Social Network Analysis and Mining (2020) 10:50
https://doi.org/10.1007/s13278-020-00664-5

ORIGINAL ARTICLE

Comparing temporal graphs using dynamic time warping

Vincent Froese[1] · Brijnesh Jain[2] · Rolf Niedermeier[1] · Malte Renken[1]

Received: 27 February 2020 / Revised: 30 May 2020 / Accepted: 4 June 2020 / Published online: 29 June 2020

Abstract
Within many real-world networks, the links between pairs of nodes change over time. Thus, there has been a recent boom in studying temporal graphs. Recognizing patterns in temporal graphs requires a proximity measure to compare different temporal graphs. To this end, we propose to study dynamic time warping on temporal graphs. We define the dynamic temporal graph warping (dtgw) distance to determine the dissimilarity of two temporal graphs. Our novel measure is flexible and can be applied in various application domains. We show that computing the dtgw-distance is a challenging (in general) NP-hard optimization problem and identify some polynomial-time solvable special cases. Moreover, we develop a quadratic programming formulation and an efficient heuristic. In experiments on real-world data, we show that the heuristic performs very well and that our dtgw-distance performs favorably in de-anonymizing networks compared to other approaches.

Keywords Temporal graph matching · Vertex signatures · Heuristic optimization · Quadratic programming · Parameterized algorithms

1 Introduction

A fundamental concept for pattern recognition is the notion of proximity (i.e., (dis)similarity) between objects. For objects that are represented by numerical feature vectors, there exist a lot of well-known proximity measures such as p-norms or positive semi-definite kernels. In structural pattern recognition, objects are often more naturally represented by complex (discrete) data structures such as graphs, strings, or time series. For these representations, one can often not simply use vector-based proximity measures. Instead, one needs to define suitable domain-specific proximity measures such as the *edit distance* on graphs or strings or the *dynamic time warping* distance[1] on time series.

The majority of graph proximity measures focus on static graphs. This includes the graph edit distance (Riesen 2015), graph kernels (Kriege et al. 2020), and geometric graph distances (Jain 2016). However, many complex systems are not static as the links between entities dynamically change over time. Thus, there is a steadily growing research interest in analyzing *temporal graphs* (we also use the term *temporal network* interchangeably) (Rozenshtein and Gionis 2019; Holme and Saramäki 2013, 2019). Such temporal graphs can be represented by a series of temporal edges between a fixed set of vertices. Examples are social contact networks, disease spreading networks, traffic networks, attack networks in computer security, or protein–protein interaction networks in biology (Kostakos 2009; Holme and Saramäki 2013, 2019; Li et al. 2017; Vijayan et al. 2017). Examples of data mining problems on temporal social networks include community detection (Dakiche et al. 2019), epidemics analysis (Rozenshtein et al. 2016), and influence spreading (Gayraud et al. 2015).

An extended abstract of this article appeared in the *Proceedings of the 8th International Conference on Complex Networks and their Applications*, Springer, SCI, vol 882, pp. 469–480, 2019 (Froese et al. 2019). This article contains all proofs in full detail and presents extended experimental results.

✉ Malte Renken
m.renken@tu-berlin.de

Vincent Froese
vincent.froese@tu-berlin.de

Brijnesh Jain
brijnesh.jain@dai-labor.de

Rolf Niedermeier
rolf.niedermeier@tu-berlin.de

[1] Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Berlin, Germany

[2] Faculty IV, Distributed Artificial Intelligence Laboratory, Technische Universität Berlin, Berlin, Germany

[1] Note that a *distance* function is not required to obey the triangle inequality, in contrast to a *metric*.

Many processes described by temporal graphs naturally vary in duration and temporal dynamics (for example, chemical reactions or the spread of a disease might proceed at different speeds), which makes data mining tasks such as classification challenging. Hence, one needs to find suitable proximity measures, which has seemingly not been done so far.

Our paper proposes such a measure. We introduce a novel proximity measure on temporal graphs based on vertex signature graph distance and dynamic time warping, called *dynamic temporal graph warping* (dtgw). Dynamic time warping allows to cope with variations in temporal dynamics. Thus, by combining established methods from graph-based pattern recognition and time series data mining in a nontrivial way, we obtain a suitable tool to analyze temporal network data. We study the computational complexity of the dtgw-distance, develop efficient algorithms and study their behavior on real-world data, the latter indicating the strong potential for future applications.

Related Work. Graph distance based on vertex mappings using local vertex signatures was introduced by Jouili and Tabbone (2009). The idea of using vertex mappings can also be found in *optimal assignment kernels* (Fröhlich et al. 2005; Kriege et al. 2016; Bento and Ioannidis 2018).

Regarding proximity measures on temporal graphs, seemingly little work has been done so far. In fact, we are not aware of other approaches for numerically measuring the proximity of two temporal graphs. Related concepts, however, have been investigated for temporal graphs. For example, one approach is based on network embeddings where nodes are mapped onto certain feature spaces incorporating the temporal behavior (Ahmed and Karypis 2015; Zuo et al. 2018; Nguyen et al. 2018). Another approach is based on network alignments (Vijayan et al. 2017; Elhesha et al. 2019) where a vertex mapping that optimizes some criteria is computed. However, dynamic time warping has not been used in this context so far.

Dynamic time warping (Sakoe and Chiba 1978) is an established measure for mining time series data (Rakthanmanon et al. 2012; Wang et al. 2013) which is specifically designed to cope with temporal distortion in the data via nonlinear alignment of time series. It can be applied to time series of different lengths (in contrast to the Euclidean distance for example) which is a relevant aspect in time series averaging. We lift this approved concept to the domain of temporal graphs.

Our Contributions. We define the dynamic temporal graph warping (dtgw) distance as a twofold discrete minimization problem involving computation of an optimal vertex mapping and an optimal "warping path" (see Sect. 3). As a by-product, our approach does not only yield a distance measure but also yields an interpretable mapping between vertices of the two temporal graphs which can, for example, be used for de-anonymization of individuals in social networks.

We show that the dtgw-distance is NP-hard to compute in general (Theorem 1). In contrast, we point out several polynomial-time solvable special cases. This includes the case when either a vertex mapping or a temporal alignment is fixed (Observation 1), the case of deciding whether the dtgw-distance is zero (Theorem 2), and the case when the lifetimes of the two temporal graphs differ only by a constant and the warping path length is restricted (Proposition 3). Moreover, we give a quadratic programming formulation (Sect. 5.2) and propose an efficient and effective heuristic approach (Sect. 5.3).

We empirically evaluate the heuristic[2] in preliminary experiments on real-world temporal social networks (face-to-face contact and spatial proximity networks) against the quadratic program and some simple baseline methods to show its efficiency and solution quality. Moreover, we demonstrate that our concept can successfully be used for de-anonymization of real-world temporal social networks and is faster than other existing methods such as DynaMAGNA++ (Vijayan et al. 2017) or HTNE (Zuo et al. 2018) (Sect. 6).

Compared to the conference version (Froese et al. 2019), this version contains the NP-hardness proof of Theorem 1, the proof of Proposition 3, and the quadratic programming formulation (Sect. 5.2). Moreover, we present additional experimental results including a comparison of the heuristic with the quadratic program (Sect. 6.2) and a clustering experiment to demonstrate robustness with respect to noisy data (Sect. 6.3).

Organization. Section 2 contains basic definitions. Section 3 presents our main definition of the dtgw-distance followed by NP-hardness results in Sect. 4 and positive algorithmic results in Sect. 5. Section 6 presents experimental results on some real-world data. We conclude in Sect. 7 with an outlook on future applications and challenges.

2 Preliminaries

For $T \in \mathbb{N}$, we define $[T] := \{1, 2, \dots, T\}$.

Temporal Graphs. A *temporal graph* $\mathcal{G} = (V, E_1, E_2, \dots, E_T)$ consists of a vertex set V and a sequence of $T \geq 1$ edge sets E_i, each being a set of unordered vertex pairs. By $G_i = (V, E_i)$, we denote the i^{th} *layer* of $\mathcal{G}$ and we call T the *lifetime* of $\mathcal{G}$. The *underlying graph* of $\mathcal{G}$ is the graph $(V, \bigcup_{i=1}^{T} E_i)$. We remark that all definitions and results in this work can easily be extended to labeled temporal graphs (with vertex and/or edge labels).

[2] An implementation is freely available at www.akt.tu-berlin.de/menue/software.

Vertex Mapping. A *vertex mapping* between two vertex sets V and W is a set $M \subseteq V \times W$ containing $\min(|V|, |W|)$ tuples such that each $x \in V \cup W$ is contained in at most one tuple of M. We denote the set of all vertex mappings between V and W by $\mathcal{M}(V, W)$. Let $V_M \subseteq V$ be the subset of vertices in V that are contained in some tuple of M ($W_M \subseteq W$ is defined analogously). Note that $V_M = V$ or $W_M = W$ holds since $|M| = \min(|V|, |W|)$.

Assignment Problem. Computing optimal vertex mappings between two temporal graphs can be solved via the Assignment Problem which is a fundamental problem in combinatorial optimization. Given two sets A and B of equal size and a cost function $c : A \times B \to \mathbb{Q}$, the goal is to find a bijection $\pi : A \to B$ such that $\sum_{a \in A} c(a, \pi(a))$ is minimized. It is well known that the Assignment Problem is solvable in $\mathcal{O}(|A|^3)$ time (Ahuja et al. 1993, Theorem 12.2).

Dynamic Time Warping. The dynamic time warping distance (Sakoe and Chiba 1978) is a distance between time series. It is based on the concept of a warping path. A *warping path of order $n \times m$* is a set $p = \{p_1, \ldots, p_L\}$ of $L \geq 1$ pairs $p_\ell = (i_\ell, j_\ell)$ such that

- $p_1 = (1, 1)$ and $p_L = (n, m)$, and
- $p_{\ell+1} \in \{(i_\ell + 1, j_\ell + 1), (i_\ell, j_\ell + 1), (i_\ell + 1, j_\ell)\}$ for all $1 \leq \ell < L$.

We denote the set of all warping paths of order $n \times m$ by $\mathcal{P}_{n,m}$. For two temporal graphs $\mathcal{G} = (V, E_1, \ldots, E_T)$, $\mathcal{H} = (W, F_1, \ldots, F_U)$, every order-$(T \times U)$ warping path p defines a *warping* between $\mathcal{G}$ and $\mathcal{H}$, that is, a pair $(i, j) \in p$ *warps* layer G_i to layer H_j.

Parameterized Complexity. We assume the reader to be familiar with basic concepts of computational complexity theory such as NP-completeness (Garey and Johnson 1979). In parameterized complexity theory (Downey and Fellows 2013; Cygan et al. 2015), one considers running times with respect to two dimensions. One dimension is the size of the input instance x, and the other dimension is a *parameter* k (usually a numerical value). An instance of a parameterized problem is a pair (x, k). The class XP contains all parameterized problems that can be solved in polynomial time for every constant parameter value, that is, in $|x|^{f(k)}$ time for some function f only depending on k.

3 Dynamic temporal graph warping (DTGW)

In this section, we define our temporal graph distance based on dynamic time warping using a vertex-signature-based graph distance as cost function. We choose this graph distance for the following reasons. First, in contrast to the NP-hard edit distance, it is polynomial-time computable. Second, it is based on a mapping between the two vertex sets which allows to enforce a consistency over time. This consistency assumption is useful in many temporal network applications where the vertices in both networks correspond to the same set of objects over time. This implicitly allows to identify vertices within the two networks. Third, vertex signatures allow for a high flexibility since they can be chosen arbitrarily in order to incorporate essential information (local or global) for the application at hand (e.g., one might use feature vectors obtained via network embedding).

Graph Distance Based on Vertex Signatures.

The following approach is due to Jouili and Tabbone (2009). For a (static) graph $G = (V, E)$, a *vertex signature function* $f_G : V \to \mathbb{Q}^k$ encodes arbitrary information about a vertex. Let $d : \mathbb{Q}^k \times \mathbb{Q}^k \to \mathbb{Q}$ be a metric.

For two (static) graphs $G = (V, E)$ and $H = (W, F)$ with vertex signatures $f_G : V \to \mathbb{Q}^k$ and $f_H : W \to \mathbb{Q}^k$ and a given vertex mapping M between V and W, we define the *cost* of M as

$$C(G, H, M) := \sum_{(u,v) \in M} d\big(f_G(u), f_H(v)\big) + \sum_{v \in V \setminus V_M} \Delta_G(v) + \sum_{v \in W \setminus W_M} \Delta_H(v),$$

where $\Delta_G(v) \in \mathbb{Q}$ is the (predefined) cost of "deleting" vertex v from G since it is not mapped by M to any vertex in the other vertex set. The value $\Delta_G(v)$ might, for example, depend on the vertex signature of v. Note that "deleting" a vertex does not affect the signatures of other vertices. Note also that one of the last two sums on the right-hand side above is always zero.

The vertex-signature-based distance between G and H is then defined as

$$D(G, H) := \min_{M \in \mathcal{M}(V,W)} C(G, H, M).$$

Depending on the application, one might normalize the distance D by some appropriate factor (typically depending on $|V|$ and $|W|$; e.g., Jouili and Tabbone (2009) normalized by $\min(|V|, |W|)^{-1}$).

Throughout this work, we assume that vertex signature functions f_G are computable in polynomial time in the size of G and we assume all metrics d to be polynomial-time computable. In the rest of the paper, we neglect the running times for computing the values of f_G and d because we assume that all vertex signatures are precomputed once in polynomial time.

Dynamic Time Warping Distance for Temporal Graphs.

We transfer the concept of dynamic time warping to temporal graphs in the following way. Let $\mathcal{G} = (V, E_1, \ldots, E_T)$ and $\mathcal{H} = (W, F_1, \ldots, F_U)$ be two temporal graphs, and let $f_{G_1}, \ldots, f_{G_T} : V \to \mathbb{Q}^k$ and $f_{H_1}, \ldots, f_{H_U} : W \to \mathbb{Q}^k$ be the corresponding vertex signature functions.

We then define the vertex-signature-based *dynamic temporal graph warping distance* (dtgw-distance) between $\mathcal{G}$ and $\mathcal{H}$ as

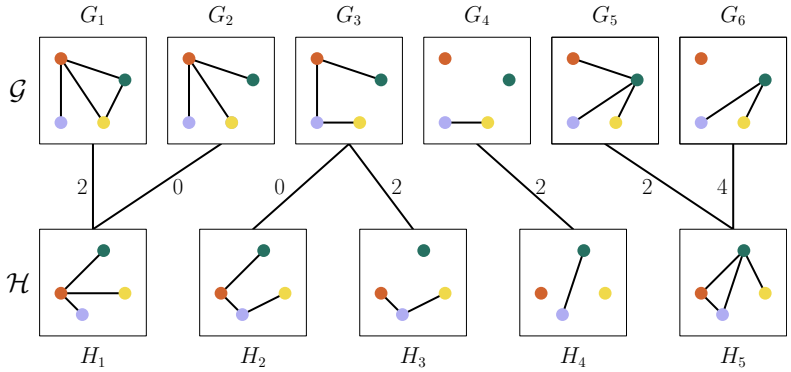


Fig. 1 Example of the dtgw-distance between two temporal graphs $\mathcal{G}$ and $\mathcal{H}$ on four vertices with lifetimes six and five. The vertex coloring indicates an optimal vertex mapping M. The connections between the boxes indicate an optimal warping path $p = \{(1,1),(2,1),(3,2),(3,3),(4,4),(5,5),(6,5)\}$. Their labels correspond to the costs $C(G_i, H_j, M)$ where the vertex signatures are the degrees and the metric is the absolute value of the difference. For example, the cost of warping G_1 to H_1 is 2, as the green and yellow vertex each have degree two in G_1 but only degree one in H_1. The resulting dtgw-distance is $\mathrm{dtgw}(\mathcal{G},\mathcal{H}) = 2+0+0+2+2+2+4 = 12$

$$\mathrm{dtgw}(\mathcal{G},\mathcal{H}) := \min_{M\in\mathcal{M}(V,W)} \min_{p\in\mathcal{P}_{T,U}} \sum_{(i,j)\in p} C(G_i, H_j, M).$$

Figure 1 depicts an example illustrating the dtgw-distance of two temporal graphs. Intuitively, the vertex mapping identifies vertices with similar behavior over time and the warping path identifies the time layers with similar vertex behavior. Note that (for $T = U$) if one fixes $p = \{(1,1),(2,2),\ldots,(T,T)\}$, then we get a temporal graph distance without time warping (similar to the Euclidean distance).

The following results are easily observed and play a central role for our subsequent algorithms.

Observation 1 *Let $\mathcal{G} = (V, E_1, \ldots, E_T)$ and $\mathcal{H} = (W, F_1, \ldots, F_U)$ be two temporal graphs with $|V| \le |W| =: n$.*

1. *For a fixed vertex mapping $M \in \mathcal{M}(V,W)$, a warping path $p \in \mathcal{P}_{T,U}$ which minimizes $\sum_{(i,j)\in p} C(G_i, H_j, M)$ can be computed in $\mathcal{O}(T \cdot U \cdot n)$ time.*
2. *For a fixed warping path $p \in \mathcal{P}_{T,U}$, a vertex mapping $M \in \mathcal{M}(V,W)$ which minimizes $\sum_{(i,j)\in p} C(G_i, H_j, M)$ can be computed in $\mathcal{O}(n^2 \cdot |p| + n^3)$ time.*

Proof *i)* For a given vertex mapping M, an optimal warping path can be computed by a well-known dynamic program for dynamic time warping in $\mathcal{O}(T \cdot U \cdot n)$ time (Sakoe and Chiba 1978). Here, $\mathcal{O}(n)$ is the time for computing the costs $C(G_i, H_j, M)$. Note that faster dynamic time warping algorithms for special cases are known (Abboud et al. 2015; Gold and Sharir 2018; Kuszmaul 2019; Froese et al. 2020).

ii) Let $V' := V \cup Q$, where Q is a set of $|W| - |V|$ dummy vertices (that is, $|V'| = n$) with $Q \cap V = \emptyset$. For every $(u,v) \in V' \times W$, let

$$\sigma(u,v) := \begin{cases} \sum_{(i,j)\in p} d\big(f_{G_i}(u), f_{H_j}(v)\big), & u \in V \\ \sum_{(i,j)\in p} \Delta_{H_j}(v), & u \in Q \end{cases}.$$

Then, we need to find a vertex mapping $M \in \mathcal{M}(V', W)$ that minimizes $\sum_{(u,v)\in M} \sigma(u,v)$. Note that M defines a bijection between V' and W. Hence, computing M is an Assignment Problem instance solvable in $\mathcal{O}(n^3)$ time (Ahuja et al. 1993, Theorem 12.2). Computing all $\sigma(u,v)$ values can be done in $\mathcal{O}(n^2 \cdot |p|)$ time. □

Note that Observation 1 i) implies that if we already know the vertex mapping up to a constant number of vertices, then dtgw can be computed in polynomial time (since we can try out all polynomially many possible vertex mappings). Furthermore, Observation 1 ii) implies that dtgw is polynomial-time computable if the optimal temporal alignment between $\mathcal{G}$ and $\mathcal{H}$ is known beforehand. In particular, dtgw can be computed in polynomial time if one temporal graph has a constant lifetime or a constant number of vertices since there are only polynomially many possible warping paths or polynomially many vertex mappings.

Corollary 1 *The dtgw-distance between two temporal graphs can be computed in polynomial time if at least one of the following applies:*

1. *The vertex mapping is known up to a constant number of vertices.*
2. *The warping path is known.*

3. *At least one of the temporal graphs has a constant lifetime or a constant number of vertices.*

For given vertex signature function and metric, we refer to the decision problem of testing whether two temporal graphs have dynamic temporal graph warping distance at most some given value c by DTGW.

DYNAMIC TEMPORAL GRAPH WARPING (DTGW)

Input : Two temporal graphs $\mathcal{G}$ and $\mathcal{H}$, $c \in \mathbb{Q}$.
Question : Is $\mathrm{dtgw}(\mathcal{G}, \mathcal{H}) \leq c$?

4 Computational hardness

Even though the dynamic time warping distance and the vertex-signature-based graph distance are both computable in polynomial time, their combined application to temporal graphs yields a distance measure that is generally NP-hard to compute. Intuitively, this is due to the fact that the vertex mapping has to be consistent for all layers. This introduces non-trivial dependencies between the time warping and the vertex mapping which render the problem computationally hard. Indeed, this is not a singular case for temporal graph problems where for many cases the temporal counterparts of problems solvable in polynomial time turn NP-hard; examples include the computation of matchings in graphs (Heeger et al. 2019; Baste et al. 2020; Mertzios et al. 2020), short path computations (Casteigts et al. 2019; Fluschnik et al. 2020), or the computation of separators (Zschoche et al. 2020).

Theorem 1 DTGW *is* NP *-complete for every metric when the vertex signatures are vertex degrees.*

Proof DTGW is clearly contained in NP since for a given vertex mapping and warping path (both having polynomial size), one can check in polynomial time whether the dtgw -distance is at most c (also see Observation 1).

To show NP-hardness, we give a polynomial-time many-one reduction from the NP-complete 3 -SAT problem. Let $d : \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$ be any metric, and let $\phi = C_1 \wedge \ldots \wedge C_m$ be an instance of 3 -SAT over the variables $x_1, \ldots, x_n$. Each clause C_j is then a disjunction of three literals $C_j =: \ell_j^1 \vee \ell_j^2 \vee \ell_j^3$, and there is a function $\nu : [m] \times [3] \to [n]$ such that $\ell_j^i \in \{x_{\nu(j,i)}, \overline{x_{\nu(j,i)}}\}$ holds for all ℓ_j^i. Without loss of generality, we assume $m > 8$.

The idea is to represent each literal by a vertex which can be mapped to either $\top$ (`true`) or $\bot$ (`false`). We then build, for each clause, a *clause box* gadget consisting of three consecutive layers. The choice of a warping path will then, for each clause, implicitly select one of its literals, and the costs caused by each clause box will attain their minimum value if and only if that particular literal is mapped to $\top$.

Now, a detailed description of the reduction follows. Let D and D' be two copies of the graph $\coprod_{i=1}^{22m} K_2$ (consisting of $22m$ disjoint edges), where for each vertex $v \in V(D)$ we denote its copy in $V(D')$ by v'. We construct two temporal graphs $\mathcal{G}$ and $\mathcal{H}$. Their vertex sets each contain $2n + 47m + 8$ vertices as follows:

$$V(\mathcal{G}) := \{x_i, \overline{x_i}; i \in [n]\} \cup \{C_j^1, C_j^2, C_j^3 : j \in [m]\} \cup \{X_i, Y_i; i \in [4]\} \cup V(D),$$

$$V(\mathcal{H}) := \{\top_i, \bot_i : i \in [n]\} \cup \{C_j'^1, C_j'^2, C_j'^3 : j \in [m]\} \cup \{X_i', Y_i' : i \in [4]\} \cup V(D').$$

Both temporal graphs have $2n + 26m$ layers defined as follows. For each $i \in [n]$, we set

$$E(G_{2i-1}) := \{\{x_i, \overline{x_i}\}\}, \quad E(H_{2i-1}) := \{\{\top_i, \bot_i\}\},$$
$$E(G_{2i}) := E(D), \qquad E(H_{2i}) := E(D').$$

For $j \in [m]$, we set

$$E(G_{2n+4j-3}) := \{\{X_i, Y_i\}; i \in [4]\},$$
$$E(G_{2n+4j-2}) := \{\{C_j^i, \ell_j^i\}; i \in [3]\},$$
$$E(G_{2n+4j-1}) := \{\{X_i, Y_i\}; i \in [4]\},$$
$$E(G_{2n+4j}) := E(D),$$

and

$$E(H_{2n+4j-3}) := \{\{C_j'^1, \top_{\nu(j,1)}\}, \{\top_{\nu(j,2)}, \bot_{\nu(j,2)}\}, \{\top_{\nu(j,3)}, \bot_{\nu(j,3)}\}, \{C_j'^2, C_j'^3\}\},$$
$$E(H_{2n+4j-2}) := \{\{C_j'^2, \top_{\nu(j,2)}\}, \{\top_{\nu(j,1)}, \bot_{\nu(j,1)}\}, \{\top_{\nu(j,3)}, \bot_{\nu(j,3)}\}, \{C_j'^1, C_j'^3\}\} \cup \{\{X_i', Y_i'\}; i \in [4]\},$$
$$E(H_{2n+4j-1}) := \{\{C_j'^3, \top_{\nu(j,3)}\}, \{\top_{\nu(j,1)}, \bot_{\nu(j,1)}\}, \{\top_{\nu(j,2)}, \bot_{\nu(j,2)}\}, \{C_j'^1, C_j'^2\}\},$$
$$E(H_{2n+4j}) := E(D').$$

Finally, for $j \in [22m]$, we set

$$E(G_{2n+4m+j}) := \{\{X_k, Y_k\} : k \in [4]\},$$
$$E(H_{2n+4m+j}) := \{\{X_k', Y_k'\} : k \in [4]\}.$$

We call the layers containing $|E(D)|$ edges *separation layers*. Furthermore, for each $j \in [m]$ we say that the layers $2n + 4j - 3$, $2n + 4j - 2$, and $2n + 4j - 1$ form the *clause block* corresponding to C_j (see Fig. 2 for an example). Let $c := 42m \cdot d(0, 1)$.

We claim that $\mathrm{dtgw}(\mathcal{G}, \mathcal{H}) \leq c$ if and only if ϕ has a satisfying truth assignment.

"$\Leftarrow$": Given a satisfying assignment $\beta : \{x_1, \ldots, x_n\} \to \{\mathtt{true}, \mathtt{false}\}$ of ϕ, we define the following vertex mapping:

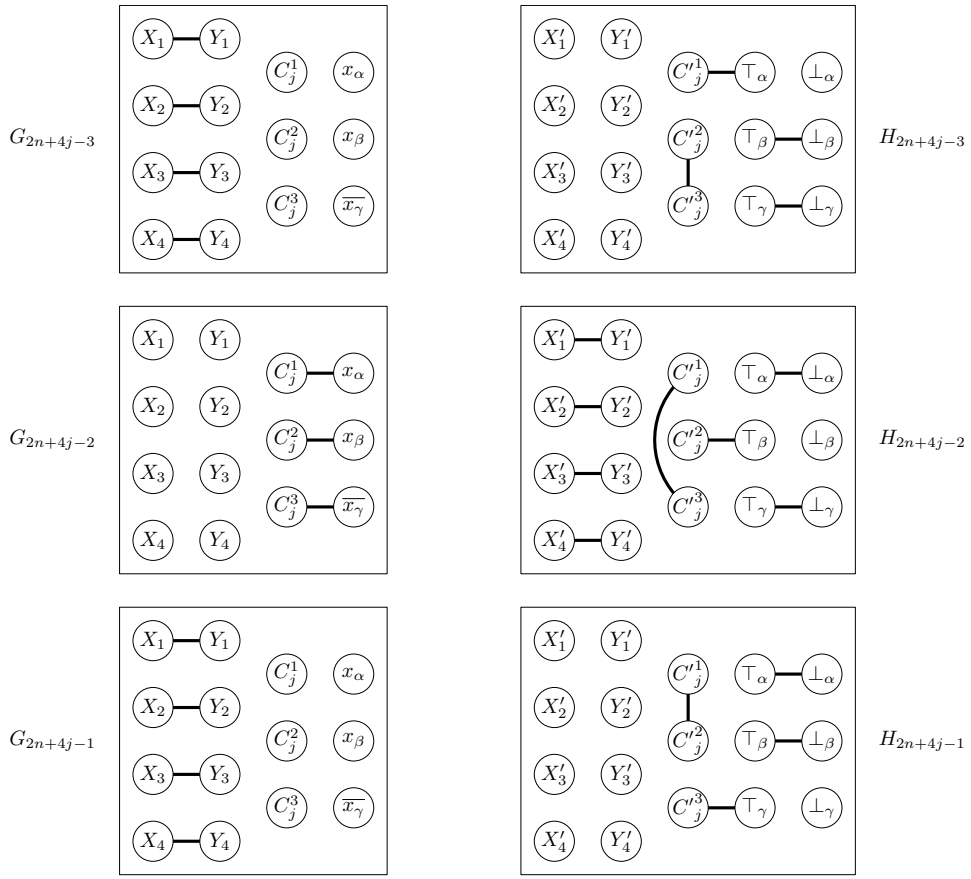


Fig. 2 Clause block for $C_j = x_\alpha \vee x_\beta \vee \overline{x_\gamma}$. Only relevant vertices are shown in each layer

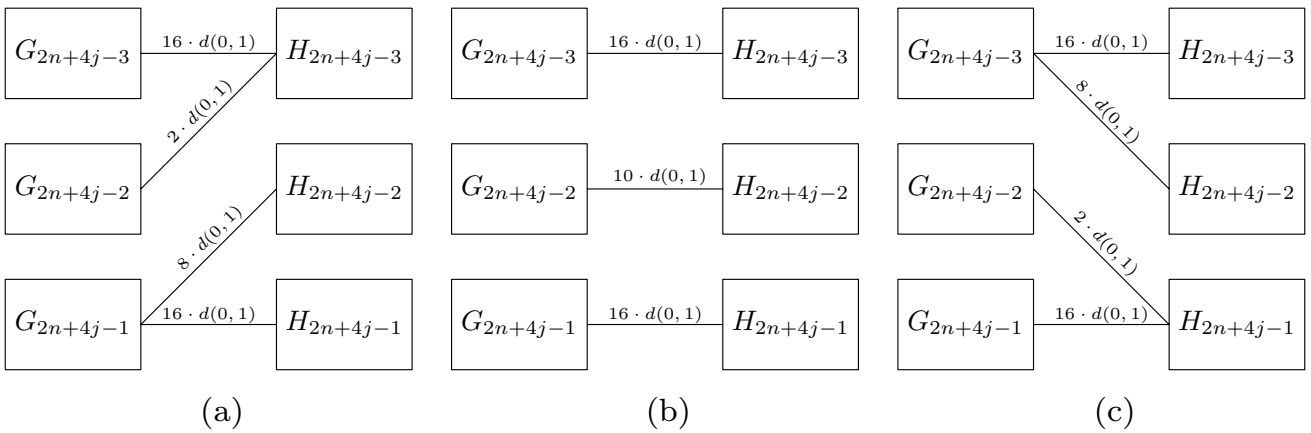


Fig. 3 The three possible warpings between layers of a clause block. Each edge is labeled with the minimal cost it causes under the assumption that the set $\{X_i, Y_i; i \in [4]\}$ is mapped to $\{X'_i, Y'_i; i \in [4]\}$

$$
\begin{aligned}
M := &\{(x_i, \top_i), (\overline{x_i}, \bot_i) : \beta(x_i) = \texttt{true}\} \\
&\cup \{(x_i, \bot_i), (\overline{x_i}, \top_i) : \beta(x_i) = \texttt{false}\} \\
&\cup \{(C_j^i, {C'}_j^i) : i \in [3], j \in [m]\} \\
&\cup \{(X_i, X_i'), (Y_i, Y_i') : i \in [4]\} \\
&\cup \{(v, v') : v \in V(D)\}.
\end{aligned}
$$

To construct a warping path, we begin by defining, for each $j \in [m]$, the following three sub-paths (see also Fig. 3):

$$
\begin{aligned}
\pi_j^1 &:= \big\{(2n+4j-2,\ 2n+4j-3), (2n+4j-1,\ 2n+4j-2)\big\}, \\
\pi_j^2 &:= \big\{(2n+4j-2,\ 2n+4j-2)\big\}, \\
\pi_j^3 &:= \big\{(2n+4j-3,\ 2n+4j-2), (2n+4j-2,\ 2n+4j-1)\big\}.
\end{aligned}
$$

For each clause $C_j = \ell_j^1 \vee \ell_j^2 \vee \ell_j^3$, pick $k_j \in [3]$ such that $\ell_j^{k_j}$ is true. We then build the warping path p as the union of all $\pi_j^{k_j}$, using the trivial warping path for all remaining layers:

$$
p := \{(i,i) : i \in [2n+4m+22m] \setminus \{2n+4j-2; j \in [m]\}\} \cup \bigcup_{j \in [m]} \pi_j^{k_j}.
$$

It is then not difficult to calculate that each clause block adds cost of exactly $42 \cdot d(0,1)$ and there are no other costs. Thus, $\mathrm{dtgw}(\mathcal{G}, \mathcal{H}) \leq 42m \cdot d(0,1) = c$.

"⇒": Now suppose that $\mathrm{dtgw}(\mathcal{G}, \mathcal{H}) \leq c$ and let (M, p) be a pair of vertex mapping M and warping path p with cost $\sum_{(i,j) \in p} C(G_i, H_j, M) = \mathrm{dtgw}(\mathcal{G}, \mathcal{H})$. Note that any non-separation layer contains at most eight edges. So if p warps any separation layer to any non-separation layer, then the resulting cost would be at least $(44m - 16) \cdot d(0,1) > c$. Thus, we may assume that every separation layer i of $\mathcal{G}$ is only warped to layer i of $\mathcal{H}$ and vice versa. Since the last $22m$ layers of each temporal graph are all identical and M and p are chosen to have minimal cost, we can conclude that

$$
p \supset \{(i,i); i \in [2n+4m+22m] \setminus \{2n+4j-2; j \in [m]\}\}.
$$

If M maps some vertex from $\{X_k, Y_k; k \in [4]\}$ to some vertex that is not in $\{X_k', Y_k'; k \in [4]\}$, then the $22m$ layers $2n+4m+1, \ldots, 2n+4m+22m$ each would cause cost of at least $2 \cdot d(0,1)$, thus exceeding c in total. Hence, M has to contain a bijection from $\{X_k, Y_k; k \in [4]\}$ to $\{X_k', Y_k'; k \in [4]\}$.

Now, consider the clause block corresponding to $C_j = \ell_j^1 \vee \ell_j^2 \vee \ell_j^3$. From the arguments above, it follows that $G_{2n+4j-3}$ and $G_{2n+4j-1}$ are warped to $H_{2n+4j-3}$ and $H_{2n+4j-1}$, respectively. This already costs $32 \cdot d(0,1)$. We distinguish three cases (corresponding to π_j^1 through π_j^3 above):

1. $G_{2n+4j-2}$ is warped to $H_{2n+4j-3}$. This causes costs of at least $2 \cdot d(0,1)$. Then, $H_{2n+4j-2}$ must be warped to $G_{2n+4j-1}$ or p would not have minimal cost. Thus, there are additional costs of at least $8 \cdot d(0,1)$. This is the situation illustrated in Fig. 3a.
2. $G_{2n+4j-2}$ is warped to $H_{2n+4j-2}$. This causes costs of at least $10 \cdot d(0,1)$. This is the situation illustrated in Fig. 3b.
3. $G_{2n+4j-2}$ is warped to $H_{2n+4j-1}$. This case is symmetric to (1) and also causes costs of at least $10 \cdot d(0,1)$. This is the situation illustrated in Fig. 3c.

In summary, the costs contributed by each clause block are at least $42 \cdot d(0,1)$. Hence, to meet the bound of c, all layers outside of clause blocks must not cause any additional cost. For each $i \in [n]$, since G_{2i-1} is warped to H_{2i-1}, this implies that either $\{(\top_i, x_i), (\bot_i, \overline{x_i})\} \subset M$ or $\{(\top_i, \overline{x_i}), (\bot_i, x_i)\} \subset M$.

Furthermore, for each $j \in [m]$, the clause block corresponding to C_j must have cost of exactly $42 \cdot d(0,1)$. If we are in Case (1) as above, then this is only possible if M maps each degree-1 vertex of $G_{2n+4j-2}$ to some degree-1 vertex of $H_{2n+4j-3}$. Thus, $\left(\ell_j^2, \top_{v(j,2)}\right) \in M$. Otherwise, if we are in Case (2), respectively, Case (3), then analogous arguments yield that $\left(\ell_j^1, \top_{v(j,1)}\right) \in M$, respectively, $\left(\ell_j^3, \top_{v(j,3)}\right) \in M$. Hence, in any case there is some $i \in [3]$ for which $\left(\ell_j^i, \top_{v(j,i)}\right) \in M$.

Consequently,

$$
\beta(x_i) := \begin{cases} \texttt{true}, & \text{if } (x_i, \top_i) \in M \\ \texttt{false}, & \text{if } (\overline{x_i}, \top_i) \in M \end{cases}
$$

is a satisfying assignment for ϕ. □

Let us take a closer look at the reduction in the proof of Theorem 1. Note that the corresponding optimal warping path is always close to the diagonal (that is, $|i-j| \leq 1$ holds for every pair (i,j)). Hence, it lies within the so-called Sakoe–Chiba band (Sakoe and Chiba 1978) of width one. Moreover, the maximum degree in each layer is one. Finally, the number of vertices and the number of layers of both temporal graphs and the target cost c are all upper-bounded linearly in the size of the 3 - SAT formula, which allows to conclude a running time lower bound based on the *Exponential Time Hypothesis*[3] Impagliazzo and Paturi (2001) [together with the *Sparsification Lemma* (Impagliazzo et al. 2001)]. These observations are summarized in the following corollary. (Recall that V, W are the vertex sets and T, U are the lifetimes of $\mathcal{G}$, $\mathcal{H}$.)

Corollary 2 DTGW *is* NP *-complete for every metric and vertex degrees as vertex signatures even when the maximum degree of each layer is one and the warping path is*

[3] The Exponential Time Hypothesis asserts that 3-SAT cannot be solved in subexponential time, that is, there is no $2^{o(n)} \cdot \mathrm{poly}(m)$-time algorithm, where n is the number of variables and m is the number of clauses of the input formula.

restricted to the Sakoe–Chiba band of width one. Moreover, even this very restricted variant cannot be solved in $2^{o(|V|+|W|+T+U+c)} \cdot \text{poly}(|\mathcal{G}| + |\mathcal{H}|)$ *time unless the Exponential Time Hypothesis fails.*

Furthermore, if the dtgw-distance is normalized (e.g., divided by the number of vertices), then we obtain NP-hardness for a constant value of c (by the reduction in the proof of Theorem 1).

Corollary 3 DTGW *with normalized vertex-signature-based distance is* NP*-complete for a constant value of* c.

Due to the proven worst-case hardness of DTGW, there is little hope to solve the problem efficiently in general. Nevertheless, in the next section we provide some methods to cope with this intractability.

5 Algorithms

In this section, we first point out polynomial-time solvable special cases. Then, we develop a mathematical programming formulation as well as a heuristic approach to approximate the dtgw-distance in practice.

5.1 Exact polynomial-time algorithms for special cases

Our first algorithmic result is to show that one can determine in polynomial time whether two temporal graphs with the same number of vertices have dtgw-distance zero. This basic case occurs when checking for duplicates within a data set. In contrast, determining whether two (static) graphs have graph edit distance zero is not known to be polynomial-time solvable (as this is equivalent to the famous Graph Isomorphism problem).

Theorem 2 *Let* $\mathcal{G} = (V, E_1, \dots, E_T)$ *and* $\mathcal{H} = (W, F_1, \dots, F_U)$ *be two temporal graphs with* $|V| = |W| = n$. *For all vertex signatures and all metrics, deciding whether* $\text{dtgw}(\mathcal{G}, \mathcal{H}) = 0$ *holds is possible in* $\mathcal{O}(n^2 \cdot (T + U) + n^3)$ *time.*

Proof We will show that for distance zero, an optimal warping path can easily be determined. Polynomial-time solvability then follows from Observation 1.

Let $\mathcal{G} = (V, E_1, \dots, E_T)$ and $\mathcal{H} = (W, F_1, \dots, F_U)$ be two temporal graphs with $V =: \{v_1, \dots, v_n\}$ and $W =: \{w_1, \dots, w_n\}$. For each $i \in [T]$, we define the i^{th} *layer signature* of $\mathcal{G}$ as $f(G_i) := \big(f_{G_i}(v_1), \dots, f_{G_i}(v_n)\big)$ (analogously, $f(H_j) := \big(f_{H_j}(w_1), \dots, f_{H_j}(w_n)\big)$ for $j \in [U]$). Assuming $\text{dtgw}(\mathcal{G}, \mathcal{H}) = 0$, it follows that there exists a vertex mapping $M \subseteq V \times W$ and a warping path $p \in \mathcal{P}_{T,U}$ such that

$$\sum_{(u,v)\in M} d\big(f_{G_i}(u), f_{H_j}(v)\big) = 0$$

holds for every $(i, j) \in p$. Since d is a metric, this implies that $f_{G_i}(u) = f_{H_j}(v)$ holds for every $(u, v) \in M$. That is, $f(H_j)$ is a permutation (determined by M) of $f(G_i)$. Let $1 \le i_1 < i_2 \dots < i_q < T$ be the indices such that

$$f(G_i) \neq f(G_{i+1}) \iff i \in \{i_k; k \in [q]\},$$

and let $1 \le j_1 < j_2 < \dots < j_r < U$ be the indices such that

$$f(H_j) \neq f(H_{j+1}) \iff j \in \{j_k; k \in [r]\}.$$

Clearly, if $f(G_i) \neq f(G_{i'})$ and layer i is warped to layer j and layer i' is warped to layer j', then $f(H_j) \neq f(H_{j'})$ since otherwise the cost will not be zero. By the definition of a warping path, it follows that the layers $1, \dots, i_1$ of $\mathcal{G}$ can only be warped to layers $1, \dots, j_1$ of $\mathcal{H}$ and the layers $i_1 + 1, \dots, i_2$ of $\mathcal{G}$ can only be warped to layers $j_1 + 1, \dots, j_2$ of $\mathcal{H}$ and so on. Note that this is only possible if $q = r$. If this is the case, then we can assume that the warping path p has the following form:

$$\begin{aligned} p = \big\{&(1,1),(1,2),\dots,(1,j_1),(2,j_1),\dots(i_1,j_1),\\ &(i_1+1,j_1+1),\dots,(i_1+1,j_2),\dots(i_2,j_2),\\ &\dots,\\ &(i_q+1,j_q+1),\dots,(i_q+1,U),\dots,(T,U)\big\}. \end{aligned}$$

By Observation 1, we can now check whether there exists a vertex mapping that yields distance zero for the warping path p in $\mathcal{O}(n^2 \cdot (T + U) + n^3)$ time. Computing p can be done in $\mathcal{O}(n(T + U))$ time. □

We remark that if the vertex signatures and the metric satisfy the property that every pair of different vertex signatures has distance at least δ for some constant $\delta > 0$, then DTGW parameterized by the resulting cost c is in XP . For example, this is the case when the vertex signatures contain only integers and d is any ℓ^p-norm (for $p \ge 1$). Then, every pair of different signatures has distance at least $\delta = 1$. The idea of the algorithm is to "guess" the tuples of a warping path which cause nonzero cost (at most c/δ many) and to check whether it is possible to complete the warping path without further costs. The latter can be done in polynomial time using similar arguments as for the case $c = 0$ (Theorem 2).

Corollary 4 DTGW *is in* XP *with respect to* c *if the vertex signatures have a constant minimum pairwise distance* $\delta > 0$.

In contrast, if the dtgw-distance is normalized, then the differences between vertex signatures can be arbitrarily small, in which case DTGW is NP-hard for constant c (Corollary 3).

To overcome this hardness, in the following, we consider special cases based on parameters regarding the warping path length. We assume that the lifetimes of the inputs differ by at most a constant, that is, $T = U + t$ for some $t \geq 0$ (which might often be the case in practice). Note that, by definition, every warping path of order $T \times U$ has length at least T. We define the parameter λ to be the difference between the warping path length and the lower bound T, that is, we consider only order-$(T \times U)$ warping paths of length at most $T + \lambda$. (In practice, overly long warping paths might be considered unnatural.) We prove that DTGW is in XP with respect to the combined parameter (λ, t).

Theorem 3 *For all vertex signatures and all metrics,* DTGW *is solvable in*

$$\mathcal{O}\big((T+\lambda)^{\lambda} \cdot T^{\lambda+t}\big(n^2 \cdot (T+\lambda) + n^3\big)\big)$$

time if $n = \max(|V|, |W|)$, $T = U + t$, *and the warping paths have length at most* $T + \lambda$.

Proof Let $\mathcal{G} = (V, E_1, \dots, E_T)$ and $\mathcal{H} = (W, F_1, \dots, F_T)$ be two temporal graphs, and let $p = \{p_1 = (i_1, j_1), \dots, p_L = (i_L, j_L)\} \in \mathcal{P}_{T,U}$ be a warping path. The warping path p contains $L-1$ *steps* $p_{\ell+1} - p_\ell = (i_{\ell+1} - i_\ell, j_{\ell+1} - j_\ell) \in \{(1,0), (0,1), (1,1)\}$ for $1 \leq \ell < L$. We call a step ℓ *horizontal* if $p_{\ell+1} - p_\ell = (1,0)$, and we call it *vertical* if $p_{\ell+1} - p_\ell = (0,1)$, and otherwise we call it *diagonal*. Let $v \leq U - 1$ denote the number of vertical steps in p. Then, p contains also $v + t$ horizontal and $U - 1 - v$ diagonal steps, that is, $L - 1 = v + v + t + U - v - 1$, which implies that $v = L - t - U$. Clearly, there are $\binom{L-1}{v}$ possible positions for the vertical steps. For each of these possible choices, there are again $\binom{L-1-v}{v+t}$ possible positions for horizontal steps. (The remaining steps are diagonal.) Therefore, the overall number of warping paths of length at most $T + \lambda$ is

$$\sum_{l=0}^{\lambda} \binom{T+l-1}{l} \binom{T-1}{l+t} \in \mathcal{O}\big((T+\lambda)^{\lambda} \cdot T^{\lambda+t}\big).$$

For each of these possible warping paths, we can compute dtgw $(\mathcal{G}, \mathcal{H})$ in

$$\mathcal{O}(\max(|V|, |W|)^2 \cdot (T+\lambda) + \max(|V|, |W|)^3)$$

time by Observation 1. □

Note that Proposition 3 implies polynomial-time solvability of DTGW if t and λ are constants. For unbounded t, however, we conjecture that DTGW is NP-hard even if the warping paths are restricted to have length $\max(T, U)$, which is the minimum possible length (that is, $\lambda = 0$).

5.2 Quadratic programming

We give a formalization of DTGW as a quadratic minimization problem with linear constraints (QP). This is NP-hard to solve in general but can be used to solve small instances exactly with the state-of-the-art QP-solvers such as Gurobi[4].

Let $\mathcal{G} = (V, E_1, \dots, E_T)$ and $\mathcal{H} = (W, F_1, \dots, F_U)$ be two temporal graphs. Denote the vertices in V by $u_1, \dots, u_{|V|}$ and the vertices in W by $v_1, \dots, v_{|W|}$. To model "vertex deletion," we add two artificial vertices $u_{|V|+1}, v_{|W|+1}$.

We use the following variables:

- For every $(i,j) \in [|V|+1] \times [|W|+1]$, we have a *vertex mapping variable* $m_{i,j} \in \{0,1\}$, where $m_{i,j} = 1$ if and only if vertex u_i is mapped to vertex v_j.
- For every $(s,t) \in [T] \times [U]$, we have a *warping variable* $w_{s,t} \in \{0,1\}$, where $w_{s,t} = 1$ if and only if G_s is warped to H_t.

Moreover, for every $(s,t,i,j) \in [T] \times [U] \times [|V|+1] \times [|W|+1]$, let

$$d_{s,t,i,j} := \begin{cases} d\big(f_{G_s}(v_i), f_{H_t}(w_j)\big), & i \in [|V|],\ j \in [|W|] \\ \Delta_{G_s}(v_i), & i \in [|V|],\ j = |W|+1 \\ \Delta_{H_t}(w_j), & i = |V|+1,\ j \in [|W|] \\ 0, & i = |V|+1,\ j = |W|+1 \end{cases}$$

denote the cost of matching vertex i in layer s to vertex j in layer t.

Computing dtgw $(\mathcal{G}, \mathcal{H})$ is the following quadratic[5] minimization problem.

$$\text{minimize} \quad \sum_{s\in[T]} \sum_{t\in[U]} \sum_{i\in[|V|+1]} \sum_{j\in[|W|+1]} d_{s,t,i,j} \cdot w_{s,t} \cdot m_{i,j} \tag{1}$$

subject to (1a)

$$\sum_{j\in[|W|+1]} m_{i,j} = 1 \quad \forall i \in [|V|] \tag{1b}$$

[4] www.gurobi.com

[5] It is also possible to convert our formulation into a linear problem by introducing further variables and constraints for replacing the product $w_{s,t} \cdot m_{i,j}$ in the objective function. However, in our experiments we found that the quadratic formulation can be solved faster.

$$\sum_{i\in[|V|+1]} m_{i,j} = 1 \quad \forall j \in [|W|] \tag{1c}$$

$$w_{1,1} = 1 \tag{1d}$$

$$w_{s,t} \le w_{s+1,t+1} + w_{s,t+1} + w_{s+1,t} \quad \forall (s,t) \in [T-1] \times [U-1] \tag{1e}$$

$$w_{T,t} \le w_{T,t+1} \quad \forall t \in [U-1] \tag{1f}$$

$$w_{s,U} \le w_{s+1,U} \quad \forall s \in [T-1]. \tag{1g}$$

The constraints 1b and 1c ensure that the vertex mapping variables define a correct vertex mapping, that is, every vertex is mapped to exactly one other vertex (or is deleted). Constraints 1d to 1g ensure that the warping variables define a valid warping path. Here, the constraints 1e to 1g imply that if the warping path contains a pair (s, t), then it also contains at least one of the pairs $(s+1, t)$, $(s, t+1)$, or $(s+1, t+1)$. (Since the objective is minimized, any solution will actually select only one of these pairs.)

The number of variables is in $\mathcal{O}(|V| \cdot |W| + T \cdot U)$, and the number of constraints is in $\mathcal{O}(|V| + |W| + T \cdot U)$.

5.3 Heuristic approaches

In this section, we present a heuristic to compute the dtgw-distance, which typically yields good (not necessarily optimal) solutions in practice.

The approach is to simply start with an arbitrary initial vertex mapping (or warping path) and to compute an optimal warping path (vertex mapping) based on Observation 1 in polynomial time. This process is then repeated by alternating between optimal warping path and optimal vertex mapping computation until the solution converges to a local minimum (or some other criterion is reached).

Note that it is a convenient feature of our heuristic to be able to stop the process after any number of iterations to obtain some approximate solution (a so-called *anytime algorithm*). It is further possible to incorporate prior knowledge, for example, by fixing the mapping for some vertices. Note also that convergence is guaranteed since we decrease the objective in each alternation and the search space is finite. We propose several initialization options.

Initial Warping Path. A first idea for initialization is to choose a shortest warping path (that is, of length $\max(T, U)$). Note that for $T \neq U$ several such paths exist. Without further knowledge about the instances, choosing a path within the Sakoe–Chiba band of small width is a reasonable default. This initialization is very simple and only requires $\mathcal{O}(T + U)$ time.

Another idea is to compute a warping path using $D(G_i, H_j)$ as a cost for warping layer i to layer j. This is of course an optimistic estimate since it allows to use a different vertex mapping for each pair of layers. Then, a vertex mapping can be computed by Observation 1. This initialization takes $\mathcal{O}(T \cdot U \cdot n^3)$ time where $n := \max(|V|, |W|)$.

Initial Vertex Mapping. The idea is to compute a vertex mapping by solving an *Assignment Problem* instance for approximate costs. Let $\sigma(u, v)$ be some approximate cost for mapping vertex $u \in V$ to $v \in W$. For example, one could use the following estimations:

$$\sigma^*(u,v) := \sum_{i\in[T]} \sum_{j\in[U]} d\big(f_{G_i}(u), f_{H_j}(v)\big),$$

$$\sigma_{\text{opt}}(u,v) := \sum_{i\in[T]} \min_{j\in[U]} d\big(f_{G_i}(u), f_{H_j}(v)\big).$$

The first option σ^* estimates the cost of mapping u to v over all possible warpings between any two layers. (This is usually more than any warping path will incur.) The definition of σ_{opt} only considers for each layer of the first temporal graph the minimal cost over all layers of the other temporal graph. (This estimate might be too low.) Both of these options require $\mathcal{O}(T \cdot U \cdot |V| \cdot |W|)$ time.

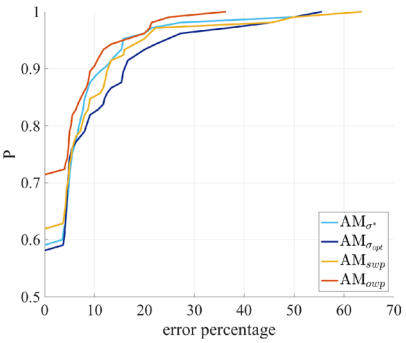
Based on the estimated costs, one computes a vertex mapping by solving an Assignment Problem instance and then computes an optimal warping path for this vertex mapping based on Observation 1.

The running time of one iteration (that is, computing a vertex mapping and an optimal warping path) is $\mathcal{O}\big((T+U) \cdot n^2 + n^3 + T \cdot U \cdot n\big)$. While the number of iterations might depend on the choice of initialization, in our experiments the heuristic always converged after very few iterations. Regarding the solution quality, while it is possible to construct adversarial examples where the heuristic performs poorly, our experiments in Sect. 6.2 indicate that it performs well in practice.

6 Experiments

We conducted several experiments[6] to demonstrate the merit of our dtgw-distance in applications and to evaluate the performance of the alternating minimization heuristic (AM) we described in Sect. 5.3. For computations, we used a 4.0 GHz i7-6700K processor (single-threaded).

[6] Source code available at www.akt.tu-berlin.de/menue/software.



heuristic	avg	std	P_0	max
AM_{σ^*}	4.5	9.1	0.59	63.6
$\mathrm{AM}_{\sigma_{\mathrm{opt}}}$	5.5	10.5	0.58	55.6
$\mathrm{AM}_{\mathrm{swp}}$	4.8	10.0	0.61	63.6
$\mathrm{AM}_{\mathrm{owp}}$	3.0	6.2	0.71	36.4

Fig. 4 The plot (left) shows the estimated cumulative distribution functions of the four AM variants. The table (right) presents the average error percentage (avg), the standard deviation (std), the fraction of optimally solved instances (P_0), and the maximum error percentage (max) of every AM variant

6.1 Data sets

We used three data sets from the SocioPatterns project (Génois and Barrat 2018). Each of these consists of two temporal social networks, both recorded simultaneously with the same individuals as vertices. The first network is a face-to-face contact network, whereas the second one is a co-presence network where edges represent spatial proximity. All six networks have a temporal resolution of 20 seconds. For our experiments, only the first day of each network was used and vertices without any edges were discarded. The three different data sets were recorded at a primary school ("LyonSchool," 237 vertices, 1700 layers), a scientific conference ("SFHH," 403 vertices, 2300 layers), and a workplace ("InVS15," 180 vertices, 2100 layers).

6.2 Comparison of heuristic and exact solutions

We compared the solutions of our AM heuristic under different initialization schemes against the optimal solutions obtained from the QP formulation.

Due to long running times of the QP-solver, we were restricted to very small temporal networks. We randomly selected 10 children from class 1A of the primary school face-to-face network and extracted 225 consecutive layers (during a high contact period) which we split into 15 temporal subnetworks with 15 consecutive layers each. We used vertex degrees as signatures (with absolute value metric) and computed all pairwise dtgw-distances between these 15 networks with the following algorithms:

- QP: exact QP-solver (Gurobi 8.0.1),
- AM $_{\sigma^*}$: AM with σ^* initialization,
- AM $_{\sigma_{\mathrm{opt}}}$: AM with σ_{opt} initialization,
- AM $_{\mathrm{swp}}$: AM with shortest warping path initialization,
- AM $_{\mathrm{owp}}$: AM with optimistic warping path initialization.

We implemented the AM heuristic in Python, using a C++ implementation[7] of the Jonker–Volgenant algorithm (Jonker and Volgenant 1987) to solve the *Assignment Problem*.

Figure 4 shows for each initialization variant the estimated cumulative distribution function (ecdf) of the error percentage $\varepsilon = 100 \cdot (d_{\mathrm{AM}} - d_{\mathrm{QP}})/d_{\mathrm{QP}}$, where d_{AM} is the approximate dtgw-distance obtained by an AM heuristic and d_{QP} is the exact dtgw-distance obtained by the QP-solver. A point (ε, P) on an ecdf-curve of an AM heuristic means that the error percentage of AM is at most ε with estimated probability P.

All AM variants found the correct solution for a majority of samples ($P_0 > 0.5$). The average error percentages are rather small and vary between 3.0 by AM $_{\mathrm{owp}}$ and 5.5 by AM $_{\sigma_{\mathrm{opt}}}$. The AM $_{\mathrm{owp}}$ heuristic performed the best, having $P_0 \approx 0.71$ and maximum error percentage max ≈ 36.4. These findings indicate that for small instances the approximations of the four heuristics are close to the optimal solution on average but may fail considerably in some cases with up to a maximum error of 63.6%. We remark that based on our experimental experience the relative error becomes smaller on larger instances.

Regarding running times, the AM heuristic took less than 0.01 seconds per instance, usually converging after at most three iterations (independently of the chosen initialization). In comparison, the QP was slower by a factor of more than 10 000, requiring 8 minutes on average (median 2 minutes) with some instances approaching 2 hours.

6.3 Sensitivity of DTGW to noise

The goal of this experiment was to assess how sensitive the dtgw-distance is to noise, that is, how well can original data be reconstructed from noisy data. We compared our dtgw-distance approach to the following two baseline methods.

[7] This code is available as a Python module at github.com/src-d/lapjv.

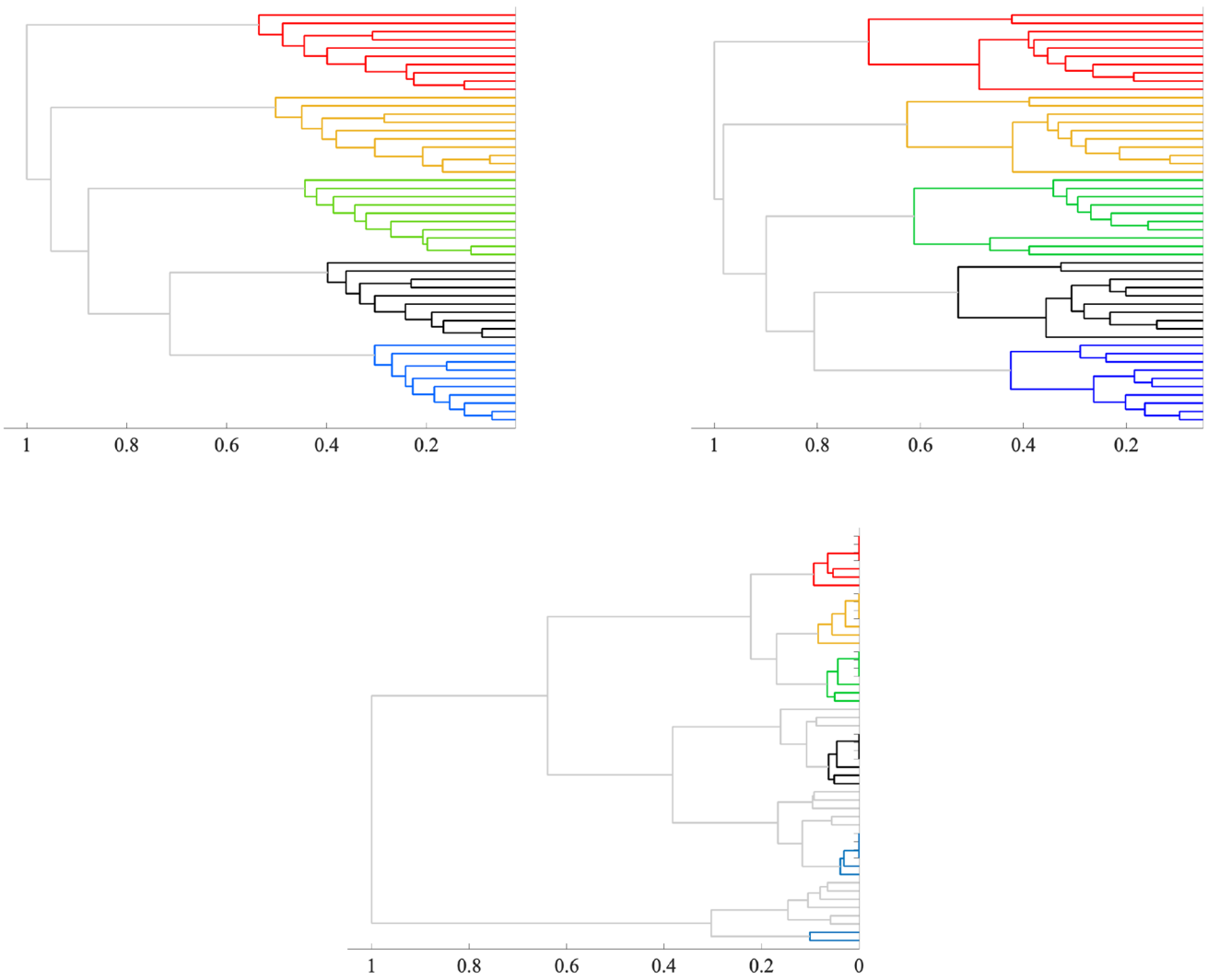


Fig. 5 Sensitivity of dtgw to noise. Shown are the dendrograms obtained by agglomerative clustering using complete linkage. Different colors represent different grades; the light gray edges connect elements of different grades. The top left dendrogram shows the result obtained by the dtgw-distance. The top right dendrogram shows the result obtained by the non-consistent benchmark. The bottom dendrogram shows the result obtained by using the non-temporal benchmark

- Non-consistent: Instead of using one consistent vertex mapping for all layers, one can allow a different mapping for each pair of layers. Note that the resulting distance can be computed in $\mathcal{O}(T \cdot U \cdot n^3)$ time, thus being faster than an exact computation of the dtgw-distance but much slower than a single iteration of the AM heuristic.
- Non-temporal: A naive approach is to ignore the time information and solely compute an optimal vertex mapping between the underlying graphs. This requires $\mathcal{O}(n^3)$ time plus the (usually linear) time to build the underlying graphs.

We used the primary school face-to-face network from which we extracted five reference temporal networks representing the contacts between children of the same grade, each containing 45–50 vertices and 3100 layers.

For each of the five reference networks, we generated nine noisy copies as follows:

1. For every $i \in [T]$, E_i is deleted with probability $p \in \{0.1, 0.2, 0.3\}$, and if not, then each edge $e \in E_i$ is deleted with probability p.
2. For every $i \in [T]$, each edge $e \in E_i$ was rewired with probability $p \in \{0.1, 0.2, 0.3\}$.
3. Each edge of the underlying graph was rewired with probability $p \in \{0.1, 0.2, 0.3\}$.

Rewiring an edge $e = \{u, v\} \in E_i$ of a temporal graph is defined as randomly picking a tuple

Table 1 Percentages (rounded) of vertices that were re-identified by the tested methods

	Data set	dtgw	Fixed dtgw	DynaMAGNA++	HTNE
School	Original	2%	1%	1%	1%
	Shifted	1%	–	0%	1%
	Randomized	1%	–	1%	0%
	Average running time	95 s	65 s	15,070 s	250 s
Conference	Original	86%	90%	80%	0%
	Shifted	86%	–	27%	0%
	Randomized	65%	–	30%	1%
	Average running time	200 s	125 s	20,320 s	90 s
Workplace	Original	38%	43%	51%	1%
	Shifted	38%	–	19%	0%
	Randomized	10%	–	8%	1%
	Average running time	45 s	20 s	1600 s	50 s

Also, average running times (in seconds) over the three versions of each data set are given (fixed dtgw is not applicable for shifted and randomized data sets)

$(e' = \{u', v'\}, t) \in \bigcup_{s=1}^{T}(E_s \times \{s\})$ and then replacing e in E_i by $\{u, v'\}$ and e' in E_t by $\{u', v\}$.[8] Rewiring of underlying edges is done analogously (see Holme and Saramäki 2012 for details).

We used the AM heuristic to approximate the pairwise dtgw-distances (using degrees as vertex signatures) between all reference and noisy temporal networks. In all of these instances, shortest warping path initialization (which is fastest) was used since preliminary tests showed that the other initializations produce very similar results.

Figure 5 shows the dendrogram obtained by hierarchical clustering using complete linkage of the approximated pairwise dtgw-distances. Both the dtgw-distance and the non-consistent baseline were able to partition the instances into five clusters, each of which consists of a reference network and its nine noisy copies. Hence, they successfully recovered the original reference networks from noise. However, the clusters produced by the dtgw-distance are more compact than the ones of the non-consistent baseline. In contrast, the non-temporal baseline was not able to separate the graphs of different grades.

In all instances, the heuristic converged within at most six iterations, taking less than 15 seconds. In comparison, the non-consistent baseline required 4 minutes on average for each instance, while the non-temporal baseline was the fastest (below 1 second).

6.4 De-anonymization

Besides measuring a distance between temporal graphs, the dtgw-distance additionally provides a mapping between the vertex sets which implicitly allows to identify vertices. This allows the de-anonymization (Narayanan and Shmatikov 2009) of temporal social networks. Since the data sets used contain the original mapping between the vertex sets, we can employ this as an easy benchmark for the accuracy of the dtgw-distance.

We used the AM heuristic (with shortest warping path initialization) to compute the dtgw-distance (with degrees as vertex signatures[9]) on the three data sets mentioned in Sect. 6.1. We counted how many vertices were correctly re-identified (that is, mapped to their copies) in the resulting vertex mapping. We compared our results to the following alternative algorithms found in the literature:

- DynaMAGNA++ (Vijayan et al. 2017): A search-based evolutionary algorithm computing a vertex mapping that maximizes edge conservation and node conservation over time.
- Temporal network embedding (Zuo et al. 2018): Hawkes process-based temporal network embedding (HTNE) computes a low-dimensional embedding of the vertices of a temporal network. From this, we computed a vertex mapping minimizing the Euclidean distances between the vertex feature vectors.
- Fixed dtgw: Note that our dtgw-distance allows to fix the warping path beforehand (Observation 1 ii)). Since in our

[8] Since edges are undirected, it is understood that the choice of which vertex to call u, respectively, v is to be made randomly. (The same holds for u' and v'.)

[9] We also tested other signatures such as size of the connected component or betweenness centrality. However, the performance was (slightly) worse.

case each pair of temporal graphs was recorded using synchronized clocks, it is natural to use a fixed warping path that aligns layer i of the first graph with layer i of the second graph.

To simulate a situation in which the temporal graphs represent processes which do not run synchronously in time, we created two modified versions of each of the data sets. In the first one, called "shifted," all events of the first graph were delayed by three minutes. In the second version, called "randomized," each layer of each of the graphs was randomly and independently replaced by X layers where $X \in \{1, 2, \dots\}$ is a random variable with $\mathbb{P}(X \geq x) = x^{-3}$. Since we pretend that the nature of these modifications is unknown to the tested algorithms, dtgw with fixed warping path is not applicable to these variants.

For DynaMAGNA++, we used a population of size 15 000 and a maximum of 10 000 generations. With HTNE, we computed 128-dimensional vertex embeddings using a batch of size 10 000, a learning rate of 0.1, a history length of 2, and 5 negative samples. Unlike dtgw, both methods utilized all four processor cores.

The results and running times are listed in Table 1. Most notably, the re-identification rate of HTNE was poor on all data sets, suggesting that these embeddings are ill-suited for comparing vertices taken from different networks. Furthermore, all methods failed to re-identify any significant number of vertices on the primary school data set. This might be explained by the fact that the co-presence network is very different from the face-to-face contacts due to a low spatial resolution (as was also noted by Génois and Barrat (2018)).

The overall performance was much better on the other two data sets, especially on the conference data where up to 90% of participants could be re-identified, whereas on the workplace data set the best result was 51%. Unsurprisingly, fixing the correct layer alignment on the unmodified graphs sped up the dtgw computation significantly while also yielding slightly better results. On these instances, dtgw performed comparably to DynaMAGNA++, being better in one case and worse in the other, although requiring much less computational effort. In contrast, on the shifted and randomized data sets dtgw always achieved the best results. (Notably, the re-identification performance using dtgw did not decrease on shifted data.)

In all cases, the AM heuristic converged after at most six iterations and DynaMAGNA++ converged within 2 000 generations.

7 Conclusion

We introduced a new proximity measure for comparing temporal graphs by transferring dynamic time warping from time series to temporal graphs. This yields a challenging computational problem for which we proposed exact algorithms and a heuristic approach to solve it. While exact solutions can only be computed for very small instances, we empirically showed that our heuristic runs fast in practice and yields good approximations of optimal solutions. In our experiments, it was also capable of de-anonymizing social networks.

Our work opens several directions for future research. We believe that the dtgw-distance is a promising tool, for example, in biology and chemistry. Processes like epidemic disease spreading or chemical reactions can naturally be viewed as temporal graphs where the vertices represent individuals or (macro) molecules. [Unfortunately, we could not test this, as there is still a lack of openly available temporal molecular data (Vijayan et al. 2017).] Since the exact timescales of these processes often vary, the ability of dynamic time warping to compensate for such differences would be especially helpful in this context. One might also use the dtgw-distance to understand the learning process of neural networks. The training phases of neural networks yield temporal networks which can be analyzed to gain insight into how different conditions influence the learning process. Another potential application is analyzing team sports data via temporal graphs to reveal similar strategies or roles of individual players. Depending on the application domain, there is a wide range of possibilities to test the performance when using different vertex signatures or even other graph distances.

Besides experimenting with various application domains and further definition variants, already the proven computational worst-case hardness of DTGW may trigger further algorithmic research. A concrete open question is whether DTGW is in XP (or even fixed-parameter tractable) with respect to λ, when the warping path length is restricted to be at most $\max(T, U) + \lambda$ where T, U are the respective lifetimes. It is also interesting to further study the influence of graph-specific parameters in the spirit of a multivariate complexity analysis (Niedermeier 2010; Fellows et al. 2013).

Acknowledgement Open Access funding provided by Projekt DEAL.

Funding BJ: Supported by the DFG project JA 2109/4-2, MR: Supported by the DFG project NI 369/17-1.

References

Abboud A, Backurs A, Williams VV (2015) Tight hardness results for LCS and other sequence similarity measures. In: 2015 IEEE 56th annual symposium on foundations of computer science (FOCS '15), pp 59–78. https://doi.org/10.1109/FOCS.2015.14

Ahmed R, Karypis G (2015) Algorithms for mining the coevolving relational motifs in dynamic networks. ACM Trans Knowl Discov Data 10(1):4:1–4:31. https://doi.org/10.1145/2733380

Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: theory, algorithms, and applications. Prentice-Hall, Upper Saddle River

Baste J, Bui-Xuan BM, Roux A (2020) Temporal matching. Theor Comput Sci 806:184–196. https://doi.org/10.1016/j.tcs.2019.03.026

Bento J, Ioannidis S (2018) A family of tractable graph distances. In: Proceedings of the 2018 SIAM international conference on data mining (SDM '18). SIAM, pp 333–341. https://doi.org/10.1137/1.9781611975321.38

Casteigts A, Himmel A, Molter H, Zschoche P (2019) The computational complexity of finding temporal paths under waiting time constraints. In: CoRR. arXiv:abs/1909.06437

Cygan M, Fomin FV, Kowalik L, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) Parameterized algorithms. Springer, New York. https://doi.org/10.1007/978-3-319-21275-3

Dakiche N, Tayeb FBS, Slimani Y, Benatchba K (2019) Tracking community evolution in social networks: a survey. Inf Process Manag 56(3):1084–1102. https://doi.org/10.1016/j.ipm.2018.03.005

Downey R, Fellows MR (2013) Fundamentals of parameterized complexity. Springer, New York. https://doi.org/10.1007/978-1-4471-5559-1

Elhesha R, Sarkar A, Cinaglia P, Boucher C, Kahveci T (2019) Coevolving patterns in temporal networks of varying evolution. In: Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics (BCB '19). ACM, pp 494–503. https://doi.org/10.1145/3307339.3342152

Fellows MR, Jansen BMP, Rosamond F (2013) Towards fully multivariate algorithmics: parameter ecology and the deconstruction of computational complexity. Eur J Combin 34(3):541–566. https://doi.org/10.1016/j.ejc.2012.04.008

Fluschnik T, Niedermeier R, Schubert C, Zschoche P (2020) Multistage *s*-*t* path: confronting similarity with dissimilarity. In: CoRR. arXiv:abs/2002.07569

Froese V, Jain B, Niedermeier R, Renken M (2019) Comparing temporal graphs using dynamic time warping. In: Proceedings of the 8th international conference on complex networks and their applications, SCI, vol 882. Springer, pp 469–480. https://doi.org/10.1007/978-3-030-36683-4_38

Froese V, Jain BJ, Rymar M (2020) Fast exact dynamic time warping on run-length encoded time series. In: CoRR. arXiv:abs/1903.03003

Fröhlich H, Wegner JK, Sieker F, Zell A (2005) Optimal assignment kernels for attributed molecular graphs. In: Proceedings of the 22nd international conference on machine learning (ICML '05). ACM, pp 225–232. https://doi.org/10.1145/1102351.1102380

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman and Company, New York

Gayraud NT, Pitoura E, Tsaparas P (2015) Diffusion maximization in evolving social networks. In: Proceedings of the 2015 ACM conference on online social networks (COSN '15). ACM, pp 125–135. https://doi.org/10.1145/2817946

Génois M, Barrat A (2018) Can co-location be used as a proxy for face-to-face contacts? EPJ Data Sci 7(1):11. https://doi.org/10.1140/epjds/s13688-018-0140-1

Gold O, Sharir M (2018) Dynamic time warping and geometric edit distance: breaking the quadratic barrier. ACM Trans Algorithms 14(4):50:1–50:17. https://doi.org/10.1145/3230734

Heeger K, Himmel A, Kammer F, Niedermeier R, Renken M, Sajenko A (2019) Multistage problems on a global budget. In: CoRR. arXiv:abs/1912.04392

Holme P, Saramäki J (2012) Temporal networks. Phys Rep 519(3):97–125. https://doi.org/10.1016/j.physrep.2012.03.001

Holme P, Saramäki J (2013) Temporal networks. Springer, New York. https://doi.org/10.1007/978-3-642-36461-7

Holme P, Saramäki J (2019) Temporal network theory. Springer, New York. https://doi.org/10.1007/978-3-030-23495-9

Impagliazzo R, Paturi R (2001) On the complexity of *k*-SAT. J Comput Syst Sci 62(2):367–375. https://doi.org/10.1006/jcss.2000.1727

Impagliazzo R, Paturi R, Zane F (2001) Which problems have strongly exponential complexity? J Comput Syst Sci 63(4):512–530. https://doi.org/10.1006/jcss.2001.1774

Jain BJ (2016) On the geometry of graph spaces. Discrete Appl Math 214:126–144. https://doi.org/10.1016/j.dam.2016.06.027

Jonker R, Volgenant A (1987) A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing 38(4):325–340. https://doi.org/10.1007/BF02278710

Jouili S, Tabbone S (2009) Graph matching based on node signatures. In: Proceedings of the 7th IAPR-TC-15 international workshop on graph-based representations in pattern recognition. LNCS, vol 5534. Springer, pp 154–163. https://doi.org/10.1007/978-3-642-02124-4_16

Kostakos V (2009) Temporal graphs. Physica A 388(6):1007–1023. https://doi.org/10.1016/j.physa.2008.11.021

Kriege NM, Giscard PL, Wilson R (2016) On valid optimal assignment kernels and applications to graph classification. In: Advances in neural information processing systems 29 (NIPS '16). Curran Associates, Inc., pp 1623–1631. http://papers.nips.cc/paper/6166-on-valid-optimal-assignment-kernels-and-applications-to-graph-classification

Kriege NM, Johansson FD, Morris C (2020) A survey on graph kernels. Appl Netw Sci 5(1):6. https://doi.org/10.1007/s41109-019-0195-3

Kuszmaul W (2019) Dynamic time warping in strongly subquadratic time: algorithms for the low-distance regime and approximate evaluation. In: Proceedings of the 46th international colloquium on automata, languages, and programming (ICALP '19), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, LIPIcs, vol 132, pp 80:1–80:15. https://doi.org/10.4230/LIPIcs.ICALP.2019.80

Li A, Cornelius SP, Liu YY, Wang L, Barabási AL (2017) The fundamental advantages of temporal networks. Science 358(6366):1042–1046. https://doi.org/10.1126/science.aai7488

Mertzios GB, Molter H, Niedermeier R, Zamaraev V, Zschoche P (2020) Computing maximum matchings in temporal graphs. In: Proceedings of the 37th international symposium on theoretical aspects of computer science (STACS '20), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, LIPIcs, pp 27:1–27:14. https://doi.org/10.4230/LIPIcs.STACS.2020.27

Narayanan A, Shmatikov V (2009) De-anonymizing social networks. In: Proceedings of the 30th IEEE symposium on security and privacy, pp 173–187. https://doi.org/10.1109/SP.2009.22

Nguyen GH, Lee JB, Rossi RA, Ahmed NK, Koh E, Kim S (2018) Continuous-time dynamic network embeddings. In: Companion proceedings of the web conference 2018 (WWW '18), International World Wide Web Conferences Steering Committee, pp 969–976. https://doi.org/10.1145/3184558.3191526

Niedermeier R (2010) Reflections on multivariate algorithmics and problem parameterization. In: Proceedings of the 27th international symposium on theoretical aspects of computer science (STACS '10), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, LIPIcs, vol 5, pp 17–32. https://doi.org/10.4230/LIPIcs.STACS.2010.2495

Rakthanmanon T, Campana B, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2012) Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining (KDD '12). ACM, pp 262–270. https://doi.org/10.1145/2339530.2339576

Riesen K (2015) Structural pattern recognition with graph edit distance. Springer, New York. https://doi.org/10.1007/978-3-319-27252-8

Rozenshtein P, Gionis A (2019) Mining temporal networks. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (KDD '19). ACM, pp 3225–3226. https://doi.org/10.1145/3292500.3332295

Rozenshtein P, Gionis A, Prakash BA, Vreeken J (2016) Reconstructing an epidemic over time. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (KDD '16).ACM, pp 1835–1844. https://doi.org/10.1145/2939672.2939865

Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 26(1):43–49. https://doi.org/10.1109/TASSP.1978.1163055

Vijayan V, Critchlow D, Milenković T (2017) Alignment of dynamic networks. Bioinformatics 33(14):i180–i189. https://doi.org/10.1093/bioinformatics/btx246

Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P, Keogh E (2013) Experimental comparison of representation methods and distance measures for time series data. Data Min Knowl Disc 26(2):275–309. https://doi.org/10.1007/s10618-012-0250-5

Zschoche P, Fluschnik T, Molter H, Niedermeier R (2020) The complexity of finding small separators in temporal graphs. J Comput Syst Sci 107:72–92. https://doi.org/10.1016/j.jcss.2019.07.006

Zuo Y, Liu G, Lin H, Guo J, Hu X, Wu J (2018) Embedding temporal network via neighborhood formation. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining (KDD '18). ACM, pp 2857–2866. https://doi.org/10.1145/3219819.3220054