Pathfinding in Self-Deleting Graphs

Michal Dvořák
Czech Technical University, Prague, Czech Republic

Dušan Knop
Czech Technical University, Prague, Czech Republic

Michal Opler
Czech Technical University, Prague, Czech Republic

Jan Pokorný
Czech Technical University, Prague, Czech Republic

Ondřej Suchý
Czech Technical University, Prague, Czech Republic

Krisztina Szilágyi
Czech Technical University, Prague, Czech Republic

Abstract

In this paper, we study the problem of pathfinding on traversal-dependent graphs, i.e., graphs whose edges change depending on the previously visited vertices. In particular, we study *self-deleting graphs*, introduced by Carmesin et al. [5], which consist of a graph $G = (V, E)$ and a function $f : V \to 2^E$, where $f(v)$ is the set of edges that will be deleted after visiting the vertex v. In the (Shortest) Self-Deleting s-t-path problem we are given a self-deleting graph and its vertices s and t, and we are asked to find a (shortest) path from s to t, such that it does not traverse an edge in $f(v)$ after visiting v for any vertex v.

We prove that Self-Deleting s-t-path is NP-hard even if the given graph is outerplanar, bipartite, has maximum degree 3, bandwidth 2 and $|f(v)| \le 1$ for each vertex v. We show that Shortest Self-Deleting s-t-path is W[1]-complete parameterized by the length of the sought path and that Self-Deleting s-t-path is W[1]-complete parameterized by the vertex cover number, feedback vertex set number and treedepth. We also show that the problem becomes FPT when we parameterize by the maximum size of $f(v)$ and several structural parameters. Lastly, we show that the problem does not admit a polynomial kernel even for parameterization by the vertex cover number and the maximum size of $f(v)$ combined already on 2-outerplanar graphs.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized complexity, self-deleting graphs, pathfinding

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2025.28

Related Version *Full Version*: `https://arxiv.org/abs/2507.12047`

Funding This work was co-funded by the European Union under the project Robotics and advanced industrial production (reg. no. CZ.02.01.01/00/22_008/0004590). Michal Dvořák and Jan Pokorný acknowledge the support of Grant Agency of the Czech Technical University in Prague, grant No. SGS23/205/OHK3/3T/18.

1 Introduction

Pathfinding in graphs is a well-studied topic, both from a theoretical and from a practical perspective. The famous Dijkstra's algorithm for finding a shortest path between two vertices runs in time that is quadratic in the number of vertices. However, this algorithm works under the assumption that the underlying graph is static, i.e., does not change. In many practical applications, this assumption does not hold.

One way to reflect the changes in the underlying graph is to model it as a temporal graph, where each edge is present in the graph only at certain times. In this setting, there are several ways to define the "best" path between two vertices: shortest path (using the smallest number of edges), fastest (the path that requires the smallest number of timesteps) and foremost (the path that requires the smallest number of timesteps starting from time zero). In all of these cases, the pathfinding problem can be solved in polynomial time [23].

In some applications, however, the way that the underlying graph changes is *traversal-dependent*. In other words, the availability of an edge is not time-dependent, but it depends on the previously visited vertices and edges. One such example is open-pit mining, where drilling at a vertex creates a pile of rubble which renders some edges impassable. Another example is autonomous harvesting, in which the vehicle should not return to the previously visited areas to avoid soil compactification [5].

In this paper, we consider the model introduced by Carmesin et al. [5], called *self-deleting graph*. A self-deleting graph is a graph $G = (V, E)$ together with a function $f \colon V \to 2^E$, which describes which edges will be deleted after visiting a vertex. We stress that the edges in $f(v)$ are not necessarily incident to v. We consider the (Shortest) Self-Deleting s-t path problem, where we are given a self-deleting graph and its vertices s and t and we are asked to find a (shortest) path from s to t that is valid (i.e., in which we are not traversing an edge in $f(v)$ after visiting the vertex v).

Related work. Several variants of pathfinding with restrictions on vertices and edges have been studied.

Wojciechowski et al. [21] introduced a problem called Optional Choice Reachability, where we are given a graph together with a set S of pairs of edges, and we are asked to find an s-t path that contains at most one edge from each pair in S. They showed that this problem is NP-complete even on directed acyclic graphs (DAGs) of pathwidth 2 and FPT parameterized by $|S|$.

The vertex analogue of this problem, where we are given a set of pairs of vertices and we are required to use at most one of them from each pair was introduced by Krause et al. [19]. On the one hand, the problem is NP-hard even for DAGs [13], even if the set of pairs has a specific structure, such as overlapping [17] or ordered [18]. On the other hand, it is polynomial time solvable, if the structure is well-parenthesized [17], halving [17], or nested [9], or in other special cases [24]. Notably, Bodlaender et al. [3] studied the problem from parameterized perspective on undirected graphs, showing that it is W[1]-hard w.r.t. vertex cover number of the input graph G, but FPT w.r.t. vertex cover number of graph H or treewidth of graph $G \cup H$, where graph H has an edge for each forbidden pair. Moreover, it does not admit polynomial kernel w.r.t. vertex cover number of graph $G \cup H$, unless $\mathsf{NP} \subseteq \mathsf{coNP}/_{\text{poly}}$ [3].

Szeider [20] studied the problem of finding a path where for each vertex, only certain combinations of incoming and outgoing edges are permitted. He provides a dichotomy between the cases which are NP-hard and cases which are linear time solvable, based on the structure of permitted combinations. In the variant where the cost of each arc depends on previously traversed arcs, introduced by Kirby and Potts [16], the shortest walk can be found in polynomial time [2, 25], but finding the shortest path is NP-hard and hard to approximate [22]. In the variant with exclusive-disjunction arc pairs conflicts, introduced by Cerulli et al. [6], we are given pairs of arcs and we have to pay a penalty if the path contains either none of the arcs or both of them. The goal is to find a path minimizing the sum of length of edges and the penalties paid. They show that the problem is NP-hard and provide heuristics.

Our results. In Section 3, we consider the (classical) complexity of (Shortest) Self-Deleting s-t-path. Our first result is that Self-Deleting s-t-path is NP-hard even on a very restricted graph class, namely outerplanar bipartite graphs of maximum degree 3. Moreover, as we show in Corollary 3.4, the result holds even when the self-deleting graph deletes at most one edge for each vertex (i.e., $|f(v)| \leq 1$ for all v). Next, we show a separation between Self-Deleting s-t-path and Shortest Self-Deleting s-t-path: namely, on cactus graphs the former problem can be solved in linear time, whereas the latter is NP-hard (Theorems 3.10 and 3.11 respectively).

In Section 4, we turn our attention to parameterized complexity. Firstly, we consider the parameterization of Shortest Self-Deleting s-t-path by solution size, and in Theorem 4.3 we show that it is W[1]-complete. For most structural parameters, Self-Deleting s-t-path turns out to be para-NP-hard or W[1]-complete. For an overview of our results on parameterizations by structural parameters, see Figure 1.

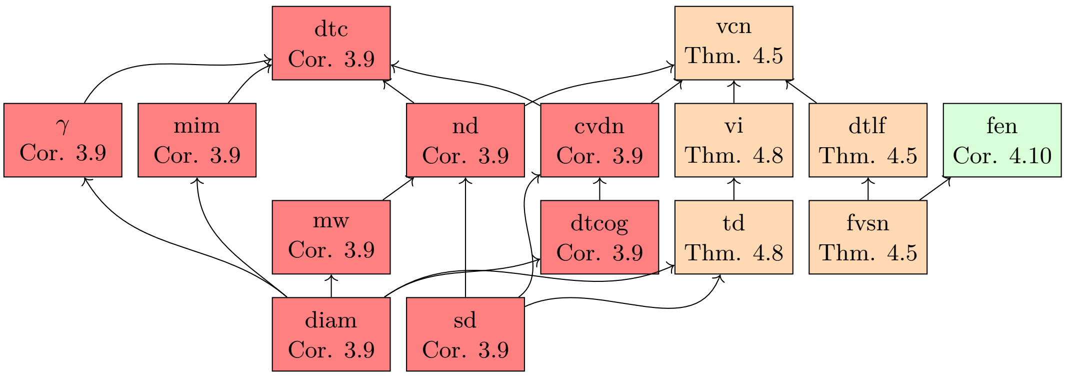

Figure 1 Results for Self-Deleting s-t-path parameterized by structural parameters. Red color stands for para-NP-hard, green is FPT and orange is W[1]-complete. Each parameter is accompanied by the corresponding statement from which the result follows. An arrow $\alpha \to \beta$ indicates a functional upper bound, i.e., $\alpha \leq g(\beta)$ for some (computable) function g. For full names of the parameters refer to Section 2.

In order to make the problem tractable, we consider the case of bounded deletion set size (i.e. we parameterize the problem by $\mu = \max_{v \in V} |f(v)|$). Although this parameterization alone does not make the problem tractable (in particular, we show that Shortest Self-Deleting s-t-path is para-NP-hard parameterized by μ), it turns out that the parameterization by μ and k (the number of vertices of the sought path) leads to an FPT algorithm. This result allows us to obtain several FPT algorithms for parameterizations by μ and structural parameters, such as vertex cover number, vertex integrity and treedepth. Using the results of Dvořák et al. [12] on the number of edges in traceable graphs with dense structure, we obtain FPT algorithms for several dense parameters and μ, such as neighborhood diversity, shrub-depth, modular-width, and size of maximum induced matching. For an overview of our results about parameterizations by structural parameters and μ, refer to Figure 2.

Lastly, in Section 5, we consider kernelizing the problem. We show that, under standard parameterized complexity assumptions, Self-Deleting s-t-path does not admit a polynomial kernel parameterized by μ and vertex cover number even on 2-outerplanar graphs. On the positive side, we obtain a linear kernel for the parameterization by feedback edge number and linear kernel for the parameterization by vertex cover number on outerplanar graphs. Moreover, while Self-Deleting s-t-path does not admit a classical kernel parameterized by μ on cliques, we obtain a linear Turing kernel on cliques parameterized by μ.

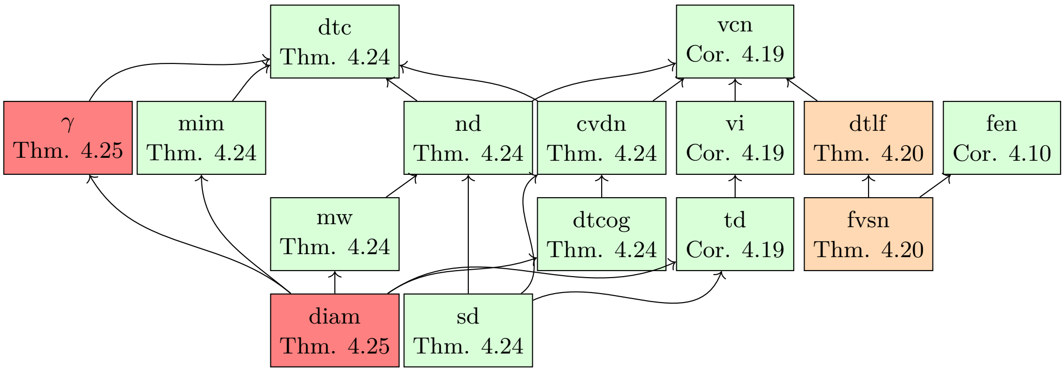

Figure 2 Results for Self-Deleting s-t-path parameterized by structural parameters and μ combined. The meaning of colors and arrows is the same as in Figure 1.

2 Preliminaries

For an integer k we let $[k] = \{1, 2, \dots, k\}$. For any function $f \colon A \to B$ and $X \subseteq A$, the restriction of f to X is denoted $f|_X$. We use the notation $\widetilde{O}(\cdot)$ to suppress polylogarithmic factors in time complexity.

Graph Theory. We use standard notations, terminology, and definitions of graph theory, refer to Diestel [11] for those. Unless explicitly stated, we consider simple undirected graphs. We denote a v_1–v_k walk (or path) in a graph by the sequence $(v_1, e_1, v_2, \dots, e_{k-1}, v_k)$ of vertices v_i and edges e_i, where $e_i = \{v_i, v_{i+1}\}$. For brevity, we may sometimes omit the edges and write just the sequence of vertices. We denote by $\mathrm{cc}(G)$ the number of connected components of a graph G. A graph is a *cactus* if every edge lies on at most one cycle. A graph is a *block graph* if every vertex-2-connected component is a clique. A $2 \times \ell$ grid for some positive integer ℓ is a *ladder*. The class of *cographs* is the minimal class of graphs containing the one-vertex graph and closed under complete joins and disjoint unions of two graphs.

Self-deleting graphs. A *self-deleting graph* is an ordered pair (G, f), where $G = (V, E)$ is an undirected graph and f is a function $f \colon V \to 2^E$. We write G instead of (G, f) if f is clear from the context. The function f is called the *deletion function*. Let (G, f) be a self-deleting graph. A path or walk $(v_1, e_1, v_2, e_2, \dots, e_{k-1}, v_k)$ in G is *f-conforming* if for every $j \le i$ we have $e_i \notin f(v_j)$. We denote $\mu_f = \max_{v \in V} |f(v)|$ and $|f| = \sum_{v \in V} |f(v)|$. If f is clear from the context, we omit the lower index and write just μ. Central to our paper are the two decision problems Self-Deleting s-t-path and Shortest Self-Deleting s-t-path. In Self-Deleting s-t-path we are given a self-deleting graph (G, f) and two vertices $s, t \in V(G)$ and the task is to decide if there is an f-conforming s-t path in G. In Shortest Self-Deleting s-t-path we are in addition given a positive integer k and the task is to decide if there is an f-conforming s-t path in G on at most k vertices.

We assume that $\mu \ge 1$ as otherwise the problems are trivial. Moreover, if μ appears in the base of an exponential or in a logarithm, μ should be replaced by $\max\{\mu, 2\}$ in order to avoid degenerate cases. We stick to writing μ for the sake of readability.

Parameterized complexity. We assume the reader is familiar with parameterized complexity. For definitions and comprehensive overview, refer to the monograph of Cygan et al. [10]. In our work we consider the following structural parameters: *bandwidth* (bw), *distance to* *clique* (dtc), *vertex cover number* (vcn), *domination number* (γ), *maximum induced matching* (mim), *neighborhood diversity* (nd), *diameter* (diam), *cluster vertex deletion number* (cvdn), *feedback vertex set number* (fvsn), *distance to linear forest* (dtlf), *vertex integrity* (vi), *feedback edge number* (fen), *treedepth* (td), *distance to cograph* (dtcog), *modular-width* (mw), and *shrub-depth* (sd). Formal definitions of all the parameters can be found in the full version of this paper.

Exponential Time Hypothesis. Exponential Time Hypothesis (ETH), introduced by Impagliazzo, Paturi and Zane [14, 15] asserts, roughly speaking, that there is no algorithm for 3SAT in time $2^{o(n')}$, where n' is the number of variables of the input formula. In fact, even $2^{o(n'+m')}$ algorithm is ruled out by using the Sparsification Lemma [15], where m' is the number of clauses of the input formula. We also utilize the result of Chen, Huang, Kanj, and Xia [7, 8] that there is no $g(k)n^{o(k)}$ algorithm for (MULTICOLORED) CLIQUE for any computable function g unless ETH fails.

Statements where proofs or details are omitted due to space constraints are marked with $\star$. The omitted material is available in the full version of this paper.

3 Classical Complexity

As observed by Carmesin et al. [5, Lemma 3], if every vertex deletes only its incident edges, SELF-DELETING s-t-PATH reduces to pathfinding in a directed graph: deleting $\{u, v\}$ at v orients the edge from u to v, and if both endpoints delete it, the edge is removed entirely. Thus, SELF-DELETING s-t-PATH is solvable in linear time when all $f(v)$ contain only edges incident to v. Another scenario when the problem is tractable is that the graph has only a constant number of s-t paths, e.g., in trees or graphs of maximum degree 2. The problem becomes hard already on graphs of maximum degree 3. We show a polynomial-time reduction from 3SAT to SELF-DELETING s-t-PATH which we also utilize later with slight modifications.

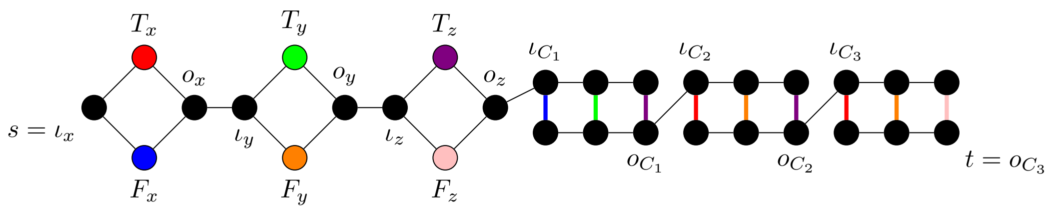

Figure 3 Example of Construction 3.1 for the input formula $\varphi = (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee y \vee z)$. The deletion sets are indicated by the colors.

▶ **Construction 3.1.** Let φ be a given CNF-formula over variables $X = \{x_1, \ldots, x_{n'}\}$ with clauses $\mathcal{C} = \{C_1, \ldots C_{m'}\}$. Our construction of an instance of SELF-DELETING s-t-PATH consist of variable and clause gadgets that we now introduce. See Figure 3 for an example.

Variable gadget The variable gadget for a variable $x \in X$ is a 4-cycle with vertices T_x, ι_x, F_x, o_x.

Clause gadget The clause gadget for a clause $C \in \mathcal{C}$ is the $2 \times |C|$ grid. The top left corner vertex is denoted ι_C (input) and the bottom right corner vertex is denoted o_C (output). Each of the $|C|$ vertical edges correspond to a literal of C. For a literal $\ell \in C$ we denote its corresponding edge by e^C_ℓ.

Create one variable gadget for each variable and one clause gadget for each clause. Add edges of the form $\{o_{x_k}, \iota_{x_{k+1}}\}$ for every $k \in [n'-1]$, add edge $\{o_{x_{n'}}, \iota_{C_1}\}$ and finally the edges $\{o_{C_i}, \iota_{C_{i+1}}\}$ for every $i \in [m'-1]$. We let $s = \iota_{x_1}$ and $t = o_{C_{m'}}$. It remains to specify the deletion function f. For all vertices v except T_x, F_x inside the variable gadgets, we set $f(v) = \emptyset$. For a variable $x \in X$ we set $f(T_x) = \{e^C_{\neg x} \mid C \in \mathcal{C}, \neg x \in C\}$ and $f(F_x) = \{e^C_x \mid C \in \mathcal{C}, x \in C\}$. In other words, T_x deletes the literal edges in clause gadgets corresponding to literals $\neg x$ and F_x deletes the literal edges corresponding to literals x. The resulting Self-Deleting s-t-path instance is (G, f, s, t).

▶ **Lemma 3.2** (⋆). *Let φ be the formula and (G, f, s, t) the Self-Deleting s-t-path instance obtained by Construction 3.1 from φ. Then φ is satisfiable if and only if there is an f-conforming s-t path in G.*

By using Construction 3.1 together with Lemma 3.2 we immediately obtain the following theorem.

▶ **Theorem 3.3.** *Self-Deleting s-t-path is* NP*-hard.*

▶ **Corollary 3.4.** *Self-Deleting s-t-path is* NP*-hard even if the deletion function f satisfies $\forall v \in V : |f(v)| \le 1$, i.e., $\mu \le 1$.*

Proof. Use Construction 3.1 with the following modification. Replace the vertex T_x (resp. F_x) by a path on $|f(T_x)|$ (resp. $|f(F_x)|$) vertices and delete one edge of the original set $f(v)$ on each vertex of the path to obtain $|f(v)| \le 1$. ◀

▶ Remark 3.5. Note that Construction 3.1 produces a graph with $O(n' + m')$ vertices and edges, where n' and m' are the number of variables and clauses in the original 3Sat formula.

▶ **Lemma 3.6** (⋆). *$2 \times \ell$ grid has bandwidth 2, treedepth at most $O(\log \ell)$ and vertex integrity at most $O(\sqrt{\ell})$.*

Since the resulting graph of Construction 3.1 is a subgraph of a ladder of size $O(n' + m')$, it is straightforward to obtain the following corollary:

▶ **Corollary 3.7.** *Self-Deleting s-t-path is* NP*-hard even when restricted to outerplanar bipartite graphs of maximum degree 3 and bandwidth 2 even with $\mu \le 1$.*

▶ **Corollary 3.8** (⋆). *Self-Deleting s-t-path remains* NP*-hard even when restricted to the classes of unit interval graphs, ladder graphs, or block graphs even when $\mu \le 1$.*

We can also easily obtain hardness on cliques by adding all non-existent incident edges to the deletion set of every vertex. However, as we will see later, we cannot hope to upper bound μ by a constant in cliques and preserve NP-hardness (Observation 4.21 and Corollary 5.3).

▶ **Corollary 3.9** (⋆). *Self-Deleting s-t-path is* NP*-hard even when restricted to cliques.*

Cactus graphs. Corollaries 3.7 and 3.8 show that even if we allow a slight generalization of trees, the problem becomes NP-hard. One particular non-trivial case that allows for a polynomial-time algorithm is the class of cactus graphs. We show that when restricted to cactus graphs, Self-Deleting s-t-path becomes polynomial-time solvable (Theorem 3.10). Interestingly, deciding an existence of an f-conforming path is easy, however Shortest Self-Deleting s-t-path remains NP-hard even in cactus graphs (Theorem 3.11). Note that in cactus graphs each edge lies on at most 1 cycle, whereas the graph resulting from Construction 3.1 has each edge on at most 2 cycles so the problem becomes NP-hard in this case.

▶ **Theorem 3.10** (⋆). *Self-Deleting s-t-path can be solved in linear time if the underlying graph is a cactus.*

Proof Sketch. Consider the block-cut tree of G and note that we can only restrict ourselves to the part of G that contains the s-t path in the block-cut tree of G. This part consists of cycles and bridges. In each cycle we have two options to choose from. Each choice possibly forbids some choices in the future. This can be encoded as an implication of the type *if we choose this path, then we cannot choose some other path in the future.* We can thus reduce the problem to 2Sat. Details can be found in the full version of this paper. ◀

▶ **Theorem 3.11** (⋆). *Shortest Self-Deleting s-t-path is* NP*-hard even when restricted to cactus graphs, and $\mu \leq 1$.*

Proof Sketch. We provide a polynomial reduction from Independent Set. For each vertex we introduce one cycle on the designated s-t-path (in the block-cut tree). One of the paths corresponds to taking the vertex and the other to not taking it. Inclusion of a vertex forbids inclusion of any of its neighbors by deleting an edge of the appropriate path. Crucially, the inclusion path is shorter, so the desired length of the path forces an appropriate number of vertices to be included in the independent set. ◀

4 Parameterized complexity

In this section we focus on parameterized complexity of (Shortest) Self-Deleting s-t-path. Note that the problem is para-NP-hard parameterized by bandwidth or maximum degree (hence also parameterized by treewidth) due to Corollary 3.7. Moreover, it is also para-NP-hard parameterized by any parameter that is constant on cliques (Corollary 3.9).

We show that Shortest Self-Deleting s-t-path is W[1]-complete parameterized by k (the number of vertices of the sought path) (Theorem 4.3) and that Self-Deleting s-t-path is W[1]-hard parameterized by the vertex cover number. We then show that Self-Deleting s-t-path is in fact W[1]-complete for parameters vertex cover number, vertex integrity, treedepth, distance to linear forest, and feedback vertex set number (Theorems 4.5 and 4.8).

The last hope for positive algorithmic results lies in the parameter feedback edge number (fen). Here, we observe that Self-Deleting s-t-path parameterized by fen can be solved in $O(2^{\text{fen}}(n+m+|f|))$ time (Corollary 4.10). Later, in Section 5 we show that Self-Deleting s-t-path even admits kernel with $O(\text{fen})$ vertices and edges (however, this does not yield as fast algorithm). Refer to Figure 1 for a graphical overview of the results for structural parameters alone.

Then, in order to obtain algorithmic results, we combine structural parameters with the parameter $\mu = \max_{v \in V} |f(v)|$. While Shortest Self-Deleting s-t-path is para-NP-hard parameterized by μ alone (Corollary 3.4), and W[1]-complete parameterized by k (Theorem 4.3), the problem becomes FPT parameterized by k and μ combined (Theorem 4.17). This also yields many FPT results for Self-Deleting s-t-path parameterized by structural parameters and μ combined (Theorem 4.24). Refer to Figure 2 for an overview of such results.

4.1 W[1]-completeness for length of the path

We begin with a reduction from Multicolored Clique to Self-Deleting s-t-path. We developed this construction independently, although later we discovered that it is similar to the one of Bodlaender et al. [3, Theorem 9] for a related problem.

▶ **Construction 4.1.** Let $G = (V, E)$ be the input graph for MULTICOLORED CLIQUE and let $V = V_1 \cup \cdots \cup V_k$ be the partition of V into k color classes. We build an instance (G', f, s, t) of SELF-DELETING s-t-PATH as follows. Create $k+1$ guard vertices $g_0, g_1, \ldots, g_k$. For every $i \in [k]$ and $v \in V_i$ add a vertex y_v and connect it to g_{i-1} and g_i by edges $e_1^v = \{g_{i-1}, y_v\}$ and $e_2^v = \{y_v, g_i\}$. We denote by P_v^i the path $(g_{i-1}, e_1^v, y_v, e_2^v, g_i)$. This completes the description of the graph G'. We let $s = g_0$ and $t = g_k$. It remains to specify the deletion sets. The only vertices with nonempty deletion sets will be the vertices y_v on the paths P_v^i. We let $f(y_v) = \bigcup\{e_1^w, e_2^w \mid w \in V_j, j \neq i, \{w, v\} \notin E(G)\}$.

▶ **Lemma 4.2** (⋆)**.** *Let $(G, V_1, \ldots, V_k)$ be an instance of MULTICOLORED CLIQUE and (G', f, s, t) be the instance of SELF-DELETING s-t-PATH obtained from it by Construction 4.1. There is a multicolored clique in G if and only if there is an f-conforming s-t path in G'.*

Note that in the instance (G', f, s, t) from Construction 4.1, any f-conforming s-t path has at most $2k+1$ vertices. We immediately obtain W[1]-hardness of SHORTEST SELF-DELETING s-t-PATH w.r.t. k.

▶ **Theorem 4.3** (⋆)**.** *SHORTEST SELF-DELETING s-t-PATH is* W[1]*-complete w.r.t. k.*

4.2 Parameterization by structural parameters alone

Observe that the resulting graph G' from Construction 4.1 has vertex cover number at most $k+1$, because $G' \setminus \{g_0, \ldots, g_k\}$ is edgeless. We immediately obtain W[1]-hardness of SELF-DELETING s-t-PATH for the parameter vertex cover number.

We establish membership in W[1] of SELF-DELETING s-t-PATH parameterized by the feedback vertex set number (Theorem 4.4) and treedepth (Observation 4.7). These results together with W[1]-hardness for vertex cover number establish W[1]-completeness for the following parameters: vertex cover, distance to linear forest, feedback vertex set, vertex integrity, and treedepth (Theorems 4.5 and 4.8).

▶ **Theorem 4.4** (⋆)**.** *SELF-DELETING s-t-PATH parameterized by the feedback vertex set number is in* W[1].

Proof Sketch. If G has a feedback vertex set S, then any path in G is split by S into at most $|S| + 1$ segments where the segments are uniquely determined by their endpoints, as $G \setminus S$ is a forest. We can thus equivalently look for a path of length $O(\text{fvsn})$ in a graph where we represent long paths in $G \setminus S$ by paths of length two and reflect the deletions along these unique u-v paths in $G \setminus S$. Full proof can be found in the full version of this paper. ◀

▶ **Theorem 4.5** (⋆)**.** *SELF-DELETING s-t-PATH parameterized by feedback vertex set, distance to linear forest, or vertex cover is* W[1]*-complete, solvable in $n^{O(\alpha)}$ time and unless ETH fails, there is no $g(\alpha)n^{o(\alpha)}$ algorithm for any of the above parameters and any computable function g.*

▶ **Lemma 4.6** (⋆)**.** *Let G be a graph with treedepth* td *and vertex integrity* vi*. Then G contains no path on more than 2^{td} or $\text{vi}^2 + 2\,\text{vi}$ vertices.*

▶ **Observation 4.7.** *SELF-DELETING s-t-PATH parameterized by treedepth is in* W[1].

Proof. We provide a parameterized reduction from SELF-DELETING s-t-PATH parameterized by treedepth to SELF-DELETING s-t-PATH parameterized by k, which is in W[1] by Theorem 4.3. Let (G, f, s, t) be an instance of SELF-DELETING s-t-PATH, we return the instance (G, f, s, t, k) where $k = 2^{\text{td}(G)}$. Correctness follows from Lemma 4.6. ◀

▶ **Theorem 4.8** ($\star$). *Self-Deleting s-t-path is* W[1]*-complete parameterized by treedepth or vertex integrity. More precisely, Self-Deleting s-t-path can be solved in $n^{O(2^{\mathrm{td}})}$ and $n^{O(\mathrm{vi}^2)}$ time. For any $\varepsilon > 0$, algorithms for Self-Deleting s-t-path with running times $n^{O(\mathrm{vi}^{2-\varepsilon})}\operatorname{poly}(n)$ or $n^{2^{o(\mathrm{td})}}\operatorname{poly}(n)$ violate ETH.*

We complement the hardness results by a positive result for the parameter feedback edge number. Note that given a path P in a self-deleting graph, it is easy to check whether P is f-conforming in time $O(n + m + |f|)$.

▶ **Lemma 4.9** ($\star$). *Let G be a graph and $s, t \in V(G)$ two fixed vertices in G. Then the number of s-t-paths in G is at most $2^{\mathrm{fen}(G)}$.*

▶ **Corollary 4.10** ($\star$). *Self-Deleting s-t-path can be solved in $O(2^{\mathrm{fen}(G)}(n + m + |f|))$ time. Moreover, $2^{o(\mathrm{fen})}\operatorname{poly}(n)$-time algorithm for Self-Deleting s-t-path violates ETH.*

4.3 FPT algorithm for k and μ combined

We demonstrate that Shortest Self-Deleting s-t-path parameterized by k and μ combined is in FPT. We utilize color-coding, introduced by Alon et al. [1]. We consider a colorful variant of the problem, where we consider coloring of the edges and we only distinguish edges based on their colors. The sought solution – an f-conforming path $P = (v_1, e_1, \ldots, e_{r-1}, v_r)$ on $r \le k$ vertices interacts with the edges e_i in the path and with the edges in the deletion sets $f(v_1), \ldots, f(v_r)$. By assumption there are at most $\mu k + k - 1$ such edges in total. Hence if we can ensure that the coloring will behave nicely on the set $\{e_1, \ldots, e_{r-1}\} \cup \bigcup_{i=1}^{r} f(v_i)$, we will find the path even in the colorful variant. We now formally define the colorful variant of our problem, which we refer to as Shortest χ-compliant s-t-path.

▶ **Definition 4.11.** *Let (G, f) be a self-deleting graph and let $\chi\colon E(G) \to [q]$ be a coloring of its edges. Let $P = (v_1, e_1, \ldots, e_{r-1}, v_r)$ be a path in G. We say that*

1. *P is* χ-compliant *if $\chi(e_i) \notin \chi(f(v_j))$ for any $j \le i$,*
2. *P is* half-χ-rainbow *if $\chi(\bigcup_{i=1}^{r} f(v_i) \setminus E(P)) \cap \chi(E(P)) = \emptyset$ and $\chi|_{E(P)}$ is injective.*
3. *P is* χ-rainbow *if for $F = E(P) \cup \bigcup_{i=1}^{r} f(v_i)$ the restriction $\chi|_F$ is injective.*

In the Shortest χ-compliant s-t-path problem we are given a self-deleting graph (G, f), positive integer k, coloring $\chi\colon E(G) \to [q]$, vertices $s, t \in V(G)$ and the task is to decide whether there is a χ-compliant s-t path on at most k vertices in G.

▶ **Lemma 4.12** ($\star$). *The following holds for any path P:*

1. *P is χ-rainbow $\Rightarrow$ P is half-χ-rainbow.*
2. *P is half-χ-rainbow and f-conforming $\Rightarrow$ P is χ-compliant.*
3. *P is χ-compliant $\Rightarrow$ P is f-conforming.*

▶ **Lemma 4.13** ($\star$). *Given an instance of Shortest χ-compliant s-t-path and a set of colors $Q \subseteq [q]$, we can in $O(2^{|Q|}(n + m + |f|))$ time decide whether there exists a χ-compliant path on at most k vertices using only colors from Q.*

▶ **Lemma 4.14.** *Shortest χ-compliant s-t-path can be solved in $O(2^q(n + m + |f|))$ or in $O(\binom{q}{k}2^k(n + m + |f|))$ time.*

Proof. We can either plug $Q = [q]$ into the algorithm of Lemma 4.13 and obtain the running time $O(2^q(n + m + |f|))$ or we can try all possible sets Q of $k - 1$ colors and obtain the running time $O(\binom{q}{k}2^k(n+m+|f|))$. Note that we seek a path on (at most) k vertices, hence (at most) $k - 1$ edges. ◀

▶ **Theorem 4.15** (⋆). *There is a randomized algorithm solving* Shortest Self-Deleting s-t-path *with the following guarantees. Given $\varepsilon \in (0, 1)$, it runs in $2^{O(k \log \mu)}(n + m) \ln \frac{1}{\varepsilon}$ time. Moreover, if the input is a no-instance, the algorithm outputs no. If the input is a yes-instance, the algorithm outputs yes with probability at least $1 - \varepsilon$.*

Proof sketch. Asume that $\mu \geq 1$ as otherwise the problem is trivial. We use the algorithm from Lemma 4.14 with running time $O(\binom{q}{k}2^k(n + m + |f|))$. We try a random edge coloring χ using $q = 4k\mu$ colors. For a fixed coloring using q colors, we lower bound the probability that a path P is χ-compliant, given that P is f-conforming. Thus, we are equivalently lower bounding the probability that the algorithm succeeds in finding the f-conforming path. Let $P = (v_1, e_1, \ldots, e_{r-1}, v_r)$. We lower bound the probability that P is half-χ-rainbow. Suppose that the edges in $\bigcup_{i=1}^{r} f(v_i)$ are colored by $w \leq |\bigcup_{i=1}^{r} f(v_i) \setminus E(P)| \leq r\mu \leq k\mu$ colors from the set $[q]$. For P to become half-χ-rainbow, we need to ensure that the edges in $E(P)$ are colored by one of the remaining $q - w \geq 4k\mu - k\mu = 3k\mu$ colors and that the coloring is injective on $E(P)$. The probability of that happening for the $r - 1$ edges is $\frac{q-w}{q} \cdot \frac{q-w-1}{q} \cdot \ldots \cdot \frac{q-w-r+2}{q}$. Note that the last term lower bounds every other term and the last term is lower bounded by $\frac{1}{2}$ because $\frac{q-w-r+2}{q} \geq \frac{3k\mu-k}{4k\mu} \geq \frac{1}{2}$ given that $\mu \geq 1$. Hence the probability that P is half-χ-rainbow is at least $2^{-r+1} \geq 2^{-k}$.

We repeat the above for $2^k \ln \frac{1}{\varepsilon}$ random choices of χ and the proof on the probability of error and running time follow. The full proof can be found in the full version of this paper. ◀

There are two ways to derandomize the algorithm given in Theorem 4.15. Neither of the two algorithms is Pareto optimal with respect to μ and k. Hence, each of them turns out to be more suitable in different scenarios.

▶ **Theorem 4.16** (⋆). Shortest Self-Deleting s-t-path *is solvable in $2^{O(\mu k)}(n+m) \log n$ time.*

▶ **Theorem 4.17** (⋆). Shortest Self-Deleting s-t-path *is solvable in $2^{O(k \log(k\mu))}(n + m) \log n$ time.*

4.4 Parameterization by structural parameters and μ combined

We utilize the FPT algorithms w.r.t. k and μ combined to obtain several FPT algorithms for various structural parameters combined with μ. In the results that follow, we apply whichever of Theorem 4.17 or Theorem 4.16 yields the better asymptotic running time for the given parameters.

We start with the vertex cover number for which we prove similar bound as in Lemma 4.6.

▶ **Observation 4.18** (⋆). *Let G be a graph with vertex cover number* vcn*. Then G contains no path on more than $2\,\mathrm{vcn}+1$ vertices.*

▶ **Corollary 4.19** (⋆). *Self-Deleting s-t-path can be solved in $2^{O(\mu\cdot \mathrm{vcn})}(n+m)\log n$ time, in $2^{O(\mu\cdot \mathrm{vi}^2)}(n+m)\log n$ time, or in $2^{O(\mu\cdot 2^{\mathrm{td}})}(n+m)\log n$ time. Moreover, algorithms with running times $2^{o(\mathrm{vcn})}\operatorname{poly}(n)$, $2^{o(\mathrm{vi}^2)}\operatorname{poly}(n)$, or $2^{2^{o(\mathrm{td})}}\operatorname{poly}(n)$ for Self-Deleting s-t-path even for $\mu = 1$ violate ETH.*

Unless FPT = W[1], the FPT algorithm for vertex cover cannot be extended already to the parameter distance to linear forest.

▶ **Theorem 4.20** (⋆). *Self-Deleting s-t-path is* W[1]*-complete parameterized by the distance to linear forest, even if $\mu \le 1$.*

Dense parameters and μ combined

In the remainder of this section we focus on parameters whose bounded values together with the existence of a long path imply dense structure of the graph in some sense. As a warmup example, recall that Self-Deleting s-t-path is NP-hard on cliques (Corollary 3.9). Observe that there is always an f-conforming path on at most $\mu + 2$ vertices when the underlying graph is a clique, if there is any f-conforming path at all. This is because the vertex s cannot delete more than μ edges, so if the path is longer, then there is a shortcut that we can take. By plugging this upper bound into Theorem 4.17, we obtain the following:

▶ **Observation 4.21.** *Self-Deleting s-t-path is solvable in $2^{O(\mu\log\mu)}(n+m)\log n$ time on cliques.*

We build upon this idea. In general, vertices of a k-vertex path can delete at most $k\mu$ edges. On the other hand, if we consider only shortest (f-conforming) paths, any shortcut edge on the vertices of the path must be deleted as otherwise the path can be shortened, contradicting that it is the shortest path. We thus obtain the following:

▶ **Observation 4.22.** *Let $(G = (V, E), f)$ be a self-deleting graph and let $P = (v_1, v_2, \ldots, v_k)$ be a shortest f-conforming v_1-v_k path in (G, f). Let $G_P = G[V(P)]$ be the subgraph of G induced by the vertices of P. Then $|E(G_P)| \le k\mu + k - 1 \le k(\mu + 1)$.*

We now utilize the results of Dvořák et al. [12]. They provide a lower bound on the number of edges in the input graph, given that it contains long (Hamiltonian) path. Recall that graphs containing a Hamiltonian path are also called *traceable*. By combining these bounds together with Observation 4.22, we obtain an upper bound on the length of a shortest f-conforming s-t path in the underlying graph in terms of μ and some structural parameter α, see Table 1 for an overview. Recall that $\widetilde{O}$ supresses polylogarithmic factors.

▶ **Lemma 4.23.** *Let α be a graph parameter monotone under taking induced subgraphs. Suppose there are global constants C, D such that any traceable graph G with $n \ge C\cdot\alpha(G)$ vertices contains at least $D\frac{n^2}{\alpha(G)}$ edges. Then Self-Deleting s-t-path can be solved in $2^{\widetilde{O}(\alpha\mu)}(n+m)\log n$ time. In particular it is* FPT *parameterized by α and μ combined.*

Proof. Let (G, f, s, t) be the input instance of Self-Deleting s-t-path and let P be a shortest f-conforming s-t path in G. Let $k = |V(P)|$ and let $G_P = G[V(P)]$ be the graph induced by vertices of P. By assumption on α, either $k < C\cdot\alpha(G_P) \le C\cdot\alpha(G)$ (because α is monotone), or $|E(G_P)| \ge D\frac{k^2}{\alpha(G_P)}$. By Observation 4.22 we also have $|E(G_P)| \le k(\mu+1)$. By combining these two bounds we obtain the bound $k \le \frac{1}{D}\alpha(G_P)(\mu+1) \le \frac{1}{D}\alpha(G)(\mu+1)$. By plugging $k = \max\{C\cdot\alpha(G), \frac{1}{D}\alpha(G)\cdot(\mu+1)\}$ into Theorem 4.17 we obtain the promised algorithm with running time $2^{O(\alpha\mu\log(\alpha\mu))}(n+m)\log n$. ◀

Table 1 Overview of the framework for parameterization by structural parameters α and μ combined. The lower bound column is the asymptotic lower bound on the number of edges for traceable graph with given parameter bounded by α proven by Dvořák et al. [12]. The third column indicates what is the implied upper bound on the length of any shortest f-conforming path in such graphs. The fourth column is the final running time of the algorithm using the better from Theorems 4.16 and 4.17. Note that α should be replaced by $\max\{\alpha, 1\}$. We write just α for readability purposes.

parameter α	lower bound	length of path	running time
distance to cograph	$n \log \frac{n}{\alpha}$	$\alpha 2^{O(\mu)}$	$2^{\alpha 2^{O(\mu)}}(n+m)\log n$
cluster vertex deletion number	$\frac{n^2}{\alpha}$	$O(\alpha\mu)$	$2^{\widetilde{O}(\alpha\mu)}(n+m)\log n$
neighborhood diversity	$\frac{n^2}{\alpha}$	$O(\alpha\mu)$	$2^{\widetilde{O}(\alpha\mu)}(n+m)\log n$
modular-width	$n \log_\alpha n$	$\alpha^{O(\mu)}$	$2^{\alpha^{O(\mu)}}(n+m)\log n$
maximum induced matching	$\frac{n^2}{\alpha}$	$O(\alpha\mu)$	$2^{\widetilde{O}(\alpha\mu)}(n+m)\log n$
shrub-depth	$n^{1+\frac{1}{2^\alpha - 1}}$	$\mu^{2^{O(\alpha)}}$	$2^{\mu^{2^{O(\alpha)}}}(n+m)\log n$

We could prove an analogous version of Lemma 4.23 also with the functions $n \mapsto D \cdot n \log \frac{n}{\alpha}$, $n \mapsto D \cdot n \log_\alpha n$, or $n \mapsto D \cdot n^{1+\frac{1}{2^\alpha - 1}}$ with different running time of the algorithm (see Table 1).

▶ **Theorem 4.24.** *Self-Deleting s-t-path is* FPT *w.r.t. α and μ combined where α is one of the following parameters: cluster vertex deletion number, neighborhood diversity, distance to cograph, modular-width, maximum induced matching, shrub-depth.*

Proof. We prove the theorem for cluster vertex deletion number, the rest is proved similarly by using suitable lower bound from Table 1. If $\mathrm{cvdn}(G) = 0$, the graph is a cluster and we can restrict ourselves to the clique where s and t lies and use Observation 4.21. Otherwise, if $\mathrm{cvdn}(G) \geq 1$, by the result of Dvořák et al. [12] any traceable graph G on $n \geq 4\,\mathrm{cvdn}(G)$ vertices contains at least $\frac{n^2}{16(\mathrm{cvdn}(G)+1)} \geq \frac{n^2}{32\,\mathrm{cvdn}(G)}$ edges. Invoke Lemma 4.23 for $D = \frac{1}{32}$ and $C = 4$ to obtain the desired algorithm. ◀

Domination Number. FPT algorithms for distance to cograph, modular-width, or maximum induced matching cannot be extended to an FPT algorithm for the diameter of the graph (and μ combined). We show that Self-Deleting s-t-path is para-NP-hard already for domination number and μ combined.

▶ **Theorem 4.25** ($\star$). *Self-Deleting s-t-path remains* NP*-hard even when the self-deleting graph (G, f) satisfies $\gamma(G) = \mu_f = 1$.*

5 Kernels

In this section, we show that while Self-Deleting s-t-path admits FPT algorithms for broad number of structural parameters with μ combined, it does not admit a polynomial kernel w.r.t. the vertex cover number and μ combined already in the class of 2-outerplanar graphs unless the polynomial hierarchy collapses. Moreover, there is no polynomial kernel in the class of cliques w.r.t. μ, but there is a linear Turing kernel w.r.t. μ in the class of cliques.

▶ **Theorem 5.1** ($\star$). *Unless* NP $\subseteq$ coNP/$_{\text{poly}}$, *Self-Deleting s-t-path does not admit a polynomial kernel with respect to*

a) vcn *and* μ *combined, even on* 2*-outerplanar graphs;*

b) vi *even with* $\mu = 1$ *and on* 2*-outerplanar graphs;*

c) μ *on cliques.*

Proof Sketch for a). We provide an OR-cross-composition of 3Sat into Self-Deleting s-t-path parameterized by vcn and μ. The idea is to use Construction 3.1 for a formula with all $O(n^3)$ clauses with auxiliary *skip edges* that allow to pass a clause for free. Before s we prepend selector vertices for τ instances that will delete appropriate skip edges and thus modify the rest of the graph to look like the reduction for the given formula. Details can be found in the full version of this paper. ◀

▶ **Theorem 5.2** ($\star$). *Self-Deleting s-t-path admits*

a) *an* $O(\text{fen})$ *kernel;*

b) *an* $O(\text{vcn})$ *kernel on outerplanar graphs;*

c) *a Turing kernel with* $O(\mu)$ *vertices on cliques.*

▶ **Corollary 5.3** ($\star$). *Self-Deleting s-t-path can be solved in* $2^{O(\mu)}n^2$ *time if the underlying graph is a clique and* $2^{o(\mu)}\,\text{poly}(n)$*-time algorithm on cliques violates ETH.*

6 Conclusion and open problems

We initiated a systematic study of complexity of finding a simple path in a self-deleting graph, which we call Self-Deleting s-t-path. While the problem is hard on very restricted graph classes, we were able to design FPT algorithm(s) parameterized by the solution size and structure of the deletion function. This further allowed us to design FPT algorithms for various structural parameters combined with the structure of the deletion function.

Our derandomization of the color-coding algorithm yields running time of $2^{O(k\log(k\mu))}\,\text{poly}(n)$ (Theorem 4.17) or $2^{O(k\mu)}\,\text{poly}(n)$ (Theorem 4.16). We were unable to derandomize it in a way to match the randomized running time of $2^{O(k\log\mu)}\,\text{poly}(n)$ from Theorem 4.15. We conjecture that with a suitable pseudorandom object, there is a way to derandomize the algorithm into a deterministic $2^{O(k\log\mu)}\,\text{poly}(n)$ time.

Our framework for FPT algorithms parameterized by k and μ together with lower bounds on the number of edges in traceable graphs with dense structure does not give optimal running times under ETH. For example, already for cliques, the framework gives running time $2^{O(\mu\log\mu)}(n+m)$ (Observation 4.21) which is not optimal (Corollary 5.3). Assuming ETH, we cannot obtain an algorithm for Shortest Self-Deleting s-t-path with running time $2^{o(k)\log\mu}\,\text{poly}(n)$ (see Remark 3.5). Similarly, an algorithm with running time $2^{k\cdot o(\log\mu)}\,\text{poly}(n)$ would imply that Shortest Self-Deleting s-t-path parameterized by k is in FPT (this is due to [4, Lemma 1]), which would then imply that FPT $=$ W[1]. Note that Shortest Self-Deleting s-t-path becomes FPT w.r.t. k if $\mu \in O(\log n)$ by plugging into the algorithm from Theorem 4.17 and using the fact that $(\log n)^{g(k)}$ is fpt-time.

Can the algorithms for structural parameters and μ be improved to match the above ETH lower bounds? For example, is it possible to solve Self-Deleting s-t-path in deterministic $2^{O(\text{vcn}\log\mu)}\,\text{poly}(n)$ time (note that we can achieve such a running time by a randomized algorithm)?

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the Association for Computing Machinery*, 42(4):844–856, July 1995. doi:10.1145/210332.210337.

2 J. Añez, T. De La Barra, and B. Pérez. Dual graph representation of transport networks. *Transportation Research Part B: Methodological*, 30(3):209–216, 1996. doi:10.1016/0191-2615(95)00024-0.

3 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernel bounds for path and cycle problems. *Theoretical Computer Science*, 511:117–136, 2013. doi:10.1016/J.TCS.2012.09.006.

4 Liming Cai and David W. Juedes. Subexponential parameterized algorithms collapse the W-hierarchy. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 273–284. Springer, 2001. doi:10.1007/3-540-48224-5_23.

5 Sarah Carmesin, David Woller, David Parker, Miroslav Kulich, and Masoumeh Mansouri. The Hamiltonian cycle and travelling salesperson problems with traversal-dependent edge deletion. *Journal of Computer Science*, 74:102156, 2023. doi:10.1016/J.JOCS.2023.102156.

6 Raffaele Cerulli, Francesca Guerriero, Edoardo Scalzo, and Carmine Sorgente. Shortest paths with exclusive-disjunction arc pairs conflicts. *Computers & Operations Research*, 152:106158, 2023. doi:10.1016/J.COR.2023.106158.

7 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Linear FPT reductions and computational lower bounds. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 212–221. ACM, 2004. doi:10.1145/1007352.1007391.

8 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006. doi:10.1016/J.JCSS.2006.04.007.

9 Ting Chen, Ming-Yang Kao, Matthew Tepel, John Rush, and George M. Church. A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology*, 8(3):325–337, 2001. doi:10.1089/10665270152530872.

10 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

11 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

12 Michal Dvořák, Dušan Knop, Michal Opler, Jan Pokorný, Ondřej Suchý, and Krisztina Szilágyi. Density of traceable graphs, 2025. arXiv:2506.22269.

13 Harold N. Gabow, Shachindra N. Maheswari, and Leon J. Osterweil. On two problems in the generation of program test paths. *IEEE Transactions on Software Engineering*, 2(3):227–231, 1976. doi:10.1109/TSE.1976.233819.

14 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/JCSS.2000.1727.

15 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/JCSS.2001.1774.

16 Ronald F. Kirby and Renfrey B. Potts. The minimum route problem for networks with turn penalties and prohibitions. *Transportation Research*, 3(3):397–408, 1969. doi:10.1016/S0041-1647(69)80022-5.

17 Petr Kolman and Ondřej Pangrác. On the complexity of paths avoiding forbidden pairs. *Discrete Applied Mathematics*, 157(13):2871–2876, 2009. doi:10.1016/J.DAM.2009.03.018.

18 Jakub Kováč. Complexity of the path avoiding forbidden pairs problem revisited. *Discrete Applied Mathematics*, 161(10):1506–1512, 2013. doi:10.1016/j.dam.2012.12.022.

19 K. Krause, R. Smith, and M. Goodwin. Optimal software test planning through automated network analysis. In *Proceedings of the IEEE Symposium on Computer Software Reliability*, pages 18–22, 1973.

20 Stefan Szeider. Finding paths in graphs avoiding forbidden transitions. *Discrete Applied Mathematics*, 126(2):261–273, 2003. doi:10.1016/S0166-218X(02)00251-2.

21 Piotr Wojciechowski, K. Subramani, and Alvaro Velasquez. Reachability in choice networks. *Discrete Optimization*, 48:100761, 2023. doi:10.1016/j.disopt.2023.100761.

22 Piotr Wojciechowski, M. Williamson, and K. Subramani. On the analysis of optimization problems in arc-dependent networks. *Discrete Optimization*, 45:100729, 2022. doi:10.1016/j.disopt.2022.100729.

23 Huanhuan Wu, James Cheng, Yiping Ke, Silu Huang, Yuzhen Huang, and Hejun Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016. doi:10.1109/TKDE.2016.2594065.

24 Hananya Yinnone. On paths avoiding forbidden pairs of vertices in a graph. *Discrete Applied Mathematics*, 74(1):85–92, 1997. doi:10.1016/S0166-218X(96)00017-0.

25 Athanasios K. Ziliaskopoulos and Hani S. Mahmassani. A note on least time path computation considering delays and prohibitions for intersection movements. *Transportation Research Part B: Methodological*, 30(5):359–367, 1996. doi:10.1016/0191-2615(96)00001-X.