Exact Algorithms for Distance to Unique Vertex Cover

Foivos Fioravantes[1], **Dušan Knop**[1], **Nikolaos Melissinos**[2], **Michal Opler**[1], **Manolis Vasilakis**[3]
[1]Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic
[2]Computer Science Institue, Faculty of Mathematics and Physics, Charles University, Prague, Chezch Republic
[3]Université Paris-Dauphine, PSL University, CNRS UMR7243, LAMSADE, Paris, France
foivos.fioravantes@fit.cvut.cz, Dusan.Knop@fit.cvut.cz, melissinos@iuuk.mff.cuni.cz, michal.opler@fit.cvut.cz, emmanouil.vasilakis@dauphine.eu

Abstract

In their AAAI 2024 paper, Horiyama et al. studied the problem of generating graph instances that possess a unique minimum vertex cover under specific conditions. Their approach involved pre-assigning certain vertices to be part of the solution or excluding them from it. Notably, for the VERTEX COVER problem, pre-assigning a vertex is equivalent to removing it from the graph. Horiyama et al. focused on maintaining the size of the minimum vertex cover after these modifications. In this work, we extend their study by relaxing this constraint: our goal is to ensure a unique minimum vertex cover, even if the removal of a vertex may not incur a decrease on the size of said cover.

Surprisingly, our relaxation introduces significant theoretical challenges. We observe that the problem is Σ_P^2-complete, and remains so even for planar graphs of maximum degree 5. Nevertheless, we provide a linear time algorithm for trees, which is then further leveraged to show that MU-VC is in FPT when parameterized by the combination of treewidth and maximum degree. Finally, we show that MU-VC is in XP when parameterized by clique-width while it is fixed-parameter tractable (FPT) if we add the size of the solution as part of the parameter.

Introduction

Addressing NP-hard problems has long been a central challenge in computer science, driving advancements in algorithmic design and computational theory. These problems, being inherently computationally intractable, have inspired diverse approaches to developing efficient and scalable solutions. Algorithmic strategies for NP-hard problems are critical to theoretical research and find applications in areas such as network optimization, scheduling, and data analysis.

A fundamental aspect of algorithmic research is the construction of robust benchmark datasets. These datasets serve for evaluating and comparing the effectiveness of different algorithms. By providing a standardized testing ground, traditional benchmarks such as TSPLIB (Reinelt 1991), UCI Machine Learning Repository (Asuncion, Newman et al. 2007), SATLIB (Hoos and Stützle 2000), MIPLIB (Koch et al. 2011), LIBSVM (Chang and Lin 2011), and NetworkX graph datasets (Hagberg, Swart, and Schult 2008) have laid the groundwork for progress in this field.

More recently, modern datasets tailored to AI-driven algorithm design have emerged. Datasets like NPHardEval (Fan et al. 2023, 2024) provide a dynamic benchmark for assessing the reasoning capabilities of large language models through algorithmic questions, including NP-hard problems. GraphArena (Tang et al. 2024) offers a comprehensive suite of real-world graph-based tasks, enabling the evaluation of AI algorithms in diverse computational challenges, from social networks to molecular structures. Meanwhile, MaxCut-Bench (Nath and Kuhnle 2024) provides an open-source platform to benchmark heuristics and methods based on machine learning specifically for the MAXIMUM CUT problem. These benchmarks not only expand the scope of algorithmic research, but also bridge traditional approaches with cutting-edge AI innovations.

Recently, the PRE-ASSIGNMENT FOR UNIFICATION OF MINIMUM VERTEX COVER (PAU-VC) problem was introduced in (Horiyama et al. 2024), and further studied in (An et al. 2025). This was motivated by the need to create challenging datasets for algorithmic evaluation. Intuitively, ensuring a solution is unique adds significant complexity, as solvers have no margin for error in identifying the correct solution. A set S of vertices in a graph G is called a *vertex cover* if S intersects all edges of G. The objective of MINIMUM VERTEX COVER is to compute a vertex cover S of G with the smallest cardinality possible. The UNIQUE VERTEX COVER problem extends this by ensuring S is unique. This guarantee imposes additional constraints, making the problem particularly challenging from both theoretical and practical perspectives.

It is important to note that, in the context of vertex cover, selecting a vertex for inclusion in the cover is equivalent to deleting it from the graph along with all its incident edges. However, ensuring that a set of vertices belongs to the deletion set can inadvertently increase the size of the minimum vertex cover under this constraint, introducing further complexity. This leads to the definition of the MODULATOR TO UNIQUE MINIMUM VERTEX COVER (MU-VC for short) problem (refer to Figure 1 for an illustration of the difference in the nature of the two problems): given a graph $G = (V, E)$ and an integer k, find a set $S \subseteq V$ such that $G - S$ has a unique minimum vertex cover and $|S| \leq k$.

Unlike PAU-VC, where the solver must adapt to preselected vertices to enforce uniqueness, MU-VC simplifies

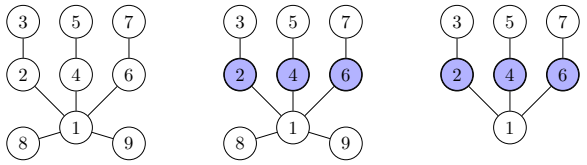


Figure 1: Left: a graph G with minimum vertex cover of size 4. There are many such – vertex 1 and then any set of vertices of size 3 that intersects the pairs $\{2,3\}$, $\{4,5\}$, and $\{6,7\}$. Middle: PAU-VC solution of size 3 (the only vertex missing at this point is 1). Right: MU-VC solution (deleted) $8,9$, and the minimum size of a vertex cover is now 3 (i.e., 5 in total); namely, $2,4,6$ which is now unique.

this process by reverting to solving the MINIMUM VERTEX COVER problem on $G - S$. This distinction makes MU-VC particularly appealing, as it maintains the standard problem formulation while achieving uniqueness. Although this property holds for VERTEX COVER, it does not necessarily extend to other problems, such as DOMINATING SET.

Our Contribution. It is easy to see that MU-VC is NP-hard, as it generalizes the UNIQUE OPTIMAL VERTEX COVER problem (for $k = 0$), which is at least as hard as UNIQUE SAT which is NP-hard (Hudry and Lobstein 2019). First, we precisely determine its complexity and show that MU-VC is actually Σ_2^P-complete, as is the case for PAU-VC (Horiyama et al. 2024); in fact, in Theorem 6 we show that both problems remain so even for very restricted graph classes, namely planar graphs of maximum degree 5 (which also improves the corresponding result in (Horiyama et al. 2024)). Motivated by these negative results we proceed to examine MU-VC when the input graph G has an even simpler structure, that of a tree. We initially present a polynomial-time algorithm for this case , which is then improved into an (optimal) linear time algorithm in Theorem 1. We then tackle the problem through the parameterized complexity point of view. Given that the natural parameterization by k cannot yield any positive results (as evidenced by the case $k = 0$), we proceed by taking into account the structure of G. We first consider the parameterization by treewidth, the most well-studied structural parameter. Building upon the algorithm of Theorem 1, in Theorem 2 we employ DP to construct an XP algorithm with running time $n^{O(2^w)}$, where w is the treewidth of G, which is then improved into an FPT algorithm in Theorem 3 when parameterized also by the maximum degree of G. Next, we consider the parameterization by clique-width and, in Theorem 4, we develop a DP algorithm showing that the problem belongs to XP and is solvable in $n^{O(2^d)}$ time, where d denotes the clique-width of G. Additionally parameterizing by the natural parameter k lifts the problem to FPT, and in Theorem 5 we develop such an algorithm of running time $k^{O(2^d)} \cdot n$.

Preliminaries

Throughout the paper we use standard graph notation (Diestel 2025). All graphs considered are simple, undirected without loops. Given a graph G and a subset of its vertices $S \subseteq V(G)$, $G[S]$ denotes the subgraph induced by S, while $G - S$ denotes $G[V(G) \setminus S]$. For $x, y \in \mathbb{Z}$, let $[x, y] = \{z \in \mathbb{Z} \mid x \leq z \leq y\}$, while $[x] = [1, x]$. Finally, a k-labeled graph G is a triple (V, E, lab_G) where $\mathrm{lab}_G \colon V \to [k]$. Let us now formally define the two problems considered in this paper.

PRE-ASSIGNMENT FOR UNIFICATION OF MINIMUM VERTEX COVER (PAU-VC)
Input: Graph $G = (V, E)$, integer k.
Task: Find a set $S \subseteq V$ of size $|S| \leq k$ s.t. there exists a unique minimum vertex cover of G that contains S.

MODULATOR TO UNIQUE MINIMUM VERTEX COVER (MU-VC)
Input: Graph $G = (V, E)$, integer k.
Task: Find a set $S \subseteq V$, s.t. $G - S$ has a unique minimum vertex cover and $|S| \leq k$.

Finally, recall that a decision problem is in Σ_2^P if and only if it can be decided by a non-deterministic Turing machine with the added use of an NP-oracle.

Parameterized Complexity. The toolkit of parameterized complexity allows us to circumvent many of the limitations of classical measures of time (and space) complexity. This is achieved by considering additional measures that can affect the running time of an algorithm; these additional measures are exactly what we refer to as *parameters*. The goal here is to construct exact algorithms that run in time $f(k) \cdot \mathrm{poly}(n)$, where f is a computable function, n is the size of the input and k is the parameter; such algorithms are referred to as *fixed-parameter tractable* (FPT). A problem admitting such an algorithm is said to belong in FPT. Failing to achieve such an algorithm for a problem, we can instead try to show that it is *slicewise polynomial*, i.e., that it can be determined in $n^{f(k)}$ time. Such a problem then belongs to the class XP. We refer the interested reader to now classical monographs (Cygan et al. 2015; Downey and Fellows 2012; Flum and Grohe 2006; Niedermeier 2006) for a more comprehensive introduction to this topic.

Structural Parameters. A *tree-decomposition* of G is a pair $(T, \{B_x \mid x \in V(T)\})$, where T is a tree rooted at a node $r \in V(T)$, each node x in T is assigned a *bag* B_x, and the following conditions hold:

- for every edge $\{u, v\} \in E(G)$ there is a node $x \in V(T)$ such that $u, v \in B_x$, and
- for every vertex $v \in V$, the set of nodes x with $v \in B_x$ induces a connected subtree of T.

The *width* of a tree-decomposition $(T, \{B_x \mid x \in V(T)\})$ is $\max_{x \in V(T)} |B_x| - 1$, and the treewidth $\mathrm{tw}(G)$ of a graph G is the minimum width of a tree-decomposition of G. It is known that computing a tree-decomposition of minimum width is in FPT when parameterized by the treewidth (Bodlaender 1996; Kloks 1994), and even more

efficient algorithms exist for obtaining near-optimal tree-decompositions (Korhonen and Lokshtanov 2023).

A tree-decomposition $(T, \{B_x \mid x \in V(T)\})$ is *nice* if every node x in T is exactly of one of the following four types: (i) Leaf: x is a leaf of T and $|B_x| = 0$. (ii) Introduce: x has a unique child y and there exists $v \in V$ such that $B_x = B_y \cup \{v\}$. (iii) Forget: x has a unique child y and there exists $v \in V$ such that $B_y = B_x \cup \{v\}$. (iv) Join: x has exactly two children y, z and $B_x = B_y = B_z$. Every graph $G = (V, E)$ admits a nice tree-decomposition of width $\mathrm{tw}(G)$ (Bodlaender 1998), which is computed in linear time for graphs of bounded treewidth.

The *clique-width* of a graph G is an important parameter generalizing the treewidth of G (Courcelle and Olariu 2000). A graph of clique-width d can be constructed through a sequence of the following operations on vertices that are labeled with at most d different labels. We can use (1) introducing a single vertex v of an arbitrary label i, denoted $i(v)$, (2) disjoint union of two labeled graphs, denoted $H_1 \oplus H_2$, (3) introducing edges between *all* pairs of vertices of two distinct labels i and j in a labeled graph H, denoted $\eta_{i,j}(H)$, and (4) changing the label of *all* vertices of a given label i in a labeled graph H to a different label j (i.e., collapsing the pair of labels i and j), denoted $\rho_{i\to j}(H)$. An expression describes a graph G if G is the final graph given by the expression (after we remove all the labels). The *width* of an expression is the number of different labels it uses. The clique-width of a graph is the minimum width of an expression describing it.

Trees

Theorem 1 *The* MU-VC *problem can be solved in linear time $O(n)$ on trees.*

We are working with *rooted trees* (T, r) such that T is a tree and $r \in V(T)$. Similarly, a *rooted forest* is just a pair (F, r) such that F is a forest and r is its one designated vertex. Let us define a way of representing rooted trees as algebraic terms similar to nice tree decompositions that allows us to moreover keep only one special vertex (the root) in our "bag". First, we denote by $\mathsf{Leaf}(r)$ the singleton rooted tree (T, r), i.e., $V(T) = \{r\}$ and $E(T) = \emptyset$. For a rooted tree (T', r') and $r \notin V(T)$, let $\mathsf{Extend}((T', r'), r)$ be the rooted tree (T, r) obtained from T' by adding r as the new root and joining it to r' via an edge, i.e., $V(T) = V(T') \cup \{r\}$ and $E(T) = E(T') \cup \{\{r', r\}\}$.

We remark that in terms of nice tree decompositions, the operation Extend corresponds to introducing the new root r and immediately afterwards forgetting the old root r'. For two rooted trees (T_1, r) and (T_2, r) such that $V(T_1) \cap V(T_2) = \{r\}$, let $\mathsf{Join}((T_1, r), (T_2, r))$ be the rooted tree (T, r) where T is the union of T_1 and T_2, i.e., $V(T) = V(T_1) \cup V(T_2)$ and $E(T) = E(T_1) \cup E(T_2)$. This corresponds to the usual join node in nice tree decompositions only restricted to graphs with bags of size exactly one.

We say that an expression built out of the operations Leaf, Extend, and Join is a *neat tree decomposition*. Furthermore, we say that a graph G *admits a neat tree decomposition* if there is such a decomposition whose resulting in exactly G. It is easy to see that every rooted tree admits a neat tree decomposition that can be efficiently computed.

Observation 1 *Every rooted tree (T, r) admits a neat tree decomposition that can be computed in time $O(|V(T)|)$.*

Polynomial Algorithm

We say that a set M of vertices in a rooted forest (F, r) is of type 1 if $r \in M$ and of type 0 otherwise, i.e., if $r \notin M$. We define the *reduced size of* M to be $|M \setminus \{r\}|$, i.e., we do not count the root r. A pair of functions (α, β) where $\alpha\colon \{0,1\} \to [0, n]$ and $\beta\colon \{0,1\} \to [2]$ is a *characteristic* of a set $S \subseteq V(T) \setminus \{r\}$ if for both $i \in \{0, 1\}$,

- $\alpha(i)$ is the reduced size of the smallest vertex cover of type i in the rooted forest $(T - S, r)$, and
- $\beta(i) = 1$ if and only if there is a unique vertex cover of reduced size $\alpha(i)$ and type i in $(T - S, r)$.

Notice that we only consider sets S that do not contain the root. Moreover, observe that a vertex cover M of type 0 in $(T - S, r)$ can be extended to a vertex cover $M \cup \{r\}$ of the same reduced size and type 1. This implies the following inequality for the characteristic of an arbitrary set S.

Observation 2 *For a rooted tree (T, r) and a set $S \subseteq T \setminus \{r\}$ with characteristic (α, β), we have $\alpha(1) \leq \alpha(0)$.*

Let G be the input tree and let $v \in V(G)$ be its arbitrary vertex. By Observation 1, the rooted tree (G, v) admits a neat tree decomposition ψ. The algorithm proceeds by dynamic programming along the neat tree decomposition. Specifically, for each rooted tree (T, r) generated by a subexpression of ψ, it stores a dynamic programming table $\mathrm{DP}^{\mathsf{t}}_T[\alpha, \beta]$ such that (α, β) is a possible characteristic, and the value of $\mathrm{DP}^{\mathsf{t}}_T[\alpha, \beta]$ contains the size of the smallest set of characteristic (α, β) in (T, r), or ∞ if no such set exists.

First, we observe that the characteristic of a given set S carries enough information to decide whether S is a feasible solution to MU-VC.

Claim 1 *A set S is a feasible solution to* MU-VC *in a rooted tree (T, r) if and only if one of the following holds*

1. *$r \notin S$ and the set S has characteristic (α, β) such that either $\alpha(0) < \alpha(1)+1$ and $\beta(0) = 1$ or $\alpha(1)+1 < \alpha(0)$ and $\beta(1) = 1$, or*
2. *$r \in S$ and the set $S \setminus \{r\}$ has characteristic (α, β) such that $\beta(1) = 1$.*

Therefore, after we compute $\mathrm{DP}^{\mathsf{t}}_G[\cdot, \cdot]$ for the input tree G, the algorithm simply returns the minimum value out of (i) $\mathrm{DP}^{\mathsf{t}}_G[\alpha, \beta]$ such that $\alpha(0) < \alpha(1) + 1$ and $\beta(0) = 1$ or $\alpha(1) + 1 < \alpha(0)$ and $\beta(1) = 1$; and (ii) $\mathrm{DP}^{\mathsf{t}}_G[\alpha, \beta] + 1$ such that $\beta(1) = 1$. Once we have received this value, we can then use the corresponding values of $\mathrm{DP}^{\mathsf{t}}_G[\cdot, \cdot]$ to build the set S.

Now, it remains to show how the individual operations act on the characteristic of a fixed set and how to use this to fill the dynamic programming tables.

Leaf node, $(T, r) = \mathsf{Leaf}(r)$**.** There is only a single possible choice of a set $S \subseteq V(T) \setminus \{r\}$ as the empty set and its characteristic is $(\alpha_{\mathsf{leaf}}, \beta_{\mathsf{leaf}})$ where $\alpha_{\mathsf{leaf}}(0) = \alpha_{\mathsf{leaf}}(1) = 0$ and $\beta_{\mathsf{leaf}}(0) = \beta_{\mathsf{leaf}}(1) = 1$ since there are unique vertex covers of both types with reduced size 0. We set

$$\mathrm{DP}^{\mathsf{t}}_T[\alpha, \beta] = \begin{cases} 0 & \text{if } (\alpha, \beta) = (\alpha_{\mathsf{leaf}}, \beta_{\mathsf{leaf}}), \text{ and} \\ \infty & \text{otherwise.} \end{cases} \tag{1}$$

Join node, $(T, r) = \mathsf{Join}((T_1, r), (T_2, r))$**.** Let us define a function f acting on pairs of characteristics $(\alpha_1, \beta_1), (\alpha_2, \beta_2)$ as $f((\alpha_1, \beta_1), (\alpha_2, \beta_2)) = (\alpha, \beta)$ where, for both $i \in \{0, 1\}$,

$$\begin{aligned} \alpha(i) &= \alpha_1(i) + \alpha_2(i), \text{ and} \\ \beta(i) &= \min(2,\ \beta_1(i) \cdot \beta_2(i)). \end{aligned} \tag{2}$$

Claim 2 *Let $S_1 \subseteq V(T_1) \setminus \{r\}$ be an arbitrary set of characteristic (α_1, β_1) in T_1, let $S_2 \subseteq V(T_2) \setminus \{r\}$ be an arbitrary set of characteristic (α_2, β_2) in T_2, and let (α, β) be the image of $(\alpha_1, \beta_1), (\alpha_2, \beta_2)$ under f. Then $S_1 \cup S_2$ has characteristic (α, β) in T.*

The computation of $\mathrm{DP}^{\mathsf{t}}_T[\cdot, \cdot]$ finds the smallest sum $\mathrm{DP}^{\mathsf{t}}_{T_1}[\alpha_1, \beta_1] + \mathrm{DP}^{\mathsf{t}}_{T_2}[\alpha_2, \beta_2]$ over the preimages of (α, β) under f. That is, we set

$$\mathrm{DP}^{\mathsf{t}}_T[\alpha, \beta] = \min_{((\alpha_1,\beta_1),(\alpha_2,\beta_2)) \in f^{-1}(\alpha,\beta)} \mathrm{DP}^{\mathsf{t}}_{T_1}[\alpha_1, \beta_1] + \mathrm{DP}^{\mathsf{t}}_{T_2}[\alpha_2, \beta_2] \tag{3}$$

where we take the minimum over the empty set to be ∞.

Extend node, $(T, r) = \mathsf{Extend}((T', r'), r)$**.** In the extend operation, we have two possibilities depending on whether we add the old root r' to the set S or not. Thus, we define two functions g_{id} and g_+ acting on characteristics that describe how the characteristic of a fixed set $S \subseteq V(T') \setminus \{r'\}$ translates to the characteristics of the sets S and $S \cup \{r'\}$ in T respectively. First, we define $g_{\mathsf{id}}(\alpha', \beta') = (\alpha, \beta)$[1] where

$$\begin{aligned} \alpha(0) &= \alpha'(1) + 1 \\ \alpha(1) &= \min(\alpha'(0),\ \alpha'(1) + 1) \\ \beta(0) &= \beta'(1) \\ \beta(1) &= \begin{cases} \beta'(0) & \text{if } \alpha'(0) < \alpha'(1) + 1, \\ \beta'(1) & \text{if } \alpha'(0) > \alpha'(1) + 1, \text{ and} \\ 2 & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

Now, we define $g_+(\alpha', \beta') = (\alpha, \beta)$ where

$$\begin{aligned} \alpha(0) &= \alpha(1) = \alpha'(1), \text{ and} \\ \beta(0) &= \beta(1) = \beta'(1). \end{aligned} \tag{5}$$

Claim 3 *Let $S \subseteq V(T') \setminus \{r'\}$ be an arbitrary set with characteristic (α', β') in T'. Then S has characteristic $g_{\mathsf{id}}(\alpha', \beta')$ in T and $S \cup \{r'\}$ has characteristic $g_+(\alpha', \beta')$ in T.*

[1] We omit the extra parentheses inside $g_{\mathsf{id}}(\cdot)$ for readability.

The computation of $\mathrm{DP}^{\mathsf{t}}_T[\cdot, \cdot]$ finds the minimum between $\mathrm{DP}^{\mathsf{t}}_{T'}[\alpha', \beta']$ over the preimages of (α, β) under g_{id} and $\mathrm{DP}^{\mathsf{t}}_{T'}[\alpha', \beta'] + 1$ over the preimages of (α, β) under g_+. That is, we set

$$\mathrm{DP}^{\mathsf{t}}_T[\alpha, \beta] = \min \begin{pmatrix} \min\limits_{(\alpha',\beta') \in g_{\mathsf{id}}^{-1}(\alpha,\beta)} \mathrm{DP}^{\mathsf{t}}_{T'}[\alpha', \beta'], \\ \min\limits_{(\alpha',\beta') \in g_+^{-1}(\alpha,\beta)} \mathrm{DP}^{\mathsf{t}}_{T'}[\alpha', \beta'] + 1 \end{pmatrix} \tag{6}$$

where we take the minimum over empty set to be ∞.

Let us now discuss the time complexity of the algorithm on an input tree G with n vertices. Any neat decomposition of G clearly has size $O(n)$ and the total number of possible characteristics is $O(n^2)$. Therefore, a straightforward implementation of the computations in (1), (3) and (6) results in a polynomial-time algorithm. Moreover, we remark that a more careful implementation achieves runtime $O(n^5)$.

The correctness of the algorithm follows straightforwardly from Claims 2 and 3 by a bottom-up induction over the neat tree decomposition.

Linear Algorithm

In order to speed up the previous algorithm, we use two ideas that allow us to group the possible characteristics into constantly many classes. First, the size of the minimum vertex cover is irrelevant in MU-VC. Thus, capturing $\alpha(0) - \alpha(1)$ suffices to see whether the minimum vertex covers of two possible types have the same size. A reasonable implementation would already bring down the runtime to $O(n^3)$. However, it actually suffices to remember whether $\alpha(0) - \alpha(1)$ is equal to 0, 1, or at least 2.

Formally, a *reduced characteristic* of a set $S \subseteq V(T) \setminus \{r\}$ with characteristic (α, β) in (T, r) is a pair (δ, β) where $\delta = \min(2, \alpha(0) - \alpha(1))$. By Observation 2, we see that $\delta \in \{0, 1, 2\}$. Moreover, β is one of 4 possible functions. Thus there are only $3 \cdot 4 = 12$ possible reduced characteristics. The algorithm follows the same scheme as before. In particular for each rooted tree (T, r) generated by a subexpression of the neat tree decomposition, it fills a dynamic programming table $\mathrm{DP}^{\mathsf{lt}}_T[\cdot, \cdot]$ such that $\mathrm{DP}^{\mathsf{lt}}_T[\delta, \beta]$ contains the size of the smallest set $S \subseteq V(T) \setminus \{r\}$ with reduced characteristic (δ, β).

We start by observing that the feasibility of a given set S as a solution to MU-VC can be deduced from its reduced characteristic.

Claim 4 *A set S is a feasible solution to* MU-VC *in a rooted tree (T, r) if and only if one of the following holds*

1. *$r \notin S$ and the set S has reduced characteristic (δ, β) such that either $\delta = 0$ and $\beta(0) = 1$, or $\delta = 2$ and $\beta(1) = 1$, or*
2. *$r \in S$ and the set $S \setminus \{r\}$ has reduced characteristic (δ, β) such that $\beta(1) = 1$.*

So after computing the whole table $\mathrm{DP}^{\mathsf{lt}}_G[\cdot, \cdot]$ for the input tree G, the algorithm returns the minimum value out of (i)

$\mathrm{DP}^{\mathsf{lt}}_G[\delta,\beta]$ where $\delta = 0$ and $\beta(0) = 1$, or $\delta = 2$ and $\beta(1) = 1$; and (ii) $\mathrm{DP}^{\mathsf{lt}}_G[\delta,\beta]+1$ where $\beta(1) = 1$.

The most important part is to show that reduced characteristics still allow a similar scheme of computation to the full characteristics. Namely, we show how the reduced characteristic of a set changes under the Leaf, Extend, and Join operations.

Leaf node, $(T,r) = \mathsf{Leaf}(r)$**.** We have seen that the characteristic of the only possible set $S \subseteq V(T) \setminus \{r\}$ is $(\alpha_{\mathsf{leaf}},\beta_{\mathsf{leaf}})$ and thus, its reduced characteristic is $(\delta_{\mathsf{leaf}},\beta_{\mathsf{leaf}})$ where $\delta_{\mathsf{leaf}} = \alpha_{\mathsf{leaf}}(0) - \alpha_{\mathsf{leaf}}(1) = 0$. Again, we set

$$\mathrm{DP}^{\mathsf{lt}}_T[\delta,\beta] = \begin{cases} 0 & \text{if } (\delta,\beta) = (\delta_{\mathsf{leaf}},\beta_{\mathsf{leaf}}), \text{ and} \\ \infty & \text{otherwise.} \end{cases} \tag{7}$$

Join node, $(T,r) = \mathsf{Join}((T_1,r),(T_2,r))$**.** Let us define a function f' acting on pairs of reduced characteristics $(\delta_1,\beta_1),(\delta_2,\beta_2)$ as $f'((\delta_1,\beta_1),(\delta_2,\beta_2)) = (\delta,\beta)$ where

$$\begin{aligned} \delta &= \min(2,\ \delta_1+\delta_2), \text{ and} \\ \beta(i) &= \min(2,\ \beta_1(i)\cdot\beta_2(i)) \text{ for both } i \in \{0,1\}. \end{aligned} \tag{8}$$

Claim 5 *Let $S_1 \subseteq V(T_1) \setminus \{r\}$ be an arbitrary set of reduced characteristic (δ_1,β_1) in T_1, let $S_2 \subseteq V(T_2) \setminus \{r\}$ be an arbitrary set of reduced characteristic (δ_2,β_2) in T_2. Then $S_1 \cup S_2$ has reduced characteristic $f'((\delta_1,\beta_1),(\delta_2,\beta_2))$ in T.*

The computation of $\mathrm{DP}^{\mathsf{lt}}_T[\cdot,\cdot]$ follows the same scheme as in the previous case. That is, we set

$$\mathrm{DP}^{\mathsf{lt}}_T[\delta,\beta] = \min_{((\delta_1,\beta_1),(\delta_2,\beta_2))\in(f')^{-1}(\delta,\beta)} \mathrm{DP}^{\mathsf{lt}}_{T_1}[\delta_1,\beta_1] + \mathrm{DP}^{\mathsf{lt}}_{T_2}[\delta_2,\beta_2] \tag{9}$$

where we take the minimum over the empty set to be ∞.

Extend node, $(T,r) = \mathsf{Extend}((T',r'),r)$**.** We again define two functions g'_{id} and g'_+ describing how the reduced characteristics of the set S and $S\cup\{r'\}$ in T depend on the reduced characteristic of a fixed set $S \subseteq V(T')\setminus\{r'\}$ in T'. First, we define g'_{id} such that $g'_{\mathsf{id}}(\delta',\beta') = (\delta,\beta)$ where

$$\begin{aligned} \delta &= \begin{cases} 0 & \text{if } \delta' \geq 1, \\ 1 & \text{otherwise, i.e., if } \delta' = 0. \end{cases} \\ \beta(0) &= \beta'(1), \text{ and} \\ \beta(1) &= \begin{cases} \beta'(0) & \text{if } \delta' = 0, \\ \beta'(1) & \text{if } \delta' \geq 2, \text{ and} \\ 2 & \text{otherwise, i.e., if } \delta' = 1. \end{cases} \end{aligned} \tag{10}$$

Now, we define g'_+ such that $g'_+(\delta',\beta') = (\delta,\beta)$ where

$$\delta = 0, \qquad \beta(0) = \beta(1) = \beta'(1). \tag{11}$$

Claim 6 *Let $S \subseteq V(T')\setminus\{r'\}$ be an arbitrary set with reduced characteristic (δ',β') in T'. Then S has reduced characteristic $g'_{\mathsf{id}}(\delta',\beta')$ in T and $S\cup\{r'\}$ has reduced characteristic $g'_+(\delta',\beta')$ in T.*

The computation of $\mathrm{DP}^{\mathsf{lt}}_T[\cdot,\cdot]$ is again analogous to the previous algorithm. That is, we set

$$\mathrm{DP}^{\mathsf{lt}}_T[\delta,\beta] = \min\left(\begin{array}{c} \min\limits_{(\alpha',\beta')\in(g'_{\mathsf{id}})^{-1}(\alpha,\beta)} \mathrm{DP}^{\mathsf{lt}}_{T'}[\alpha',\beta'], \\ \min\limits_{(\alpha',\beta')\in(g'_+)^{-1}(\alpha,\beta)} \mathrm{DP}^{\mathsf{lt}}_{T'}[\alpha',\beta']+1 \end{array}\right) \tag{12}$$

where we take the minimum over empty set to be ∞.

This finishes the description of the computation. The correctness of the algorithm follows from Claims 5 and 6 by a bottom-up induction over the nice tree decomposition.

It remains to argue about the running time. As already observed, the total number of possible reduced characteristics is 12. It follows that a straightforward computation of the table $\mathrm{DP}^{\mathsf{lt}}_T[\cdot,\cdot]$ in each node of the neat decomposition finishes in $O(1)$ time (refer to (7), (9), and (12)). Therefore, the overall running time is $O(n)$.

Treewidth

The algorithm for treewidth follows the same general scheme as the algorithms for trees in the previous section: we define a suitable characteristic of any subset S of vertices such that (i) we can decide whether S is a feasible solution just from its characteristic, (ii) the way the characteristic of S changes in a node of a tree decomposition depends only on its previous characteristic, and (iii) the total number of characteristics is polynomial in the size of the input graph. We then compute by a dynamic-programming scheme in every node of a nice tree decomposition the minimum size of a set S with each possible characteristic.

Theorem 2 *The* MU-VC *problem can be solved by an* XP*-algorithm parameterized by the treewidth d of G in time $n^{O(2^d)}$.*

Sketch of proof. Due to space limitations, we only define the generalization of (reduced) characteristics. We first generalize rooted trees. A *terminal graph* is a pair (G,X) where G is a graph, $X \subseteq V(G)$ is a subset of its vertices, and $G[X]$ is an independent set, i.e., there are no edges between vertices of X. For a terminal graph (G,X), we say that a set $M \subseteq V(G)$ is of *type* D if $M \cap X = D$. Moreover, the *reduced size* of a set $M \subseteq V(G)$ is defined as $|M \setminus X|$.

A pair of functions (α,β) where $\alpha\colon 2^X \to \{0,\ldots,n\}$ and $\beta\colon \{0,1\} \to \{1,2\}$ is a *characteristic* of a set $S \subseteq V(G)\setminus X$ in a terminal graph (G,X) if for every $D \subseteq X$,

- $\alpha(D)$ is the reduced size of the smallest vertex cover of type D in the terminal graph $(G-S,X)$, and
- $\beta(D) = 1$ if and only if there is a unique vertex cover of reduced size $\alpha(D)$ and type D in $(G-S,X)$.

The characteristic is well defined: there always exists a vertex cover of any type D, e.g., $V(G)\setminus X \cup D$. Finally, remark that a rooted tree (T,r) can be interpreted as the terminal graph $(T,\{r\})$ where the sets of type 0 and 1 in (T,r) are exactly the sets of type $\emptyset$ and $\{r\}$ in $(T,\{r\})$ respectively.

We again define reduced characteristics. A *reduced characteristic* of a set $S \subseteq V(G)\setminus X$ with characteristic (α,β) in

a terminal graph (G, X) is a pair of functions (δ, β) where $\delta\colon 2^X \to \{0, \ldots, n\}$ such that $\delta(D) = \alpha(\emptyset) - \alpha(D)$ for every $D \subseteq X$.

Let x be a node of a nice tree decomposition $(T, \{B_x \mid x \in V(T)\})$ of the graph G. Let Y_x be the set of all vertices contained in the bags of the subtree rooted in t. We associate with the node x a terminal graph (G_x, B_x) such that G_x is obtained from the graph $G[Y_x]$ by removing all edges connecting two vertices of B_x. In other words, we consider an edge only when one of its endpoints is being forgotten.

The algorithm computes a nice tree decomposition of the graph G. Then it fills for each node x in the nice tree decomposition a dynamic programming table $\mathrm{DP}_x^{\mathsf{tw}}[\delta, \beta]$ such that (δ, β) is a reduced characteristic, and the entry $\mathrm{DP}_x^{\mathsf{tw}}[\delta, \beta]$ contains the size of the smallest set with reduced characteristic (δ, β) in (G_x, B_x), or ∞ if no such set exists. ◇

Furthermore, we show that the same dynamic programming scheme yields an FPT-algorithm when we additionally parameterize by the maximum degree of the input graph.

Theorem 3 *The* MU-VC *problem can be solved by an* FPT*-algorithm parameterized by the treewidth w of G plus the maximum degree Δ in time $\Delta^{O(2^w)} \cdot n$.*

Sketch of proof. Crucially, in low degree graphs, the sizes of minimum vertex covers of different types cannot be too far away from each other. Precisely, let (G, X) be a terminal graph with maximum degree Δ and let $S \subseteq V(G) \setminus X$ be a set of its vertices with reduced characteristic (δ, β). Then we have $0 \leq \delta(D) \leq \Delta \cdot |D|$ for every $D \subseteq X$. It follows that the number of possible reduced characteristics in a terminal graph (G, X) is at most $(\Delta \cdot w)^{O(2^w)} = \Delta^{O(2^w)}$. ◇

Clique-Width

Theorem 4 *The* MU-VC *problem can be solved by an* XP*-algorithm w.r.t the clique-width d of G in $n^{O(2^d)}$ time.*

Sketch of proof. The algorithm follows a scheme analogous to the algorithms in previous sections. Due to the lack of space, we only describe what is the suitable definition of a characteristic of a set S in this case.

For a subset of vertices X of a d-labeled graph G, we denote by $\mathrm{full}_G(X)$ the set of labels from $[d]$ that are fully contained within X.Conversely for $I \subseteq [d]$, we say that T is a *set of type* I if $\mathrm{full}_G(T) = I$. Additionally for a type $I \subseteq [d]$ and a set T we say that T *extends type* I if $I \subseteq \mathrm{full}_G(T)$.

A pair of functions (α, β) where $\alpha\colon 2^{[d]} \to \{0, \ldots, n\}$ and $\beta\colon 2^{[d]} \to \{1, 2\}$ is a *characteristic* of a set $S \subseteq V(G)$ if for every $I \subseteq [d]$,

- $\alpha(I)$ is the size of the smallest vertex cover that extends type I in $G - S$, and
- $\beta(I) = 1$ if and only if the smallest vertex cover of size $\alpha(I)$ that extends type I in $G - S$ is unique.

The algorithm then proceeds by dynamic programming along a given clique-width d-expression ψ of G. Specifically, for each d-labeled graph H generated by a subexpression of ψ, it stores a dynamic programming table $\mathrm{DP}_H^{\mathsf{cw}}[\alpha, \beta]$ such that (α, β) is a possible characteristic, and the value of $\mathrm{DP}_H^{\mathsf{cw}}[\alpha, \beta]$ contains the size of the smallest set with characteristic (α, β) in H, or ∞ if no such set exists. ◇

Theorem 5 *The* MU-VC *problem can be solved by an* FPT*-algorithm parameterized by the clique-width d of G plus the size of solution k in time $k^{O(2^d)} \cdot n$.*

Sketch of proof. The result is obtained by truncating the dynamic programming table of the algorithm in Theorem 4. That is possible because the characteristic of a small set S cannot be too far away from the characteristic of the empty set. To be more precise, let $S \subseteq V(H)$ be a set of vertices in a d-labeled graph H with characteristic (α_S, β_S) and let $(\alpha_\emptyset, \beta_\emptyset)$ be the characteristic of the empty set in H. Then for arbitrary type $I \subseteq [d]$, we have $0 \leq \alpha_\emptyset(I) - \alpha_S(I) \leq |S|$.

The algorithm proceeds in two passes over a clique-width d-expression ϕ of the input graph G. In the first pass, it computes the characteristic of the empty set in every d-labeled graph generated by a subexpression of ϕ. In the second pass, it follows the computation of the algorithm from Theorem 4 restricted to characteristics with small distance to the characteristic of the empty set. ◇

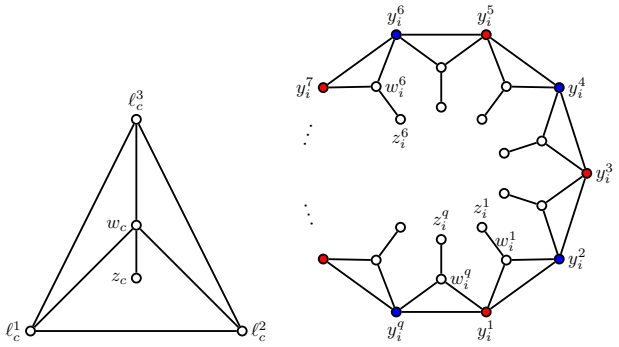
Hardness on Planar Graphs

Theorem 6 *Both the* MU-VC *and* PAU-VC *problems are Σ_2^P-complete even when the input graph G is planar and of maximum degree 5.*

Sketch of proof. We first argue about MU-VC belonging in Σ_2^P. Given a graph G and integer k, assume we have a set $S \subseteq V(G)$ such that $|S| \leq k$ and $G' = G - S$ has a unique minimum vertex cover U. To verify that U is indeed as required, it suffices to solve PAU-VC on G' for $k = 0$, which can be done in polynomial time with the help of an NP-oracle (Horiyama et al. 2024). We then focus on proving that MU-VC is Σ_2^P-hard for planar graphs of maximum degree 5. Slight modifications in our proof can lead to the same hardness result for the same family of graphs for PAU-VC.

The reduction is from UQ PLANAR 1-IN-3 SAT (Demaine et al. 2018): we are given a 3CNF formula ϕ on the set of variables $\{x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2}\}$ and clauses $C = \{c_1, \ldots, c_m\}$. Variables in $\{x_1, \ldots, x_{n_1}\}$ ($\{y_1 \ldots, y_{n_2}\}$ resp.) are of *type* x (*type* y resp.). The task is to find a truth-assignment of the variables of type x such that there exists a unique truth-assignment of the variables of type y where each clause of C is satisfied by exactly one literal. We will construct a graph G which has an MU-VC of order n_1 if and only if ϕ is a yes-instance of UQ PLANAR 1-IN-3 SAT.

To construct the graph G, we first build an auxiliary graph H as follows. We define a *variable* (*clause* resp.) vertex for each variable (clause resp.) in ϕ. We then add an edge between a variable and a clause vertex if the corresponding variable appears in the corresponding clause; let H be the resulting graph. Observe that H is a planar graph (due to the "planarity" of ϕ). We now go from H to G. First, replace each clause vertex c of H by a copy of the c-gadget shown in



(a) The c-gadget. (b) The y_i-gadget.

(c) The x_i-gadget.

Figure 2: The gadgets used in the proof of Theorem 6. Inner colored vertices of the first appearance of y_i are y_i^2 and y_i^3.

Figure 2(a). Then, replace each variable vertex by either a y-gadget or a x-gadget, shown in Figures 2(b) and 2(c) respectively, according to the type of the corresponding variable in ϕ. The x- and y-gadgets are similar. Consider the y_i-gadget, i.e., the gadget corresponding to the variable y_i that appears in ϕ. It will have four colored vertices (see Figure 2(b)) for each appearance of y_i in ϕ; two red vertices and two blue. That is, the index q that appears in Figure 2(b) is equal to four times the number of appearances of y_i in ϕ. Moreover, going in an anti-clockwise fashion, the colored vertices that correspond to each appearance of the y_i variable will alternate, starting with a red and finishing with a blue; for the j-th appearance of variable y_i, we say that vertices $y_i^{4(j-1)+2}$ and $y_i^{4(j-1)+3}$ denote its *inner colored vertices* (see Figure 2(b) for an example). The inner blue (red resp.) vertex included in the gadget for an appearance of the variable y_i will model that this variable is set to false (true resp.), while the other inner colored vertices will serve as points of additional connection between the gadgets. The same holds true for the x_i-gadget which, in addition, contains an extra set of colored vertices together with two pending paths, illustrated by the bold vertices and edges in Figure 2(c).

At this stage, all the original edges of H are removed. We add the new edges between the gadgets through a two-step *edge-adding procedure*. First we deal with the edges connecting the c-gadgets to the x- and/or y-gadgets. Consider a clause c and its corresponding c-gadget and assume that, in the graph H there was an edge between the clause vertex c and the variable vertex of type x. Also, let x be the j-th appearance of the x variable in ϕ (according to a carefully chosen ordering of the variables. Then, going anti-clockwise, we locate the j-th quadruple Q of colored vertices of the x-gadget. We then add an edge between any vertex of the c-gadget that is currently of degree three and the blue (red resp.) inner vertex of Q if this appearance of x is positive (negative resp.). We repeat the same procedure for all the edges of H that are between the clause vertex c and any variable vertex of type y. Once we are done, we move on and repeat this procedure for every clause vertex of H. This completes the first step of the edge-adding procedure.

In the second step, we connect the x- and/or y-gadgets whose corresponding variables appear in a common clause. To ease the exhibition, and since we treat these gadgets in the same way, we will assume we only have to deal with x-gadgets. So, consider a clause gadget c, with the corresponding clause being comprised of three literals on the variables x_{i_1}, x_{i_2}, and x_{i_3} (for some $i_1, i_2, i_3 \in [n_1]$). According to the construction of G up to this point, there are

- a c-gadget, containing the vertices ℓ_c^1, ℓ_c^2, and ℓ_c^3, and
- the x_{i_1}, x_{i_2}, and x_{i_3}-gadgets, containing some inner colored vertices $x_{i_1}^\alpha$, $x_{i_2}^\beta$, and $x_{i_3}^\gamma$ respectively such that G contains the edges $\ell_c^1 x_{i_1}^\alpha$, $\ell_c^2 x_{i_2}^\beta$, and $\ell_c^3 x_{i_3}^\gamma$.

Note that since $x_{i_1}^\alpha$, $x_{i_2}^\beta$, and $x_{i_3}^\gamma$ are inner colored vertices, and according to the first step of the edge-adding procedure, the two colored neighbors of these vertices that lie in the x_{i_1}, x_{i_2}, and x_{i_3}-gadgets respectively are currently of degree 4. The second step of the edge-adding procedure consists in adding the edges $x_{i_3}^{\gamma-1} x_{i_1}^{\alpha+1}$, $x_{i_1}^{\alpha-1} x_{i_2}^{\beta+1}$, and $x_{i_2}^{\beta-1} x_{i_3}^{\gamma+1}$. This step is repeated for every clause gadget c.

This marks the end of the construction of G. We ensure the planarity of G based on the planarity of H and by carefully choosing the ordering used in the first step and by bending the edges added in the second step of the edge-adding procedure to closely shadow the preexisting edges of G.

We are now ready to sketch our reduction. Assume that we have a yes-instance of UQ PLANAR 1-IN-3 SAT testified by the truth-assignment σ. We will show that the constructed graph $G = (V, E)$ has a MU-VC of order exactly n_1. Due to the x-gadgets, it suffices to provide a set $S \subseteq V$ such that $G' = G - S$ has a unique minimum vertex cover U, and $|S| = n_1$. We proceed as follows: for each $i \in [n_1]$, if $\tau(x_i) = \texttt{true}$ ($\tau(x_i) = \texttt{false}$ resp.), we include u_i^5 (u_i^2 resp.) in S. It follows that $|S| = n_1$. It remains to show that $G' = G - S$ has a unique minimum vertex cover U. Observe that for every variable gadget, if U contains a blue (red resp.) vertex, then it should also contain all other blue (red resp.) vertices, and no red (blue resp.) vertex of the same gadget. In particular, U will contain exactly all the ws, the blue (red resp.) vertices of the x-gadgets and y-gadgets that are set to true (false resp.) and two out the three outer vertices of each c-gadget whose corresponding literals do not satisfy c. The uniqueness and minimiality of this U is guaranteed by the formula ϕ being 1-in-3 satisfied by σ and vice versa. $\diamond$

Acknowledgments

The first, second, and fourth authors were supported by the European Union under the project Robotics and advanced industrial production (reg. no. CZ.02.01.01/00/22_008/0004590). The first author was additionally supported by the International Mobility of Researchers MSCA-F-CZ-III at CTU in Prague, CZ.02.01.01/00/22_010/0008601 Programme Johannes Amos Comenius. The third author was partially supported by Charles University projects UNCE 24/SCI/008 and PRIMUS 24/SCI/012, and by the project 25-17221S of GAČR. The fifth author was supported by the ANR project ANR-21-CE48-0022 (S-EX-AP-PE-AL) and the Barrande Fellowship programme.

References

An, S.; Chang, Y.; Cho, K.; Kwon, O.; Lee, M.; Oh, E.; and Shin, H. 2025. Pre-Assignment Problem for Unique Minimum Vertex Cover on Bounded Clique-Width Graphs. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence*, 26886–26894. AAAI Press.

Asuncion, A.; Newman, D.; et al. 2007. UCI machine learning repository.

Bodlaender, H. L. 1996. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM Journal on Computing*, 25(6): 1305–1317.

Bodlaender, H. L. 1998. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1): 1–45.

Chang, C.-C.; and Lin, C.-J. 2011. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3): 1–27.

Courcelle, B.; and Olariu, S. 2000. Upper bounds to the clique width of graphs. *Discret. Appl. Math.*, 101(1-3): 77–114.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer. ISBN 978-3-319-21274-6.

Demaine, E. D.; Ma, F.; Schvartzman, A.; Waingarten, E.; and Aaronson, S. 2018. The fewest clues problem. *Theor. Comput. Sci.*, 748: 28–39.

Diestel, R. 2025. *Graph Theory*, volume 173 of *Graduate texts in mathematics*. Springer. ISBN 978-3-662-70106-5.

Downey, R. G.; and Fellows, M. R. 2012. *Parameterized complexity*. Springer Science & Business Media.

Fan, L.; Hua, W.; Li, L.; Ling, H.; and Zhang, Y. 2023. NPHardEval: Dynamic Benchmark on Reasoning Ability of Large Language Models via Complexity Classes. arXiv:2312.14890.

Fan, L.; Hua, W.; Li, X.; Zhu, K.; Jin, M.; Li, L.; Ling, H.; Chi, J.; Wang, J.; Ma, X.; and Zhang, Y. 2024. NPHardEval4V: A Dynamic Reasoning Benchmark of Multimodal Large Language Models. arXiv:2403.01777.

Flum, J.; and Grohe, M. 2006. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer. ISBN 978-3-540-29952-3.

Hagberg, A.; Swart, P. J.; and Schult, D. A. 2008. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States).

Hoos, H. H.; and Stützle, T. 2000. SATLIB: An online resource for research on SAT. *Sat*, 2000: 283–292.

Horiyama, T.; Kobayashi, Y.; Ono, H.; Seto, K.; and Suzuki, R. 2024. Theoretical Aspects of Generating Instances with Unique Solutions: Pre-assignment Models for Unique Vertex Cover. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 20726–20734.

Hudry, O.; and Lobstein, A. 2019. Complexity of unique (optimal) solutions in graphs: Vertex Cover and Domination. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 110: 217–240.

Kloks, T. 1994. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer. ISBN 3-540-58356-4.

Koch, T.; Achterberg, T.; Andersen, E.; Bastert, O.; Berthold, T.; Bixby, R. E.; Danna, E.; Gamrath, G.; Gleixner, A. M.; Heinz, S.; et al. 2011. MIPLIB 2010: Mixed integer programming library version 5. *Mathematical Programming Computation*, 3: 103–163.

Korhonen, T.; and Lokshtanov, D. 2023. An Improved Parameterized Algorithm for Treewidth. In Saha, B.; and Servedio, R. A., eds., *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, 528–541. ACM.

Nath, A.; and Kuhnle, A. 2024. A Benchmark for Maximum Cut: Towards Standardization of the Evaluation of Learned Heuristics for Combinatorial Optimization. arXiv:2406.11897.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press. ISBN 9780198566076.

Reinelt, G. 1991. TSPLIB—A traveling salesman problem library. *ORSA journal on computing*, 3(4): 376–384.

Tang, J.; Zhang, Q.; Li, Y.; and Li, J. 2024. GraphArena: Benchmarking Large Language Models on Graph Computational Problems. arXiv:2407.00379.