The Parameterized Complexity of Finding Concise Local Explanations

Sebastian Ordyniak[1] , **Giacomo Paesani**[1] and **Stefan Szeider**[2]
[1]University of Leeds, UK
[2]Algorithms and Complexity Group, TU Wien, Vienna, Austria
{s.ordyniak, g.paesani}@leeds.ac.uk, sz@ac.tuwien.ac.at

Abstract

We consider the computational problem of finding a smallest local explanation (anchor) for classifying a given feature vector (example) by a black-box model. After showing that the problem is NP-hard in general, we study various natural restrictions of the problem in terms of problem parameters to see whether these restrictions make the problem fixed-parameter tractable or not. We draw a detailed and systematic complexity landscape for combinations of parameters, including the size of the anchor, the size of the anchor's coverage, and parameters that capture structural aspects of the problem instance, including rank-width, twin-width, and maximum difference.

1 Introduction

Explainable AI aims to help humans understand the complex decisions made by the opaque inner workings of machine learning (ML) models. Ideally, one tries to explain the entire ML model (global explanation) or replace an opaque model with an interpretable one [Angelino *et al.*, 2017; Hu *et al.*, 2019; Rudin, 2019; Ignatiev *et al.*, 2021]. However, this approach has limits, as many applications require using non-interpretable models such as Neural Networks. In such cases, one can fall back to a local model-agnostic approach, which generates post-hoc explanations of individual decisions made by the opaque model (local explanation), treating the model as a black box [Ribeiro *et al.*, 2016; Lundberg and Lee, 2017; Shrikumar *et al.*, 2017; Shrotri *et al.*, 2022].

A central concept for local model-agnostic explanations is the notion of an *anchor* [Ribeiro *et al.*, 2018]. An anchor provides a local explanation in terms of a subset of features that "anchors" the prediction or classification of a given data point (or example) locally. That means changes to the rest of the feature values of the example do not matter. Thus, an anchor determines a small set of features sufficient to anchor the prediction. Practical heuristics for computing anchors have been successfully applied in many scenarios and come with the advantage of offering human-understandable explanations. For this reason, one prefers anchors that are small, providing concise explanations, while still applying to a large region.

For instance, assume a loan application has been declined, and we want to compute an anchor that could consist of two features, say, the applicant's income and age. Ideally, there should be no other applicants with the same income and age whose application has been accepted. The anchor's robustness can be measured in terms of the anchor's *coverage* (previous applicants with the same income and age whose application has been rejected), either in absolute numbers or as the fraction between the size of the coverage and the total number of applications.

In this paper, we provide the first study of the computation complexity of computing small anchors. We show that the exact computation of small anchors is NP-hard. Based on this insight, we roll out a systematic investigation of the *parameterized complexity* of this problem by taking various properties of the sought-for anchor or structural properties of the input data in terms of problem parameters into account. Our results draw a detailed complexity-landscape of the problem (see Figure 1), indicating regions where the problem is

1. fixed-parameter tractable (i.e., solvable in uniform polynomial time for fixed parameter values),
2. XP-tractable (i.e., solvable in nonuniform-polynomial time for fixed parameter values), and
3. paraNP-hard (remains NP-hard even for fixed parameter values).

Parameterized complexity considers problems in a two-dimensional setting, where a problem instance is a pair (I, k), where I is the main part and k is the parameter. A parameterized problem is *fixed-parameter tractable* if there exists a computable function f such that instances (I, k) can be solved in time $f(k)\|I\|^{O(1)}$; one can show that a problem is unlikely to be fixed-parameter tractable by showing it to be hard for the complexity classes W[1] or W[2]. The problem is XP-*tractable* if it can be solved in time $\|I\|^{f(k)}$. Finally, a parameterized problem is paraNP-hard if the problem is NP-hard if the parameter is replaced by some constant. paraNP-hard problems are not XP-tractable unless P = NP. For a more in-depth treatment of parameterized complexity we refer to other sources [Cygan *et al.*, 2015; Downey and Fellows, 2013].

Our results extend a recent line of research that investigates explainability and interpretability of ML models from a (parameterized) complexity theoretic perspective [Barceló

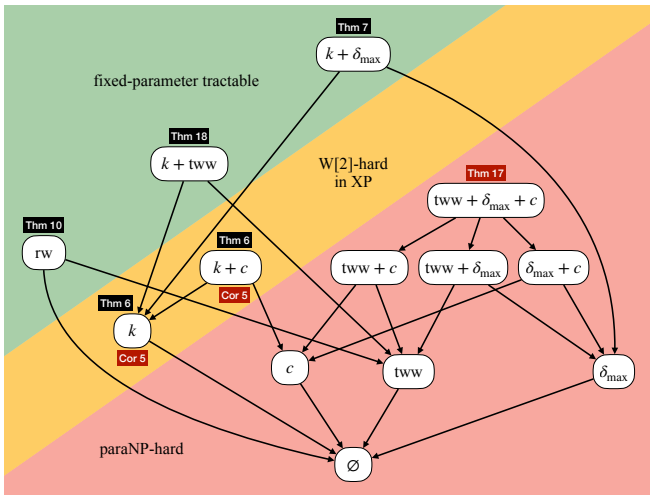

Figure 1: Diagram of the parameterized complexity of MA when parameterized by subsets of $\{k, rw, tww, c, \delta_{\max}\}$; we refer to Sections 2 and 3 for a definition of these parameters. The background color indicates the parameterized complexity classification as indicated, black and red boxes indicate the results establishing the classification. An arrow from a combination of parameters A to a combination of parameters B indicates that B dominates A. Note that the tractability result for $tww{+}k$ requires a witness for small twin-width to be given as part of the input. We omit combinations of parameters for which a subset is already fpt. Thus, from all 32 combinations or the parameters k, rw, tww, $\delta_{\max}$, and c, eight are paraNP-hard, two are W[2]-hard and in XP, and the remaining 22 are FPT.

et al., 2020; Ganian *et al.*, 2018; Kobourov *et al.*, 2022; Ordyniak and Szeider, 2021; Simonov *et al.*, 2019].

2 Problem Definition

A *classified data set (CDS)* is a triple (E, F, m) where E is a multi-set of *examples* over the set F of features and m is a mapping, called *classification*, that assigns each example $e \in E$ a non-negative integer $m(e)$. Here, an *example* is a mapping $e : F \to \mathbb{Z}$. We think of m as being computed by a black-box model that has been trained earlier from a separate set of examples and whose classifications we want to explain.

We say that two examples e_1, e_2 *agree* on a feature $f \in F$ if $e_1(f) = e_2(f)$; otherwise they *disagree on* f.

Given a CDS (E, F, m), an *anchor* for an example $x \in E$ is a set $A \subseteq F$ of features such that all $e \in E$ with $m(x) \neq m(e)$ disagree with x on some feature in A. The *coverage* of A is the number of all examples $e \in E \setminus \{x\}$ with $m(e) = m(x)$ that agree with x on all features in A. We can now formulate our central computational problem:

MINIMUM ANCHOR (MA)	
Input:	A CDS (E, F, m), an example $x \in E$, and two integers k, c.
Question:	Is there an anchor A for x of size at most k and coverage at least c?

For an instance $I = ((E, F, m), x, k, c)$ of MA, we denote by pE and nE the subsets of E containing all examples $e \in E$ such that $m(e) = m(x)$ and $m(e) \neq m(x)$, respectively. We will refer to the examples in pE and nE as *positive* and *negative* examples, respectively. Instead of the size c of the coverage we might want to achieve a coverage that includes a certain fraction q of the examples in E; in this case, we can put $c = q \cdot |pE|$. Interestingly, since our results show that the parameter c is not necessary for any of our tractability results as can seen from Figure 1, i.e., for all considered parameterization the complexity remains the same after adding c as an additional parameter, it follows that all our results still hold when we require the coverage to have a certain fraction q instead of a size c.

Let $I = ((E, F, m), x, k, c)$ be an instance of MA. In the following we will show that we can normalize I into an equivalent instance of MA that has only two classes and where the domain of every feature is Boolean. We denote by $N(I) = ((E', F, m'), x', k, c)$ the *normalized* instance of I that is defined as follows. E' is the set $\{e' \mid e \in E\}$ such that, for every $f \in F$, $e'(f) = 0$ if $e(f) = x(f)$ and $e'(f) = 1$ otherwise. Moreover, $m'(e') = 0$ if $m(e) \neq m(x)$ and $m'(e') = 1$ otherwise. Note that x' now satisfies that $m(x') = 1$ and $x(f) = 0$ for every $f \in F$. We say that I is *normalized* if $I = N(I)$. Thus, all the features of a normalized instance are Boolean, i.e., range over the domain $\{0, 1\}$ and m is Boolean as well, i.e., $m(e) \in \{0, 1\}$ for all $e \in E$. The following lemma shows that I and $N(I)$ are equivalent instances of MA.

Lemma 1. *An instance $I = ((E, F, m), x, k, c)$ of* MA *has a solution if and only if so does the instance $N(I)$.*

We will consider several parameters that restrict the graphical structure of the MA instance I by means of its *incidence graph* $G_I(I)$. This is the bipartite graph with vertex-partition (E, F) having an edge between an example $e \in E$ and a feature $f \in F$ iff $e(f) \neq x(f)$, i.e., the edge indicates that the example e disagrees with x on feature f.

Finally, we would like to note that MA is computational equivalent to the following interesting and natural generalization of the well-known DOMINATING SET problem.

MA DOMINATING SET	
Input:	A graph $G = (V, E)$, where V is partitioned into B, P, and N, and two integers k and c.
Question:	Is there a set $D \subseteq B$ with $\lvert D\rvert \leq k$ that dominates N, i.e., $N \subseteq N_G(D)$, and leaves at least c vertices in P not dominated, i.e., $\lvert P \setminus N_G(D)\rvert \geq c$? Here, $N_G(D)$ denotes the set of all neighbors of any vertex in D.

The equivalence between the two problems follows because there is a one-to-one correspondence between the incidence graph of an MA instance and an instance of MA DOMINATING SET; after setting B to the set of features, P to the positive examples, and N to the set of negative examples.

3 Structural Parameters

Besides the *anchor size* k and the *coverage size* c, we consider *structural parameters* that depend on the given instance $I = ((E, F, m), x, k, c)$ of MA.

We first consider maximum difference, which has been used in a related context by Ordyniak and Szeider [2021]. The *maximum difference* $\delta_{\max}(X)$ of a CDS $X = (E, F, m)$ is the largest number of features on which any two examples $e, e' \in E$ disagree on. This parameters is attractive as it has been observed that for real-world instances, the maximum difference tends to be surprisingly small [Ordyniak and Szeider, 2021]. In our setting, we define the *maximum difference* $\delta_{\max}(I)$ of an MA instance $I = ((E, F, m), x, k, c)$ as the largest number of features any example $e \in E \setminus \{x\}$ with $m(e) \neq m(x)$ differs from x. Consequently $\delta_{\max}(I) \leq \delta_{\max}(X)$, thus it is likely that the parameter is even smaller than the one considered by Ordyniak and Szeider.

The following observation (whose proof follows along the same lines as the proof of Lemma 1) shows that neither the parameter $\delta_{\max}(I)$ nor the incidence graph of I change after normalization. This will later allow us to work solely on the simpler normalized instances.

Observation 2. *Let $I = ((E, F, m), x, k, c)$ be an instance of* MA. *Then, $G_I(I) = G_I(N(I))$ and $\delta_{\max}(I) = \delta_{\max}(N(I))$.*

There is a plethora of width parameters for graphs that we can utilize for MA by means of the incidence graph. We cover this field of width parameters by considering three of the most fundamental ones. For comparing the generality of width parameters p and q, the following notion is useful. We say that p *dominates* q if, for every class of instances for which q is bounded, p is also bounded; p *strictly dominates* q if p dominates q, but q does not dominate p. If p and q dominate each other, they are *domination-equivalent*.

treewidth (tw) is perhaps the most famous width parameter which measures how "tree-like" a graph is. Graphs of bounded treewidth are sparse [Bodlaender, 2007].

rank-width (rw) strictly dominates treewidth, as also dense graphs (even cliques) can have small rank-width, but any class of graphs of bounded treewidth has also bounded rank-width [Oum and Seymour, 2006].

twin-width (tww) has been recently introduced [Bonnet *et al.*, 2022b]. It strictly dominates rank-width, as any class of graphs of bounded rank-width has also bounded twin-width, but the reverse isn't true. For instance, planar graphs have bounded twin-width but can have arbitrarily high rank-width[1].

Thus, for an instance $I = ((E, F, m), x, k, c)$ of MA we simply write $tw(I) = tw(G_I(I))$, $rw(I) = rw(G_I(I))$, and $tww(I) = tww(G_I(I))$.

The following is now a direct consequence of Observation 2 and implies that none of our parameters changes after normalization.

Corollary 3. *Let $I = ((E, F, m), x, k, c)$ be an instance of* MA. *Then, $\delta_{\max}(I) = \delta_{\max}(N(I))$, $tw(I) = tw(N(I))$, $rw(I) = rw(N(I))$, and $tww(I) = tww(N(I))$.*

[1]Bonnet *et al.* [2022b; 2021a; 2021b] list more than a dozen of diverse fundamental graph classes which have bounded twin-width, and for all of them corresponding decompositions can be fpt-approximated.

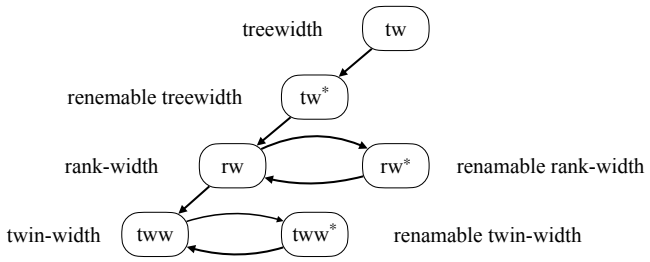

Figure 2: Domination between width parameters and their renaming variants. $a \to b$ means that b dominates a.

E	f_1	f_2	f_3	f_4	f_5	f_6	class
x	1	0	-2	1	0	-1	1
e_1	1	0	-2	3	2	-2	1
e_2	1	0	-2	2	0	-1	1
e_3	1	0	-2	5	0	-2	1
e_4	1	5	-2	4	1	4	1
e_5	-1	0	-2	0	1	1	2
e_6	1	0	-1	1	-1	0	2
e_7	0	0	0	0	0	0	3
x'	0	0	0	0	0	0	1
e'_1	0	0	0	1	1	1	1
e'_2	0	0	0	1	0	0	1
e'_3	0	0	0	1	0	1	1
e'_4	0	1	0	1	1	1	1
e'_5	1	0	0	1	1	1	0
e'_6	0	0	1	0	1	1	0
e'_7	1	0	1	1	0	1	0

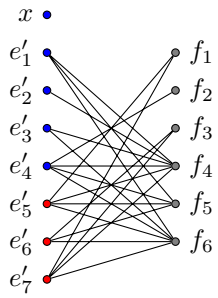

Figure 3: An instance I of MA with CDS X and example x (top part of the table), the normalized instance $N(I)$ (bottom part of the table) and the incidence graph $G_I(I)$ (with blue/red vertices indicating positive/negative examples) of I (right). Note that, for example, $\{f_1, f_3\}$ is an anchor for x with coverage 4.

Therefore, in the following (and throughout the rest of the paper), we can assume that any instance $I = ((E, F, m), x, k, c)$ of MA is normalized.

By the notion of *renaming*, borrowed from a similar notion for propositional CNF formulas [Lewis, 1978], we can further strengthen width parameters for MA instances. A renaming I_R of a normalized MA instance $I = ((E, F, m), x, k, c)$ with respect to a subset $R \subseteq F$ is the MA instance $I_R = ((E_R, F, m_R), x_R, k, c)$ where $E_R = \{ e_R \mid e \in E \}$ with $e_R(f) = 1 - e(f)$ if $f \in R$ and $e_R(f) = e(f)$ if $f \notin R$; $m_R(e_R) = m(e)$. For each of the above width measures $w \in \{tw, rw, tww\}$, we define a renaming variant w^* where $w^*(X) = \min_{R \subseteq F} w(X_R)$.

tw^* strictly dominates tw (domination follows by definition, to see that the domination is strict, construct a normalized MA instance $I = ((E, F, m), x, k, c)$, $|E| = |F| = n$ where $e(f) = 1 \neq x(f) = 0$ for all $e \in E \setminus \{x\}$ and $f \in F$; the incidence graph of I is a complete bipartite graph and thus $tw(I) = n$; however the incidence graph of I_F is edge-less and thus $tw^*(I) = 0$). *rw* and *tww*, however, are domination-equivalent with their renaming variants, since these two parameters are known to be closed under local complementation [Oum, 2005; Bonnet *et al.*, 2023]. We therefore have the relationship in terms of domination as indicated in Figure 2.

4 Hardness Results

Here, we show our hardness results for the MA problem. We start by giving a simple polynomial-time reduction from the well-known HITTING SET (HS) problem to the MA problem. In the HS problem one is given a family $\mathcal{S}$ of sets over some universe U and an integer h and one asks whether $\mathcal{S}$ has a *hitting set* of size at most h, i.e., a set $H \subseteq U$ (with $|H| \leq h$) such that $H \cap S \neq \emptyset$ for every $S \in \mathcal{S}$. For a given instance $I = (\mathcal{S}, U, h)$ of HS, we construct the instance $((E, F, m), x, k, c)$ of MA, denoted by $\mathsf{LA}(I)$, as follows. We set $c = 0$, $k = h$, $F = U$ and for every set $S \in \mathcal{S}$, E contains the example e_S with $m(e_S) = 0$ such that $e_S(f) = 1$ if $f \in S$ and $e_S(f) = 0$ otherwise for every $f \in F$. Finally, let x be the example with $x(f) = 0$ for every $f \in F$ and $m(x) = 1$; note that x is also part of E. Clearly, $\mathsf{LA}(I)$ can be computed in polynomial-time. The following lemma shows that $\mathsf{LA}(I)$ is indeed a reduction from HS to MA.

Lemma 4. *An instance $I = (\mathcal{S}, U, h)$ of* HS *has a solution if and only if the instance $\mathsf{LA}(I)$ of* MA *does.*

Proof. Towards showing the forward direction, let $H \subseteq U$ be a hitting set of size at most h for I. Then, H is an anchor for x, because for every example e_S for some $S \in \mathcal{S}$, it holds that $H \cap S \neq \emptyset$ and moreover e_S disagrees with x on all features in $H \cap S$. Because $|H| \leq h = k$ and $c = 0$, we obtain that H is a solution for $\mathsf{LA}(I)$.

Towards showing the reverse direction, let A be an anchor of x of size at most k. Then, A is a hitting set for I, because for every set $S \in \mathcal{S}$, the example e_S disagrees with x on some feature $f \in A$, which implies that $f \in S$ and therefore $A \cap S \neq \emptyset$. Because $|A| \leq k = h$, it follows that A is a solution for I. □

Because HS is W[2]-hard parameterized by solution size h and because the reduction $\mathsf{LA}(I)$ maps the parameter h to k, we obtain.

Corollary 5. MA *is* W[2]*-hard parameterized by k even if $c = 0$.*

We will show one more related hardness result for MA in Section 7, i.e., we will show that MA is NP-hard even if $\delta_{\max} \leq 2$, $c = 0$, and the incidence graph has bounded twin-width.

5 Algorithms Using $\delta_{\max}$ and k

In this section we show that MA is in XP parameterized by k alone and fpt parameterized by $\delta_{\max} + k$. The former algorithm is a simple brute-force algorithm that uses the fact that we can enumerate all potential anchors A in time $\mathcal{O}(|F|^k)$ and that we can then check whether A is indeed an anchor having the required coverage in time $\mathcal{O}(|E||A|) = \mathcal{O}(|E|k)$. This gives us the following theorem.

Theorem 6. MA *can be solved in time $\mathcal{O}(|F|^k|E|k)$ and is therefore in* XP *parameterized by k.*

The remainder of this section is devoted to a proof of the following theorem, which shows that MA can be solved efficiently for instances with small $\delta_{\max} + k$.

Theorem 7. MA *can be solved in time $\mathcal{O}(\delta_{\max}^k|I||E|k)$ and is therefore fpt parameterized by $\delta_{\max} + k$.*

The main idea behind the algorithm is to show that we can efficiently enumerate all (minimal) anchors of size at most k. Towards showing this, we use a simple reduction to the HS problem and the following well-known result.

Proposition 8 (Downey and Fellows 2013)**.** *Let $I = (\mathcal{S}, U, h)$ be an instance of* HS *with all sets in $\mathcal{S}$ having size at most γ. Then, there is an algorithm that enumerates all minimal hitting sets of size at most h for I in time $\mathcal{O}(\gamma^h|I|)$.*

Consider an instance $I = ((E, F, m), x, k, c)$ of MA. Then, a set A of features is an anchor of x if and only if for every example $e \in nE$, A contains a feature f with $e(f) \neq x(f)$. In other words, for every example $e \in nE$, an anchor A has to contain (hit) at least one feature from the set $X(e) = \{ f \mid e(f) \neq x(f) \}$. Therefore, A is an anchor for x if and only if A is a hitting set of $(\mathcal{S}(nE), F)$ with $\mathcal{S}(nE) = \{ X(e) \mid e \in nE \}$. Because the cardinality of every set in $\mathcal{S}(nE)$ is at most $\delta_{\max}$, we obtain from Proposition 8 that:

Lemma 9. *Let $I = ((E, F, m), x, k, c)$ be an instance of* MA*. Then, there is an algorithm that enumerates all minimal anchors of size at most k for I in time $\mathcal{O}(\delta_{\max}^k|I|)$.*

We are now ready to prove Theorem 7.

Proof of Theorem 7. Given an instance $I = ((E, F, m), x, k, c)$ of MA, we use Lemma 9 to enumerate all minimal anchors of I in time $\mathcal{O}(\delta_{\max}^k|I|)$. Then, for every such anchor A, we count the number $n(e)$ of examples $e \in pE$ that agree with x on all features in A in time $\mathcal{O}(|E|k)$. If $n(e) \geq c$ for one of those anchors, we return **yes**. Otherwise, we continue with the next anchor. Finally, if none of the anchors covers at least c examples, we return **no**. The total run-time of the algorithm is therefore $\mathcal{O}(\delta_{\max}^k|I||E|k)$ as stated. □

6 An Algorithm for Rank-width

In this section, we provide an fpt-algorithm for MA parameterized by the rank-width of the incidence graph of the input instance. Indeed, we will show something slightly stronger, namely, that given an instance $I = ((E, F, m), x, k, c)$ of MA we can find an anchor for x of minimum size that has coverage at least c or output correctly that no such anchor exists. Throughout this section, we will assume that I is normalized, which we are allowed to do due to Lemma 1 and Corollary 3. The following is the main result of this section.

Theorem 10. MA *is fixed-parameter tractable parameterized by the rank-width of $G_I(I)$.*

The remainder of this section is devoted to a proof of Theorem 10. At a first glance one might be tempted to think that Theorem 10 can be obtained by a simple application of Courcelle's theorem or more specifically by the known extensions of Courcelle's theorem for optimization problems [Arnborg *et al.*, 1991]. Unfortunately, on a second glance, this does not seem to be the case because the MA problem has two orthogonal optimization criteria, i.e., we want to minimize

the size of the anchor while at the same time maximizing the coverage of the anchor. We therefore have to resort to provide a dedicated algorithm that is based on a dynamic programming algorithm. along a given rank decomposition; indeed for simplicity we will provide the algorithm on a NLC-decomposition, which is known to be asymptotically equivalent to rank-width [Wanke, 1994]. Before we start with an informal description of the algorithm in Subsection 6.3, let us first fix the main notation required to describe the algorithm.

6.1 Graphs and NLC-width

We will assume that the reader is familiar with basic graph theory (see e.g., [Bang-Jensen and Gutin, 2009; Diestel, 2017]). We consider (vertex and edge labelled) undirected graphs. Let $G = (V, E)$ be a graph. We write $V(G) = V$ and $E(G) = E$ for the sets of vertices and edges of G, respectively. We denote an edge between $u \in V$ and $v \in V$ by $\{u, v\}$. For a set $V' \subseteq V$ of vertices, we let $G[V']$ denote the graph induced by the vertices in V', i.e. $G[V']$ has vertex set V' and edge set $E \cap \{\, \{u, v\} \mid u, v \in V' \,\}$ and we let $G - V'$ denote the graph $G[V \setminus V']$. For a set $E' \subseteq E$ of edges we let $G - E'$ denote the graph with vertex set V and edge set $E \setminus E'$.

A *ω-graph* is a pair (G, λ), where $G = (V, E)$ is a graph and $\lambda : V \to [\omega]$ is a *vertex label mapping* that labels every vertex $v \in V$ with a label $\lambda(v)$ from $[\omega] = \{1, \ldots, \omega\}$. An *initial ω-graph* is a ω-graph consisting of exactly one vertex v (say, with label i) which is denoted by $i(v)$.

Node label control-width (*NLC-width*) is a graph parameter defined as follows [Wanke, 1994]: let $\omega \in \mathbb{N}$ be a positive integer. A *ω-NLC-expression tree* of a graph $G = (V, E)$ is a subcubic tree B, where every node b of B is associated with a ω-graph (denoted by (G_b, λ_b)), such that:

1. Every leaf represents an initial ω-graph $i(v)$ with $i \in [\omega]$ and $v \in V$.
2. Every non-leaf node b with one child c is a *relabelling node* and has an associated relabelling function $\gamma_b : [\omega] \to [\omega]$. Moreover, (G_b, λ_b) is obtained from (G_c, λ_c) by relabelling all vertices of G_c with label i to label $\gamma_b(i)$ for every $i \in [\omega]$.
3. Every non-leaf node b with two children, i.e. a left child l and a right child r, is a *join node* and has an associated *join matrix*, i.e. a binary $\omega \times \omega$ matrix M_b. Moreover, (G_b, λ_b) is obtained from the disjoint union of (G_l, λ_l) and (G_r, λ_r) by adding an edge from all vertices labeled i in G_l to all vertices labelled j in G_r whenever $M_b[i, j] = 1$.
4. G is equal to the G_r for the root node r of B.

The NLC-width of a graph G, denoted by $\mathit{nlcw}(G)$, is the minimum ω for which G has a ω-NLC-expression tree. A ω-NLC-expression tree is *nice* if every relabelling node b has a relabelling function $\gamma_b : [\omega] \to [\omega]$ such that for some $i, j \in [\omega]$, $\gamma_b(i) = j$ and $\gamma_b(c) = c$ for all $c \in [\omega] \setminus \{i\}$. Let b be a node in a ω-NLC-expression tree of a graph G. We let V_b denote the set of vertices of G_b. By the definition of a ω-NLC-expression tree, if $u, v \in V_b$ have the same label in (G_b, λ_b) and $w \in V(G) \setminus V_b$, then u is adjacent to w in G if and only if v is. Computing the NLC-width of a graph is NP-hard [Gurski and Wanke, 2005], however, we have the following.

Proposition 11. *Let G be a graph and ω be an integer. There is an fpt-algorithm (w.r.t. ω) which either correctly concludes that G has NLC-width larger than ω or outputs a nice 2^ω-NLC-expression tree for G with $\mathcal{O}(2^\omega n)$ nodes.*

6.2 Main Notation

To simplify the description of the proof, we will show the result for NLC-width, which is known to be asymptotically equivalent to rank-width [Wanke, 1994]. Because of Proposition 11, it suffices to show the result for the case when we are provided with a nice ω-NLC-expression tree.

Lemma 12. *Let $I = ((E, F, m), x, k, c)$ be an instance of* MA *and let B be a nice ω-NLC-expression tree for $G_I(I)$. Then, finding an anchor of smallest size, whose coverage is at least c or deciding correctly that no such anchor exists, is fixed-parameter tractable parameterized by ω.*

The remainder of this section is devoted to a proof of Lemma 12, which implies Theorem 10. Let $I = ((E, F, m), x, k, c)$ be a normalized instance of MA. Let G be the graph $G_I(I)$ after removing the isolated example x. Recall that we denote by pE and nE the subsets of E containing all examples $e \in E$ such that $m(e) = m(x)$ and $m(e) \neq m(x)$, respectively, and that we refer to the examples in pE and nE as *positive* and *negative* examples, respectively.

Let B be a ω-NLC-expression tree for G and let b be a node of B. Recall that (G_b, λ_b) is the ω-graph associated with node b and that V_b is the set of vertices of G_b. We denote by I_b the MA instance associated with b, i.e., I_b contains all features and examples in V_b and the feature values of the examples in I_b are given by the edge relation of G_b, i.e., $e(f) = 1$ if G_b has an edge between e and f and otherwise $e(f) = 0$. We denote by $\lambda_b^{-1}(c)$ the set of features and examples v in V_b with $\lambda_b(v) = c$ for every $c \in [\omega]$. For convenience, we denote by $\mathit{feat}(b)$ and $\mathit{exam}(b)$ the set of features and examples in V_b respectively, i.e., $\mathit{feat}(b) = F \cap V_b$ and $\mathit{exam}(b) = E \cap V_b$. Moreover, we denote by $\mathit{pexam}(b)$ and $\mathit{nexam}(b)$ the set of all positive and negative examples in $\mathit{exam}(b)$, respectively, i.e., $\mathit{pexam}(b) = \mathit{exam}(b) \cap pE$ and $\mathit{nexam}(b) = \mathit{exam}(b) \cap nE$. With a slight abuse of notation, we will in the following assume that the feature values for any example in $\mathit{exam}(b)$ are defined with respect to the instance I_b; if not explicitly stated otherwise.

6.3 Overview and Informal Description of the Algorithm

As is usual for algorithms exploiting NLC-width, the algorithm uses a dynamic programming approach to compute a set of valid records for every node b of the decomposition B in a leaf-to-root manner. The crucial part of the algorithm is the correct definition of the records. Informally, a (valid) record for a node b of B represents a compact representation of an equivalence class of solutions (anchors) for the whole instance from the perspective of the subinstance I_b. For the purpose of the DP algorithm, it is best to think of a solution for $I = ((E, F, m), x, k, c)$ in terms of a pair (A, P), where

$A \subseteq F$ with $|A| \leq k$ is an anchor for x and $P \subseteq \textit{pexam}$ with $|P| = c$ is a set of positive examples that are not distinguished from x by any feature in A and that witnesses that A has coverage at least c. Then, the first question is what information do we need to store for such a solution (A, P), when viewed from the perspective of the subinstance I_b? Consider the part (A_b, P_b) of the solution (A, P) that lives inside I_b, i.e., $A_b = A \cap \textit{feat}(b)$ and $P_b = P \cap \textit{pexam}(b)$. Then, (A_b, P_b) is not necessarily a solution for the subinstance I_b, because A_b is not necessarily an anchor for x in I_b; this is because there can be negative examples $e \in \textit{nexam}(b)$ that are not yet distinguished from x by A_b (but are only latter distinguished from x using some feature in $A \setminus A_b$). To take this into account, one also needs to consider the set $N_b \subseteq \textit{nexam}(b)$ of negative examples in I_b that are not yet distinguished from x by A_b in I_b. Therefore, the triple (A_b, P_b, N_b) now provides a complete representation of the solution (A, P) from the perspective of I_b and we will therefore refer to it as a partial solution. Unfortunately, it is not possible to store every such triple for b, because the number of those triples is too large (the size is at least $\Omega(|I_b|^{k+c})$). However, because of the properties of an NLC-decomposition and in particular the fact that every feature (or example) of I_b with the same label will behave the same w.r.t. every example (or feature) that is not yet in I_b, it is not necessary to remember the exact sets of features and examples. In particular, instead of storing the partial solution (A_b, P_b, N_b), it is sufficient to store the tuple (C, C_-, C_+, p, s), where:

- $C \subseteq [\omega]$ is the set of all labels in G_b that contain a feature in A_b,
- $C_- \subseteq [\omega]$ is the set of all labels in G_b that contain a negative example in N_b,
- $C_+ \subseteq [\omega]$ is the set of all labels in G_b that contain a positive example in P_b,
- p and s are integers with $p = |P_b|$ and $s = |A_b|$.

We will later refer to the tuple (C, C_-, C_+, p, s) as a record and we will refer to a partial solution (A_b, P_b, N_b) that is represented by the record as a witness. Clearly, we are only interested in records that represent some partial solution and we will refer to such records as semi-valid. Note that a record can be seen as a compact representation of an equivalence class of partial solutions since many partial solutions can result in the same record. However, since all partial solutions that result in the same record behave in the same manner due to the properties of an NLC-decomposition it is sufficient to represent all those partial solutions by a record. More importantly, if we have two semi-valid records that only differ in the size s of the (partial) anchor, then it is sufficient to remember only the tuple with the smaller (partial) anchor. This now provides us with the notion of a valid record, i.e., a record (C, C_-, C_+, p, s) is valid for a node b if s is equal to the minimum number s' such that (C, C_-, C_+, p, s') is semi-valid.

6.4 Formal Definition of Records and Preliminary Results

We are now ready to define the records and their semantics. A *record* for b is a tuple (C, C_-, C_+, p, s) where C, C_- and C_+ are set of labels (subsets of $[\omega]$) and p and s are integers with $p \leq c$. We say that a record (C, C_-, C_+, p, s) is *semi-valid* for b if there is a set of features $A \subseteq \textit{feat}(b)$ of size s and a set of positive examples $P \subseteq \textit{pexam}(b)$ of size p such that:

(a) C is the set of all labels $c \in [\omega]$ such that $\lambda_b^{-1}(c) \cap A \neq \emptyset$.

(b) C_- is the set of all labels $c \in [\omega]$ such that $\lambda_b^{-1}(c) \cap N \neq \emptyset$. Here, and in the following, N is the set of all negative examples in $\textit{nexam}(b)$ that are not yet distinguished from x by a feature in A w.r.t. I_b, i.e., $N = \{ e \in \textit{nexam}(b) \mid e(f) = x(f) \text{ for every feature } f \in A \}$; recall the the feature values of e are defined with respect to I_b.

(c) C_+ is the set of all labels $c \in [\omega]$ such that $\lambda_b^{-1}(c) \cap P \neq \emptyset$.

(d) the examples in P agree with x on all features in A with respect to I_b.

Let $R = (C, C_-, C_+, p, s)$ be a semi-valid record for node b. A witness for R is a pair (A, P), where $A \subseteq feat(b)$ is a set of features of size s and $P \subseteq \textit{pexam}(b)$ is a set of positive examples of size p that satisfy all the conditions **(a)–(d)**.

We say that a record (C, C_-, C_+, p, s) is *valid* if s is equal to the minimum number s' such that (C, C_-, C_+, p, s') is semi-valid. We denote by $\mathcal{R}(b)$ the set of all valid records for the node b. The following corollary follows immediately from the definition of valid records and gives an upper bound on the size of $\mathcal{R}(b)$.

Observation 13. $|\mathcal{R}(b)| \leq 2^{3\omega}c$.

6.5 Putting It All Together

We will now show that we can compute $\mathcal{R}(b)$ for each of the three node types of a nice ω-NLC expression tree provided that $\mathcal{R}(c)$ has already been computed for every child c of b. Recall that b is either a leaf node associated with a ω-graph $i(v)$, a relabelling node with one child and with relabelling function γ_b, or a join node with a left child, a right child and a join matrix M_b. Moreover, recall that (G_b, λ_b) is the ω-graph associated with b (whose unlabelled version is a subgraph of G) and V_b is the set of vertices of G_b.

Lemma 14 (leaf node)**.** *Let $b \in V(B)$ be a leaf node. Then $\mathcal{R}(b)$ can be computed in $\mathcal{O}(1)$ time.*

Proof. Let $i(v)$ be the initial ω-graph associated with b. We distinguish the following cases. If v is a feature, then $\mathcal{R}(b)$ contains the following two records: The record $(\emptyset, \emptyset, \emptyset, 0, 0)$ which is witnessed by the set $A = \emptyset$ of features and the set $P = \emptyset$ of positive examples and the record $(\{i\}, \emptyset, \emptyset, 0, 1)$, which is witnessed by the sets $A = \{v\}$ and $P = \emptyset$.

If v is a negative example, then $\mathcal{R}(b)$ contains the record $(\emptyset, \{i\}, \emptyset, 0, 0)$, which is witnessed by the sets $A = \emptyset$ and $P = \emptyset$.

If v is a positive example, then $\mathcal{R}(b)$ contains the two records $(\emptyset, \emptyset, \{i\}, 1, 0)$, which is witnessed by the sets $A = \emptyset$ and $P = \{v\}$, and $(\emptyset, \emptyset, \emptyset, 0, 0)$, which is witnessed by the sets $A = \emptyset$ and $P = \emptyset$.

Clearly, $\mathcal{R}(b)$ can be computed in constant time. Moreover, the correctness of the definition of $\mathcal{R}(b)$ follows directly from the definition and the given witnesses. □

Lemma 15 (join node)**.** *Let $b \in V(B)$ be a join node. Then $\mathcal{R}(b)$ can be computed in $\mathcal{O}((2^{3\omega}c)^2)$ time.*

Proof Sketch. Let b^l and b^r be the left and right child of b in B, respectively. Let M_b be the join matrix for the node b, i.e. M_b is a $\omega \times \omega$ binary matrix.

Let $R^l = (C^l, C^l_-, C^l_+, p^l, s^l) \in \mathcal{R}(b^l)$ and $R^r = (C^r, C^r_-, C^r_+, p^r, s^r) \in \mathcal{R}(b^r)$ be two records. Let Q^l be the set of all labels $c \in [\omega]$ such that there is a label $c' \in C^r$ with $M_b[c, c'] = 1$, i.e., informally Q^l is the set of all labels occurring in G_{b^l} that are connected to at least one label in C^r. Note that if e is an example in I_{b^l} whose label is in Q^l, then e is distinguished from x in the instance I_b by any feature with label $c' \in C^r$. We say that R^l is *compatible* with R^r if they satisfy:

(*) $Q^l \cap C^l_+ = \emptyset$ and $Q^r \cap C^r_+ = \emptyset$.

Intuitively, R^l and R^r are compatible if all positive examples in C^l_+ and C^r_+ are not distinguished from x in the instance I_b; assuming that those examples were not distinguished from x already in the instance I_{b^l} and I_{b^r}, respectively.

We are now ready to define $\mathcal{R}(b)$. A record $R = (C, C_-, C_+, p, s)$ is valid for b, that is, belongs to $\mathcal{R}(b)$ if and only if there is a pair of compatible records $R^l = (C^l, C^l_-, C^l_+, p^l, s^l) \in \mathcal{R}(b^l)$ and $R^r = (C^r, C^r_-, C^r_+, p^r, s^r) \in \mathcal{R}(b^r)$ such that:

(i) $C = C^l \cup C^r$;

(ii) $C_- = (C^l_- \setminus Q^l) \cup (C^r_- \setminus Q^r)$;

(iii) $C_+ = C^l_+ \cup C^r_+$;

(iv) $p = p^l + p^r$ and $s = s^l + s^r$.

We now show how and in what time $\mathcal{R}(b)$ can be computed. We compute $\mathcal{R}(b)$ as follows. First we compute Q^l and Q^r in time $\mathcal{O}(\omega^2)$. Then, for every pair of records $R^l \in \mathcal{R}(b^l)$ and $R^r \in \mathcal{R}(b^r)$, we do the following. First we check whether both records are compatible in time $\mathcal{O}(\omega)$. If not, we continue with the next pair. Otherwise, we compute R from R^l and R^r in time $\mathcal{O}(\omega + \log c + \log k)$ and add it to $\mathcal{R}(b)$. Therefore, the total run-time for computing $\mathcal{R}(b)$ is at most $\mathcal{O}(\omega^2 + |\mathcal{R}(b^l)||\mathcal{R}(b^r)|(\omega + \log c + \log k))$, which because of Observation 13 is at most $\mathcal{O}((2^{3\omega}c)^2)$. The correctness-proof of the definition of $\mathcal{R}(b)$ can be found in the supplementary material. □

Lemma 16 (relabel node)**.** *Let $b \in V(B)$ be relabelling node in B. Then $\mathcal{R}(b)$ can be computed in $\mathcal{O}(2^{3\omega}c\omega)$ time.*

We are now ready to prove the main result of this section.

Proof of Lemma 12. Given $I = ((E, F, m), x, k, c)$ and B, we use Lemmas 14, 15 and 16 to compute the set $\mathcal{R}(b)$ of valid records for every node b of B in a leaf-to-root manner. We then go through all records in $\mathcal{R}(r)$ for the root r of B to find a record R such that s is minimum among all records of the form $(C, \emptyset, C_+, c, s)$. If no such record R exists or the size s for R is larger than k, we correctly output **no**. Otherwise, we use standard dynamic programming methods to obtain a witness (A, P) for R and output A. The correctness of the algorithm follows directly from the definition of the valid records together with Lemmas 14, 15 and 16. Moreover, we obtain the run-time of the algorithm as follows. The maximum time spent on any of the nodes b (which is achieved for a join-node) is $\mathcal{O}((2^{3\omega}c)^2)$. Because B has at most $\mathcal{O}(\omega|E \cup F|)$ (see Proposition 11) nodes, we obtain $\mathcal{O}((2^{3\omega}c)^2\omega|E \cup F|)$ as the total run-time of the algorithm, which shows that MA is fixed-parameter tractable parameterized by the NLC-width of $G_I(I)$. □

7 Twin-Width

In this section, we provide our complexity results for twin-width. In particular, we show that the fpt-result of the previous section for rank-width cannot be extended to the more general (less restrictive) parameter twin-width; indeed we show that MA is paraNP-hard parameterized by twin-width alone. However, we also show that MA becomes fixed-parameter tractable parameterized by twin-width plus k (given that a witness for small twin-width is provided), which given the generality of twin-width is an encouraging result. We start with our hardness result.

Theorem 17. MA *is* paraNP*-hard parameterized by the twin-width of its incidence graph even if $c = 0$ and $\delta_{\max}(I) = 2$.*

We are now ready to show that MA is fixed-parameter tractable parameterized by twin-width plus k. While it is relatively straightforward to show that MA is fixed-parameter tractable when additionally parameterized by c (this follows immediately from the recently shown first order logic meta-theorem for twin-width [Bonnet *et al.*, 2022b]), this is no longer the case if one merely parametrizes by *tww* $+ k$. We therefore developed a tailor-made dynamic programming algorithm, which we believe to be interesting in its own right; an alternative proof can be obtained using a very recently established meta-theorem [Bergougnoux *et al.*, 2023].

Theorem 18. MA *is fpt parameterized by the twin-width of $G_I(I)$ plus k provided that one is given an optimal contraction sequence of the incidence graph.*

8 Conclusion

We have provided the first complexity analysis of finding concise local explanations (MA), a central computational problem in explainable AI. Starting from an NP-hardness result of the problem in general, we have studied it in a parameterized complexity setting with several natural parameters. Our results drawn a comprehensive complexity landscape for all combinations of the considered parameters. Particularly interesting is our finding that adding the coverage size c to any of the considered parameterizations does not change the complexity of the MA problem. On a more technical level and of independent interest is that our fpt-results for rank-width and twin-width can both not be obtained via the corresponding meta-theorems but require a tailor-suited dynamic programming approach. We hope that our study stimulates further research on the computational complexity of problems arising in explainable AI.

Acknowledgements

Stefan Szeider acknowledges support by the Austrian Science Fund (FWF, project P32441) and the Vienna Science and Technology Fund (WWTF, project ICT19-065). Giacomo Paesani and Sebastian Ordyniak acknowledge support from the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1).

References

[Angelino *et al.*, 2017] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo I. Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, 18(234):1–78, 2017.

[Arnborg *et al.*, 1991] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[Bang-Jensen and Gutin, 2009] Jørgen Bang-Jensen and Gregory Gutin. *Digraphs*. Springer Monographs in Mathematics. Springer-Verlag London Ltd., London, second edition, 2009.

[Barceló *et al.*, 2020] Pablo Barceló, Mikaël Monet, Jorge Pérez, and Bernardo Subercaseaux. Model interpretability through the lens of computational complexity. *Proc. NeurIPS 2020*, 2020.

[Bergougnoux *et al.*, 2023] Benjamin Bergougnoux, Jan Dreier, and Lars Jaffke. A logic-based algorithmic metatheorem for mim-width. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3282–3304. SIAM, 2023.

[Bodlaender, 2007] Hans L. Bodlaender. Treewidth: Structure and algorithms. *Proc. SIROCCO 2007*, 4474:11–25, 2007.

[Bonnet *et al.*, 2021a] Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width II: small classes. *Proc. SODA 2021*, pages 1977–1996, 2021.

[Bonnet *et al.*, 2021b] Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width III: max independent set, min dominating set, and coloring. *Proc. ICALP 2021*, 198(35):1–20, 2021.

[Bonnet *et al.*, 2022a] Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Torunczyk. Twin-width IV: ordered graphs and matrices. *Proc. STOC 2022*, pages 924–937, 2022.

[Bonnet *et al.*, 2022b] Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. *Journal of the ACM*, 69(1):1–46, 2022.

[Bonnet *et al.*, 2023] Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, and Stéphan Thomassé. Twin-width V: linear minors, modular counting, and matrix multiplication. *Proc. STACS 2023*, 254(15):1–16, 2023.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Diestel, 2017] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer Verlag, New York, 5th edition, 2017.

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

[Ganian *et al.*, 2018] Robert Ganian, Iyad Kanj, Sebastian Ordyniak, and Stefan Szeider. Parameterized algorithms for the matrix completion problem. *Proc. ICML 2018*, pages 1642–1651, 2018.

[Gurski and Wanke, 2005] Frank Gurski and Egon Wanke. Minimizing NLC-width is NP-complete. *Graph-theoretic concepts in computer science*, 3787:69–80, 2005.

[Hu *et al.*, 2019] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Proc. NeurIPS 2019*, pages 7265–7273, 2019.

[Ignatiev *et al.*, 2021] Alexey Ignatiev, João Marques-Silva, Nina Narodytska, and Peter J. Stuckey. Reasoning-based learning of interpretable ML models. *Proc. IJCAI 2021*, pages 4458–4465, 2021.

[Kobourov *et al.*, 2022] Stephen G. Kobourov, Maarten Löffler, Fabrizio Montecchiani, Marcin Pilipczuk, Ignaz Rutter, Raimund Seidel, Manuel Sorge, and Jules Wulms. The influence of dimensions on the complexity of computing decision trees. *CoRR*, 2022.

[Lewis, 1978] Harry R. Lewis. Renaming a set of clauses as a Horn set. *Journal of the ACM*, 25(1):134–135, January 1978.

[Lundberg and Lee, 2017] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Proc. NIPS 2017*, pages 4765–4774, 2017.

[Ordyniak and Szeider, 2021] Sebastian Ordyniak and Stefan Szeider. Parameterized complexity of small decision tree learning. *Proc. AAAI 2021*, pages 6454–6462, 2021.

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

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

[Ribeiro *et al.*, 2016] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. *Proc. KDD 2016*, pages 1135–1144, 2016.

[Ribeiro *et al.*, 2018] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. *Proc. AAAI 2018*, pages 1527–1535, 2018.

[Rudin, 2019] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[Shrikumar *et al.*, 2017] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *Proc. ICML 2017*, 70:3145–3153, 2017.

[Shrotri *et al.*, 2022] Aditya A. Shrotri, Nina Narodytska, Alexey Ignatiev, Kuldeep S. Meel, João Marques-Silva, and Moshe Y. Vardi. Constraint-driven explanations for black-box ML models. *Proc. AAAI 2022*, pages 8304–8314, 2022.

[Simonov *et al.*, 2019] Kirill Simonov, Fedor V. Fomin, Petr A. Golovach, and Fahad Panolan. Refined complexity of PCA with outliers. *Proc. ICML 2019*, 97:5818–5826, 2019.

[Wanke, 1994] Egon Wanke. k-NLC graphs and polynomial algorithms. *Discrete Applied Mathematics*, 54(2):251–266, 1994.