An FPT Algorithm and a Polynomial Kernel for Linear Rankwidth-1 Vertex Deletion

Mamadou Moustapha Kanté[1], Eun Jung Kim[2], O-joung Kwon[3], and Christophe Paul[4]

1 **LIMOS, CNRS – Clermont Université, Université Blaise Pascal, France**
mamadou.kante@isima.fr
2 **LAMSADE, CNRS – Université Paris Dauphine, France**
eunjungkim78@gmail.com
3 **Institute for Computer Science and Control, Hungarian Academy of Sciences, Hungary***
ojoungkwon@gmail.com
4 **LIRMM, CNRS – Université Montpellier, France†**
christophe.paul@lirmm.fr

Abstract

Linear rankwidth is a linearized variant of rankwidth, introduced by Oum and Seymour [Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.], and it is similar to pathwidth, which is the linearized variant of treewidth. Motivated from the results on graph modification problems into graphs of bounded treewidth or pathwidth, we investigate a graph modification problem into the class of graphs having linear rankwidth at most one, called the Linear Rankwidth-1 Vertex Deletion (shortly, LRW1-Vertex Deletion). In this problem, given an n-vertex graph G and a positive integer k, we want to decide whether there is a set of at most k vertices whose removal turns G into a graph of linear rankwidth at most one and if one exists, find such a vertex set. While the meta-theorem of Courcelle, Makowsky, and Rotics implies that LRW1-Vertex Deletion can be solved in time $f(k) \cdot n^3$ for some function f, it is not clear whether this problem allows a runtime with a modest exponential function. We establish that LRW1-Vertex Deletion can be solved in time $8^k \cdot n^{\mathcal{O}(1)}$. The major obstacle to this end is how to handle a long induced cycle as an obstruction. To fix this issue, we define the *necklace graphs* and investigate their structural properties. We also show that the LRW1-Vertex Deletion has a polynomial kernel.

1998 ACM Subject Classification F.2.2 Computation on Discrete Structures, G.2.1 Combinatorics Algorithms, G.2.2 Graph Agorithms

Keywords and phrases (linear) rankwidth, distance-hereditary graphs, thread graphs, parameterized complexity, kernelization

Digital Object Identifier 10.4230/LIPIcs.IPEC.2015.138

1 Introduction

In a parameterized problem, we are given an instance (x, k), where k is a secondary measurement, called as the *parameter*. The central question in parameterized complexity is whether a parameterized problem admits an algorithm with runtime $f(k) \cdot |x|^{\mathcal{O}(1)}$, equivalently an *FPT*

* Supported by ERC Starting Grant PARAMTIGHT (No. 280152).
† Supported by the "Chercheur d'avenir – Languedoc-Roussillon" project KERNEL

10th International Symposium on Parameterized and Exact Computation (IPEC 2015).
Editors: Thore Husfeldt and Iyad Kanj; pp. 138–150

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

algorithm, where f is a function depending on the parameter k alone, and $|x|$ is the input size. As we study a parameterized problem when its unparameterized decision version is NP-hard, the function f is super-polynomial in general. A parameterized problem admitting such an algorithm is said to be *fixed-parameter tractable*, or *FPT* in short. For many natural parameterized problems, the function f is overwhelming [15] or even non-explicit [23], especially when the algorithm is indicated by a meta-theorem. Therefore, a lot of research effort focus on designing an FPT algorithm with affordable super-exponential part in the runtime. We are especially interested in solving a parameterized problem in *single-exponential* time, that is, in time $c^k \cdot n^{O(1)}$ for some constant c.

A powerful technique to handle parameterized problems is the *kernelization algorithm*. A kernelization algorithm takes an instance (x, k) and outputs an instance (x', k') in time polynomial in $|x| + k$ satisfying that (1) (x, k) is a YES-instance if and only if (x', k') is a YES-instance, (2) $k' \leq k$, and $|x'| \leq g(k)$ for some function g. The reduced instance is called a *kernel* and the function g is called the *size* of the kernel. A parameterized problem is said to admit a *polynomial kernel* if there is a kernelization algorithm that reduces the input instance into an instance with size bounded by a polynomial function $g(k)$ in k.

Many natural graph problems can be expressed as a graph modification problem. Generally, given an input graph G and a fixed set O of elementary operations and a graph property Π, the objective is to transform G into a graph $H \in \Pi$ by applying at most k operations from O. Vertex deletion, edge deletion/addition or contraction are examples of such elementary operations.

The graph property Π having treewidth or pathwidth at most w has received in-depth attention as many problems become tractable on graphs of small treewidth. The celebrated Courcelle's theorem [8] implies that every graph property expressible in monadic second order logic of the second type (MSO_2) can be verified in time $f(w) \cdot n$, when the input n-vertex graph has treewidth at most w. Furthermore, having small treewidth frequently facilitates the design of a dynamic programming algorithm whose runtime is much faster than that of the all-round algorithm from the Courcelle's meta-theorem. Therefore, it is reasonable to measure how close an instance is from "an island of tractability within an ocean of intractable problems" [18].

In the context of treewidth, the deletion problems for the special cases of $w = 0$ and $w = 1$ correspond to the well-known VERTEX COVER and FEEDBACK VERTEX SET problems respectively. More generally, for any fixed w, the corresponding graph modification problem TREEWIDTH-w VERTEX DELETION can be solved in time $f(w, k) \cdot n$ implied by Graph Minor Theory [23] and Courcelle's meta-theorem [8]. As the function f subsumed in the meta theorem is gigantic, it is natural to ask whether the exponential function in the runtime can be rendered realistic. Recent endeavor pursuing this question culminated in establishing that for any fixed w, the TREEWIDTH-w VERTEX DELETION is single-exponential fixed parameter tractable with the deletion number k as the parameter [13, 19].

As for pathwidth, PATHWIDTH-1 VERTEX DELETION was first studied by Philip, Raman, Villanger [22], and later Cygan, Pilipczuk, Pilipczuk, Wojtaszczyk [12] showed that PATHWIDTH-1 VERTEX DELETION can be solved in time $4.65^k \cdot n^{\mathcal{O}(1)}$ and it admits a quadratic kernel. Using the general method developed for TREEWIDTH-w VERTEX DELETION [13, 19], the PATHWIDTH-w VERTEX DELETION problem admits a single exponential FPT algorithm.

Linear rankwidth. *Rankwidth* was introduced by Oum and Seymour [21] for efficiently approximating *clique-width*. *Linear rankwidth* is a linearized variation of rankwidth like

pathwidth is the linearized variant of treewidth. While treewidth and pathwidth are small only on sparse graphs, dense graphs may have small rankwidth or linear rankwidth. For instance, complete graphs, complete bipartite graphs, and threshold graphs [7] have linear rankwidth at most one even though all of them have unbounded treewidth. Rankwidth and linear rankwidth have been intensively studied to generalize the known results for treewidth and pathwidth [2, 9, 17, 20, 21].

Ganian [16] pointed out that some NP-hard problems, such as computing pathwidth, can be solved in polynomial time on graphs of linear rankwidth at most 1. Generally, the meta-theorem by Courcelle, Makowsky, Rotics [10] states that for every graph property Π expressible in monadic second order logic of the first type (MSO_1) and fixed k, there is a cubic-time algorithm for testing whether a graph of rankwidth at most k has property Π. As rankwidth is always less than or equal to linear rankwidth, those problems are tractable on graphs of bounded linear rankwidth as well.

In the same context, it is natural to ask whether there is an FPT algorithm for (Linear) Rankwidth-w Vertex Deletion, that is, a problem asking whether for a given graph G and a positive integer k, G contains a vertex subset of size at most k whose deletion makes G a graph of (linear) rankwidth at most w. It is only known that for fixed w, both problems are FPT from the meta-theorem on graphs of bounded rankwidth [10] and the fact that one vertex deletion can decrease rankwidth or linear rankwidth by at most one. We discuss it in more detail in the last section. However, as the function of k obtained from the meta-theorem is enormous, it is interesting to know whether there is a single-exponential FPT algorithm for both problems, like Treewidth-w Vertex Deletion. Also, to the best of our knowledge, there was no known previous result whether (Linear) Rankwidth-w Vertex Deletion admits a polynomial kernel for any integer w.

Our contributions. In this paper, we show that the Linear Rankwidth-1 Vertex Deletion problem admits a single-exponential FPT algorithm and a polynomial kernel. This is a first step towards a goal of investigating whether (Linear) Rankwidth-w Vertex Deletion is single-exponential FPT or has a polynomial kernel.

Linear Rankwidth-1 Vertex Deletion (LRW1-Vertex Deletion)
Input : A graph G, a positive integer k
Parameter : k
Question : Does G have a vertex subset S of size at most k whose removal makes G a graph of linear rankwidth at most one?

▶ **Theorem 1.1.** *For fixed k and a given graph G with n vertices, the* LRW1-Vertex Deletion *problem can be solved in $8^k \cdot n^{\mathcal{O}(1)}$ time.*

▶ **Theorem 1.2.** *The* LRW1-Vertex Deletion *problem has a polynomial kernel.*

We note that several graph classes with a certain path-like structure have been studied for parameterized vertex deleting problems. Such classes include graphs of pathwidth 1, proper interval graphs [14, 25], unit interval graphs [24, 5], and interval graphs [6]. A common approach in the previous work is to use the characterization of the structures obtained after removing small obstructions. We also characterize graphs excluding small obstructions for graphs of linear rankwidth at most 1 to develop an FPT algorithm and a polynomial kernel.

The main ingredient is to investigate a new class of graphs, called *necklace graphs*, which are close to graphs of linear rankwidth at most 1. To define this class, we use the induced subgraph obstructions for graphs of linear rankwidth at most 1, which are listed in Figures 1

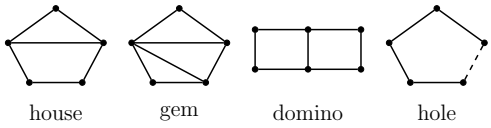

Figure 1 The induced subgraph obstructions for distance-hereditary graphs.

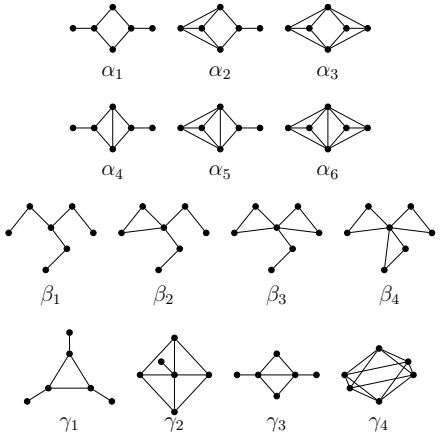

Figure 2 The distance-hereditary induced subgraph obstructions for thread graphs.

and 2 [1]. Briefly speaking, necklace graphs, when viewed locally, are graphs of linear rankwidth at most 1, but they may have a long induced cycle. We show that every connected graph having no obstructions of size at most 8 is a necklace graph, and we can easily find a minimum deleting set on a connected necklace graph. In our FPT algorithm, we first use a simple branching algorithm to remove the obstructions of size at most 8 with the time complexity $8^k \cdot n^{\mathcal{O}(1)}$. Since a final instance does not have obstructions of size at most 8, it is a disjoint union of thread graph and necklace graphs, and we compare the remaining budget with the sum of minimum deleting set over all necklace components to decide whether it is a Yes-instance.

To obtain a polynomial kernel, we start with packing obstructions of size at most 8 using the Sunflower lemma, and taking a minimum deleting set on the remaining necklace graph. The union of two sets will have size bounded by a polynomial in k, and its removal makes an input graph into a graph of linear rankwidth at most 1. Graphs of linear rankwidth at most 1 can be seen as graphs obtained by connecting certain blocks, called *thread blocks*, like a path (Theorem 2.1). The main difficulty for reducing the remaining part is to shrink a large thread block. Even though there is a simple pattern of constructions on thread blocks, it is not at all obvious how to find an irrelevant vertex in a sufficiently large thread block regarding all individual small obstructions. We can resolve this issue using the set obtained by the Sunflower lemma, which has the nice property that any minimum deleting set for the small obstructions in G is contained in the prescribed set. Using this, we will show how to find another obstruction from the long induced cycle containing a potential irrelevant vertex. We remark that a similar idea was used by Fomin, Saurabh, and Villanger [14] to obtain a polynomial kernel for the Proper Interval Vertex Deletion problem.

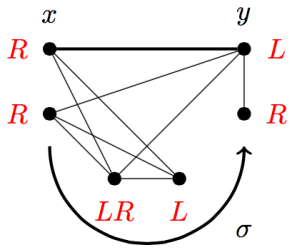

Figure 3 An example of a thread block.

2 Preliminaries

In this paper, all graphs are finite and undirected, if not mentioned otherwise. For a graph G, we denote by $V(G)$ and $E(G)$ the vertex set and the edge set of G, respectively, if they are not specified. Let $G = (V, E)$ be a graph. Let $N_G(x)$ denote the neighborhood of a vertex $x \in V$. For $S \subseteq V$, $G[S]$ denotes the subgraph of G induced on S and we denote by $G \setminus S := G[V \setminus S]$. For $W \subseteq E$, we denote by $G \setminus W := (V, E \setminus W)$. For short we write $G \setminus x$ instead of $G \setminus \{x\}$ for $x \in V \cup E$. A vertex v of G is called a *pendant vertex* if $|N_G(x)| = 1$. An edge e of a connected graph G is called a *cut-edge* if $G \setminus e$ is disconnected. The length of a path is defined as the number of edges in the path. For $n \geq 3$, we denote by C_n the cycle with n vertices. For a set $\mathcal{F}$ of graphs, a graph G is *$\mathcal{F}$-free* if G has no induced subgraph isomorphic to a graph in $\mathcal{F}$.

A (linear) ordering on a finite set S is a bijective mapping $\sigma : S \to \{1, \ldots, |S|\}$, and we write $x <_\sigma y$ if $\sigma(x) < \sigma(y)$, and σ^{-1} as the inverse bijective mapping. For an $X \times Y$-matrix M and $X' \subseteq X, Y' \subseteq Y$, let $M[X', Y']$ be the submatrix of M whose rows and columns are indexed by X' and Y', respectively.

Linear rankwidth and thread graphs. The *adjacency matrix* of a graph $G = (V, E)$, which is a $(0, 1)$-matrix over the binary field, will be denoted by A_G. The *width* of a linear ordering σ of V in G is $\max_{1 \leq i \leq |V|} \operatorname{rank}(A_G[\{\sigma^{-1}(1), \ldots, \sigma^{-1}(i)\}, V \setminus \{\sigma^{-1}(1), \ldots, \sigma^{-1}(i)\}])$, where the rank is computed over the binary field. The *linear rankwidth* of a graph G is defined as the minimum width over all linear orderings of V.

Graphs of linear rankwidth at most one are called *thread graphs* by Ganian [16], and each connected thread graph consists of a sequence of thread blocks. We follow the definition of thread blocks given by Adler, Farley, and Proskurowski [1], and develop a more convenient way to create a thread graph, which is useful to define *necklace graphs*.

A triple $B(x, y) = (G, \sigma, \ell)$, where x and y are two vertices of the graph $G = (V, E)$, σ is a ordering on V, and ℓ is a function from V to $\{\{L\}, \{R\}, \{L, R\}\}$, is a *thread block* if:

1. $\ell(x) = \{R\}$ and $\ell(y) = \{L\}$,
2. for $v, w \in V$ with $v <_\sigma w$, $vw \in E(G)$ if and only if $R \in \ell(v)$ and $L \in \ell(w)$,
3. $\ell(\sigma^{-1}(2)) \neq \{L\}$ if $\sigma^{-1}(2) \neq y$.

See Figure 3 for an example. The aim of the third condition is to guarantee a unique decomposition of thread graphs into thread blocks. For a digraph $D = (V_D, A_D)$, a set of thread blocks $\{B(x, y) = (G_{xy}, \sigma_{xy}, \ell_{xy}) \mid xy \in A_D\}$ is said to be *mergeable with* D if for any two arcs x_1y_1, x_2y_2 of A_D, $V(G_{x_1y_1}) \cap V(G_{x_2y_2}) = \{x_1, y_1\} \cap \{x_2, y_2\}$. For a digraph $D = (V_D, A_D)$ and a mergeable set of thread blocks $\mathcal{B}_D = \{B(x, y) = (G_{xy}, \sigma_{xy}, \ell_{xy}) \mid xy \in$

$A_D\}$, $G = D \odot \mathcal{B}_D$ is the graph with the vertex set $V = \bigcup_{xy \in A_D} V(G_{xy})$ and the edge set $E = \bigcup_{xy \in A_D} E(G_{xy})$.

A connected graph G is a *thread graph* if G is an one vertex graph or $G = P \odot \mathcal{B}_P$ for some directed path P, called the *underlying path*, and some set of thread blocks $\mathcal{B}_P$ mergeable with P. A graph is a *thread graph* if each of its connected components is a thread graph.

The induced subgraph obstructions for graphs of linear rankwidth at most 1 consist of the set of induced subgraph obstructions for graphs of rankwidth at most 1 (equivalently, distance-hereditary graphs) [3], which are a house, a gem, a domino, and induced cycles of length at least 5 in Figure 1, and the set of 14 induced subgraph obstructions for linear rankwidth at most 1 that are graphs of rankwidth 1, depicted in Figure 2. For convenience, we define that

- Ω_U is the set of 14 graphs in Figure 2,
- $\Omega_T := \{\text{house, gem, domino}\} \cup \{C_k \mid k \geq 5\} \cup \Omega_U$, and
- $\Omega_N := \{\text{house, gem, domino}, C_5, C_6, C_7, C_8\} \cup \Omega_U$.

▶ **Theorem 2.1** ([16, 1]). *Let G be a graph. The following are equivalent.*

- *G has linear rankwidth at most 1.*
- *G is a thread graph.*
- *G has no induced subgraph isomorphic to a graph in Ω_T.*

It is known that one can recognize a graph of linear rankwidth 1 in polynomial time using split decompositions of graphs [4, 2], and easily decompose a connected thread graph into thread blocks.

▶ **Theorem 2.2** ([4, 2]). *Let G be a graph on with n vertices and m edges. Then in time $\mathcal{O}(n+m)$, we can test whether G is a thread graph, and if G is a thread graph, then we can decompose each connected component into a sequence of thread blocks in the same time.*

In the remaining part, we frequently use the term 'thread graphs' for graphs of linear rankwidth at most 1. For a graph G and $S \subseteq V$, S is called a *LRW1-deletion set* if $G \setminus S$ is a thread graph.

Necklace graphs. We generalize the construction of thread graphs from directed paths to directed cycles. A connected graph G is called a *necklace graph* if $G = C \odot \mathcal{B}_C$ for some directed cycle C, called the *underlying cycle*, and some set of thread blocks $\mathcal{B}_C$ mergeable with C. Our FPT algorithm and the construction of a polynomial kernel relies on the following characterization of Ω_N-free graphs.

▶ **Theorem 2.3.** *A connected Ω_N-free graph is either a connected thread graph or a necklace graph whose underlying cycle has length at least 9.*

Let us sketch the proof of Theorem 2.3, which constructs the underlying cycle and a set of thread blocks as follows. Let $G = (V, E)$ be a connected Ω_N-free graph and suppose that G is not a thread graph. Since G is Ω_N-free and it is not a thread graph, by Theorem 2.1, G has an induced subgraph isomorphic to C_k for some $k \geq 9$. We prove by induction on $|V|$ that if C is a shortest cycle among induced cycles of length at least 9 in G, then G is a necklace graph whose underlying cycle is C. Let $C := v_1 v_2 \cdots v_k v_1$ be a shortest cycle among induced cycles of length at least 9 in G and for convenience, let $v_{k+1} := v_1$ and $v_{k+2} := v_2$. We regard C as a directed cycle where for each $1 \leq j \leq k$, $v_j v_{j+1}$ is an arc.

If $G = C$, then we are done because C itself is a necklace graph with the underlying cycle C. We may assume that $|V| > |V(C)|$, and choose a vertex $v \in V \setminus V(C)$ such that $G \setminus v$ is

connected. Clearly, $G \setminus v$ is again Ω_N-free graph, and C is a shortest cycle among induced cycles of length at least 9 in $G \setminus v$. By the induction hypothesis, there exists some set of thread blocks $\mathcal{B}_C$ mergeable with C such that $G \setminus v = C \odot \mathcal{B}_C$. The rest of the proof consists in showing that $G = C \odot \mathcal{B}'_C$ for some set of thread blocks $\mathcal{B}'_C$ mergeable with C.

3 An FPT algorithm for LRW1-Vertex Deletion

Our FPT algorithm is a branching algorithm that reduces an input instance to a Ω_N-free graph. As each graph of Ω_N has size at most 8, the announced complexity follows. It remains to prove that given a Ω_N-free graph, a minimum vertex deletion set for LRW1-Vertex Deletion can be found in polynomial time. In fact, we prove that such a set has size at most one per necklace component and identifying such a vertex requires polynomial time.

Using the following proposition, we can find a minimum LRW1-deletion set of a Ω_N-free graph in polynomial time.

▶ **Proposition 3.1.** *Let G be a Ω_N-free graph with n vertices and m edges. We can compute the minimum size of a LRW1-deletion set of G in time $\mathcal{O}(n+m)$, and find such a set S in the same time.*

To prove it, we use the following lemma.

▶ **Lemma 3.2.** *Let G be a connected necklace graph with the underlying cycle C of length at least 4. For each $v \in V(C)$, $G \setminus v$ is a thread graph.*

Proof of Proposition 3.1. Let k be the minimum size of a LRW1-deletion set of G. We remark that each connected component of G is either a thread graph or a necklace graph by Theorem 2.3. For each component H of G, we test whether H is a thread graph or not in time $\mathcal{O}(|V(H)|+|E(H)|)$ using Theorem 2.2. Note that we should remove at least one vertex for each necklace component of G, and moreover, by Lemma 3.2, it is enough to remove exactly one vertex for each component. Thus, k is the number of its necklace components.

To identify such a vertex in each necklace component, it is sufficient to find a vertex on the underlying cycle by Lemma 3.2. Let H be a necklace component and C be the underlying cycle of H. Observe that for $v \in V(H) \setminus V(C)$, $H \setminus v$ is still a connected necklace graph with the same underlying cycle, because we can adjust the ordering of the thread block containing v into an ordering without v with the restricted labeling. We first test whether H has a cut vertex, and if it has a cut vertex w, then $w \in V(C)$. Otherwise, we search a vertex cut of size 2. Since every necklace graph whose underlying cycle has length at least 4 contains a vertex cut of size 2, we can find it, say $\{v, w\}$. Then one of the two vertices should be contained in C. We test whether $H \setminus v$ or $H \setminus w$ is a thread graph or not. If $H \setminus v$ is a thread graph, then $v \in V(C)$, and otherwise, $w \in V(C)$. Since finding a cut vertex or a vertex cut of size two can be done in linear time, we are done with the time complexity. ◀

Proof of Theorem 1.1. Let (G, k) be an instance of the LRW1-Vertex Deletion problem. First find an induced subgraph of G isomorphic to a graph in Ω_N and branch by removing one of the vertices in the subgraph. Because the maximum size of graphs in Ω_N is 8, we can find such a vertex subset in time $\mathcal{O}(n^8)$ if exists. After the branching algorithm, we transform the given instance (G, k) into at most 8^k sub-instances (G', k') such that each sub-instance consists of a Ω_N-free graph G' and a remaining budget k'. Clearly, (G, k) is a Yes-instance if and only if one of sub-instances (G', k') is a Yes-instance.

Let (G', k') be a sub-instance obtained from the branching algorithm. Since G' is Ω_N-free, by Theorem 2.3, each connected component of G' is either a thread graph or a necklace

graph with the underlying cycle of length at least 9. By Proposition 3.1, we can compute a minimum LRW1-deletion set of G' in time $\mathcal{O}(|V(G'|+|E(G')|)$, and decide whether (G', k') is a YES-instance. By checking all sub-instances, we can decide whether (G, k) is a YES-instance in time $8^k \cdot \mathcal{O}(n+m)$ where m is the number of edges of G. We conclude that the LRW1-VERTEX DELETION problem can be solved in time $8^k \cdot \mathcal{O}(n^8)$. ◀

4 A polynomial kernel for LRW1-Vertex Deletion

We use the Sunflower lemma for packing obstructions of small size. It consists in finding a subset T of the input graph $G = (V, E)$ whose removal turns G into a thread graph with the property that for every set $S \subseteq V$ of size at most k, the following are equivalent (Lemma 4.2):

- S is a minimal vertex set such that $G \setminus S$ has no obstructions in Ω_N.
- S is a minimal vertex set such that $G[T] \setminus S$ has no obstructions in Ω_N.

From this property, if we choose a minimal LRW1-deletion set S in the input graph, then the vertices of $S \setminus T$ should be used to remove at least one long induced cycle. This property is essentially used to find an irrelevant vertex in a large thread block. Moreover, with this set, we can preprocess the instance so that there is no obstruction containing exactly one vertex of T. This would be used to bound the length of the sequence of thread blocks in each connected component, and the number of non-trivial components.

Let $(G = (V, E), k)$ be an instance of LRW1-VERTEX DELETION. We start with an easy reduction rule.

▶ **Reduction Rule 1.** *If G has a component that is a thread graph, then we remove it from G.*

4.1 Packing small obstructions

Let $\mathcal{F}$ be a family of subsets over a set U. A subset $U' \subseteq U$ is called *a hitting set* of $\mathcal{F}$ if for every set $F \in \mathcal{F}$, $F \cap U' \neq \emptyset$. For a graph G and a family of graphs $\mathcal{F}$, a set $S \subseteq V(G)$ is also called *a hitting set* for $\mathcal{F}$ if for every induced subgraph H of G that is isomorphic to a graph in $\mathcal{F}$, $V(H) \cap S \neq \emptyset$. The following lemma can be obtained from the Sunflower lemma.

▶ **Lemma 4.1** ([14]). *Let $\mathcal{F}$ be a family of sets of size at most d over a set U, and let k be a positive integer. Then in time $\mathcal{O}(|\mathcal{F}|(k+|\mathcal{F}|))$, we can find a nonempty set $\mathcal{F}' \subseteq \mathcal{F}$ such that*

1. *for every $U' \subseteq U$ of size at most k, U' is a minimal hitting set of $\mathcal{F}$ if and only if U' is a minimal hitting set of $\mathcal{F}'$, and*
2. *$|\mathcal{F}'| \leq d!(k+1)^d$.*

Using Proposition 3.1 and Lemma 4.1, we identify a subset T of vertices of G of polynomial size in k that allows us to forget about small obstructions in G.

▶ **Lemma 4.2.** *Let $(G = (V, E), k)$ be an instance of* LRW1-VERTEX DELETION. *There is a polynomial time algorithm that either concludes that (G, k) is a* NO*-instance or finds a non-empty set $T \subseteq V$ such that*

1. *$G \setminus T$ is a thread graph,*
2. *for every set $S \subseteq V$ of size at most k, S is a minimal hitting set for Ω_N in G if and only if it is a minimal hitting set for Ω_N contained in $G[T]$, and*
3. *$|T| \leq 8 \cdot 8!(k+1)^8 + k$.*

Let us fix a subset T of V obtained by Lemma 4.2. We preprocess using the following reduction rule.

▶ **Reduction Rule 2.** *Let $U \subseteq T$ such that for every $u \in U$, there exists an induced subgraph H of G isomorphic to a graph in Ω_N with $V(H) \cap T = \{u\}$. If $|U| > k$, then (G, k) is a* No*-instance; otherwise, remove U from G and reduce k by $|U|$, and use $T \setminus U$ instead of T.*

It can be done in polynomial time because we only need to look at obstructions of Ω_N in G.

▶ **Lemma 4.3.** *Reduction Rule 2 is safe.*

From now on, we assume that G is reduced under Reduction Rules 1 and 2. A vertex v of G is called *irrelevant* if (G, k) is a Yes-instance if and only if $(G \setminus v, k)$ is a Yes-instance.

4.2 Bounding the size of components of $G \setminus T$

The goal is to shrink $G \setminus T$ while preserving the solutions. For convenience, let $\mu(k) := 8 \cdot 8!(k+1)^8 + k$. We first show that if a thread block in $G \setminus T$ is large, then we can always find an irrelevant vertex in there.

▶ **Proposition 4.4.** *If $G \setminus T$ contains a thread block $(G_{xy}, \sigma_{xy}, \ell_{xy})$ of size at least $(k+2)(\mu(k)+2)^2+1$, then we can find an irrelevant vertex in G_{xy} in polynomial time.*

We mainly use the following lemma.

▶ **Lemma 4.5.** *Let G be a graph and let $v_1v_2v_3v_4v_5$ be an induced path of length 4 in G. If two distinct vertices w_1, w_2 in $V(G) \setminus \{v_1, v_2, \ldots, v_5\}$ have the neighbors v_2 and v_4 in G, then $G \setminus v_3$ contains an induced subgraph isomorphic to a graph in Ω_N.*

Proof of Proposition 4.4. Suppose that $G \setminus T$ contains a thread block of size at least $(k+2)(\mu(k)+2)^2+1$. We can find such a thread block in polynomial time using Theorem 2.2. Let $B := B(x, y) = (B, \sigma, \ell)$ be a thread block of size at least $(k+2)(\mu(k)+2)^2+1$. For convenience, let σ' be the ordering obtained from σ by removing the end vertices x and y.

In the following procedure, we mark some vertices of B in order to find an irrelevant vertex in B. We set $Z := \emptyset$. (1) For each v of T, choose the first $k+2$ vertices z of σ' that are neighbors of v with $R \in \ell(z)$, and choose the last $k+2$ vertices z of σ' that are neighbors of v with $L \in \ell(z)$, and add them to Z. (2) For each pair of two vertices v, v' in T, choose $k+2$ common neighbors of v and v' in B, and add them to Z. (3) Choose the first $k+2$ vertices z of σ' with $R \in \ell(z)$, and choose the last $k+2$ vertices z of σ' with $L \in \ell(z)$, and add them to Z. In each case, If there are at most $k+1$ such vertices, then we add all of them to Z. Then

$$|Z| \leq |T|(2k+4) + |T|^2(k+2) + (2k+4) \leq (k+2)(\mu(k)+2)^2 - 2.$$

Since $|V(B)| \geq (k+2)(\mu(k)+2)^2+1$, there exists a vertex w in $V(B) \setminus Z \setminus \{x, y\}$. We claim that w is an irrelevant vertex. If (G, k) is a Yes-instance, then $(G \setminus w, k)$ is clealry a Yes-instance.

Suppose that $(G \setminus w, k)$ is a Yes-instance and let $X \subseteq V(G) \setminus \{w\}$ such that $|X| \leq k$ and $G \setminus (X \cup \{w\})$ is a thread graph. We may assume that $G \setminus X$ is not a thread graph. So, $G \setminus X$ must have an obstruction in Ω_T that contains the vertex w. Let $X' \subseteq X \cup \{w\}$ be a minimal hitting set for Ω_N in G. From the property of the set T, X' is a minimal hitting set for Ω_N in $G[T]$, which implies that $X' \subseteq T$. Thus $G \setminus X$ must have an induced cycle of length at least 9 that contains w. Let C be an induced cycle of length at least 9 containing w in $G \setminus X$.

We will find an induced subgraph of $G \setminus (X \cup \{w\})$ that is isomorphic to a graph in Ω_N using Lemma 4.5, which leads a contradiction. Let v_1, v_2, w, v_3, v_4 be the consecutive vertices on C. To apply Lemma 4.5, we will find two vertices that are adjacent to v_2 and v_3.

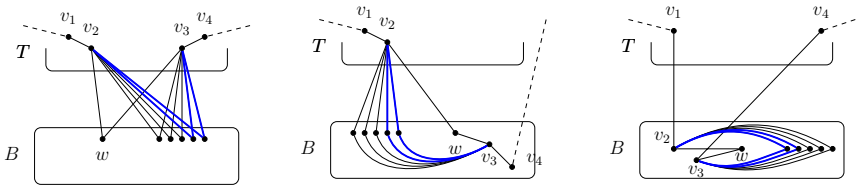

Figure 4 Cases 1–3 in Proposition 4.4.

1. (Case 1. $v_2, v_3 \in T$.) Since v_2 and v_3 have a common neighbor w in $V(B) \setminus Z$, Z contains $k+2$ common neighbors of v_2 and v_3. Since $|X| \leq k$, there exist two vertices $w_1, w_2 \in Z \setminus X$ that are common neighbors of v_2 and v_3.
2. (Case 2. One of v_2 and v_3 is contained in T.) From the symmetry, we may assume that $v_2 \in T$ and $v_3 \notin T$. Since $w \notin \{x, y\}$, v_3 is contained in B. If $R \in \ell(w)$ and $w <_\sigma v_3$, then Z contains the first $k+2$ vertices z of σ' that are neighbors of v_2 with $R \in \ell(z)$. We choose two vertices of them that are not in X. In case when $L \in \ell(w)$ and $v_3 <_\sigma w$, we use the last $k+2$ vertices z of σ' that are neighbors of v_2 with $L \in \ell(z)$ to identify two vertices similarly.
3. (Case 3. Neither v_2 nor v_3 is contained in T.) Since $w \notin \{x, y\}$, v_2 and v_3 are contained in B. If $v_2 <_\sigma w <_\sigma v_3$, then $R \in \ell(v_2)$, $L \in \ell(v_3)$ and it implies that $v_2v_3 \in E$, which is contradiction. Also, $v_3 <_\sigma w <_\sigma v_2$ cannot happen. Thus, both of v_2 and v_3 appear either before w in σ or after w in σ. By the symmetry, we may assume that v_2 and v_3 appear before w in σ. So, $R \in \ell(v_2)$, $R \in \ell(v_3)$, and $L \in \ell(w)$. Since Z contains the last $k+2$ vertices z of σ' with $L \in \ell(z)$, there exist two vertices w_1, w_2 from those $k+2$ vertices that are not in X and C.

In all cases, $G \setminus (X \cup \{w\})$ has an induced subgraph isomorphic to a graph in Ω_N by Lemma 4.5. It contradicts to the assumption that $X \cup \{w\}$ is a LRW1-deletion set of G. Therefore, $G \setminus X$ is a thread graph, and we conclude that (G, k) is a YES-instance. ◀

We show that if a vertex v in T has neighbors on 7 distinct blocks, then we can find a subgraph H isomorphic to one of $\{\beta_1, \beta_2, \beta_3, \beta_4\}$ such that $V(H) \cap T = \{v\}$. However, there is no obstruction in Ω_N containing exactly one vertex from T by Reduction Rule 2. Thus, if a component of $G \setminus T$ has many thread blocks, then we can identify a sequence of consecutive thread blocks not touched by any obstruction in Ω_N. This allows us to contract one of these "safe" thread blocks, say $B(x, y)$, to a vertex v such that $N_{G \setminus T}(v) = (N_{G \setminus T}(x) \cup N_{G \setminus T}(y)) \setminus B(x, y)$.

▶ **Lemma 4.6.** *If $G \setminus T$ has a connected component with at least $19(6\mu(k)+1)$ thread blocks, then we can in polynomial time transform G into a graph $G' = (V', E')$ with $|V'| < |V|$ such that (G, k) is a* YES*-instance if and only if (G', k) is a* YES*-instance.*

4.3 Kernel size

We bound the number of connected components using the following lemma.

▶ **Lemma 4.7.**

1. *The graph $G \setminus T$ has at most $2\mu(k)$ connected components containing at least two vertices.*
2. *If $G \setminus T$ has at least $\mu(k)^2 \cdot (k+2) + 1$ isolated vertices, then we can find an irrelevant vertex in polynomial time.*

Let us now piece everything together and analyze the kernel size.

▶ **Theorem 4.8.** *The* LRW1-Vertex Deletion *problem has a kernel of size* $\mathcal{O}(k^{33})$.

Proof. Let $(G = (V, E), k)$ be an instance of LRW1-Vertex Deletion. By Reduction Rule 1, we may safely assume that G has no components that are thread graphs. Let $T \subset V$ be a vertex subset satisfying Lemma 4.2, and we preprocess using Reduction Rule 2.

By Lemma 4.3, we may assume that for every vertex subset $S \subseteq V$ such that $G[S]$ is a graph of $\{\beta_1, \beta_2, \beta_3, \beta_4\}$, $|S \cap T| \geq 2$. Combining Proposition 4.4 and Lemma 4.6, we can assume that every connected component of $G \setminus T$ has size at most $(k+2)(\mu(k)+2)^2 \cdot 19(6\mu(k)+1)$ (otherwise the instance can be reduced in polynomial time). Note that for each connected component H of $G \setminus T$, there exists a vertex in H that has a neighbor in T, otherwise, H is a component of G that is a thread graph. Therefore, by Lemma 4.7, we can assume that the number of non-trivial components of $G \setminus T$ is at most $2\mu(k)$ and the number of isolated vertices in $G \setminus T$ is at most $\mu(k)^2(k+2)$. It follows that

$$\begin{aligned} |T| + |V \setminus T| &\leq \mu(k) + \big(2\mu(k) \cdot 19(6\mu(k)+1) \cdot (k+2)(\mu(k)+2)^2 + \mu(k)^2 \cdot (k+2)\big) \\ &= \mathcal{O}(k \cdot \mu(k)^4) = \mathcal{O}(k^{33}). \end{aligned}$$

◀

5 Concluding remarks

We consider the problem Linear rankwidth-w Vertex Deletion when $w = 1$. A next step is to investigate the problem for bigger w, or for any fixed w. A closely related problem is Rankwidth-w Vertex Deletion, which asks whether G has a vertex subset of size at most k such that $G \setminus S$ has rankwidth at most w. This problem is fixed-parameter tractable for the following reason. Note that any Yes-instance has rankwidth at most $w + k$. Having bounded rankwith can be characterized by a finite list of forbidden vertex-minors [20]. From [11], having a vertex-minor can be expressed in $\mathsf{C}_2\mathsf{MSO}$, i.e., monadic second order logic without edge set quantification where we can express the parity of $|X|$ for a vertex set X. Fixed-parameter tractability follows as a consequence of Courcelle, Makowsky, Rotics [10].

This result can be turned into a constructive algorithm as [20] provides an explicit upper bound on the size of vertex-minor obstructions for rankwidth k. However, the exponential blow-up in the runtime is huge with respect to both w and k. It is a challenging question whether a reasonable dependency on k can be achieved. A single-exponential time would be ideal, which was achievable for its treewidth counterpart. A first realistic goal is to consider the case when $w = 1$, i.e. the Distance-Hereditary Vertex Deletion. We leave it as an open question whether this problem can be solved in time $c^k \cdot n^{\mathcal{O}(1)}$ time for some constant c.

References

1 Isolde Adler, Arthur M. Farley, and Andrzej Proskurowski. Obstructions for linear rank-width at most 1. *Discrete Applied Mathematics*, 168:3–13, 2014.

2 Isolde Adler, Mamadou Moustapha Kanté, and O-joung Kwon. Linear rank-width of distance-hereditary graphs. In *International Workshop Graph-Theoretic Concepts in Computer Science – WG*, volume 8747 of *Lecture Notes in Computer Science*, pages 42–55, 2014.

3 Hans-Jürgen Bandelt and Henry M. Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.

4 Binh-Minh Bui-Xuan, Mamadou Moustapha Kanté, and Vincent Limouzy. A note on graphs of linear rank-width 1. *CoRR*, abs/1306.1345, 2013.

5 Yixin Cao. Unit interval editing is fixed-parameter tractable. In *International Colloquium on Automata, Languages, and Programming – ICALP*, volume 9134 of *Lecture Notes in Computer Science*, pages 306–317, 2015.

6 Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(3):21–35, 2015.

7 Václav Chvátal and Peter L. Hammer. *Studies in integer programming*, chapter Aggregation of inequalities in integer programming, pages 145–162. Number 1 in Annals of Discrete Mathematics. North-Holland, 1977.

8 Bruno Courcelle. The Monadic Second-Order Theory of Graphs. I. Recognizable Sets of Finite graphs. *Information and Computation*, 85:12–75, 1990.

9 Bruno Courcelle and Mamadou Moustapha Kanté. Graph operations characterizing rank-width. *Discrete Applied Mathematics*, 157(4):627–640, 2009.

10 Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

11 Bruno Courcelle and Sang-il Oum. Vertex-minors, monadic second-order logic, and a conjecture by Seese. *Journal of Combinatorial Theory, Series B*, 97(1):91–126, 2007.

12 Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. An improved fpt algorithm and a quadratic kernel for pathwidth one vertex deletion. *Algorithmica*, 64(1):170–188, 2012.

13 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar f-deletion: Approximation, kernelization and optimal FPT algorithms. In *Annual IEEE Symposium on Foundations of Computer Science – FOCS*, pages 470–479, 2012.

14 Fedor V. Fomin, Saket Saurabh, and Yngve Villanger. A polynomial kernel for proper interval vertex deletion. *SIAM Journal on Discrete Mathematics*, 27(4):1964–1976, 2013.

15 Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, 130(1-3):3–31, 2004.

16 Robert Ganian. Thread graphs, linear rank-width and their algorithmic applications. In *International Workshop on Combinatorial Algorithms – IWOCA*, volume 6460 of *Lecture Notes in Computer Science*, pages 38–42, Heidelberg, 2011.

17 Robert Ganian and Petr Hlinený. On parse trees and myhill-nerode-type tools for handling graphs of bounded rank-width. *Discrete Applied Mathematics*, 158(7):851–867, 2010.

18 Serge Gaspers and Stefan Szeider. Backdoors to satisfaction. In *The Multivariate Algorithmic Revolution and Beyond – Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 287–317, 2012.

19 Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. In *International Colloquium on Automata, Languages, and Programming – ICALP*, volume 7965 of *Lecture Notes in Computer Science*, pages 613–624, 2013.

20 Sang-il Oum. Rank-width and vertex-minors. *Journal of Combinatorial Theory, Series B*, 95(1):79–100, 2005.

21 Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.

22 Geevarghese Philip, Venkatesh Raman, and Yngve Villanger. A quartic kernel for pathwidth-one vertex deletion. In *International Workshop on Graph Theoretic Concepts in Computer Science – WG*, volume 6410 of *Lecture Notes in Computer Science*, pages 196–207, 2010.

23 Neil Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Combin. Theory Ser. B*, 92(2):325–357, 2004.

24 René van Bevern, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Measuring indifference: unit interval vertex deletion. In *International Workshop on Graph Theoretic Concepts in Computer Science – WG*, volume 6410 of *Lecture Notes in Computer Science*, pages 232–243, 2010.

25 Pim van't Hof and Yngve Villanger. Proper interval vertex deletion. *Algorithmica*, 65(4):845–867, 2013.