Structural Parameters for Dense Temporal Graphs

Jessica Enright
School of Computing Science, University of Glasgow, UK

Samuel D. Hand
School of Computing Science, University of Glasgow, UK

Laura Larios-Jones
School of Computing Science, University of Glasgow, UK

Kitty Meeks
School of Computing Science, University of Glasgow, UK

Abstract

Temporal graphs provide a useful model for many real-world networks. Unfortunately, the majority of algorithmic problems we might consider on such graphs are intractable. There has been recent progress in defining structural parameters which describe tractable cases by simultaneously restricting the underlying structure and the times at which edges appear in the graph. These all rely on the temporal graph being sparse in some sense. We introduce temporal analogues of three increasingly restrictive static graph parameters – cliquewidth, modular-width and neighbourhood diversity – which take small values for highly structured temporal graphs, even if a large number of edges are active at each timestep. The computational problems solvable efficiently when the temporal cliquewidth of the input graph is bounded form a subset of those solvable efficiently when the temporal modular-width is bounded, which is in turn a subset of problems efficiently solvable when the temporal neighbourhood diversity is bounded. By considering specific temporal graph problems, we demonstrate that (up to standard complexity theoretic assumptions) these inclusions are strict.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Graph algorithms analysis; Theory of computation → Fixed parameter tractability

Keywords and phrases Graph algorithms, Parameterized Algorithms, Temporal Graphs

Digital Object Identifier 10.4230/LIPIcs.MFCS.2024.52

Related Version *Full Version*: `https://doi.org/10.48550/arXiv.2404.19453`

Funding *Jessica Enright*: Supported by EPSRC grant EP/T004878/1.
Samuel D. Hand: Supported by an EPSRC doctoral training account.
Kitty Meeks: Supported by EPSRC grant EP/T004878/1.

1 Introduction

Temporal graphs, in which the set of active edges changes over time, are a useful formalism for modelling numerous real-world phenomena, from social networks in which friendships evolve over time to transport networks in which a particular connection is only available on particular days and times. This has inspired a large volume of research into the algorithmic aspects of such graphs in recent years [8, 26, 34], but unfortunately in many cases even problems which admit polynomial-time algorithms on static graphs become intractable in the temporal setting.

This has motivated the study of computational problems on restricted classes of temporal graphs, with mixed success: in a few cases, restricting the structure of the underlying static graph (e.g. to be a path or a tree) is effective, but numerous natural temporal problems remain intractable even when the underlying graph is very strongly restricted (e.g. when it is required to be a path [33] or a star [2]). Recently, several promising new parameters have been introduced that simultaneously restrict properties of the static underlying graph and the times at which edges are active in the graph; these include several analogues of treewidth for temporal graphs [18, 30], the temporal feedback edge/connection number [23], the timed vertex feedback number [7] and the (vertex-)interval-membership-width of the temporal graph [6]. However, all of these new temporal parameters are only small for temporal graphs that are, in some sense, sparse: none of them can be bounded on a temporal graph which has a superlinear (in the number of vertices) number of active edges at every timestep.

In this paper, we attempt to fill this gap in the toolbox of parameters for temporal graphs by introducing three new parameters which can take small values on temporal graphs which are dense but are sufficiently highly structured. Specifically, we define natural temporal analogues of cliquewidth, modular-width and neighbourhood diversity, all of which have proved highly effective in the design of efficient algorithms for static graphs.

Importantly, the neighbourhood diversity of a static graph upper bounds its modular-width, which upper bounds its cliquewidth. Both cliquewidth (introduced by Courcelle et al. [13]) and modular-width (introduced by Gajarský et al. [19], using the long-standing notion of modular decompositions [20]) can be defined in terms of width measures over composition trees allowing particular operations. Cliquewidth constructions have greater flexibility due to the fact that we are allowed to use an additional "relabelling" operation; this makes it possible, for example, to build long induced paths, which cannot exist in graphs of small modular width. Courcelle et al. [11] show that any graph property expressible in monadic second order is solvable in linear time on graphs of bounded cliquewidth. Gajarský et al. [19] provide examples of problems (HAMILTON PATH and COLOURING) that are hard with respect to cliquewidth but tractable with respect to the more restrictive parameter modular-width. Neighbourhood diversity is a highly restrictive parameter, introduced by Lampis [28], which requires that large sets of vertices have identical neighbourhoods. Cordasco [9] demonstrated that EQUITABLE COLOURING is hard with respect to modular-width but tractable with respect to neighbourhood diversity.

These three static parameters are the inspiration for our temporal parameters. Informally, our new parameters are defined as follows:

- A temporal graph has *temporal neighbourhood diversity (TND)* at most k if its vertices can be partitioned into at most k classes such that each class induces either an independent set or a clique in which all edges are active at exactly the same times, and any two vertices in the same class have exactly the same neighbours outside the class at each timestep.
- *Temporal modular-width (TMW)* is a generalisation of TND: a temporal graph has TMW at most k if its vertices can be partitioned into modules such that two vertices in the same module must have the exactly the same neighbours outside the class at each timestep, but now each module need only be a temporal graph which itself has TMW at most k, rather than a clique or independent set.
- Like the static version, *temporal cliquewidth (TCW)* is defined to be the minimum number of labels needed to construct a temporal graph using four operations (create a vertex with a new label; take a disjoint union of two graphs; add all edges between vertices of two specified labels; relabel all occurrences of one label to another); the difference from the static case is that when adding edges between two sets of vertices these edges must all be active at exactly the same times.

We note that in every case we will recover the corresponding static parameter if all edges are active at the same times. It is immediate that the TND of a temporal graph is an upper bound on its TMW, and it is straightforward to show (see Section 3) that the TMW is an upper bound on the TCW. Thus, the most general algorithmic result we hope to obtain is to show that a problem is tractable when the TCW of the input temporal graph is bounded, but we expect to be able to show tractability for more problems as we impose stronger restictions on the input by bounding respectively the TMW and the TND.

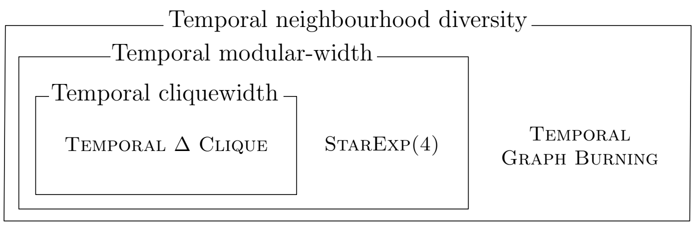

Figure 1 A diagram of our parameters and the problems we show to be tractable with respect to each. A problem is in a rectangle if it is tractable with respect to the parameter it is labelled with. Assuming P≠NP, each problem is in the rectangle for the most general of the three parameters for which it is tractable.

To illustrate the value of considering this hierarchy of parameters, we provide examples of problems which can be solved efficiently when each of our three new parameters is bounded, but which (in the case of TND and TMW) remain intractable when we restrict only the next most restrictive parameter, as illustrated in Figure 1. Specifically, we prove that:

- Temporal Clique is solvable in linear time when a temporal cliquewidth decomposition of constant width is given (see Section 2).
- StarExp(4) (the problem of deciding whether there is a closed temporal walk visiting all vertices in a star when each edge is active at no more than four times) remains NP-hard on graphs with TCW at most three, but is solvable in polynomial time when the TMW of the input graph is bounded by a constant; in fact we provide an fpt-algorithm[1] with respect to TMW (see Section 3).
- Graph Burning is NP-hard on temporal graphs with constant TMW, but is solvable in polynomial time when the TND of the input graph is bounded by a constant; again this is an fpt-algorithm with respect to TND (see Section 4).

We also (in Section 4) provide an fpt-algorithm to solve SingMinReachDelete parameterised by TND (when the number of appearances of each edge is bounded), in order to illustrate additional techniques that may be used when working with this new temporal parameter. This problem asks, for a given source vertex, what the cardinality of the smallest set of time-edges is such that their deletion leaves at most r vertices temporally reachable from the source. We conjecture that SingMinReachDelete is another example of a problem that is tractable with respect to TND but intractable when only the TMW is restricted.

The remainder of the paper is organised as follows. We conclude this section by introducing some key notation and definitions used throughout the paper. The following three sections are devoted to TCW, TMW and TND respectively, with each section containing the formal definition of a parameter as well as results about problems which can be solved efficiently when that parameter is bounded. Many proof details are omitted due to space constraints. A full version of the paper can be found on arXiv. Statements with proofs omitted are highlighted with a ($\star$).

[1] We use fpt (lowercase) as a descriptor of algorithms witnessing the inclusion of a problem in the parameterised complexity class FPT.

1.1 Notation and definitions

We use a number of standard notations for temporal graphs and related notions. A *temporal graph* $\mathcal{G} = (G, \lambda)$ consists of an underlying static graph $\mathcal{G}_{\downarrow} = G$, and a time-labeling function $\lambda : E \to 2^{\mathbb{N}}$, assigning to each edge a set of timesteps at which it is active. We refer to a pair (e, t) consisting of an edge $e \in E(G)$ and time $t \in \lambda(e)$ as a *time-edge*. The set of all time-edges of a temporal graph is denoted by $\varepsilon(\mathcal{G})$ and the *lifetime* Λ of a temporal graph refers to the final time at which any edge is active, i.e. $\Lambda = \max\{\max \lambda(e) : e \in E(G)\}$. The *snapshot* $\mathcal{G}_t$ of a temporal graph $\mathcal{G}$ at time t is the static graph $G = (V, E_t)$ where E_t is the set of edges active at time t.

A *temporal path* on the temporal graph $\mathcal{G} = (G, \lambda)$ is a sequence of edge, time pairs $(e_1, t_1), ..., (e_\ell, t_\ell)$, such that $e_1, ..., e_\ell$ is a path on G, and $t_i \in \lambda(e_i)$ for every $i \in [\ell]$, and $t_1, ..., t_\ell$ is a strictly increasing sequence of times. Given a temporal path $(e_1, t_1), ..., (e_\ell, t_\ell)$ we refer to the time t_1 as its *departure time*, and t_ℓ as its *arrival time*.

We refer the reader to [15] for standard definitions from the field of parameterised complexity.

2 Tractability with respect to Temporal Cliquewidth

In this section we give the formal definition of the first of our new parameters, temporal cliquewidth, and demonstrate that the problem of finding a temporal clique admits an fpt-algorithm parameterised by temporal cliquewidth. Before defining temporal cliquewidth, we start by recalling the definition of cliquewidth in the static setting, as introduced by Courcelle and Olariu [14].

▶ **Definition 1** (Cliquewidth). *The* cliquewidth *of a static graph $G = (V, E)$ is the number of labels required to construct G using only the following operations:*
1. *Creating a new vertex with label i.*
2. *Taking the disjoint union of two labeled graphs.*
3. *Adding edges to join all vertices labeled i to all vertices labeled j, where $i \neq j$.*
4. *Renaming label i to label j.*

We refer to an algorithm which constructs a graph G using the above operations as a cliquewidth construction *of G.*

Computing the cliquewidth of a graph is NP-hard [17], although there exists a polynomial-time algorithm to recognise graphs of cliquewidth at most three [10].

Translating this definition into the temporal setting, and preserving the idea that vertices with the same label should be indistinguishable when we add edges – which imposes additional restrictions on the times at which new edges are active – we obtain our definition of temporal cliquewidth.

▶ **Definition 2** (Temporal Cliquewidth). *The* temporal cliquewidth *of a temporal graph $\mathcal{G} = (G, \lambda)$ is the number of labels required to construct $\mathcal{G}$ using only the following operations:*
1. *Creating a new vertex with label i.*
2. *Taking the disjoint union of two labeled graphs.*
3. *Adding edges to join all vertices labeled i to all vertices labeled j, where $i \neq j$, such that all the added edges are active at the same set of times T.*
4. *Renaming label i to label j.*

We refer to an algorithm which constructs a temporal graph $\mathcal{G}$ using the above operations as a temporal cliquewidth construction *of $\mathcal{G}$.*

We note that, if temporal graph $\mathcal{G}$ has bounded temporal cliquewidth k, then the underlying graph $G_{\downarrow}$ of $\mathcal{G}$ has cliquewidth at most k. The construction of the underlying graph is found by adding a static edge (if one does not already exist) whenever a time-edge is added in the construction of the temporal graph. In addition, the snapshot of $\mathcal{G}$ at any time t, G_t, also has cliquewidth at most k. This follows from a similar argument. If we have a temporal graph where the edges all appear at the same times, the cliquewidth of the underlying graph is the same as the temporal cliquewidth and the cliquewidth of any snapshots where edges are active. It follows immediately that it is NP-hard to compute temporal cliquewidth, as the NP-hard problem of computing the cliquewidth of a static graph is a special case.

The remainder of this section is devoted to proving that the problem Temporal Δ Clique is in FPT parameterised by the temporal cliquewidth of the input graph. This problem was introduced by Viard et al. [37] and asks, in any interval of times of size Δ, whether there is a set of at least h vertices such that there is an appearance of an edge between every pair of vertices in every sub-interval of Δ times. Hermelin et al. [25] investigate the variant of this problem where the interval in question is the entire lifetime of the temporal graph. More formally, it asks if there exists a set V' of cardinality at least h such that for each pair of distinct vertices $u, v \in V'$ and each time $0 \leq i \leq \Lambda - \Delta$ where Λ is the lifetime of $\mathcal{G}$, there exists a time-edge at time $t' \in [i, \Delta + i]$.

Hermelin et al. note that this is the case if and only if there is a set of vertices of size at least k such that they form a clique on the static graph consisting of edges that appear in every interval of Δ timesteps. They name this static graph a Δ-*association graph.* It is more formally defined as $G = \left(V, \bigcap_{i=1}^{\Lambda(\mathcal{G})-\Delta+1} \bigcup_{j=i}^{i+\Delta-1} E_j\right)$ where E_j is the set of edges active at time j. This reformulates the problem as follows.

Temporal Δ Clique

Input: A temporal graph $\mathcal{G} = (V, E, \lambda)$ and two integers Δ and h where $\Delta \leq T(\mathcal{G})$.
Output: Is there a set $V' \subseteq V$ of vertices such that $|V'| \geq h$ and V' is a clique in the Δ-association graph G of $\mathcal{G}$?

Note that a temporal graph $\mathcal{G}$ with integers Δ and h is a yes-instance of Temporal Δ Clique if and only if its Δ-association graph G and h are a yes-instance of Clique.

▶ **Proposition 1** (⋆). *If a temporal graph $\mathcal{G}$ has temporal cliquewidth k, then the Δ-association graph G of $\mathcal{G}$ has cliquewidth at most k.*

Gurski [22, Theorem 4.4] shows that Clique is solvable in linear time in graphs of bounded cliquewidth if its construction is given. This gives us the following result.

▶ **Theorem 2.** *Given a cliquewidth construction of the Δ-association graph of a temporal graph $\mathcal{G}$, Temporal Δ Clique can be solved in linear time.*

3 Tractability with respect to Temporal Modular-width

We now introduce a more restrictive parameter, temporal modular-width, and show (in Section 3.1) that there exist problems which are efficiently solvable when this parameter is bounded even though they remain intractable on temporal graphs with constant temporal cliquewidth.

We begin with the formal definition of the parameter. Again, we start by recalling the definition of the corresponding static parameter on which our definition is based.

▶ **Definition 3** (Modular-width, Section 2.5 [19]). *Suppose a static graph G can be constructed by the algebraic expression A which uses the following operations:*

1. *Creating an isolated vertex.*
2. *Taking the disjoint union of two graphs.*
3. *Taking the complete join of two graphs. That is, for graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, $V(G_1 \otimes G_2) = V(G_1) \cup V(G_2)$ and $E(G_1 \otimes G_2) = E(G_1) \cup E(G_2) \cup \{(v, w) : v \in V(G_1) \text{ and } w \in V(G_2)\}$.*
4. *The substitution of graphs $G_1, \ldots, G_n$ into a graph G' with vertices $v_1, \ldots, v_n$. This gives the graph $G'(G_1, \ldots, G_n)$ with vertex set $\bigcup_{1\leq i\leq n} V(G_i)$ and edge set $\bigcup_{1\leq i\leq n} E(G_i) \cup \{(v, w) : v \in V(G_i), w \in V(G_j), \text{ and } (v_i, v_j) \in E(G')\}$.*

The width of an expression A is the maximum number of operands in an occurrence of the operation 4 in A. The modular-width *of G, written $MW(G)$, is this minimum width of an expression A which constructs G.*

We refer to the graphs $G_1, \ldots, G_n$ which we substitute into G' as *modules.* It is known that for any graph G an algebraic expression of modular-width $MW(G)$ can be found in linear time [31, 36]. Observe that operations 2 and 3 are special cases of operation 4.

We now define our temporal analogue of this parameter. For simplicity we do not explicitly include the disjoint union and complete join operations, noting that once again these are special cases of the substitution operation.

▶ **Definition 4** (Temporal Modular-width). *Suppose a temporal graph $\mathcal{G}$ can be constructed by the algebraic expression A which uses the following operations:*

1. *Creating an isolated vertex.*
2. *Taking the disjoint union of two temporal graphs.*
3. *Taking the complete join of two temporal graphs at a set of times T. That is, for graphs $\mathcal{G}_1 = ((V_1, E_1), \lambda_1)$ and $\mathcal{G}_2 = ((V_2, E_2), \lambda_2)$, $V(\mathcal{G}_1 \otimes \mathcal{G}_2) = V(\mathcal{G}_1) \cup V(\mathcal{G}_2)$ and $\varepsilon(\mathcal{G}_1 \otimes \mathcal{G}_2) = \varepsilon(\mathcal{G}_1) \cup \varepsilon(\mathcal{G}_2) \cup \{(vw, t) : v \in V(\mathcal{G}_1), w \in V(\mathcal{G}_2) \text{ and } t \in T\}$.*
4. *The substitution of temporal graphs $\mathcal{G}_1, \ldots, \mathcal{G}_n$ into a temporal graph $\mathcal{G}'$ with vertices $v_1, \ldots, v_n$. This gives the graph $\mathcal{G}'(\mathcal{G}_1, \ldots, \mathcal{G}_n)$ with vertex set $\bigcup_{1\leq i\leq n} V(\mathcal{G}_i)$ and time-edge set $\bigcup_{1\leq i\leq n} \varepsilon(\mathcal{G}_i) \cup \{(vw, t) : v \in V(\mathcal{G}_i), w \in V(\mathcal{G}_j), \text{ and } (v_i v_j, t) \in \varepsilon(\mathcal{G}')\}$.*

The width of an expression A is the maximum number of operands in an occurrence of the operation 4 in A. The temporal modular-width *of G is this minimum width of an expression A which constructs G.*

As for temporal cliquewidth, we observe that the temporal modular-width of a temporal graph is equal to the modular-width of the underlying graph if all edges have the same temporal assignment. It follows that, as in the static case, the temporal modular-width bounds the length of the longest induced path in the underlying graph.

We now argue that, as claimed, bounding the temporal modular-width of a temporal graph is a strictly stronger restriction than bounding the temporal cliquewidth.

▶ **Theorem 3** ($\star$). *For any temporal graph $\mathcal{G} = (G, \lambda)$, the temporal cliquewidth is upper bounded by the temporal modular-width.*

Habib and Paul [24] describe a simple algorithm for finding the modular decomposition of a static graph. This operates by finding and repeatedly adding *splitters* to a candidate module - intuitively, splitters are vertices outside of a module that distinguish between vertices inside of a candidate module. We can use a similar splitter-closure method to find the unique maximal temporal modular decomposition:

▶ **Theorem 4** (⋆). *We can find the maximal temporal modular decomposition in time $O(n^4\Lambda)$, where n is the number of vertices in the temporal graph.*

3.1 Star Exploration

In this section we consider the following problem, demonstrating that it remains NP-hard even on temporal graphs with temporal cliquewidth at most three, but that it is solvable in constant time on graphs with bounded temporal modular-width (provided we are given the temporal modular-width of the graph). This problem was first introduced by Akrida et al. [2].

StarExp(τ)

Input: A temporal star (S_n, λ) where $|\lambda(e)| \leq \tau$ for every edge e in the star S_n.
Output: Is there a strict temporal walk, starting and finishing at the centre of the star, which visits every vertex of S_n?

We begin with a simple observation about the temporal cliquewidth of temporal graphs whose underlying graph is a star.

▶ **Lemma 5** (⋆). *A temporal star S_n has temporal cliquewidth at most 3.*

StarExp(τ) is known to be NP-hard even for constant τ [2, 6]. Then, by Lemma 5, StarExp(τ) is an example of a problem which is NP-hard on graphs of bounded temporal cliquewidth. We now show that StarExp(τ) is tractable on graphs of bounded temporal modular-width. We begin with the following lemma.

▶ **Lemma 6** (⋆). *If a temporal star S_n has temporal modular-width at most k, the leaves of S_n can be partitioned into $k-1$ subsets such that, if u and v are in the same subset and c is the central vertex in the star, the edges uc and cv are active at the same times.*

▶ **Lemma 7.** *If there are strictly more than $\tau/2$ leaves of S_n whose incident edges are active at the same times, we have a no-instance of StarExp(τ).*

Proof. By definition of the problem StarExp(τ), each edge in the star is active at most τ times. Therefore, if there are $\lfloor\tau/2\rfloor + 1$ vertices $u_1, \ldots, u_{\lfloor\tau/2\rfloor+1}$ such that $\lambda(u_1c) = \cdots = \lambda(u_{\lfloor\tau/2\rfloor+1}c)$ where c is the central vertex in the star, there is no temporal walk which starts at c and visits all of $u_1, \ldots, u_{\lfloor\tau/2\rfloor+1}$. To see this, note that visiting a leaf and returning requires the use of two distinct time-edges. Therefore, a walk visiting any $\lfloor\tau/2\rfloor + 1$ leaves which departs from and returns to c must consist of at least $\tau + 1$ distinct time-edges. This is not possible if these vertices have incident edges are active at the same times and $|\lambda(uc)| \leq \tau$. ◀

▶ **Theorem 8.** *StarExp(τ) is solvable in $(k\tau)!(k\tau)^{O(1)}$ time when the temporal modular-width of the graph is at most k.*

Proof. Given that the temporal modular-width of the graph is at most k, we check whether the number of leaves is more than $(k-1)\lfloor\tau/2\rfloor$. If the number of leaves is at least $(k-1)\lfloor\tau/2\rfloor+1$ then, by the pigeon-hole principle and Lemma 6, there must be at least $\lfloor\tau/2\rfloor + 1$ leaves whose edges to the central vertex are active at exactly the same times. In this case, by Lemma 7, we conclude that we have a no-instance of StarExp(τ).

Else, we have at most $(k-1)\lfloor\tau/2\rfloor < k\tau$ leaves. Note that, given an ordering of leaves to visit, we can check whether such a walk is valid in time polynomial in k and τ: we need only check for each leaf in order whether there are two appearances of its incident edge following the time-edges used to visit the previous leaf (and if so, greedily use the first two such appearances). Since there are fewer than $(k\tau)!$ possible orderings of the leaves, we can check each possibility in turn in time $(k\tau)!(k\tau)^{O(1)}$. ◀

4 Tractability with respect to Temporal Neighbourhood Diversity

We now turn our attention to our final parameter, temporal neighbourhood diversity, which is the most restrictive and hence allows for the most problems to be solved efficiently. In Section 4.1 we demonstrate that Temporal Graph Burning is solvable in polynomial time when the temporal neighbourhood diversity is bounded by a constant (in fact we give an fpt-algorithm with respect to this parameterisation), even though the problem remains NP-hard when restricted to temporal graphs with constant temporal modular-width. To illustrate further techniques that may be used to design efficient algorithms on graphs of bounded temporal neighbourhood diversity, in Section 4.2 we also give an fpt-algorithm for the problem SingMinReachDelete with a single source vertex.

We begin with the formal definition of temporal neighbourhood diversity. Once again, the definition is modelled on that for static graphs, which was first introduced by Lampis [28] and adpated by Ganian [21] to describe uncoloured graphs. In a static graph, we define the *neighbourhood* $N(v)$ of a vertex v as the set of vertices which share an edge with v.

▶ **Definition 5** (Type, Definition 2.2 [21]). *Two vertices v, v' have the same* type *if and only if $N(v) \setminus \{v'\} = N(v') \setminus \{v\}$.*

▶ **Definition 6** (Neighbourhood Diversity, Definition 2 [28]). *A graph $G = (V, E)$ has* neighbourhood diversity *at most k if and only if there exists a partition of $V(G)$ into at most k sets where all vertices in each set have the same type. We refer to this partition as a* neighbourhood partition.

We note that the neighbourhood diversity of a graph can be computed in linear time [28].

We now define the analogous temporal parameter, where we require that the edges between sets are all active at the same times.

▶ **Definition 7** (Temporal Neighbourhood). *The* temporal neighbourhood *of a vertex v in a temporal graph (G, λ) is the set $TN(v)$ of vertex time pairs (w, t) where $(w, t) \in TN(V)$ if and only if $vw \in E(G)$ and $t \in \lambda(vw)$.*

▶ **Definition 8** (Temporal Type). *Two vertices u, v have the same* temporal type *if and only if $\{(w, t) \in TN(v) : w \neq v\} = \{(w, t) \in TN(v') : w \neq u\}$.*

▶ **Definition 9** (Temporal Neighbourhood Diversity). *A graph $\mathcal{G}$ has* temporal neighbourhood diversity *at most k if and only if there exists a partition of $V(\mathcal{G})$ into at most k sets where all vertices in each set have the same temporal type. We refer to this partition as a* temporal neighbourhood partition.

It is immediate from this definition that, when all edges are assigned the same times, the temporal neighbourhood diversity of the graph is the same as the neighbourhood diversity of the underlying graph. We now argue that, as in the static case, the subgraph induced by any class must form a clique or independent set; moreover, in the temporal setting, this must be true at every timestep.

▶ **Lemma 9** ($\star$). *At any snapshot $\mathcal{G}_t$ of $\mathcal{G}$, the subgraph induced by the vertices in a class X of a temporal neighbourhood partition of $\mathcal{G}$ either forms an independent set or a clique.*

As stated, temporal neighbourhood diversity is the most restrictive parameter in our hierarchy. Observe that for any temporal graph $\mathcal{G}$ the temporal modular-width is upper bounded by the temporal neighbourhood diversity: each class in the temporal neighbourhood partition forms a module, and it follows from Lemma 9 that each of these modules can be constructed by operations 1, 2, and 3 of temporal modular-width construction.

Finally, we argue that we can compute the temporal neighbourhood diversity efficiently.

▶ **Proposition 10** ($\star$). *The temporal neighbourhood diversity of a temporal graph (G, λ) can be calculated in $O(\Lambda n^3)$ time.*

4.1 Temporal Graph Burning

In this section we define Temporal Graph Burning, a temporal analogue of the static Graph Burning problem first proposed by Bonato et al.[5]. Informally, a fire is placed at a new vertex in each time-step and the fire spreads along incident active edges. The problem asks if, after h placements of fire, all vertices are burning. Static Graph Burning is NP-hard on general graphs [4] and was recently shown to be in FPT parameterised by static modular width [27]. In contrast, we prove that Temporal Graph Burning remains NP-hard on graphs with constant temporal modular-width. This difference arises from the fact that, in the static setting, the length of a longest induced path in the graph (which is upper bounded by the modular-width) gives an upper bound on the time taken to burn the graph. In the temporal setting, on the other hand, the times assigned to edges mean that even graphs with small diameter may take many steps to burn. In contrast, we show that Temporal Graph Burning can be solved in time $O(n^5 \Lambda k! 4^k)$ on temporal graphs with n vertices, lifetime Λ and temporal neighbourhood diversity k.

The Temporal Graph Burning problem asks how quickly a fire can be spread over the vertices of a temporal graph in the following discrete time process, where a fire is placed at a vertex of a graph on each timestep.

1. At time $t = 0$ a fire is placed at a chosen vertex. All other vertices are unburnt.
2. At all times $t \geq 1$, the fire spreads, burning all vertices u adjacent to an already burning vertex v where the edge between u and v is active at time t. Then, another fire is placed at a chosen vertex.
3. This process ends once all vertices are burning.

We refer to a sequence of vertices at which fires are placed as a strategy.

▶ **Definition 10** (Burning Strategy). *A* burning strategy *for a temporal graph (G, λ) is a sequence of vertices $S = s_1, s_2, ..., s_h$ such that $s_i \in V(G)$ for all $i \leq h$, and on each timestep i a fire is placed at s_i.*

We say that a strategy $S = s_1, s_2, ..., s_h$ has length h. For convenience, we allow for strategies that place fires at already burning vertices, although it is worth noting that such moves may be omitted. If every vertex in the graph is burning after a strategy is played, we say that strategy is successful.

▶ **Definition 11** (Successful Burning Strategy). *A burning strategy $S = s_1, s_2, ..., s_h$ for a temporal graph (G, λ) is* successful *if every vertex in G is burning on timestep h when the moves from S are played.*

The decision problem asks how many timesteps it takes to burn a given temporal graph.

TEMPORAL GRAPH BURNING

Input: A temporal graph (G, λ) and an integer h.
Output: Does there exist a successful burning strategy for (G, λ) of length less than or equal to h?

This problem is in NP, with a strategy providing a certificate. Given a strategy it can be checked in polynomial time if it is successful and of length less than or equal to h by simulating temporal graph burning on the input graph.

We show that TEMPORAL GRAPH BURNING is NP-hard even on graphs of bounded temporal modular-width. This is achieved by reducing from (3, 2B)-SAT, an NP-hard variant of the Boolean satisfiability problem in which each variable appears exactly twice both positively and negatively [3]. Our reduction produces a graph where each edge is active on exactly one timestep, and furthermore every connected component has a bounded temporal neighbourhood diversity, and hence the graph has bounded temporal modular-width by our earlier observation.

▶ **Theorem 11** ($\star$). *TEMPORAL GRAPH BURNING is NP-hard even when restricted to graphs with constant temporal modular-width.*

We now show that TEMPORAL GRAPH BURNING is solvable efficiently when the temporal neighbourhood diversity of the input graph is bounded. Throughout we assume that the lifetime Λ of the input temporal graph is at most the number of vertices n, as it is possible to burn any temporal graph in n timesteps by placing a fire at every vertex in turn. We begin by defining notation for the burning set of vertices on a given timestep when a strategy is played.

▶ **Definition 12** (Burning Set). *Given a strategy S the* burning set *$B_t(S)$ at timestep t is the set of burning vertices immediately after a fire is placed on timestep t when S is played.*

We now give a number of results about how successful strategies can be modified to have certain desirable properties; these results allow us to bound the number of possible strategies we must check to determine whether there is a successful strategy of the desired length.

▶ **Lemma 12** ($\star$). *Let S be a successful strategy for (G, λ). Suppose that there is some timestep $t_1 < |S|$ and strategy R with $|R| = |S|$ such that $B_{t_1}(S) \subseteq B_{t_1}(R)$ and on every timestep after t_1, R places a fire at the same vertex as S. Then R is also successful.*

▶ **Lemma 13** ($\star$). *Let (G, λ) be a temporal graph with temporal neighbourhood partition $(X_i)_{i \in I}$. Now let S be a strategy that burns this graph, and let u be a vertex at which S places a fire on a timestep t_1, and X_i be the class to which this vertex belongs. Let S' be a strategy which plays as follows until t_2, for any timestep $t_2 > t_1$:*

$$s'_t = \begin{cases} s_t & \text{if } t < t_1 \\ s_{t+1} & \text{if } t_1 \leq t < t_2 \\ u & \text{if } t = t_2 \end{cases}$$

Providing there exists a vertex $w \in X_i$ which is burning before the end of timestep t_1 when S' is played, we have that $B_{t_2}(S) \subseteq B_{t_2}(S')$.

▶ **Lemma 14** ($\star$). *Let (G, λ) be a temporal graph with temporal neighbourhood partition $(X_i)_{i \in I}$. Let S and S' both be strategies with $|S'| = |S|$, such that on every timestep, S and S' both place fires in the same class, that is, for any $i \leq |S|$ we have that $\{s_i, s'_i\} \subseteq X_j$.*

Furthermore, assume that S' places a fire at an already burning vertex on a timestep i if and only if S also places a fire at an already burning vertex on timestep i. Then S is successful if and only if S' is.

▶ **Definition 13** (Placement Classes). *The placement classes for a strategy S denoted $C(S)$ is the set of classes from the temporal neighbourhood partition in which S places fires.*

▶ **Lemma 15** (⋆). *Given a temporal graph (G, λ), let S be any successful strategy. There is then a successful strategy S' with $|S'| = |S|$, and $C(S') = C(S)$, such that the first $|C(S')|$ burns are in distinct equivalence classes in the temporal neighbourhood partition.*

Finally we show that, given a strategy S that places fires only in distinct classes for the first $|C(S)|$ moves, we can arbitrarily reorder all subsequent moves made after timestep $|C(S)|$.

▶ **Lemma 16** (⋆). *Let (G, λ) be a temporal graph, and S a successful strategy such that the first $|C(S)|$ fires placed by S are placed in distinct classes from the temporal neighbourhood partition. Let $f : [|C(S)| + 1, |S|] \to [|C(S)| + 1, |S|]$ be any bijection. Then the strategy S' given by*

$$s'_t = \begin{cases} s_t & \text{if } t \leq |C(S)| \\ s_{f(t)} & \text{otherwise} \end{cases}$$

is successful, and burns the graph in the same or less time as S.

We now present an algorithm for Temporal Graph Burning (Algorithm 1), and show that this algorithm is an fpt-algorithm with respect to temporal neighbourhood diversity.

Algorithm 1 TND Graph Burning Algorithm.

```
Input: A temporal graph $\mathcal{G}$, and an integer $k$.
Output: True if and only if there exists a successful burning strategy of length at most $h$.
1: Compute the temporal neighbourhood partition $\Theta$ of $(G, \lambda)$.
2: for all possible subsets $A \subseteq \Theta$ do
3:     for all possible orderings of $A$ do
4:         for all possible subsets $B \subseteq A$ do
5:             Compute a strategy that first places a fire in order in every class from $A$, and
               then places fires at every unburnt vertex in $B$ in any order.
6:             if this strategy is successful and consists of $k$ or fewer moves then
7:                 return true.
8: If no such strategy is found, return false.
```

▶ **Lemma 17** (⋆). *The TND Graph Burning Algorithm returns true for a temporal graph (G, λ) and integer h if and only if there exists a strategy S that burns the graph in h or fewer timesteps.*

This allows us to obtain fixed parameter tractability, as bounding the temporal neighbourhood diversity bounds the number of such strategies that we have to check.

▶ **Theorem 18** (⋆). *Temporal Graph Burning is solvable in time $O(n^5 \Lambda k! 4^k)$, where n is the size of the input temporal graph $\mathcal{G}$, Λ the lifetime, and k the temporal neighbourhood diversity. If the temporal neighbourhood partition is given, we obtain a runtime of $O(n^2 \Lambda k! 4^k)$.*

4.2 Minimum Reachability Edge Deletion

Here we give another problem which is tractable with respect to temporal neighbourhood diversity. Given a specified source vertex, we seek a minimum set of edge appearances that can be deleted to limit the number of vertices reachable from that source.

We say a vertex v is *temporally reachable* from a vertex u in $\mathcal{G}$ if there exists a temporal path from u to v. We say a vertex v is temporally reachable from a set S if there is a vertex in S from which v is temporally reachable. The *reachability set* $\text{reach}(v)$ of a vertex v is the set of vertices temporally reachable from v. We can now give the formal problem definition; this is a special case of the problem MinReachDelete studied by Molter et al. [35] in which multiple sources are allowed.

Singleton Minimum Temporal Reachability Edge Deletion (SingMinReachDelete)

Input: A temporal graph $\mathcal{G} = (G, \lambda)$, a vertex $v_s \in V(G)$ and positive integer r.
Output: What is the cardinality of the smallest set of time-edges E such that the vertex v_s has temporal reachability at most r after their deletion from $\mathcal{G}$?

SingMinReachDelete was shown by Enright et al. [16] to be NP-hard (and W[1]-hard parameterised by the maximum number of vertices that are allowed to be reached following deletion) even when the lifetime of the input temporal graph is 2 and every edge is active at exactly one timestep. In their problem, they ask if there exists a deletion such that no vertex in the resulting temporal graph reaches more than r vertices. While the result of [16] is for a version of the problem when the source set S is the entire vertex set, it is clear from the construction that hardness also holds with a single source vertex.

We show that this problem is in FPT when parameterised by temporal neighbourhood diversity and the *temporality* of the input graph $\tau(\mathcal{G})$, which was defined by Mertzios et al. [32] to be the maximum number of times any edge appears. When the temporal graph in question is clear from context, we just refer to τ. We note that it remains open whether the problem belongs to FPT parameterised by temporal neighbourhood diversity alone, or indeed parameterised by temporal modular width or temporal cliquewidth; the techniques we use here do not extend naturally to these less restrictive settings.

We now give a formal statement of our result.

▶ **Theorem 19** ($\star$). *SingMinReachDelete is solvable in time $g(k, \tau) \log^{O(1)} r + \Lambda n^3$, where g is a computable function. If a temporal neighbourhood decomposition is given, we can solve the problem in time $g(k, \tau) \log^{O(1)} r$.*

Note that, given two vertices of the same temporal type, their reachability sets must consist of the same vertices except for the vertices themselves; as a result, the reachability sets of vertices in a given class all have the same cardinality. Moreover, all vertices of the same type are first reached from the source at the same time (where we say a vertex is “first reached” at time t if the final time-edge in an earliest-arriving temporal path from the source is at time t). Our strategy for solving SingMinReachDelete is then as follows. We partition each class according to the time at which the vertices are first reached after the deletion of time-edges, and consider all possibilities for which of these subclasses are non-empty. Given a function ϕ telling us which subclasses are non-empty, we argue that we can determine efficiently whether there is indeed a deletion such that precisely these subclasses are non-empty, and if so compute exactly the pairs of subclasses between which we must delete time-edges to achieve this. For a fixed ϕ, we then encode the problem as an instance of Integer Quadratic Programming, where the variables are the sizes of the subclasses and the objective function seeks to minimise the number of time-edges we must delete, and use the algorithm of Lokshtanov [29].

5 Conclusion and Open Questions

We have described three temporal parameters that form a hierarchy mirroring the one formed by their static analogues, and that can all be small when the temporal graph is dense at every timestep. We provide examples of problems demonstrating that there is a separation between the classes of problems admitting efficient algorithms when each of the parameters is bounded. As is the case for the corresponding static parameters, we expect that there will be many problems for which these temporal parameters give fixed-parameter tractability, and suggest exploration of temporal extensions of static problems for which there are known fpt-algorithms as future work. From a practical perspective, it would also be interesting to investigate the values of these new parameters on dense real-world temporal networks.

One of the most celebrated results involving static cliquewidth is a metatheorem due to Courcelle et al. [11] which guarantees the existence of a linear-time algorithm for any problem expressible in a suitable fragment of logic (MSO_1) on graphs of bounded cliquewidth. It is a natural question whether an analogous metatheorem exists for temporal cliquewidth. A promising approach might be to encode a temporal graph as an arbitrary relational structure (as has been done for a temporal version of treewidth [18]). A major challenge here, however, is that to the best of our knowledge there is no single notion of cliquewidth for relational structures: several alternatives have been introduced [1, 12], but none has all of the desirable properties. Moreover, we believe that any encoding of a temporal graph of bounded temporal cliquewidth as a relational structure that preserves all the information in the original is unlikely to have bounded width for any cliquewidth-style measure unless we also bound the lifetime of the temporal graph. Nevertheless, this general direction merits further investigation, and there is potential for a useful metatheorem even if it is necessary to further restrict the fragment of logic considered or the structure of the temporal graph.

References

1 Hans Adler and Isolde Adler. A note on clique-width and tree-width for structures. *CoRR*, abs/0806.0103, 2008. `arXiv:0806.0103`.

2 Eleni C. Akrida, George B. Mertzios, Paul G. Spirakis, and Christoforos Raptopoulos. The temporal explorer who returns to the base. *Journal of Computer and System Sciences*, 120:179–193, September 2021. `doi:10.1016/j.jcss.2021.04.001`.

3 Piotr Berman, Marek Karpinski, and Alex D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. *Electron. Colloquium Comput. Complex.*, TR03-049, 2003. `arXiv:TR03-049`.

4 Stéphane Bessy, Anthony Bonato, Jeannette C. M. Janssen, Dieter Rautenbach, and Elham Roshanbin. Burning a graph is hard. *Discret. Appl. Math.*, 232:73–87, 2017. `doi:10.1016/J.DAM.2017.07.016`.

5 Anthony Bonato, Jeannette C. M. Janssen, and Elham Roshanbin. Burning a graph as a model of social contagion. In Anthony Bonato, Fan Chung Graham, and Pawel Pralat, editors, *Algorithms and Models for the Web Graph - 11th International Workshop, WAW 2014, Beijing, China, December 17-18, 2014, Proceedings*, volume 8882 of *Lecture Notes in Computer Science*, pages 13–22. Springer, 2014. `doi:10.1007/978-3-319-13123-8_2`.

6 Benjamin Merlin Bumpus and Kitty Meeks. Edge Exploration of Temporal Graphs. *Algorithmica*, 85(3):688–716, March 2023. `doi:10.1007/s00453-022-01018-7`.

7 Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding Temporal Paths Under Waiting Time Constraints. *Algorithmica*, 83(9):2754–2802, September 2021. `doi:10.1007/s00453-021-00831-w`.

8 Arnaud Casteigts, Kitty Meeks, George B. Mertzios, and Rolf Niedermeier. Temporal Graphs: Structure, Algorithms, Applications (Dagstuhl Seminar 21171). *Dagstuhl Reports*, pages 16–46, 2021. Publisher: Schloss Dagstuhl – Leibniz Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/DagRep.11.3.16, doi:10.4230/DagRep.11.3.16.

9 Gennaro Cordasco, Luisa Gargano, and Adele A. Rescigno. Iterated Type Partitions. In Leszek Gąsieniec, Ralf Klasing, and Tomasz Radzik, editors, *Combinatorial Algorithms*, Lecture Notes in Computer Science, pages 195–210, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-48966-3_15.

10 Derek G. Corneil, Michel Habib, Jean-Marc Lanlignel, Bruce Reed, and Udi Rotics. Polynomial-time recognition of clique-width ≤ 3 graphs. *Discrete Applied Mathematics*, 160(6):834–865, April 2012. doi:10.1016/j.dam.2011.03.020.

11 B. Courcelle, J. A. Makowsky, and U. Rotics. Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width. *Theory of Computing Systems*, 33(2):125–150, April 2000. doi:10.1007/s002249910009.

12 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Cambridge University Press, 1 edition, June 2012. doi:10.1017/CBO9780511977619.

13 Bruno Courcelle, Joost Engelfriet, and Grzegorz Rozenberg. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2):218–270, April 1993. doi:10.1016/0022-0000(93)90004-G.

14 Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77–114, April 2000. doi:10.1016/S0166-218X(99)00184-5.

15 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-21275-3.

16 Jessica Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119:60–77, August 2021. doi:10.1016/j.jcss.2021.01.007.

17 Michael R. Fellows, Frances A. Rosamond, Udi Rotics, and Stefan Szeider. Clique-Width is NP-Complete. *SIAM Journal on Discrete Mathematics*, 23(2):909–939, January 2009. doi:10.1137/070687256.

18 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. As Time Goes By: Reflections on Treewidth for Temporal Graphs. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms: Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science, pages 49–77. Springer International Publishing, Cham, 2020. doi:10.1007/978-3-030-42071-0_6.

19 Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized Algorithms for Modular-Width. In Gregory Gutin and Stefan Szeider, editors, *Parameterized and Exact Computation*, Lecture Notes in Computer Science, pages 163–176, Cham, 2013. Springer International Publishing. doi:10.1007/978-3-319-03898-8_15.

20 T. Gallai. Transitiv orientierbare Graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(1):25–66, March 1967. doi:10.1007/BF02020961.

21 Robert Ganian. Using Neighborhood Diversity to Solve Hard Problems, January 2012. arXiv:1201.3091 [cs]. doi:10.48550/arXiv.1201.3091.

22 Frank Gurski. A comparison of two approaches for polynomial time algorithms computing basic graph parameters, June 2008. arXiv:0806.4073 [cs]. doi:10.48550/arXiv.0806.4073.

23 Roman Haag, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Feedback edge sets in temporal graphs. *Discrete Applied Mathematics*, 307:65–78, January 2022. doi:10.1016/j.dam.2021.09.029.

24 Michel Habib and Christophe Paul. A survey of the algorithmic aspects of modular decomposition. *Comput. Sci. Rev.*, 4(1):41–59, 2010. doi:10.1016/J.COSREV.2010.01.001.

25 Danny Hermelin, Yuval Itzhaki, Hendrik Molter, and Rolf Niedermeier. Temporal interval cliques and independent sets. *Theoretical Computer Science*, 961:113885, June 2023. doi:10.1016/j.tcs.2023.113885.

26 Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, October 2012. URL: https://www.sciencedirect.com/science/article/pii/S0370157312000841, doi:10.1016/j.physrep.2012.03.001.

27 Yasuaki Kobayashi and Yota Otachi. Parameterized complexity of graph burning. *Algorithmica*, 84(8):2379–2393, 2022. doi:10.1007/S00453-022-00962-8.

28 Michael Lampis. Algorithmic Meta-theorems for Restrictions of Treewidth. *Algorithmica*, 64(1):19–37, September 2012. doi:10.1007/s00453-011-9554-x.

29 Daniel Lokshtanov. Parameterized Integer Quadratic Programming: Variables and Coefficients, April 2017. arXiv:1511.00310 [cs]. doi:10.48550/arXiv.1511.00310.

30 Bernard Mans and Luke Mathieson. On the treewidth of dynamic graphs. *Theoretical Computer Science*, 554:217–228, October 2014. doi:10.1016/j.tcs.2013.12.024.

31 Ross M. McConnell and Jeremy P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1):189–241, April 1999. doi:10.1016/S0012-365X(98)00319-7.

32 George B. Mertzios, Othon Michail, and Paul G. Spirakis. Temporal Network Optimization Subject to Connectivity Constraints. *Algorithmica*, 81(4):1416–1449, April 2019. doi:10.1007/s00453-018-0478-6.

33 George B. Mertzios, Hendrik Molter, Rolf Niedermeier, Viktor Zamaraev, and Philipp Zschoche. Computing maximum matchings in temporal graphs. *Journal of Computer and System Sciences*, 137:1–19, November 2023. doi:10.1016/j.jcss.2023.04.005.

34 Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016. ISBN: 1542-7951 Publisher: Taylor & Francis.

35 Hendrik Molter, Malte Renken, and Philipp Zschoche. Temporal Reachability Minimization: Delaying vs. Deleting. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 76:1–76:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISSN: 1868-8969. doi:10.4230/LIPIcs.MFCS.2021.76.

36 Marc Tedder, Derek Corneil, Michel Habib, and Christophe Paul. Simpler Linear-Time Modular Decomposition Via Recursive Factorizing Permutations. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 634–645, Berlin, Heidelberg, 2008. Springer. doi:10.1007/978-3-540-70575-8_52.

37 Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, January 2016. doi:10.1016/j.tcs.2015.09.030.