Paths and connectivity in temporal graphs

Andrea Marino
Ana Silva

34º Colóquio Brasileiro de Matemática

impa
Instituto de Matemática Pura e Aplicada

Paths and connectivity in temporal graphs

Paths and connectivity in temporal graphs
Primeira impressão, setembro de 2023
Copyright © 2023 Andrea Marino e Ana Silva.
Publicado no Brasil / Published in Brazil.

ISBN 978-85-244-0541-9 (print)
ISBN 978-85-244-0536-5 (ebook)
MSC (2020) Primary: 05C40, Secondary: 05C38, 68R10, 68Q17, 68Q27

Coordenação Geral Carolina Araujo

Produção Books in Bytes **Capa** IMPA

Realização da Editora do IMPA
IMPA
Estrada Dona Castorina, 110
Jardim Botânico
22460-320 Rio de Janeiro RJ

`www.impa.br`
`editora@impa.br`

```
M339p     Marino, Andrea
              Paths and connectivity in temporal graphs / Andrea Marino e Ana
          Silva. - 1.ed. -- Rio de Janeiro: IMPA, 2023.
              34 Colóquio Brasileiro de Matemática; v. 5, 78p.: il.; 23cm
              ISBN 978-85-244-0541-9 (print)
              ISBN 978-85-244-0536-5 (ebook)
              1. Temporal Graphs. 2. Connectivity. I. Silva, Ana. II. Série.
          III. Título
              UDC: 519.1
                                          Carolina Celano Lima/CRB-7: 2438
```

Preface

As tool to model practical problems, graphs are increasingly prominent. Thanks to its simplicity, they capture relationships between objects in an easily comprehensible manner, in addition to providing a rich model of the underlying structure. This becomes particularly relevant in a world where an enormous volume of data is generated by the minute and where companies become increasingly interested in collecting and analyzing this data. Thus, the use of graphs for the representation, visualization and study of practical problems have become common practice in many applications, such as web searches through search engine, route planning, etc.

Such a wide usage of graphs (also called networks depending on the application) is not surprising as *graphs* occur everywhere, representing a wide variety of relationships between objects in different applications, from biology to archaeology, from physics to sociology, from chemistry to politics. In this scenario, graph nodes represent entities and graph edges represent relationships between these entities. For example, in the case of social networks, the nodes (called vertices) are users and their relationships (called edges) correspond to friendships. In the case of public transport networks, the vertices are the stops and the edges are the transport lines connecting these stops. In the case of frequency assignment, vertices are antennas and edges represent physical proximity, which indicates possible interference among the antennas. These are just a few among numerous real-life situations that can be modeled as a graph.

Additionally, many of the practical scenarios mentioned above have a time component related to it, capturing the dynamic nature of the connections between the involved entities. For instance, in a public transportation network, one wishes to know not only which stops are connected by bus lines, but also at what time such buses pass by each stop. This type of situation lead the scientific community to define *temporal graphs*, that is, graphs in which vertices and edges can appear and disappear over time. In other words, in a temporal graph vertices and edges are active at specific time intervals. The interest in these structures is because they occur naturally in practice. To cite an example with great current relevance, consider a graph that models the physical proximity between people. In this graph, the vertices are people and we connect two vertices by an edge if these people were within a radius of less than 2 meters from each other at any given time. In the context of the COVID-19 pandemic, this model can be used to trace possible cases of infection. In it, a person will be at risk if she has had contact with a sick person. Note however that the contact will only be relevant if it occurs within a specific time window, the one in which exactly one of the two persons was sick. Therefore, any application using this model should take the time of contact into account before alerting users of any risk. This is just one example of the many possible real-life applications of temporal graphs.

A disadvantage of this diversity of applications is that many concepts have been rediscovered or developed in parallel, using different terminology. Even the name of the field itself undergoes many variations, such as *time-varying networks*, *dynamic networks* *temporal graphs*, and *link streams*. The wide range of applications has been one of the main reasons why this field has developed so quickly in the past two decades. In fact, the interest in these structures has been so expressive that two international conferences dedicated to the subject were recently created: a satellite event of the "EATCS International Colloquium on Automata, Languages and Programming - ICALP", an international event considered to be of the highest level that held its 49th edition in Paris in July 2022; and the "Symposium on Algorithmic Foundations of Dynamic Networks - SAND", which held its 1st edition in 2022 in an entirely virtual way, and whose 2nd edition will be held in June 2023 in Pisa, Italy.

Considering that the theoretical basis for these structures is still being consolidated, and that most of the existing works analyze problems from the point of view of applications, we see that there is great importance in investigating more basic problems in these structures. In particular, among the problems of greatest interest, both from a practical and theoretical point of view, there are problems that involve finding paths between objects, or subsets of objects with large interconnection with each other, or even critical subsets of objects whose removal (failure) can compromise the robustness of a network. For example, consider again the model for a public transportation network. The existence of paths that take any stop of the network to any other stop is extremely important, since it is desired that the inhabitants can move freely between regions. In a similar way, identifying which are the critical bottlenecks that separate one region from another can help in planning line schedules. These and other examples are problems that in graphs are known as *connectivity problems*.

We highlight that such connectivity problems are inherently different from the corresponding ones in static graphs as they rely on a more constrained definition of path. For temporal graphs, a path between two nodes makes sense *only if* the time of the traversed edges are time-increasing, in the same way as the bus rides of a same journey must be consecutive. As a toy example, suppose that one wants to go by bus from A to C passing through B. This is possible if one can leave from A and arrive in B before the bus from B to C has left. In the same way, an infected individual A can transfer the virus to C indirectly trough B only if A first meets B and *after* (within a certain time window) B meets C. This difference in the notion of a path dramatically changes the nature of connectivity problems, meaning that static graph theory cannot be directly applied but it can be interestingly revisited to deal with temporal graphs.

This book was written to serve as a textbook for a 5 hours mini-course offered at the 34th Brazilian Mathematics Colloquium, held in Rio de Janeiro in July 2023. However, we tried to cover many of the results on paths problems, and vertex connectivity problems, that existed in the literature by the time of writing. It could then be used in a longer course on advanced topics of graph theory. Most of the presented results concern computational complexity and parameterized computational complexity of problems on temporal graphs. We first focus on reachability and routing, i.e. computing optimal paths according to certain criteria. We then present theoretical results on possible versions of Menger's Theorem. The book is mostly self contained, but previous knowledge of basic graph theory and computational complexity is desired.

Contents

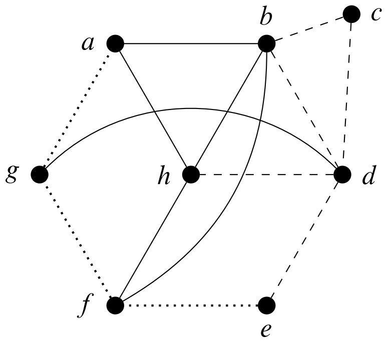
1 Basic definitions and terminology

In this chapter, we will present most of the notation that will be used throughout the book. Some further definitions and notation will be presented in the following chapters. If the reader is comfortable with basic graph theory terminology and (parameterized) computational complexity, then Sections 1.1 and 1.4 can be skipped. It should be noted that our aim in this chapter is simply to set out the tools needed in the following chapters, and not show all possible terminology being used by the community. In fact, at the time of production, there is no consensus about terminology. We refer the reader to the surveys by Holme (2015) and Latapy, Viard, and Magnien (2018) for alternative notation.

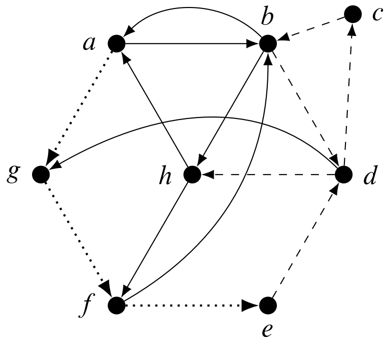
1.1 Graph terminology

A *graph* is pair $G = (V, E)$ of finite sets, where $V \neq \emptyset$, together with a function f that assigns to each element of E a subset of size at most 2 of V. Set V is called the set of *vertices*, while set E, the set of *edges* of the graph. If a graph is denoted by G, then we can also refer to the set of vertices and edges of G as $V(G)$ and $E(G)$, respectively. Given a graph G, if $e \in E(G)$ has size 1, then it is called a *loop*. Here we will be working only on graphs without loops. Additionally, if $f(e) = \{u, v\}$ for $e \in E(G)$, then we say that e has *endpoints u and v*. In general, the same set $\{u, v\}$ can be the image of more than one edge in $E(G)$. In such cases, some authors choose to call G a *multigraph* and call the set $\{u, v\}$ a *multiedge*. And if the graph has no loops and no set is the image of more than one edge of $E(G)$, then G is called a *simple graph*. Since here we will be working only on simple graphs, from now on we refrain from using the word “simple” and of making reference to the function f, i.e., an edge e will be represented directly by its endpoints. Additionally, if $\{u, v\}$ is an edge of G, then for simplicity we write uv.

Similarly, a (simple) *directed graph* is a pair $G = (V, E)$ of finite sets, where $V \neq \emptyset$ and $E \subseteq V \times V$. Some authors choose to refer to the elements of the set E as *arcs*, but since in what follows we will sometimes simultaneously refer to graphs and directed graphs, we opt to call them edges as well. Hence, as before, V is called the set of vertices of G, and E the set of edges, and we also use $V(G)$ and $E(G)$ to refer to these sets. Additionally, if $e = (u, v)$ is an edge of G, then we say that e has *endpoints* u and v, that u is the *tail* of e, and that v is the *head* of v. For simplicity we also write uv instead of (u, v). We will also be working with simple directed graphs, which means that at most one occurrence of uv is possible (i.e. all edges have multiplicity one), and that $uu \notin E(G)$ for every $u \in V(G)$ (i.e., no loops are allowed). See Figures 1.1a and 1.1b for an example of a graph, and of a directed graph. If we want to emphasize the fact that we are dealing with a *graph* instead of a *directed graph*, we might also say *undirected graph*.



(a) Example of a graph.



(b) Example of a directed graph.

Given a (directed) graph $G = (V, E)$, if $uv \in E(G)$, we say that u and v are *adjacent* or that they are *neighbors*. We also say that e is *incident* in u and v. We denote by $N(u)$ the set of neighbors of u, and call $|N(u)|$ the *degree of* u, denoting it by $d(u)$. In the graph of Figure 1.1a, for instance, the degree of vertex a is equal to 3, while its neighborhood is the set $\{b, g, h\}$. If G is directed and $uv \in E(G)$, then we say that u is a *in-neighbor* of v, while v is an *out-neighbor* of u. We denote the set of in-neighbors (out-neighbors) of u by $N^-(u)$ $(N^+(u))$, and the *in-degree* (*out-degree*) of u is equal to $|N^-(u)|$ $(|N^+(u)|)$ and is denoted by $d^-(u)$ $(d^+(u))$. In the directed graph of Figure 1.1b, for instance, a has in-degree and out-degree 2, with $N^-(a) = \{b, h\}$ and $N^+(a) = \{b, g\}$. A vertex is called *isolated* if it has degree 0, when G is undirected, or if it has in-degree and out-degree 0, when G is directed. Finally, the *maximum degree* of a (directed) graph is equal to the maximum over all (in and out) degrees of vertices of G, and is denoted by $\Delta(G)$. We can also talk about *maximum in-degree* and *maximum out-degree*, denoting them by $\Delta^-(G)$

and $\Delta^+(G)$, respectively.

A *subgraph* of a (directed) graph G is a another (directed) graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq V(G)$. It is said to be *induced* if all edges appearing in G between vertices of H are also present in H; formally, when $uv \in E(H)$ if and only if $u, v \in V(H)$ and $uv \in E(G)$. And it is said to be *spanning* if $V(H) = V(G)$. We also write $G[X]$ to denote the induced subgraph of G with vertex set X (alternatively the *subgraph of* G *induced by* X). In Figures 1.1a and 1.1b, if we pick vertices $X = \{b, h, f\}$, then $G[X]$ contains all edges between these vertices.

A (directed) graph G is *complete* if $uv \in E(G)$ for every $u, v \in V(G)$, $u \neq v$, and it is empty if $E(G) = \emptyset$. A subset $X \subseteq V(G)$ is called a *clique* if $G[X]$ is complete, and is called a *stable (or independent) set* if $G[X]$ is empty. In Figure 1.1a, the set $\{b, f, h\}$ is a clique, while $\{a, c, e\}$ is a stable set. As for the directed graph in Figure 1.1b, $\{a, b\}$ is a clique, while $\{a, c, e\}$ is also a stable set. Note that $\{a, b\}$ is the only clique of size bigger than one. If $C \subseteq V(G)$ is a clique of size k, we also say that C is a *k-clique*.

A *walk* in a (directed) graph G is a sequence $P = (v_0, \dots, v_q)$ such that $v_{i-1}v_i \in E(G)$ for every $i \in [q]$. We also write v_0, v_q-walk and say that v_0, v_q are the *extremities* (or *endpoints*) of P, and that v_0 *reaches* v_q. If P is such that $v_i \neq v_j$ for every $i, j \in \{0, \dots, q\}$ with $i \neq j$, then we say that P is a *v_0, v_q-path*. And if P is a path and $v_q v_0 \in E(G)$, then we also say that P is a *cycle*. Observe Figures 1.1a and 1.1b to see that the dashed edges form a walk, while the dotted edges form a path. Finally, we denote by $V(P)$ the set $\{v_0, \dots, v_q\}$, and by $E(P)$ the set of edges $\{v_0v_1, \dots, v_{q-1}v_q\}$.

A *cycle* in a (directed) graph G is a path $P = (v_0, \dots, v_q)$ such that $v_0 v_q \in E(G)$. We say that G is *acyclic* if it does not contain a cycle. If G is undirected, this is also called a *forest*. And we use DAG to write directed acyclic graph for short.

Now, given two u, v-walks/paths $P = (u = v_0, \dots, v_q = v)$ and $P' = (u = v'_0, \dots, v'_r = v)$, we say that P and P' are *internally vertex disjoint* if $V(P) \cap V(P') = \{u, v\}$. Since here we will be dealing only with internally vertex disjointness, we refrain from using the word "internally" from hereon. In Figure 1.1a, the dashed and dotted walks are two vertex disjoint a, e-walks. Observe that in Figure 1.1b, there are no two vertex disjoint a, e-walks, as e has in-degree 1. Additionally, supposing $uv \notin E(G)$, we say that a subset $S \subseteq V(G) \setminus \{u, v\}$ is a *u, v-separator* if there is no u, v-path in $G - S$. In Figure 1.1a, the set $\{f, d\}$ is an a, e-separator, while in Figure 1.1b, the set $\{f\}$ is an a, e-separator. We denote by $p(u, v)$ the maximum number of vertex disjoint u, v-paths and, when $uv \notin E(G)$, we denote by $c(u, v)$ the minimum size of a u, v-separator. These metrics would be the same if defined in terms of walks instead. The famous Menger's Theorem tells us that these two values are equal.

Theorem 1 (Menger (1927)). *Let G be a (directed) graph, and $u, v \in V(G)$ be two vertices in G such that $uv \notin E(G)$. Then $p(u, v) = c(u, v)$.*

Finally, given a (directed) graph G and vertices $u, v \in V(G)$, the *identification of u and v* is the graph H obtained from G by removing u and v, and adding a new vertex z and edges in a way that: if G is undirected, then $N_H(z) = N_G(u) \cup N_G(v)$; and if G is directed, then $N^+_H(z) = N^+_G(u) \cup N^+_G(v)$ and $N^-_H(z) = N^-_G(u) \cup N_G(v)$.

1.2 Temporal graphs

In what follows, given a positive integer n, we denote the set $\{1, \dots, n\}$ by $[n]$. Also, we use $[t, t']$ to denote the set $\{t, t+1, \dots, t'\}$ and we call it *interval* $[t, t']$.

A *temporal (directed) graph* is a pair $\mathcal{G} = (G, \lambda)$ where $G = (V, E)$ is a (directed) graph (called *base graph*) and $\lambda : E \to 2^{\mathbb{N}}$ is a *time-function* from the edges of G into nonempty finite subsets of positive integers. The value $\tau = \max_{e \in E}\{t \mid t \in \lambda(e)\}$ is called the *lifetime* of $\mathcal{G}$. We can also see $\mathcal{G}$ as a sequence of (directed) graphs $(G_i)_{i=1}^{\tau}$, where every G_i is a spanning subgraph of G and $e \in E(G_i)$ if and only if $i \in \lambda(e)$. See Figures 1.2 and 1.3 for the two possible representations that will be used throughout the text. The graph G_i is called *snapshot* of G at time i, while a value $i \in [\tau]$ is called *timestep*. For an edge $e \in E(G)$ and $i \in \lambda(e)$, we call the pair (e, i) a *temporal edge* and say that e is *active in timestep* i. Additionally, for each $v \in V(G)$ and $i \in [\tau]$, we call the pair (v, i) a *temporal vertex*. The set of temporal edges of $\mathcal{G}$ is denoted by $E^T(\mathcal{G})$, while the set of temporal vertices is denoted by $V^T(\mathcal{G})$. Similarly, the set of vertices of $\mathcal{G}$ (e.g., vertices of the base graph G) is denoted by $V(\mathcal{G})$, while the set of edges, by $E(\mathcal{G})$. If not clear from the context, we might also use $\lambda_{\mathcal{G}}$ to denote the time-function, and $\tau(\mathcal{G})$ to denote the lifetime.

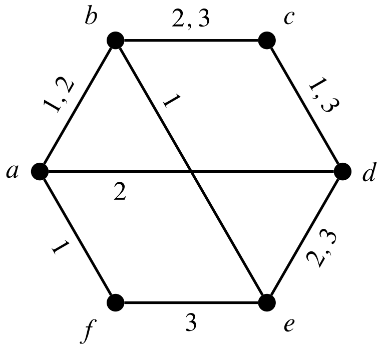


Figure 1.2: Example of a temporal graph, with the λ function being represented on top of each edge. We omit brackets in order to make the figure cleaner.

Observe that an equivalent definition is to have a pair (H, γ) where H is a multigraph and $\gamma : E(H) \to \mathbb{Z}_+$. Indeed, if (G, λ) is a temporal graph as defined above, then we can obtain a pair (H, γ) by adding $|\lambda(e)|$ copies of e to H, making each edge active in a distinct timestep in $\lambda(e)$. On the other hand, if (H, γ) is such that H is a multigraph and $\gamma : E(H) \to \mathbb{Z}_+$, then we can collapse all the occurrences of a multiedge uv to a single occurrence, making $\lambda(uv)$ be equal to the union of $\gamma(e)$, for every $e \in E(H)$ with endpoints uv. Such alternative definition can sometimes be more convenient and has been used by Ibiapina and Silva (2022) to characterize Mengerian graphs, which will be defined

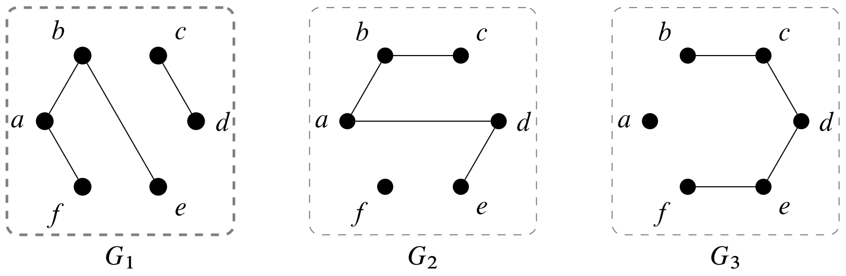


Figure 1.3: Example of a temporal graph represented as a sequence of subgraphs. For simplicity, we omit isolated vertices. Observe that this is the same temporal graph as the one in Figure 1.2.

shortly.

Another possible, more general, notion is the following. A *temporal graph with lifetime* τ is a triple $\mathcal{G} = (G, \alpha, \phi)$ such that G is the *base multigraph*, α is the *starting-time function* that assigns to each $e \in E(G)$ a value in $[\tau]$, and ϕ is the *travel-time function* that assigns to each $e \in E(G)$ a value in $[\tau - \lambda(e)]$. The idea is that an edge $e = (u, v)$ leaves from u at time $\alpha(e)$ and arrives in v at time $\alpha(e) + \phi(e)$. Observe that this generalizes the previous notions, with $\phi(e) = 0$ for every $e \in E(G)$ for the non-strict model, and $\phi(e) = 1$ for the strict model. Such model is more convenient to present positive results, and indeed this is used in Chapter 2 to present polynomial algorithms for the many minimum paths notions. In fact, we will use a more convenient structure, which can be seen as a temporal graph where the edges are ordered according to their starting time. This will be presented in Section 1.3. Nevertheless, this more general notion will also be used in Chapter 3 to present some negative results. The sets $V(\mathcal{G})$, $E(\mathcal{G})$, $V^T(\mathcal{G})$ and $E(\mathcal{G})$ as before, except that now a temporal edge is a tuple (u, v, t, d) where $t = \lambda(e)$ and $d = \phi(e)$. In what follows, we continue to use the simpler model (G, λ), but the reader should observe that all the notions can be adapted for this more general notion.

A *temporal subgraph* of a temporal (directed) graph $\mathcal{G}$ is a another temporal (directed) graph $\mathcal{H}$ such that $V(\mathcal{H}) \subseteq V(\mathcal{G})$ and $E^T(\mathcal{H}) \subseteq E^T(\mathcal{G})$. Note that this implies $E(\mathcal{H}) \subseteq E(\mathcal{G})$, as $\lambda_H(e) \neq \emptyset$ for every $e \in E(\mathcal{H})$. We say that $\mathcal{H}$ is *spanning* if $V(\mathcal{H}) = V(\mathcal{G})$. Given a temporal (directed) graph $\mathcal{G} = (G, \lambda)$ with lifetime τ, and a set $S \subseteq V(G)$, we denote by $\mathcal{G}[S]$ the *temporal (directed) subgraph induced by* S, which is equal to $(G[S], \lambda')$ with λ' being equal to λ restricted to $E(G[S])$. Additionally, given a subset of timesteps $T \subseteq [\tau]$, we denote by $\mathcal{G}[T]$ the *temporal (directed) subgraph induced by* T, which is equal to (G, λ') with $\lambda'(uv) = \lambda(uv) \cap T$ for every $uv \in E(G)$. Finally, given $S \subseteq V(G)$, we denote by $\mathcal{G} - S$ the temporal graph obtained from $\mathcal{G}$ by removing all vertices of S. We use the same notation in case $S \subseteq E(G)$ or $S \subseteq E^T(\mathcal{G})$.

Given a temporal (directed) graph $\mathcal{G} = (G, \lambda)$ and vertices $v_0, v_q \in V(G)$, a *temporal*

v_0, v_q-*walk* in $\mathcal{G}$ is defined as a sequence of vertices and timesteps

$$P = (v_0, t_1, v_1, \dots, t_q, v_q)$$

such that $t_1 \leqslant t_2 \leqslant \dots \leqslant t_q$, and for each $i \in [q]$, we have that $(v_{i-1}v_i, t_i)$ is a temporal edge of $\mathcal{G}$. If $t_1 < t_2 < \dots < t_q$, then we say that P is *strict*. We also say that P *starts in time* t_1 and *finishes in time* t_q, and, for each $i \in [q-1]$, that P *waits in* v_i *from* t_i *to* t_{i+1}. Additionally, if no vertices of G are repeated in P, then we say that P is a *temporal* v_0, v_q-*path*. As we will see shortly, differently from static graphs, when dealing with temporal graphs, the notions of paths and walks might lead to different measures, depending on the type of disjointness being considered. Given two vertices $u, v \in V(G)$, we say that u *reaches* v in $\mathcal{G}$ if there exists a temporal u, v-walk in $\mathcal{G}$.

In the example of Figures 1.2 and 1.3, the sequence $(e, 1, b, 2, a, 2, d, 2, e, 3, f)$ is a temporal e, f-walk, while only $(e, 3, f)$ is a temporal e, f-path. Additionally, such walk is not strict, hence if we are considering only strict walks/paths, then $(e, 3, f)$ is the only possible temporal e, f-walk/path.

For the more general notion in which the edges have a travel time, a temporal walk/path is defined similarly, except that we must arrive in the next vertex of the walk/path in time to use the next temporal edge. Because of this, it is not enough to represent the walk as a sequence of vertices and timesteps. Instead, it will be a sequence of vertices and temporal edges, $P = (v_0, e_1, v_1, \dots, e_q, v_q)$, such that, for every $i \in [q]$, we have that $e_i = (v_{i-1}, v_i, t_i = \lambda(a_i), d_i = \phi(e_i))$, and $t_{i+1} \geqslant t_i + d_i$. We do not need to define strict paths, as the strictness is implicit in the travel-time function. Also, the notions of starting time, finishing (or *arrival*) time, waiting, and reachability are as before.

Still considering a temporal walk $P = (v_0, t_1, v_1, \dots, t_q, v_q)$, we denote the set $\{v_0, \dots, v_q\}$ by $V(P)$, the set of edges $\{v_0v_1, \dots, v_{q-1}v_q\}$ by $E(P)$, and the set of temporal edges $\{(v_0v_1, t_1), \dots, (v_{q-1}v_q, t_q)\}$ by $E^T(P)$. Additionally, we denote by $V^T(P)$ the set of temporal vertices contained in P. Note that this includes the temporal vertices on which we are simply waiting. Formally, $V^T(P) = \{(v_0, t_1), (v_q, t_q)\} \cup \{(v_i, t) \mid i \in [q-1], t \in \{t_i, t_i + 1 \dots, t_{i+1}\}$. For example, in Figure 1.2, the temporal a, e-path $(a, 1, f, 3, e)$ has vertex set $\{a, f, e\}$, edge set $\{af, fe\}$, temporal edge set $\{(af, 1), (fe, 3)\}$ and temporal vertex set $\{(a, 1), (f, 1), (f, 2), (f, 3), (e, 3)\}$. We always assume that no waiting occurs in the beginning nor in the end of every temporal walk. Formally, the only copy of v_0 added to $V^T(P)$ due to the temporal edge (v_0v_1, t_1) is the temporal vertex (v_0, t_1); similarly, only (v_q, t_q) is added due to $(v_{q-1}v_q, t_q)$. Additionally, for $i, j \in \{0, \dots, q\}$, we denote by v_iPv_j the v_i, v_j-walk $(v_i, t_{i+1}, v_{i+1}, \dots, t_j, v_j)$. Finally, if the more general notion (G, λ, ϕ) is being considered, for a temporal walk $P = (v_0, e_1, v_1, \cdots, e_q, v_q)$, we define $V(P)$, $E(P)$ and $E^T(P)$ as before, while $V^T(P)$ is defined as $V^T(P) = \{(v_0, t_1), (v_q, t_q + d_q)\} \cup \{(v_i, t) \mid i \in [q-1], t \in \{t_i + d_i, t_i + d_i + 1, \dots, t_{i+1}\}\}$, where $t_i = \lambda(e_i)$ and $d_i = \phi(e_i)$ for each $i \in [q]$.

1.3 Link Stream

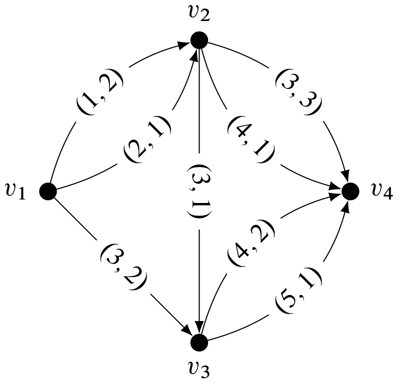
The first definition of a temporal graphs given in the previous section was as a pair (G, λ), where the travel-time of all edges was set either to 0 or to 1, when only strict walks are considered. While this assumption is reasonable to prove stronger negative results, in practical applications, when providing positive results, we aim to work also with travel-time of the edges. This indeed makes sense if your network models for instance the streets of a city, with the nodes being street crossings. In such cases, when leaving a crossing at a given time t, if the granularity used is small enough (e.g. in minutes or seconds), then it is natural that you will arrive at the next crossing only at a later time $t + \phi$. Additionally, the time ϕ needed to go between a pair of crossings might differ during the day, depending on rush hours. Hence why work on the model (G, λ, ϕ) instead.

However, in practice, there is an even more convenient model. Naturally, it matters how temporal graphs are given as input to the programs. In particular, as the number of temporal edges in a temporal graph can be much more than n^2, where n is the number of vertices, it is reasonable to assume that the edges are stored in a file (in main memory). As it would be preferably not to load all the whole file in main memory, algorithms read the list of temporal edges doing passes on this file, while storing only $O(n)$ memory. This approach is very common in the case of streaming algorithms when dealing with big data and has been applied also for temporal graphs. This leads us to the notion of link stream, which mathematically speaking is equivalent to the notion with edge travel-time, but is more convenient as the edges are sorted according to their starting time.

A *link stream* is a pair $(V, \mathbb{E})$, where $\mathbb{E}$ is a list of *temporal edges* $e = (u, v, t, \phi)$, where $(u, v) \in V \times V$, $t \in T$ is the *starting of* e, and ϕ denotes the *travel* time of e. We also say that the *arrival time of* e is equal to $t + \phi$. Additionally, if the list is sorted in non-decreasing according to the starting time of the edges, then we write $\mathbb{E}_{\downarrow}$, and if it is sorted non-increasing order, we write $\mathbb{E}_{\uparrow}$. See Figure 1.4 for an example.

Clearly, given a link stream $(V, \mathbb{E})$, then a subjacent directed graph is defined; we call it the *base graph*. We can also assume *undirected* link streams by assuming that each temporal edge $(u, v, t, \phi) \in \mathbb{E}$ is undirected, i.e., that $\{u, v\} \subseteq V$. Finally, we will say that the link stream is *delay-1* if all temporal edges have the same positive travel time, which, without loss of generality, can be assumed to be equal to 1.

As in the previous section, given a link stream $\mathcal{G} = (V, \mathbb{E})$, the *lifetime* of $\mathcal{G}$ is the value $\tau(\mathcal{G}) = \max_{(u,v,t,d)\in\mathbb{E}} t + d$, omitting $\mathcal{G}$ when it is clearly from the context. Also, we denote V by $V(\mathcal{G})$, the set $\{uv \mid \exists(u, v, t, d) \in \mathbb{E}\}$ by $E(\mathcal{G})$, the set $V \times [\tau]$ by $V^T(\mathcal{G})$, and the set $\{(u, v, d, t) \mid (u, v, d, t) \in \mathbb{E}\}$ by $E^T(\mathcal{G})$. Observe that there is little difference between $E^T(\mathcal{G})$ and $\mathbb{E}$, with the former being a set, and the latter being the same set put in a (sorted) list. This is why sometimes refer to them interchangeably.



List $\mathbb{E}_\downarrow$	List $\mathbb{E}_\uparrow$
$(v_1, v_2, 1, 2)$	$(v_3, v_4, 5, 1)$
$(v_1, v_2, 2, 1)$	$(v_3, v_4, 4, 2)$
$(v_1, v_3, 3, 2)$	$(v_2, v_4, 4, 1)$
$(v_2, v_3, 3, 1)$	$(v_2, v_4, 3, 3)$
$(v_2, v_4, 3, 3)$	$(v_2, v_3, 3, 1)$
$(v_2, v_4, 4, 1)$	$(v_1, v_3, 3, 2)$
$(v_3, v_4, 4, 2)$	$(v_1, v_2, 2, 1)$
$(v_3, v_4, 5, 1)$	$(v_1, v_2, 1, 2)$

Figure 1.4: An example of link stream with the corresponding lists of temporal edges. The label of each temporal edge denotes its starting and travel times (separated by a comma). The list $\mathbb{E}_\downarrow$ (respectively, $\mathbb{E}_\uparrow$) is sorted in non-decreasing (respectively, non-increasing) order with respect to the edge starting times.

1.4 Computational complexity

A *decision problem* is a language $L \subseteq \Sigma^*$, where Σ is a fixed finite alphabet. For instance, in modern computers all objects are represented as a sequence of 0's and 1's. Hence, a language is any family of such sequences. This concept however is more useful when studying the theory behind computational complexity classes and relations between them. Since here we instead want to classify some problems into such computational classes, it will be more useful for us to see a decision problem as a "yes/no" question posed on some object. For instance, given a graph G and a positive integer k, one might be interested to know whether G has a k-clique. We call such problem the Clique problem, and we say that it receives G and k as *input*, and that the related *question* is "Does G have a k-clique?". In general, we will be describing a decision problem in the following way:

Clique
Input. A graph G, and an integer k.
Question. Does G have a k-clique?

We call a particular pair given as input an *instance*. Applying the formal definition to the above problem, the language L related to the problem Clique would contain a sequence in Σ^* representing each of the instances of Clique whose answer to the question is "yes".

Since the scope of this book is not to dwell into the world of formal computational complexity theory, here we define the computational classes of interest in a somewhat informal way, meaning that some concepts are assumed to be understood. For the reader interested in learning more about this theory, we refer to Sipser (1996). The *size* of an instance x of a problem Π is equal to the size of the sequence needed to represent x; it will be denoted by $|x|$. We say that a decision problem is *polynomial-time solvable* if there exists an algorithm that correctly computes the answer to the problem's question running in time $O(|x|^c)$, for some constant c. The set of all polynomial-time solvable problems is denoted by P. Additionally, a problem Π is said to be *non-deterministically polynomial-time solvable* if, for every positive instance x of Π, there exists a *certificate for x* that can be checked by a polynomial-time algorithm. For example, given a positive instance (G, k) of Clique, a certificate for (G, k) can be a k-clique C of G; one can check whether C is indeed a k-clique in polynomial time simply by testing, for every $u, v \in C$ with $u \neq v$, whether $uv \in E(G)$. One can think of a certificate as a "short proof" that x is indeed a positive instance of Π. The set of all non-deterministically polynomial-time solvable problems is denoted by NP. We also define the class co-NP to contain the complement of all NP problems. Formally, given a language $L \subseteq \Sigma^*$, the *complement of* L is the language $\overline{L} = \Sigma^* \setminus L$. Hence, we define co-$\mathsf{NP} = \{\overline{L} \mid L \in \mathsf{NP}\}$. Informally, these are the decision problems for which there exist "short proofs" for negative instances. Note that this is not the same as the complement of NP. Indeed, it can be proved that $\mathsf{P} \subseteq \mathsf{NP} \cap \text{co-}\mathsf{NP}$. As an example of co-$\mathsf{NP}$ problem, consider the problem Tautology defined below. Observe that a short proof for a negative instance would be a truth assignment that valuates the formula as false.

Tautology
Input. A CNF formula ϕ.
Question. Is ϕ a tautology, i.e., is ϕ satisfied by every possible truth assignment to its variables?

Now, given two decision problems, Π and Π', and instances x and x' of Π and Π', respectively, we say that x and x' are *equivalent* if x is a "yes" instance of Π if and only if x' is a "yes" instance of Π'. Then we say that Π is *polynomial-time (Karp) reducible to* Π' if there exists a polynomial time computable function f that transforms any instance x of Π into an equivalent instance $f(x)$ of Π'. In the remainder of the text, we omit Karp's name when referring to such reductions; we also write $\Pi \leqslant_p \Pi'$ if such reduction exists. The interest of such definition is the fact that it preserves polynomiality (see Exercise 2). Finally, it is said that a problem Π' is *NP-hard* if $\Pi \leqslant_p \Pi'$ for every $\Pi \in \mathsf{NP}$, and that it is *NP-complete* if it is NP-hard and $\Pi' \in \mathsf{NP}$.

Intuitively, the class of NP-complete problems represents all problems in NP in terms of complexity. In other words, since $\leqslant_p$ is a transitive relation (see Exercise 3), we get that if any NP-complete problem is polynomial-time solvable, then so are all NP problems, i.e., $\mathsf{P} = \mathsf{NP}$. Now, Cook-Levin's Theorem is the seed for the construction of a family of problems that we know to be NP-hard, as it tells us that 3-SAT, defined below for general k, is NP-complete. Throughout the book, whenever we want to show that a problem Π is NP-complete, we simply present a polynomial-time reduction to Π from another previously known NP-complete problem.

k-SAT
Input. A CNF formula ϕ such that each clause contains at most k literals.
Question. Is there a satisfying truth assignment to the variables of ϕ?

Observe that, in some kinds of problems where there is also a related metric, like it was the case for CLIQUE, we can sometimes be interested in finding cliques with a given fixed time. For instance, one could be interested simply in finding a triangle (clique of size 3). Such test can be done in time $O(n^3)$ on a graph with n vertices, as it suffices to test every subset S of size 3 whether S is a clique. Generalizing for a fixed value k, we have an algorithm running in time $O(n^k)$ to solve CLIQUE. This contrasts with some other problems that do not admit algorithms running in time $O(n^k)$, and some that admit even better algorithms, running in time $O(f(k) \cdot n)$, for some computable function k. This brings us to the granularity introduced by the study of parameterized complexity.

Formally, a *parameterized problem* is a language $\Pi \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed finite alphabet. A pair $(x, k) \in \Sigma^* \times \mathbb{N}$ is called an *instance* of Π with *parameter* k, and we say that it is a *"yes" instance* if $(x, k) \in \Pi$. Following the chosen notation for decision problems, we will be describing parameterized problems in natural language instead. For the reader interested in the formal definitions, we refer to Cygan et al. (2015). Hence, a parameterized problem is exactly like a decision problem, equipped with a parameter. For instance, we can talk about the CLIQUE problem parameterized by k. We emphasize that the parameter is not necessarily part of the input. We can investigate CLIQUE parameterized by the maximum degree of the input graph G, for instance. Other examples of possible parameters in graphs are the vertex cover number, the treewidth, the neighborhood diversity, the more recently defined twinwidth, etc. If the considered problem has an input related to the size of a searched object, we usually say that this value is the *natural parameter*.

Now, a parameterized problem Π with parameter k is said to be *fixed-parameter tractable* (FPT for short) if there exists an algorithm A that correctly answers to the problem's question and runs in time $f(k) \cdot n^O(1)$, for some computable function f, where n denotes the size of the corresponding input. We say that A is an FPT algorithm, and the set of all fixed-parameter tractable problems is also denoted by FPT. Making an abuse of language, we also say that a decision problem Π is FPT *when parameterized by k* if the related parameterized problem is FPT.

Class FPT works like a sort of P class inside the parameterized complexity theory. However, another class defined by positive results (i.e., existence of algorithms) is the following. We say that a parameterized problem Π is *slice-wise polynomial* (XP for short) if there exists an algorithm A that correctly answers to the problem's question and runs in time $n^{f(k)}$, for some computable function f, where n denotes the size of the corresponding input. We say that A is an XP algorithm, and the set of all slice-wise polynomial problems is also denoted by XP. Making an abuse of language, we also say that a decision problem Π is XP *when parameterized by k* if the related parameterized problem is XP. We saw previously that Clique is XP when parameterized by the natural parameter. A natural question is whether the $O(n^k)$ algorithm can be improved to an FPT algorithm. The answer is "no, unless FPT = W[1]-hard", where W[1]-hard is defined next. It can be thought of as the NP-hard counterpart in the parameterized complexity theory.

Given two parameterized problems $\Pi, \Pi' \subseteq \Sigma^* \times \mathbb{N}$, and instances (x, k) and (x', k') of Π and Π', respectively, it is said that they are *equivalent* if (x, k) is a "yes" instance of Π if and only if (x', k') is a "yes" instance of Π'. A *parameterized reduction* from Π to Π' is a function that, given an instance (x, k) of Π, computes an equivalent instance (x', k') of Π' such that $k' \leqslant g(k)$ in time $f(k) \cdot |x|^{O(1)}$, where f and g are computable functions. If such a reduction exists, we write $\Pi \leqslant_p^* \Pi'$. Let Clique$_k$ denote Clique parameterized by the natural parameter. Finally, we say that a parameterized problem Π is *W*[1]*-hard* if Clique$_k \leqslant_p^* \Pi$. As Clique$_k$ is not believed to be FPT, and $\leqslant_p^*$ preserves solvability in parameterized time (see Exercise 4), intuitively this means that Π is also not believed to be FPT. We refer the reader to Cygan et al. (ibid.) for further background on parameterized complexity.

1.5 Common constructions

When dealing with a generalization of graphs, a natural approach is trying to model the new problems as a graph problem. A graph theorist might even be tempted to believe that all the time function overhead can be dealt with as a classical graph problem. This however is not a good approach as very often the time aspect of the problems indeed plays a crucial role on the problem's complexity, with some easily solvable graph problems becoming hard even on very strict cases when generalized to temporal graphs. Nevertheless, there are indeed some constructions that have appeared in a number of results, and hence deserve attention. In what follows, we present the ones that we considered the most relevant.

The first trivial constructions are to allow to go from an undirected temporal graph to a directed one, or vice-versa. If $\mathcal{G}$ is a temporal graph, then the *directed version* of $\mathcal{G}$ is obtained by replacing each edge uv by two directed edges uv and vu with the same λ function. And if $\mathcal{G}$ is a temporal directed graph, then its *undirected version* is obtained by ignoring the directions of the edges. Observe that if uv and vu are in $\mathcal{G}$, then these will be merged in a single undirected edge, appearing in the same timesteps as uv and vu.

A more widely used construction is the following. Given a temporal graph $\mathcal{G} = (G, \lambda)$ with lifetime τ, the *strict static expansion* of $\mathcal{G}$ is the DAG (V', E'), where $V' = V(G) \times \{0, \dots, \tau\}$ and, for each $i \in [\tau]$, we have that $(u, i-1)(v, i) \in E'$ if and only if either $(uv, i) \in E^T(\mathcal{G})$, or $u = v$. See Figure 1.5. In words, it is the DAG obtained by making $\tau + 1$ copies of each vertex, and representing the temporal edges of snapshot i by linking the corresponding copies in layer $i - 1$ to layer i. Additionally, a path from the first to the last copy of a vertex, passing by all other copies in order, is added. One can see that this transformation preserves existence of strict temporal paths (see Exercise 5), but not of non-strict temporal paths. Indeed, in Figure 1.5, the shaded path is related to the strict temporal a, d-path $(a, 1, b, 2, c, 3, d)$, but the non-strict path $(a, 1, b, 1, e, 3, d)$ does not

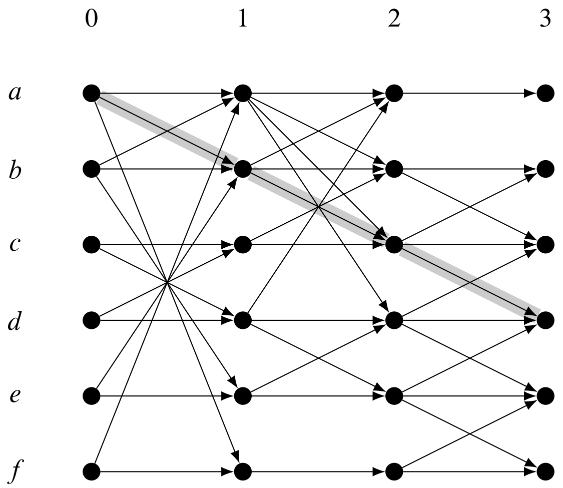


Figure 1.5: Strict static expansion of the temporal graph depicted in Figure 1.2. The shaded edges represent the strict temporal path $(a, 1, b, 2, c, 3, d)$.

appear in the expansion.

In order to preserve non-strict paths, we define the *static expansion* of $\mathcal{G}$ as the directed graph (V', E'), where $V' = V(G) \times [\tau]$, and $(u, i)(v, j) \in E'$ if and only if either $i = j$ and $uv \in E(G_i)$, or $j = i+1$ and $u = v$. See Figure 1.6. In words, this can be seen as the directed graph obtained by taking the disjoint union of the snapshots (if G is undirected, then we consider the directed version of G), and adding a path linking the copies of a given vertex. Observe that now the path $(a, 1, b, 1, e, 3, d)$ appears as the shaded path between vertices $(a, 1)$ and $(d, 3)$. The concept of strict static expansion has been used by Mertzios, Michail, and Spirakis (2019) and Michail (2016), while the definition of static expansion has been used by Campos et al. (2021). It will also be used in Section 3.3 to prove a version of Menger's Theorem for t-vertex disjoint walks.

Another common construction is what we call *(strict) temporal line graph*. In this, given a temporal graph $\mathcal{G} = (G, \lambda)$, we construct a directed graph $L = L(\mathcal{G})$ having as vertex set the set of temporal edges of $\mathcal{G}$, and where $ef \in E(L)$ if and only if e and f have a common endpoint, say u, and $\lambda(e) \leqslant \lambda(f)$ (resp. $\lambda(e) < \lambda(f)$). In other words, e followed by f form a (strict) temporal path. See Figure 1.7. One can easily check that there is a one-to-one correspondence between (strict) temporal paths in $\mathcal{G}$ and paths in L (see Exercise 6). In Figure 1.7, the temporal a, d-path $(a, 1, b, 1, e, 3, d)$ is shaded and corresponds to the path $((ab, 1), (be, 1), (ef, 3))$ in the constructed graph. Observe that, in particular, the temporal line graph must contain the line graph of each snapshot, but the same does not hold for the strict temporal line graph. A similar concept has been used by Kempe, Kleinberg, and Kumar (2000) to prove that the temporal edge disjoint version

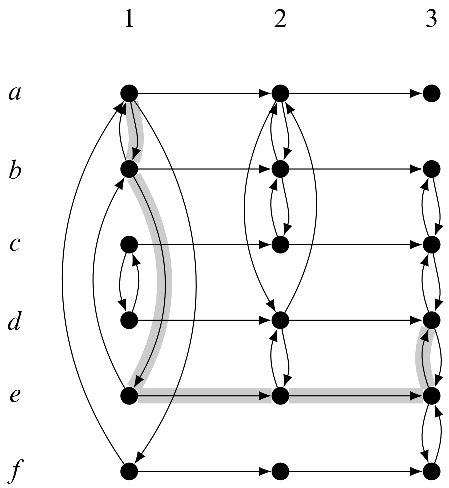


Figure 1.6: Static expansion of the temporal graph depicted in Figure 1.2. The shaded edges represent the temporal path $(a, 1, b, 1, e, 3, d)$.

of Menger's Theorem holds.

Recall that we also introduced a more general notion that allows for the edges to have a travel-time. Now, we present a useful construction that allows us to go between this to the simpler model with no travel-time. Such construction has been proposed by Kempe et al. Kempe, Kleinberg, and Kumar (ibid.) and works as follows (observe Figure 1.8). So, let (G, α, ϕ) be a temporal directed graph with travel-time on the edges. For each $uv \in V \times V$, let E_{uv} be the set of edges of G with endpoints in uv. Let G' be such that $V(G') = V \cup \bigcup_{uv \in V \times V} E_{uv}$ (in words, the vertex set is equal to V plus a vertex for each possible multiedge of G). Then, for each $uv \in V \times V$ and each $e \in E_{uv}$, let w_e be the related vertex of G'. Add to G' edges uw_e and $w_e v$, and define $\lambda(uw_e) = \{\alpha(e)\}$ and $\lambda(w_e v) = \{\alpha(e) + \phi(e)\}$. Observe that there is a bijection between temporal paths in (G', λ) and in (G, α, ϕ) (see Exercise 7). We call (G', λ) the *undelayed version* of (G, α, ϕ). In case G is an undirected graph, we construct (G', λ) as before, except that we interpret $uv \in E(G)$ as two separate edges, uv and vu, and that the edges added to the undelayed version are also undirected. Finally, a less general construction used to transform strict paths problems into non-strict is used by Zschoche et al. (2020), and the same one is presented by Crescenzi, Magnien, and Marino (2019) in the context of *link streams*. The latter will be defined and used in Chapter 2. The undelayed version is also going to be useful in Section 3.2.2.

Finally, we present a construction that allows to go from vertex disjoint version of problems to edge disjoint. Such construction is quite well known for static directed graphs.

If the construction is applied to an undirected temporal graph, we are implicitly con-

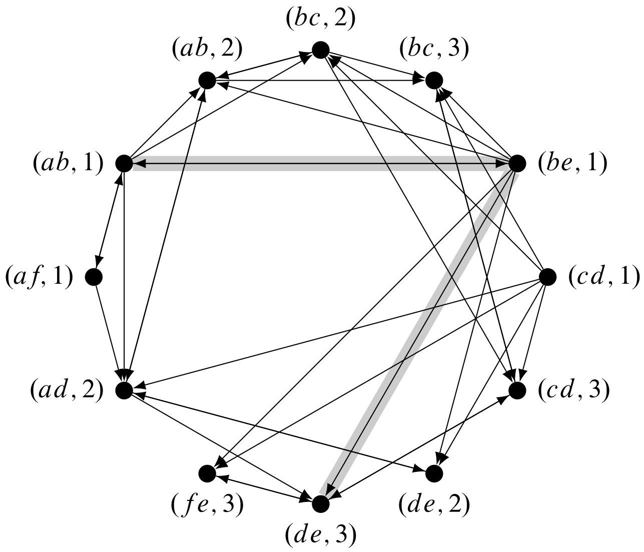


Figure 1.7: Temporal Line graph of the temporal graph depicted in Figure 1.2. The shaded edges represent the temporal path $(a, 1, b, 1, e, 3, d)$.

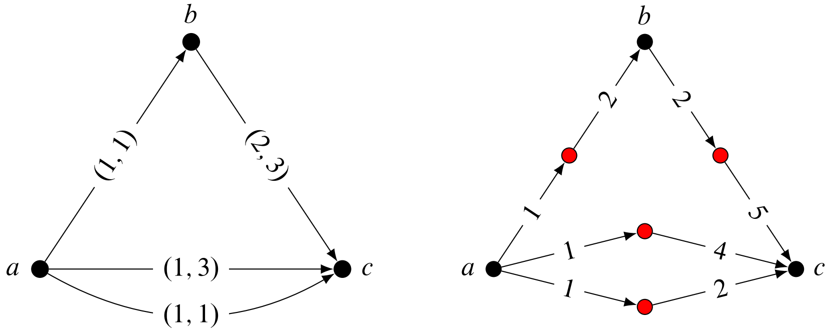


Figure 1.8: Example of the undelayed version construction. In the left, a label (t, d) on top of an edge e denotes the fact that $\alpha(e) = t$ and $\phi(e) = d$. In the right, red vertices are the new vertices added for each $e \in E(G)$.

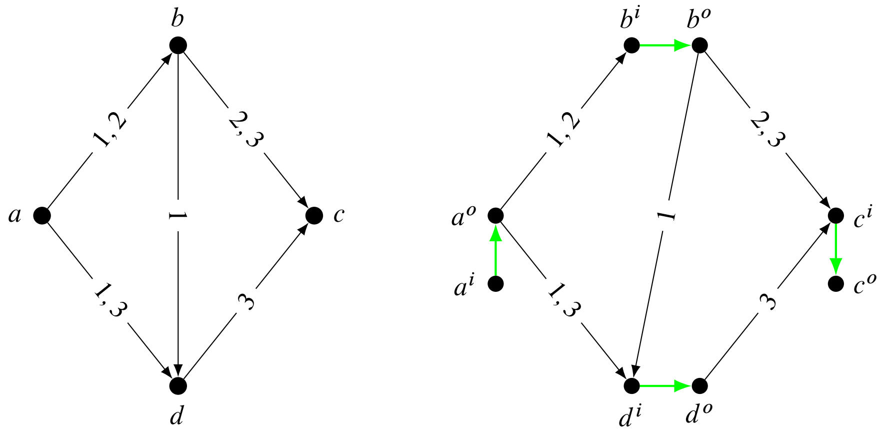


Figure 1.9: Example of the split vertex construction. In the right, green edges are always active, i.e., have λ' defined as $\{1, 2, 3\}$.

sidering the application to its directed version. So now consider a temporal directed graph $\mathcal{G} = (G, \lambda)$ with lifetime τ; observe Figure 1.9 to follow the construction. For each vertex $u \in V(G)$, add to $\mathcal{G}'$ two new vertices u^i and u^o, and an edge from u^i to u^o which is always active, i.e., such that $\lambda'(u^i u^o) = [\tau]$. Then, for each edge $uv \in E(G)$, add to $\mathcal{G}'$ an edge from u^o to v^i, active in the same time steps as uv, i.e., make $\lambda'(u^o v^i) = \lambda(uv)$. We call $\mathcal{G}'$ the *vertex split* of $\mathcal{G}$. Kempe, Kleinberg, and Kumar (2000) combine the line graph with split vertex in order to apply mincut-maxflow on the obtained directed graph.

1.6 Exercises

1. Prove that the minimum size of $S \subseteq V(\mathcal{G}) \setminus \{u, v\}$ such that there are no temporal u, v-paths in $\mathcal{G} - S$ is equal to the minimum size of a temporal vertex u, v-separator.

2. Prove that if $\Pi' \in \mathsf{P}$ and $\Pi \leqslant_p \Pi'$, then $\Pi \in \mathsf{P}$;

3. Prove that if $\Pi \leqslant_p \Pi'$ and $\Pi' \leqslant_p \Pi''$, then $\Pi \leqslant_p \Pi''$;

4. Prove that if $\Pi' \in \mathsf{FPT}$ and $\Pi \leqslant_p^* \Pi'$, then $\Pi \in \mathsf{FPT}$ (observe that this is analogous to Exercise 2).

5. Let $\mathcal{G}$ be a temporal graph, and $u, v \in V(\mathcal{G})$. Prove that there exists a (strict) temporal u, v-path in $\mathcal{G}$ if and only if there exists a $(u, i), (v, j)$-path in the (strict) static expansion of $\mathcal{G}$.

6. Let $\mathcal{G}$ be a temporal graph, and $L = L(\mathcal{G})$. Then, every (strict) path in L corresponds to a (strict) temporal path in $\mathcal{G}$, and vice-versa.

7. Let $\mathcal{G} = (G, \alpha, \phi)$ be a general temporal graph where the edges have travel times, and let $\mathcal{G}' = (G', \lambda)$ be the undelayed version of $\mathcal{G}$. Then every temporal path in $\mathcal{G}$ is uniquely related to a temporal path in $\mathcal{G}'$.

8. Let $\mathcal{G} = (G, \lambda)$ be a temporal graph and $\mathcal{G}' = (G', \lambda')$ be its vertex split. Given $s, z \in V(G)$, prove that there are k internally vertex disjoint temporal s, z-paths $\mathcal{G}$ if and only if there are k edge disjoint temporal s, z-paths in $\mathcal{G}'$.

2 Paths

In the previous chapter, we have given two notions of temporal graphs, with the more general notion being the one that allowed the edges to have a duration. The more general model is of course more appropriate when we are able to offer positive results on it, and it is the case in this chapter. The simpler model will be used in Chapter 3 to present negative results. In fact, in this chapter we will be working with the even more convenient model of link streams, where the temporal edges are sorted. Recall that we write $(V, \mathbb{E}_{\downarrow})$ to represent a link stream where the temporal edges are sorted in non-decreasing way according to their starting time, while we write $(V, \mathbb{E}_{\uparrow})$ when they are sorted in non-increasing order. We will also assume that the travel time of any temporal edge is a *positive integer value*. Hence we get only strict temporal paths. Nevertheless, the results extend to non-strict paths at the price of higher computational cost.

2.1 Weighted Paths and Reachable Sets

In Section 1.2, we defined temporal walks/paths as alternating sequences of vertices and temporal edges. However, observe that the temporal edges already give the vertices involved in such a sequence. So for convenience, we slightly change the definition given there to consider only temporal edges. Given a link stream $\mathcal{G} = (V, \mathbb{E})$ and vertices $u, v \in V(\mathcal{G})$, a *temporal* u, v*-walk* is a sequence of temporal edges (for simplicity, we omit embracing parenthesis in the sequence)

$$(u = w_1, w_2, t_1, \phi_1), \ldots, (w_k, w_{k+1} = v, t_k, \phi_k)$$

such that for each i with $1 < i \leqslant k$, we have that $t_i \geqslant t_{i-1} + \phi_{i-1}$. If the vertices $w_1, \ldots, w_{k+1}$ are all distinct, this is a *path*.

The *departure time* of the walk is t_1, its *arrival time* is $t_k + \phi_k$, its *duration* is $t_k + \phi_k - t_1$, and its *travel time* is $\sum_{i=1}^{k} \phi_i$. In the following, for any time interval $[t_\alpha, t_\omega]$, we will say that the path is $[t_\alpha, t_\omega]$-*compatible* if its departure time is at least t_α and its arrival time at most t_ω. Moreover, for any vertex u, we will denote with $\mathtt{reach}^{[t_\alpha, t_\omega]}(u)$ the set of vertices v such that there exists a $[t_\alpha, t_\omega]$-compatible temporal u, v-walk. We assume the empty sequence to be a u, u-walk, and hence $u \in \mathtt{reach}^{[t_\alpha, t_\omega]}(u)$ for every t_α and t_ω. Finally, we denote the set of ordered pairs (u, v) such that $v \in \mathtt{reach}^{[t_\alpha, t_\omega]}(u)$ by $\mathcal{R}^{[t_\alpha, t_\omega]}$.

In the following, for the sake of simplicity, we omit the time interval in the superscript, whenever this interval is the one in which t_α is the minimum departure time of all temporal edges and t_ω is the maximum arrival time. Additionally, as all paths/walks considered are temporal, we omit the word.

By referring to the weighted link stream shown in Figure 1.4, the v_1, v_4-path $\pi_1 = (v_1, v_2, 1, 2), (v_2, v_3, 3, 1), (v_3, v_4, 5, 1)$ has departure time 1, arrival time 6, duration 5, and travel time 4. On the other hand, the v_1, v_4-path $\pi_2 = (v_1, v_2, 1, 2), (v_2, v_4, 3, 3)$ has the same departure, arrival, and duration times of π_1, but its travel time is 5, while the v_1, v_4-path $\pi_3 = (v_1, v_2, 2, 1), (v_2, v_3, 3, 1), (v_3, v_4, 4, 2)$ has the same arrival and duration times of π_1, but its departure time is 2 and its duration time is 4. Moreover, π_1 and π_2 are $[1, 6]$-compatible but they are not $[2, 6]$-compatible, while π_3 is both $[1, 6]$-compatible and $[2, 6]$-compatible. Finally, we have that $\mathtt{reach}(v_1) = \{v_1, v_2, v_3, v_4\}$, $\mathtt{reach}(v_2) = \{v_2, v_3, v_4\}$, $\mathtt{reach}(v_3) = \{v_3, v_4\}$, and $\mathtt{reach}(v_4) = \{v_4\}$. This implies that $|\mathcal{R}| = 10$.

2.2 Notions of Distances

By making use of the above four cost functions (that is, starting, arrival, duration, and travel times), we can then define, for any two distinct vertices u and v, the following four corresponding distances[1] from u to v, in a specific time interval $[t_\alpha, t_\omega]$. For all distances, we assume that their value is ∞, if $v \notin \mathtt{reach}^{[t_\alpha, t_\omega]}(u)$ (see Bui-Xuan, Ferreira, and Jarry (2003), Wu, Cheng, et al. (2014), and Wu, Huang, et al. (2016)).

Earliest arrival time $d_{\text{EAT}}^{[t_\alpha, t_\omega]}(u, v)$ is the minimum arrival time minus t_α of any $[t_\alpha, t_\omega]$-compatible u, v-walk.

Latest departure time $d_{\text{LDT}}^{[t_\alpha, t_\omega]}(u, v)$ is equal to t_ω minus the maximum departure time of any $[t_\alpha, t_\omega]$-compatible u, v-walk.

Fastest time $d_{\text{FT}}^{[t_\alpha, t_\omega]}(u, v)$ is the minimum duration time of any $[t_\alpha, t_\omega]$-compatible walk from u to v.

[1]Note that the term "distance" does not refer to the mathematical notion of distance or metric function, since neither the symmetry nor the triangle inequality axioms are usually satisfied.

Shortest time $d_{\text{ST}}^{[t_\alpha,t_\omega]}(u,v)$ is the minimum travel time of any $[t_\alpha,t_\omega]$-compatible walk from u to v.

For example, in the case of the weighted link stream shown in Figure 1.4, we have that $d_{\text{EAT}}(v_1,v_4) = 4$, $d_{\text{FT}}(v_1,v_4) = 3$, and $d_{\text{ST}}(v_1,v_4) = 2$ (as witnessed by the path $(v_1,v_2,2,1),(v_2,v_4,4,1)$), and we have that $d_{\text{LDT}}(v_1,v_4) = 3$ (as witnessed by the path $(v_1,v_3,3,2),(v_3,v_4,5,1)$). In Table 2.1, for all pairs of vertices (v_i,v_j) of the weighted link stream (with $i \neq j$), we show the four distances from v_i to v_j.

We leave the proof of the following simple observation as exercise.

Observation 2. *Let $\mathcal{G}$ be a link stream, and $u,v \in V(\mathcal{G})$. If $\mathcal{G}$ has a temporal u,v-walk having duration x, arrival time y, and departure time z, then $\mathcal{G}$ has a temporal u,v-path having duration at most x, arrival time at most y, and departure time at least z.*

This implies that for D $\in$ {EAT, LDT, FT}, we can assume the existence a path realising $d_{\text{D}}(u,v)$, whenever a u,v-walk exists. In the case of ST, instead, a realising walk will always be a path. Observe that the lemma only holds for positive values of ϕ, meaning that if non-strict paths is considered, then it does not hold. Notice also that, in such case, we get that the weaker Observation 2 holds for travel time.

Lemma 3. *Let $\mathcal{G}$ be a link stream, and $u,v \in V(\mathcal{G})$. If there is a temporal u,v-walk in $\mathcal{G}$, then any temporal u,v-walk realising $d_{\text{ST}}(u,v)$ is also a path.*

Proof. Let $\pi = (u = w_1, w_2, t_1, \phi_1), \dots, (w_k, w_{k+1} = v, t_k, \phi_k)$ be a temporal u,v-walk realising $d_{\text{ST}}(u,v)$, and suppose by contradiction that $w_i = w_j$, for some $i < j$. Because we consider only simple (directed) graphs, we get that $i < j + 1$. If $w_i = u$, then consider the u,v-walk starting in $(w_j, w_{j+1}, t_j, \phi_j)$; if $w_i = v$, then consider the u,v-walk finishing in $(w_{j-1}, w_j, t_{j-1}, \phi_{j-1})$; and if $w_i \notin \{u,v\}$, then consider the u,v-walk

$$(u = w_1, w_2, t_1, \phi_1), \dots, (w_{i-1}, w_i, t_{i-1}, \phi_{i-1}), (w_j = w_i, w_{j+1}, t_j, \phi_j), \dots,\\ (w_k, w_{k+1} = v, t_k, \phi_k),$$

in other words, the walk obtained from π by removing the w_i, w_j-walk contained in π. Because $\phi_\ell \geqslant 1$ for every ℓ, in each case we end up with a temporal u,v-walk with smaller travel time, a contradiction. □

As noticed in Wu, Cheng, et al. (2014), best paths according to all the distances considered here, i.e. the ones realising the distance, have not optimal substructure.

Observation 4. *For any D $\in$ {EAT, LDT, FT, ST}, any subpath of a path realising D is not necessarily a path realising D.*

In Figure 2.1, we give an example of a link stream where every temporal u,v-walk P realizing $d_{\text{FT}}(u,v)$ is such that the x,y-path contained in P does not realize $d_{\text{FT}}(x,y)$, for every x,y in P such that $\{x,y\} \neq \{u,v\}$. Observe that such property is even stronger than what is said above. We leave similar constructions for D $\in$ {LDT, ST} as exercise. As for EAT, such an example is not possible, as the following lemma holds.

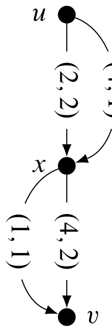


Figure 2.1: In this example $d_{\text{FT}}(u, v) = 4$ but the edges used to realize this distance do not realize $d_{\text{FT}}(u, x) = 1$ and $d_{\text{FT}}(x, v) = 1$.

	d_{EAT}				d_{LDT}				d_{FT}				d_{ST}			
	v_1	v_2	v_3	v_4	v_1	v_2	v_3	v_4	v_1	v_2	v_3	v_4	v_1	v_2	v_3	v_4
v_1	–	2	3	4	–	4	3	3	–	1	2	3	–	1	2	2
v_2	∞	–	3	4	∞	–	3	2	∞	–	1	1	∞	–	1	1
v_3	∞	∞	–	5	∞	∞	–	1	∞	∞	–	1	∞	∞	–	1
v_4	∞	∞	∞	–	∞	∞	∞	–	∞	∞	∞	–	∞	∞	∞	–

Table 2.1: The four distances from v_i to v_j for all pairs of vertices (v_i, v_j) with $i \neq j$ in the weighted link stream of Figure 1.4, when $t_\alpha = 1$ and $t_\omega = 6$.

Lemma 5 (Wu, Cheng, et al. (2014)). *Let $\mathcal{G}$ be a link stream, and $u, v \in V(\mathcal{G})$. If u reaches v in the time interval $[t_\alpha, t_\omega]$, then there is an earliest arrival path traversing $u = v_1, \ldots, v_k = v$ in this order such that every prefix subpath, i.e. from v_1 to v_i for every $i \in [k]$, is an earliest arrival path.*

In order to prove Theorem 5, by contradiction suppose that there is no such path, i.e. for every path from u to v traversing $u = v_1, \ldots, v_k = v$ not every prefix is an earliest arrival path. Let us consider one earliest arrival path traversing $u = v_1, \ldots, v_k = v$ and let v_j be the rightmost vertex in this path for which we are not realizing an earliest arrival path. Then we can replace the path traversing $v_1, \ldots, v_j$ with an earliest arrival path from $u = v_1$ to v_j and still obtain an earliest arrival path from u to v. We can hence repeat the process, finally finding an earliest arrival path from u to v such that every prefix is an earliest arrival path.

2.3 Computing Distances

In this section, we present some algorithms to compute distances in temporal graphs, and as previously said, we will use sorted link streams. Several algorithms have been proposed in the literature in order to compute the distance from a source vertex s to all other vertices, for each of the distance definitions considered here. For any $\textsc{d} \in \{\textsc{eat}, \textsc{ldt}, \textsc{ft}, \textsc{st}\}$, let $\texttt{ssbp}_{\textsc{d}}$ be a *single source best path* algorithm that, given a link stream $(V, \mathbb{E}_{\downarrow})$ and a source vertex u, returns an array containing, for each vertex v, the value $d_{\textsc{d}}(u, v)$, and let $\textsc{ss-time}_{\textsc{d}}$ (respectively, $\textsc{ss-space}_{\textsc{d}}$) be the worst-case time (respectively, space) complexity of this single source best path algorithm, as a function of the number n of vertices and of the number m of temporal edges in the link stream. In this book, we mostly refer to the algorithms proposed in Crescenzi, Magnien, and Marino (2019, 2020), Wu, Cheng, et al. (2014), and Wu, Huang, et al. (2016), which are the ones with best time and space complexity so far (see the second and third column of Table 2.2). We observe that if the temporal graph is not given as a link stream, then all time complexities of Table 2.2 become $O(m \log m)$, as the temporal edges need to be ordered according to their starting time.

We also define a "backward" variation of the single source best path algorithms. For any $\textsc{d} \in \{\textsc{eat}, \textsc{ldt}, \textsc{ft}, \textsc{st}\}$, let $\texttt{stbp}_{\textsc{d}}$ be a *single target best path* algorithm that, given input $\mathbb{E}_{\uparrow}$ and a target vertex v, returns an array containing, for each vertex u, the value $d_{\textsc{d}}(u, v)$, and let $\textsc{st-time}_{\textsc{d}}$ (respectively, $\textsc{st-space}_{\textsc{d}}$) be the worst-case time (respectively, space) complexity of this algorithm, as a function of the number n of vertices and of the number m of temporal edges in the link stream. Once again, here we mostly refer to the algorithms proposed in Crescenzi, Magnien, and Marino (2019, 2020), Wu, Cheng, et al. (2014), and Wu, Huang, et al. (2016), as they have the best time and space complexity so far (see the fifth and sixth column of Table 2.2).

2.3.1 Two useful constructions

In the following, we present two useful link stream transformations. In Section 1.5, we already saw the *undelayed version* construction that transforms the more general temporal graph (G, α, ϕ) in a simpler temporal graph with no duration on the edges. We define an analogous construction, but for link stream and in a way that all edges in the obtained link stream have travel time equal to 1. So consider a link stream $(V, \mathbb{E})$; we construct a link stream $(V \cup I, \mathbb{F})$, where I is a set of at most $|\mathbb{E}|$ *intermediate* vertices. Intuitively, this transformation changes the travel time of a temporal edge into a waiting time in the corresponding intermediate vertex. More precisely (see Figure 2.2), for each temporal edge $e = (u, v, t, 1) \in \mathbb{E}$, e is also included in $\mathbb{F}$. For each temporal edge $e = (u, v, t, \phi) \in \mathbb{E}$ with $\phi > 1$, a new vertex i_e is inserted in I, and the two temporal edges $(u, i_e, t, 1)$ and $(i_e, v, t + \phi - 1, 1)$ are included in $\mathbb{F}$. It is easy to verify that, for any two vertices $u, v \in V$, $d_{\textsc{d}}(u, v)$ in $(V, \mathbb{E})$ is equal to $d_{\textsc{d}}(u, v)$ in $(V \cup I, \mathbb{F})$, where $\textsc{d} \in \{\textsc{eat}, \textsc{ldt}, \textsc{ft}\}$. We call the obtained link stream the *delay-1 of* $(V, \mathbb{E})$. This construction has been introduced in Crescenzi, Magnien, and Marino (2019).

The second useful construction can be seen as reversing the time flow of a link stream. So again consider a link stream $\mathcal{G} = (V, \mathbb{E})$, and let τ be the lifetime of $\mathcal{G}$. Let $(V, \mathbb{F})$ be the link stream obtained by adding to $\mathbb{F}$ the temporal edge $\rho(e) = (v, u, \tau + 1 - (t + \phi), \phi)$, for each temporal edge $e = (u, v, t, \phi) \in \mathbb{E}$. We call $(V, \mathbb{F})$ the *reverse of* $\mathcal{G}$. The analogous

Table 2.2: The time and space complexity of the single source best path (upper part) and the single target best path (lower part) algorithms with input $\mathbb{E}_\downarrow$ and $\mathbb{E}_\uparrow$, respectively (without considering the time and space necessary for sorting the link stream).

D	SS-TIME$_D$	SS-SPACE$_D$	Ref.
EAT	$O(m)$	$O(n)$	Wu, Cheng, et al. (2014) Wu, Huang, et al. (2016)
LDT	$O(m)$	$O(m)$	Crescenzi, Magnien, and Marino (2019) Crescenzi, Magnien, and Marino (2020)
FT	$O(m)$	$O(m)$	Crescenzi, Magnien, and Marino (2019) Wu, Cheng, et al. (2014) Wu, Huang, et al. (2016)
ST	$O(m \log m)$	$O(m)$	Wu, Cheng, et al. (2014) Wu, Huang, et al. (2016)
D	ST-TIME$_D$	ST-SPACE$_D$	Ref.
EAT	$O(m)$	$O(m)$	Crescenzi, Magnien, and Marino (2019) Crescenzi, Magnien, and Marino (2020)
LDT	$O(m)$	$O(n)$	Wu, Cheng, et al. (2014) Wu, Huang, et al. (2016)
FT	$O(m)$	$O(m)$	Crescenzi, Magnien, and Marino (2019) Wu, Cheng, et al. (2014) Wu, Huang, et al. (2016) & Theorem 6
ST	$O(m \log m)$	$O(m)$	Wu, Cheng, et al. (2014) Wu, Huang, et al. (2016) & Theorem 6

construction on the simpler model (G, λ) has been used in Ibiapina and Silva (2022).

The following lemma allows us to easily design a backward version of a best path search and to relate the EAT distance and the LDT distance (see Figure 2.3).

Lemma 6. *Consider a link stream $(V, \mathbb{E})$ with lifetime τ, its reverse $(V, \mathbb{F})$, and $D \in \{FT, ST\}$. Then, for any two vertices $u, v \in V$ and for any time interval $[t_\alpha, t_\omega]$, we have that $d_D^{[t_\alpha,t_\omega]}(u, v)$ in $(V, \mathbb{E})$ is equal to $d_D^{[\tau+1-t_\omega,\tau+1-t_\alpha]}(v, u)$ in $(V, \mathbb{F})$. Moreover, $d_{EAT}^{[t_\alpha,t_\omega]}(u, v)$ in $(V, \mathbb{E})$ is equal to $d_{LDT}^{[\tau+1-t_\omega,\tau+1-t_\alpha]}(v, u)$ in $(V, \mathbb{F})$, and $d_{LDT}^{[t_\alpha,t_\omega]}(u, v)$ in $(V, \mathbb{E})$ is equal to $d_{EAT}^{[\tau+1-t_\omega,\tau+1-t_\alpha]}(v, u)$ in $(V, \mathbb{F})$.*

Proof. In order to prove the first assertion, it suffices to show that there exists a $[t_\alpha, t_\omega]$-compatible u, v-path in $(V, \mathbb{E})$ whose duration (respectively, travel time) is d if and only if there exists a $[\tau + 1 - t_\omega, \tau + 1 - t_\alpha]$-compatible u, v-path in $(V, \mathbb{F})$ whose duration (respectively, travel time) is d. Let $P = e_1e_2 \dots e_k$ be a $[t_\alpha, t_\omega]$-compatible path from u to v in $(V, \mathbb{E})$, and write $e_i = (u_i, v_i, t_i, \phi_i)$ for each $i \in [k]$. Recall that the reverse of e_i is $\rho(e_i) = (v_i, u_i, \tau + 1 - (t_i + \phi_i), \phi_i)$. Hence the starting time of the reverse path $\rho(P) = \rho(e_k)\rho(e_{k-1}) \dots \rho(e_1)$ is $\tau + 1 - (t_k + \phi_k) \geqslant \tau + 1 - t_\omega$, as $(t_k + \phi_k) \leqslant t_\omega$, while the arrival time is $\tau + 1 - (t_1 + \phi_1) + \phi_1 \leqslant \tau + 1 - t_\alpha$ as $t_1 \geqslant t_\alpha$. Hence $\rho(P)$

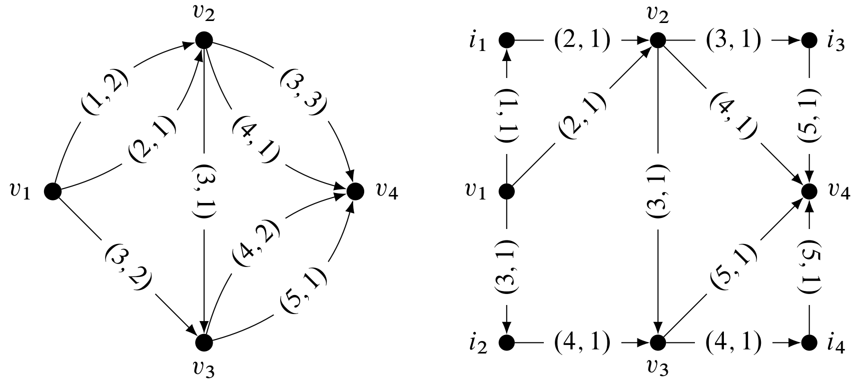


Figure 2.2: On the right, the delay-1 of the link stream on the left.

is within the window $[\tau + 1 - t_\omega, \tau + 1 - t_\alpha]$, as desired. One can also see that $\rho(P)$ is a v_k, v_1-walk in $(V, \mathbb{F})$, and that the duration and travel time of $\rho(P)$ is the same as P. Observe that this is enough since the reverse of $(V, \mathbb{F})$ is equal to $(V, \mathbb{E})$.

In order to prove the second assertion, again because the reverse of $(V, \mathbb{F})$ is equal to $(V, \mathbb{E})$, it suffices to show that there exists a $[t_\alpha, t_\omega]$-compatible u, v-path in $(V, \mathbb{E})$ whose arrival time is d if and only if there exists a $[\tau+1-t_\omega, \tau+1-t_\alpha]$-compatible u, v-path in $(V, \mathbb{F})$ whose departure time is $\tau + 1 - d$. Pick P and $\rho(P)$ as in the previous paragraph. We already know that $\rho(P)$ is a $[\tau + 1 - t_\omega, \tau + 1 - t_\alpha]$-compatible v_k, v_1-path in $(V, \mathbb{F})$. The arrival time of P is $t_k + \phi_k$, while the departure time of $\rho(P)$ is $\tau + 1 - (t_k + \phi_k)$, as we wanted. The opposite direction can be proved similarly, and this concludes the proof of the lemma. □

Observe that the above lemma gives us that computing single target best paths for D $\in$ {FT, ST} is equivalent to computing single source best paths on the reverse of the given link stream, which can be obtained in linear time. Additionally, computing single source/target best paths for LDT is equivalent to computing single target/source best paths for EAT. Hence, in order to compute all possible values, it is enough to solve single source best paths for D $\in$ {EAT, FT, ST} and single target best path for EAT (as the latter algorithm can be also used for single source best path for LDT).

2.3.2 Single Source Earliest Arrival Paths

Given a link stream $(V, \mathbb{E}_\downarrow)$, a vertex $x \in V$ and a time interval $[t_\alpha, t_\omega]$, we show how to compute earliest arrival time distances from x to every vertex $v \in V$ within $[t_\alpha, t_\omega]$ in Algorithm 1. Algorithm 1 processes each edge in $\mathbb{E}_\downarrow$ and keeps track of current earliest-arrival times from x to every vertex v that has been seen until then by means of an array $t[v]$. Thanks to the condition in Line 4, $t[v]$ will be updated with the smallest arrival

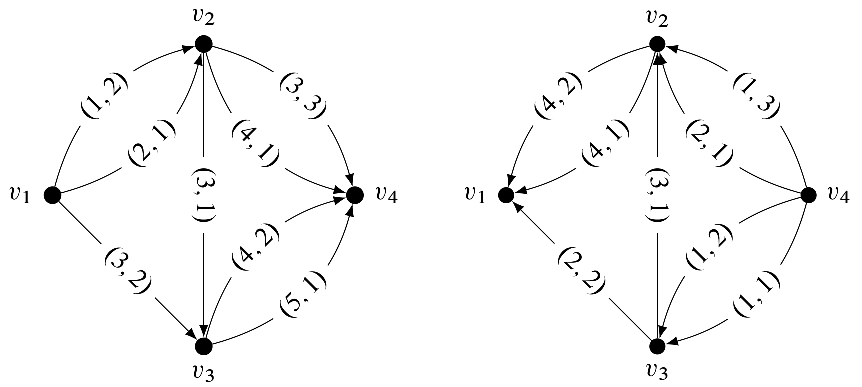


Figure 2.3: The transformation from a link stream to another link stream to be used in order to obtain a single target best algorithm starting from a single source best path one. For instance, the duration time of the path $(v_1, v_3, 3, 2), (v_3, v_4, 5, 1)$ in the original link stream is equal to $(5 + 1) - 3 = 3$, and the duration time of the path $(v_4, v_3, 1, 1), (v_3, v_1, 2, 2)$ in the transformed link stream is equal to $(2 + 2) - 1 = 3$.

time of any path from u to v within the time interval $[t_\alpha, t]$. In particular, for each edge $e = (u, v, t, \phi)$ in $\mathbb{E}_\downarrow$ the algorithm checks if e is compatible with the time constraint of a temporal path within $[t_\alpha, t_\omega]$ (i.e. whether $t + \phi \leqslant t_\omega$ and $t \geqslant t[u]$). If the time constraints are respected, $t[v]$ will be updated in case $t + \phi$ is smaller than $t[v]$. The algorithm ends when all edges in $\mathbb{E}_\downarrow$ have been processed or when we meet one edge that has starting time greater than or equal to t_ω. The correctness of the algorithm is proven in Wu, Cheng, et al. (2014) and basically follows from Theorem 5.

2.3.3 Single Target Earliest Arrival Paths, assuming unitary weights

Now, for the single target best paths algorithm, we assume that all the travel times of the input graph are 1 by working on the delay-1 version of the given link stream (see Figure 2.2).

We now show a sort of "reverse" version of Algorithm 1 that is applied to a destination vertex d, and that will allow us to compute, for any other vertex x, the earliest arrival time from x to d. Such algorithm can be seen as an adaptation of the earliest arrival profile algorithms proposed in Dibbelt et al. (2018), and has been proposed in Crescenzi, Magnien, and Marino (2020) in order to approximate the temporal closeness, a centrality measure for temporal graphs. Differently from the case of Algorithm 1, we consider a link stream $(V, \mathbb{E}_\uparrow)$, i.e. the temporal edges are sorted in non-increasing order with respect to their starting times. We assume that the appearing times of all temporal edges are distinct, and in the next section we show how the algorithm can be adapted to the more general case. We also assume that the temporal graph is directed, and if this not the case, then we simply

Algorithm 1: ssbp$_{\text{EAT}}$ Wu, Cheng, et al. (2014)

```
Input  : Link stream $(V, \mathbb{E}_{\downarrow})$, source vertex $x$, time interval $[t_\alpha, t_\omega]$
Output : $d_{\text{EAT}}(x, v)$ for each $v \in V$ within $[t_\alpha, t_\omega]$
1 Initialize $t[x] = t_\alpha$, and $t[v] = \infty$ for all $v \in V \setminus \{x\}$
2 foreach $e = (u, v, t, \phi)$ in the link stream do
3     if $t + \phi \leqslant t_\omega$ and $t \geqslant t[u]$ then
4         if $t + \phi < t[v]$ then
5             $t[v] \leftarrow t + \phi$
6     else if $t \geqslant t_\omega$ then
7         Break the for-loop and go to line 8
8 return $t[v]$ for each $v \in V$
```

have to examine each edge twice by inverting the source and the destination. The algorithm maintains, for each vertex $x \in V$, a triple $\ell[x] = (l_x, r_x, s_x)$ which indicates that if we want to be at the destination d in any time instant t in $[l_x, r_x)$, then the latest departure time from x is at most s_x, i.e. we have to leave no later than s_x from x (see Algorithm 2). More formally, we have that $d_{\text{LDT}}^{[l_x, r_x - 1]}(x, d) \leqslant s_x$. At the beginning, we do not know anything about the reachability of d from a vertex x; hence, we set the starting time of x equal to ∞ for an arbitrary time interval (for example, $[t_\omega + 1, t_\omega + 2)$) following t_ω (line 3). When we read a new temporal edge $(x, y, t, t + 1)$, we first set $\ell[d]$ to $(t, t + 1, t + 1)$, since, clearly, the destination vertex can always reach itself even starting after the appearance of the edge (line 7). Let $\ell[x] = (l_x, r_x, s_x)$ and $\ell[y] = (l_y, r_y, s_y)$ be the two triples associated with x and y, respectively. If $l_x > l_y$, then we update the triple associated with x by setting $\ell[x]$ to (l_y, l_x, t) (line 12). This update is justified by Theorem 7.

Algorithm 2: stbp$_{\text{EAT}}$ Crescenzi, Magnien, and Marino (ibid.)

```
Data: Link stream $(V, \mathbb{E}_{\uparrow})$ whose travel-time of every edge is equal to 1, a
      vertex $d \in V$, and a time interval $[t_\alpha, t_\omega]$.
Result: Earliest arrival time from each vertex to $d$.
1  for $x \leftarrow 1$ to $|V|$ do
3      $\ell[x] = (t_\omega + 1, t_\omega + 2, \infty)$; $S[x] \leftarrow \infty$;
4  while there are other edges to be read do
5      let $e \leftarrow (x, y, t, t + 1)$ be the next edge;
7      $\ell[d] \leftarrow (t, t + 1, t + 1)$; $S[d] \leftarrow t$;
8      $(l_x, r_x, s_x) \leftarrow \ell[x]$;
9      $(l_y, r_y, s_y) \leftarrow \ell[y]$;
10     if $l_y < l_x$ then
12         $\ell[x] \leftarrow (l_y, l_x, t)$; $S[x] \leftarrow s_x$;
13 return $l_x$ for each $x \in V$, where $(l_x, r_x, s_x)$ is $\ell[x]$.
```

Lemma 7. *Let $\mathcal{G} = (V, \mathbb{E}_{\uparrow})$ be a link stream and $d \in V$. For any $u \in V$ with $u \neq d$, let $\Xi_u = \langle \tau_{u,1}, \tau_{u,2}, \ldots, \tau_{u,h_u}, \tau_{u,h_u+1} \rangle$ be the sequence of triples $\tau_{u,i} = (l_{u,i}, r_{u,i}, s_{u,i})$ such that $l_{u,h_u+1} = t_\omega + 1$, $r_{u,h_u+1} = t_\omega + 2$, $s_{u,h_u+1} = \infty$, and, for $1 \leqslant i \leqslant h_u$,*

$(l_{u,i}, r_{u,i}, s_{u,i})$ is the triple assigned to $\tau[u]$ at the (h_u+1-i)-th execution of line 12 with $x = u$ during the running of Algorithm 2 with input G and d (note that $h_u = 0$ if this line is never executed with $x = u$). Then, for any $u \in V$ with $u \neq d$, the intervals $[l_{u,i}, r_{u,i})$, for $i \leqslant i \leqslant h_u + 1$, form a partition of the interval $[l_{u,1}, t_\omega + 2)$, and, for any $t \in [t_\alpha, t_\omega]$,

$$d_{\mathit{EAT}}^{[t,t_\omega]}(u,d) = \begin{cases} r_{u,i} - t + 1 & \textit{if } s_{u,i} < t \leqslant s_{u,i+1} \textit{ with } 1 \leqslant i \leqslant h_u, \\ \infty & \textit{otherwise.} \end{cases}$$

Proof. We prove the lemma by induction on the number k of temporal edges that have been read. In particular, for any k with $0 \leqslant k \leqslant |E|$, let $\mathcal{S}(k)$ be the following statement.

> For any $u \in V$ with $u \neq s$, let $\Xi_u^k = \langle \tau_{u,h_u^k}, \ldots, \tau_{u,h_u+1}\rangle$ be the suffix of Ξ_u containing the triples assigned to $\tau[u]$ at line 12 with $x = u$ after having read k edges. The intervals $[l_{u,i}, r_{u,i})$, for $h_u^k \leqslant i \leqslant h_u + 1$, form a partition of the interval $[l_{u,h_u^k}, t_\omega + 2)$, and, for any $t \in [l_{u,h_u^k}, t_\omega]$, if $s_{u,i} < t \leqslant s_{u,i+1}$ then $d_{\text{EAT}}^{[t,t_\omega]}(u,d) = r_{u,i} - t + 1$.

We now prove by induction on k that $\mathcal{S}(k)$ is true for any k with $0 \leqslant k \leqslant |E|$.

Base case. $k = 0$. In this case, no edge has been read yet and, hence, line 12 has never been executed with $x = u$. We then have that, for any $u \in V$ with $u \neq d$, $h_u^0 = h_u + 1$, $\Xi_u^0 = \langle \tau_{u,h_u+1}\rangle$ with $\tau_{u,h_u+1} = (t_\omega + 1, t_\omega + 2, \infty)$, and, hence, $[l_{u,h_u+1}, r_{u,h_u+1}) = [t_\omega + 1, t_\omega + 2) = [l_{u,h_u^0}, t_\omega + 2)$. Moreover, the interval $[l_{u,h_u^0}, t_\omega] = [t_\omega + 1, t_\omega]$ is empty and the condition on the t-distances is "vacuously" true. Hence, $\mathcal{S}(0)$ is true.

Induction step. Given k with $1 \leqslant k \leqslant |E|$, suppose that $\mathcal{S}(k-1)$ is true. We now prove that $\mathcal{S}(k)$ is also true. Let $e = (x, y, t, t+1)$ be the k-th temporal edge read by the algorithm. Clearly, this edge has no influence on any other vertex than x (since the graph is directed). Hence, we have just to prove that the value of $\tau[x]$ is correctly updated. By the induction hypothesis, we know that the current value of $\tau[x] = (l_x, r_x, s_x)$ is such that, for any $t' \in [l_x, t_\omega]$, the starting time of any latest starting t'-path from x to d is at least $s_x > t$. Hence, the edge e cannot improve these starting times since its appearing time is t. Analogously, we know that the current value of $\tau[y] = (l_y, r_y, s_y)$ is such that $s_y > t$ is the starting time of any latest starting t'-path from y to d with $t' \in [l_y, r_y)$. If $l_y \geqslant l_x$, the edge e does not add any information for the vertex x, since we already know the starting time of any latest starting t'-path from x to d, for any $t' \geqslant l_x$. On the contrary (see Figure 2.4), if $l_y < l_x$, then, for any time instant $t' \in [l_y, l_x)$ (for which we did not know yet the corresponding latest starting time from x), we can now say that we can first reach y (at time t with $t < s_y \leqslant l_y$ by using the temporal edge e), and then wait until starting the path from y to d at time s_y: hence, for all these time instants, the latest starting time at x can now be set equal to t, that is, the value of $\tau[x]$ becomes (l_y, l_x, t) (note that subsequent edges cannot improve this value since their appearing times are smaller than t). Hence, if $\Xi_y^{k-1} = \langle \tau_{x,h_x^{k-1}}, \ldots, \tau_{u,h_x+1}\rangle$, we have that $\Xi_x^k = \langle \tau_{x,h_x^k}, \tau_{x,h_x^{k-1}}, \ldots, \tau_{u,h_x+1}\rangle$ with $h_x^k = h_x^{k-1} - 1$ and $\tau_{u,h_x^k} = (l_y, l_x, t)$. By the induction hypothesis, the intervals $[l_{x,i}, r_{x,i})$, for $h_x^{k-1} \leqslant i \leqslant h_x$, form a partition of the interval $[l_{x,h_x^{k-1}}, t_\omega + 2)$: by adding

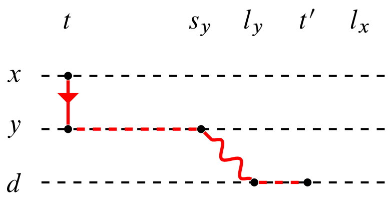


Figure 2.4: The update rule of the "backward" version of the temporal breadth-first search algorithm.

the triple (l_y, l_x, t), we obtain a partition of the interval $[l_{x,h_x^k}, t_\omega + 2)$ (since $l_x = l_{x,h_x^k - 1}$ and $l_y = r_{x,h_x^k}$). From the previous argument, it also follows that, for any $t' \in [l_{x,h_x^k}, t_\omega]$, if $s_{x,i} < t' \leqslant s_{x,i+1}$ then $d_{t'}(x,d) = r_{x,i} - t' + 1$. We have thus proved that $\mathcal{S}(k)$ is satisfied.

The lemma follows from the fact that its statement is exactly equivalent to $\mathcal{S}(|E|)$. □

How to Deal with Multiple Edges

In the previous sections, we have assumed that, for each time t in the interval $[t_\alpha, t_\omega]$, there exists at most one edge whose appearing time is equal to t. Clearly, this assumption is not realistic since, in the vast majority of real-world temporal graphs, many edges can appear at the same time. In this section, we show how we can modify Algorithm 2, in order to deal with this more general case. For the sake of clarity, we will assume that, for each vertex u, the algorithm maintains a list I_u of triples (instead of just one triple). However, it is not hard to show that only the last two triples are really necessary at each iteration of the algorithm, thus assuring that the algorithm itself has linear space complexity.

Consider the iteration of temporal edge $e = (x, y, t, t+1)$, and let (l_x, r_x, s_x) (respectively, (l_y, r_y, s_y)) be the last triple inserted in I_x (respectively, I_y). This implies that that if we want to arrive at the destination d in the interval $[l_x, r_x)$ (respectively, $[l_y, r_y)$), then we cannot start from x (respectively, y) later than s_x (respectively, s_y). Remember that, in Algorithm 2, the temporal edges are scanned in non-increasing order with respect to their appearing times; hence, we know that $t \leqslant s_x \leqslant l_x < r_x$ (respectively, $t \leqslant s_y \leqslant l_y < r_y$). We now distinguish the following cases.

1. $t < s_x$ and $t < s_y$. In this case, neither x nor y has yet used an edge at time t. Hence, we can update the set of intervals as we did in the case of edges with distinct appearing times. That is, if $l_y < l_x$, then add to I_x the triple (l_y, l_x, t).

2. $t < s_x$ and $t = s_y$. In this case, y has already "encountered" an edge at time t. Let (l'_y, r'_y, s'_y) be the triple just before (l_y, r_y, s_y) in I_y (note that $l'_y = r_y$ and that $t = s_y < s'_y$). If $l'_y < l_x$, then we add to I_x the triple (l'_y, l_x, t): indeed, since

$t < s'_y$, we now know that, to arrive at d in the interval $[l'_y, l_x)$, we can start from u at time t (by using the edge e), wait until time s'_y, and then follow the journey from y to d.

3. $t = s_x$ and $t < s_y$. In this case, x has already "encountered" an edge at time t. If $l_y < l_x$, then we extend to the left the triple of x until l_y: indeed, since $s_x < s_y$, we now know that, even to arrive at d in the interval $[l_y, l_x)$, we can start at time t (by using the edge e), wait until time s_y, and then follow the journey from y to d.

4. $t = s_x$ and $t = s_y$. In this case, both x and y have already "encountered" an edge at time t. Let (l'_y, r'_y, s'_y) be the triple just before (l_y, r_y, s_y) in I_y (note that $l'_y = r_y$ and $t = s_y < s'_y$). Similarly to the previous case, if $l'_y < l_x$, then we extend to the left the triple of x until l'_y.

2.3.4 Single Source: Fastest and Shortest Time

In this section we show how to compute $\texttt{ssbp}_{\text{FT}}$ and $\texttt{ssbp}_{\text{ST}}$ in log linear time. For clarity purposes, the algorithm we present here does not work in a streaming fashion (i.e., sorted link stream) like the ones we have provided for $\texttt{ssbp}_{\text{EAT}}$ and $\texttt{stbp}_{\text{EAT}}$. A better algorithm which works in streaming can be found in Wu, Cheng, et al. (2014). Moreover, in the cited paper, the algorithm given for FT is linear and, hence, meets the bound shown in Table 2.2.

The approach we follow here uses a variation of the strict static expansion introduced in the previous chapter (see Section 1.5). We do some modifications to this transformation, in order to make it linear on the size of the link stream and to deal with travel time on the edges higher than 1. Indeed, given a temporal graph on n vertices and lifetime τ, note that the strict static expansion in Section 1.5 requires τ copies of the vertices appearing in the temporal graph, independently of how many temporal edges there are. Hence, if we have only two temporal edges in the temporal graph, one at time 1 and one at time τ, then the strict static expansion has $O(n\tau)$ vertices and edges instead of $O(1)$.

The *static expansion* of a link stream $\mathcal{G} = (V, \mathbb{E})$ is the DAG $SE(\mathcal{G}) = (V', E')$, where:

- $V' = \cup_{(u,v,s,\phi)\in\mathbb{E}}\{(u,s),(v,s+\phi)\}$
- $E' = M \cup W$, where M and W are defined next.
- $M = \{((u,s),(v,s+\phi)) : (u,v,s,\phi) \in \mathbb{E}\}$. These are called *moving edges*.
- For every vertex $v \in V$, let $((v,t_1),\ldots,(v,t_k))$ be the ordered sequence of pairs in V' having as first component v. Then $W = \{((v,t_i),(v,t_{i+1})) : i \in [k-1]\}$. These are called *waiting edges*.

Observe that $SE(\mathcal{G}) = (V', E')$ has linear size on the number of temporal edges m in $\mathbb{E}$, as it has $O(m)$ vertices and $O(m)$ edges.

Now, let us suppose that we want to compute $\texttt{ssbp}_{\text{FT}}$ from a vertex r in V. To this aim we define a suitable weight function for the edges of $SE(\mathcal{G}) = (V', E')$ and run

Dijkstra's algorithm for computing lightest paths in the resulting weighted DAG.[2] The weight function of an edge is defined as follows.

- For each edge $e = ((u, s), (v, s + \phi))$ in M, we set $w_M(e) = \phi$.
- For each edge $e = ((v, t_i), (v, t_{i+1}))$ in W, we set $w_W(e) = t_{i+1} - t_i$ if $v \neq r$; otherwise, we set $w_W(e) = 0$ otherwise.

Analogously, if we want to compute $\texttt{ssbp}_{\text{ST}}$ from a vertex r, we define the following weight function for the edges of $SE(\mathcal{G}) = (V', E')$. For the edges in M, we use the same w_M defined above, while, for each edge e in W, we set $w'_W(e)$ to 0.

Running Dijkstra's algorithm in $SE(\mathcal{G}) = (V', E')$ with the weights w_M and w_E, and starting from (r, t_α), we obtain for each vertex $(u, t) \in V'$ a value of distance corresponding to the minimum path from (r, t_α)[3], denoted as $d'((r, t_\alpha), (u, t))$. Hence,

$$d_{\text{FT}}(r, u) = \min_{(u,t)\in V'} d'((r, t_\alpha), (u, t)).$$

Denoting by $d''(\cdot)$ the distance in $SE(\mathcal{G}) = (V', E')$ with the edges weighted using w_M and w'_W, we obtain:

$$d_{\text{ST}}(r, u) = \min_{(u,t)\in V'} d''((r, t_\alpha), (u, t)).$$

We leave the correctness of such algorithms as exercise.

2.4 Computing Diameter

Once a definition of distance is adopted, the corresponding notions of eccentricity and of diameter can be introduced, analogously to the case of standard graphs. That is, given a link stream $\mathcal{G} = (V, \mathbb{E})$, for each $\text{D} \in \{\text{EAT}, \text{LDT}, \text{FT}, \text{ST}\}$, the (forward) *eccentricity* $\text{ECCF}_{\text{D}}(u)$ of a vertex u is its maximum finite distance to any other vertex, and the *diameter* $\mathcal{D}_{\text{D}}(\mathcal{G})$ of $\mathcal{G}$ is the maximum eccentricity of the vertices whose eccentricity is defined (i.e., finite). For example, in the case of the link stream shown in Figure 1.4 and by referring to the values of Table 2.1, we have that $\text{ECCF}_{\text{EAT}}(v_1) = 4$, $\text{ECCF}_{\text{EAT}}(v_2) = 4$, $\text{ECCF}_{\text{EAT}}(v_3) = 5$, and, hence, $\mathcal{D}_{\text{EAT}}(\mathcal{G}) = 5$, while $\mathcal{D}_{\text{LDT}}(\mathcal{G}) = 4$, $\mathcal{D}_{\text{FT}}(\mathcal{G}) = 3$, and $\mathcal{D}_{\text{ST}}(\mathcal{G}) = 2$.

We denote by $\text{ECCB}_{\text{D}}(u)$ the *backward eccentricity* of a vertex u, that is its maximum finite distance from any other vertex. Note that the diameter $\mathcal{D}_{\text{D}}(\mathcal{G})$ of a link stream can also be defined as the maximum (defined) backward eccentricity of all its vertices. For example, in the case of the link stream shown in Figure 1.4 and by referring to the values

[2]Dijkstra's Algorithm computes the lightest path from a node to all the other nodes in a weighted graph with non-negative weights in log linear time wrt the size of the graph (number of nodes plus number of edges). See Cormen et al. (2022).

[3]Wlog, we can suppose that there is at least one temporal edge leaving r at time t_α.

of Table 2.1, we have that $\text{ECCB}_{\text{EAT}}(v_2) = 2$, $\text{ECCB}_{\text{EAT}}(v_3) = 3$, $\text{ECCB}_{\text{EAT}}(v_4) = 5$, and, hence, $\mathcal{D}_{\text{EAT}}(\mathcal{G}) = 5$, as we already noticed.

For any $\text{D} \in \{\text{EAT}, \text{LDT}, \text{FT}, \text{ST}\}$, in order to compute the corresponding diameter $\mathcal{D}_{\text{D}}(\mathcal{G})$, we can execute the algorithm $\texttt{ssbp}_{\text{D}}$, for each source vertex u; we refer to such "text-book" approach as Algorithm TB_{D}. However, the time complexity of this approach is $O(n \cdot \text{SS-TIME}_{\text{D}}(n, m))$. By looking at Table 2.2, we have that this time complexity is not affordable whenever we have thousands or millions of vertices, and millions or billions of temporal edges in the link stream. Unfortunately, in the following section, we show that it is very unlikely that there exists an algorithm computing any of the four diameters in time sub-quadratic in the number of temporal edges. In other words, it is reasonable to conjecture that the known algorithms for computing any of the four diameters are, indeed, optimal. An algorithm that is often linear in practice when computing the diameter of real-world link streams has been proposed in Calamai, Crescenzi, and Marino (2021).

2.4.1 On the complexity of computing diameters

In this section, we present a conditional lower bound on the complexity of computing the diameter of a link stream proven in Calamai, Crescenzi, and Marino (ibid.).

The *Strong Exponential Time Hypothesis* (in short, *SETH*) states that there is no algorithm to solve k-SAT in time $O((2-\epsilon)^n)$, where $\epsilon > 0$ does not depend on k (see Impagliazzo, Paturi, and Zane (2001)). This hypothesis has been repeatedly used in the past few years in order to prove lower bounds for the time complexity of algorithms for some polynomial-time solvable problem. See, for example, Williams and Williams (2010), which is one of the first papers along this line of research, where the authors give bounds of many problems, like computing all pairs shortest paths and finding triangles in a graph. We use the same approach here in order to prove that the diameter of a link stream cannot be computed in time sub-quadratic on the number of temporal edges, even if the travel-time of each edge is equal to 1.

To this aim, we will refer to the *k-Two Disjoint Sets* (in short, *k-TDS*) problem, which is defined below.

k-TDS
Input. A set X and a collection $\mathcal{C}$ of subsets of X such that $|X| < \log^k(|\mathcal{C}|)$.
Question. Are there two disjoint sets in $\mathcal{C}$?

Clearly, this problem can be solved in quadratic time, up to poly-logarithmic factors. It is also known that, for any k, the k-TDS problem is not solvable in time $\widetilde{O}(|\mathcal{C}|^{2-\epsilon})$, unless the SETH fails (see Borassi, Crescenzi, and Habib (2016)), where the $\widetilde{O}$ notation ignores poly-logarithmic factors.

Theorem 8. *Given a link stream $\mathcal{G} = (V, \mathbb{E})$, for any $\textit{D} \in \{\textit{EAT}, \textit{FT}, \textit{ST}\}$, computing the diameter $\mathcal{D}_{\textit{D}}(\mathcal{G})$ cannot be done in time $\widetilde{O}(|\mathbb{E}|^{2-\epsilon})$ for any $\epsilon > 0$, unless SETH fails, even if the link stream is delay-1 and undirected, and the diameter is either 2 or 3.*

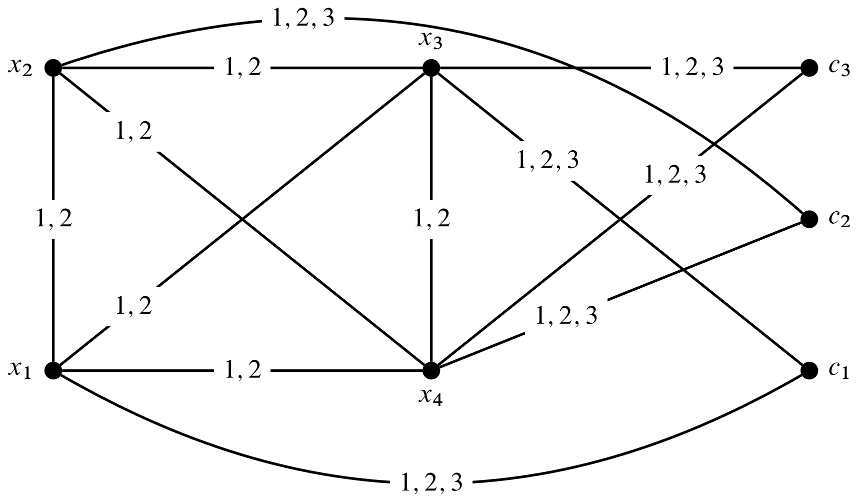


Figure 2.5: The reduction from disjoint sets to diameter computation. In this case, $c_1 = \{x_1, x_3\}$, $c_2 = \{x_2, x_4\}$, and $c_3 = \{x_3, x_4\}$. All temporal edges have travel time equal to 1, so for simplicity we represent on top of each edge only the starting time of each temporal edge with same endpoints. For any distance, the diameter is 3, and, indeed, c_1 and c_2 are disjoint.

Proof. We show that the k-TDS problem is reducible (in quasi-linear time) to the link stream diameter computation problem, even in the very constrained described situations. This reduction has been presented in Calamai, Crescenzi, and Marino (2021) and is a temporal adaptation of the reduction shown in the extended version of Borassi, Crescenzi, and Habib (2016).

Given an input $(X = \{x_1, \ldots, x_{|X|}\}, \mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\})$ of k-TDS with $|X| < \log^k(|\mathcal{C}|)$, we construct a delay-1 undirected link stream $(X \cup \mathcal{C}, \mathbb{E})$, where the set $\mathbb{E}$ of temporal edges is defined as follows (see Figure 2.5).

- For each $x_i, x_j \in X$, $\mathbb{E}$ contains the two temporal edges $(x_i, x_j, 1, 1)$ and $(x_i, x_j, 2, 1)$.
- For each c_j and for each $x_i \in c_j$, $\mathbb{E}$ contains the three temporal edges $(x_i, c_j, 1, 1)$, $(x_i, c_j, 2, 1)$, and $(x_i, c_j, 3, 1)$.

For any $\textsc{d} \in \{\textsc{eat}, \textsc{ft}, \textsc{st}\}$, let us now compute the eccentricities of all vertices, by distinguishing between vertices in X and vertices in $\mathcal{C}$.

vertices in X For each $x_i, x_j \in X$, $d_{\textsc{d}}(x_i, x_j) = 1$, since we can use the temporal edge $(x_i, x_j, 1, 1)$. For each $x_i \in X$ and $c_j \in \mathcal{C}$, $d_{\textsc{d}}(x_i, c_j) = 1$ if $x_i \in c_j$, since we can use the temporal edge $(x_i, c_j, 1, 1)$. Otherwise, by letting x_k be any element in c_j, we get that $(x_i, x_k, 1, 1), (x_k, c_j, 2, 1)$ is a temporal x_i, c_j-path; hence $d_{\text{D}}(x_i, c_j) = 2$. Therefore for each $x_i \in X$, we have that $\text{ECCF}_{\text{D}}(x_i) \leqslant 2$.

vertices in $\mathcal{C}$ For each $c_j \in \mathcal{C}$ and $x_i \in X$, $d_{\text{D}}(c_j, x_i) = 1$ if $x_i \in c_j$, since we can use the temporal edge $(c_j, x_i, 1, 1)$. Otherwise, again by picking any $x_k \in c_j$, we get the temporal c_j, x_i-path $(c_j, x_k, 1, 1), (x_k, x_i, 2, 1)$. Therefore we have that $d_{\text{D}}(c_j, x_i) \leqslant 2$, for every $x_i \in X$.

Now consider $c_h \in \mathcal{C}$ with $h \neq j$. If $c_j \cap c_h \neq \emptyset$, then we have the temporal c_j, c_h-path $(c_j, x_i, 1, 1), (x_i, c_h, 2, 1)$, where $x_i \in c_j \cap c_h$; hence $d_{\text{D}}(c_j, c_h) = 2$. Finally consider $c_j \cap c_h = \emptyset$. We argue that $d_{\text{D}}(c_j, c_h) = 3$. To see that $d_{\text{D}}(c_j, c_h) \leqslant 3$, just consider the temporal c_j, c_h-path $(c_j, x_i, 1, 1), (x_i, x_\ell, 2, 1), (x_\ell, c_h, 3, 1)$, where $x_i \in c_j$ and $x_\ell \in c_h$. Now to see that $d_{\text{D}}(c_j, c_h) \geqslant 3$, note that there is no way of arriving in c_h starting from c_j before time 3, since no neighbor of c_j is also a neighbor of c_h, and, hence, we are forced to pass through two vertices in X. Therefore, for each $c_j \in \mathcal{C}$, we have that $\text{ECCF}_{\text{D}}(c_j) = 3$ if and only if there exists $c_h \in \mathcal{C}$ such that $c_j \cap c_h = \emptyset$.

We can the conclude that the diameter $\mathcal{D}_{\text{D}}$ of the link stream is either at most 2 or equal to 3: it is equal to 3 if and only if there exist two $c_j, c_h \in \mathcal{C}$ which are disjoint.

Since $|X| < \log^k(|\mathcal{C}|)$, the reduction can be executed in $\widetilde{O}(|\mathcal{C}|)$ time, and $|\mathbb{E}| = \widetilde{O}(|\mathcal{C}|)$. Hence, if we can compute the diameter of the link stream in $\widetilde{O}(|\mathbb{E}|^{2-\epsilon})$ for some $\epsilon > 0$, then we could solve the k-TDS in $\widetilde{O}(|\mathcal{C}|^{2-\epsilon})$ for some $\epsilon > 0$. From the result of Borassi, Crescenzi, and Habib (2016), it follows that the SETH would fail, and the theorem is proved. □

Note that the proof of the above theorem gives also strong evidence that a sub-quadratic $(3/2 - \epsilon)$-approximation algorithm for the diameter may be very hard to find, even for undirected delay-1 link streams.

2.5 Exercises

1. Prove Observation 2.

2. $\texttt{reach}^{[t_\alpha, t_\omega]}(u)$ for every time t_α and t_ω, and for every u can be computed using Algorithm 1 assuming that each edge has traversal time at least one. For this reason, this algorithm works for the strict variation defined in Section 1.2 but not for the non-strict one. Write an algorithm to compute $\texttt{reach}^{[t_\alpha, t_\omega]}(u)$ which works for the non-strict case.

3. For each $\text{D} \in \{\text{LDT}, \text{ST}\}$, give an example of a link stream where every temporal u, v-walk P realizing $d_{\text{D}}(u, v)$ is such that the x, y-path contained in P does not realize $d_{\text{D}}(x, y)$, for every x, y in P such that $\{x, y\} \neq \{u, v\}$.

4. Give an example of a link stream containing a temporal u, v-walk P realizing $d_{\text{EAT}}(u, v)$ is such that the x, y-path contained in P does not realize $d_{\text{EAT}}(x, y)$, for every x, y in P such that $\{x, y\} \neq \{u, v\}$
5. Prove the correctness of the single source best paths algorithms presented in Section 2.3.4.
6. Give a quadratic algorithm to solve k-TDS.

3 Connectivity

In this chapter, we present existing results on problems related to connectivity concepts, which intuitively are concepts related to the robustness of the temporal graph. More specifically, given a temporal graph $\mathcal{G}$ and a pair of vertices s, t of $\mathcal{G}$, connectivity problems are concerned with the existence of multiple ways of going from s to t. The more options there are to do this trip in independent (or disjoint) ways, the more robust is the graph. Alternatively, one can also think of robustness as a measure of how easy it would be to disconnect these two vertices by the removal of structures in the graph. In static graphs, these concepts lead to the famous Menger's Theorem (see Theorem 1). In temporal graphs, instead, there are multiple ways of interpreting what "independence" between possible routes mean, not all of them leading to a valid version of Menger's Theorem. The possible interpretations made so far are presented in the following sections.

3.1 Basic definitions and results

For the basic definitions of temporal graphs and temporal paths, go to Section 1.2. In what follows, we introduce the definitions and problems related to temporal graph connectivity. We also lay out some basic results that will be useful in the rest of this chapter.

Given two temporal v_0, v_q-walks, $P = (u = v_0, t_1, v_1, \dots, v_{q-1}, t_q, v_q = v)$ and $P = (u = v_0, t'_1, v'_1, \dots, v'_{r-1}, t'_r, v_r = v)$, we say that they are *vertex disjoint* if $V(P) \cap V(P') = \{v_0, v_q\}$, and that they are *temporal vertex disjoint* (t-vertex disjoint for short) if $V^T(P) \cap V^T(P') \subseteq \{(u, t_1), (v, t_q), (u, t'_1), (v, t'_r)\}$. In other words, they are allowed to intersect in their extremities, but not allowed to have common internal (temporal) vertices. If $uv \notin E(G)$, then a set $S \subseteq V(\mathcal{G}) \setminus \{u, v\}$ is a *temporal vertex u, v-separator* if there is no temporal u, v-walk in $\mathcal{G} - S$, and a set $S \subseteq (V(\mathcal{G}) \setminus \{u, v\}) \times [\tau]$ is a *temporal t-vertex u, v-separator* if $V^T(P) \cap S \neq \emptyset$ for every temporal u, v-walk P in $\mathcal{G}$. Observe that the latter can be informally described as a subset S such that there is no temporal u, v-walk in $\mathcal{G} - S$. However this is not well-defined as the removal of temporal vertices does not lead to a temporal graph in the used notation.

We reproduce again the example seen in Chapter 1 (see Figure 3.1). The temporal b, d-walks $P_1 = (b, 1, a, 1, f, 3, e, 3, d)$ and $P_2 = (b, 1, e, 2, d)$ are not vertex disjoint, since they intersect in e, but are t-vertex disjoint, since their intersection in e occurs in different timesteps (namely, P_1 contains $(e, 3)$ while P_2 contains $(e, 1)$ and $(e, 2)$). We first show that, when dealing with vertex disjointness, if we replace “walks” by “paths” in the above definitions, then there would be no difference in the related optimization parameters.

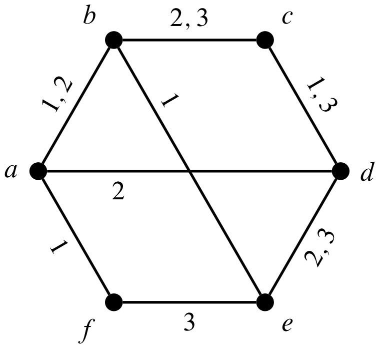


Figure 3.1: Example of a temporal graph, with the λ function being represented on top of each edge. This example has also been shown in Chapter 1 (Figure 1.2).

Proposition 9. *Let $\mathcal{G}$ be a temporal (directed) graph and $u, v \in V(\mathcal{G})$ be such that $uv \notin E(\mathcal{G})$. Then, the maximum number of vertex disjoint temporal u, v-paths is equal to the maximum number of vertex disjoint temporal u, v-walks. Additionally, the minimum size of $S \subseteq V(\mathcal{G}) \setminus \{u, v\}$ such that there are no temporal u, v-paths in $\mathcal{G} - S$ is equal to the minimum size of a temporal vertex u, v-separator. The same hold if strict temporal paths are considered.*

Proof. Denote the maximum number of vertex disjoint temporal u, v-paths by p and the maximum number of vertex disjoint temporal u, v-walks by w. We want to prove that $p = w$. Since a temporal path is also a temporal walk, we get $p \leqslant w$. To prove that $p \geqslant w$, the idea is to pick a set of vertex disjoint temporal u, v-walks, $P_1, \ldots, P_k$, and obtain a temporal u, v-path from P_i, for every $i \in [k]$. For each P_i, write P_i as $(u = v^i_1, \ldots, v^i_{q_i} = v)$. If $v^i_j \neq v^i_\ell$ for every $j, \ell \in [k]$ with $j \neq \ell$, then just define P'_i to be equal to P_i.

Otherwise, let j be minimum such that $v^i_j = v^i_\ell$ for some $\ell \in [k]$, $\ell \neq j$. Suppose ℓ is maximum, i.e., that v^i_ℓ is the last occurrence of v^i_j in P_i. If $v^i_j = v$, then consider $P'_i = (v^i_1, \ldots, v^i_j)$. Otherwise, define $P'_i = (v^i_1, \ldots, v^i_j, t_\ell, v^i_{\ell+1}, \ldots, v^i_{q_i})$, where $(v^i_j v^i_{\ell+1}, t_\ell)$ is the temporal edge used in P to get from $v^i_j = v^i_\ell$ to $v^i_{\ell+1}$. This decreases the number of repeated vertices in P'_i, and we can exhaustively apply this argument until all the obtained walks, $P'_1, \ldots, P'_k$, are paths. Also, because $V(P'_i) \subseteq V(P_i)$ for every $i \in [k]$, it follows that $P'_1, \ldots, P'_k$ are also vertex-disjoint. Therefore $p \geqslant w$ and we are done. The reader should notice that the same arguments hold for temporal directed graphs, and in case only strict temporal paths are considered.

We leave the second part of the proposition as exercise (Exercise 4). □

Now, when we deal with t-vertex disjointness, there is a difference between picking walks or paths. To see this, consider the graph in Figure 3.2. Because in a path we are not allowed to go out of x and come back to x in a later timestep, and since every temporal s,z-path contains the temporal vertex $(x,2)$, we get that there are no 2 t-vertex disjoint temporal s,z-paths. In contrast, there are 2 t-vertex disjoint temporal s,z-walks, namely $(s,1,x,1,y,3,x,3,z)$ and $(s,2,x,2,z)$. In fact, such example can be generalized to obtain a temporal graph with no 2 t-vertex disjoint s,z-paths, and arbitrarily many t-vertex disjoint s,z-walks (see Exercise 5).

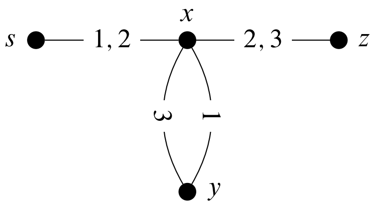


Figure 3.2: Temporal graph having 2 t-vertex disjoint temporal s,z-walks, and no 2 t-vertex disjoint temporal s,z-paths.

We will then treat separately the cases of t-vertex disjoint walks and paths. To do this, we need one last definition. We say that $S \subseteq (V(\mathcal{G}) \setminus \{u,v\}) \times [\tau]$ is a *temporal t-vertex u,v-path-separator* if $V^T(P) \cap S \neq \emptyset$ for every temporal u,v-path P in $\mathcal{G}$. We define also the following parameters, related to each definition and each type of disjointness.

- $p_\mathcal{G}(u,v)$: maximum number of vertex disjoint temporal u,v-paths in $\mathcal{G}$;
- $c_\mathcal{G}(u,v)$: minimum size of a temporal vertex u,v-separator in $\mathcal{G}$;
- $tw_\mathcal{G}(u,v)$: maximum number of t-vertex disjoint temporal u,v-walks in $\mathcal{G}$;
- $tc_\mathcal{G}(u,v)$: minimum size of a temporal t-vertex u,v-separator in $\mathcal{G}$;
- $tp_\mathcal{G}(u,v)$: maximum number of t-vertex disjoint temporal u,v-paths in $\mathcal{G}$;

- $tpc_{\mathcal{G}}(u, v)$: minimum size of a temporal t-vertex u, v-path-separator in $\mathcal{G}$.

If $\mathcal{G}$ is clear from the context, we omit it in the subscript. Finally, we formally define all the problems whose computational complexity are going to be studied in the following sections.

Vertex Disjoint Paths
Input. A temporal graph $\mathcal{G}$, a pair of vertices $s, z \in V(\mathcal{G})$, and an integer k.
Question. Are there k vertex disjoint temporal s, z-paths in $\mathcal{G}$?

Vertex Separator
Input. A temporal graph $\mathcal{G}$, a pair of vertices $s, z \in V(\mathcal{G})$, and an integer k.
Question. Is there a temporal vertex s, z-separator in $\mathcal{G}$ of size at most k?

The following are simple adaptations of the above definitions to the other contexts.

t-vertex Disjoint Walks
Input. A temporal graph $\mathcal{G}$, a pair of vertices $s, z \in V(\mathcal{G})$, and an integer k.
Question. Are there k t-vertex disjoint temporal s, z-walks in $\mathcal{G}$?

t-vertex Separator
Input. A temporal graph $\mathcal{G}$, a pair of non-adjacent vertices $s, z \in V(\mathcal{G})$, and an integer k.
Question. Is there a temporal t-vertex s, z-separator in $\mathcal{G}$ of size at most k?

t-vertex Disjoint Paths
Input. A temporal graph $\mathcal{G}$, a pair of vertices $s, z \in V(\mathcal{G})$, and an integer k.
Question. Are there k t-vertex disjoint temporal s, z-paths in $\mathcal{G}$?

t-vertex Path Separator
Input. A temporal graph $\mathcal{G}$, a pair of non-adjacent vertices $s, z \in V(\mathcal{G})$, and an integer k.
Question. Is there a temporal t-vertex s, z-path-separator in $\mathcal{G}$ of size at most k?

Note that each of the above problems can be defined also on temporal directed graphs; in such cases, we add the prefix Dir to each of them. Additionally, each of them can also be defined in terms of strict temporal paths; in such cases, we add the prefix Strict. Therefore, concerning just the first problem, we can get also the variations Dir Vertex Disjoint Paths, Strict Vertex Disjoint Paths, and Strict Dir Vertex Disjoint Paths.

In what follows, we always present the main results in terms of the problems defined above, but also comment on the complexities of these variations.

3.2 Vertex disjointness

3.2.1 Menger's Theorem and vertex disjointness

Recall that Menger's Theorem on static graphs tells us that the maximum number of vertex disjoint s, z-paths is equal to the minimum size of an s, z-separator. Translating this to the temporal context, and using the previously defined notation, would be equal to say that $p(s, z) = c(s, z)$. This is shown not to hold by Berman (1996) still in 96, and here we present a more famous example, presented by Kempe, Kleinberg, and Kumar (2000). See Figure 3.3. Observe that there are no 2 vertex disjoint temporal s, z-paths. In contrast, one can check that no single vertex in $\{u, v, w\}$ can be a temporal vertex s, z-separator. It thus follows that $p(s, z) = 1 < c(s, z) = 2$. Note that this example works even if only strict paths are considered. Also, we can orient the its edges to obtain a temporal directed graph instead. This means that $p(s, z) = 1 < c(s, z) = 2$ for all possible variations. Kempe, Kleinberg, and Kumar (ibid.) generalize this construction in order to obtain an example where $p(s, z) = 1 < c(s, z) = k$, for arbitrary k.

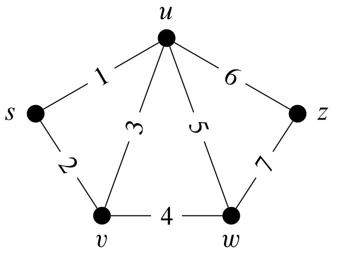


Figure 3.3: Temporal graph $\mathcal{G}$ having no 2 t-vertex disjoint temporal s, z-paths, and whose minimum size of a temporal vertex s, z-separator is 2.

The interesting thing about the above example is that, if each edge is active exactly once, then such graph is exactly the minimal structure that must occur in order to have inequality $p(s, z) < c(s, z)$. We formalize such idea next.

Given a graph G, an *edge subdivision* consists in replacing an edge of G by a path. For example, in Figure 3.4, graph H is obtained from G by subdividing edge uv. If H is obtained from G by a sequence of edge subdivisions, then we say that H is a *subdivision of* G. It is also said that G is a *topological minor* of H.

Kempe, Kleinberg, and Kumar (ibid.) defined a *Mengerian graph* as being a graph G such that, for every $\lambda : E(G) \to \binom{\mathbb{N}}{1}$ (i.e., edges receive subsets of size 1), and every non-adjacent $s, z \in V(G)$, we get that $p_{\mathcal{G}}(s, z) = c_{\mathcal{G}}(s, z)$, where $\mathcal{G} = (G, \lambda)$. In words,

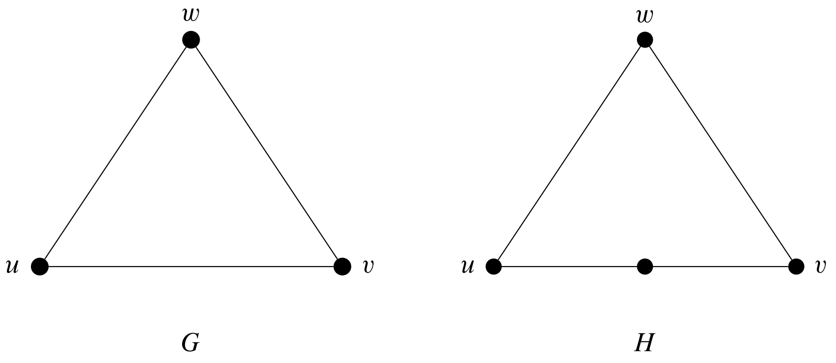


Figure 3.4: Edge subdivision operation.

a graph is Mengerian if, constrained to timefunctions allowing for only one appearance of each edge, we get that G can never be used to get an example where the vertex disjoint version of Menger does not hold. They then prove the following (the *gem* is the base graph of the temporal graph presented in Figure 3.3).

Theorem 10 (Kempe, Kleinberg, and Kumar (ibid.)). *A graph G is Mengerian if and only if G does not have the gem as topological minor.*

Kempe et al.'s result has been recently generalized by Ibiapina and Silva (2022) to allow for multiple appearances of an edge. Because these proofs are long and consist mainly of structural analysis of static graphs, we refrain from presenting them here. Nevertheless, we present in Figure 3.5 a temporal graph $\mathcal{G}$ not having the gem as a topological minor, but such that $p(s, z) < c(s, z)$. This shows how indeed the theorem above only holds with the additional constraint that each edge is active exactly once.

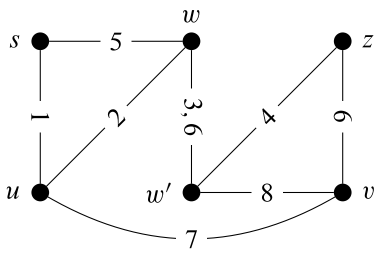


Figure 3.5: Example of temporal graph $\mathcal{G} = (G, \lambda)$ where $p(s, z) < c(s, z)$, while G does not have the gem as topological minor.

3.2.2 Complexity of vertex disjoint problems

In this section, we present the many known results concerning the (parameterized) complexity of problems Vertex Disjoint Paths and Vertex Separator, and their variations. We start by the simple reduction presented by Berman (1996). Such reduction helps us understand why it is often the case that the directed version of paths-related problems are harder than their undirected versions. Indeed, it happens because a reduction from Linkage (defined below) is made, and it is known that such problem is NP-complete on directed graphs even if $k = 2$ (Fortune, Hopcroft, and Wyllie (1980)), while it is NP-complete on undirected graphs (Karp (1975)), but polynomial-time solvable if k is a fixed value (Robertson and Seymour (1990)).

Linkage
Input. A (directed) graph G, and k pairs of vertices $\{(s_1, z_1), \dots, (s_k, z_k)\}$, $k \geqslant 2$.
Question. Are there vertex disjoint paths $P_1, \dots, P_k$, where P_i is an s_i, z_i-path for every $i \in [k]$?

Theorem 11 (Berman (1996)). *Let $\mathcal{G}$ be a temporal (directed) graph with lifetime τ, s, z be a pair of vertices of $\mathcal{G}$, and k be a positive integer. Solving Vertex Disjoint Paths is NP-complete on $(\mathcal{G}, s, z, k)$. And if $\mathcal{G}$ is directed, then Dir Vertex Disjoint Paths is NP-complete even if $k = \tau = 2$.*

Proof. Because a set of s, z-paths can be checked to be vertex disjoint temporal s, z-paths in polynomial time (Exercise 1), we get that Vertex Disjoint Paths is in NP. To prove hardness, we make a reduction from Linkage. Let $(G, \mathcal{P})$ be an instance of Linkage, where $\mathcal{P} = \{(s_1, z_1), \dots, (s_k, z_k)\}$. We construct a temporal graph $\mathcal{G}$ from G as follows. First, add two new vertices, s and z, and make s adjacent to s_i and z adjacent to z_i, for every $i \in [k]$. Then, as timefunction, assign to each edge $e \in E(G)$ all values within $[k]$ (i.e., $\lambda(e) = [k]$). Additionally, for each $i \in [k]$, let $\lambda(ss_i) = \lambda(z_i z) = \{i\}$. We show that $(G, \mathcal{P})$ is equivalent to $(\mathcal{G}, s, z, k)$, i.e., that there are vertex disjoint paths $P_1, \dots, P_k$, where P_i is an s_i, z_i-path for every $i \in [k]$, if and only if there are k vertex disjoint temporal s, z-paths in $\mathcal{G}$.

First, suppose that $P_1, \dots, P_k$ are vertex disjoint paths solving Linkage. Write P_i as $(s_i = x^i_1, x^i_2, \dots, z_i = x^i_{\ell_i})$, and define the temporal s, z-path

$$P'_i = (s, i, s_i, i, x^i_2, \dots, i, z_i, i, z);$$

in words, P'_i is equal to P_i in timestep i extended by (ss_i, i) and $(z_i z, i)$. Because $P_1, \dots, P_k$ are vertex disjoint, it follows directly that $P'_1, \dots, P'_k$ are (internally) vertex disjoint. Now, let $P'_1, \dots, P'_k$ be (internally) vertex disjoint temporal s, z-paths. Because s has exactly k temporal edges incident to it, we get that each such edge belongs to one of these paths. Suppose, without loss of generality, that P'_i starts with the temporal edge (ss_i, i) for each $i \in [k]$. By a similar argument, we get that each $(z_i z, i)$ belongs to exactly one of these paths. Observe that $(z_1z, 1)$ can only belong to P_1', and hence P_1' contains a s_1, z_1-path of G, P_1. Then, because P_1' has already arrived to z before timestep 2, we get that $(z_2z, 2)$ can only belong to P_2', and hence P_2' contains a s_2, z_2-path of G, P_2. By iteratively applying this argument, we get the k desired paths.

Observe that the construction also works if G is a directed graph. Indeed, in such case it is enough to add edges from s to s_i and from z_i to z, for every $i \in [k]$. □

The theorem above raises the natural question about whether the problem is NP-complete also on temporal undirected graphs when $k = 2$. This was answered positively by Kempe et al. in their seminal paper. In fact, the proof is an interesting application of the undelayed construction (see definition in Section 1.5). Before we present the simple proof, we show the following important property. Recall that a more general model for temporal graphs is a triple (G, α, ϕ), where G is a multigraph, α gives the starting time of each edge, and ϕ gives the travel time.

Lemma 12. *Let $\mathcal{G} = (G, \alpha, \phi)$ be a general temporal graph, with travel time on the edges. Also, let $\mathcal{G}' = (G', \lambda)$ be the undelayed version of $\mathcal{G}$. Given $s, z \in V(G)$, we have that there are k vertex disjoint temporal s, z-paths in $\mathcal{G}$ if and only if there are k vertex disjoint temporal s, z-paths in $\mathcal{G}'$.*

Proof. Given $e \in E(G)$, denote by w_e the vertex of $\mathcal{G}'$ related to e. First, consider $P_1, \dots, P_k$ to be vertex disjoint s, z-paths in $\mathcal{G}$. Then, for each $i \in [k]$, let P_i' be obtained from P_i by replacing each edge $e = uv \in E(P_i)$ with the subpath $(u, \alpha(e), w_e, \alpha(e) + \phi(e), v)$. Because $P_1, \dots, P_k$ are vertex disjoint, and the only new vertices in $P_1', \dots, P_k'$ are of the type w_e, one can see that also the latter paths are vertex disjoint. Now, if $P_1', \dots, P_k'$ are vertex disjoint temporal s, z-paths in $\mathcal{G}'$, because G' is a bipartite graph with parts $V(G)$ and $W = \{w_e \mid e \in E(G)\}$, and each $w_e \in W$ has degree exactly two, one can see that we can obtain temporal s, z-paths $P_1, \dots, P_k$ in $\mathcal{G}$ by replacing subpaths (u, t, w_e, t', v) with the corresponding edge of G. Additionally, as $V(P_i) \subseteq V(P_i')$ for every $i \in [k]$, and $V(P_i') \cap V(P_j') = \emptyset$ for every $i, j \in [k], i \neq j$, we get that the same holds for $P_1, \dots, P_k$, i.e., these are vertex disjoint in $\mathcal{G}$. □

Now, we apply the above lemma to make a simple reduction from the following problem to our problem of interest.

Bounded Length Disjoint Paths
Input. A (directed) graph G, a pair of vertices s, z, and two positive integers ℓ and k.
Question. Are there internally vertex disjoint s, z-paths $P_1, \dots, P_k$ in G such that each P_i has length at most ℓ?

The problem above has been thoroughly investigated, both on directed and undirected graphs, as well as from the point of view of parameterized complexity, and even its edge version. The complete picture has been summarized by Golovach and Thilikos (2011), where the authors also present new results. Particularly, it is known that the problem is

NP-complete when G is undirected, even if $k = 2$ (Li, McCormick, and Simchi-Levi (1990)). This is what is used in the theorem below.

Theorem 13 (Kempe, Kleinberg, and Kumar (2000)). *Let $\mathcal{G}$ be a temporal graph with lifetime τ, s, z be a pair of vertices of $\mathcal{G}$, and $k \geqslant 2$ be a fixed positive integer. Solving* Vertex Disjoint Paths *is* NP*-complete on $(\mathcal{G}, s, z, k)$.*

Proof. The problem is in NP, as argued already in the proof of Theorem 11. Consider now an instance $(H, s, z, \ell, 2)$ of Bounded Length Disjoint Paths, where H is undirected. Observe Figure 3.6 to follow the construction. Let G be the multigraph obtained from H by adding $\ell - 1$ copies of each edge (hence, each edge of G has multiplicity ℓ). Finally, let $\mathcal{G} = (G, \alpha, \phi)$, be constructed in a way that $\phi(e) = 1$ for every $e \in E(G)$, and such that each copy of $e \in E(H)$ starts in a different time. Formally, for every $e' \in E(H)$, we have that $\{\alpha(e) \mid e \text{ has same endpoints as } e'\} = \{1, \ldots, \ell\}$. It is not hard to see that each path of bounded length in H is related to a temporal s, z-path in $\mathcal{G}$ and vice-versa. Finally, applying Theorem 12 the theorem follows.

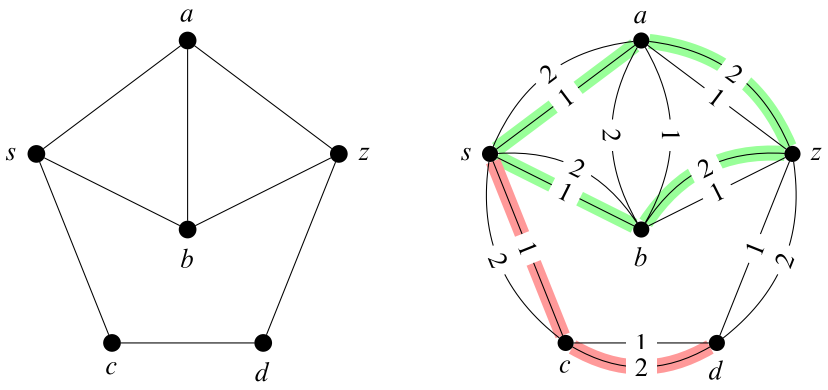


Figure 3.6: Construction in the proof of Theorem 13, with $\ell = 2$. As all travel times are equal to 1, we refrain from representing them in the figure. Observe that the two paths of length at most 2 in H (to the left) are related to the green paths in $\mathcal{G}$ (to the right). The non-valid path of length 3 in H is also not a temporal path in $\mathcal{G}$ as it arrives too late in d in order to use any of the edges from d to z (red path).

To obtain the same result for higher values of k, one can simply add artificial s, z-paths of length 2. ☐

Observe that the proof of Theorem 13 actually works also for the strict case, since a temporal path in (G, α, ϕ) is exactly a strict temporal path in (G, α), and vice-versa. Additionally, the lifetime of the constructed temporal (directed) graph is either $\ell + 1$ (non-strict case) or ℓ (strict case). Because Bounded Length Disjoint Paths is NP-complete on directed and undirected graphs when $k = 2$, and on directed or undirected graphs when

	k	τ	$k+\tau$
Vertex Disjoint Paths	pNP $k \geqslant 2$ - Theorem 13	pNP $\tau \geqslant 5$	Open
Dir VDP	pNP $k \geqslant 2, \tau \geqslant 2$ - Theorem 11		
Strict VDP	pNP $k \geqslant 2$ - Theorem 13	pNP $\tau \geqslant 5$	Open
Strict Dir VDP	pNP $k \geqslant 2$ - Theorem 13	pNP $\tau \geqslant 5$	Open

Table 3.1: VDP stands for Vertex Disjoint Paths; pNP stands for para-NP-complete. Parameterized complexity of paths problems when parameterized by: the number of paths k; the lifetime τ; and the sum $k + \tau$. Results by Itai, Perl, and Shiloach (1982) directly imply the entries of the table for column τ by using the same reduction as the one in Theorem 13.

$\ell \geqslant 5$ (Itai, Perl, and Shiloach (1982)), we get that Vertex Disjoint Paths, Dir Vertex Disjoint Paths, Strict Vertex Disjoint Paths and Strict Dir Vertex Disjoint Paths according to parameters k and τ, where τ denotes the lifetime. As for the parameter $k + \tau$, Golovach and Thilikos (2011) give FPT algorithms for all variations of Bounded Length Disjoint Paths. Because we do not have a reduction on the opposite direction, i.e., from our problems to Bounded Length Disjoint Paths, it remains open whether our problems can also be solved in FPT time when parameterized by $k + \tau$, except for the case Dir Vertex Disjoint Paths which is NP-complete even for $k = \tau = 2$ (Berman (1996)). Even though the observations about the complexity parameterized by τ are quite straightforward, they were not mentioned by Kempe, Kleinberg, and Kumar (2000). This is why we cite Itai, Perl, and Shiloach (1982) in Table 3.3.

Concerning Vertex Separator, Kempe, Kleinberg, and Kumar (2000) also give an NP-complete proof for it. Observe however, that this problem cannot be NP-hard for fixed values of k (see Exercise 7). In other words, while the paths problem is para-NP-complete when parameterized by k, the separator problem is in XP, and hence polynomial-time solvable for fixed values of k. A natural question therefore is whether the XP complexity could be improved to FPT. The answer is no, as proved by Zschoche et al. (2020). In fact, such reduction relies again on the complexity of length bounded paths and a lemma similar to Theorem 12. Zschoche et al. (ibid.) treat only the simpler case where $\phi(e) = 1$ in order to reduce the problem on strict increasing paths to the problem on non-strict paths. We leave the proof of the following lemma as exercise. Again, the notion of a separator in $\mathcal{G} = (G, \alpha, \phi)$ is analogous to ours. Formally, given non-adjacent $s, z \in V(G)$, a temporal vertex s, z-separator is a subset $X \subseteq V(G) \setminus \{s, z\}$ such that X intersects every temporal s, z-path in $\mathcal{G}$.

Lemma 14. *Let $\mathcal{G} = (G, \alpha, \phi)$ be a general temporal graph, with travel time on the edges. Also, let $\mathcal{G}' = (G', \lambda)$ be the undelayed version of $\mathcal{G}$. Given $s, z \in V(G)$, we have that there is a temporal vertex s, z-separator in $\mathcal{G}$ of size at most k if and only if there is a temporal vertex s, z-separator in $\mathcal{G}'$ of size at most k.*

Now, the reduction is made from the following problem, known to be W[1]-hard when parameterized by k (Golovach and Thilikos (2011)).

Separator for Bounded Length Paths
Input. A (directed) graph G, a pair of non-adjacent vertices s, z, and two positive integers ℓ and k.
Question. Is there a set $X \subseteq V(G) \setminus \{s, z\}$ of size at most k that intersects every s, z-path of length at most ℓ?

Theorem 15 (Zschoche et al. (2020)). *Let $\mathcal{G}$ be a temporal (directed) graph, s, z be a pair of non-adjacent vertices of $\mathcal{G}$, and k be a positive integer. Solving Vertex Separator is $W[1]$-hard on $(\mathcal{G}, s, z, k)$, when parameterized by k.*

Proof. We make a reduction from Separator for Bounded Length Paths. Consider then an instance of such problem, (H, s, z, ℓ, k). We show that the same construction as the one in the proof of Theorem 13 works here; so let $\mathcal{G} = (G, \alpha, \phi)$ be obtained as before. We prove that (H, s, z, ℓ, k) is a "yes" instance for Separator for Bounded Length Paths if and only if $\mathcal{G}$ has a temporal vertex s, z-separator of size at most k. The theorem then follows by applying Theorem 14.

So suppose that $X \subseteq V(H) \setminus \{s, z\}$ is a set of size at most k that intersects every s, z-path of length at most ℓ. Because a temporal s, z-path in $\mathcal{G}$ gives also an s, z-path in H of length at most ℓ, we get that X must be also a temporal vertex s, z-separator in $\mathcal{G}$. Now, if $X \subseteq V(G) \setminus \{s, z\}$ is a temporal vertex s, z-separator in $\mathcal{G}$, then note that we can suppose that $X \cap W = \emptyset$. Indeed, let $w_e \in X \cap W$, where $e = uv \in E(H)$. Since s, t are not adjacent, we get that either u or v is not in $\{s, t\}$, suppose u. Then, since w_e has degree 2 in $\mathcal{G}$, we get that all temporal s, z-path passing by w_e must also pass by u. Therefore, $(X \setminus \{w_e\}) \cup \{u\}$ is also a temporal vertex s, z-separator in $\mathcal{G}$. Now, because every s, z-path of length at most ℓ in H defines also a temporal s, z-path in $\mathcal{G}$, we get that X must intersect every such path, i.e., (H, s, z, ℓ, k) is a "yes" instance for Separator for Bounded Length Paths. □

As before, one can see that the proof also works for the strict variation of the problem. And, again, because Separator for Bounded Length Paths is W[1]-hard when parameterized by k even on directed graphs, it follows that Vertex Separator, Dir Vertex Separator, Strict Vertex Separator and Strict Dir Vertex Separator are all W[1]-hard when parameterized by k. To close this section, we mention that another parameter of interest is the lifetime of the temporal graph. Table 3.4 summarizes all known results. The polynomiality results in the table for $\tau \leqslant 4$ are related to the fact that a version of Menger for bounded length paths in static graphs holds when the length is bounded by 4 (Lovász, Neumann-Lara, and Plummer (1978)). A good question is whether the polinomiality results when $\tau \leqslant 4$ extend for the paths problems.

Before we proceed, we comment about the classical complexity of Vertex Separator. When proving W[1]-hardness, the authors usually do not bother to prove completeness (i.e., that the problem is also contained in the class of problems W[1]). To do this, one needs

	k	τ	$k + \tau$
Vertex Separator	W[1]	pNP - $\tau \geqslant 2$	XP (T)
Dir VSP	W[1]	pNP - $\tau \geqslant 2$	XP (T)
Strict VSP	W[1]	pNP - $\tau \geqslant 5$ Poly - $\tau \leqslant 4$	XP (T)
Strict Dir VSP	W[1]	pNP - $\tau \geqslant 5$ Poly - $\tau \leqslant 4$	XP (T)

Table 3.2: All results presented in this table are presented by Zschoche et al. (2020), except the ones marked with T, which stands for "trivial". VSP stands for Vertex Separator; pNP stands for para-NP-complete; W[1] stands for W[1]-hard. Parameterized complexity of separator problems when parameterized by: the size of the separator k; the lifetime τ; and the sum $k + \tau$.

to provide what is called a *circuit with weft at most* 1 to solve the problem, which entails in giving all the appropriate definitions, and building such circuit. The authors hence usually choose not to do it as such work can be tedious and not very enlightening as to how hard the problem really is (after all, W[1]-hardness is already established). Since the scope of this book is not parameterized complexity theory, we also refrain from dwelling in such deep waters. Nevertheless, a simpler question is whether the problem is NP-complete. Observe that, since Separator for Bounded Length Paths is also NP-hard (Golovach and Thilikos (2011)), we get that Vertex Separator is NP-hard too. Additionally, note that the algorithms seen in Chapter 2 can be used to prove that the following problem is polynomial-time solvable.

Temporal Vertex Separator Testing
Input. A temporal (directed) graph $\mathcal{G}$, a pair of non-adjacent vertices s, z, and a subset $X \subseteq V(G) \setminus \{s, z\}$.
Question. Is X a temporal vertex s, z-separator?

Now, observe that this means that Vertex Separator is also in NP (and hence is NP-complete), as a polynomial certificate would be a temporal vertex s, z-separator of size at most k. As we will see later, the analogous problem for temporal t-vertex s, z-separators is also in P, but the same is not true for temporal t-vertex s, z-path-separators.

3.3 Temporal Vertex disjoint walks

Recall that $tw(s, z)$ denotes the maximum number of t-vertex disjoint temporal s, z-walks in $\mathcal{G}$, while $tc(s, z)$ denotes the minimum size of a temporal t-vertex s, z-separator in $\mathcal{G}$. In this section, we prove that a version of Menger's Theorem concerning t-vertex disjoint walks holds, on both directed and undirected temporal graphs. This is a nice application of the static expansion graph (see Section 1.5).

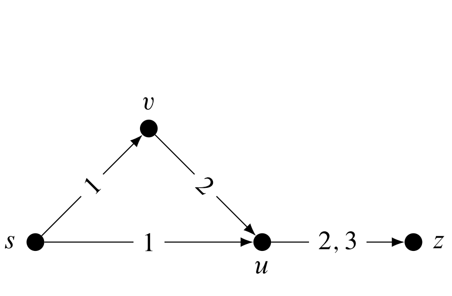
Theorem 16 (Ibiapina, Lopes, et al. (2022)). *Let $\mathcal{G} = (G, \lambda)$ be a temporal (directed) graph with lifetime τ, and $s, z \in V(G)$ be such that $sz \notin E(G)$. Then $tw(s,z) = tc(s,z)$, and such values can be computed in polynomial time.*

Proof. Let D denote the directed graph obtained from the static expansion of $\mathcal{G}$ by identifying all vertices $\{(s,i) \mid i \in [\tau]\}$ into a single vertex, s, and all vertices $\{(z,i) \mid i \in [\tau]\}$ into a single vertex, z. We prove that $tw(s,z)$ is equal to the maximum number of vertex disjoint s,z-paths in D, while $tc(s,z)$ is equal to the minimum size of an s,z-separator in D. The theorem thus follows by Menger's Theorem on static directed graphs, and the fact that computing these parameters in a static directed graph is largely known to be polynomial-time solvable (see e.g. West (2000)).

First, given an s,z-path P in D, we explain how to construct a temporal s,z-walk P' in $\mathcal{G}$. So let $P = (\alpha_0 = s, e_1, \ldots, e_q, \alpha_q = z)$ be an s,z-path in D. We start with P' being equal to P, and replace objects in P' until we obtain a temporal s,z-walk in $\mathcal{G}$. So for each $i \in [q-1]$, write α_i as (u_i, t_i). Also, for simplicity of notation, we consider α_0 to be equal to (s, t_1) and α_q to be equal to (z, t_{q-1}). Observe that $e_1 = s(u_1, t_1)$ and $e_q = (u_{q-1}, t_{q-1})z$, i.e., t_1 and t_{q-1} are the starting and finishing times of P, respectively. Now, for each $i \in [q]$, if $u_{i-1} = u_i$ (and hence $t_i = t_{i-1} + 1$), remove e_i and α_i from P'. And if $u_{i-1} \neq u_i$ (and hence $t_i = t_{i-1}$), then replace e_i in P' by t_i. Note that in the latter case, we get that $(u_{i-1}u_i, t_i)$ is a temporal edge of $\mathcal{G}$; call such fact (*). Observe that we are now left with a sequence that alternates temporal vertices and timesteps. Because of (*), and since $t_1 \leqslant t_2 \leqslant \ldots \leqslant t_q$ as $(u,i)(v,j)$ is not an edge of D whenever $i > j$, it suffices to replace each temporal vertex (u_i, t_i) in the sequence by simply u_i in order to obtain a temporal walk in $\mathcal{G}$. One can verify that $V^T(P') \setminus (\{s,z\} \times [\tau]) = \{\alpha_1, \ldots, \alpha_{q-1}\} = V(P) \setminus \{s,t\}$. Additionally, observe that the backward transformation satisfying the same property can also be defined, i.e., given a temporal s,z-walk P', we can construct an s,z-path P in D such that $V^T(P') \setminus (\{s,z\} \times [\tau]) = V(P) \setminus \{s,z\}$. This directly implies that $tw(s,z)$ is equal to the maximum number of vertex disjoint s,z-paths in D.

So now suppose that $X \subseteq (V(G) \setminus \{s,z\}) \times [\tau]$ is a minimum temporal vertex s,z-separator. Note that $X \subseteq V(D)$ and that, if there is an s,z-path not intersecting X in D, then there is a temporal s,z-walk not intersecting X in $\mathcal{G}$ by the previous paragraph, a contradiction. On the other hand, consider $X \subseteq V(D) \setminus \{s,z\}$ to be an s,z-separator in D. By construction, $X \subseteq (V(G) \setminus \{s,z\}) \times [\tau]$. And, again by the previous paragraph, there cannot be a temporal s,z-walk in $\mathcal{G}$ not intersecting X. □

The above theorem then gives us that (Dir) t-vertex Disjoint Paths and (Dir) t-vertex Separator are all polynomial-time solvable. Now, one might be tempted to apply the same argument to the strict versions of such problems, but using the strict static expansion of $\mathcal{G}$ instead. Such argument cannot work, as witnessed by the example in Figure 3.7. In it, we have 2 vertex disjoint s,z-paths in the obtained static digraph D, when instead, because $(u,2)$ is contained in every temporal s,z-walk, we get that the maximum number of t-vertex disjoint strict temporal s, z-walks in $\mathcal{G}$ is just 1. The problem with this approach is that it makes no distinction between vertices $(u, i-1)$ and (u, i), when such vertices are actually representing the single temporal vertex (u, i). We believe that this mistake was made by Mertzios, Michail, and Spirakis (2019), where they state that a version of Menger's Theorem on t-vertex disjoint strict walks holds. Indeed, as can be witnessed by the temporal graph depicted in Figure 3.8b, an analogous of Theorem 16 for the strict case does not hold, contrary to what is stated by Mertzios, Michail, and Spirakis (ibid.) in Corollary 2 of their work. The fact that such temporal graph is indeed an example where Menger does not hold for strict t-vertex disjoint temporal walks follows from Theorem 18, seen in the next section, and the fact that every strict temporal s, z-walk in such temporal graph is also a strict temporal s, z-path. We then get that the complexity of Strict (Dir) t-vertex Disjoint Paths and Strict (Dir) t-vertex Separator are all open.



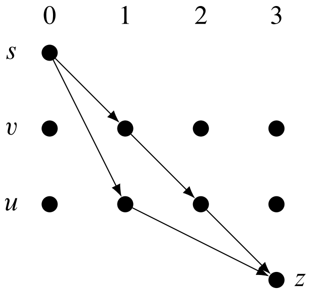


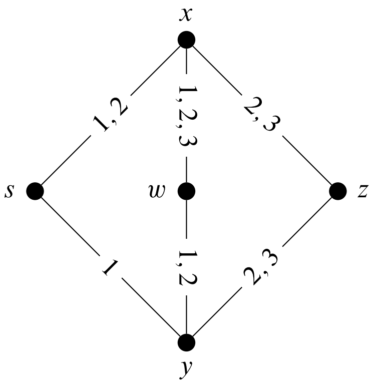
Figure 3.7: Construction analogous to the one in Theorem 16, but using the strict static expansion. For simplicity, we omit the edges linking the copies of u and of v, presenting only the 2 vertex disjoint s, z-paths.

3.4 Temporal Vertex disjoint paths

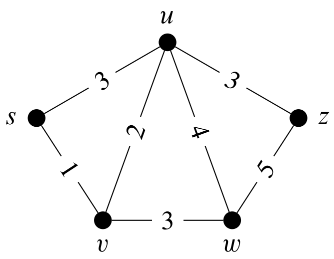
In this section, we study the last type of (temporal) vertex disjointness that has been investigated in the literature so far, namely t-vertex disjointness among temporal paths. This concept has been recently introduced by the authors of this book and co-authors (Ibiapina, Lopes, et al. (2022)). This is why all results known so far are due to them.

As we have seen in the previous section, if we consider t-vertex disjoint walks, then a version of Menger's Theorem holds. This is not the case for paths, as can be seen by the example in Figure 3.8a. Recall that $tp(u, v)$ denotes the maximum number of t-vertex disjoint temporal u, v-paths, while $tpc(u, v)$ denotes the minimum size of a temporal t-vertex u, v-path-separator.

Proposition 17 (Ibiapina, Lopes, et al. (ibid.)). *Let $\mathcal{G}$ be the temporal graph depicted in Figure 3.8a. Then $tp(s, z) = 2$ while $tpc(s, z) = 3$.*



(a) Non-strict case.



(b) Strict case.

Figure 3.8: Examples of temporal graphs where the maximum number of (strict) temporal s, z-paths is smaller that the minimum size of a (strict) temporal t-vertex s, z-path-separator.

Proof. Because $(s, 1, x, 2, z)$ and $(s, 1, y, 2, z)$ are t-vertex disjoint, we get that $tp(s, z) \geqslant 2$. To see that $tp(s, z) \leqslant 2$, suppose by contradiction that P_1, P_2, P_3 are 3 t-vertex disjoint temporal s, z-paths. Observe that each of the 3 temporal edges incident in s must be used, so we can suppose, without loss of generality, that P_1 starts with $(s, 1, x)$, P_2 with $(s, 2, x)$ and P_3 with $(s, 1, y)$. Now, since P_1, P_2 are t-vertex disjoint, we get that P_2 must use the temporal edge $(xw, 1)$ as otherwise P_1 and P_2 would intersect in $(x, 2)$. But then, because P_2 is a path, not a walk, it must continue through y. If it contains temporal edge $(wy, 1)$, then it intersects P_3 in $(y, 1)$; hence it must contain $(wy, 2)$. But note that P_3 also cannot use $(wy, 1)$ nor $(wy, 2)$, which means that P_3 is "stuck" in y until time at least 2. This is a contradiction as in this case P_2 and P_3 intersect in $(y, 2)$. We leave the proof of $tpc(s, z) = 3$ as exercise (Exercise 11). □

One can now recall the example seen in Section 3.2, Figure 3.3, where we had $p(s, z) = 1 < c(s, z) = 2$, and wonder whether an example where $tp(s, z) = 1 < tpc(s, z) = 2$ exists. The answer is no, as a version of Menger's Theorem holds when $tp(s, z) = 1$. This will be discussed in Section 3.4.1. Then, as usual, we present complexity results in Section 3.4.2. But before moving on, we emphasize the fact that, as happened in the t-vertex walks case, the mentioned version of Menger only holds for non-strict paths. Indeed, the same example showing that, in the strict case, we can have $tw(s, z) = 1 < tc(s, z) = 2$, also shows that we can have $tp(s, z) = 1 < tpc(s, z) = 2$. It is presented in Figure 3.8b.

Proposition 18 (Ibiapina, Lopes, et al. (2022)). *Let $\mathcal{G}$ be the temporal graph depicted in Figure 3.8a. Considering strict temporal paths, we get $tp(s, z) = 1$ while $tpc(s, z) = 2$.*

3.4.1 Menger for t-vertex disjoint paths

In this section, we will present the main ideas behind the proof of Theorem 19, presented below. We will not present the entire proof, as it is quite long and can be found in the manuscript by Ibiapina, Lopes, et al. (ibid.), and because we believe that the most interesting exercise will be to contrast the ideas behind the proof of Ibiapina et al. with the proof of Menger's Theorem on static graphs.

Theorem 19 (Ibiapina, Lopes, et al. (ibid.)). *Let $\mathcal{G}$ be a temporal (directed) graph, and let $s, z \in V(\mathcal{G})$ be such that $sz \notin E(\mathcal{G})$. Then, $tp(s, z) = 1$ if and only if $tpc(s, z) = 1$.*

Observe that $tp(s, z) \leqslant tpc(s, z)$ always holds and if there are no temporal s, z-paths, then no temporal vertex would be needed (i.e., the emptyset would be a temporal t-vertex s, z-path-separator). Therefore the sufficient part of the theorem follows. It remains to prove that, in case $tp(s, z) = 1$, there must exist a temporal vertex which is contained in every temporal s, z-path (i.e., $tpc(s, z) = 1$). Before we comment on this proof, we revisit some of the ideas behind the proof of Menger's Theorem. The complete version of the proof studied here can be found in the book byWest (2000).

So let us recall Menger's Theorem, which is stated as Theorem 1 in Section 1.1. There, given a (directed) graph G and vertices $s, z \in V(G)$, we used $p(s, z)$ and $c(s, z)$ to denote the maximum number of vertex disjoint s, z-paths and the minimum size of an s, z-separator in G. There is an abuse of language, as this notation is used also in the vertex disjoint temporal context. This is why, when we use this notation for the temporal context in what follows, we will always accompany with the temporal graph in the underline. Coming back to Menger's Theorem, it then states that $p(s, z) = c(s, z)$. As it happens with temporal paths, the inequality $p(s, z) \leqslant c(s, z)$ holds in a straightforward way, since a separator must contain at least one vertex of each path in a maximum set of vertex disjoint paths. Hence, the hard part of the proof of Menger's Theorem is also proving that $p(s, z) \geqslant c(s, z)$. A first difference with relation to Theorem 19 is that, in the latter, the inequality $tp(s, z) \geqslant tpc(s, z)$ only holds with certainty if $tp(s, z) = 1$.

Now, to prove that $p(s, z) \geqslant c(s, z)$ holds on static graphs, we apply induction on $n = |V(G)|$. For this, consider a minimum s, z-separator, X, and denote $c(s, z)$ by k. The objective is to construct k vertex disjoint s, z-paths in G. For this, consider V_1 to be the set of vertices contained in paths between s and X. Similarly, let V_2 be the set of vertices contained in paths between X and z. Let also G_1 be obtained from $G[V_1]$ by adding a vertex z', and adding edge xz' for every $x \in X$. Similarly, let G_2 be obtained from $G[V_2]$ by adding a vertex s', and adding edge $s'x$ for every $x \in X$ (see Figure 3.9). Applying the induction hypothesis on G_1 and G_2, we obtain k vertex disjoint s, z'-paths in G_1 that can be combined with k vertex disjoint s', z-paths in G_2 in order to obtain the desired s, z-paths in G (see Figure 3.10). The difficulty however is that if either G_1 or G_2 is not smaller than G, then we cannot apply the induction hypothesis. Observe that this happens when either $X = N(s)$ or $X = N(z)$. To overcome this, one first argues that $V(G) = N[s] \cup N[z]$, then apply the notion of vertex cover and matchings, as well as König-Egerváry's Theorem, in order to obtain the desired disjoint paths. As we will see shortly, this last part of the proof is also quite different from what is needed in Theorem 19.

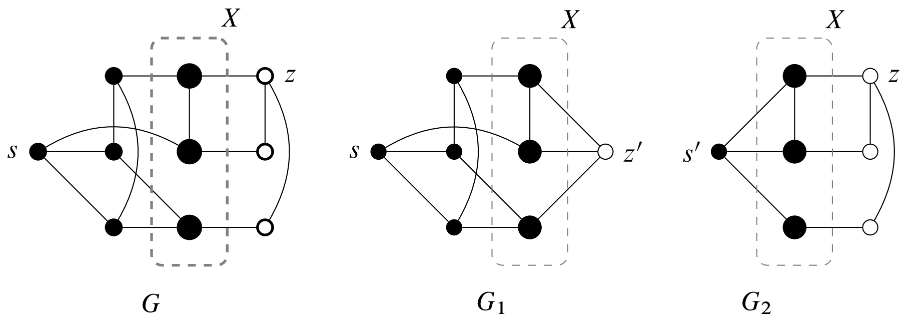


Figure 3.9: Example of the construction in the proof of Menger's Theorem on static graphs. Small black vertices denote $V_1 \setminus X$, big black vertices denote X and white vertices denote $V_2 \setminus X$.

Going back to the proof of Theorem 19, we need to prove that if $tp(s, z) = 1$, then $tpc(s, z) = 1$. This is done by the counterpositive (i.e., by proving that if $tpc(s, z) > 1$, then $tp(s, z) > 1$) in a way that reminds us somehow of the proof of Menger's Theorem. We apply induction on $|V(\mathcal{G})| + |E^T(\mathcal{G})|$. The first part of the proof, which we leave as exercise (see Exercise 14), consists in showing that if $tpc(s, z) > 2$, then we can apply induction to obtain the desired paths. So, suppose that $tpc(s, z) = 2$, and let $X = \{(u, t_u), (v, t_v)\}$ be a temporal t-vertex s, z-path-separator in $\mathcal{G}$. The idea here is also to combine temporal paths from s to X, and from X to z in order to obtain at least two t-vertex disjoint temporal s, z-paths. However, there are many difficulties in this approach. The first one concerns the temporal graphs on which induction hypothesis will be applied. These are defined by first picking the following temporal graphs, which are spanning subgraphs of $\mathcal{G}$ (i.e., by "union of paths" below, we mean that we pick the temporal edges contained in such paths, while all temporal vertices of $\mathcal{G}$ are taken).

- $\mathcal{G}^S_{su} = (G_{su}, \lambda_{su})$: union of all temporal s, u-paths not passing by v;
- $\mathcal{G}^S_{sv} = (G_{sv}, \lambda_{sv})$: union of all temporal s, v-paths not passing by u;
- $\mathcal{G}^S_{uz} = (G_{ut}, \lambda_{ut})$: union of all temporal u, z-paths not passing by v; and
- $\mathcal{G}^S_{vz} = (G_{vt}, \lambda_{vt})$: union of all temporal v, z-paths not passing by u.

Then, a temporal graph $\mathcal{G}_1$ is constructed from the union of $\mathcal{G}^S_{su}$ and $\mathcal{G}^S_{sv}$ by adding a new vertex z' adjacent to each $x \in \{u, v\}$ in timestep τ. Similarly, $\mathcal{G}_2$ is constructed from the union of $\mathcal{G}^S_{uz}$ and $\mathcal{G}^S_{vz}$ by adding a new vertex s' adjacent to each $x \in \{u, v\}$ in timestep 1. The idea then is to apply induction hypothesis in $\mathcal{G}_1$ to pick temporal s, z'-paths P_1, P_2, apply induction hypothesis also in $\mathcal{G}_2$ to obtain temporal s', z-paths P'_1, P'_2, then finally combine P_1 and P'_1, and P_2 and P'_2 to obtain temporal s, z-paths. This brings us

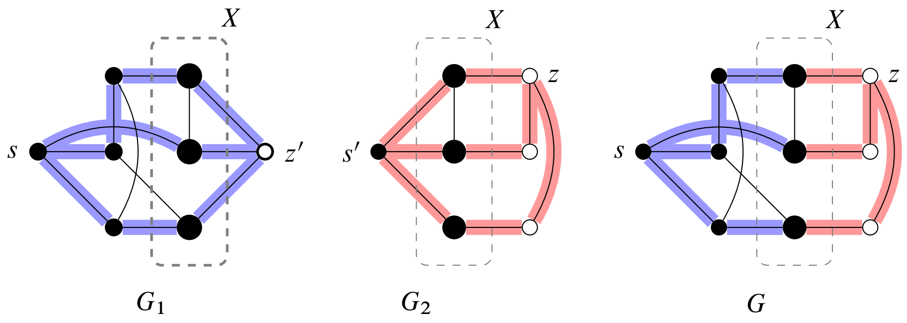


Figure 3.10: Example of the construction in the proof of Menger's Theorem on static graphs. Blue s, X-paths in G_1 are combined with red X, z-paths in G_2 to form 3 vertex disjoint s, z-paths in G.

to the next three difficulties. First, it is not clear that these paths agree on the arrival and starting time, i.e., it might happen that P_1 arrives in $x \in \{u, v\}$ after the starting time of P_1'. Second, even if they do agree, it could be that the combination of such paths produce walks instead of paths. Indeed, as can be seen if Figure 3.11, two t-vertex disjoint s, z-walks do not necessarily contain two t-vertex disjoint paths. And third, as it happens in the proof of Menger's Theorem on static graphs, it is not clear that $\mathcal{G}_1$ and $\mathcal{G}_2$ are in fact smaller than $\mathcal{G}$, and that induction hypothesis can be applied. All of these issues are dealt with by the definition of what was called *extreme separator*, which can be seen as a separator where (u, t_u) is the closest to s as possible, while (v, t_v) is the closest to z as possible. By then investigating the properties of the temporal graph, the authors are able to prove that such a separator either exists, or we can trivially find the desired paths. Finally, they show that such separator, if it exists, solves all the other issues. The interested reader can access the full proof by Ibiapina, Lopes, et al. (2022).

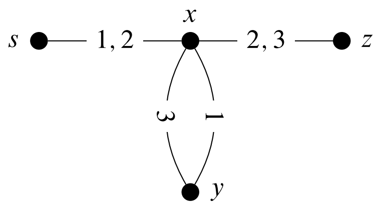


Figure 3.11: Example of graph containing 2 t-vertex disjoint s, z-walks and no 2 t-vertex disjoint s, z-paths.

3.4.2 Complexity of t-vertex disjoint paths problems

In this section, we study the (parameterized) complexity of problems t-vertex Disjoint Paths and t-vertex Path Separator, as well as their directed and strict versions. We start by presenting some negative results.

Negative results on t-vertex disjoint paths problems

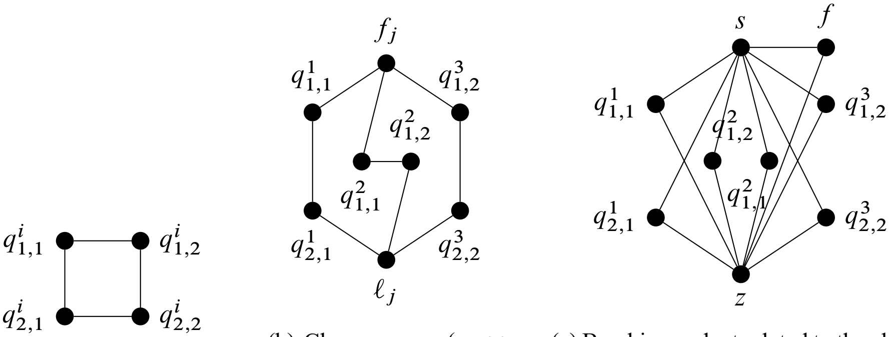
Ibiapina, Lopes, et al. (2022) present two main constructions from which they derive all their negative results. The first one is a reduction from a well known variation of SAT, defined below and known to be NP-complete (Dehghan and Ahadi (2019)).

(2, 2, 3)-SAT
Input. A CNF formula ϕ where each clause has at most 3 literals, and each literal appears exactly twice positively and exactly twice negatively.
Question. Is there a satisfying assignment to ϕ?

We present how the construction works and leave the proof of correctness as an exercise (Exercise 15).

Theorem 20 (Ibiapina, Lopes, et al. (2022))**.** *t-vertex Disjoint Paths is NP-complete.*

Proof. The reader should convince themselves that this problem is in NP. For the hardness part, we make a reduction from (2, 2, 3)-SAT, as previously said. So consider an instance ϕ of (2, 2, 3)-SAT, and let $\{x_1, \dots, x_n\}$ be its set of variables and $\{c_1, \dots, c_m\}$ be its set of clauses. Let s_j denote the number of literals in c_j for each $j \in [m]$.



(a) Variable x_i.

(b) Clause $c_j = (x_1 \vee \neg x_2 \vee x_3)$.

(c) Breaking gadget related to the clause in 3.12b.

Figure 3.12: Reduction in Theorem 20.

We construct a temporal graph $\mathcal{G}$ together with a pair of vertices s, z such that ϕ is satisfiable if and only if there exist ρ t-vertex disjoint s, z-paths in $\mathcal{G}$, where $\rho = 1 + m +$

$\sum_{j=1}^{m-1} s_j$. Start by adding two vertices s and z and creating the variable gadgets. The idea is that each variable x_i will be related to a square whose columns will represent the positive appearances of x_i and rows will represent the negative appearances of x_i. Formally, let Q_i be the square (cycle on 4 vertices) $(q^i_{1,1}, q^i_{1,2}, q^i_{2,2}, q^i_{2,1})$ (see Figure 3.12a). Now, for each clause c_j, do as follows (see Figure 3.12b). Add two new vertices, f_j and ℓ_j, and for each variable x_i appearing in c_j, add an f_j, ℓ_j-path passing by the h-th column of Q_i, if this is the h-th appearance of x_i, or by the h-th row of Q_i, if this is the h-th appearance of $\neg x_i$. This subgraph is denoted by H_j. We also create what is called a *breaking gadget* related to c_j (see Figure 3.12c), which consists simply of adding all edges between $\{s, z\}$ and $V(H_j) \setminus \{\ell_j\}$; denote this by B_j. These are created only for $c_1, \dots, c_{m-1}$. Finally, add edges $\{sf_1, \ell_m z\}$, identify vertices ℓ_j and f_{j+1} for each $j \in [m-1]$, and let G be the obtained graph. Now, we assign time labels in a way that sf_1 is the first active edge, then each clause gadget is active in its own timestep, followed by its breaking gadget, with the last edge being $\ell_m z$. Formally, let $\mathcal{G} = (G, \lambda)$ be such that $\lambda(sf_1) = \{1\}$, $2j \in \lambda(e)$ for every $j \in [m]$ and every $e \in E(H_j)$, $\lambda(e) = \{2j+1\}$ for every $j \in [m-1]$ and every $e \in E(B_j)$, and $\lambda(\ell_m z) = \{2m+1\}$. This is best seen as a sequence of graphs. See Figure 3.13. In the proof of correctness, left as exercise, in order to ensure consistency of variable assignment, it is crucial to use the fact that the path passing by all clause gadgets must indeed be a path (i.e, no vertex repetition is allowed).

□

The reader should also observe that the construction presented in Theorem 20 can produce directed temporal graphs. Additionally, by "spreading the snapshots", one can deduce a proof of hardness also for the strict problems. We leave these as exercises (Exercise 16).

Corollary 21 (Ibiapina, Lopes, et al. (ibid.)). *Dir t-vertex Disjoint Paths, Strict t-vertex Disjoint Paths, and Dir Strict t-vertex Disjoint Paths are* NP*-complete.*

Now, let $N^T(s)$ denote the set $\{(u,i) \mid (su,i) \in E^T(\mathcal{G})\}$; we call this set the *temporal neighborhood of s*. As an example, in Figure 3.13 we have

$$N^T(s) = \{(f_1,1), (f_1,3), (q^1_{1,1},3), (q^1_{2,1},3), (q^2_{1,1},3), (q^2_{1,2},3)\}.$$

Observe that $X = N^T(s) \setminus \{(f_1,1)\}$ is a temporal t-vertex s,z-path-separator if and only if there are at most $\rho - 1$ t-vertex disjoint s,z-paths in $\mathcal{G}$. In other words, the answer to the following problem on $(\mathcal{G}, s, z, X)$ is "yes" if and only if the answer to t-vertex Disjoint Paths on $(\mathcal{G}, s, z, \rho)$ is "no". This gives us that such problem is co-NP-hard, in contrast with the analogous versions of the problem for temporal vertex s,z-separators and temporal t-vertex s,z-separator, which are polynomial-time solvable.

Temporal t-vertex Path Separator Testing
Input. A temporal (directed) graph $\mathcal{G}$ with lifetime τ, a pair of non-adjacent vertices s, z, and a subset $X \subseteq (V(G) \setminus \{s,z\}) \times [\tau]$.
Question. Is X a temporal t-vertex s,z-path-separator?

Observe additionally that the problem is in co-NP. Indeed, if X is *not* a temporal t-vertex s, z-path-separator, then a temporal s, z-path not intersecting X is a certificate that can be checked in polynomial time.

Theorem 22 (Ibiapina, Lopes, et al. (2022)). *Temporal t-vertex Path Separator Testing is co-NP-complete.*

Now, observe also that any temporal t-vertex s, z-path-separator must contain X as $X \subseteq N^T(s) \cap N^T(z)$, i.e., the only way to break the temporal path passing by $\alpha \in X$ is by removing α. This gives us that $\mathcal{G}$ has a temporal t-vertex s, z-path-separator of size $|X|$ if and only if X is a temporal t-vertex s, z-path-separator. Because the proof can be adapted to work on the directed and strict versions, we get the following corollary. Note also that in this case a polynomial certificate for a no instance is not clear to exist, i.e., it is not known whether the problems below are also in co-NP (and hence are co-NP-complete).

Corollary 23 (Ibiapina, Lopes, et al. (ibid.)). *(Dir) (Strict) t-vertex Path Separator is co-NP-hard.*

Unfortunately, this construction does not bound the number of paths/size of separator, nor the lifetime. This means that it does not give us any insight into how hard it would be to get FPT algorithms when parameterizing by such parameters. Ibiapina et al. present a proof bounding the number of paths, and the lifetime, but only for the directed case. We refrain to present this reduction, as it is from 2-Linkage and has same characteristics as the one presented in Section 3.2.2.

Theorem 24 (Ibiapina, Lopes, et al. (ibid.)). *Let $\mathcal{G}$ be a temporal directed graph, and s, z be a pair of non-adjacent vertices of $\mathcal{G}$. Solving Dir Vertex Disjoint Paths is NP-complete on $(\mathcal{G}, s, z, 3)$ even if $\mathcal{G}$ has lifetime 3.*

Positive results on t-vertex disjoint paths problems

By Theorem 23, Temporal t-vertex Path Separator Testing is co-NP-complete, which is in stark contrast with the other types of separator (see Exercises 6 and 13). A natural question is whether such problem is still hard even for bounded-length sets. This is answered negatively with an algorithm presented in the next theorem, which is FPT when parameterized by the size of the input set X. Observe that as a consequence we also get that (Dir) (Strict) t-vertex Path Separator is XP when parameterized by the size of the separator.

Theorem 25 (Ibiapina, Lopes, et al. (ibid.)). *Let $\mathcal{G}$ be a temporal (directed) graph with lifetime τ, s, z be a pair of non-adjacent vertices of $\mathcal{G}$, and $X \subseteq (V(\mathcal{G}) \setminus \{s, z\}) \times [\tau]$ be of size h. One can solve Temporal t-vertex Path Separator Testing in time $O(h^h)$. The same holds if only strict temporal paths are allowed.*

Proof. Let V_X denote the set $\{u \in V(\mathcal{G}) \mid (u, i) \in X \text{ for some } i \in [\tau]\}$. Observe that, if P is a (strict) temporal s, z-path not passing by X, but containing some $u \in V_X$, then the temporal edges incident in u belonging to P, say (vu, i) and (uw, j), are such that $(u, \ell) \notin X$ for every $\ell \in \{i, i+1, \dots, j\}$. On the other hand, if suppose we obtain an auxiliary temporal (directed) graph $\mathcal{G}'$ such that, for every $u \in V_X$, whenever (vu, i) and (uw, j) are temporal edges of $\mathcal{G}'$, we have that $(u, \ell) \notin X$ for every $\ell \in \{i, i+1, \dots, j\}$, then we can argue that any (strict) temporal s, z-walk in $\mathcal{G}$ contains a (strict) temporal s, z-path that does not intersect X. Since finding (strict) temporal walks can be done in polynomial time (see Chapter 2), by creating the right set of instances $\mathcal{G}'$, one can arrive to the desired algorithm. We leave the formalization as exercise. □

As previously observed, the theorem above gives us an XP algorithm to solve (Dir) (Strict) t-vertex Path Separator. Indeed, it suffices to solve Temporal t-vertex Path Separator Testing on X, for every $X \subseteq (V(\mathcal{G}) \setminus \{s, z\}) \times [\tau]$. This takes total time $O((hn\tau)^h)$, where n is the number of vertices in $\mathcal{G}$.

Corollary 26. *(Dir) (Strict) t-vertex Path Separator can be solved in time $O((hn\tau)^h)$ on instance $(\mathcal{G}, s, z, h)$.*

Now, consider problem (Dir) t-vertex Disjoint Paths on $(\mathcal{G}, s, z, 2)$. We can apply the described algorithm running in time $O(n\tau)$ to find out whether $\mathcal{G}$ has a temporal t-vertex s, z-path-separator of size 1. If so, by Theorem 19 we get that the maximum number of t-vertex disjoint s, z-paths is also 1, and we can return "no". Otherwise, again by Theorem 19 we get that the maximum number of t-vertex disjoint s, z-paths is at least 2, and hence we can return "yes". This is a simple algorithm to solve (Dir) t-vertex Disjoint Paths when the number of searched paths is 2, which contrasts with the vertex disjoint paths problem (recall the theorems in Section 3.2.2). In fact, Ibiapina et al. have used Theorem 19 to provide a polynomial-time algorithm that also *finds* 2 t-vertex disjoint s, z-paths, if they exist.

Theorem 27 (Ibiapina, Lopes, et al. (ibid.)). *Let $\mathcal{G}$ be a temporal (directed) graph with lifetime τ, and s, z be a pair of non-adjacent vertices of $\mathcal{G}$. One can find 2 t-vertex disjoint s, z-paths in $\mathcal{G}$ in time $O(mn\tau^2)$, if they exist, where $m = |E(\mathcal{G})|$ and $n = |V(\mathcal{G})|$.*

Proof. The general idea of the proof is to remove temporal edges from $\mathcal{G}$ while ensuring the existence of 2 t-vertex disjoint temporal s, z-paths. This is done by applying Theorem 19 and the algorithm described in Theorem 25 as an oracle. Indeed, if it holds that $tpc(s, z) \geqslant 2$, then Theorem 19 also holds, i.e., we can test whether $tp_{\mathcal{G}}(s, z) \geqslant 2$ in time $O(n\tau)$, as described previously. So we start by applying such test, returning "no" if this is the case. Otherwise, we try to remove a temporal edge α, testing whether $tp_{\mathcal{G}-\{\alpha\}}(s, z) \geqslant 2$; if the answer is "yes", then we remove α, and we keep it otherwise. By iteratively applying this, one arrives to a temporal graph $\mathcal{G}'$ formed exactly by 2 t-vertex disjoint s, z-paths; this takes time $O(mn\tau^2)$, as there are $O(m\tau)$ temporal edges. Finally, we finish the algorithm by applying a search that runs in time $O(n)$ and finds the paths in $\mathcal{G}'$. The reader interested in the formal arguments should go to the manuscript by Ibiapina, Lopes, et al. (ibid.). □

Observe that the algorithm described previously cannot be applied to the strict case, as Theorem 19 does not hold for strict temporal paths (recall the example in Figure 3.8b).

	k	τ	$k+\tau$
tVDP	NPc - Theorem 20 Poly $k = 2$ - Theorem 27	NPc - Theorem 20 Poly $\tau = 2$ (Exercise 18)	NPc - Theorem 20
Dir tVDP	pNP $k = 3$ - Theorem 24 Poly $k = 2$ - Theorem 27	pNP $\tau = 3$ - Theorem 24 Poly $\tau = 2$ (Exercise 18)	pNP $k = \tau = 3$ Theorem 24
Strict tVDP	NPc - Th.21		
Strict Dir tVDP	NPc - Th.21		

Table 3.3: All results presented in this table are presented by Ibiapina, Lopes, et al. (2022). tVDP stands for t-vertex Disjoint Paths; NPc stands for NP-complete; pNP stands for para-NP-complete. Note that in the second row, we get para-NP-completeness for all parameters, while the others are open.

Summing up

We close this section presenting tables that summarize the known complexity results. Observe that most of the results actually classify the problems in the classical computational complexity, and hence the parameterized complexity of most of them is still open. We present in terms of the parameters anyway just to maintain the same pattern as the one used in Section 3.2.

	h	τ	$h+\tau$
t-vertex Path Separator Dir tVSP Strict tVSP Strict Dir tVSP	co-NP-hard - Theorem 23 XP in h and $h+\tau$ - Theorem 26		

Table 3.4: All results presented in this table are presented by Ibiapina, Lopes, et al. (2022). All problems in this table are co-NP-hard, and XP when parameterized by the size of the separator h, which means that it is also in XP when parameterized by the sum $k + \tau$. All other parameterized complexities are open.

3.5 Exercises

1. Prove that Vertex Disjoint Paths is in NP, as well as t-vertex Disjoint Walks and t-vertex Disjoint Paths. Can your algorithms be adapted to work also on the directed and on the strict versions of such problems?

2. Prove that Vertex Separator is in NP, as well as t-vertex Separator. Can your algorithms be adapted to work also on the directed and on the strict versions of such

problems?

3. Prove that there are no 2 vertex disjoint temporal s, z-paths in the temporal graph depicted in Figure 3.3.

4. Prove the second part of Theorem 9.

5. Find a temporal graph $\mathcal{G}$, together with a pair of vertices s and z, such that there are no 2 t-vertex disjoint s, z-paths, and arbitrarily many t-vertex disjoint s, z-walks.

6. Find a polynomial-time algorithm that, given a temporal (directed) graph $\mathcal{G}$, a pair of non-adjacent vertices $s, z \in V(\mathcal{G})$, and $S \subseteq V(\mathcal{G}) \setminus \{s, z\}$, decides whether S is a temporal vertex s, z-separator in $\mathcal{G}$. Do the same for the case of strict temporal walks.

7. Give an XP algorithm for the problem VERTEX SEPARATOR when parameterized by the size of the desired separator.

8. Prove Theorem 14.

9. Let $\mathcal{G}$ be the temporal graph depicted in Figure 3.8a. Prove that $tpc(s, z)$ is equal to 3.

10. Construct an example where $tp(s, z)$ is arbitrarily smaller than $tpc(s, z)$.

11. Prove Theorem 18

12. In the strict model, construct an example where $tp(s, z)$ is arbitrarily smaller than $tpc(s, z)$.

13. Find a polynomial-time algorithm that decides, given a temporal (directed) graph $\mathcal{G}$ with lifetime τ, a pair of vertices $s, z \in V(\mathcal{G})$, and $S \subseteq (V(\mathcal{G}) \setminus \{s, z\}) \times [\tau]$, decides whether S is a temporal t-vertex s, z-separator in $\mathcal{G}$. Do the same for the case of strict temporal paths.

14. Suppose that, for every $\mathcal{G}'$ such that $|V(\mathcal{G}')| + |E^T(\mathcal{G}')| \leqslant n$ and every $s, z \in V(\mathcal{G}')$ with $s'z' \notin E(\mathcal{G}')$, we have that if $tpc_{\mathcal{G}'}(s', z') > 1$, then $tp_{\mathcal{G}'}(s', z') > 1$. Now, consider $\mathcal{G}$ such that $|V(\mathcal{G})| + |E^T(\mathcal{G})| = n+1$, and let $s, z \in V(\mathcal{G})$ with $sz \notin E(\mathcal{G})$. Prove that if $tpc_{\mathcal{G}}(s, z) > 2$, then $tp_{\mathcal{G}'}(s, z) > 1$.

15. Prove that the reduction presented in Theorem 20 is correct.

16. Adapt the proof of Theorem 20 to prove Theorems 21 and 23.

17. Formalize the algorithm described in the proof of Theorem 25.

18. Let $\mathcal{G}$ be a temporal graph with lifetime 2 and consider $s, z \in V(\mathcal{G})$. Prove that the maximum number of t-vertex disjoint temporal s, z-paths in $\mathcal{G}$ is equal to the maximum number of t-vertex disjoint temporal s, z-walks. Use this to deduce that (DIR) T-VERTEX DISJOINT PATHS is polynomial-time solvable on $\mathcal{G}$.

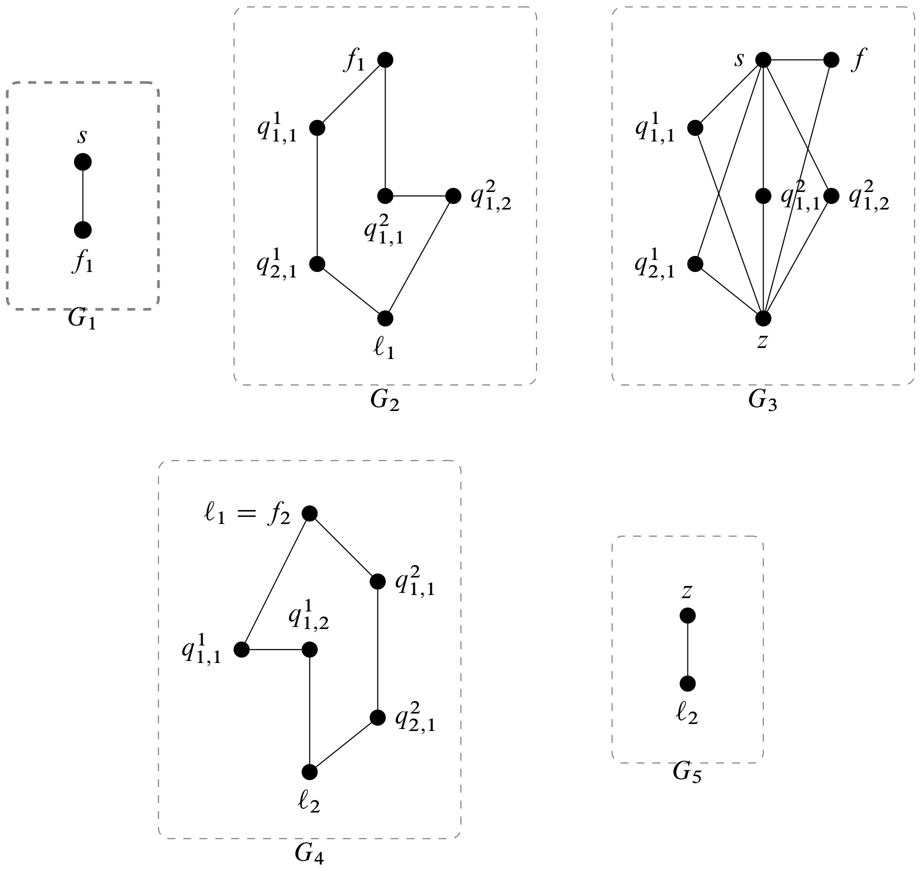


Figure 3.13: Example of construction of Theorem 20 related to formula $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$. Isolated vertices are omitted in the snapshots.

Bibliography

K. A. Berman (1996). "Vulnerability of scheduled networks and a generalization of Menger's theorem." *Networks* 28, pp. 125–134. MR: 1418583. Zbl: 0865.90048 (cit. on pp. 38, 40, 43).

M. Borassi, P. Crescenzi, and M. Habib (2016). "Into the Square: On the Complexity of Some Quadratic-time Solvable Problems." *Electron. Notes Theor. Comput. Sci.* 322, pp. 51–67. MR: 3515493. Zbl: 1345.68170 (cit. on pp. 30–32).

B. Bui-Xuan, A. Ferreira, and A. Jarry (2003). "Computing Shortest, Fastest, and Foremost Journeys in Dynamic Networks." *Int. J. Found. Comput. Sci.* 14.2, pp. 267–285. Zbl: 1075.68545 (cit. on p. 18).

M. Calamai, P. Crescenzi, and A. Marino (2021). "On Computing the Diameter of (Weighted) Link Streams." In: *19th International Symposium on Experimental Algorithms, SEA 2021, June 7-9, 2021, Nice, France*. Ed. by D. Coudert and E. Natale. Vol. 190. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 11:1–11:21. MR: 4288690 (cit. on pp. 30, 31).

V. Campos, R. Lopes, A. Marino, and A. Silva (2021). "Edge-Disjoint Branchings in Temporal Graphs." *The Eletronic Journal of Combinatorics* 28, a.4. MR: 4328901. Zbl: 1476.05065 (cit. on p. 12).

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein (2022). *Introduction to algorithms*. MIT press (cit. on p. 29).

P. Crescenzi, C. Magnien, and A. Marino (2019). "Approximating the Temporal Neighbourhood Function of Large Temporal Graphs." *Algorithms* 12.10, p. 211. MR: 4026822 (cit. on pp. 13, 21, 22).

— (2020). "Finding Top-k Nodes for Temporal Closeness in Large Temporal Graphs." *Algorithms* 13.9, p. 211. MR: 4158546 (cit. on pp. 21, 22, 24, 25).

M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh (2015). *Parameterized Algorithms*. 1st. Springer Publishing Company, Incorporated. MR: 3380745. Zbl: 1334.90001 (cit. on pp. 10, 11).

A. Dehghan and A. Ahadi (2019). “(2/2/3)-SAT problem and its applications in dominating set problems.” *Discrete Mathematics & Theoretical Computer Science* 21. MR: 3995591. Zbl: 1430.05031 (cit. on p. 52).

J. Dibbelt, T. Pajor, B. Strasser, and D. Wagner (2018). “Connection Scan Algorithm.” *ACM Journal of Experimental Algorithmics*. MR: 3863870. Zbl: 07043408 (cit. on p. 24).

S. Fortune, J. Hopcroft, and J. Wyllie (1980). “The directed subgraph homeomorphism problem.” *Theoretical Computer Science* 10.2, pp. 111–121. MR: 0551599. Zbl: 0419.05028 (cit. on p. 40).

P. A. Golovach and D. M. Thilikos (2011). “Paths of bounded length and their cuts: Parameterized complexity and algorithms.” *Discrete Optimization* 8.1, pp. 72–86. MR: 2772562. Zbl: 1248.90071 (cit. on pp. 41, 43–45).

P. Holme (2015). “Modern temporal network theory: a colloquium.” *The European Physical Journal B* 88.39, p. 234 (cit. on p. 1).

A. Ibiapina, R. Lopes, A. Marino, and A. Silva (2022). “Menger’s Theorem for Temporal Paths (Not Walks).” arXiv: 2206.15251 (cit. on pp. 46–49, 51–56).

A. Ibiapina and A. Silva (2022). “Mengerian graphs: characterization and recognition.” arXiv: 2208.06517 (cit. on pp. 4, 22, 39).

R. Impagliazzo, R. Paturi, and F. Zane (2001). “Which Problems Have Strongly Exponential Complexity?” *J. Comput. Syst. Sci.* 63.4, pp. 512–530. MR: 1894519. Zbl: 1006.68052 (cit. on p. 30).

A. Itai, Y. Perl, and Y. Shiloach (1982). “The complexity of finding maximum disjoint paths with length constraints.” *Networks* 12.3, pp. 277–286. MR: 0671829. Zbl: 0504.68041 (cit. on p. 43).

R. M. Karp (1975). “On the computational complexity of combinatorial problems.” *Networks* 5.1, pp. 45–68. Zbl: 0324.05003 (cit. on p. 40).

D. Kempe, J. Kleinberg, and A. Kumar (2000). “Connectivity and inference problems for temporal networks.” In: *STOC ’00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*. MR: 2115287. Zbl: 1296.68015 (cit. on pp. 12, 13, 15, 38, 39, 42, 43).

M. Latapy, T. Viard, and C. Magnien (2018). “Stream graphs and link streams for the modeling of interactions over time.” *Social Network Analysis and Mining* 8 (1), p. 61. Zbl: 1426.68227 (cit. on p. 1).

C.-L. Li, S. T. McCormick, and D. Simchi-Levi (1990). “The complexity of finding two disjoint paths with min-max objective function.” *Discrete Applied Mathematics* 26.1, pp. 105–115. MR: 1028879. Zbl: 0693.05035 (cit. on p. 42).

L. Lovász, V. Neumann-Lara, and M. Plummer (1978). “Mengerian theorems for paths of bounded length.” *Periodica Mathematica Hungarica* 9.4, pp. 269–276. MR: 0509677. Zbl: 0393.05033 (cit. on p. 44).

K. Menger (1927). “Zur allgemeinen kurventheorie.” *Fundamenta Mathematicae* 10.1, pp. 96–115. Zbl: 53.0561.01 (cit. on p. 3).

G. Mertzios, O. Michail, and P. Spirakis (2019). “Temporal Network Optimization Subject to Connectivity Constraints.” *Algorithmica* 81, pp. 1416–1449. MR: 3936163. Zbl: 1421.68139 (cit. on pp. 12, 47).

O. Michail (2016). “An Introduction to Temporal Graphs: An Algorithmic Perspective.” *Internet Mathematics* 12.4, pp. 239–280. MR: 3508358. Zbl: 1461.68161 (cit. on p. 12).

N. Robertson and P. Seymour (1990). *An outline of a disjoint paths algorithm, in “Paths, Flows, and VLSI-Layout”(B. Korte, L. Lov! asz, HJ Pr. omel, and A. Schrijver, Eds.)* MR: 1083383 (cit. on p. 40).

M. Sipser (1996). “Introduction to the Theory of Computation.” *ACM Sigact News* 27.1, pp. 27–29 (cit. on p. 9).

D. B. West (Sept. 2000). *Introduction to Graph Theory*. 2nd ed. Prentice Hall. Zbl: 0845.05001 (cit. on pp. 46, 49).

V. V. Williams and R. Williams (2010). “Subcubic Equivalences between Path, Matrix and Triangle Problems.” In: *51th Annual IEEE Annual Symposium on Foundations of Computer Science*. Las Vegas, Nevada, USA: IEEE Computer Society, pp. 645–654. MR: 3025239 (cit. on p. 30).

H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu (2014). “Path Problems in Temporal Graphs.” *PVLDB* 7.9, pp. 721–732 (cit. on pp. 18–22, 24, 25, 28).

H. Wu, Y. Huang, J. Cheng, J. Li, and Y. Ke (2016). “Reachability and time-based path queries in temporal graphs.” In: *ICDE*, pp. 145–156 (cit. on pp. 18, 21, 22).

P. Zschoche, T. Fluschnik, H. Molter, and R. Niedermeier (2020). “The complexity of finding small separators in temporal graphs.” *Journal of Computer System and Sciences* 107, pp. 72–92. MR: 4015681. Zbl: 1436.68265 (cit. on pp. 13, 43–45).

Index

Andrea Marino

PhD in Computer Science at University of Florence (Italy) in 2013, advised by Pierluigi Crescenzi. Currently Associate Professor at University of Florence. Best Italian PhD Thesis on ”Algorithms, Automata, Complexity and Game Theory” 2013 and Best Italian Young Researcher in ”Theoretical Computer Science” 2022 awarded by Italian Chapter of the EATCS (European Association for Theoretical Computer Science). Interested in: Algorithms and Complexity, Complex Networks analysis, Enumeration Algorithms, Temporal graphs, Boxe, Music (currently into MPB), and Guitars.

Ana Silva

PhD at the University of Grenoble (France) in 2010, advised by Frédéric Maffray. Associate Professor at the Federal University of Ceará since 2011, Brazil. Won the Brazilian edition of the L’Óreal Prize for Women in Science in 2014 (math track), and is currently an affilliated member of the Brazilian Academy of Science (ABC). In addition to her passion for math and general science, Ana also loves (as most “cearences” do) the beach, the sun, a good beer and lots of laughter.

Paths and connectivity in temporal graphs

impa

Instituto de
Matemática
Pura e Aplicada

ISBN 978-85-244-0536-5
9 788524 405365