A Single-Exponential Fixed-Parameter Algorithm for Distance-Hereditary Vertex Deletion*

Eduard Eiben[1], Robert Ganian[2], and O-joung Kwon[3]

1 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
2 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
3 **Institute for Computer Science and Control, Hungarian Academy of Sciences, Budapest, Hungary.**

Abstract

Vertex deletion problems ask whether it is possible to delete at most k vertices from a graph so that the resulting graph belongs to a specified graph class. Over the past years, the parameterized complexity of vertex deletion to a plethora of graph classes has been systematically researched. Here we present the first single-exponential fixed-parameter algorithm for vertex deletion to distance-hereditary graphs, a well-studied graph class which is particularly important in the context of vertex deletion due to its connection to the graph parameter rank-width. We complement our result with matching asymptotic lower bounds based on the exponential time hypothesis.

1998 ACM Subject Classification G.2.1 Combinatorial Algorithms – G.2.2 Graph Algorithms

Keywords and phrases distance-hereditary graphs, fixed-parameter algorithms, rank-width

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.34

1 Introduction

Vertex deletion problems include some of the best studied NP-hard problems in theoretical computer science, including Vertex Cover or Feedback Vertex Set. In general, the problem asks whether it is possible to delete at most k vertices from a graph so that the resulting graph belongs to a specified graph class. While these problems are studied in a variety of contexts, they are of special importance for the parameterized complexity paradigm [11, 9], which measures the performance of algorithms not only with respect to the input size but also with respect to an additional numerical parameter. Vertex deletion problems allow a highly natural choice of the parameter (specifically, k), and many vertex deletion problems are known to admit so-called *single-exponential fixed-parameter algorithms*, which are algorithms running in time $\mathcal{O}(c^k \cdot n^{\mathcal{O}(1)})$ for input size n and some constant c.

Over the past years, the parameterized complexity of vertex deletion to a plethora of graph classes has been systematically researched. However, there still remain a few important classes where the existence of a single-exponential fixed-parameter algorithm remains open. One such class has, until now, been the class of *distance-hereditary graphs* [17] (also called *completely separable graphs* [15]). Distance-hereditary graphs have several equivalent

* The authors acknowledge support by ERC Starting Grant PARAMTIGHT (No. 280152) and the Austrian Science Fund (FWF, projects P26696 and W1255-N23). Robert Ganian is also affiliated with FI MU, Brno, Czech Republic.

characterizations; for instance, they are the graphs where every induced path is a shortest path. But perhaps the main reason why distance-hereditary graphs are particularly important in the context of vertex deletion problems is their connection to the structural parameter *rank-width* [24, 23]. While Treewidth-t Vertex Deletion[1] is known to admit a single-exponential fixed-parameter algorithm for every fixed t [12, 22], the existence of such algorithms for the analogous Rank-width-t Vertex Deletion is a challenging open problem. Since distance-hereditary graphs are exactly the graphs of rank-width 1 [23], a single-exponential fixed-parameter algorithm for Distance-Hereditary Vertex Deletion represents the first step towards handling Rank-width-t Vertex Deletion.

Distance-Hereditary Vertex Deletion
Instance : A graph G and an integer k.
Parameter : k.
Task : Is there a vertex set $Q \subseteq V(G)$ with $\lvert Q\rvert \leq k$ such that $G-Q$ is distance-hereditary?

The main result of this paper is an $\mathcal{O}(37^k \cdot |V(G)|^7(|V(G)| + |E(G)|))$-time algorithm for Distance-Hereditary Vertex Deletion, solving an open problem of Kanté, Kim, Kwon, and Paul [20]. The core of our approach exploits two distinct characterizations of distance-hereditary graphs: one by forbidden induced subgraphs (obstructions), and the other by admitting a special kind of split decomposition [7].

The algorithm can be conceptually divided into three parts. First, we use the well-known iterative compression technique [25] to reduce the problem to the easier Disjoint Distance-Hereditary Vertex Deletion, where we assume that the instance additionally contains a certain form of advice to aid us in our computation. Specifically, this advice is a vertex deletion set S to distance-hereditary graphs which is disjoint from and slightly larger than the desired solution. Then we exhaustively apply two branching rules to simplify the given instance of Disjoint Distance-Hereditary Vertex Deletion. At a high level, these branching rules allow us to assume that the resulting instance contains no small obstructions and furthermore that certain connectivity conditions hold on $G[S]$. Lastly, we compute the split decomposition of $G - S$ and exploit the properties of our instance G guaranteed by branching to prune the decomposition. In particular, we show that the connectivity conditions and non-existence of small obstructions mean that S must interact with the split decomposition of $G - S$ in a special way, and this allows us to identify irrelevant vertices in $G - S$. This is by far the most technically challenging part of the algorithm.

A more detailed explanation of our algorithm is provided in Section 3, after the definition of required notions. We complement this result with an algorithmic lower bound which rules out a subexponential fixed-parameter algorithm for Distance-Hereditary Vertex Deletion under well-established complexity assumptions.

The set of induced subgraph obstructions for distance-hereditary graphs consists of three small graphs, and induced cycles of length at least 5. We remark that Heggernes et al. [16] showed that the problem asking whether it is possible to delete k vertices so that the resulting graph has no induced cycles of length at least 5 is W[2]-hard. Therefore, one cannot simply obtain a single-exponential fixed-parameter algorithm for Distance-Hereditary Vertex Deletion using the problem of hitting induced cycles of length at least 5.

The paper is organized as follows. Section 2 contains the necessary preliminaries and notions required for our results. In Section 3, we set the stage for the process of simplifying

[1] Treewidth-t Vertex Deletion asks whether it is possible to delete k vertices so that the resulting graph has treewidth at most t.

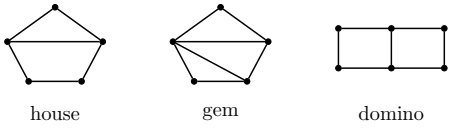

Figure 1 Small DH obstructions which are not cycles.

the split decomposition, which entails the definition of Disjoint Distance-Hereditary Vertex Deletion, introduction of our branching rules, and a few technical lemmas which will be useful throughout the later sections. Section 4 then introduces and proves the safeness of 8 polynomial-time reduction rules; crucially, the exhaustive application of these rules guarantees that the resulting instance will have a certain "inseparability" property. In Section 5, we introduce and prove the safeness of our final reduction rule using this inseparability property. Finally, the proof of our main result as well as the corresponding lower bound are presented in Section 6.

2 Preliminaries

For a graph G, let $V(G)$ and $E(G)$ denote the vertex set and the edge set of G, respectively. For $S \subseteq V(G)$, let $G[S]$ denote the subgraph of G induced on S. For $v \in V(G)$ and $S \subseteq V(G)$, let $G - v$ be the graph obtained from G by removing v, and let $G - S$ be the graph obtained by removing all vertices in S. For $v \in V(G)$, the set of neighbors of v in G is denoted by $N_G(v)$. For $A \subseteq V(G)$, let $N_G(A)$ denote the set of all vertices in $G - A$ that have a neighbor in A. The *length* of a path is the number of edges on the path. For $v \in V(G)$ and a subgraph H of $G - v$, we say v is adjacent to H if it has a neighbor in H.

Two vertices v, w in a graph G are called *twins* if they have the same set of neighbors on $V(G) \setminus \{v, w\}$. For two vertex sets A and B, we say that

- A is *complete* to B if for every $a \in A$, $b \in B$, a is adjacent to b,
- A is *anti-complete* to B if for every $a \in A$, $b \in B$, a is not adjacent to b.

2.1 Distance-Hereditary Graphs

A graph G is called *distance-hereditary* if for every connected induced subgraph H of G and every $v, w \in V(H)$, the distance between v and w in H is the same as the distance between v and w in G. This graph class was first introduced by Howorka [17], and deeply studied by Bandelt and Mulder [3].

The house, the gem, the domino graphs are depicted in Figure 1. A graph isomorphic to one of the house, the gem, the domino, and induced cycles of length at least 5 will be called a *distance-hereditary obstruction* or shortly a *DH obstruction*. A DH obstruction with at most 6 vertices will be called a *small DH obstruction*. Note that every DH obstruction does not contain any twins.

It is known that distance-hereditary graphs are precisely the graphs not containing any DH obstruction as an induced subgraph [3]. The following lemma will be used to find DH obstructions later on.

▶ **Lemma 1** (Kantè, Kim, Kwon, and Paul [20]). *Let G be a graph obtained from an induced path of length at least 3 by adding a vertex v adjacent to its end vertices where v may be adjacent to some internal vertices of the path. Then G has a DH obstruction containing v.*

In particular, if the given path has length at most 4*, then* G *has a small DH obstruction containing* v*.*

2.2 Split decompositions

We follow the notations in [4]. A *split* of a connected graph G is a vertex partition (X, Y) of G such that $|X| \geq 2, |Y| \geq 2$, and $N_G(Y)$ is complete to $N_G(X)$. Splits are also called *1-joins*, or simply *joins* [13]. A connected graph G is called a *prime graph* if $|V(G)| \geq 5$ and it has no split.

A connected graph D with a distinguished set of edges $M(D)$ is called a *marked graph* if the edges in $M(D)$ form a matching and each edge in $M(D)$ is a cut edge. An edge in $M(D)$ is called a *marked edge*, and every other edge is called an *unmarked edge*. A vertex incident with a marked edge is called a *marked vertex*, and every other vertex is called an *unmarked vertex*. Each connected component of $D - M(D)$ is called a *bag* of D.

When G admits a split (X, Y), we construct a marked graph D on the vertex set $V(G) \cup \{x', y'\}$ such that

- for vertices x, y with $\{x, y\} \subseteq X$ or $\{x, y\} \subseteq Y$, $xy \in E(G)$ if and only if $xy \in E(D)$,
- $x'y'$ is a new marked edge,
- X is anti-complete to Y,
- $\{x'\}$ is complete to $N_G(Y) \cap X$ and $\{y'\}$ is complete to $N_G(X) \cap Y$ (with unmarked edges).

The marked graph D is called a *simple decomposition of* G. A *split decomposition* of a connected graph G is a marked graph D defined inductively to be either G or a marked graph defined from a split decomposition D' of G by replacing a connected component H of $D' - M(D')$ with a simple decomposition of H. See Figure 2 for an example of a split decomposition. We note that when D is a split decomposition of a graph G and u, v are two vertices in G, $uv \in E(G)$ if and only if there is a path from u to v in D where its first and last edges are unmarked, and an unmarked edge and a marked edge alternatively appear in the path [1, Lemma 2].

Naturally, we can define a reverse operation of decomposing into a simple decomposition; for a marked edge xy of a split decomposition D, *recomposing* xy is the operation of removing two vertices x and y and making $N_D(x) \setminus \{y\}$ complete to $N_D(y) \setminus \{x\}$ with unmarked edges. It is not hard to observe that if D is a split decomposition of G, then G can be obtained from D by recomposing all marked edges.

Note that there are many ways of decomposing a complete graph or a star, because every its non-trivial vertex partition is a split. Cunningham and Edmonds [8] developed a canonical way to decompose a graph into a split decomposition by not allowing to decompose a bag which is a star or a complete graph. A split decomposition D of G is called a *canonical split decomposition* if each bag of D is either a prime graph, a star, or a complete graph, and D cannot be obtained from a split decomposition with the same property by recomposing a marked edge. It is not hard to observe that every canonical split decomposition has no marked edge linking two complete bags, and no marked edge linking a leaf of a star bag and the center of another star bag [4]. Furthermore, for each pair of twins a, b in G, it holds that a, b must both be located in the same bag of the canonical split decomposition.

▶ **Theorem 2** (Cunningham and Edmonds [8])**.** *Every connected graph has a unique canonical split decomposition, up to isomorphism.*

▶ **Theorem 3** (Dahlhaus [10])**.** *The canonical split decomposition of a graph* G *can be computed in time* $\mathcal{O}(|V(G)| + |E(G)|)$*.*

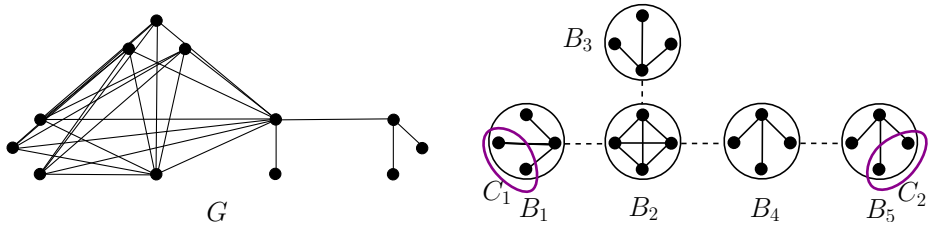

Figure 2 A graph G and its canonical split decomposition. Marked edges are represented by dashed edges, and bags are indicated by circles. Note that $\text{path}(B_1, B_5) = \{B_1, B_2, B_4, B_5\}$, bags B_4, B_5 are (C_1, C_2)-separator bags, and B_4 is a (B_1, B_5)-separator bag.

We can now give the second characterization of distance-hereditary graphs that is crucial for our results. For convenience, we call a bag a *star bag* or a *complete bag* if it is a star or a complete graph, respectively.

▶ **Theorem 4** (Bouchet [4]). *A graph is a distance-hereditary graph if and only if every bag in its canonical split decomposition is either a star bag or a complete bag.*

We will later on also need a little bit of additional notation related to split decompositions. Let D be a canonical split decomposition. For two distinct bags B_1 and B_2, we denote by $\text{comp}(B_1, B_2)$ the connected component of $D - V(B_1)$ containing B_2. Technically, when $B_1 = B_2$, we define $\text{comp}(B_1, B_2)$ to be the empty set. For two bags B_1 and B_2, we denote by $\text{path}(B_1, B_2)$ the set of all bags containing a vertex in a shortest path from B_1 to B_2 in D. Note that $\text{path}(B_1, B_2)$ contains B_1 and B_2. See Figure 2 for an example.

Let C_1, C_2 be two disjoint vertex subsets of D such that each C_1, C_2 is a set of unmarked vertices contained in (not necessarily distinct) bags B_1, B_2, respectively. A bag B is called *a (C_1, C_2)-separator bag* if B is a star bag contained in $\text{path}(B_1, B_2)$ whose center is adjacent to neither $\text{comp}(B, B_1)$ nor $\text{comp}(B, B_2)$. We remark that B can be B_i for some $i \in \{1, 2\}$, and especially when $B_1 = B_2$ and it is a star bag and each C_i consists of leaves of B, B_1 is a (C_1, C_2)-separator bag. For convenience, we also say that a bag B is *a (B_1, B_2)-separator bag* if B is a star bag contained in $\text{path}(B_1, B_2) \setminus \{B_1, B_2\}$ whose center is adjacent to neither $\text{comp}(B, B_1)$ nor $\text{comp}(B, B_2)$. For this notation, B cannot be B_1 nor B_2. It is not hard to check that the length of the shortest path from C_1 to C_2 in the original graph is exactly the same as one plus the number of (C_1, C_2)-separator bags.

3 Setting the Stage

We begin by applying the *iterative compression technique* [25]. This technique allows us to transform our problem to a simpler problem called Disjoint Distance-Hereditary Vertex Deletion. Our goal for the majority of the paper will be to obtain a single-exponential fixed-parameter algorithm for Disjoint Distance-Hereditary Vertex Deletion; this is then used to obtain the sought after algorithm for Distance-Hereditary Vertex Deletion in Section 6.

Disjoint Distance-Hereditary Vertex Deletion
Instance : A graph G, an integer k, and $S \subseteq V(G)$ with $\lvert S\rvert \leq k+1$ such that $G - S$ is distance-hereditary.
Parameter : k.
Task : Is there $Q \subseteq V(G) \setminus S$ with $\lvert Q\rvert \leq k$ such that $G - Q$ is distance-hereditary?

We will denote instances of Disjoint Distance-Hereditary Vertex Deletion as a tuple (G, S, k). By Theorem 4, every connected component of $G - S$ admits a canonical split decomposition whose bags are either a star or a complete graph.

Before explaining the general approach for solving Disjoint Distance-Hereditary Vertex Deletion, it will be useful to introduce a few definitions. Since the canonical split decomposition guaranteed by Theorem 4 only helps us classify twins in $G - S$ and not in G, we explicitly define an equivalence $\sim$ on the vertices of $G - S$ which allows us to classify twins in G: for two vertices $u, v \in V(G - S)$, $u \sim v$ iff they are twins in G.

We denote by $\mathbf{tc}(G - S)$ the set of equivalence classes of $\sim$ on $V(G - S)$, and each individual equivalence class will be called a *twin class* in $G - S$. We can observe that if $U \in \mathbf{tc}(G - S)$ lies in a single connected component of $G - S$, then U must be contained in precisely one bag of the split decomposition of this connected component of $G - S$, as U is a set of twins in $G - S$ as well. A twin class is *S-attached* if it has a neighbor in S, and *non-S-attached* if it has no neighbors in S. Similarly, we say that a bag in the canonical split decomposition of $G - S$ is *S-attached* if it has a neighbor in S, and *non-S-attached* otherwise.

3.1 Overview of the Approach

Now that we have introduced the required terminology, we can provide a high-level overview of our approach for solving Disjoint Distance-Hereditary Vertex Deletion.

1. We exhaustively apply the branching rules described in Section 3.2. Branching rules will be applied when G has a small subset $X \subseteq V(G - S)$ such that $S \cup X$ induces a DH obstruction, or there is a small connected subset $X \subseteq V(G - S)$ such that adding X to S decreases the number of connected components in $G[S]$.
2. We exhaustively apply the initial reduction rules described in Section 4. Each of these rules runs in polynomial time, finds a part in the canonical split decomposition of a connected component of $G - S$ that can be simplified, and modifies the decomposition. Each application of a reduction rule from Section 4 either reduces the number of vertices in $G - S$ or reduces the total number of bags in the canonical split decomposition (of a connected component of $G - S$). It is well known that the total number of bags in the canonical split decomposition of a graph is linear in the number of vertices. Therefore, the total number of applications of these initial reduction rules will also be at most linear in the number of vertices.
3. We say that G and the canonical split decompositions of $G - S$ are *reduced* if the branching rules in Section 3.2 and reduction rules in Section 4 cannot be applied anymore. We will obtain the following simple structure of the decompositions in the reduced instance:
   - Each canonical split decomposition D of a connected component of $G - S$ contains at least two distinct S-attached twin classes (Reduction Rule 1).
   - Each bag contains at most one S-attached twin class (Reduction Rule 3).
   - When B is a bag and D' is a connected component of $D - V(B)$ containing no bags having a neighbor in S, D' consists of one bag and B is a star bag whose center is adjacent to D' (Lemma 8).
   - When B is a bag and D' is a connected component of $D - V(B)$ such that D' contains exactly one S-attached bag B', there is no (B', B)-separator bag (Lemma 10).
4. We choose a canonical split decomposition D of a connected component of $G - S$ and assign any bag as a root bag of D. We choose a bag farthest from the root bag such that there are two descendant bags having S-attached twin classes C_1 and C_2, respectively. Then the length of every shortest path from C_1 to C_2 in $G - S$ is at most 2, and we

introduce a special polynomial-time reduction rule in Section 5 which simplifies this configuration.

Whenever we introduce a new rule, we need to show that it is *safe*; for branching rules this means that there exists at least one subinstance resulting from the rule which is a YES-instance iff the original graph was a YES-instance, while for reduction rules this means that the application of the rule preserves the property of being a YES-instance.

A vertex v in $G - S$ is called *irrelevant* if (G, S, k) is a YES-instance if and only if $(G - v, S, k)$ is a YES-instance. We will be identifying and removing irrelevant vertices in several of our reduction rules. When removing a vertex v from $G - S$, it is easy to modify the canonical split decomposition containing v, and thus it is not necessary to recompute the canonical split decomposition of the resulting graph from scratch [14].

3.2 Branching Rules

We state our two branching rules below.

▶ Branching Rule 1. For every vertex subset X of $G - S$ with $|X| \leq 5$, if $G[S \cup X]$ is not distance-hereditary, then we remove one of the vertices in X, and reduce k by 1.

▶ Branching Rule 2. For every vertex subset X of $G - S$ with $|X| \leq 5$ such that $G[X]$ is connected and the set $N_G(X) \cap S$ is not contained in a connected component of $G[S]$, then we either remove one of the vertices in X and reduce k by 1, or put all of them into S (which reduces the number of connected components of $G[S]$).

The safeness of Branching Rules 1 and 2 are clear, and these rules can be performed in polynomial time. The exhaustive application of these branching rules guarantees the following structure of the instance.

▶ **Lemma 5.** *Let (G, S, k) be an instance reduced under Branching Rules 1 and 2.*

1. *G has no small DH obstructions.*
2. *Let $v \in V(G - S)$. For every two vertices $x, y \in N_G(v) \cap S$, they are contained in the same connected component of $G[S]$ and there is no induced path of length at least 3 from x to y in $G[S]$. Specifically, if $xy \notin E(G)$, then there is an induced path xpy for some $p \in S$.*
3. *There is no induced path $v_1 \cdots v_5$ of length 4 in $G - S$ where v_1 and v_5 have neighbors in S but v_2 and v_4 have no neighbors in S.*
4. *There is no induced path $v_1 \cdots v_4$ of length 3 in $G - S$ where v_1 and v_4 have neighbors on S but v_2 has no neighbors on S.*

Lemma 5, and especially point (2) in the lemma, is used in many parts of our proofs. Since we will apply the branching rules exhaustively at the beginning and also after each new application of a reduction rule, these properties will be implicitly assumed to hold in subsequent sections.

4 Reduction Rules in Split Decompositions

In this section, we assume that the given instance (G, S, k) is reduced under Branching Rules 1 and 2. The reduction rules introduced here either remove some irrelevant vertex, or move some vertex into S, or reduce the number of bags in the decomposition by modifying the instance into an equivalent instance. After we apply any of these reduction rules, we will run the two branching rules from Section 3 again.

Before we move on to the reduction rules themselves, we introduce a generic way of finding an irrelevant vertex which will be used in many reduction rules. For a vertex v in $G - S$ and an induced cycle H of length at least 5 in G containing a vertex v and two neighbors w, z of v in H, a vertex v' in S is called a *bad vertex* for H and v if v' is adjacent to w and z. If such a vertex v' exists, it is clear that v' is not contained in H because $vwv'zv$ is a cycle of length 4. More importantly, since $H - v$ is an induced path of length at least 3 from w to z and v' is adjacent to both of its endpoints, by Lemma 1, $G[(V(H) \setminus \{v\}) \cup \{v'\}]$ contains a DH obstruction. This implies that one of the vertices in $V(H) \setminus \{v\}$ must be contained in every solution (note that $v' \in S$ and so v' itself cannot be part of a solution). This property results in the following two lemmas.

▶ **Lemma 6.** *Let (G, S, k) be an instance reduced under Branching Rule 1. Let v be a vertex in $G - S$ such that for every induced cycle H of length at least 7 containing v, there is a bad vertex for H and v. Then v is irrelevant.*

▶ **Lemma 7.** *Let (G, S, k) be an instance reduced under Branching Rules 1 and 2. Let v be a vertex in $G - S$ and H be an induced cycle of length at least 7 containing v, and let w, z be the two neighbors of v in H. If $w, z \in S$, then there is a bad vertex for H and v, and thus $G[(V(H) \setminus \{v\}) \cup S]$ contains a DH obstruction.*

We are now ready to start with our reduction rules. For the remainder of this section, let us fix a canonical split decomposition D of a connected component of $G - S$.

▶ Reduction Rule 1. If D has at most one S-attached twin class, then we remove all unmarked vertices of D from G.

▶ Reduction Rule 2. Let B be a star bag whose center is unmarked, and let v be a leaf unmarked vertex in B. If v has no neighbor in S, then we remove v. If v has a neighbor in S, then we move v into S.

We remark that when we move v into S in Reduction Rule 2, $k + cc(G[S])$ does not increase. Next, we introduce an important rule which reduces the number of S-attached twin classes in each bag.

▶ Reduction Rule 3. Let B be either a complete bag or a star bag whose center is marked. Let C_1, C_2 be two distinct S-attached twin classes in B such that $(N_G(C_1) \setminus N_G(C_2)) \cap S$ is non-empty. Then we remove C_1.

We proceed by introducing a reduction rule which sequentially arranges non-S-attached bags in a canonical split decomposition. The number of bags in D is strictly reduced when applying Reduction Rule 4.

▶ Reduction Rule 4. Let B be a leaf bag and B' be the neighbor bag of B.

1. If B is a complete bag having exactly one twin class and B' is a star bag whose leaf is adjacent to B, then we transform B into a star whose center is adjacent to B', and recompose the marked edge connecting B and B'.
2. If B is a star bag having exactly one twin class, the center of B is adjacent to B', and B' is a complete bag, then we transform B into a complete graph, and recompose the marked edge connecting B and B'.

The next reduction rule allows us to remove a non-S-attached twin class under certain conditions (see Figure 3).

▶ Reduction Rule 5. Let B_1 be a leaf bag containing at most one S-attached twin class and B_2 be a bag distinct from B_1 such that

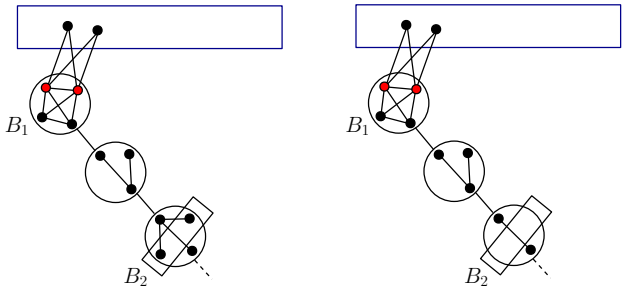

Figure 3 Reduction Rule 5.

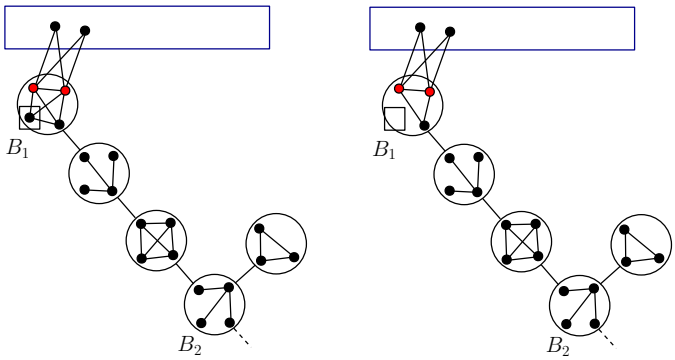

Figure 4 Reduction Rule 6.

- every bag in $\text{path}(B_1, B_2) \setminus \{B_1, B_2\}$ is non-S-attached, not a (B_1, B_2)-separator bag, and has exactly two neighbor bags, and
- B_2 is a star bag whose center is adjacent to $\text{comp}(B_2, B_1)$.

If B_2 contains a non-S-attached twin class C, then we remove C.

We can now show that after the exhaustive application of the reduction rules introduced up to this point, every connected component of $D - V(B)$ containing no S-attached bags is "simple", as formalized in the next lemma.

▶ **Lemma 8.** *Let D be the canonical split decomposition of a connected component of $G - S$ reduced under Reduction Rules 1–5. Let B be a bag and D' be a connected component of $D - V(B)$ containing no S-attached bags. Then D' consists of one bag and B is a star bag whose center is adjacent to D'.*

Next, we introduce some rules simplifying connected components of $D - V(B)$ for some bag B containing one S-attached twin class. The following rule is depicted in Figure 4.

▶ Reduction Rule 6. Let B_1 be a leaf bag having exactly one S-attached twin class and B_2 be a bag distinct from B_1 such that

- B_1 is not a star whose leaf is adjacent to a neighboring bag,
- every bag in $\text{path}(B_1, B_2) \setminus \{B_1, B_2\}$ is non-S-attached, not a (B_1, B_2)-separator bag and has exactly two neighbor bags, and
- B_2 is a star whose center is either an unmarked vertex, or adjacent to a connected component of $D - V(B_2)$ consisting of one non-S-attached bag.

If B_1 contains a non-S-attached twin class C, then we can safely remove C.

By applying Reduction Rules 4, 5, and 6, we can simplify the decomposition near an S-attached leaf containing one S-attached twin class; for instance, in Figure 4, B_1 will be

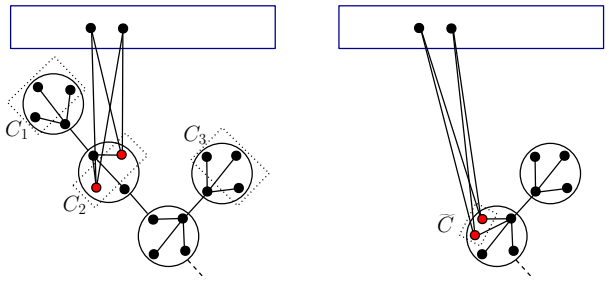

Figure 5 Reduction Rule 8.

eventually merged with B_2. We state the properties that are guaranteed by the reduction rules introduced up to this point in the following lemma.

▶ **Lemma 9.** *Let D be the canonical split decomposition of a connected component of $G - S$ reduced under Reduction Rules 1–6. Let B be a star bag whose center is unmarked or adjacent to a connected component of $D - V(B)$ consisting of one non-S-attached bag. Let D' be a connected component of $D - V(B)$ such that*

- *D' contains exactly one S-attached bag B', and*
- *there is no (B', B)-separator bag.*

Then B' is a star whose leaf is adjacent to $\mathrm{comp}(B', B)$ and there is a leaf bag B'' where the center of B' is adjacent to B''.

The final two rules in this section help us simplify the configuration specified in Lemma 9; using Reduction Rule 7 we can remove all unmarked vertices in $\mathrm{path}(B, B') \setminus \{B, B'\}$, and then Reduction Rule 8 allows us to merge B' with B.

▶ Reduction Rule 7. Let B_1 and B_2 be two star bags in D such that

- for each i, either the center of B_i is an unmarked vertex, or the center of B_i is adjacent to a connected component of $D - V(B_i)$ consisting of one non-S-attached bag,
- every bag in $\mathrm{path}(B_1, B_2) \setminus \{B_1, B_2\}$ is a non-S-attached bag, has two neighbor bags, and is not a (B_1, B_2)-separator bag.

Then we remove every unmarked vertex in every bag in $\mathrm{path}(B_1, B_2) \setminus \{B_1, B_2\}$.

▶ Reduction Rule 8. Let B_1, B_2, B_3 be distinct bags in D such that

- B_1 is a non-S-attached leaf bag whose neighbor bag is B_2, and it is not a star whose leaf is adjacent to B_2,
- B_2 has exactly two neighbor bags B_1 and B_3, it is a star whose center is adjacent to B_1, and the set of unmarked vertices in B_2 is the unique S-attached twin class C_2 in B_2, and
- B_3 is a star whose center is either an unmarked vertex, or adjacent to a connected component of $D - V(B_3)$ consisting of one non-S-attached bag.

Let C_1 be the set of unmarked vertices in B_1. Then we remove B_1 and B_2, and add a leaf set of unmarked vertices $\widetilde{C}$ with $\min(|C_1|, |C_2|)$ vertices to B_3, that is complete to $N_G(C_2) \cap S$ and has no other neighbors in S.

We provide an illustration of Reduction Rule 8 in Figure 5.

Finally, after applying all the reduction rules in this section, our instance has the desired inseparability property. We formalize and prove this property below.

▶ **Lemma 10.** *Let D be the canonical split decomposition of a connected component of $G - S$ reduced under Reduction Rules 1–8. Let B be a bag and let D' be a connected component of $D - V(B)$ such that D' contains exactly one S-attached bag B'. Then there is no (B', B)-separator bag.*

▶ **Proposition 11.** Let (G, S, k) be an instance reduced under Branching Rules 1 and 2. Given a canonical split decomposition D of a connected component of $G - S$, we can in time $\mathcal{O}(|V(G)|^2)$ either apply one of Reduction Rules 1–8, or correctly answer that Reduction Rules 1–8 cannot be applied anymore.

5 Twin Class Reduction Rule

In this section, we introduce our last, but perhaps most important, reduction rule. Later on in the proof of Theorem 13, we will show that whenever the other rules cannot be applied, we can either apply Reduction Rule 9 or our instance is trivial.

▶ Reduction Rule 9. Suppose that (G, S, k) and all canonical split decompositions of connected components of $G-S$ are reduced under Branching Rules 1–2 and Reduction Rules 1–8. Let D be the canonical split decomposition of a connected component of $G - S$, and let B be a bag, and B_1, B_2 be two distinct S-attached bags (possibly $B_i = B$ for some $i \in \{1, 2\}$). Furthermore, let C_1, C_2 be two distinct S-attached twin classes in B_1, B_2, respectively, such that for each $i \in \{1, 2\}$, either $B_i = B$ or C_i is the unique S-attached twin class in $\mathrm{comp}(B, B_i)$. Then we apply one of the following:

1. If the distance from C_1 to C_2 in $G - S$ is 2 and the unique (C_1, C_2)-separator bag is contained in $\mathrm{comp}(B, B_2)$, then we remove C_2. (We show that B cannot be the (C_1, C_2)-separator bag.)
2. If C_1 is complete to C_2, $B \neq B_2$, and B is a star bag whose center is adjacent to $\mathrm{comp}(B, B_2)$, then we remove C_1.
3. If C_1 is complete to C_2, $B \neq B_1$, and B is a complete bag, then B_1 contains a non-S-attached twin class C_1' and we remove C_1'.

▶ **Proposition 12.** Reduction Rule 9 is safe.

Sketch of Proof. Here we prove the proposition for one important special case. Suppose that C_1 is anti-complete to C_2 and the (C_1, C_2)-separator bag is contained in $\mathrm{comp}(B, B_2)$. We claim that every vertex in C_2 is irrelevant. For each $i \in \{1, 2\}$, let $c_i \in C_i$ and let $T_i = N_G(C_i)$. Let B' be the (C_1, C_2)-separator bag. We first confirm that $B_2 = B'$. If not, then B' is a (B_2, B)-separator bag. However, since $\mathrm{comp}(B, B_2)$ has exactly one S-attached bag B_1, by Lemma 10, there is no (B_2, B)-separator bag, a contradiction. We conclude that $B' = B_2$. There is a leaf bag B_2' where the center of B_2 is adjacent to B_2', otherwise, we can apply Reduction Rule 2.

Let $v \in C_2$. We claim that for every induced cycle H of length at least 7 containing v, there is a bad vertex for H and v. If this is true, then the result follows from Lemma 6. Let w and z be the two neighbors of v in H. If w and z are contained in S, then by Lemma 7, there is a bad vertex. On the other hand, w and z cannot be contained in $V(G-S)$ together, because the vertices in B_2' form a twin class. We may assume that $w \in (T_1 \cap T_2) \cap V(G-S)$ and $z \in S$. We actually show that this is not possible. Note that since $w \in V(B_2')$, w has no neighbors in S.

We divide cases depending on the location of z: specifically, to conclude the proof, we separately consider the case of $z \in (T_2 \setminus T_1) \cap S$ and $z \in (T_1 \cap T_2) \cap S$. We show that the former case always leads to a contradiction with w having no neighbors in S. On the other hand, it can be shown that the latter case necessarily implies the existence of a small DH obstruction, contradicting the exhaustive application of Branching Rules 1–2. ◀

6 The Algorithm and Lower Bounds

Our goal in this section is to give a proof of our main result, Theorem 14, and prove corresponding lower bounds.

▶ **Theorem 13.** Disjoint Distance-Hereditary Vertex Deletion *can be solved in time* $\mathcal{O}(36^k \cdot |V(G)|^6(|V(G)| + E(G)))$.

Sketch of Proof. The main argument in the proof is that whenever we cannot apply one of Branching Rules 1–2 and Reduction Rules 1–8, either we have a trivial instance, or we run into a situation where we can apply Reduction Rule 9. Suppose that D is the canonical split decomposition of a connected component of $G - S$ such that G and D are reduced under those rules. If D contains at most one S-attached twin class, then we can apply Reduction Rule 1. Thus, we know that D contains at least two distinct S-attached twin classes.

We choose a root bag of D, and choose a bag B that is farthest from the root bag such that there are two descendant bags B_1, B_2 of B having distinct S-attached twin classes C_1, C_2, respectively. By Reduction Rule 3, we have $B_1 \neq B_2$. Using the structure that if $B_i \neq B$, then there is no (B_i, B)-separator bag by Lemma 10, we can observe that the distance between C_1 and C_2 in $G - S$ is at most 2, and then C_1 and C_2 satisfy one of the conditions in Reduction Rule 9.

We can notice that each branching rule reduces either k or the number of connected components in S and branch into at most 6 subinstances. Since none of the reduction rules change k or the number of components in S, branching rules are applied at most $2k$ times. Due to the application of reduction rules (which we also consider as nodes of the branching tree and which may be applied independently in different branches), the branching tree will have at most $\mathcal{O}(36^k \cdot |V(G)|)$ nodes, and the runtime in every node will not exceed $\mathcal{O}(|V(G)|^5(|V(G)| + |E(G)|))$. Hence, the whole algorithm for Disjoint Distance-Hereditary Vertex Deletion can be implemented in time $\mathcal{O}(36^k \cdot |V(G)|^6(|V(G)| + |E(G)|))$. ◀

▶ **Theorem 14.** Distance-Hereditary Vertex Deletion *can be solved in time* $\mathcal{O}(37^k \cdot |V(G)|^7(|V(G)| + |E(G)|))$.

Sketch of Proof. Let $n := |V(G)|$ and $m := |E(G)|$. Fix an arbitrary labeling $v_1, \ldots, v_n$ of $V(G)$ and let $G_i := G[\{v_1, \ldots, v_i\}]$ for $1 \leq i \leq n$. From $i = 1$ up to n, given a graph G_i and $S_i \subseteq V(G_i)$ with $|S_i| \leq k + 1$ such that $G_i - S_i$ is distance-hereditary, we aim to find a set $S_i' \subseteq V(G_i)$ with $|S_i'| \leq k$ such that $G_i - S_i'$ is distance-hereditary if one exists. We further guess all possible $S_i' \cap S_i$ as I, and we aim to find a deletion set S_i'' of size at most $k - |I|$ in $G_i - I$ where $S_i'' \cap (S_i \setminus I) = \emptyset$ if one exists. We can recursively resolve this problem using Disjoint Distance-Hereditary Vertex Deletion. As we iterate the subproblem n times, we obtain the runtime $n \cdot \sum_{i=0}^{k} \binom{k+1}{i} \cdot \mathcal{O}(36^{k-i} \cdot n^6(n+m)) = \mathcal{O}(37^k \cdot n^7(n+m))$. ◀

Our lower bound result is based on the well-established exponential time hypothesis [19], and specifically uses the fact that there is no $2^{o(k)} \cdot |V(G)|^{\mathcal{O}(1)}$ algorithm for Vertex Cover, unless ETH fails [5]. The proof relies on a reduction which is similar to the one used for vertex deletion to graphs of linear rank-width 1 [20].

▶ **Theorem 15.** *There is no* $2^{o(k)} \cdot |V(G)|^{\mathcal{O}(1)}$ *algorithm for* Distance-Hereditary Vertex Deletion *unless ETH fails.*

7 Concluding Notes

We conclude with a few remarks on why we believe that the presented algorithm is of high interest. First, it intrinsically exploits the properties guaranteed by distinct, seemingly unrelated characterization of distance-hereditary graphs; this approach can likely be used to design or improve algorithms for other vertex deletion problems. Second, it uses highly nontrivial reduction rules which simplify canonical split decompositions, and an adaptation or extension of the presented rules could be highly relevant for other graph classes characterized by special canonical split decompositions, such as parity graphs [6] or circle graphs [13]. Third, it is the first of its kind which targets a "full" class of graphs of bounded rank-width (contrasting previous results for specific subclasses of graphs of rank-width 1 [18, 2, 21, 20]).

It is worth noting that there remains a number of interesting open problems in this general area. Perhaps the most prominent one is the question of whether vertex deletion to graphs of rank-width c, for any constant c, admits a single-exponential fixed-parameter algorithm. Our algorithm represents the first steps in this general direction. The existence of a polynomial kernel or an approximation algorithm for such vertex deletion problems also remains open, even for the case of distance-hereditary graphs.

References

1 Isolde Adler, Mamadou Moustapha Kanté, and O-joung Kwon. Linear rank-width of distance-hereditary graphs. In *Graph-Theoretic Concepts in Computer Science - 40th International Workshop, WG 2014, Nouan-le-Fuzelier, France, June 25-27, 2014. Revised Selected Papers*, pages 42–55, 2014. doi:10.1007/978-3-319-12340-0_4.

2 Akanksha Agrawal, Sudeshna Kolay, Daniel Lokshtanov, and Saket Saurabh. A faster FPT algorithm and a smaller kernel for block graph vertex deletion. In *LATIN 2016: Theoretical Informatics - 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings*, pages 1–13, 2016.

3 Hans-Jürgen Bandelt and Henry M. Mulder. Distance-hereditary graphs. *J. Combin. Theory Ser. B*, 41(2):182–208, 1986. doi:10.1016/0095-8956(86)90043-2.

4 André Bouchet. Transforming trees by successive local complementations. *J. Graph Theory*, 12(2):195–207, 1988.

5 Liming Cai and David Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences*, 67(4):789 – 807, 2003. doi:10.1016/S0022-0000(03)00074-6.

6 Serafino Cicerone and Gabriele Di Stefano. On the extension of bipartite to parity graphs. *Discrete Applied Mathematics*, 95(1–3):181–195, 1999. doi:10.1016/S0166-218X(99)00074-8.

7 William H. Cunningham. Decomposition of directed graphs. *SIAM J. Algebraic Discrete Methods*, 3(2):214–228, 1982.

8 William H. Cunningham and Jack Edmonds. A combinatorial decomposition theory. *Canad. J. Math.*, 32(3):734–765, 1980. doi:10.4153/CJM-1980-057-7.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

10 Elias Dahlhaus. Parallel algorithms for hierarchical clustering, and applications to split decomposition and parity graph recognition. *Journal of Algorithms*, 36(2):205–240, 2000. doi:10.1006/jagm.2000.1090.

11 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

12 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar f-deletion: Approximation, kernelization and optimal FPT algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 470–479, 2012.

13 Csaba P. Gabor, Kenneth J. Supowit, and Wen Lian Hsu. Recognizing circle graphs in polynomial time. *J. Assoc. Comput. Mach.*, 36(3):435–473, 1989.

14 Emeric Gioan and Christophe Paul. Split decomposition and graph-labelled trees: characterizations and fully dynamic algorithms for totally decomposable graphs. *Discrete Appl. Math.*, 160(6):708–733, 2012. doi:10.1016/j.dam.2011.05.007.

15 Peter L. Hammer and Frédéric Maffray. Completely separable graphs. *Discrete Applied Mathematics*, 27(1-2):85–99, 1990. doi:10.1016/0166-218X(90)90131-U.

16 Pinar Heggernes, Pim van 't Hof, Bart M.P. Jansen, Stefan Kratsch, and Yngve Villanger. Parameterized complexity of vertex deletion into perfect graph classes. *Theoretical Computer Science*, 511:172 – 180, 2013. doi:10.1016/j.tcs.2012.03.013.

17 E. Howorka. A characterization of distance-hereditary graphs. In *The Quarterly Journal of Mathematics, Oxford, Second Series*, 28 (112):417–420, 1977.

18 Falk Hüffner, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.*, 47(1):196–217, 2010. doi:10.1007/s00224-008-9150-x.

19 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.

20 Mamadou Moustapha Kanté, Eun Jung Kim, O-joung Kwon, and Christophe Paul. An FPT Algorithm and a Polynomial Kernel for Linear Rankwidth-1 Vertex Deletion. In Thore Husfeldt and Iyad Kanj, editors, *10th International Symposium on Parameterized and Exact Computation (IPEC 2015)*, volume 43 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 138–150, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.IPEC.2015.138.

21 Eun Jung Kim and O-joung Kwon. A Polynomial Kernel for Block Graph Deletion. In Thore Husfeldt and Iyad Kanj, editors, *10th International Symposium on Parameterized and Exact Computation (IPEC 2015)*, volume 43 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 270–281, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.IPEC.2015.270.

22 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 *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 613–624, 2013. doi:10.1007/978-3-642-39206-1_52.

23 Sang-il Oum. Rank-width and vertex-minors. *J. Comb. Theory, Ser. B*, 95(1):79–100, 2005. URL: http://dx.doi.org/10.1016/j.jctb.2005.03.003, doi:10.1016/j.jctb.2005.03.003.

24 Sang-il Oum. Rank-width and well-quasi-ordering. *SIAM J. Discrete Math.*, 22(2):666–682, 2008. doi:10.1137/050629616.

25 Bruce Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299 – 301, 2004. doi:10.1016/j.orl.2003.10.009.