

Lemma 4.19. *Let $P \in \mathcal{P}^{(\text{sh-gen})}$ be an arbitrary c-optimal path starting from some source $s \in V$. Then, for any $(v, t) \in V \times \mathcal{T}$ that P goes through, $P[\bullet, (v, t)]$ is a c-optimal s-(v, t)-path.* □

We find that the (rough) converse of the lemma also holds.

Lemma 4.20. *Let c be the criterion functionthat induces $\mathcal{P}^{(\text{sh-gen})}$. Let P be any c-optimal s-z-path and $(v, t) \in V \times \mathcal{T}$ any appearance that P goes through. Then any optimal s-(v, t)-path can be extended into an optimal s-z-path.*

Proof. Let Q be any optimal s-(v, t)-path. By Lemma 4.19, $P[\bullet, (v, t)]$ is optimal, hence it has, by definition, the same optimality criteria as Q. Therefore, the only reason why $Q \oplus P[(v, t), \bullet]$ could not be an optimal s-z-path is that it is not a path. However, in that case we can use Lemma 4.11 on $Q \oplus P[(v, t), \bullet]$ to exhibit a path that is shorter than P, with no worse arrival time and duration. That would contradict the optimality of P. Therefore, every optimal s-(v, t)-path can be extended into an optimal s-z-path. □

Lemma 4.19 and Lemma 4.20 taken together imply a crucial property for devising an algorithm using the framework of Theorem 3.1.

Algorithm 4.3 $\mathcal{P}^{(\text{sh-gen})}$ betweenness in temporal graphs.

Input: A temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$, a source $s \in V$, and a criterion function c, with path length as at least one entry of the output tuple.

Output: (Super-) Set of relevant appearances R and $\delta_{sv}^{\mathcal{P}^{(\text{sh-gen})}}(v,t)$, $\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}^{(\text{sh-gen})}}(v,t)$, and $\text{Pre}_{s}^{\mathcal{P}^{(\text{sh-gen})}}(v,t)$ for those appearances.

```
Q ← Initialize(s)
while Q not empty do
    (v,t) ← Q.extract-min()
    for (w,t') ∈ N_G(v,t) do                    ▷ N_G^>(v,t) for strict
        if dist[w,t'] = ∞ then                  ▷ First arrival at (w,t')
            R ← R ∪ {(w,t')}
            Q.enqueue(w,t')
        Relax((v,t),(w,t'))
        Update-Path-Counts((v,t),(w,t'))
return Transform()
```

Corollary 4.21. *For the set $\mathcal{P}^{(\text{sh-gen})}$, for any vertex $s \in V$ and vertex appearance $(v,t) \in V \times \mathcal{T}$, we have $\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}^{(\text{sh-gen})}} = \sigma_{s(v,t)}^{\mathcal{P}^{(\text{sh-gen})}}$.* [3] □

Moreover, we know that prefix-compatibility implies acyclicity of the predecessor graph (see Lemma 3.13). So for each source vertex, we can proceed with finding optimal paths/counts to each appearance in a Dijkstra-like manner, as weighted by the criterion function.

Below we provide the main part of an algorithm implementing our extended Dijkstra approach. Because the full algorithm is quite lengthy and most of the work in all of the auxiliary functions is nothing more than a vast amount of bookkeeping that is not particularly enlightening to see, we defer it along with a more detailed analysis and a formal proof of its correctness to Appendix A.

Be as it may, Algorithm 4.3 allows us to finally use Theorem 3.1 to make statements about computational complexity of all combinations of path optimality concepts (that are not #P-hard).

Theorem 4.2. *Let $\mathcal{P}^{(\text{sh-gen})}$ be an induced path set of a criterion function c that has path length as at least one entry of its output tuple. Then $\mathcal{P}^{(\text{sh-gen})}$-Betweenness-Centrality can be solved in $\mathcal{O}(n^3T^2(\log(n)+\log(T)))$ time.*

Proof. Algorithm 4.3 provides us with the required Count-Prefixes function for the case of $\mathcal{P}^{(\text{sh-gen})}$.

The correctness of Algorithm 4.3 follows from the observations above (for a more detailed proof of correctness, see Appendix A). For the running time, we shall mention

[3] Strictly speaking, this is only true for (v,t) that appear on *some* path in $\mathcal{P}^{(\text{sh-gen})}$ and otherwise $\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}^{(\text{sh-gen})}}$ is equal to zero. However, such appearances cannot be relevant, and the returned value of $\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}^{(\text{sh-gen})}}$ for irrelevant vertices does not matter.

Table 4.2: Tractability of different variants of $\mathcal{P}$-Betweenness-Centrality. For example, the first entry in the third column refers to shortest-foremost betweenness, which is solvable in polynomial time, as shown by Buß et al. [Buß+20]. New results shown in **bold**.

	Shortest	Fastest	Foremost
Shortest	P	**P**	P
Fastest	**P**	#P-hard	#P-hard
Foremost	**P**	#P-hard	#P-hard

that Initialize takes $\mathcal{O}(n + M) = \mathcal{O}(n^2T)$ time while also adding the complete neighborhood $N_{\mathcal{G}}(s)$ to the priority queue Q. Transform runs in $\mathcal{O}(nT)$ time. The other helper functions take constant time and do not otherwise affect the running time.

In the loop starting on Line 4, we see that the size of a temporal neighborhood of a vertex appearance is upper bounded by $\mathcal{O}(nT)$. Furthermore, every vertex appearance is added to the queue at most once (on Line 7). Therefore, the total amount of time for the while loop of Line 2 is upper bounded by the maximum size of a temporal neighborhood of a vertex appearance times the number of possible vertex appearances, times a logarithmic factor for the queue operations, that is, $\mathcal{O}(nT \cdot nT \log(nT)) = \mathcal{O}(n^2T^2(\log(n) + \log(T)))$.

Since, by Theorem 4.1, $\mathcal{P}^{(\text{sh-gen})}$ is prefix-compatible, we can now use Theorem 3.1 with $f(n, M, T) = \mathcal{O}(n^2T^2(\log(n) + \log(T)))$ to conclude that $\mathcal{P}^{(\text{sh-gen})}$-Betweenness-Centrality can be solved in $\mathcal{O}(n^3T^2(\log(n) + \log(T)))$ time. □

4.4 Summary

In this chapter, we have used our framework of Theorem 3.1 to solve the problem of computing betweenness centrality for a variety of practically-motivated path optimality concepts. We started with a simple warm-up example in Section 4.1, before moving to Section 4.2, where we have shown connections between this and previous work.

Finally, in Section 4.3 we described a way of combining the canonical path optimality concepts and then used the framework to show that all combinations of two optimality criteria that had not been studied before yield a betweenness measure that is computable in polynomial time, see Table 4.2. Moreover, we have also shown that all possible combinations of the three criteria which are not redundant, i.e., those that yield a measure different from the ones in Table 4.2, also lead to $\mathcal{P}$-Betweenness-Centrality problem that is solvable in polynomial time.

Remarkably, while we have only considered criterion functions that are based on the three canonical measures of temporal paths, we could generalize the results of Section 4.3 to some other sets which have the necessary properties. For example, if we wanted to consider slowest paths (i.e., paths of maximal duration), then we could easily show that the corresponding results also hold. Therefore, we could solve $\mathcal{P}$-Betweenness-Centrality with respect to, e.g., shortest-slowest paths in polynomial time. It remains an open question whether a simple property exists which would allow us to give a necessary and sufficient condition on path sets for which the results from Section 4.3 to hold. Such a property would allow a more streamlined approach to computing betweenness

centrality using more new measures derived from real-world scenarios.

In his paper, Brandes [Bra01] also cites a few related measures of centrality—stress centrality [Shi53], graph centrality [HH95], and closeness centrality [Sab66] that can easily be computed with only a slightly modification to his algorithm. Notably, just as in the case of Brandes' algorithm, our algorithm from Section 4.3 can also be easily tweaked to compute those other similar centrality measures.

Chapter 5

Conclusion

We have analyzed the problem of computing betweenness centrality in the temporal setting. We generalized the insights seen in the previous work on the problem to devise a uniform approach for computing the betweenness centrality in this general setting. More specifically, we have first defined betweenness centrality with respect to an arbitrary set of paths in a temporal graph. Then, we have presented a property of a path set—prefix-compatibility—that allows us to simplify the process of computation of betweenness centrality. Notably, while the condition is a sufficient one for our results to hold, it is not known at the time whether said condition is also necessary. It remains an open problem to determine whether that is the case and if not, to find a more general condition that is both necessary and sufficient.

It is known that for some variants of betweenness centrality, like fastest or foremost (or even the unrestricted variant, where every possible path needs to be considered), the problem of computing the measure is #P-hard [Buß+20]. A question for further research could be to see if the theoretical framework presented in this work could be used to tackle those computationally intractable variants. For example, one could try and develop parameterized algorithms for the problems, with the parameter being some notion of distance of the path set to prefix-compatibility. Similarly, it might be of interest to try and use the framework to devise approaches leading to approximate computation of betweenness centrality.

In the second part of this work, we have shown practical uses of the framework, motivated by some real-world scenarios. First, in Section 4.1, we have shown how to compute the betweenness centrality in a scenario where only some vertices in the network are important to us. In Section 4.2, we have shown how our framework is connected to previous research in the area. Then, in Section 4.3 we have applied our framework to solve multiple variants of betweenness at the same time (twenty-two, if we count the strict and non-strict variants as separate). Some of the variants had already been studied in literature, but to the best of our knowledge most have first been analyzed here. It is worth to mention that for the problems for which a solution had been known, the previously discovered algorithm runs in $\mathcal{O}(n^3T^2)$ time, in contrast to $\mathcal{O}(n^3T^2(\log(n) + \log(T)))$ of this work. While the former appears to be algorithmically tight with respect to our framework, it remains an open problem to determine if the additional logarithmic factor can be removed in the general case shown here.

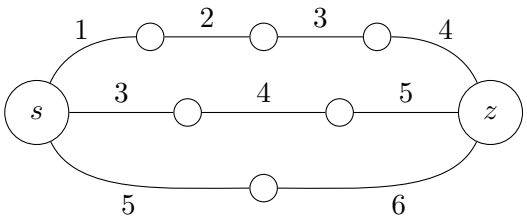


Figure 5.1: A graph showing how Pareto-optimal paths with respect to arrival time and length are not simply the same as the union of shortest-foremost and foremost-shortest paths. The only foremost (and so also the only foremost-shortest) s-z-path is the top one, while the only shortest path is the bottom one. However, the middle path is also Pareto-optimal: in comparison to the middle path, each of the top and bottom path has one of arrival time or length that is worse than the respective value of the middle path.

Studying other path optimality concepts and their related betweenness remains a natural avenue for further research. For example, one could try and compute betweenness centrality with respect to the set of paths of bounded waiting time, i.e., paths such that the difference between adjacent temporal edges is either at least Δ or at most Δ, for some constant Δ. Both problems are likely to be hard: the former problem is likely to be reducible to the unrestricted betweenness (since we can stretch the timespan of the graph by a factor of Δ), while for the latter, the problem of even *finding* such paths has recently been shown to be NP-hard [Cas+20]. It is interesting to see what other real-world-motivated concepts of path optimality might emerge and how the framework developed here might apply to them.

Similarly, and perhaps more in the spirit of Section 4.3, we can also consider different ways of combining path optimality measures. Instead of criterion functions outputting tuples to be ordered lexicographically, we could consider criterion functions that output an arbitrary convex combination of their arguments, effectively allowing us to have a continuous balance between the optimality concepts. We conjecture that for any such variant where the weight on path length is some $\epsilon > 0$, the corresponding betweenness measure can be computed efficiently. In fact, while a careful investigation of the topic is outside the scope of this work, it is possible that Algorithm 4.3 would work for the problem in its exact form, the only difference being in the criterion function outputting a real number instead of a tuple.

Yet another way of combining optimality concepts that could prove of interest is that of finding Pareto-optimal paths. In Pareto-optimal setting we would also consider criterion functions, exactly like those already seen in Section 4.3. The only difference being, instead of comparing the output tuples lexicographically, for tuples a and b, we only have $a < b$ if a is "strictly better" than b. More formally, we have $a < b$ only if *all* entries in a are less than or equal to the corresponding ones in b and *at least one* of the entries in a is *strictly less than* the corresponding entry in b. A path is then Pareto-optimal, if it has a minimal value according to the ordering described above. For a short example showing how Pareto-optimal paths differ from the lexicographic ordering case, consult Figure 5.1.

The last part, i.e., Section 4.3 also spawned another type of questions for further

work. An intuitive summary of the results from the section is that introducing shortness into the optimality criteria yields a tractable betweenness centrality measure. However, our results have only been proven with an approach that effectively amounted to brute force. That being said, we have also presented an example showing that the results do not hold in all generality—where introducing shortness into the optimality criteria does not simplify the situation (enough). Hence it remains an interesting question whether there exists a simple necessary and/or sufficient condition on a path set for which introducing shortness will lead to a simple betweenness centrality measure.

Literature

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009 (cit. on p. 21).

[AGW15] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. *Subcubic Equivalences Between Graph Centrality Problems, APSP and Diameter*. In: *Proceedings of the 2015 Annual ACM-SIAM Symposium on Discrete Algorithms*. 2015, pp. 1681–1697 (cit. on p. 12).

[Alc+18] José Carlos R. Alcantud, María J. Campión, Juan C. Candeal, Raquel G. Catalán, and Esteban Induráin. *On the Structure of Acyclic Binary Relations*. In: *Communications in Computer and Information Science*. Springer International Publishing, 2018, pp. 3–15 (cit. on p. 15).

[Ant71] J.M. Anthonisse. *The Rush in a Directed Graph*. (Unpublished notes). Jan. 1971 (cit. on p. 12).

[ARFG17] Amir Afrasiabi Rad, Paola Flocchini, and Joanne Gaudet. *Computation and analysis of temporal betweenness in a knowledge mobilization network*. In: *Computational Social Networks* 4.1 (2017), p. 5 (cit. on pp. 10, 13).

[Bad+07] David A Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail. *Approximating betweenness centrality*. In: *Proceedings of the 2007 International Workshop on Algorithms and Models for the Web-Graph*. Springer. 2007, pp. 124–137 (cit. on p. 12).

[Bar+16] Albert-László Barabási et al. *Network Science*. Cambridge University Press, 2016 (cit. on pp. 9, 12).

[BM15] Elisabetta Bergamini and Henning Meyerhenke. *Fully-Dynamic Approximation of Betweenness Centrality*. Springer Berlin Heidelberg, 2015, pp. 155–166 (cit. on p. 12).

[Bra01] Ulrik Brandes. *A faster algorithm for betweenness centrality*. In: *The Journal of Mathematical Sociology* 25.2 (2001), pp. 163–177 (cit. on pp. 12–14, 23, 28, 29, 31, 36, 51).

[Buß+20] Sebastian Buß, Hendrik Molter, Rolf Niedermeier, and Maciej Rymar. *Algorithmic Aspects of Temporal Betweenness*. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, 2020, 2084–2092 (cit. on pp. 10, 13, 14, 23, 26, 29, 31, 35, 37–41, 43, 45, 50, 53).

[Cas+20] Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. *The Computational Complexity of Finding Temporal Paths under Waiting Time Constraints*. In: *Proceedings of the 31st International Symposium on Algorithms and Computation (ISAAC '20)*. LIPIcs. Accepted for publication. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020 (cit. on pp. 12, 13, 41, 54).

[Cor+09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009 (cit. on p. 62).

[Die17] Reinhard Diestel. *Graph Theory*. 5th. Springer Publishing Company, Incorporated, 2017 (cit. on p. 15).

[Eul41] Leonhard Euler. *Solutio problematis ad geometriam situs pertinentis*. In: *Commentarii academiae scientiarum Petropolitanae* (1741), pp. 128–140 (cit. on p. 9).

[Fre77] Linton C. Freeman. *A Set of Measures of Centrality Based on Betweenness*. In: *Sociometry* 40.1 (1977), pp. 35–41 (cit. on pp. 10, 12).

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. 2nd. Addison-Wesley Longman Publishing Co., Inc., 1994 (cit. on p. 15).

[HH95] Per Hage and Frank Harary. *Eccentricity and centrality in networks*. In: *Social Networks* 17.1 (1995), pp. 57–63 (cit. on p. 51).

[Him+19] Anne-Sophie Himmel, Matthias Bentert, André Nichterlein, and Rolf Niedermeier. *Efficient Computation of Optimal Temporal Walks under Waiting-Time Constraints*. In: *Proceedings of the 8th International Conference on Complex Networks and their Applications*. Vol. 882. SCI. Springer, 2019, pp. 494–506 (cit. on pp. 13, 41).

[Hol15] Petter Holme. *Modern temporal network theory: a colloquium*. In: *The European Physical Journal B* 88.9 (2015) (cit. on pp. 13, 41).

[HS12] Petter Holme and Jari Saramäki. *Temporal networks*. In: *Physics Reports* 519.3 (2012), 97–125 (cit. on pp. 9, 13, 16, 41).

[JSK18] F. Jamour, S. Skiadopoulos, and P. Kalnis. *Parallel Algorithm for Incremental Betweenness Centrality on Large Graphs*. In: *IEEE Transactions on Parallel and Distributed Systems* 29.3 (2018), pp. 659–672 (cit. on p. 12).

[KCC14] Miray Kas, Kathleen M. Carley, and L. Richard Carley. *An incremental algorithm for updating betweenness centrality and k-betweenness centrality and its performance on realistic dynamic social network data*. In: *Social Network Analysis and Mining* 4.1 (2014), p. 235 (cit. on p. 12).

[Kos09] Vassilis Kostakos. *Temporal graphs*. In: *Physica A: Statistical Mechanics and its Applications* 388.6 (2009), pp. 1007–1023 (cit. on p. 9).

[LVM18] Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. *Stream graphs and link streams for the modeling of interactions over time*. In: *Social Network Analysis and Mining* 8.1 (2018), p. 61 (cit. on pp. 9, 13, 16).

[Mic16] Othon Michail. *An Introduction to Temporal Graphs: An Algorithmic Perspective*. In: *Internet Mathematics* 12.4 (2016), pp. 239–280 (cit. on p. 9).

[PR15] Matteo Pontecorvi and Vijaya Ramachandran. *A Faster Algorithm for Fully Dynamic Betweenness Centrality*. In: (2015) (cit. on p. 12).

[RU16] Matteo Riondato and Eli Upfal. *ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages*. In: (2016) (cit. on p. 12).

[Sab66] Gert Sabidussi. *The centrality index of a graph*. In: *Psychometrika* 31.4 (1966), pp. 581–603 (cit. on p. 51).

[Sco88] John Scott. *Social network analysis*. In: *Sociology* 22.1 (1988), pp. 109–127 (cit. on pp. 9, 12).

[Shi53] Alfonso Shimbel. *Structural parameters of communication networks*. In: *The Bulletin of Mathematical Biophysics* 15.4 (1953), pp. 501–507 (cit. on p. 51).

[Tan+10] John Tang, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Vincenzo Nicosia. *Analysing Information Flows and Key Mediators through Temporal Centrality Metrics*. In: *Proceedings of the 3rd Workshop on Social Network Systems*. SNS '10. Paris, France: Association for Computing Machinery, 2010 (cit. on pp. 10, 13).

[Tsa+20] Ioanna Tsalouchidou, Ricardo Baeza-Yates, Francesco Bonchi, Kewen Liao, and Timos Sellis. *Temporal betweenness centrality in dynamic graphs*. In: *International Journal of Data Science and Analytics* 9.3 (2020), pp. 257–272 (cit. on pp. 10, 13).

[Val79a] Leslie G. Valiant. *The Complexity of Enumeration and Reliability Problems*. In: *SIAM Journal on Computing* 8.3 (1979), pp. 410–421 (cit. on p. 21).

[Val79b] L.G. Valiant. *The complexity of computing the permanent*. In: *Theoretical Computer Science* 8.2 (1979), pp. 189–201 (cit. on p. 20).

[WB94] Douglas R. White and Stephen P. Borgatti. *Betweenness centrality measures for directed graphs*. In: *Social Networks* 16.4 (1994), pp. 335–346 (cit. on p. 12).

[WF94] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*. Cambridge University Press, 1994 (cit. on pp. 9, 12).

[WM16] Matthew J. Williams and Mirco Musolesi. *Spatio-temporal networks: reachability, centrality and robustness*. In: *Royal Society Open Science* 3.6 (2016) (cit. on pp. 10, 13).

[Wu+14] Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. *Path problems in temporal graphs*. In: *Proceedings of the VLDB Endowment* 7.9 (2014), pp. 721–732 (cit. on pp. 13, 41).

[Wu+16] Huanhuan Wu, James Cheng, Yiping Ke, Silu Huang, Yuzhen Huang, and Hejun Wu. *Efficient Algorithms for Temporal Path Computation*. In: *IEEE Transactions on Knowledge and Data Engineering* 28.11 (2016), pp. 2927–2942 (cit. on pp. 39, 41).

Appendix A

Section 4.3—full algorithm

In this appendix we provide the full version of the algorithm from Section 4.3. We reproduce the main part of the algorithm, as seen in Section 4.3, as well as supply the definitions of the auxiliary functions used therein. Our goal is to give a short overview of the full algorithm and then provide a formal proof of its correctness. The running time analysis has already been provided in Section 4.3, in Theorem 4.2.

The algorithm is pictured below, see Algorithm A.1. For convenience, we assume that the input graph $\mathcal{G}$ and the criterion function c, as well as the arrays P, σ, dist, dur, $t_{\min}$, δ are all global variables, hence we do not need to constantly pass all of them around functions. Also note that σ, dist, and dur are all overloaded: if the array is accessed with two indices, it is a different one than if accessed with only one index. However, they both represent the same concept. For example, $\sigma[v, t]$ represents the number of optimal paths to the appearance (v, t), whereas $\sigma[v]$ represents the number of optimal paths to v. Any invocations of the criterion function c are used with the current values stored in the appropriate dist and dur array (and the $t_{\min}$ array if invoked on a vertex instead of a vertex appearance).

The algorithm performs a Dijkstra-like search through the graph in order to find optimal paths to each reachable vertex appearance as well as compute the necessary auxiliary data. The main logic of the algorithm is a simple priority-queue-based walk through the graph, while the helper functions initialize all the necessary data structures, transform the data into the values expected by Theorem 3.1, as well as do the necessary bookkeeping.

The function Initialize initializes all data structures to their default values; initially all vertices (and vertex appearances) apart from the source s are considered unreachable and the respective entries in the appropriate arrays are initalized in lines 2-8. The set R, initialized on Line 9, will store the set of vertices reachable from s.

It is important to note that we also need to do the first round of search in the initialization function and not in the main loop of Algorithm A.1, since we have to treat $(s, 0)$ differently to all appearances that come thereafter. That is because of the duration: in the typical case, if we can get to the appearance (v, t) with duration d and then make the transition $v \xrightarrow{t'} w$, then the resulting path to w will have a duration of $d + (t' - t)$. However, when dealing with the source, it is not the case: we associate the appearance $(s, 0)$ with it, but by definition of path duration we can get to any

Algorithm A.1 $\mathcal{P}^{(\text{sh-gen})}$ betweenness in temporal graphs.

Input: A temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$, a source $s \in V$, and a criterion function c, with path length as at least one entry of the output tuple.

Output: (Super-) Set of relevant appearances R and $\delta_{sv}^{\mathcal{P}^{(\text{sh-gen})}}(v,t)$, $\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}^{(\text{sh-gen})}}(v,t)$, and $\text{Pre}_s^{\mathcal{P}^{(\text{sh-gen})}}(v,t)$ for those appearances.

```
1:  Q ← Initialize(s)
2:  while Q not empty do
3:      (v,t) ← Q.extract-min()
4:      for (w,t') ∈ N_G(v,t) do                    ▷ N_G^>(v,t) for strict
5:          if dist[w,t'] = ∞ then                  ▷ First arrival at (w,t')
6:              R ← R ∪ {(w,t')}
7:              Q.enqueue(w,t')
8:          Relax((v,t),(w,t'))
9:          Update-Path-Counts((v,t),(w,t'))
10: return Transform()
```

neighbor (v,t) of the source in duration zero, and not $0 + (t - 0) = t$. However, after that point, the departure times of all possible paths are set in stone and so all other appearances can be handled in the main loop.

The function Relax, just like in standard SSSP problems, relaxes the transition from (v,t) to (w,t'): it first checks if a better path to (w,t') has been found. If so, it updates the auxiliary data structures storing the parameters of optimal paths to (w,t') and then resets the data about predecessors and path counts. Additionally, we also need to keep track of paths to the vertex w as a whole, which we do on Line 7.

The function Update-Path-Counts is responsible for the majority of bookkeeping. It checks whether the currently analyzed transition yields an optimal path to the vertex appearance (w,t') (and possibly to the vertex w as a whole, as well). If so, then it updates the necessary data structures.

Finally, Transform simply prepares the return values to fit those expected in Theorem 3.1. Note that this function is mostly there for making the presentation a bit clearer and an analysis of the algorithm a bit easier; we could just as easily have returned the data structures directly.

We shall now prove the algorithm's correctness. Broadly speaking, we can partition the problem into two parts: first, we show that the algorithm correctly finds optimal paths. Second, we show that the path counts and predecessor sets returned by the algorithm are correct.

The proof of the former will broadly follow that of the standard Dijkstra algorithm (for example, see Cormen et al. [Cor+09]). We start by making a few simple observations. First, the algorithm does a (prioritized) breadth-first-search, so it puts no restrictions on visited appearances. Hence, we get the following observation:

Observation A.1. *A vertex apperance will be added to the queue by Algorithm A.1 if and only if it is reachable from the source vertex $s \in V$.*

Now, in the following analysis of the algorithm, we denote by $c(v)$ or $c(v,t)$ the

Algorithm A.2 Initialization function for Algorithm A.1.

```
1:  function INITIALIZE(s)
2:      for v ∈ V do                                   ▷ Optimal values for each vertex
3:          dist[v] ← ∞; dur[v] ← ∞; t_min[v] ← ∞; σ[v] ← 0
4:      for (v, t) ∈ V × T do                          ▷ Optimal values for each vertex appearance
5:          σ[v, t] ← 0; P[v, t] ← ∅
6:          dist[v, t] ← ∞; dur[v, t] ← ∞
7:      dist[s] ← 0; dist[s, 0] ← 0; t_min[s] ← 0      ▷ s gets a special appearance
8:      σ[s] ← 1; σ[s, 0] ← 1
9:      R ← {(s, 0)}                                   ▷ Relevant vertex appearances
10:     Q ← empty min-priority queue of vertex apperances, with c as the key
11:     ▷ Duration means the successors of s are a special case, handle them separately
12:     for (w, t′) ∈ N_G(s) do                        ▷ Always first discovery of (w, t′)
13:         dist[w, t′] ← 1; dist[w] ← 1
14:         dur[w, t′] ← 0; dur[w] ← 0
15:         σ[w, t′] ← 1; σ[w] ← σ[w] + 1
16:         P[w, t′] ← {(s, 0)}
17:         if t′ < t_min[w] then
18:             t_min[w] ← t′
19:         R ← R ∪ {(w, t′)}
20:         Q.enqueue(w, t′)
21:     return Q
```

values of the criterion function with the appropriate entries of dist, dur, $t_{\min}$ fed into the function. To refer to the optimal values, we make the following definition.

Definition A.2. Let $\mathcal{G}$ be a temporal graph and c a criterion function. Fix a source $s \in V$. Then for every vertex $v \in V$ (vertex appearance (v,t)) we denote by $c^*(v)$ (respectively, $c^*(v,t)$) the optimal value of criterion function. That is, for every s-v-path P (respectively, every s-(v,t)-path P') we have $c^*(v) \le c(P)$ (respectively, $c^*(v,t) \le c(P')$).

We have the following simple property of c^*:

Lemma A.3. *Let (w,t') be an arbitrary appearance and let $(v,t) \in \mathrm{Pre}_s^{\mathcal{P}^{(\text{sh-gen})}}(w,t')$ be a direct predecessor. Then $c^*(v,t) < c^*(w,t')$.*

Proof. Let $P \in \mathcal{P}^{(\text{sh-gen})}$ be an arbitrary optimal path making the transition $(v,t) \xrightarrow{t'} (w,t')$. (Such a path must exist since $(v,t) \in \mathrm{Pre}_s^{\mathcal{P}^{(\text{sh-gen})}}(w,t')$.) Then we have $t \le t'$, so $P[\bullet,(v,t)]$ has duration and arrival time no more than that of $P[\bullet,(w,t')]$, but is shorter by one edge. Since path length is one of the criteria in c by definition of $\mathcal{P}^{(\text{sh-gen})}$, we must have $c^*(v,t) < c^*(w,t')$. □

Applying the lemma inductively yields the following:

Corollary A.4. *Let (w,t') be an arbitrary vertex appearance and let $(v,t) \neq (w,t')$ be an arbitrary predecessor of (w,t') on some path $P \in \mathcal{P}^{(\text{sh-gen})}$. Then $c^*(v,t) < c^*(w,t')$.* □

Algorithm A.3 Function relaxing the transition from (v, t) to (w, t'), under the criteria in c.

```
1:  function RELAX((v, t), (w, t'))
2:      if c(dist[v, t] + 1, dur[v, t] + (t' − t), t') < c(w, t') then    ▷ Better path to (w, t')
3:          dist[w, t'] ← dist[v, t] + 1
4:          dur[w, t'] ← dur[v, t] + (t' − t)
5:          σ[w, t'] = 0
6:          P[w, t'] ← ∅
7:      if c(w, t') < c(w) then                                            ▷ Better path to w as a whole
8:          dist[w] ← dist[w, t']
9:          dur[w] ← dur[w, t']
10:         σ[w] ← 0
11:         t_min[w] ← t'                                                  ▷ t' must be foremost among optimal paths
```

Algorithm A.4 Function doing the bookkeeping required for maintaining the path count and predecessor arrays up-to-date.

```
1:  function UPDATE-PATH-COUNTS((v, t), (w, t'))
2:      if c(dist[v, t] + 1, dur[v, t] + (t' − t), t') = c(w, t') then
3:          σ[w, t'] ← σ[w, t'] + σ[v, t]
4:          P[w, t'] ← P[w, t'] ∪ {(v, t)}
5:          if c(w, t') = c(w) then                                        ▷ Also found an optimal path to w
6:              σ[w] ← σ[w] + σ[v, t]
7:              t_min[w] ← min(t_min[w], t')                               ▷ t' foremost among the optimal paths
```

One more important property of the algorithm is that for any vertex appearance (v, t), the value $c(v, t)$ is only affected in the function RELAX and in fact, the value is only ever decreased.

Observation A.5. *Let (v, t) be any vertex appearance and $d = c(v, t)$ be the value of criterion function at some iteration of the main loop of Algorithm A.1 or before the call to* INITIALIZE. *Then at any later iteration of the loop, we have $c(v, t) \leq d$. Since in the beginning we have $c(v, t) = \infty$, this means that c is monotonically non-increasing with each iteration.*

Towards a formal proof of Algorithm A.1, we show that the following invariant holds:

Lemma A.6. *At any iteration of the main loop of Algorithm A.1, let S be the set of vertex appearances that have been processed (that is, $(s, 0)$ and those that have been removed from the queue). Then for every vertex appearance $(v, t) \in S$ we have $c(v, t) = c^*(v, t)$.*

Proof. We wish to show that on each iteration of the loop of Line 2, for the vertex appearance (v, t) added to S, we have $c(v, t) = c^*(v, t)$. For the purpose of contradiction, assume that there exists a vertex appearance without this property. Let (w, t') be the first such appearance that is added to S. Since for $(s, 0)$ we initialize the trivially optimal values in Line 7 of INITIALIZE, we cannot have $(w, t') = (s, 0)$. Similarly, (w, t')

Algorithm A.5 Translation function to the form of Theorem 3.1.

```
1: function TRANSFORM
2:     for (v, t) ∈ R do
3:         if c(v, t) = c(v) then
4:             δ^{P^(sh-gen)}_{sv}(v, t) ← σ[v,t] / σ[v]
5:     σ̃^{P^(sh-gen)}_{s(v,t)} ← σ
6:     Pre^{P^(sh-gen)}_s ← P
7:     return R, δ^{P^(sh-gen)}_{sv}(v, t), σ̃^{P^(sh-gen)}_{s(v,t)}, Pre^{P^(sh-gen)}_s
```

cannot be any of s's direct neighbors since for them, the optimal path is that of duration zero and length one, which are the values assigned on Line 12 of INITIALIZE. (And by Observation A.5, those values never change thereafter.)

Now, since all appearances added to S are reachable (by Observation A.1), there must exist an optimal s-(w, t')-path P. Let (v, t) be the first appearance on P that is not an element of S. Therefore, we partition P into $P[\bullet, (v, t)]$ and $P[(v, t), \bullet]$ (with the latter possibly having no edges, if $(v, t) = (w, t')$).

We claim that $c(v, t) = c^*(v, t)$ when (w, t') is added to S. Why? Let (u, t'') be the predecessor of (v, t) on P. (Since $(v, t) \notin S$, such a predecessor must exist). As (v, t) is the *first* appearance that lies outside S, we must have $(u, t'') \in S$. Furthermore, by Lemma 4.19 and Lemma 4.20 we see that an optimal path to (v, t) can be found through (u, t''). Since (w, t') is the first appearance violating our invariant, we must also have that $c(u, t'') = c^*(u, t'')$. Moreover, we know that (u, t'') has been processed and, in particular, that the transition $u \xrightarrow{t} v$ has been relaxed, implying, together with Observation A.5, that $c(v, t) = c^*(v, t)$.

Since we have $c(v, t) = c^*(v, t)$ and since (v, t) is a predecessor of (w, t') (so we can use Corollary A.4), we can derive the following inequality chain:

$$\begin{aligned} c(v,t) &= c^*(v,t) \\ &\overset{\textit{Corollary A.4}}{\leq} c^*(w,t') \\ &\overset{\textit{Observation A.5}}{\leq} c(w,t'). \end{aligned}$$

Now, because c is the key in our priority queue and (w, t') is being processed before (v, t), we must also have $c(w, t') \leq c(v, t)$ and so all the inequalities must, in fact, be equalities, yielding $c(v, t) = c^*(v, t) = c^*(w, t') = c(w, t')$. However, $c^*(w, t') = c(w, t')$ contradicts our choice of (w, t'). □

Thus we have effectively completed the first part of our task, namely showing that Algorithm A.1 correctly finds optimal paths. Using the invariant above, we can show another important property of the algorithm.

Corollary A.7. *Let (w, t') be an arbitrary vertex appearance. Then all its direct predecessors $(v, t) \in \mathrm{Pre}^{\mathcal{P}}_s(w, t')$ are processed by Algorithm A.1 before (w, t') is itself processed.*

Proof. Using Lemma A.3 as well as Corollary A.4 and Observation A.5, for any $(v,t) \in \mathrm{Pre}_s^{\mathcal{P}}(w,t')$, we have $c^*(v,t) < c^*(w,t') \leq c(w,t')$ at any point in the algorithm. However, Lemma A.6 implies that at the time of processing of (v,t), we have $c(v,t) = c^*(v,t)$. This then implies $c(v,t) < c(w,t')$ at the time of processing, and so (v,t) is processed before (w,t'). □

We are now in the final stretch. Only one major proof remains. Before we start, however, we need to make the following simple observation.

Observation A.8. *For any vertex appearance $(w,t') \neq (s,0)$ we have*

$$\sigma_{s(w,t')}^{\mathcal{P}^{(\text{sh-gen})}} = \sum_{(v,t)\in \mathrm{Pre}_s^{\mathcal{P}}(w,t')} \sigma_{s(v,t)}^{\mathcal{P}^{(\text{sh-gen})}}$$

Armed with all the previous results, we embark on our last journey.

Lemma A.9. *Algorithm A.1 correctly computes the number of c-optimal paths to and the predecessor set of each appearance (w,t').*

Proof. If (w,t') is unreachable, then the statement is trivially correct. Otherwise, we prove the statement by an inductive argument. Consider the following invariant: at each iteration of the outermost loop of Algorithm A.1, let S be the set of processed vertices. Then for every $(w,t') \in S$, the optimal path count and the predecessor set is correct.

Initialization: The statement is obviously correct for $(s,0)$ and all its neighbors.

Maintenance: We prove that at the time (w,t') is removed from the queue, it has the correct value of path count and the correct predecessor set. Let $(v,t) \in \mathrm{Pre}_s^{\mathcal{P}^{(\text{sh-gen})}}(w,t')$ be the first direct predecessor of (w,t') that gets processed. By Corollary A.7, it gets processed before (w,t'). By Lemma A.6, at the time of processing of (v,t), we have $c(v,t) = c^*(v,t)$. This implies that at exactly this iteration of the main loop of the algorithm, Relax will find a path to (w,t') with the optimal cost $c^*(w,t')$. At this point it will reset (w,t')'s path count and predecessor set for the last time (since a path P with $c(P) < c^*(w,t')$ will never be found). Afterwards, Update-Path-Counts will add (v,t) to the list of predecessors of (w,t') and set the path count to the number of paths to (v,t).

From that point on, Relax will not further affect the path count and predecessor set of (w,t'). However, by Corollary A.7 we know that every other predecessor $(v',t'') \in \mathrm{Pre}_s^{\mathcal{P}^{(\text{sh-gen})}}(w,t')$ will be processed before (w,t'). From Lemma A.6 it follows that, just as for (v,t), the optimal paths going through (v',t'') will be found. Therefore, Update-Path-Counts will similarly add (v,t'') to the appropriate predecessor list and add the number of paths to (v,t'') to the number of paths to (w,t'). Therefore, by Observation A.8, the number of paths to (w,t') will be correct by the time it gets processed. Those values will then be similarly used for updating the successors of (w,t'), if any are found.

Termination: At termination we have $Q = \emptyset$ which, by Observation A.1, implies that S is the set of all reachable vertices. □

Now, since we obviously have $\sigma_{sv}^{\mathcal{P}^{(\text{sh-gen})}} = \sum_{t \in \mathcal{T}} [c^*(v,t) = c^*(v)]_{\mathbb{1}} \, \sigma_{s(v,t)}^{\mathcal{P}^{(\text{sh-gen})}}$ and the functions Relax as well as Update-Path-Counts obviously handle the appropriate arrays so that they compute the desired sum above, we also get the corresponding lemma about vertices as a whole.

Lemma A.10. *Algorithm A.1 correctly computes the number of optimal paths to each vertex $v \in V$.* □

We can finally complete our proof.

Theorem A.1. *Algorithm A.1 is correct.*

Proof. The function Transform prepares the returns values of Algorithm A.1. We shall consider their validity in turn.

First, by Observation A.1, at the end of the algorithm, R is equal to the set of all reachable appearances. Since an unreachable appearance is certainly not relevant, R definitely constitutes a superset of relevant appearances.

Second, we have $\delta_{sv}^{\mathcal{P}^{(\text{sh-gen})}}(v,t) = \frac{\sigma_{sv}^{\mathcal{P}^{(\text{sh-gen})}}(v,t)}{\sigma_{sv}^{\mathcal{P}^{(\text{sh-gen})}}}$ (see Definition 3.8). By Lemmas A.9 and A.10 we know that Algorithm A.1 computes the correct values of $\sigma_{sv}^{\mathcal{P}^{(\text{sh-gen})}}(v,t)$ and $\sigma_{sv}^{\mathcal{P}^{(\text{sh-gen})}}$. Therefore, we see that Line 2 of Transform correctly computes the appearance dependencies of all appearances in R.

Third, since our algorithm correctly computes the values of $\sigma_{sv}^{\mathcal{P}^{(\text{sh-gen})}}(v,t)$ and $\sigma_{sv}^{\mathcal{P}^{(\text{sh-gen})}}$, we have by Corollary 4.21 that it returns the correct values $\widetilde{\sigma}_{s(v,t)}^{\mathcal{P}^{(\text{sh-gen})}}$ for any appearance (v,t).

Finally, the fact that our algorithm correctly computes the predecessor sets has already been established by Lemma A.9. □