The Complexity of Transitively Orienting Temporal Graphs

George B. Mertzios[a,1], Hendrik Molter[b,2], Malte Renken[c,3], Paul G. Spirakis[d,4], Philipp Zschoche[c]

[a] *Department of Computer Science, Durham University, UK*
[b] *Department of Computer Science, Ben-Gurion University of the Negev, Israel*
[c] *Algorithmics and Computational Complexity, Technische Universit"at Berlin, Germany*
[d] *Department of Computer Science, University of Liverpool, UK*

Abstract

In a *temporal network* with discrete time-labels on its edges, entities and information can only "flow" along sequences of edges whose time-labels are non-decreasing (resp. increasing), i.e. along temporal (resp. strict temporal) paths. Nevertheless, in the model for temporal networks of [Kempe, Kleinberg, Kumar, JCSS, 2002], the individual time-labeled edges remain undirected: an edge $e = \{u, v\}$ with time-label t specifies that "u communicates with v at time t". This is a symmetric relation between u and v, and it can be interpreted that the information can flow in either direction.

In this paper we make a first attempt to understand how the direction of information flow on one edge can impact the direction of information flow on other edges. More specifically, naturally extending the classical notion of a transitive orientation in static graphs, we introduce the fundamental notion of a *temporal transitive orientation* and we systematically investigate its algorithmic behavior in various situations. An orientation of a temporal graph is called *temporally transitive* if, whenever u has a directed edge towards v with time-label t_1 and v has a directed edge towards w with time-label $t_2 \geq t_1$, then u also has a directed edge towards w with some time-label $t_3 \geq t_2$. If we just demand that this implication holds whenever $t_2 > t_1$, the orientation is called *strictly* temporally transitive, as it is based on the fact that there is a strict directed

⋆An extended abstract of this work was published in the proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021) [38].

Email addresses: george.mertzios@durham.ac.uk (George B. Mertzios), molterh@post.bgu.ac.il (Hendrik Molter), m.renken@tu-berlin.de (Malte Renken), p.spirakis@liverpool.ac.uk (Paul G. Spirakis), zschoche@tu-berlin.de (Philipp Zschoche)

[1]Supported by the EPSRC grant EP/P020372/1.

[2]Supported by the DFG, project MATE (NI 369/17), by the ISF, grants No. 1456/18 and No. 1070/20, and the ERC, grant number 949707.

[3]Supported by the DFG, project MATE (NI 369/17).

[4]Supported by the NeST initiative of the School of EEE and CS at the University of Liverpool and by the EPSRC grant EP/P02002X/1.

temporal path from u to w.

Our main result is a conceptually simple, yet technically quite involved, polynomial-time algorithm for recognizing whether a given temporal graph $\mathcal{G}$ is transitively orientable. In wide contrast we prove that, surprisingly, it is NP-hard to recognize whether $\mathcal{G}$ is strictly transitively orientable. Additionally we introduce and investigate further related problems to temporal transitivity, notably among them the *temporal transitive completion* problem, for which we prove both algorithmic and hardness results.

Keywords: Temporal graph, transitive orientation, transitive closure, polynomial-time algorithm, NP-hardness, satisfiability.

1. Introduction

A *temporal* (or *dynamic*) network is, roughly speaking, a network whose underlying topology changes over time. This notion concerns a great variety of both modern and traditional networks; information and communication networks, social networks, and several physical systems are only few examples of networks which change over time [40, 43, 28]. Due to its vast applicability in many areas, the notion of temporal graphs has been studied from different perspectives under several different names such as *time-varying*, *evolving*, *dynamic*, and *graphs over time* (see [16, 14, 15] and the references therein). In this paper we adopt a simple and natural model for temporal networks which is given with discrete time-labels on the edges of a graph, while the vertex set remains unchanged. This formalism originates in the foundational work of Kempe et al. [29].

Definition 1 (Temporal Graph [29])**.** A *temporal graph* is a pair $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$ is an underlying (static) graph and $\lambda : E \to 2^{\mathbb{N}}$ is a *time-labeling* function which assigns to every edge of G a discrete-time label. Whenever $|\lambda(e)| = 1$ for every edge $e \in E$, then $\mathcal{G}$ is a *single-labeled temporal graph*.

In this paper we only consider *single-labeled temporal graphs*, while, for simplicity of presentation, we refer to them as just *temporal graphs*. Mainly motivated by the fact that, due to causality, entities and information in temporal graphs can only "flow" along sequences of edges whose time-labels are non-decreasing (resp. increasing), Kempe et al. introduced the notion of a *(strict) temporal path*, or *(strict) time-respecting path*, in a temporal graph (G, λ) as a path in G with edges $e_1, e_2, \ldots, e_k$ such that $\lambda(e_1) \leq \ldots \leq \lambda(e_k)$ (resp. $\lambda(e_1) < \ldots < \lambda(e_k)$). This notion of a temporal path naturally resembles the notion of a *directed* path in the classical static graphs, where the direction is from smaller to larger time-labels along the path. Nevertheless, in temporal paths the individual time-labeled edges remain undirected: an edge $e = \{u, v\}$ with time-label $\lambda(e) = t$ can be abstractly interpreted as "u communicates with v at time t". Here the relation "communicates" is symmetric between u and v, i.e. it can be interpreted that the information can flow in either direction.

In this paper we make a first attempt to understand how the direction of information flow on one edge can impact the direction of information flow on other edges. More specifically, naturally extending the classical notion of a transitive orientation in static graphs [25], we introduce the fundamental notion of a *temporal transitive orientation* and we thoroughly investigate its algorithmic behavior in various situations. Imagine that v receives information from u at time t_1, while w receives information from v at time $t_2 \geq t_1$. Then w *indirectly* receives information from u through the intermediate vertex v. Now, if the temporal graph correctly records the transitive closure of information passing, the directed edge from u to w must exist and must have a time label $t_3 \geq t_2$. In such a *transitively oriented* temporal graph, whenever an edge is oriented from a vertex u to a vertex w with time-label t, we have that *every* temporal path from u to w arrives no later than t, and that there is no temporal path from w to u. Different notions of temporal transitivity have also been used for automated temporal data mining [42] in medical applications [41], text processing [47]. These notions of temporal transitivity are defined on the so-called "Allen's temporal relations" [5], which are relations defined on time intervals. These transitivity notions are conceptually very different from our setting where we focus on a temporal ordering of events that happen on the edges of a temporal graph. Furthermore, in behavioral ecology, researchers have used a notion of orderly (transitive) triads A-B-C to quantify dominance among species. In particular, animal groups usually form dominance hierarchies in which dominance relations are transitive and can also change with time [34].

One natural motivation for our temporal transitivity notion may come from applications where confirmation and verification of information is vital, where vertices may represent entities such as investigative journalists or police detectives who gather sensitive information. Suppose that v queried some important information from u (the information source) at time t_1, and afterwards, at time $t_2 \geq t_1$, w queried the important information from v (the intermediary). Then, in order to ensure the validity of the information received, w might want to verify it by *subsequently* querying the information directly from u at some time $t_3 \geq t_2$. Note that w might first receive the important information from u through various other intermediaries, and using several channels of different lengths. Then, to maximize confidence about the information, w should query u for verification only after receiving the information from the latest of these indirect channels.

It is worth noting here that the model of temporal graphs given in Definition 1 has been also used in its extended form, in which the temporal graph may contain multiple time-labels per edge [36]. This extended temporal graph model has been used to investigate temporal paths [50, 10, 12, 17, 36, 3] and other temporal path-related notions such as temporal analogues of distance and diameter [1], reachability [2] and exploration [3, 1, 22, 21], separation [23, 51, 29], and path-based centrality measures [30, 13], as well as recently non-path problems too such as temporal variations of coloring [39], vertex cover [4], matching [37], cluster editing [19], and maximal cliques [49, 27, 9]. However, in order to better investigate and illustrate the inherent combinatorial structure of temporal transitivity orientations, in this paper we mostly follow the original definition of temporal graphs given by Kempe et al. [29] with one time-label per edge [18, 20, 8]. Throughout the paper, whenever we assume multiple time-labels per edge we will state it explicitly; in all other cases we consider a single label per edge.

In static graphs, the transitive orientation problem has received extensive attention which resulted in numerous efficient algorithms. A graph is called *transitively orientable* (or a *comparability* graph) if it is possible to orient its edges such that, whenever we orient u towards v and v towards w, then the edge between u and w exists and is oriented towards w. The first polynomial-time algorithms for recognizing whether a given (static) graph G on n vertices and m edges is comparability (i.e. transitively orientable) were based on the notion of *forcing* an orientation and had running time $O(n^3)$ (see Golumbic [25] and the references therein). Faster algorithms for computing a transitive orientation of a given comparability graph have been later developed, having running times $O(n^2)$ [45] and $O(n + m \log n)$ [31], while the currently fastest algorithms run in linear $O(n + m)$ time and are based on efficiently computing a modular decomposition of G [33, 32]; see also Spinrad [46]. It is fascinating that, although all the latter algorithms compute a valid transitive orientation if G is a comparability graph, they fail to recognize whether the input graph is a comparability graph; instead they produce an orientation which is non-transitive if G is not a comparability graph. The fastest known algorithm for determining whether a given orientation is transitive requires matrix multiplication, currently achieved in $O(n^{2.37286})$ time [6].

Our contribution. In this paper we introduce the notion of *temporal transitive orientation* and we thoroughly investigate its algorithmic behavior in various situations. An orientation of a temporal graph $\mathcal{G} = (G, \lambda)$ is called *temporally transitive* if, whenever u has a directed edge towards v with time-label t_1 and v has a directed edge towards w with time-label $t_2 \geq t_1$, then u also has a directed edge towards w with some time-label $t_3 \geq t_2$. If we just demand that this implication holds whenever $t_2 > t_1$, the orientation is called *strictly* temporally transitive, as it is based on the fact that there is a strict directed temporal path from u to w. Similarly, if we demand that the transitive directed edge from u to w has time-label $t_3 > t_2$, the orientation is called *strongly* (resp. *strongly strictly)* temporally transitive.

Although these four natural variations of a temporally transitive orientation seem superficially similar to each other, it turns out that their computational complexity (and their underlying combinatorial structure) varies massively. Indeed we obtain a surprising result in Section 3: deciding whether a temporal graph $\mathcal{G}$ admits a *temporally transitive* orientation is solvable in polynomial time (Section 3.2), while it is NP-hard to decide whether it admits a *strictly temporally transitive* orientation (Section 3.1). On the other hand, it turns out that, deciding whether $\mathcal{G}$ admits a *strongly* or a *strongly strictly* temporal transitive orientation is (easily) solvable in polynomial time as they can both be reduced to 2SAT satisfiability.

Our main result is that, given a temporal graph $\mathcal{G} = (G, \lambda)$, we can decide in polynomial time whether $\mathcal{G}$ is transitively orientable, and at the same time we can output a temporal transitive orientation if it exists. Although the analysis and correctness proof of our algorithm is technically quite involved, our algorithm is simple and easy to implement, as it is based on the notion of *forcing* an orientation.[5] Our algorithm extends and generalizes the classical polynomial-time algorithm for computing a transitive orientation in static graphs described by Golumbic [25]. The main technical difficulty in extending the algorithm from the static to the temporal setting is that, in temporal graphs we cannot simply use orientation forcings to eliminate the condition that a *triangle* is not allowed to be cyclically oriented. To resolve this issue, we first express the recognition problem of temporally transitively orientable graphs as a Boolean satisfiability problem of a *mixed* Boolean formula $\phi_{3\text{NAE}} \wedge \phi_{2\text{SAT}}$. Here $\phi_{3\text{NAE}}$ is a 3NAE formula, i.e., the disjunction of clauses with three literals each, where every clause $\texttt{NAE}(\ell_1, \ell_2, \ell_3)$ is satisfied if and only if at least one of the literals $\{\ell_1, \ell_2, \ell_3\}$ is equal to 1 and at least one of them is equal to 0. Note that every clause $\texttt{NAE}(\ell_1, \ell_2, \ell_3)$ corresponds to the condition that a specific triangle in the temporal graph cannot be cyclically oriented. Furthermore $\phi_{2\text{SAT}}$ is a 2SAT formula, i.e., the disjunction of 2CNF clauses with two literals each, where every clause $(\ell_1 \vee \ell_2)$ is satisfied if and only if at least one of the literals $\{\ell_1, \ell_2\}$ is equal to 1. However, although deciding whether $\phi_{2\text{SAT}}$ is satisfiable can be done in linear time with respect to the size of the formula [7], the problem Not-All-Equal-3-SAT is NP-complete [44].

In the second part of our paper (Section 4) we consider a natural extension of the temporal orientability problem, namely the *temporal transitive completion* problem. In this problem we are given a (partially oriented) temporal graph $\mathcal{G}$ and a natural number k, and the question is whether it is possible to add at most k new edges (with the corresponding time-labels) to $\mathcal{G}$ such that the resulting temporal graph is (strongly/strictly/strongly strictly) transitively orientable. We prove that all four versions of temporal transitive completion are NP-complete, even when the input temporal graph is completely unoriented. In contrast we show that, if the input temporal graph $\mathcal{G}$ is *directed* (i.e. if every time-labeled edge has a fixed orientation) then all versions of temporal transitive completion are solvable in polynomial time. As a corollary of our results it follows that all four versions of temporal transitive completion are fixed-parameter-tractable (FPT) with respect to the number q of unoriented time-labeled edges in $\mathcal{G}$.

In the third and last part of our paper (Section 5) we consider the *multilayer transitive orientation* problem. In this problem we are given an undirected temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$, and we ask whether there exists an orientation F of its edges (i.e. with exactly one orientation for each edge of G) such that, for every 'time-layer" $t \geq 1$, the (static) oriented graph induced by the edges having time-label t is transitively oriented in F. Problem definitions

[5]That is, orienting an edge from u to v *forces* us to orient another edge from a to b.

of this type are commonly referred to as multilayer problems [11]. Observe that this problem trivially reduces to the static case if we assume that each edge has a single time-label, as then each layer can be treated independently of all others. However, if we allow $\mathcal{G}$ to have multiple time-labels on every edge of G, then we show that the problem becomes NP-complete, even when every edge has at most two labels.

2. Preliminaries and Notation

Given a (static) undirected graph $G = (V, E)$, an edge between two vertices $u, v \in V$ is denoted by the unordered pair $\{u, v\} \in E$, and in this case the vertices u, v are said to be *adjacent*. If the graph is directed, we will use the ordered pair (u, v) (resp. (v, u)) to denote the oriented edge from u to v (resp. from v to u). For simplicity of the notation, we will usually drop the parentheses and the comma when denoting an oriented edge, i.e. we will denote (u, v) just by uv. Furthermore, $\widehat{uv} = \{uv, vu\}$ is used to denote the set of both oriented edges uv and vu between the vertices u and v.

Let $S \subseteq E$ be a subset of the edges of an undirected (static) graph $G = (V, E)$, and let $\widehat{S} = \{uv, vu : \{u, v\} \in S\}$ be the set of both possible orientations uv and vu of every edge $\{u, v\} \in S$. Let $F \subseteq \widehat{S}$. If F contains *at least one* of the two possible orientations uv and vu of each edge $\{u, v\} \in S$, then F is called an *orientation* of the edges of S. F is called a *proper orientation* if it contains *exactly one* of the orientations uv and vu of every edge $\{u, v\} \in S$. Note here that, in order to simplify some technical proofs, the above definition of an orientation allows F to be not proper, i.e. to contain *both* uv and vu for a specific edge $\{u, v\}$. However, whenever F is not proper, this means that F can be discarded as it cannot be used as a part of a (temporal) transitive orientation. For every orientation F denote by $F^{-1} = \{vu : uv \in F\}$ the *reversal* of F. Note that $F \cap F^{-1} = \emptyset$ if and only if F is proper.

In a temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$, whenever $\lambda(\{v, w\}) = t$ (or simply $\lambda(v, w) = t$), we refer to the tuple $(\{v, w\}, t)$ as a *time-edge* of $\mathcal{G}$. A triangle of (G, λ) on the vertices u, v, w is a *synchronous triangle* if $\lambda(u, v) = \lambda(v, w) = \lambda(w, u)$. Let $G = (V, E)$ and let F be a proper orientation of the whole edge set E. Then $(\mathcal{G}, F)$, or (G, λ, F), is a *proper orientation* of the temporal graph $\mathcal{G}$; for simplicity we may also write that F is a proper orientation of $\mathcal{G}$. A *partial proper orientation* F of a temporal graph $\mathcal{G} = (G, \lambda)$ is an orientation of a subset of E. To indicate that the edge $\{u, v\}$ of a time-edge $(\{u, v\}, t)$ is oriented from u to v (that is, $uv \in F$ in a (partial) proper orientation F), we use the term $((u, v), t)$, or simply (uv, t). For simplicity we may refer to a (partial) proper orientation just as a (partial) orientation, whenever the term "proper" is clear from the context.

A static graph $G = (V, E)$ is a *comparability graph* if there exists a proper orientation F of E which is *transitive*, that is, if $F \cap F^{-1} = \emptyset$ and $F^2 \subseteq F$, where $F^2 = \{uw : uv, vw \in F \text{ for some vertex } v\}$ [25]. Analogously, in a temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$, we define a proper orientation F of E to be *temporally transitive*, if:

whenever (uv, t_1) and (vw, t_2) are oriented time-edges in $(\mathcal{G}, F)$ such that $t_2 \geq t_1$, there exists an oriented time-edge (uw, t_3) in $(\mathcal{G}, F)$, for some $t_3 \geq t_2$.

In the above definition of a temporally transitive orientation, if we replace the condition "$t_3 \geq t_2$" with "$t_3 > t_2$", then F is called *strongly temporally transitive*. If we instead replace the condition "$t_2 \geq t_1$" with "$t_2 > t_1$", then F is called *strictly temporally transitive*. If we do both of these replacements, then F is called *strongly strictly temporally transitive*. Note that strong (strict) temporal transitivity implies (strict) temporal transitivity, while (strong) temporal transitivity implies (strong) strict temporal transitivity. Furthermore, similarly to the established terminology for static graphs, we define a temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$, to be a *(strongly/strictly) temporal comparability graph* if there exists a proper orientation F of E which is *(strongly/strictly) temporally transitive*.

We are now ready to formally introduce the following decision problem of recognizing whether a given temporal graph is temporally transitively orientable or not.

Temporal Transitive Orientation (TTO)

Input: A temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$.
Question: Does $\mathcal{G}$ admit a temporally transitive orientation F of E?

In the above problem definition of TTO, if we ask for the existence of a strictly (resp. strongly, or strongly strictly) temporally transitive orientation F, we obtain the decision problem Strict (resp. Strong, or Strong Strict) Temporal Transitive Orientation (TTO).

Let $\mathcal{G} = (G, \lambda)$ be a temporal graph, where $G = (V, E)$. Let $G' = (V, E')$ be a graph such that $E \subseteq E'$, and let $\lambda' \colon E' \to \mathbb{N}$ be a time-labeling function such that $\lambda'(u, v) = \lambda(u, v)$ for every $\{u, v\} \in E$. Then the temporal graph $\mathcal{G}' = (G', \lambda')$ is called a *temporal supergraph of* $\mathcal{G}$. We can now define our next problem definition regarding computing temporally orientable supergraphs of $\mathcal{G}$.

Temporal Transitive Completion (TTC)

Input: A temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$, a (partial) orientation F of $\mathcal{G}$, and an integer k.
Question: Does there exist a temporal supergraph $\mathcal{G}' = (G', \lambda')$ of (G, λ), where $G' = (V, E')$, and a transitive orientation $F' \supseteq F$ of $\mathcal{G}'$ such that $|E' \setminus E| \leq k$?

Similarly to TTO, if we ask in the problem definition of TTC for the existence of a strictly (resp. strongly, or strongly strictly) temporally transitive orientation F', we obtain the decision problem Strict (resp. Strong, or Strong Strict) Temporal Transitive Completion (TTC).

Now we define our final problem which asks for an orientation F of a temporal graph $\mathcal{G} = (G, \lambda)$ (i.e. with exactly one orientation for each edge of G) such that, for every "time-layer" $t \geq 1$, the (static) oriented graph defined by the edges having time-label t is transitively oriented in F. This problem does not make much sense if every edge has exactly one time-label in $\mathcal{G}$, as in this case it can be easily solved by just repeatedly applying any known static transitive orientation algorithm. Therefore, in the next problem definition, we assume that in the input temporal graph $\mathcal{G} = (G, \lambda)$ every edge of G potentially has multiple time-labels, i.e. the time-labeling function is $\lambda : E \to 2^{\mathbb{N}}$.

Multilayer Transitive Orientation (MTO)

Input: A temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$ and $\lambda : E \to 2^{\mathbb{N}}$.

Question: Is there an orientation F of the edges of G such that, for every $t \geq 1$, the (static) oriented graph induced by the edges having time-label t is transitively oriented?

3. The recognition of temporally transitively orientable graphs

In this section we investigate the computational complexity of all variants of TTO. We show that TTO as well as the two variants Strong TTO and Strong Strict TTO, are solvable in polynomial time, whereas Strict TTO turns out to be NP-complete.

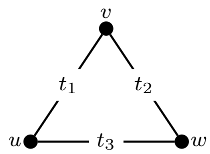
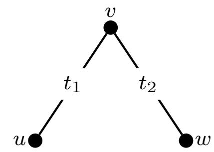
The main idea of our approach to solve TTO and its variants is to create Boolean variables for each edge of the underlying graph G and interpret setting a variable to 1 or 0 as the two possible ways of directing the corresponding edge.

More formally, for every edge $\{u, v\}$ we introduce a variable x_{uv} and setting this variable to 1 corresponds to the orientation uv while setting this variable to 0 corresponds to the orientation vu. Now consider the example of Figure 3(a), i.e. an induced path of length two in the underlying graph G on three vertices u, v, w, and let $\lambda(u, v) = 1$ and $\lambda(v, w) = 2$. Then the orientation uv "forces" the orientation wv. Indeed, if we otherwise orient $\{v, w\}$ as vw, then the edge $\{u, w\}$ must exist and be oriented as uw in any temporal transitive orientation, which is a contradiction as there is no edge between u and w. We can express this "forcing" with the implication $x_{uv} \implies x_{wv}$. In this way we can deduce the constraints that all triangles or induced paths on three vertices impose on any (strong/strict/strong strict) temporal transitive orientation. We collect all these constraints in Table 1.

When looking at the conditions imposed on temporal transitive orientations collected in Table 1, we can observe that all conditions except "non-cyclic" are expressible in 2SAT. Since 2SAT is solvable in linear time [7], it immediately follows that the strong variants of temporal transitivity are solvable in polynomial time, as the next theorem states.

Theorem 2. Strong TTO *and* Strong Strict TTO *are solvable in polynomial time.*

In the variants TTO and Strict TTO, however, we can have triangles which impose a "non-cyclic" orientation of three edges (Table 1). This can be

	$t_1 = t_2 = t_3$	$t_1 < t_2 = t_3$	$t_1 \leq t_2 < t_3$	$t_1 = t_2$	$t_1 < t_2$
TTO	non-cyclic	$wu = wv$	$vw \implies uw$ $vu \implies wu$	$uv = wv$	$uv \implies wv$
Strong TTO	$\perp$	$wu \wedge wv$	$vw \implies uw$ $vu \implies wu$	$uv = wv$	$uv \implies wv$
Strict TTO	$\top$	non-cyclic	$vw \implies uw$ $vu \implies wu$	$\top$	$uv \implies wv$
Str. Str. TTO	$\top$	$vu \implies wu$ $uv \implies wv$	$vw \implies uw$ $vu \implies wu$	$\top$	$uv \implies wv$

Table 1: Orientation conditions imposed by a triangle (left) and an induced path of length two (right) in the underlying graph G for the decision problems (Strict/Strong/Strong Strict) TTO. Here, $\top$ means that no restriction is imposed, $\perp$ means that the graph is not orientable, and in the case of triangles, "non-cyclic" means that all orientations except the ones that orient the triangle cyclicly are allowed.

naturally modeled by a not-all-equal (NAE) clause.[6] However, if we now naïvely model the conditions with a Boolean formula, we obtain a formula with 2SAT clauses and 3NAE clauses. Deciding whether such a formula is satisfiable is NP-complete in general [44]. Hence, we have to investigate these two variants more thoroughly.

The only difference between the triangles that impose these "non-cyclic" orientations in these two problem variants is that, in TTO, the triangle is *synchronous* (i.e. all its three edges have the same time-label), while in Strict TTO two of the edges are synchronous and the third one has a smaller time-label than the other two. As it turns out, this difference of the two problem variants has important implications on their computational complexity. In fact, we obtain a surprising result: TTO is solvable in polynomial time while Strict TTO is NP-complete.

In Section 3.1 we prove that Strict TTO is NP-complete and in Section 3.2 we provide our polynomial-time algorithm for TTO.

3.1. Strict TTO *is NP-Complete*

In this section we show that in contrast to the other variants, Strict TTO is NP-complete.

[6] A not all equal clause is a set of literals and it evaluates to `true` if and only if at least two literals in the set evaluate to different truth values.

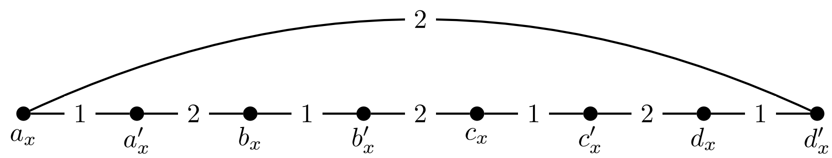


Figure 1: Illustration of the variable gadget used in the reduction in the proof of Theorem 3.

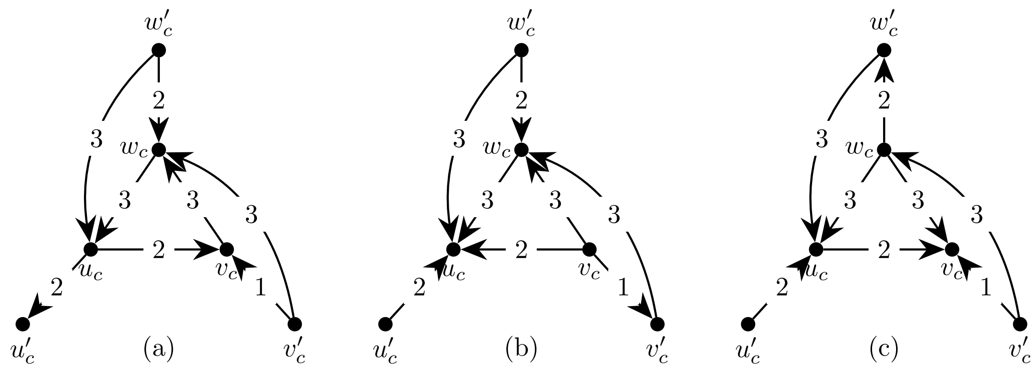


Figure 2: Illustration of the clause gadget used in the reduction in the proof of Theorem 3 and three ways how to orient the edges in it.

Theorem 3. Strict TTO *is NP-complete even if the temporal input graph has only four different time labels.*

Proof. We present a polynomial time reduction from (3,4)-SAT [48] where, given a CNF formula ϕ where each clause contains exactly three literals and each variably appears in exactly four clauses, we are asked whether ϕ is satisfiable or not. Given a formula ϕ, we construct a temporal graph $\mathcal{G}$ as follows.

Variable gadget. For each variable x that appears in ϕ, we add eight vertices $a_x, a'_x, b_x, b'_x, c_x, c'_x, d_x, d'_x$ to $\mathcal{G}$. We connect these vertices as depicted in Figure 1, that is, we add the following time edges to $\mathcal{G}$: $(\{a_x, a'_x\}, 1)$, $(\{a'_x, b_x\}, 2)$, $(\{b_x, b'_x\}, 1)$, $(\{b'_x, c_x\}, 2)$, $(\{c_x, c'_x\}, 1)$, $(\{c'_x, d_x\}, 2)$, $(\{d_x, d'_x\}, 1)$, $(\{d'_x, a_x\}, 2)$.

Clause gadget. For each clause c of ϕ, we add six vertices $u_c, u'_c, v_c, v'_c, w_c, w'_c$ to $\mathcal{G}$. We connect these vertices as depicted in Figure 2, that is, we add the following time edges to $\mathcal{G}$: $(\{u_c, u'_c\}, 2)$, $(v_c, v'_c\}, 1)$, $(\{w_c, w'_c\}, 2)$, $(\{u_c, v_c\}, 2)$, $(\{v_c, w_c\}, 3)$, $(\{w_c, u_c\}, 3)$, $(\{v_c, w'_c\}, 3)$, $(\{w_c, v'_c\}, 3)$.

Connecting variable gadgets and clause gadgets. Let variable x appear for the ith time in clause c and let x appear in the jth literal of c. The four vertex pairs $(a_x, a'_x), (b_x, b'_x), (c_x, c'_x), (d_x, d'_x)$ from the variable gadget of x correspond to the first, second, third, and fourth appearance of x, respectively. The three vertices u'_c, v'_c, w'_c correspond to the first, second, and third literal of c, respectively.

Let $i = 1$ and $j = 1$. If x appears non-negated, then we add the time edge $(\{a_x, u'_c\}, 4)$. Otherwise, if x appears negated, we add the time edge $(\{a'_x, u'_c\}, 4)$. For all other values of i and j we add time edges analogously.

This finishes the reduction. It can clearly be performed in polynomial time.

($\Rightarrow$): Assume that we have a satisfying assignment for ϕ, then we can orient $\mathcal{G}$ as follows. Then if a variable x is set to `true`, we orient the edges of the corresponding variable gadgets as follows: (a_x, a'_x), (b_x, a'_x), (b_x, b'_x), (c_x, b'_x), (c_x, c'_x), (d_x, c'_x), (d_x, d'_x), (a_x, d'_x). Otherwise, if x is set to false, we orient as follows: (a'_x, a_x), (a'_x, b_x), (b'_x, b_x), (b'_x, c_x), (c'_x, c_x), (c'_x, d_x), (d'_x, d_x), (d'_x, a_x). It is easy so see that both orientations are transitive.

Now consider a clause in ϕ with literals u, v, w corresponding to vertices u'_c, v'_c, w'_c of the clause gadget, respectively. We have that at least one of the three literals satisfies the clause. If it is u, then we orient the edges in the clause gadgets as illustrated in Figure 2 (a). It is easy so see that this orientation is transitive. Furthermore, we orient the three edges connecting the clause gadgets to variable gadgets as follows: By construction the vertices u'_c, v'_c, w'_c are each connected to a variable gadget. Assume, we have edges $\{u'_c, x\}, \{v'_c, y\}, \{w'_c, z\}$. Then we orient as follows: $(x, u'_c), (v'_c, y), (w'_c, z)$, that is, we orient the edge connecting the literal that satisfies the clause towards the clause gadget and the other two edges towards the variable gadgets. This yields a transitive in the clause gadget. Note that the variable gadgets have time labels 1 and 2 so we can always orient the connecting edges (which have time label 4) towards the variable gadget. We do this with all connecting edges except (x, u'_c). This edge is oriented from the variable gadget towards the clause gadget, however it also corresponds to a literal that satisfies the clause. Then by construction, the edges incident to x in the variable gadget are oriented away from x, hence our orientation is transitive.

Otherwise and if v satisfies the clause, then we orient the edges in the clause gadgets as illustrated in Figure 2 (b). Otherwise (in this case w has to satisfy the clause), we orient the edges in the clause gadgets as illustrated in Figure 2 (c). It is easy so see that each of these orientation is transitive. In both cases we orient the edges connecting the clause gadgets to the variable gadgets analogously to the first case discussed above. By analogous arguments we get that the resulting orientation is transitive.

($\Leftarrow$): Note that all variable gadgets are cycles of length eight with edges having labels alternating between 1 and 2 and hence the edges have to also be oriented alternately. Consider the variable gadget corresponding to x. We interpret the orientation (a_x, a'_x), (b_x, a'_x), (b_x, b'_x), (c_x, b'_x), (c_x, c'_x), (d_x, c'_x), (d_x, d'_x), (a_x, d'_x) as setting x to `true` and we interpret the orientation (a'_x, a_x), (a'_x, b_x), (b'_x, b_x), (b'_x, c_x), (c'_x, c_x), (c'_x, d_x), (d'_x, d_x), (d'_x, a_x) as setting x to true. We claim that this yields a satisfying assignment for ϕ.

Assume for contradiction that there is a clause c in ϕ that is not satisfied by this assignment. Then by construction of the connection of variable gadgets and clause gadgets, the connecting edges have to be oriented towards the variable gadget in order to keep the variable gadget transitive. Let the three connect-

Figure 3: The orientation uv forces the orientation wu and vice-versa in the examples of (a) a static graph G where $\{u, v\}, \{v, w\} \in E(G)$ and $\{u, w\} \notin E(G)$, and of (b) a temporal graph (G, λ) where $\lambda(u, w) = 3 < 5 = \lambda(u, v) = \lambda(v, w)$.

ing edges be $\{u'_c, x\}, \{v'_c, y\}, \{w'_c, z\}$ and their orientation $(u'_c, x), (v'_c, y), (w'_c, z)$. Then we have that (u'_c, x) forces (u'_c, u_c) which in turn forces (w_c, u_c). We have that (v'_c, y) forces (v'_c, v_c) which in turn forces (v_c, u_c). Furthermore, we now have that (w_c, u_c) and (v_c, u_c) force (w_c, v_c). Lastly, we have that (w'_c, z) forces (w'_c, w_c) which in turn forces (v_c, w_c), a contradiction to the fact that we forced (w_c, v_c) previously. $\square$

3.2. A polynomial-time algorithm for TTO

Let $G = (V, E)$ be a static undirected graph. There are various polynomial-time algorithms for deciding whether G admits a transitive orientation F. However our results in this section are inspired by the transitive orientation algorithm described by Golumbic [25], which is based on the crucial notion of *forcing* an orientation. The notion of forcing in static graphs is illustrated in Figure 3 (a): if we orient the edge $\{u, v\}$ as uv (i.e., from u to v) then we are forced to orient the edge $\{v, w\}$ as wv (i.e., from w to v) in any transitive orientation F of G. Indeed, if we otherwise orient $\{v, w\}$ as vw (i.e. from v to w), then the edge $\{u, w\}$ must exist and it must be oriented as uw in any transitive orientation F of G, which is a contradiction as $\{u, w\}$ is not an edge of G. Similarly, if we orient the edge $\{u, v\}$ as vu then we are forced to orient the edge $\{v, w\}$ as vw. That is, in any transitive orientation F of G we have that $uv \in F \Leftrightarrow wv \in F$. This forcing operation can be captured by the binary forcing relation Γ which is defined on the edges of a static graph G as follows [25].

$$uv \;\Gamma\; u'v' \quad \text{if and only if} \quad \begin{cases} \text{either } u = u' \text{ and } \{v, v'\} \notin E \\ \text{or } v = v' \text{ and } \{u, u'\} \notin E \end{cases} . \tag{1}$$

We now extend the definition of Γ in a natural way to the binary relation Λ on the edges of a temporal graph (G, λ), see Equation (2). For this, observe from Table 1 that the only cases, where we have $uv \in F \Leftrightarrow wv \in F$ in any temporal transitive orientation of (G, λ), are when (i) the vertices u, v, w induce a path of length 2 (see Figure 3 (a)) and $\lambda(u, v) = \lambda(v, w)$, as well as when (ii) u, v, w induce a triangle and $\lambda(u, w) < \lambda(u, v) = \lambda(v, w)$. The latter situation is illustrated in the example of Figure 3 (b). The binary forcing relation Λ is only defined on pairs of edges $\{u,v\}$ and $\{u',v'\}$ where $\lambda(u,v) = \lambda(u',v')$, as follows.

$$uv \,\Lambda\, u'v' \text{ if and only if } \lambda(u,v) = \lambda(u',v') = t \text{ and } \begin{cases} u = u' \text{ and } \{v,v'\} \notin E, \text{or} \\ v = v' \text{ and } \{u,u'\} \notin E, \text{or} \\ u = u' \text{ and } \lambda(v,v') < t, \text{or} \\ v = v' \text{ and } \lambda(u,u') < t. \end{cases} \tag{2}$$

Note that, for every edge $\{u,v\} \in E$ we have that $uv \,\Lambda\, uv$. The forcing relation Λ for temporal graphs shares some properties with the forcing relation Γ for static graphs. In particular, the reflexive transitive closure Λ^* of Λ is an equivalence relation, which partitions the edges of each set $E_t = \{\{u,v\} \in E : \lambda(u,v) = t\}$ into its Λ-*implication classes* (or simply, into its *implication classes*). Two edges $\{a,b\}$ and $\{c,d\}$ are in the same Λ-implication class if and only $ab \;\Lambda^*\; cd$, i.e. there exists a sequence

$$ab = a_0b_0 \;\Lambda\; a_1b_1 \;\Lambda\; \ldots \;\Lambda\; a_kb_k = cd, \text{ with } k \geq 0.$$

Note that, for this to happen, we must have $\lambda(a_0,b_0) = \lambda(a_1,b_1) = \ldots = \lambda(a_k,b_k) = t$ for some $t \geq 1$. Such a sequence is called a Λ-chain from ab to cd, and we say that ab (eventually) Λ-forces cd. Furthermore note that $ab \;\Lambda^*\; cd$ if and only if $ba \;\Lambda^*\; dc$. The next observation helps the reader understand the relationship between the two forcing relations Γ and Λ.

Observation 4. *Let $\{u,v\} \in E$, where $\lambda(u,v) = t$, and let A be the Λ-implication class of uv in the temporal graph (G,λ). Let G' be the static graph obtained by removing from G all edges $\{p,q\}$, where $\lambda(p,q) < t$. Then A is also the Γ-implication class of uv in the static graph G'.*

For the next lemma, we use the notation $\widehat{A} = \{uv, vu : uv \in A\}$.

Lemma 5. *Let A be a Λ-implication class of a temporal graph (G,λ). Then either $A = A^{-1} = \widehat{A}$ or $A \cap A^{-1} = \emptyset$.*

Proof. Suppose that $A \cap A^{-1} \neq \emptyset$, and let $uv \in A \cap A^{-1}$, i.e. $uv, vu \in A$. Then, for any $pq \in A$ we have that $pq \;\Lambda^*\; uv$ and $qp \;\Lambda^*\; vu$. Since Λ^* is an equivalence relation and $uv, vu \in A$, it also follows that $pq, qp \in A$. Therefore also $pq, qp \in A^{-1}$, and thus $A = A^{-1} = \widehat{A}$. □

Definition 6. Let F be a proper orientation and A be a Λ-implication class of a temporal graph (G,λ). If $A \subseteq F$, we say that F *respects* A.

Lemma 7. *Let F be a proper orientation and A be a Λ-implication class of a temporal graph (G,λ). Then F respects either A or A^{-1} (i.e. either $A \subseteq F$ or $A^{-1} \subseteq F$), and in either case $A \cap A^{-1} = \emptyset$.*

Proof. We defined the binary forcing relation Λ to capture the fact that, for any temporal transitive orientation F of (G,λ), if $ab \;\Lambda\; cd$ and $ab \in F$, then also $cd \in F$. Applying this property repeatedly, it follows that either $A \subseteq F$ or

$F \cap A = \emptyset$. If $A \subseteq F$ then $A^{-1} \subseteq F^{-1}$. On the other hand, if $F \cap A = \emptyset$ then $A \subseteq F^{-1}$, and thus also $A^{-1} \subseteq F$. In either case, the fact that $F \cap F^{-1} = \emptyset$ by the definition of a temporal transitive orientation implies that also $A \cap A^{-1} = \emptyset$. $\square$

Let now $ab = a_0b_0 \ \Lambda \ a_1b_1 \ \Lambda \ \ldots \ \Lambda \ a_kb_k = cd$ be a given Λ-chain. Note by Equation (2) that, for every $i = 1, \ldots, k$, we have that either $a_{i-1} = a_i$ or $b_{i-1} = b_i$. Therefore we can replace the Λ-implication $a_{i-1}b_{i-1} \ \Lambda \ a_ib_i$ by the implications $a_{i-1}b_{i-1} \ \Lambda \ a_ib_{i-1} \ \Lambda \ a_ib_i$, since either $a_ib_{i-1} = a_{i-1}b_{i-1}$ or $a_ib_{i-1} = a_ib_i$. Thus, as this addition of this middle edge is always possible in a Λ-implication, we can now define the notion of a canonical Λ-chain, which always exists.

Definition 8. Let $ab \ \Lambda^* \ cd$. Then any Λ-chain of the from

$$ab = a_0b_0 \ \Lambda \ a_1b_0 \ \Lambda \ a_1b_1 \ \Lambda \ \ldots \ \Lambda \ a_kb_{k-1} \ \Lambda \ a_kb_k = cd$$

is a *canonical* Λ-chain.

The next lemma extends an important known property of the forcing relation Γ for static graphs [25, Lemma 5.3] to the temporal case.

Lemma 9 (Temporal Triangle Lemma)**.** *Let (G, λ) be a temporal graph with a synchronous triangle on the vertices a, b, c, where $\lambda(a, b) = \lambda(b, c) = \lambda(c, a) = t$. Let A, B, C be three Λ-implication classes of (G, λ), where $ab \in C$, $bc \in A$, and $ca \in B$, where $A \neq B^{-1}$ and $A \neq C^{-1}$.*

1. *If some $b'c' \in A$, then $ab' \in C$ and $c'a \in B$.*
2. *If some $b'c' \in A$ and $a'b' \in C$, then $c'a' \in B$.*
3. *No edge of A touches vertex a.*

Proof. 1. Let $b'c' \in A$, and let $bc = b_0c_0 \ \Lambda \ b_1c_0 \ \Lambda \ \ldots \ \Lambda \ b_kc_{k-1} \ \Lambda \ b_kc_k = b'c'$ be a canonical Λ-chain from bc to $b'c'$. Thus note that all edges b_ic_{i-1} and b_ic_i of this Λ-chain have the same time-label t in (G, λ). We will prove by induction that $ab_i \in C$ and $c_ia \in B$, for every $i = 0, 1, \ldots, k$. The induction basis follows directly by the statement of the lemma, as $ab = ab_0 \in C$ and $ca = c_0a \in B$.
Assume now that $ab_i \in C$ and $c_ia \in B$. If $b_{i+1} = b_i$ then clearly $ab_{i+1} \in C$ by the induction hypothesis. Suppose now that $b_{i+1} \neq b_i$. If $\{a, b_{i+1}\} \notin E$ then $ac_i \ \Lambda \ b_{i+1}c_i$. Then, since $c_ia \in B$ and $b_{i+1}c_i \in A$, it follows that $A = B^{-1}$, which is a contradiction to the assumption of the lemma. Therefore $\{a, b_{i+1}\} \in E$. Furthermore, since $b_ic_i \ \Lambda \ b_{i+1}c_i$, it follows that either $\{b_i, b_{i+1}\} \notin E$ or $\lambda(b_i, b_{i+1}) < t$. In either case it follows that $ab_i \ \Lambda \ ab_{i+1}$, and thus $ab_{i+1} \in C$.
Similarly, if $c_{i+1} = c_i$ then $c_{i+1}a \in B$ by the induction hypothesis. Suppose now that $c_{i+1} \neq c_i$. If $\{a, c_{i+1}\} \notin E$ then $ab_{i+1} \ \Lambda \ c_{i+1}b_{i+1}$. Then, since $ab_{i+1} \in C$ and $b_{i+1}c_{i+1} \in A$, it follows that $A = C^{-1}$, which is a contradiction to the assumption of the lemma. Therefore $\{a, c_{i+1}\} \in E$.

Furthermore, since $b_{i+1}c_i \ \Lambda \ b_{i+1}c_{i+1}$, it follows that either $\{c_i, c_{i+1}\} \notin E$ or $\lambda(c_i, c_{i+1}) < t$. In either case it follows that $c_i a \ \Lambda \ c_{i+1}a$, and thus $c_{i+1}a \in C$. This completes the induction step.

2. Let $b'c' \in A$ and $a'b' \in C$. Then part 1 of the lemma implies that $c'a \in B$. Now let $ab = a_0b_0 \ \Lambda \ a_1b_0 \ \Lambda \ \ldots \ \Lambda \ a_\ell b_{\ell-1} \ \Lambda \ a_\ell b_\ell = a'b'$ be a canonical Λ-chain from ab to $a'b'$. Again, note that all edges a_ib_{i-1} and a_ib_i of this Λ-chain have the same time-label t in (G, λ). We will prove by induction that $c'a_i \in B$ and $b_ic' \in A$ for every $i = 0, 1, \ldots, k$. First recall that $c'a = c'a_0 \in B$. Furthermore, by applying part 1 of the proof to the triangle with vertices a_0, b_0, c and on the edge $c'a_0 \in B$, it follows that $b_0c' \in A$. This completes the induction basis.
   For the induction step, assume that $c'a_i \in B$ and $b_ic' \in A$. If $a_{i+1} = a_i$ then clearly $c'a_{i+1} \in B$. Suppose now that $a_{i+1} \neq a_i$. If $\{a_{i+1}, c'\} \notin E$ then $a_{i+1}b_i \ \Lambda \ c'b_i$. Then, since $a_{i+1}b_i \in C$ and $b_ic' \in A$, it follows that $A = C^{-1}$, which is a contradiction to the assumption of the lemma. Therefore $\{a_{i+1}, c'\} \in E$. Now, since $a_ib_i \ \Lambda \ a_{i+1}b_i$, it follows that either $\{a_i, a_{i+1}\} \notin E$ or $\lambda(a_i, a_{i+1}) < t$. In either case it follows that $c'a_i \ \Lambda \ c'a_{i+1}$. Therefore, since $c'a_i \in B$, it follows that $c'a_{i+1} \in B$.
   If $b_{i+1} = b_i$ then clearly $b_{i+1}c' \in A$. Suppose now that $b_{i+1} \neq b_i$. Then, since $c'a_{i+1} \in B$, $a_{i+1}b_i \in C$, and $b_ic' \in A$, we can apply part 1 of the lemma to the triangle with vertices a_{i+1}, b_i, c' and on the edge $a_{i+1}b_{i+1} \in C$, from which it follows that $b_ic' \in A$. This completes the induction step, and thus $c'a_k = c'a' \in B$.

3. Suppose that $ad \in A$ (resp. $da \in A$), for some vertex d. Then, by setting $b' = a$ and $c' = d$ (resp. $b' = d$ and $c' = a$), part 1 of the lemma implies that $ab' = aa \in C$ (resp. $c'a = aa \in B$). Thus is a contradiction, as the underlying graph G does not have the edge aa. $\square$

Deciding temporal transitivity using Boolean satisfiability. Starting with any undirected edge $\{u, v\}$ of the underlying graph G, we can clearly enumerate in polynomial time the whole Λ-implication class A to which the oriented edge uv belongs (cf. Equation (2)). If the reversely directed edge $vu \in A$ then Lemma 5 implies that $A = A^{-1} = \widehat{A}$. Otherwise, if $vu \notin A$ then $vu \in A^{-1}$ and Lemma 5 implies that $A \cap A^{-1} = \emptyset$. Thus, we can also decide in polynomial time whether $A \cap A^{-1} = \emptyset$. If we encounter at least one Λ-implication class A such that $A \cap A^{-1} \neq \emptyset$, then it follows by Lemma 7 that (G, λ) is not temporally transitively orientable.

In the remainder of the section we will assume that $A \cap A^{-1} = \emptyset$ for every Λ-implication class A of (G, λ), which is a *necessary* condition for (G, λ) to be temporally transitive orientable. Moreover it follows by Lemma 7 that, if (G, λ) admits a temporally transitively orientation F, then either $A \subseteq F$ or $A^{-1} \subseteq F$. This allows us to define a Boolean variable x_A for every Λ-implication class A, where $x_A = \overline{x_{A^{-1}}}$. Here $x_A = 1$ (resp. $x_{A^{-1}} = 1$) means that $A \subseteq F$ (resp. $A^{-1} \subseteq F$), where F is the temporally transitive orientation which we are looking for. Let $\{A_1, A_2, \ldots, A_s\}$ be a set of Λ-implication classes such that

Algorithm 1 Building the Λ-implication classes and the edge-variables.

Input: A temporal graph (G, λ), where $G = (V, E)$.

Output: The variables $\{x_{uv}, x_{vu} : \{u, v\} \in E\}$, or the announcement that (G, λ) is temporally not transitively orientable.

$s \leftarrow 0$; $E_0 \leftarrow E$
while $E_0 \neq \emptyset$ **do**
  $s \leftarrow s + 1$; Let $\{p, q\} \in E_0$ be arbitrary
  Build the Λ-implication class A_s of the oriented edge pq (by Equation (2))
  if $qp \in A_s$ **then** $\{A_s \cap A_s^{-1} \neq \emptyset\}$
    return "NO"
  else
    x_s is the variable corresponding to the directed edges of A_s
    for every $uv \in A_s$ **do**
      $x_{uv} \leftarrow x_s$; $x_{vu} \leftarrow \overline{x_s}$ $\{x_{uv}$ and x_{vu} become aliases of x_s and $\overline{x_s}\}$
    $E_0 \leftarrow E_0 \setminus \widehat{A_s}$
return Λ-implication classes $\{A_1, A_2, \ldots, A_s\}$ and variables $\{x_{uv}, x_{vu} : \{u, v\} \in E\}$

$\{\widehat{A_1}, \widehat{A_2}, \ldots, \widehat{A_s}\}$ is a partition of the edges of the underlying graph G.[7] Then any truth assignment τ of the variables $x_1, x_2, \ldots, x_s$ (where $x_i = x_{A_i}$ for every $i = 1, 2, \ldots, s$) corresponds bijectively to one possible orientation of the temporal graph (G, λ), in which every Λ-implication class is oriented consistently.

Now we define two Boolean formulas ϕ_{3NAE} and ϕ_{2SAT} such that (G, λ) admits a temporal transitive orientation if and only if there is a truth assignment τ of the variables $x_1, x_2, \ldots, x_s$ such that both ϕ_{3NAE} and ϕ_{2SAT} are simultaneously satisfied. Intuitively, ϕ_{3NAE} captures the "non-cyclic" condition from Table 1 while ϕ_{2SAT} captures the remaining conditions. Here ϕ_{3NAE} is a 3NAE formula, i.e., the disjunction of clauses with three literals each, where every clause $\texttt{NAE}(\ell_1, \ell_2, \ell_3)$ is satisfied if and only if at least one of the literals $\{\ell_1, \ell_2, \ell_3\}$ is equal to 1 and at least one of them is equal to 0. Furthermore ϕ_{2SAT} is a 2SAT formula, i.e., the disjunction of 2CNF clauses with two literals each, where every clause $(\ell_1 \vee \ell_2)$ is satisfied if and only if at least one of the literals $\{\ell_1, \ell_2\}$ is equal to 1.

For simplicity of the presentation we also define a variable x_{uv} for every directed edge uv. More specifically, if $uv \in A_i$ (resp. $uv \in A_i^{-1}$) then we set $x_{uv} = x_i$ (resp. $x_{uv} = \overline{x_i}$). That is, $x_{uv} = \overline{x_{vu}}$ for every undirected edge $\{u, v\} \in E$. Note that, although $\{x_{uv}, x_{vu} : \{u, v\} \in E\}$ are defined as variables, they can equivalently be seen as *literals* in a Boolean formula over the variables $x_1, x_2, \ldots, x_s$. The process of building all Λ-implication classes and all variables $\{x_{uv}, x_{vu} : \{u, v\} \in E\}$ is given by Algorithm 1.

Description of the 3NAE formula ϕ_{3NAE}. The formula ϕ_{3NAE} captures the "non-cyclic" condition of the problem variant TTO (presented in Table 1). The formal description of ϕ_{3NAE} is as follows. Consider a synchronous

[7] Here we slightly abuse the notation by identifying the undirected edge $\{u, v\}$ with the set of both its orientations $\{uv, vu\}$.

triangle of (G, λ) on the vertices u, v, w. Assume that $x_{uv} = x_{wv}$, i.e., x_{uv} is the same variable as x_{wv}. Then the pair $\{uv, wv\}$ of oriented edges belongs to the same Λ-implication class A_i. This implies that the triangle on the vertices u, v, w is never cyclically oriented in any proper orientation F that respects A_i or A_i^{-1}. Note that, by symmetry, the same happens if $x_{vw} = x_{uw}$ or if $x_{wu} = x_{vu}$. Assume, on the contrary, that $x_{uv} \neq x_{wv}$, $x_{vw} \neq x_{uw}$, and $x_{wu} \neq x_{vu}$. In this case we add to $\phi_{3\text{NAE}}$ the clause $\texttt{NAE}(x_{uv}, x_{vw}, x_{wu})$. Note that the triangle on u, v, w is transitively oriented if and only if $\texttt{NAE}(x_{uv}, x_{vw}, x_{wu})$ is satisfied, i.e., at least one of the variables $\{x_{uv}, x_{vw}, x_{wu}\}$ receives the value 1 and at least one of them receives the value 0.

Description of the 2SAT formula $\phi_{2\text{SAT}}$. The formula $\phi_{2\text{SAT}}$ captures all conditions apart from the "non-cyclic" condition of the problem variant TTO (presented in Table 1). The formal description of $\phi_{2\text{SAT}}$ is as follows. Consider a triangle of (G, λ) on the vertices u, v, w, where $\lambda(u, v) = t_1$, $\lambda(v, w) = t_2$, $\lambda(w, v) = t_3$, and $t_1 \leq t_2 \leq t_3$. If $t_1 < t_2 = t_3$ then we add to $\phi_{2\text{SAT}}$ the clauses $(x_{uw} \vee x_{wv}) \wedge (x_{vw} \vee x_{wu})$; note that these clauses are equivalent to $x_{wu} = x_{wv}$. If $t_1 \leq t_2 < t_3$ then we add to $\phi_{2\text{SAT}}$ the clauses $(x_{wv} \vee x_{uw}) \wedge (x_{uv} \vee x_{wu})$; note that these clauses are equivalent to $(x_{vw} \Rightarrow x_{uw}) \wedge (x_{vu} \Rightarrow x_{wu})$. Now consider a path of length 2 that is induced by the vertices u, v, w, where $\lambda(u, v) = t_1$, $\lambda(v, w) = t_2$, and $t_1 \leq t_2$. If $t_1 = t_2$ then we add to $\phi_{2\text{SAT}}$ the clauses $(x_{vu} \vee x_{wv}) \wedge (x_{vw} \vee x_{uv})$; note that these clauses are equivalent to $(x_{uv} = x_{wv})$. Finally, if $t_1 < t_2$ then we add to $\phi_{2\text{SAT}}$ the clause $(x_{vu} \vee x_{wv})$; note that this clause is equivalent to $(x_{uv} \Rightarrow x_{wv})$.

In what follows, we say that $\phi_{3\text{NAE}} \wedge \phi_{2\text{SAT}}$ is *satisfiable* if and only if there exists a truth assignment τ which simultaneously satisfies both $\phi_{3\text{NAE}}$ and $\phi_{2\text{SAT}}$. Given the above definitions of $\phi_{3\text{NAE}}$ and $\phi_{2\text{SAT}}$, it is easy to check that their clauses model all conditions of the oriented edges imposed by the row of "TTO" in Table 1.

Observation 10. *The temporal graph (G, λ) is transitively orientable if and only if $\phi_{3NAE} \wedge \phi_{2SAT}$ is satisfiable.*

Although deciding whether $\phi_{2\text{SAT}}$ is satisfiable can be done in linear time with respect to the size of the formula [7], the problem Not-All-Equal-3-SAT is NP-complete [44]. We overcome this problem and present a polynomial-time algorithm for deciding whether $\phi_{3\text{NAE}} \wedge \phi_{2\text{SAT}}$ is satisfiable as follows.

Roadmap of the entire process. Our algorithm iteratively produces at iteration j a formula $\phi_{3\text{NAE}}^{(j)} \wedge \phi_{2\text{SAT}}^{(j)}$, which is computed from the previous formula $\phi_{3\text{NAE}}^{(j-1)} \wedge \phi_{2\text{SAT}}^{(j-1)}$ by (almost) simulating the classical greedy algorithm that solves 2SAT [7]. The classical 2SAT-algorithm proceeds greedily as follows. For every variable x_i, if setting $x_i = 1$ (resp. $x_i = 0$) leads to an immediate contradiction, the algorithm is forced to set $x_i = 0$ (resp. $x_i = 1$). Otherwise, if each of the truth assignments $x_i = 1$ and $x_i = 0$ does not lead to an immediate contradiction, the algorithm arbitrarily chooses to set $x_i = 1$ or $x_i = 0$, and thus some clauses are removed from the formula as they were satisfied. The argument for the correctness of this classical 2SAT-algorithm is that new clauses are *never added* to the formula at any step. The main technical difference between the 2SAT-algorithm and our algorithm is that, in our case, the formula $\phi^{(j)}_{3\text{NAE}} \wedge \phi^{(j)}_{2\text{SAT}}$ is *not* necessarily a sub-formula of $\phi^{(j-1)}_{3\text{NAE}} \wedge \phi^{(j-1)}_{2\text{SAT}}$, as in some cases we need to also add clauses.

Our main technical result is that, nevertheless, if the algorithm does not return "NO" while applying variable *forcings* at the initialization phase (during which $\phi^{(0)}_{3\text{NAE}} \wedge \phi^{(0)}_{2\text{SAT}}$ is computed), then the input instance is a *yes*-instance. In this case, the algorithm proceeds by computing the formulas $\phi^{(j)}_{3\text{NAE}} \wedge \phi^{(j)}_{2\text{SAT}}$, for $j = 1, 2, \ldots$, which eventually determine a valid temporally transitive orientation of the input temporal graph. The proof of this result (see Lemma 19 and Theorem 20) relies on a sequence of structural properties of temporal transitive orientations which we establish. This phenomenon of deducing a polynomial-time algorithm for an algorithmic graph problem by deciding satisfiability of a mixed Boolean formula (i.e. with both clauses of two and three literals) occurs rarely; this approach has been successfully used for the efficient recognition of simple-triangle (known also as "PI") graphs [35].

Brief outline of the algorithm. In the *initialization phase*, we exhaustively check which truth values are *forced* in $\phi_{3\text{NAE}} \wedge \phi_{2\text{SAT}}$ by using Initial-Forcing (see Algorithm 2) as a subroutine. During the execution of Initial-Forcing, we either replace the formulas $\phi_{3\text{NAE}}$ and $\phi_{2\text{SAT}}$ by the equivalent formulas $\phi^{(0)}_{3\text{NAE}}$ and $\phi^{(0)}_{2\text{SAT}}$, respectively, or we reach a contradiction by showing that $\phi_{3\text{NAE}} \wedge \phi_{2\text{SAT}}$ is unsatisfiable.

The *main phase* of the algorithm starts once the formulas $\phi^{(0)}_{3\text{NAE}}$ and $\phi^{(0)}_{2\text{SAT}}$ have been computed. During this phase, we iteratively modify the formulas such that, at the end of iteration j we have the formulas $\phi^{(j)}_{3\text{NAE}}$ and $\phi^{(j)}_{2\text{SAT}}$. Note that, during the execution of the algorithm, we can *both add and remove* clauses from $\phi^{(j)}_{2\text{SAT}}$. On the other hand, we can *only remove* clauses from $\phi^{(j)}_{3\text{NAE}}$. Thus, at some iteration j, we obtain $\phi^{(j)}_{3\text{NAE}} = \emptyset$, and after that iteration we only need to decide satisfiability of $\phi^{(j)}_{2\text{SAT}}$ which can be done efficiently [7].

Two crucial technical lemmas. For the remainder of the section we write $x_{ab} \overset{*}{\Rightarrow}_{\phi_{2\text{SAT}}} x_{uv}$ (resp. $x_{ab} \overset{*}{\Rightarrow}_{\phi^{(j)}_{2\text{SAT}}} x_{uv}$) if the truth assignment $x_{ab} = 1$ forces (in 0 or more iterations) the truth assignment $x_{uv} = 1$ from the clauses of $\phi_{2\text{SAT}}$ (resp. of $\phi^{(j)}_{2\text{SAT}}$ at the iteration j of the algorithm); in this case we say that x_{ab} *implies* x_{uv} in $\phi_{2\text{SAT}}$ (resp. in $\phi^{(j)}_{2\text{SAT}}$). We next introduce the notion of *uncorrelated* triangles, which lets us formulate some important properties of the implications in $\phi_{2\text{SAT}}$ and $\phi^{(0)}_{2\text{SAT}}$.

Definition 11. Let u, v, w induce a synchronous triangle in (G, λ), where each of the variables of the set $\{x_{uv}, x_{vu}, x_{vw}, x_{wv}, x_{wu}, x_{uw}\}$ belongs to a *different* Λ-implication class. If *none* of the variables of the set $\{x_{uv}, x_{vu}, x_{vw}, x_{wv}, x_{wu}, x_{uw}\}$ implies any other variable of the same set in the formula $\phi_{2\text{SAT}}$ (resp. in the formula $\phi_{2\text{SAT}}^{(0)}$), then the triangle of u, v, w is *ϕ_{2SAT}-uncorrelated* (resp. *$\phi_{2SAT}^{(0)}$-uncorrelated*).

Now we present our two crucial technical lemmas (Lemmas 12 and 13) which prove some structural properties of the 2SAT formulas $\phi_{2\text{SAT}}$ and $\phi_{2\text{SAT}}^{(0)}$. These structural properties will allow us to prove the correctness of our main algorithm in this section (Algorithm 4). In a nutshell, these two lemmas guarantee that, whenever we have specific implications in $\phi_{2\text{SAT}}$ (resp. in $\phi_{2\text{SAT}}^{(0)}$), then we also have some specific *other* implications in the same formula.

Lemma 12. *Let u, v, w induce a synchronous and ϕ_{2SAT}-uncorrelated triangle in (G, λ), and let $\{a, b\} \in E$ be an edge of G such that $|\{a, b\} \cap \{u, v, w\}| \leq 1$. If $x_{ab} \overset{*}{\Rightarrow}_{\phi_{2SAT}} x_{uv}$, then x_{ab} also implies in ϕ_{2SAT} at least one of the four variables in the set $\{x_{vw}, x_{wv}, x_{uw}, x_{wu}\}$.*

Proof. Let t be the common time-label of all the edges in the synchronous triangle of the vertices u, v, w. That is, $\lambda(u, v) = \lambda(v, w) = \lambda(w, u) = t$. Denote by A, B, and C the Λ-implication classes in which the directed edges uv, vw, and wu belong, respectively. Let $x_{ab} = x_{a_0 b_0} \Rightarrow_{\phi_{2\text{SAT}}} x_{a_1 b_1} \Rightarrow_{\phi_{2\text{SAT}}} \ldots \Rightarrow_{\phi_{2\text{SAT}}} x_{a_{k-1} b_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{a_k b_k} = x_{uv}$ be a $\phi_{2\text{SAT}}$-implication chain from x_{ab} to x_{uv}. Note that, without loss of generality, for each variable $x_{a_i b_i}$ in this chain, the directed edge $a_i b_i$ is a representative of a different Λ-implication class than all other directed edges in the chain (otherwise we can just shorten the $\phi_{2\text{SAT}}$-implication chain from x_{ab} to x_{uv}). Furthermore, since $x_{a_k b_k} = x_{uv}$, note that $a_k b_k$ and uv are both representatives of the same Λ-implication class A. Therefore Lemma 9 (the temporal triangle lemma) implies that $w a_k \in C$ and $b_k w \in B$. Therefore we can assume without loss of generality that $u = a_k$ and $v = b_k$. Moreover, let $A' \notin \{A, A^{-1}, B, B^{-1}, C, C^{-1}\}$ be the Λ-implication class in which the directed edge $a_{k-1} b_{k-1}$ belongs. Since $x_{a_{k-1} b_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{a_k b_k}$, note that without loss of generality we can choose the directed edge $a_{k-1} b_{k-1}$ to be such a representative of the Λ-implication class A' such that either $a_{k-1} = a_k$ or $b_{k-1} = b_k$. We now distinguish these two cases.

Case 1: $u = a_k = a_{k-1}$ **and** $v = b_k \neq b_{k-1}$**.** Then, since $x_{a_{k-1} b_{k-1}} = x_{a_k b_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{a_k b_k} = x_{uv}$ and $\lambda(a_k, b_k) = t$, it follows that $\lambda(u, b_{k-1}) \geq t + 1$. Suppose that $\{w, b_{k-1}\} \notin E$. Then $x_{u b_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{uw}$, which proves the lemma. Now suppose that $\{w, b_{k-1}\} \in E$. If $\lambda(w, b_{k-1}) \leq \lambda(u, b_{k-1}) - 1$ then $x_{u b_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{uw}$, which proves the lemma. Suppose that $\lambda(w, b_{k-1}) \geq \lambda(u, b_{k-1}) + 1$. Then $x_{u b_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{w b_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{wu}$, i.e. $x_{u b_{k-1}} \overset{*}{\Rightarrow}_{\phi_{2\text{SAT}}} x_{wu}$, which again proves the lemma. Suppose finally that $\lambda(w, b_{k-1}) = \lambda(u, b_{k-1})$. Then, since $\lambda(u, w) = t < \lambda(w, b_{k-1}) = \lambda(u, b_{k-1})$, it follows that $w b_{k-1} \, \Lambda \, u b_{k-1}$. If $\{v, b_{k-1}\} \notin E$ then $x_{u b_{k-1}} = x_{w b_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{wv}$, which proves the lemma. Now let $\{v, b_{k-1}\} \in E$. If $\lambda(v, b_{k-1}) \leq \lambda(w, b_{k-1}) - 1$ then $x_{u b_{k-1}} = x_{w b_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{wv}$, which proves the lemma. If $\lambda(v, b_{k-1}) \geq$

$\lambda(w, b_{k-1}) + 1$ then $x_{ub_{k-1}} = x_{wb_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{vb_{k-1}} \Rightarrow_{\phi_{2\text{SAT}}} x_{wv}$, which proves the lemma. If $\lambda(v, b_{k-1}) = \lambda(w, b_{k-1})$ then $ub_{k-1}\ \Lambda\ vb_{k-1}$, and thus $x_{ub_{k-1}} = x_{a_{k-1}b_{k-1}} \not\Rightarrow_{\phi_{2\text{SAT}}} x_{a_k b_k} = x_{uv}$, which is a contradiction.

Case 2: $u = a_k \neq a_{k-1}$ **and** $v = b_k = b_{k-1}$**.** Then, since $x_{a_{k-1}b_{k-1}} = x_{a_{k-1}b_k} \Rightarrow_{\phi_{2\text{SAT}}} x_{a_k b_k} = x_{uv}$ and $\lambda(a_k, b_k) = t$, it follows that $\lambda(v, a_{k-1}) \leq t-1$. Suppose that $\{w, a_{k-1}\} \notin E$. Then $x_{a_{k-1}v} \Rightarrow_{\phi_{2\text{SAT}}} x_{wv}$, which proves the lemma. Now suppose that $\{w, a_{k-1}\} \in E$. If $\lambda(w, a_{k-1}) \leq t-1$ then $x_{a_{k-1}v} \Rightarrow_{\phi_{2\text{SAT}}} x_{wv}$, which proves the lemma. Suppose that $\lambda(w, a_{k-1}) = t$. Then, since $\lambda(v, a_{k-1}) \leq t-1$, it follows that $vw\ \Lambda\ a_{t-1}w$. If $\{u, a_{k-1}\} \notin E$ then also $a_{t-1}w\ \Lambda\ uw$, and thus $x_{wv} = x_{wu}$, which is a contradiction to the assumption that the triangle of u, v, w is uncorrelated. Thus $\{u, a_{k-1}\} \in E$. If $\lambda(u, a_{k-1}) \leq t-1$ then again $a_{k-1}w\ \Lambda\ uw$, which is a contradiction. On the other hand, if $\lambda(u, a_{k-1}) \geq t$ then $x_{a_{k-1}v} = x_{a_{k-1}b_{k-1}} \not\Rightarrow_{\phi_{2\text{SAT}}} x_{a_k b_k} = x_{uv}$, which is a contradiction.

Finally suppose that $\lambda(w, a_{k-1}) \geq t+1$. Then, since $\lambda(v, w) = t$ and $\lambda(v, a_{k-1}) \leq t-1$, it follows that $x_{vw} \Rightarrow_{\phi_{2\text{SAT}}} x_{a_{k-1}w} \Rightarrow_{\phi_{2\text{SAT}}} x_{a_{k-1}v}$. However, since $x_{a_{k-1}v} = x_{a_{k-1}b_k} \Rightarrow_{\phi_{2\text{SAT}}} x_{a_k b_k} = x_{uv}$, it follows that $x_{vw} \overset{*}{\Rightarrow}_{\phi_{2\text{SAT}}} x_{uv}$, which is a contradiction to the assumption that the triangle of u, v, w is uncorrelated. ☐

Lemma 13. *Let u, v, w induce a synchronous and $\phi^{(0)}_{2SAT}$-uncorrelated triangle in (G, λ), and let $\{a, b\} \in E$ be an edge of G such that $|\{a, b\} \cap \{u, v, w\}| \leq 1$. If $x_{ab} \overset{*}{\Rightarrow}_{\phi^{(0)}_{2SAT}} x_{uv}$, then x_{ab} also implies in $\phi^{(0)}_{2SAT}$ at least one of the four variables in the set $\{x_{vw}, x_{wv}, x_{uw}, x_{wu}\}$.*

Proof. Assume we have $|\{a, b\} \cap \{u, v, w\}| \leq 1$ and $x_{ab} \overset{*}{\Rightarrow}_{\phi^{(0)}_{2\text{SAT}}} x_{uv}$. Then we make a case distinction on the last implication in the implication chain $x_{ab} \overset{*}{\Rightarrow}_{\phi^{(0)}_{2\text{SAT}}} x_{uv}$.

1. The last implication is an implication from $\phi_{2\text{SAT}}$, i.e., $x_{ab} \overset{*}{\Rightarrow}_{\phi^{(0)}_{2\text{SAT}}} x_{pq} \Rightarrow_{\phi_{2\text{SAT}}} x_{uv}$. If $\{p, q\} \subseteq \{u, v, w\}$ then we are done, since we can assume that $\{p, q\} \neq \{u, v\}$ because no such implications are contained in $\phi_{2\text{SAT}}$. Otherwise Lemma 12 implies that x_{pq} also implies at least one of the four variables in the set $\{x_{vw}, x_{wv}, x_{uw}, x_{wu}\}$ in $\phi_{2\text{SAT}}$. If follows that x_{ab} also implies at least one of the four variables in the set $\{x_{vw}, x_{wv}, x_{uw}, x_{wu}\}$ in $\phi^{(0)}_{2\text{SAT}}$.
2. The last implication is *not* an implication from $\phi_{2\text{SAT}}$, i.e., $x_{ab} \overset{*}{\Rightarrow}_{\phi^{(0)}_{2\text{SAT}}} x_{pq} \Rightarrow_{\phi_{\text{INIT}}} x_{uv}$, there the implication $x_{pq} \Rightarrow_{\phi_{\text{INIT}}} x_{uv}$ was added to $\phi^{(0)}_{2\text{SAT}}$ by Initial-Forcing. If $x_{pq} \Rightarrow_{\phi_{\text{INIT}}} x_{uv}$ was added in Line 7 or Line 10 of Initial-Forcing, then we have that $\{p, q\} \subseteq \{u, v, w\}$ and $\{p, q\} \neq \{u, v\}$, hence the u, v, w is not a $\phi^{(0)}_{2\text{SAT}}$-uncorrelated triangle, a contradiction. If $x_{pq} \Rightarrow_{\phi_{\text{INIT}}} x_{uv}$ was added in Line 14 of Initial-Forcing, then we have that $x_{pq} \Rightarrow_{\phi_{\text{INIT}}} x_{uw}$, hence we are done. ☐

Detailed description of the algorithm. We are now ready to present our polynomial-time algorithm (Algorithm 4) for deciding whether a given temporal graph (G, λ) is temporally transitively orientable. The main idea of our algorithm is as follows. First, the algorithm computes all Λ-implication classes $A_1, \ldots, A_s$ by calling Algorithm 1 as a subroutine. If there exists at least one Λ-implication class A_i where $uv, vu \in A_i$ for some edge $\{u, v\} \in E$, then we announce that (G, λ) is a *no*-instance, due to Lemma 7. Otherwise we associate to each Λ-implication class A_i a variable x_i, and we build the 3NAE formula ϕ_{3NAE} and the 2SAT formula ϕ_{2SAT}, as described in Section 3.2.

In the *initialization phase* of Algorithm 4, we call algorithm Initial-Forcing (see Algorithm 2) as a subroutine. Starting from the formulas ϕ_{3NAE} and ϕ_{2SAT}, in Initial-Forcing we build the formulas $\phi^{(0)}_{\text{3NAE}}$ and $\phi^{(0)}_{\text{2SAT}}$ by both (i) checking which truth values are being *forced* in $\phi_{\text{3NAE}} \wedge \phi_{\text{2SAT}}$ (lines 2-10), and (ii) adding to ϕ_{2SAT} some clauses that are implicitly implied in $\phi_{\text{3NAE}} \wedge \phi_{\text{2SAT}}$ (lines 11-14). More specifically, Initial-Forcing proceeds as follows: (i) whenever setting $x_i = 1$ (resp. $x_i = 0$) forces $\phi_{\text{3NAE}} \wedge \phi_{\text{2SAT}}$ to become unsatisfiable, we choose to set $x_i = 0$ (resp. $x_i = 1$); (ii) if $x \Rightarrow_{\phi^{(0)}_{\text{2SAT}}} a$ and $x \Rightarrow_{\phi^{(0)}_{\text{2SAT}}} b$, and if we also have that $\texttt{NAE}(a, b, c) \in \phi^{(0)}_{\text{3NAE}}$, then we add $x \Rightarrow_{\phi^{(0)}_{\text{2SAT}}} \bar{c}$ to $\phi^{(0)}_{\text{2SAT}}$, since clearly, if $x = 1$ then $a = b = 1$ and we have to set $c = 0$ to satisfy the NAE clause $\texttt{NAE}(a, b, c)$. The next observation follows easily by Observation 10 and by the construction of $\phi^{(0)}_{\text{3NAE}}$ and $\phi^{(0)}_{\text{2SAT}}$ in Initial-Forcing.

Observation 14. *The temporal graph (G, λ) is transitively orientable if and only if $\phi^{(0)}_{3NAE} \wedge \phi^{(0)}_{2SAT}$ is satisfiable.*

The *main phase* of the algorithm starts once the formulas $\phi^{(0)}_{\text{3NAE}}$ and $\phi^{(0)}_{\text{2SAT}}$ have been computed. As we prove in Lemma 19, if the algorithm does not conclude at the initialization phase that the input instance is a *no*-instance, the the instance is a *yes*-instance. During any iteration $j \geq 1$ of the algorithm, we pick an arbitrary variable x_i and we assign it the truth value 1 (note that this is an arbitrary choice; we could equally choose to assign to x_i the value 0). Once we have set $x_i = 1$, we call algorithm Boolean-Forcing (see Algorithm 3) as a subroutine to check which implications this value of x_i has on the current formulas $\phi^{(j-1)}_{\text{3NAE}}$ and $\phi^{(j-1)}_{\text{2SAT}}$ and which other truth values of variables are forced. The correctness of Boolean-Forcing can be easily verified by checking all subcases of Boolean-Forcing. During such a call of Boolean-Forcing (i.e. during an iteration $j \geq 1$ in the main phase of the algorithm), we replace the current formulas by $\phi^{(j)}_{\text{3NAE}}$ and $\phi^{(j)}_{\text{2SAT}}$, respectively. Summarizing, in its initialization phase, the algorithm decides whether the input temporal graph can be transitively oriented (i.e. solves the decision version of the problem), while in its main phase it computes a temporally transitive orientation.

Correctness of the algorithm. We now formally prove that Algorithm 4 is correct. More specifically, we show that if Algorithm 4 gets a *yes*-instance

Algorithm 2 INITIAL-FORCING

Input: A 2-SAT formula ϕ_{2SAT} and a 3-NAE formula ϕ_{3NAE}

Output: A 2-SAT formula $\phi^{(0)}_{\text{2SAT}}$ and a 3-NAE formula $\phi^{(0)}_{\text{3NAE}}$ such that $\phi^{(0)}_{\text{2SAT}} \wedge \phi^{(0)}_{\text{3NAE}}$ is satisfiable if and only if $\phi_{\text{2SAT}} \wedge \phi_{\text{3NAE}}$ is satisfiable, or the announcement that $\phi_{\text{2SAT}} \wedge \phi_{\text{3NAE}}$ is not satisfiable.

1: $\phi^{(0)}_{\text{3NAE}} \leftarrow \phi_{\text{3NAE}}$; $\phi^{(0)}_{\text{2SAT}} \leftarrow \phi_{\text{2SAT}}$ {initialization}
2: **for** every variable x_i appearing in $\phi^{(0)}_{\text{3NAE}} \wedge \phi^{(0)}_{\text{2SAT}}$ **do**
3:   **if** BOOLEAN-FORCING $\left(\phi^{(0)}_{\text{3NAE}}, \phi^{(0)}_{\text{2SAT}}, x_i, 1\right)$ = "NO" **then**
4:     **if** BOOLEAN-FORCING $\left(\phi^{(0)}_{\text{3NAE}}, \phi^{(0)}_{\text{2SAT}}, x_i, 0\right)$ = "NO" **then**
5:       **return** "NO" {both $x_i = 1$ and $x_i = 0$ invalidate the formulas}
6:     **else**
7:       $\left(\phi^{(0)}_{\text{3NAE}}, \phi^{(0)}_{\text{2SAT}}\right) \leftarrow$ BOOLEAN-FORCING $\left(\phi^{(0)}_{\text{3NAE}}, \phi^{(0)}_{\text{2SAT}}, x_i, 0\right)$
8:   **else**
9:     **if** BOOLEAN-FORCING $\left(\phi^{(0)}_{\text{3NAE}}, \phi^{(0)}_{\text{2SAT}}, x_i, 0\right)$ = "NO" **then**
10:       $\left(\phi^{(0)}_{\text{3NAE}}, \phi^{(0)}_{\text{2SAT}}\right) \leftarrow$ BOOLEAN-FORCING $\left(\phi^{(0)}_{\text{3NAE}}, \phi^{(0)}_{\text{2SAT}}, x_i, 1\right)$
11: **for** every clause $\texttt{NAE}(x_{uv}, x_{vw}, x_{wu})$ of $\phi^{(0)}_{\text{3NAE}}$ **do**
12:   **for** every variable x_{ab} **do**
13:     **if** $x_{ab} \overset{*}{\Rightarrow}_{\phi^{(0)}_{\text{2SAT}}} x_{uv}$ and $x_{ab} \overset{*}{\Rightarrow}_{\phi^{(0)}_{\text{2SAT}}} x_{vw}$ **then** {add $(x_{ab} \Rightarrow x_{uw})$ to $\phi^{(0)}_{\text{2SAT}}$}
14:       $\phi^{(0)}_{\text{2SAT}} \leftarrow \phi^{(0)}_{\text{2SAT}} \wedge (x_{ba} \vee x_{uw})$
15: Repeat Lines 2 and 11 until no changes occur on $\phi^{(0)}_{\text{2SAT}}$ and $\phi^{(0)}_{\text{3NAE}}$
16: **return** $\left(\phi^{(0)}_{\text{3NAE}}, \phi^{(0)}_{\text{2SAT}}\right)$

as input then it outputs a temporally transitive orientation, while if it gets a *no*-instance as input then it outputs "NO". The *main result* of this section is Theorem 20, in which we prove that TEMPORAL TRANSITIVE ORIENTATION (TTO) is correct and runs in polynomial time.

The next crucial observation follows immediately by the construction of ϕ_{3NAE} in Section 3.2, and by the fact that, at every iteration j, Algorithm 4 can only remove clauses from $\phi^{(j-1)}_{\text{3NAE}}$.

Observation 15. *When* BOOLEAN-FORCING *(Algorithm 3) removes a clause from $\phi^{(j-1)}_{3NAE}$, then this clause is satisfied by all satisfying assignments of $\phi^{(j)}_{2SAT}$.*

Next, we prove a crucial and involved technical lemma about the Boolean forcing steps of Algorithm 4. This lemma will allow us to deduce that, during the *main phase* of Algorithm 4, whenever a new clause is added to the 2SAT part of the formula, this happens only in lines 17 and 19 of BOOLEAN-FORCING

Algorithm 3 BOOLEAN-FORCING

Input: A 2-SAT formula ϕ_2, a 3-NAE formula ϕ_3, and a variable x_i of $\phi_2 \wedge \phi_3$, and a truth value $\text{VALUE} \in \{0,1\}$

Output: A 2-SAT formula ϕ_2' and a 3-NAE formula ϕ_3', obtained from ϕ_2 and ϕ_3 by setting $x_i = \text{VALUE}$, or the announcement that $x_i = \text{VALUE}$ does not satisfy $\phi_2 \wedge \phi_3$.

1: Let a and b be such that $x_{ab} = x_i$; $\quad x_{ab} \leftarrow \text{VALUE}$

2: $\phi_2' \leftarrow \phi_2$; $\quad \phi_3' \leftarrow \phi_3$

3: **while** ϕ_2' has a clause $(x_{uv} \vee x_{pq})$ and $x_{uv} = 1$ **do**
4:     Remove the clause $(x_{uv} \vee x_{pq})$ from ϕ_2'

5: **while** ϕ_2' has a clause $(x_{uv} \vee x_{pq})$ and $x_{uv} = 0$ **do**
6:     **if** $x_{pq} = 0$ **then return** "NO"
7:     Remove the clause $(x_{uv} \vee x_{pq})$ from ϕ_2'
8:     $x_{pq} \leftarrow 1$; Repeat lines 3 and 5 until no changes occur in ϕ_2'. {Implement all changes to ϕ_2' that are implied by setting $x_{pq} = 1$}

9: **for** every clause $\texttt{NAE}(x_{uv}, x_{vw}, x_{wu})$ of ϕ_3' **do** {synchronous triangle on vertices u, v, w}
10:     **if** $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ **then** {add $(x_{uv} \Rightarrow x_{uw}) \wedge (x_{uw} \Rightarrow x_{vw})$ to ϕ_2'}
11:         $\phi_2' \leftarrow \phi_2' \wedge (x_{vu} \vee x_{uw}) \wedge (x_{wu} \vee x_{vw})$
12:         Remove the clause $\texttt{NAE}(x_{uv}, x_{vw}, x_{wu})$ from ϕ_3'

13:     **if** x_{uv} already got the value 1 or 0 **then**
14:         Remove the clause $\texttt{NAE}(x_{uv}, x_{vw}, x_{wu})$ from ϕ_3'

15:         **if** x_{vw} and x_{wu} do not have yet a truth value **then**

16:             **if** $x_{uv} = 1$ **then** {add $(x_{vw} \Rightarrow x_{uw})$ to ϕ_2'}
17:                 $\phi_2' \leftarrow \phi_2' \wedge (x_{wv} \vee x_{uw})$

18:             **else** {$x_{uv} = 0$; in this case add $(x_{uw} \Rightarrow x_{vw})$ to ϕ_2'}
19:                 $\phi_2' \leftarrow \phi_2' \wedge (x_{wu} \vee x_{vw})$

20:         **if** $x_{vw} = x_{uv}$ and x_{wu} does not have yet a truth value **then**
21:             $x_{wu} \leftarrow 1 - x_{uv}$; Repeat lines 3 and 5 until no changes occur in ϕ_2'. {Implement all changes to ϕ_2' that are implied by setting $x_{wu} = 1 - x_{uv}$}

22:         **if** $x_{vw} = x_{wu} = x_{uv}$ **then return** "NO"

23: Repeat lines 3, 5, and 9 until no changes occur in ϕ_2' and ϕ_3'.

24: **return** (ϕ_2', ϕ_3')

(Algorithm 3). That is, whenever a new clause is added to the 2SAT part of the formula in line 11 of Algorithm 3, this can only happen during the *initialization phase* of Algorithm 4.

Lemma 16. *Consider an execution of* BOOLEAN-FORCING *(Algorithm 3) which is called in an iteration $j \geq 1$ (i.e. in the main phase) of Algorithm 4. Then Lines 11 and 12 of* BOOLEAN-FORCING *are not executed.*

Proof. Assume for contradiction that Lines 11 and 12 of Algorithm 3 are executed in iteration j of Algorithm 4. Let $j \geq 1$ be the first iteration where this happens. This means that there is a clause $\texttt{NAE}(x_{uv}, x_{vw}, x_{wu})$ of ϕ_3' and an implication $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ during the execution of Algorithm 3.

We first partition the implication chain $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ into "old" and "new"

Algorithm 4 Temporal transitive orientation.

Input: A temporal graph (G, λ), where $G = (V, E)$.
Output: A temporal transitive orientation F of (G, λ), or the announcement that (G, λ) is temporally not transitively orientable.

Execute Algorithm 1 to build the Λ-implication classes $\{A_1, A_2, \ldots, A_s\}$ and the Boolean variables $\{x_{uv}, x_{vu} : \{u, v\} \in E\}$
if Algorithm 1 returns "NO" **then return** "NO"

Build the 3NAE formula $\phi_{3\text{NAE}}$ and the 2SAT formula $\phi_{2\text{SAT}}$

if Initial-Forcing $(\phi_{3\text{NAE}}, \phi_{2\text{SAT}}) \neq$ "NO" **then** {Initialization phase}

  $\left(\phi^{(0)}_{3\text{NAE}}, \phi^{(0)}_{2\text{SAT}}\right) \leftarrow$ Initial-Forcing $(\phi_{3\text{NAE}}, \phi_{2\text{SAT}})$

else $\{\phi_{3\text{NAE}} \wedge \phi_{2\text{SAT}}$ leads to a contradiction$\}$
  return "NO"

$j \leftarrow 1$; $F \leftarrow \emptyset$ {Main phase}
while a variable x_i appearing in $\phi^{(j-1)}_{3\text{NAE}} \wedge \phi^{(j-1)}_{2\text{SAT}}$ did not yet receive a truth value **do** {arbitrary choice of x_i}

  $\left(\phi^{(j)}_{3\text{NAE}}, \phi^{(j)}_{2\text{SAT}}\right) \leftarrow$ Boolean-Forcing $\left(\phi^{(j-1)}_{3\text{NAE}}, \phi^{(j-1)}_{2\text{SAT}}, x_i, 1\right)$
  $j \leftarrow j + 1$

for $i = 1$ to s **do**
  if x_i did not yet receive a truth value **then** $x_i \leftarrow 1$
  if $x_i = 1$ **then** $F \leftarrow F \cup A_i$ **else** $F \leftarrow F \cup \overline{A_i}$

return the temporally transitive orientation F of (G, λ)

implications, where "old" implications are contained in $\phi^{(0)}_{2\text{SAT}}$ and all other implications (that were added in the previous iterations $1, 2, \ldots, j-1$) are considered "new". For simplicity of notation, we will refer to these "new" implications using the symbol "$\Rightarrow_{\text{BF}}$". Recall here that, whenever $x_{ab} \Rightarrow_{\text{BF}} x_{cd}$, we have that $\lambda(a, b) = \lambda(c, d)$ by Boolean-Forcing. If there are several NAE clauses and implication chains that fulfill the condition in Line 10 of Algorithm 3, we assume that $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ is one that contains a minimum number of "new" implications. Observe that, since we assume $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ is a condition for the first execution of Lines 11 and 12 of Algorithm 3, it follows that all "new" implications in $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ were added in Line 17 or Line 19 of Boolean-Forcing (i.e. Algorithm 3) in previous iterations.

Assume that $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ contains only "old" implications. Then, this execution of Lines 11 and 12 of Algorithm 3 happens during the initialization phase of Algorithm 4. This is a contradiction to the assumption that this execution of Lines 11 and 12 of Algorithm 3 happens at iteration $j \geq 1$ of Algorithm 4. Therefore $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ contains at least one "new" implication. We now distinguish the cases where $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ contains "old" implications or not.

Case I: $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ contains at least one "old" implication. We assume without loss of generality that $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vw}$ contains an "old" implication that is directly followed by a "new" implication (if this is not the case, then we can consider the contraposition of the implication chain).

Note that, since the "new" implication was added in Line 17 or Line 19 of Algorithm 3, we can assume without loss of generality that the "new" implication is $x_{ab} \Rightarrow_{\mathrm{BF}} x_{cb}$ and that $x_{ca} = 1$ for some synchronous triangle on the vertices a, b, c (this is the Line 17 case, Line 19 works analogously). That is, we have $\texttt{NAE}(x_{ab}, x_{bc}, x_{ca}) \in \phi^{(0)}_{\mathrm{3NAE}}$. Let $x_{pq} \Rightarrow_{\phi^{(0)}_{\mathrm{2SAT}}} x_{ab}$ be the "old" implication. Then we have that $x_{pq} \Rightarrow_{\phi^{(0)}_{\mathrm{2SAT}}} x_{ab} \Rightarrow_{\mathrm{BF}} x_{cb}$ is contained in $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vw}$. Furthermore, by definition of $\phi^{(0)}_{\mathrm{2SAT}}$, we have that $|\{p, q\} \cap \{a, b, c\}| \leq 1$, hence we can apply Lemma 13 and obtain one of the following four scenarios:

1. $x_{pq} \overset{*}{\Rightarrow}_{\phi^{(0)}_{\mathrm{2SAT}}} x_{cb}$:
   In this case we can replace $x_{pq} \Rightarrow_{\phi^{(0)}_{\mathrm{2SAT}}} x_{ab} \Rightarrow_{\mathrm{BF}} x_{cb}$ with $x_{pq} \Rightarrow_{\phi^{(0)}_{\mathrm{2SAT}}} x_{cb}$ in the implication chain $x_{uv} \overset{*}{\Rightarrow}_{\phi^{(j)}_{\mathrm{2SAT}}} x_{vw}$ to obtain an implication chain from x_{uv} to x_{vw} with strictly fewer "new" implications, a contradiction.
2. $x_{pq} \overset{*}{\Rightarrow}_{\phi^{(0)}_{\mathrm{2SAT}}} x_{bc}$:
   Now we have that $x_{pq} \Rightarrow_{\phi^{(0)}_{\mathrm{2SAT}}} x_{ab}$ and $x_{pq} \overset{*}{\Rightarrow}_{\phi^{(0)}_{\mathrm{2SAT}}} x_{bc}$. Then by definition of $\phi^{(0)}_{\mathrm{2SAT}}$ we also have that $x_{pq} \Rightarrow_{\phi^{(0)}_{\mathrm{2SAT}}} x_{ac}$. Recall that we have set $x_{ca} = 1$, that is, $x_{ac} = 0$. Therefore, by Lines 8 and 21 of Boolean-Forcing, we have already set $x_{pq} = 0$, and therefore the implication $x_{pq} \Rightarrow_{\phi^{(0)}_{\mathrm{2SAT}}} x_{ab}$ does not exist in ϕ'_2 anymore, which is a contradiction.
3. $x_{pq} \overset{*}{\Rightarrow}_{\phi^{(0)}_{\mathrm{2SAT}}} x_{ca}$:
   Now we have that $x_{pq} \Rightarrow_{\phi^{(0)}_{\mathrm{2SAT}}} x_{ab}$ and $x_{pq} \overset{*}{\Rightarrow}_{\phi^{(0)}_{\mathrm{2SAT}}} x_{ca}$. Then by definition of $\phi^{(0)}_{\mathrm{2SAT}}$ we also have that $x_{pq} \Rightarrow_{\phi^{(0)}_{\mathrm{2SAT}}} x_{cb}$. From here it is the same as Case 1.
4. $x_{pq} \overset{*}{\Rightarrow}_{\phi^{(0)}_{\mathrm{2SAT}}} x_{ac}$: Same as Case 2.

Hence, we have a contradiction in every case and can conclude that $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vw}$ does not contain any "old" implications.

Case II: $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vw}$ contains only "new" implications. To analyze this case, we first introduce the notion of *alternating* and *non-alternating* sequences of "new" implications, as follows. Whenever the sequence $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vw}$ contains at least one pair of consecutive direct implications of the form $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ac} \Rightarrow_{\mathrm{BF}} x_{ad}$ (see Figure 4(a)), or of the form $x_{ba} \Rightarrow_{\mathrm{BF}} x_{ca} \Rightarrow_{\mathrm{BF}} x_{da}$ (see Figure 4(b)), we call $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vw}$ a *non-alternating* sequence of implications; otherwise we call it *alternating* (see Figure 4(c)). That is, if $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vw}$ is *alternating*, then it either has the form

$$x_{uv} = x_{u_1 v_1} \Rightarrow_{\mathrm{BF}} x_{u_2 v_1} \Rightarrow_{\mathrm{BF}} x_{u_2 v_2} \Rightarrow_{\mathrm{BF}} x_{u_3 v_2} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{u_j v_i} = x_{vw}, \tag{3}$$

or it has the form

$$x_{uv} = x_{u_1 v_1} \Rightarrow_{\mathrm{BF}} x_{u_1 v_2} \Rightarrow_{\mathrm{BF}} x_{u_2 v_2} \Rightarrow_{\mathrm{BF}} x_{u_2 v_3} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{u_i v_j} = x_{vw}, \tag{4}$$

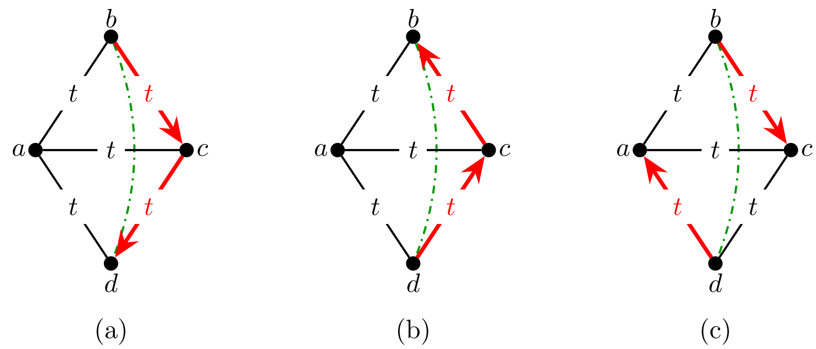


Figure 4: Illustration of alternating and non-alternating sequences of implications that can occur at the two synchronous triangles on the vertices $\{a,b,c\}$ and $\{a,c,d\}$. The red directed edges illustrate variables that have already been set to 1 by the algorithm Boolean-Forcing. Figure (a): *non-alternating* implications $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ac} \Rightarrow_{\mathrm{BF}} x_{ad}$, which occur whenever $x_{bc} = x_{cd} = 1$ (red edges). Figure (b): *non-alternating* implications $x_{ba} \Rightarrow_{\mathrm{BF}} x_{ca} \Rightarrow_{\mathrm{BF}} x_{da}$, which occur whenever $x_{cb} = x_{dc} = 1$ (red edges). Figure (c): *alternating* implications $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ac} \Rightarrow_{\mathrm{BF}} x_{dc}$, which occur whenever $x_{bc} = x_{da} = 1$ (red edges). In all three figures, the green dash-dotted line indicates that edge $\{a,d\}$ may exist (with some time label) or may not exist.

where either $j = i$ or $j = i+1$. Figure 4 illustrates some examples of alternating and non-alternating sequences of implications.

We now distinguish the cases where $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vw}$ is an alternating or a non-alternating sequence of implications. Note that, as all these are "new" implications, all edges which are involved in $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vw}$ have the same label t. That is, for every variable x_{ab} that appears in the sequence $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vw}$ of implications, we have that $\lambda(a,b) = t$.

Case II-A: $x_{uv} \overset{*}{\Rightarrow}_{\mathbf{BF}} x_{vw}$ is a non-alternating sequence of implications. Without loss of generality, let this sequence $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vw}$ contain the pair of consecutive direct implications $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ac} \Rightarrow_{\mathrm{BF}} x_{ad}$ (the case where it contains the implications $x_{ba} \Rightarrow_{\mathrm{BF}} x_{ca} \Rightarrow_{\mathrm{BF}} x_{da}$ can be treated in exactly the same way).

Let a, b, c be the vertices of the synchronous triangle that caused the implication $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ac}$, and let a', c', d be the vertices of the synchronous triangle that caused the implication $x_{ac} \Rightarrow_{\mathrm{BF}} x_{ad}$, where $x_{ac} = x_{a'c'}$ and $x_{ad} = x_{a'd}$. Then, Lemma 9 (the temporal triangle lemma) implies that the edges $\{a,d\}$ and $\{c,d\}$ exist in the graph and that ad (resp. cd) belongs to the same Λ-implication class with $a'd$ (resp. $c'd$). Therefore we can assume without loss of generality that $a = a'$ and $c = c'$.

Then, since $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ac}$ and $x_{ac} \Rightarrow_{\mathrm{BF}} x_{ad}$ are direct "new" implications, it follows that $x_{bc} = x_{cd} = 1$ (as these implications have only been added by Lines 17 or 19 of Boolean-Forcing).

Let $\{b,d\} \notin E$ or $\lambda(b,d) < t$. Then $\phi_{2\mathrm{SAT}}^{(0)}$ by definition contains $x_{ab} \Rightarrow_{\phi_{2\mathrm{SAT}}^{(0)}} x_{ad}$. Thus, we can replace within $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vw}$ the two "new" implications $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ac} \Rightarrow_{\mathrm{BF}} x_{ad}$ by the "old" implication $x_{ab} \Rightarrow_{\phi_{2\mathrm{SAT}}^{(0)}} x_{ad}$, thus resulting to a a sequence of implications from x_{uv} to x_{vw} that has fewer "new" implications, a contradiction to our assumption.

Let $\lambda(b,d) > t$. Then $\phi_{2\mathrm{SAT}}^{(0)}$ by definition contains $x_{cd} \Rightarrow_{\phi_{2\mathrm{SAT}}^{(0)}} x_{bd}$ and $x_{bd} \Rightarrow_{\phi_{2\mathrm{SAT}}^{(0)}} x_{ba}$. Thus, since $x_{cd} = 1$, it follows Boolean-Forcing sets $x_{ab} = 0$, which is a contradiction to the assumption that the implication $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ac}$ belongs to ϕ_2'.

Let now $\lambda(b,d) = t$. Then $\mathtt{NAE}(x_{bc}, x_{cd}, x_{db}) \in \phi_{3\mathrm{NAE}}^{(0)}$. If x_{bc} is set to 1 before x_{cd} is set to 1 (i.e. at an earlier iteration of Boolean-Forcing), then Boolean-Forcing adds (in Line 17) to ϕ_2' the direct implication $x_{cd} \Rightarrow_{\mathrm{BF}} x_{bd}$. In this case, when x_{cd} is set to 1 at a subsequent iteration of Boolean-Forcing, x_{bd} is also set to 1. Similarly, if x_{cd} is set to 1 before x_{bc} is set to 1, then Boolean-Forcing adds to ϕ_2' the direct implication $x_{db} \Rightarrow_{\mathrm{BF}} x_{cb}$, which is equivalent to $x_{bc} \Rightarrow_{\mathrm{BF}} x_{bd}$. In this case, when x_{bd} is set to 1 at a subsequent iteration of Boolean-Forcing, x_{bd} is also set to 1. Finally, if both x_{bc} and x_{cd} are set to 1 at the same iteration, Boolean-Forcing also sets x_{bd} to 1 in Line 21. Summarizing, in any case Boolean-Forcing always sets $x_{bd} = 1$, and thus it also adds to ϕ_2' the implication $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ad}$. Thus, we can replace within $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vw}$ the two implications $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ac} \Rightarrow_{\mathrm{BF}} x_{ad}$ by the single implication $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ad}$, thus resulting to a sequence of implications from x_{uv} to x_{vw} that has fewer "new" implications, a contradiction to our assumption.

Case II-B: $x_{uv} \overset{*}{\Rightarrow}_{\mathbf{BF}} x_{vw}$ is an alternating sequence of implications. Let this sequence be of the form of (3) where $j = i$ (the cases where $j = i+1$ or where the sequence is of the form of (4) can be treated analogously), that is,

$$x_{uv} = x_{u_1v_1} \Rightarrow_{\mathrm{BF}} x_{u_2v_1} \Rightarrow_{\mathrm{BF}} x_{u_2v_2} \Rightarrow_{\mathrm{BF}} x_{u_3v_2} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{u_iv_i} = x_{vw}, \tag{5}$$

Similarly to Case II-A, by iteratively applying Lemma 9 (the temporal triangle lemma), we may assume without loss of generality that all implications of (5) are added to ϕ_2' by the synchronous triangles on the vertices $\{u_1, v_1, u_2\}$, $\{v_1, u_2, v_2\}$, $\{u_2, v_2, u_3\}$, $\ldots$, $\{v_{i-1}, u_i, v_i\}$. Furthermore, as all the implications of (5) have been added to ϕ_2' by Boolean-Forcing, it follows that $x_{u_iu_{i-1}} = x_{u_{i-1}u_{i-2}} = \ldots = x_{u_2u_1} = 1$ and $x_{v_1v_2} = x_{v_2v_3} = \ldots = x_{v_{i-1}v_i} = 1$.

Now, since $x_{u_iv_i} = x_{vw}$ (i.e. u_iv_i belongs to the same Λ-implication class with vw), it follows by Lemma 9 (the temporal triangle lemma) that the edge $\{u_1, u_i\}$ exists in the graph and that u_1u_i belongs to the same Λ-implication class with $u_1v = uv$ (and thus, in particular, $\lambda(u_1, u_i) = \lambda(u_1, v) = t$).

Recall that $\lambda(u_1, u_2) = t$ and $x_{u_2u_1} = 1$. We now prove by induction that, for every $j = 3, \ldots, i$, we have $\lambda(u_1, u_j) \geq t$ and $x_{u_ju_1} = 1$.

For the induction basis, let $j = 3$. If $\{u_1, u_3\} \notin E$ or $\lambda(u_1, u_3) < t$, then $\phi_{2\mathrm{SAT}}^{(0)}$ by definition contains $x_{u_3u_2} \Rightarrow_{\phi_{2\mathrm{SAT}}^{(0)}} x_{u_1u_2}$. This is a contradiction, as $x_{u_3u_2} = x_{u_2u_1} = 1$. Therefore $\{u_1, u_3\} \in E$ and $\lambda(u_1, u_3) \geq t$. If $\lambda(u_1, u_3) = t$ then Boolean-Forcing sets $x_{u_3u_1} = 1$ (see Line 21 of Boolean-Forcing). If $\lambda(u_1u_3) > t$ then $\phi_{2\mathrm{SAT}}^{(0)}$ contains $x_{u_2u_1} \Rightarrow_{\phi_{2\mathrm{SAT}}^{(0)}} x_{u_3u_1}$. Therefore, since $x_{u_2u_1} = 1$, it follows in this case as well that Boolean-Forcing sets $x_{u_3u_1} = 1$. This completes the induction basis.

For the induction step, let $4 \leq j \leq i$, and assume by the induction hypothesis that $t' = \lambda(u_1, u_{j-1}) \geq t$ and $x_{u_{j-1}u_1} = 1$. Recall that $\lambda(u_{j-1}, u_j) = t$ and $x_{u_j u_{j-1}} = 1$. If $\{u_1, u_j\} \notin E$ or $\lambda(u_1, u_j) < t'$, then $\phi_{\text{2SAT}}^{(0)}$ by definition contains $x_{u_j u_{j-1}} \Rightarrow_{\phi_{\text{2SAT}}^{(0)}} x_{u_1 u_{j-1}}$. This is a contradiction, as $x_{u_j u_{j-1}} = x_{u_{j-1} u_1} = 1$. Therefore $\{u_1, u_j\} \in E$ and $\lambda(u_1, u_j) \geq t'$. If $\lambda(u_1, u_j) = t' = t$ then Boolean-Forcing sets $x_{u_j u_1} = 1$ (see Line 21 of Boolean-Forcing). If $\lambda(u_1, u_j) = t' > t$ or if $\lambda(u_1, u_j) > t' \geq t$ then $\phi_{\text{2SAT}}^{(0)}$ contains $x_{u_{j-1} u_1} \Rightarrow_{\phi_{\text{2SAT}}^{(0)}} x_{u_j u_1}$. Therefore, since $x_{u_{j-1} u_1} = 1$, it follows in this case as well that Boolean-Forcing sets $x_{u_j u_1} = 1$. This completes the induction step.

Therefore, in particular, for $j = i$ we have that $x_{u_i u_1} = 1$. Thus, since $u_1 u_i$ belongs to the same Λ-implication class with $u_1 v = uv$, it follows that $x_{uv} = 1$, which is a contradiction to the assumption that $x_{uv} \overset{*}{\Rightarrow}_{\text{BF}} x_{vw}$ is contained in ϕ_2'. This completes the proof. $\square$

In the next lemma we prove that, if Algorithm 4 does not return "NO" after the initialization phase (in Line 7), then the 2SAT formula $\phi_{\text{2SAT}}^{(0)}$ is satisfiable. Furthermore, as we prove in Lemma 18, in this case also the 2SAT formulas $\phi_{\text{2SAT}}^{(j)}$ are satisfiable for every $j \geq 1$.

Lemma 17. *Assume that Algorithm 4 does not return "NO" in the initialization phase (i.e. in Line 7). Then there exists no variable x_{uv} in $\phi_{2SAT}^{(0)}$ such that $x_{uv} \overset{*}{\Rightarrow}_{\phi_{2SAT}^{(0)}} x_{vu}$, and thus $\phi_{2SAT}^{(0)}$ is satisfiable.*

Proof. Since Algorithm 4 does not return "NO" in Line 7, it follows that Line 5 of Initial-Forcing (Algorithm 2) is not executed, when Initial-Forcing is called by Algorithm 4. Furthermore, before Initial-Forcing finishes, it checks in Line 15 whether any of the formulas $\phi_{\text{3NAE}}^{(0)}$ or $\phi_{\text{2SAT}}^{(0)}$ have been changed since the last iteration of Lines 2 and 11.

Let x_{uv} be an arbitrary variable in $\phi_{\text{2SAT}}^{(0)}$, i.e. in the 2SAT part of the formula after Initial-Forcing has finished. Since x_{uv} did not get a Boolean value during the execution of Initial-Forcing, it follows that, when Initial-Forcing stops, setting x_{uv} to 1 (resp. to 0) does not cause a contradiction. Indeed, otherwise Initial-Forcing would set x_{uv} equal to 0 (resp. 1). Therefore, once Initial-Forcing finishes, there cannot exist any variable x_{uv} in $\phi_{\text{2SAT}}^{(0)}$ such that $x_{uv} \overset{*}{\Rightarrow}_{\phi_{\text{2SAT}}^{(0)}} x_{vu}$ (as otherwise Initial-Forcing would set $x_{uv} = 0$). This completes the lemma. $\square$

Lemma 18. *Assume that Algorithm 4 does not return "NO" in the initialization phase (i.e. in Line 7). Then, at any point during an arbitrary call of* Boolean-Forcing *at the iteration $j \geq 1$ of Algorithm 4, there does not exist any variable x_{uv} in ϕ_2' such that $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vu}$, and thus ϕ_2' is satisfiable.*

Proof. Let $j = 1$. At the very beginning of iteration $j = 1$ (where no changes have been made to ϕ_2' by Boolean-Forcing) it follows immediately by Lemma 17 that there is no variable x_{uv} in $\phi_2' = \phi_{\text{2SAT}}^{(0)}$ such that $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vu}$.

Now, let $j \geq 1$. Assume that, at the very beginning of iteration j, there is no variable x_{uv} in $\phi'_2 = \phi^{(j-1)}_{2\text{SAT}}$ such that $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$. For the sake of contradiction, assume that, at some point during the execution of this call of Boolean-Forcing, there exists a variable x_{uv} in ϕ'_2 such that $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$. Assume that this is the earliest point during the execution of this call of Boolean-Forcing where such an implication chain $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$ exists in ϕ'_2. Furthermore, among all implication chains $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$, consider one that has the smallest number of "new" implications.

Similarly to the proof of Lemma 16, we partition the implication chain $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$ into "old" implications (which are also present in $\phi^{(0)}_{2\text{SAT}}$) and "new" implications (which were added by Boolean-Forcing during some iteration $j' \in \{1, 2, \ldots, j\}$). Similarly to Lemma 16, for simplicity of notation we refer to these "new" implications using the symbol "$\Rightarrow_{\text{BF}}$". Recall that, whenever $x_{ab} \Rightarrow_{\text{BF}} x_{cd}$, we have that $\lambda(a, b) = \lambda(c, d)$ by Boolean-Forcing. Note that $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$ contains at least one "new" implication, as otherwise $x_{uv} \overset{*}{\Rightarrow}_{\phi^{(0)}_{2\text{SAT}}} x_{vu}$ which is a contradiction by Lemma 17.

Case I: $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$ contains at least one "old" implication. Consider an "old" implication $x_{pq} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{ab}$ within the implication chain $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$, which is followed by a "new" implication (if there is no such pair of consecutive implications, then there is one in the contraposition of the implication chain). By Lemma 16, the "new" implication was added by Boolean-Forcing in Line 17 or Line 19. We can assume without loss of generality that the "new" implication is $x_{ab} \Rightarrow_{\text{BF}} x_{cb}$ and that $x_{ca} = 1$ for some synchronous triangle on the vertices a, b, c (this is the case of Line 17, Line 19 works analogously). That is, we have $\texttt{NAE}(x_{ab}, x_{bc}, x_{ca}) \in \phi^{(0)}_{3\text{NAE}}$. Summarizing, we have that $x_{pq} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{ab} \Rightarrow_{\text{BF}} x_{cb}$ is contained in $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$. Furthermore, by construction of $\phi^{(0)}_{2\text{SAT}}$, we have that $|\{p, q\} \cap \{a, b, c\}| \leq 1$, hence we can apply Lemma 13 and obtain one of the following four scenarios:

1. $x_{pq} \overset{*}{\Rightarrow}_{\phi^{(0)}_{2\text{SAT}}} x_{cb}$:
   In this case we can replace $x_{pq} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{ab} \Rightarrow_{\text{BF}} x_{cb}$ with $x_{pq} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{cb}$ in the implication chain $x_{uv} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$ to obtain an implication chain from x_{uv} to x_{vu} in ϕ'_2 with strictly fewer "new" implications, a contradiction.
2. $x_{pq} \overset{*}{\Rightarrow}_{\phi^{(0)}_{2\text{SAT}}} x_{bc}$:
   Now we have that $x_{pq} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{ab}$ and $x_{pq} \overset{*}{\Rightarrow}_{\phi^{(0)}_{2\text{SAT}}} x_{bc}$. Then by definition of $\phi^{(0)}_{2\text{SAT}}$ we also have that $x_{pq} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{ac}$. Recall that we have set $x_{ca} = 1$ (which triggered the addition of the implication $x_{ab} \Rightarrow_{\text{BF}} x_{cb}$), that is, $x_{ac} = 0$. Therefore, by Lines 8 and 21 of Boolean-Forcing, we have already set $x_{qp} = 1$, i.e. $x_{pq} = 0$, and therefore the implication $x_{pq} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{ab}$ does not exist in ϕ'_2 anymore, which is a contradiction.
3. $x_{pq} \overset{*}{\Rightarrow}_{\phi^{(0)}_{2\text{SAT}}} x_{ca}$:

Now we have that $x_{pq} \Rightarrow_{\phi_{2\mathrm{SAT}}^{(0)}} x_{ab}$ and $x_{pq} \overset{*}{\Rightarrow}_{\phi_{2\mathrm{SAT}}^{(0)}} x_{ca}$. Then by definition of $\phi_{2\mathrm{SAT}}^{(0)}$ we also have that $x_{pq} \Rightarrow_{\phi_{2\mathrm{SAT}}^{(0)}} x_{cb}$. From here it is the same as Case 1.

4. $x_{pq} \overset{*}{\Rightarrow}_{\phi_{2\mathrm{SAT}}^{(0)}} x_{ac}$: Same as Case 2.

Case II: $x_{uv} \overset{*}{\Rightarrow}_{\phi_2'} x_{vu}$ contains only "new" implications. Similarly to Case II of the proof of Lemma 16, we use the notion of *alternating* and *non-alternating* sequences of "new" implications. In a nutshell, whenever the sequence $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vu}$ contains at least one pair of consecutive direct implications of the form $x_{ab} \Rightarrow_{\mathrm{BF}} x_{ac} \Rightarrow_{\mathrm{BF}} x_{ad}$, or of the form $x_{ba} \Rightarrow_{\mathrm{BF}} x_{ca} \Rightarrow_{\mathrm{BF}} x_{da}$, the sequence of implications $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vu}$ is called *non-alternating*; otherwise it is called *alternating*. That is, if $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vu}$ is *alternating*, then it either has the form

$$x_{uv} = x_{u_1v_1} \Rightarrow_{\mathrm{BF}} x_{u_2v_1} \Rightarrow_{\mathrm{BF}} x_{u_2v_2} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vu} = x_{v_1u_1}, \tag{6}$$

or it has the form

$$x_{uv} = x_{u_1v_1} \Rightarrow_{\mathrm{BF}} x_{u_1v_2} \Rightarrow_{\mathrm{BF}} x_{u_2v_2} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vu} = x_{v_1u_1}. \tag{7}$$

We now distinguish the cases where $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vw}$ is an alternating or a non-alternating sequence of implications. Note that, as all these are "new" implications, all edges which are involved in $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vu}$ have the same label t. That is, for every variable x_{ab} that appears in the sequence $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vu}$ of implications, we have that $\lambda(a, b) = t$.

Case II-A: $x_{uv} \overset{*}{\Rightarrow}_{\mathbf{BF}} x_{vu}$ is a non-alternating sequence of implications. This case can be treated in exactly the same way as Case II-A in the proof of Lemma 16, where we just replace "x_{vw}" with "x_{vu}". The main idea of the proof is that, if $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vu}$ is non-alternating, then there exists an implication sequence that contains fewer "new" implications, which is a contradiction.

Case II-B: $x_{uv} \overset{*}{\Rightarrow}_{\mathbf{BF}} x_{vu}$ is an alternating sequence of implications. First let this sequence be of the form of (6). As the implication $x_{u_1v_1} \Rightarrow_{\mathrm{BF}} x_{u_2v_1}$ of (6) has been added to ϕ_2' by Boolean-Forcing, it follows that $x_{u_2u_1} = 1$ and $\lambda(u_1, u_2) = t$. That is, there is a synchronous triangle on the vertices $\{u_1, v_1, u_2\}$, and we have the implication sequence $x_{u_2v_1} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{v_1u_1}$. Therefore, Lines 11 and 12 of Boolean-Forcing are executed during some iteration $j \geq 1$ (i.e. in the main phase) of Algorithm 4, which is a contradiction by Lemma 16.

Now let the sequence $x_{uv} \overset{*}{\Rightarrow}_{\mathrm{BF}} x_{vu}$ be of the form of (7). Similarly to Case II-A in the proof of Lemma 16, by iteratively applying Lemma 9 (the temporal triangle lemma), we may assume without loss of generality that the first two implications of (7) are added to ϕ_2' by the synchronous triangles on the vertices $\{u_1, v_1, v_2\}$ and $\{u_1, v_2, u_2\}$. Furthermore, as the implications $x_{u_1v_1} \Rightarrow_{\mathrm{BF}} x_{u_1v_2}$ and $x_{u_1v_2} \Rightarrow_{\mathrm{BF}} x_{u_2v_2}$ of (7) have been added to ϕ_2' by Boolean-Forcing, it follows that $x_{u_2u_1} = 1$ and $x_{v_1v_2} = 1$.

Assume that $\{u_2, v_1\} \notin E$ or $\lambda(u_2, v_1) < t$. Then $\phi^{(0)}_{2\text{SAT}}$ by definition contains $x_{u_2u_1} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{v_1u_1}$. Thus, since $x_{u_2u_1} = 1$, it follows Boolean-Forcing sets $x_{v_1u_1} = 1$, which is a contradiction to the assumption that the implication $x_{u_1v_1} \Rightarrow_{\text{BF}} x_{u_1v_2}$ belongs to ϕ'_2.

Assume that $\lambda(u_2, v_1) > t$. Then $\phi^{(0)}_{2\text{SAT}}$ by definition contains $x_{u_1v_1} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{u_2v_1}$ and $x_{u_2v_1} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{u_2v_2}$, while both these implications are "old" (as these are implications that involve non-synchronous edges). Therefore there exists the sequence of implications $x_{uv} = x_{u_1v_1} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{u_2v_1} \Rightarrow_{\phi^{(0)}_{2\text{SAT}}} x_{u_2v_2} \overset{*}{\Rightarrow}_{\text{BF}} x_{vu} = x_{v_1u_1}$, which contains fewer "new" implications, a contradiction.

Finally assume that $\lambda(u_2, v_1) = t$. Then, since $x_{u_2u_1} = 1$ and $x_{v_1v_2} = 1$ and the triangles on the vertices $\{u_1, v_1, u_2\}$ and $\{v_1, u_2, v_2\}$ are synchronous, it follows that we also have the implications $x_{u_1v_1} \Rightarrow_{\text{BF}} x_{u_2v_1} \Rightarrow_{\text{BF}} x_{u_2v_2}$. Therefore, additionally to (7), also (6) is a sequence of (equally many) "new" implications from x_{uv} to x_{vu}, and thus a contradiction is implied as explained above. This completes the proof. □

In the next lemma we prove a strong structural property of our algorithm. Given this property, we will be able to show that, if the algorithm does not return "NO" during the initialization phase, then the instance is actually a *yes*-instance. That is, during all the subsequent iterations $j \geq 1$, the algorithm only constructs a valid transitive orientation, while the decision variant of the problem can simply be answered at the end of the initialization phase.

Lemma 19. *For every iteration $j \geq 1$ of Algorithm 4, every call of* Boolean-Forcing *(in Line 10 of Algorithm 4) does* not *return "NO".*

Proof. Boolean-Forcing can possibly return "NO" either in Lines 5-7 or in Line 22. First note that, for every call of Boolean-Forcing in Algorithm 4, there is a variable x_{ab} which is set to 1 (in Line 10 of Algorithm 4).

Assume that Boolean-Forcing returns "NO" in Lines 5-7. Let $(x_{uv} \vee x_{pq})$ be the clause of ϕ'_2 which is considered in Line 5 of Boolean-Forcing. As all forcings during the execution of Boolean-Forcing are made by assuming that a specific variable $x_{ab} = 1$, we have the following:

- $x_{ab} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu}$ (as $x_{uv} = 0$ in Line 5 of Boolean-Forcing)
- $x_{ab} \overset{*}{\Rightarrow}_{\phi'_2} x_{qp}$ (as $x_{pq} = 0$ in Line 6 of Boolean-Forcing)
- $x_{vu} \Rightarrow_{\phi'_2} x_{pq}$ (due to the existence of the clause $(x_{uv} \vee x_{pq})$ in ϕ'_2)

From the above implications we have that

$$x_{ab} \overset{*}{\Rightarrow}_{\phi'_2} x_{vu} \Rightarrow_{\phi'_2} x_{pq} \overset{*}{\Rightarrow}_{\phi'_2} x_{ba},$$

which is a contradiction by Lemma 18.

Assume that Boolean-Forcing returns "NO" in Line 22. Then, there exists a clause $\text{NAE}(x_{uv}, x_{vw}, x_{wu})$ in $\phi^{(0)}_{3\text{NAE}}$ such that, during the execution of iteration j of Algorithm 4, we are forced to set each of the variables x_{uv}, x_{vw}, x_{wu} to the same truth value, say without loss of generality, $x_{uv} = x_{vw} = x_{wu} = 1$. Furthermore assume without loss of generality that, among these three variables, x_{uv} is the first one that is set to 1 by BOOLEAN-FORCING.

Let x_{uv} be set to 1 at an earlier iteration of BOOLEAN-FORCING than x_{vw} and x_{wu}. Then BOOLEAN-FORCING adds (in Line 17) to ϕ_2' the clause $(x_{wv} \vee x_{uw})$. In this case, when x_{vw} (resp. x_{wu}) is set to 1 at a subsequent iteration of BOOLEAN-FORCING, x_{uw} (resp. x_{wv}) is also set to 1 (in Lines 5-8 of BOOLEAN-FORCING). This is a contradiction to our assumption that BOOLEAN-FORCING sets $x_{uv} = x_{vw} = x_{wu} = 1$.

Let x_{uv} be set to 1 at the same iteration of BOOLEAN-FORCING as one of the variables x_{vw} or x_{wu}; say, without loss of generality, with x_{vw}. Then, as $x_{uv} = x_{vw} = 1$, BOOLEAN-FORCING sets $x_{wu} = 0$ (in Line 21). This is again a contradiction to our assumption that BOOLEAN-FORCING sets $x_{uv} = x_{vw} = x_{wu} = 1$. □

We are now ready to combine all the above technical results to obtain the main result of this section in the next theorem, regarding the correctness and the running time of Algorithm 4.

Theorem 20. *Algorithm 4 correctly solves* TTO *in polynomial time.*

Proof. First assume that Algorithm 4 returns "NO". Due to Lemma 19, this can only happen in Line 7 of Algorithm 4, which means that INITIAL-FORCING has found a contradiction in $\phi_{3\text{NAE}}^{(0)} \wedge \phi_{2\text{SAT}}^{(0)}$. Thus, clearly $\phi_{3\text{NAE}}^{(0)} \wedge \phi_{2\text{SAT}}^{(0)}$ is not satisfiable, i.e. (G, λ) is not transitively orientable.

Now assume that Algorithm 4 does not return "NO". Than, during its main phase, Algorithm 4 repeatedly calls BOOLEAN-FORCING, and it repeatedly removes clauses from $\phi_{3\text{NAE}}^{(0)}$, until they are all removed. By Observation 15, whenever such a clause is removed during the iteration $j \geq 1$ of Algorithm 4, this clause is satisfied by all satisfying assignments of $\phi_{2\text{SAT}}^{(j)}$, and thus it remains satisfied by the truth assignment that is eventually computed by Algorithm 4. Let $j_0 \geq 1$ be the iteration of Algorithm 4, after which all clauses of $\phi_{3\text{NAE}}^{(0)}$ have been removed. Then $\phi_{2\text{SAT}}^{(j_0)}$ is satisfiable by Lemma 18, and the subsequent calls of BOOLEAN-FORCING (in Line 10 of Algorithm 4) provide a satisfying assignment of $\phi_{2\text{SAT}}^{(j_0)}$.

Let $j_1 \geq j_0$ be the last iteration of Algorithm 4; note that $\phi_{3\text{NAE}}^{(j_1)} \wedge \phi_{2\text{SAT}}^{(j_1)}$ is empty. Then, in Line 13, the algorithm gives the arbitrary truth value $x_i = 1$ to every variable x_i which did not yet get any truth value yet. This is a correct decision as all these variables are not involved in any Boolean constraint of $\phi_{3\text{NAE}}^{(j_1)} \wedge \phi_{2\text{SAT}}^{(j_1)}$ (which is empty). Finally, the algorithm orients in Line 14 all edges of G according to the corresponding truth assignment. The returned orientation F of (G, λ) is temporally transitive as every variable was assigned a truth value according to the Boolean constraints throughout the execution of the algorithm.

Summarizing, the truth assignment produced by Algorithm 4 satisfies $\phi_{3\mathrm{NAE}}^{(0)} \wedge \phi_{2\mathrm{SAT}}^{(0)}$, and thus the algorithm returns a valid temporally transitive orientation of the input temporal graph (G, λ). This completes the proof of correctness of Algorithm 4.

Lastly, we prove that Algorithm 4 runs in polynomial time. The Λ-implication classes of (G, λ) can be clearly computed by Algorithm 1 in polynomial time. Boolean-Forcing iteratively adds and removes clauses from the 2SAT formula ϕ_2', while it can only remove clauses from the 3NAE formula ϕ_3'. Whenever a clause is added to ϕ_2', a clause of ϕ_3' is removed. Therefore, as the initial 3NAE formula ϕ_3 has at most polynomially-many clauses, we can add clauses to ϕ_2' only polynomially-many times. In all remaining steps, Boolean-Forcing just checks clauses of ϕ_2' and ϕ_3' and it forces certain truth values to variables, and thus the total running time of Boolean-Forcing is polynomial. Furthermore, in Initial-Forcing and Algorithm 4 (the main algorithm) the Boolean-Forcing subroutine is only invoked at most four times for every variable in $\phi_{3\mathrm{NAE}}^{(0)} \wedge \phi_{2\mathrm{SAT}}^{(0)}$. Hence, we have an overall polynomial running time. □

4. Temporal Transitive Completion

We now study the computational complexity of Temporal Transitive Completion (TTC). In the static case, the so-called *minimum comparability completion* problem, i.e. adding the smallest number of edges to a static graph to turn it into a comparability graph, is known to be NP-hard [26]. Note that minimum comparability completion on static graphs is a special case of TTC and thus it follows that TTC is NP-hard too. Our other variants, however, do not generalize static comparability completion in such a straightforward way. Note that for Strict TTC we have that the corresponding recognition problem Strict TTO is NP-complete (Theorem 3), hence it follows directly that Strict TTC is NP-hard. For the remaining two variants of our problem, we show in the following that they are also NP-hard, giving the result that all four variants of TTC are NP-hard. Furthermore, we present a polynomial-time algorithm for all four problem variants for the case that all edges of underlying graph are oriented, see Theorem 22. This allows directly to derive an FPT algorithm for the number of unoriented edges as a parameter.

Theorem 21. *All four variants of* TTC *are NP-hard, even when the input temporal graph is completely unoriented.*

Proof. We give a reduction from the NP-hard Max-2-Sat problem [24].

Max-2-Sat

Input:	A boolean formula ϕ in implicative normal form[8] and an integer k.
Question:	Is there an assignment of the variables which satisfies at least k clauses in ϕ?

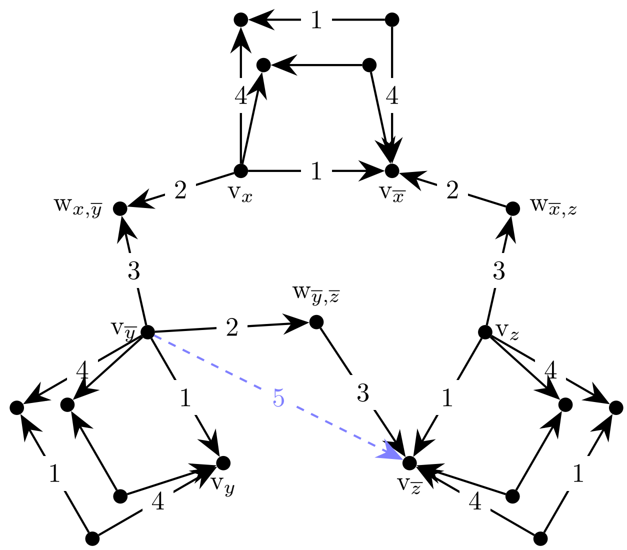


Figure 5: Temporal graph constructed from the formula $(x \Rightarrow \overline{y}) \wedge (\overline{x} \Rightarrow z) \wedge (\overline{y} \Rightarrow \overline{z})$ for $k = 1$ with orientation corresponding to the assignment $x = \texttt{true}$, $y = \texttt{false}$, $z = \texttt{true}$. Since this assignment does not satisfy the third clause, the dashed blue edge is required to make the graph temporally transitive.

We only describe the reduction from Max-2-Sat to TTC. However, the same construction can be used to show NP-hardness of the other variants.

Let (ϕ, k) be an instance of Max-2-Sat with m clauses. We construct a temporal graph $\mathcal{G}$ as follows. For each variable x of ϕ we add two vertices denoted v_x and $\mathrm{v}_{\overline{x}}$, connected by an edge with label 1. Furthermore, for each $1 \leq i \leq m - k + 1$ we add two vertices v_x^i and $\mathrm{v}_{\overline{x}}^i$ connected by an edge with label 1. We then connect v_x^i with $\mathrm{v}_{\overline{x}}$ and $\mathrm{v}_{\overline{x}}^i$ with v_x using two edges labeled 4. Thus $\mathrm{v}_x, \mathrm{v}_{\overline{x}}, \mathrm{v}_x^i, \mathrm{v}_{\overline{x}}^i$ is a 4-cycle whose edges alternating between 1 and 4. Afterwards, for each clause $(a \Rightarrow b)$ of ϕ with a, b being literals, we add a new vertex $\mathrm{w}_{a,b}$. Then we connect $\mathrm{w}_{a,b}$ to v_a by an edge labeled 2 and to v_b by an edge labeled 3. Consider Figure 5 for an illustration. Observe that $\mathcal{G}$ can be computed in polynomial time.

We claim that $(\mathcal{G} = (G, \lambda), \emptyset, m - k)$ is a yes-instance of TTC if and only if ϕ has a truth assignment satisfying k clauses.

For the proof, begin by observing that $\mathcal{G}$ does not contain any triangle. Thus an orientation of $\mathcal{G}$ is (weakly) (strict) transitive if and only if it does not have any oriented temporal 2-path, i.e. a temporal path of two edges with both edges being directed forward. We call a vertex v of $\mathcal{G}$ *happy* about some orientation if v is not the center vertex of an oriented temporal 2-path. Thus an orientation of $\mathcal{G}$ is transitive if and only if all vertices are happy.

[8] i.e. a conjunction of clauses of the form $(a \Rightarrow b)$ where a, b are literals.

($\Leftarrow$): Let α be a truth assignment to the variables (and thus literals) of ϕ satisfying k clauses of ϕ. For each literal a with $\alpha(a) = \texttt{true}$, orient all edges such that they point away from v_a and v_a^i, $1 \leq i \leq m-k+1$. For each literal a with $\alpha(a) = \texttt{false}$, orient all edges such that they point towards v_a and v_a^i, $1 \leq i \leq m-k+1$. Note that this makes all vertices v_a and v_a^i happy. Now observe that a vertex $\mathrm{w}_{a,b}$ is happy unless its edge with v_a is oriented towards $\mathrm{w}_{a,b}$ and its edge with v_b is oriented towards v_b. In other words, $\mathrm{w}_{a,b}$ is happy if and only if α satisfies the clause $(a \Rightarrow b)$. Thus there are at most $m-k$ unhappy vertices. For each unhappy vertex $\mathrm{w}_{a,b}$, we add a new oriented edge from v_a to v_b with label 5. Note that this does not make v_a or v_b unhappy as all adjacent edges are directed away from v_a and towards v_b. The resulting temporal graph is transitively oriented.

($\Rightarrow$): Now let a transitive orientation F' of $\mathcal{G}' = (G', \lambda')$ be given, where $\mathcal{G}'$ is obtained from $\mathcal{G}$ by adding at most $m-k$ time edges. Clearly we may also interpret F' as an orientation induced of $\mathcal{G}$. Set $\alpha(x) = \texttt{true}$ if and only if the edge between v_x and $\mathrm{v}_{\overline{x}}$ is oriented towards $\mathrm{v}_{\overline{x}}$. We claim that this assignment α satisfies at least k clauses of ϕ.

First observe that for each variable x and $1 \leq i \leq m-k+1$, F' is a transitive orientation of the 4-cycle $\mathrm{v}_x, \mathrm{v}_{\overline{x}}, \mathrm{v}_x^i, \mathrm{v}_{\overline{x}}^i$ if and only if the edges are oriented alternatingly. Thus, for each variable, at least one of these $k+1$ 4-cycles is oriented alternatingly. In particular, for every literal a with $\alpha(a) = \texttt{true}$, there is an edge with label 4 that is oriented away from v_a. Conversely, if $\alpha(b) = \texttt{false}$, then there is an edge with label 1 oriented towards v_b (this is simply the edge from $\mathrm{v}_{\overline{b}}$).

This implies that every edge with label 2 or 3 oriented from some vertex $\mathrm{w}_{c,d}$ (where either $a = c$ or $a = d$) towards v_a with $\alpha(a) = \texttt{true}$ requires $E(G') \setminus E(G)$ to contain an edge from $\mathrm{w}_{c,d}$ to some $\mathrm{v}_{\overline{a}}^i$. Analogously every edge with label 2 or 3 oriented from v_a with $\alpha(a) = \texttt{false}$ to some $\mathrm{w}_{c,d}$ requires $E(G') \setminus E(G)$ to contain an edge from $\mathrm{v}_{\overline{a}}$ to $\mathrm{w}_{c,d}$.

Now consider the alternative orientation F'' obtained from α as detailed in the converse orientation of the proof. For each edge between v_a and $\mathrm{w}_{c,d}$ where F' and F'' disagree, F'' might potentially require $E(G') \setminus E(G)$ to contain the edge $\mathrm{v}_c\mathrm{v}_d$ (labeled 5, say), but in turn saves the need for some edge $\mathrm{w}_{c,d}\mathrm{v}_{\overline{a}}^i$ or $\mathrm{v}_{\overline{a}}\mathrm{w}_{c,d}$, respectively. Thus, overall, F'' requires at most as many edge additions as F', which are at most $m-k$. As we have already seen in the converse direction, F'' requires exactly one edge to be added for every clause of ϕ which is not satisfied. Thus, α satisfies at least k clauses of ϕ. $\square$

We now show that TTC can be solved in polynomial time, if all edges are already oriented, as the next theorem states. While we only discuss the algorithm for TTC the algorithm only needs marginal changes to work for all other variants.

Theorem 22. *An instance $(\mathcal{G}, F, k)$ of* TTC *where $\mathcal{G} = (G, \lambda)$ and $G = (V, E)$, can be solved in $O(m^2)$ time if F is an orientation of E, where $m = |E|$.*

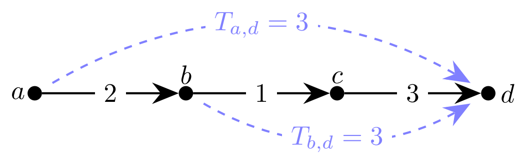


Figure 6: Example of a tail-heavy path.

The actual proof of Theorem 22 is deferred to the end of this section. The key idea for the proof is based on the following definition. Assume a temporal graph $\mathcal{G}$ and an orientation F of $\mathcal{G}$ to be given. Let $G' = (V, F)$ be the underlying graph of $\mathcal{G}$ with its edges directed according to F. We call a (directed) path P in G' *tail-heavy* if the time-label of its last edge is largest among all edges of P, and we define $t(P)$ to be the time-label of that last edge of P. For all $u, v \in V$, denote by $T_{u,v}$ the maximum value $t(P)$ over all tail-heavy (u,v)-paths P of length at least 2 in G'; if such a path does not exist then $T_{u,v} = \perp$. If the temporal graph $\mathcal{G}$ with orientation F can be completed to be transitive, then adding the time edges of the set

$$X(\mathcal{G}, F) \coloneqq \{(uv, T_{u,v}) \mid T_{u,v} \neq \perp\},$$

which are not already present in $\mathcal{G}$ is an optimal way to do so. Consider Figure 6 for an example.

Lemma 23. *The set $X(\mathcal{G}, F)$ can be computed in $O(m^2)$ time, where $\mathcal{G}$ is a temporal graph with m time-edges and F an orientation of $\mathcal{G}$.*

Proof. For each edge vw, we can take G' (defined above), remove w and all arcs whose label is larger than $\lambda(v, w)$, and do a depth-first-search from v to find all vertices u which can reach v in the resulting graph. Each of these then has $T_{u,w} \geq \lambda(v, w)$. By doing this for every edge vw, we obtain $T_{u,w}$ for every vertex pair u, w. The overall running time is clearly $O(m^2)$. $\square$

Until the end of this section we are only considering the instance $(\mathcal{G}, F, k)$ of TTC, where $\mathcal{G} = (G, \lambda)$, $G = (V, E)$, and F is an orientation of $\mathcal{G}$. Hence, we can say a set X of oriented time-edges is a *solution* to I if $X' := \{\{u, v\} \mid (uv, t) \in X\}$ is disjoint from E, satisfies $|X| = |X'| \leq k$, and $F' := F \cup \{uv \mid (uv, t) \in X\}$ is a transitive orientation of the temporal graph $\mathcal{G} + X := ((V, E \cup X'), \lambda')$, where $\lambda'(e) := \lambda(e)$ if $e \in E$ and $\lambda'(u, v) := t$ if X contains (uv, t) or (vu, t).

The algorithm we use to show Theorem 22 will use $X(\mathcal{G}, F)$ to construct a solution (if there is any) of a given instance $(\mathcal{G}, F, k)$ of TTC where F is a orientation of E. To prove the correctness of this approach, we make use of the following.

Lemma 24. *Let $I = (\mathcal{G} = (G, \lambda), F, k)$ be an instance of* TTC*, where $G = (V, E)$ and F is an orientation of E and X an solution for I. Then, for any $(vu, T_{v,u}) \in X(\mathcal{G}, F)$ there is a (vu, t) in $\mathcal{G} + X$ with $t \geq T_{v,u}$.*

Proof. Let $(v_0v_\ell, T_{v_0,v_\ell}) \in X(\mathcal{G}, F)$, and $G' = (V, F)$. Hence, there is a tail-heavy (v_0, v_ℓ)-path P in G' of length $\ell \geq 2$. If $\ell = 2$, then clearly $\mathcal{G} + X$ must contain the time edge (v_1v_ℓ, t) such that $t \geq T_{v_1,v_\ell}$. Now let $\ell > 2$ and $V(P) := \{v_i \mid i \in \{0, 1, \ldots, \ell\}\}$ and $E(P) = \{v_{i-1}v_i \mid i \in [\ell]\}$. Since there is a tail-heavy $(v_{\ell-2}, v_\ell)$-path in G' of length 2, $\mathcal{G} + X$ must contain a time-edge $(v_{\ell-2}v_\ell, t)$ with $t \geq T_{v_0,v_\ell}$. Therefore, the (directed) underlying graph of $\mathcal{G} + X$ contains a tail-heavy (v_0, v_ℓ)-path of length $\ell - 1$. By induction, $\mathcal{G} + X$ must contain the time edge (v_1v_ℓ, t') such that $t' \geq t \geq T_{v_0,v_\ell}$. □

Form Lemma 24, it follows that we can use $X(\mathcal{G}, F)$ to identify *no*-instances in some cases.

Corollary 25. *Let $I = (\mathcal{G} = (G, \lambda), F, k)$ be an instance of* TTC*, where $G = (V, E)$ and F is an orientation of E. Then, I is a* no*-instance, if for some $v, u \in V$*

1. *there are time-edges $(vu, t) \in X(\mathcal{G}, F)$ and $(uv, t') \in X(\mathcal{G}, F)$,*
2. *there is an edge $uv \in F$ such that $(vu, T_{v,u}) \in X(\mathcal{G}, F)$, or*
3. *there is an edge $vu \in F$ such that $(vu, T_{v,u}) \in X(\mathcal{G}, F)$ with $\lambda(v, u) < T_{v,u}$.*

We are now ready to prove Theorem 22.

Proof of Theorem 22. Let $I = (\mathcal{G} = (G, \lambda), F, k)$ be an instance of TTC, where F is a orientation of E. First we compute $X(\mathcal{G}, F)$ in polynomial time, see Lemma 23. Let $Y = \{(vu, t) \in X(\mathcal{G}, F) \mid \{v, u\} \notin E\}$ and report that I is a *no*-instance if $|Y| > k$ or one of the conditions of Corollary 25 holds true. Otherwise report that I is a *yes*-instance. This gives an overall running time of $O(m^2)$.

Clearly, if one of the conditions of Corollary 25 holds true, then I is a *no*-instance. Moreover, by Lemma 24 any solution contains at least $|Y|$ time edges. Thus, if $|Y| > k$, then I is a *no*-instance.

If we report that I is a *yes*-instance, then we claim that Y is a solution for I. Let $F' \supseteq F$ be a orientation of $\mathcal{G} + Y$. Assume towards a contradiction that F' is not transitive. Then, there is a temporal path $((vu, t_1), (uw, t_2))$ in $\mathcal{G} + Y$ such that there is no time-edge (uw, t) in $\mathcal{G} + Y$, with $t \geq t_2$. By definition of $X(\mathcal{G}, F)$, the directed graph $G' = (V, F)$ contains a tail-heavy (v, u)-path P_1 with $t_1 = t(P_1)$ and a tail-heavy (u, w)-path P_2 with $t_2 = t(P_2) \geq t_1$. By concatenation of P_1 and P_2, we obtain that the G' contains a (v, w)-path P' of length at least two such that $t_2 = t(P')$. Thus, $t_2 \leq T_{v,w}$ and $(vw, T_{v,w}) \in X(\mathcal{G})$—a contradiction. □

Using Theorem 22 we can now prove that TTC is fixed-parameter tractable (FPT) with respect to the number of unoriented edges in the input temporal graph $\mathcal{G}$.

Corollary 26. *Let $I = (\mathcal{G} = (G, \lambda), F, k)$ be an instance of* TTC*, where $G = (V, E)$. Then I can be solved in $O(2^q \cdot m^2)$, where $q = |E| - |F|$ and m the number of time edges.*

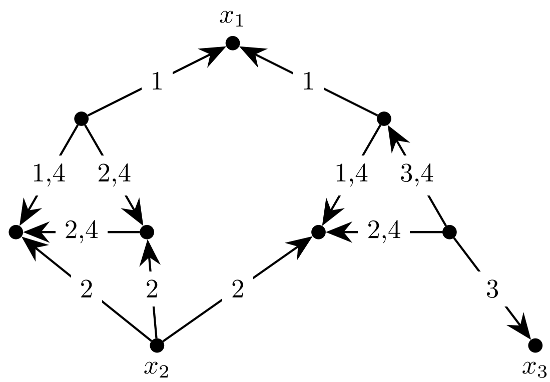


Figure 7: Temporal graph constructed from the formula $\texttt{NAE}(x_1, x_2, x_2) \wedge \texttt{NAE}(x_1, x_2, x_3)$ and orientation corresponding to setting $x_1 = \texttt{false}$, $x_2 = \texttt{true}$, and $x_3 = \texttt{false}$. Each attachment vertex is at the clockwise end of its edge.

Proof. Note that there are 2^q ways to orient the q unoriented edges. For each of these 2^q orientations of these q edges, we obtain a fully oriented temporal graph. Then we can solve TTC on each of these fully oriented graphs in $O(m^2)$ time by Theorem 22. Summarizing, we can solve TTC on I in $2^q \cdot m^2$ rime. $\square$

5. Deciding Multilayer Transitive Orientation

In this section we prove that Multilayer Transitive Orientation (MTO) is NP-complete, even if every edge of the given temporal graph has at most two labels. Recall that this problem asks for an orientation F of a temporal graph $\mathcal{G} = (G, \lambda)$ (i.e. with exactly one orientation for each edge of G) such that, for every "time-layer" $t \geq 1$, the (static) oriented graph defined by the edges having time-label t is transitively oriented in F. As we discussed in Section 2, this problem makes more sense when every edge of G potentially has multiple time-labels, therefore we assume here that the time-labeling function is $\lambda : E \to 2^{\mathbb{N}}$.

Theorem 27. MTO *is NP-complete, even on temporal graphs with at most two labels per edge.*

Proof. We give a reduction from monotone Not-All-Equal-3Sat, which is known to be NP-hard [44]. So let $\phi = \bigwedge_{i=1}^{m} \texttt{NAE}(y_{i,1}, y_{i,2}, y_{i,3})$ be a monotone Not-All-Equal-3Sat instance and $X := \{x_1, \ldots, x_n\} := \bigcup_{i=1}^{m} \{y_{i,1}, y_{i,2}, y_{i,3}\}$ be the set of variables.

Start with an empty temporal graph $\mathcal{G}$. For every clause $\texttt{NAE}(y_{i,1}, y_{i,2}, y_{i,3})$, add to $\mathcal{G}$ a triangle on three new vertices and label its edges $\mathrm{a}_{i,1}, \mathrm{a}_{i,2}, \mathrm{a}_{i,3}$. Give all these edges label $n+1$. For each of these edges, select one of its endpoints to be its *attachment vertex* in such a way that no two edges share an attachment vertex. Next, for each $1 \leq i \leq n$, add a new vertex v_i. Let $A_i := \{\mathrm{a}_{i,j} \mid y_{i,j} =$

$x_i\}$. Add the label i to every edge in A_i and connect its attachment vertex to v_i with an edge labeled i. See also Figure 7.

We claim that $\mathcal{G}$ is a *yes*-instance of MTO if and only if ϕ is satisfiable.

($\Leftarrow$): Let $\alpha : X \to \{\texttt{true}, \texttt{false}\}$ be an assignment satisfying ω. For every $x_i \in X$, orient all edges adjacent to v_i away from v_i if $\alpha(x_i) = \texttt{true}$ and towards v_i otherwise. Then, orient every edge $\mathrm{a}_{i,j}$ towards its attachment vertex if $\alpha(y_{i,j}) = \texttt{true}$ and away from it otherwise.

Note that in the layers 1 through n every vertex either has all adjacent edges oriented towards it or away from it. Thus these layers are clearly transitive. It remains to consider layer $n+1$ which consists of a disjoint union of triangles. Each such triangle $\mathrm{a}_{i,1}, \mathrm{a}_{i,2}, \mathrm{a}_{i,3}$ is oriented non-transitively (i.e. cyclically) if and only if $\alpha(y_{i,1}) = \alpha(y_{i,2}) = \alpha(y_{i,3})$, which never happens if α satisfies ϕ.

($\Rightarrow$): Let ω be an orientation of the underlying edges of $\mathcal{G}$ such that every layer is transitive. Since they all share the same label i, the edges adjacent to v_i must be all oriented towards or all oriented away from v_i. We set $\alpha(x_i) = \texttt{false}$ in the former and $\alpha(x_i) = \texttt{true}$ in the latter case. This in turn forces each edge $\mathrm{a}_{i,j}$ to be oriented towards its attachment vertex if and only if $\alpha(\mathrm{a}_{i,j}) = \texttt{true}$. Therefore, every clause $\texttt{NAE}(y_{i,1}, y_{i,2}, y_{i,3})$ is satisfied, since the three edges $\mathrm{a}_{i,1}, \mathrm{a}_{i,2}, \mathrm{a}_{i,3}$ form a triangle in layer $n+1$ and can thus not be oriented cyclically (i.e. all towards or all away from their respective attachment vertices). $\square$

6. Conclusion

We introduced and studied four natural variants of temporal graph transitivity. Although these four variants look superficially similar, they turn out to have massive differences in their computational complexity. Two variants (Strong TTO and Strong Strict TTO) are solvable by straightforward reductions to 2SAT. For TTO we provided a technically involved polynomial-time algorithm which solves the problem by first reducing it to the satisfiability of a mixed Boolean formula (having both clauses with three and with two literals) and by then using a series of structural properties to devise a polynomial-time algorithm. That is, we reduce TTO to the satisfiability problem of a special subclass of mixed Boolean formulas which turns out to be efficiently solvable. We leave it open for future research whether a compact set of conditions can be given which define this subclass of mixed Boolean formulas, as this might be of independent interest. The last variant Strict TTO turns out to be NP-hard.

We further studied the "completion"-problem corresponding to each of the four temporal transitivity variants, that is, finding the minimum number of time edges that need to be added to a given temporal graph to make it transitive. We show for all four completion problem variants that they are NP-hard. However if the edges of the temporal input graph are already oriented, we obtain polynomial-time solvability which we can easily generalize to an FPT-algorithm for the number of unoriented edges as a parameter. Here, we in particular leave the parameterized complexity with respect to the solution size or other parameters open for future research. Lastly, we investigate a natural extension of transitivity to multilayer graphs and show that deciding whether a given multilayer graph is transitive is NP-hard.

References

[1] E. C. Akrida, L. Gasieniec, G. B. Mertzios, and P. G. Spirakis. Ephemeral networks with random availability of links: The case of fast networks. *Journal of Parallel and Distributed Computing*, 87:109–120, 2016. 3

[2] E. C. Akrida, L. Gasieniec, G. B. Mertzios, and P. G. Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944, 2017. 3

[3] E. C. Akrida, G. B. Mertzios, S. E. Nikoletseas, C. L. Raptopoulos, P. G. Spirakis, and V. Zamaraev. How fast can we reach a target vertex in stochastic temporal graphs? *Journal of Computer and System Sciences*, 114:65–83, 2020. 3

[4] E. C. Akrida, G. B. Mertzios, P. G. Spirakis, and V. Zamaraev. Temporal vertex cover with a sliding time window. *Journal of Computer and System Sciences*, 107:108–123, 2020. 3

[5] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. 3

[6] J. Alman and V. V. Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539, 2021. 4

[7] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979. 5, 8, 17, 18

[8] K. Axiotis and D. Fotakis. On the size and the approximability of minimum temporally connected subgraphs. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 149:1–149:14, 2016. 4

[9] M. Bentert, A.-S. Himmel, H. Molter, M. Morik, R. Niedermeier, and R. Saitenmacher. Listing all maximal k-plexes in temporal graphs. *ACM Journal of Experimental Algorithmics*, 24(1):13:1–13:27, 2019. 3

[10] M. Bentert, A.-S. Himmel, A. Nichterlein, and R. Niedermeier. Efficient computation of optimal temporal walks under waiting-time constraints. *Applied Network Science*, 5(1):73, 2020. 3

[11] R. Bredereck, C. Komusiewicz, S. Kratsch, H. Molter, R. Niedermeier, and M. Sorge. Assessing the computational complexity of multilayer subgraph detection. *Network Science*, 7(2):215–241, 2019. 6

[12] B.-M. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003. 3

[13] S. Buß, H. Molter, R. Niedermeier, and M. Rymar. Algorithmic aspects of temporal betweenness. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2084–2092. ACM, 2020. 3

[14] A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Technical report, Defence R&D Canada, April 2013. 2

[15] A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. Technical report, Defence R&D Canada, April 2013. 2

[16] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012. 2

[17] A. Casteigts, A.-S. Himmel, H. Molter, and P. Zschoche. Finding temporal paths under waiting time constraints. *Algorithmica*, 83(9):2754–2802, 2021. 3

[18] A. Casteigts, J. G. Peters, and J. Schoeters. Temporal cliques admit sparse spanners. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 134:1–134:14, 2019. 4

[19] J. Chen, H. Molter, M. Sorge, and O. Suchý. Cluster editing in multi-layer and temporal graphs. In *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC)*, pages 24:1–24:13, 2018. 3

[20] J. Enright, K. Meeks, G. Mertzios, and V. Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119:60–77, 2021. 4

[21] J. Enright, K. Meeks, and F. Skerman. Assigning times to minimise reachability in temporal graphs. *Journal of Computer and System Sciences*, 115:169–186, 2021. 3

[22] T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 444–455, 2015. 3

[23] T. Fluschnik, H. Molter, R. Niedermeier, M. Renken, and P. Zschoche. Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*, 806:197–218, 2020. 3

[24] M. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976. 33

[25] M. C. Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2nd edition, 2004. 3, 4, 5, 6, 12, 14

[26] S. L. Hakimi, E. F. Schmeichel, and N. E. Young. Orienting graphs to optimize reachability. *Information Processing Letters*, 63(5):229–235, 1997. 33

[27] A.-S. Himmel, H. Molter, R. Niedermeier, and M. Sorge. Adapting the Bron-Kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1):35:1–35:16, 2017. 3

[28] P. Holme and J. Saramäki. *Temporal network theory*, volume 2. Springer, 2019. 2

[29] D. Kempe, J. M. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002. 2, 3, 4

[30] H. Kim and R. Anderson. Temporal node centrality in complex networks. *Physical Review E*, 85(2):026107, 2012. 3

[31] R. M. McConnell and J. P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 536–545, 1994. 4

[32] R. M. McConnell and J. P. Spinrad. Linear-time transitive orientation. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 19–25, 1997. 4

[33] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1-3):189–241, 1999. 4

[34] D. B. McDonald and D. Shizuka. Comparative transitive and temporal orderliness in dominance networks. *Behavioral Ecology*, 24(2):511–520, 2013. 3

[35] G. B. Mertzios. The recognition of simple-triangle graphs and of linear-interval orders is polynomial. *SIAM Journal on Discrete Mathematics*, 29(3):1150–1185, 2015. 18

[36] G. B. Mertzios, O. Michail, and P. G. Spirakis. Temporal network optimization subject to connectivity constraints. *Algorithmica*, 81(4):1416–1449, 2019. 3

[37] G. B. Mertzios, H. Molter, R. Niedermeier, V. Zamaraev, and P. Zschoche. Computing maximum matchings in temporal graphs. *Journal of Computer and System Sciences*, 137:1–19, 2023. 3

[38] G. B. Mertzios, H. Molter, M. Renken, P. G. Spirakis, and P. Zschoche. The complexity of transitively orienting temporal graphs. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 202 of *LIPIcs*, pages 75:1–75:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 1

[39] G. B. Mertzios, H. Molter, and V. Zamaraev. Sliding window temporal graph coloring. *Journal of Computer and System Sciences*, 120:97–115, 2021. 3

[40] O. Michail and P. G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72–72, Jan. 2018. 2

[41] R. Moskovitch and Y. Shahar. Medical temporal-knowledge discovery via temporal abstraction. In *Proceedings of the AMIA Annual Symposium*, page 452, 2009. 3

[42] R. Moskovitch and Y. Shahar. Fast time intervals mining using the transitivity of temporal relations. *Knowledge and Information Systems*, 42(1):21–48, 2015. 3

[43] V. Nicosia, J. Tang, C. Mascolo, M. Musolesi, G. Russo, and V. Latora. Graph metrics for temporal networks. In *Temporal Networks*. Springer, 2013. 2

[44] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978. 5, 9, 17, 38

[45] J. P. Spinrad. On comparability and permutation graphs. *SIAM Journal on Computing*, 14(3):658–670, 1985. 4

[46] J. P. Spinrad. *Efficient graph representations*, volume 19 of *Fields Institute Monographs*. American Mathematical Society, 2003. 4

[47] X. Tannier and P. Muller. Evaluating temporal graphs built from texts via transitive reduction. *Journal of Artificial Intelligence Research (JAIR)*, 40:375–413, 2011. 3

[48] C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984. 10

[49] T. Viard, M. Latapy, and C. Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016. 3

[50] H. Wu, J. Cheng, Y. Ke, S. Huang, Y. Huang, and H. Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016. 3

[51] P. Zschoche, T. Fluschnik, H. Molter, and R. Niedermeier. The complexity of finding separators in temporal graphs. *Journal of Computer and System Sciences*, 107:72–92, 2020. 3