# Pre-Assignment Problem for Unique Minimum Vertex Cover on Bounded Clique-Width Graphs

**Shinwoo An[1], Yeonsu Chang[2], Kyungjin Cho[1], O-joung Kwon[2,3*], Myounghwan Lee[2], Eunjin Oh[1], Hyeonjun Shin[1]**

[1] Department of Computer Science and Engineering, POSTECH, Pohang, South Korea
[2] Department of Mathematics, Hanyang University, Seoul, South Korea
[3] Discrete Mathematics Group, Institute for Basic Science (IBS), Daejeon, South Korea
shinwooan@postech.ac.kr, yeonsu@hanyang.ac.kr, kyungjincho@postech.ac.kr, {ojoungkwon, sycuel}@hanyang.ac.kr, {eunjin.oh, tlsguswns119}@postech.ac.kr

## Abstract

Horiyama et al. (AAAI 2024) considered the problem of generating instances with a unique minimum vertex cover under certain conditions. The PRE-ASSIGNMENT FOR UNIQUIFICATION OF MINIMUM VERTEX COVER problem (shortly PAU-VC) is the problem, for given a graph $G$, to find a minimum set $S$ of vertices in $G$ such that there is a unique minimum vertex cover of $G$ containing $S$. We show that PAU-VC is fixed-parameter tractable parameterized by clique-width, which improves an exponential algorithm for trees given by Horiyama et al. Among natural graph classes with unbounded clique-width, we show that the problem can be solved in polynomial time on split graphs and unit interval graphs.

## Introduction

Designing AI algorithms to tackle NP-hard graph problems has become a prominent trend in the field of artificial intelligence. The inherent complexity of NP-complete problems presents a significant challenge, making them an ideal testbed for AI-driven approaches that aim to push the boundaries of what can be achieved in terms of efficiency and scalability. To evaluate the performance of those AI algorithms, it is essential to have robust benchmark datasets. Such datasets provide a controlled environment where the strengths and weaknesses of different algorithms can be systematically analyzed. As constructing a benchmark dataset is a critical aspect of AI research, several well-known benchmark datasets were presented such as TSPLIB, UCI, SATLIB, and DIMACS for various NP-hard combinatorial problems (Reinelt 1991; Asuncion, Newman et al. 2007; Hoos and Stützle 2000).

However, it seems hard to use them to evaluate the performances of AI algorithms for the *uniqueness version* of combinatorial problems where a solution is unique. In several problems, the presence of a unique solution can lead to more efficient algorithms (Thomason 1978; Gabow, Kaplan, and Tarjan 1999). Also, algorithms for the unique SAT problem are used as subroutines for its search version (Scheder and Steinberger 2017; Hertli 2014a). Due to these reasons, the uniqueness version also has been extensively studied from both theory and practice (Calabro et al. 2008; Hertli 2014b). Therefore, generating graphs with a unique solution offers a valuable addition to benchmark datasets, enabling a more thorough evaluation of AI-driven solvers for the uniqueness version of combinatorial problems.

One natural approach for generating graphs with a unique solution is to make use of graphs in well-known benchmark datasets. More specifically, we choose a graph $G$ in a well-known benchmark dataset, and pre-assign a part of $G$ so that only one solution is consistent with this assignment. This *pre-assignment for uniquification* has been studied for classic NP-hard problems such as the coloring and clique problems (Harary, Slany, and Verbitsky 2007), the dominating set problem and its variants (Chartrand et al. 1997; Bozeman et al. 2019; Ferrero et al. 2018) and the vertex cover problem (Horiyama et al. 2024). Also, several pencil/video puzzles such as SUDOKU and Picross 3D have been studied in the context of the pre-assignment for uniquification (Demaine et al. 2018; Kimura, Kamehashi, and Fujito 2018; Tjusila et al. 2024).

In this paper, we focus on the PRE-ASSIGNMENT FOR UNIQUIFICATION OF MINIMUM VERTEX COVER (PAU-VC) problem introduced by (Horiyama et al. 2024). A set $S$ of vertices in a graph $G$ is called a *vertex cover* of $G$ if $S$ meets all edges of $G$. The goal of this problem is to compute a minimum-cardinality vertex cover of a given graph. In the unique vertex cover problem, it is assured that an input graph has a unique minimum vertex cover. The formal definition of PAU-VC is the following.

PAU-VC
**Input :** A graph $G$
**Question :** Find a minimum set $S \subseteq V(G)$ such that there is a unique minimum vertex cover of $G$ containing $S$.

Notice that one can use an algorithm for PAU-VC to generate a graph with a unique solution for the vertex cover problem. Consider an arbitrary graph $G$ (possibly from a known benchmark dataset), and compute an optimal solution $S$ for PAU-VC on $G$. Since there is a unique minimum vertex cover of $G$ containing $S$, $G-S$ has a unique minimum vertex cover as well, where $G - S$ is the graph obtained from $G$ by removing all vertices of $S$ and their incident edges.

Although the pre-assignment for uniquification of the dominating set problem and its variants has been studied exten-

*Corresponding author

sively, little is known about PAU-VC, except for (Horiyama et al. 2024). More specifically, (Horiyama et al. 2024) proved that PAU-VC is $\Sigma_2^P$-complete on general graphs, and NP-complete on bipartite graphs. On the positive side, they provided an algorithm that runs in time $\mathcal{O}(2.1996^n)$ for general graphs, an algorithm that runs in time $\mathcal{O}(1.9181^n)$ for bipartite graphs, and an algorithm that runs in time $\mathcal{O}(1.4143^n)$ for trees, where $n$ denotes the number of vertices.

As PAU-VC is $\Sigma_2^P$-complete and NP-complete for general graphs and bipartite graphs, respectively, it is unlikely to admit polynomial-time algorithms for either general or bipartite graphs. However, the time complexity for trees remains an open question. In fact, (Horiyama et al. 2024) also mentioned this explicitly: *"Many readers might consider that* PAU-VC *for trees is likely solvable in polynomial time. On the other hand, not a few problems are intractable (e.g.,* NODE KAYLES*) in general, but the time complexity for trees still remains open, and only exponential-time algorithms are known. In the case of* PAU-VC*, no polynomial-time algorithm for trees is currently known."*

**Our results.** In this paper, we resolve this open problem by presenting a polynomial-time algorithm for PAU-VC on trees, which significantly improves the exponential-time algorithm by Horiyama et al. Moreover, we showed that it can be extended to classes of bounded *clique-width*. Clique-width is a graph parameter that measures the complexity of constructing a graph using a set of specific operations, including the creation of new vertices, disjoint union of graphs, relabeling of vertex labels, and connecting vertices based on their labels. Trees have clique-width at most 3 (Courcelle and Olariu 2000) and complete graphs have clique-width at most 2. A precise definition will be given in the clique-width section.

More precisely, we prove the following theorem. We say a problem is *fixed-parameter tractable* with parameter $k$ if it can be solved in time $f(k)n^{\mathcal{O}(1)}$, where $n$ denotes the input size and $f(\cdot)$ is a computable function.

**Theorem 1.** PAU-VC *is fixed-parameter tractable parameterized by clique-width.*

One may ask whether we can further obtain polynomial-time algorithms for PAU-VC on natural classes of graphs of unbounded clique-width. We investigate two such classes. *Split graphs* are graphs that can be partitioned into an independent set and a clique. *Unit interval graphs* are intersection graphs of intervals of the same length on the real line. It is known that split graphs have unbounded clique-width (Makowsky and Rotics 1999) and unit interval graphs have unbounded clique-width (Golumbic and Rotics 2000). Split graphs and unit interval graphs are well-known graph classes that have been widely studied (Corneil et al. 1995; Hell, Shamir, and Sharan 2001; Bertossi 1984). We prove that PAU-VC can be solved in linear time on both classes.

**Theorem 2.** PAU-VC *can be solved in linear time on unit interval graphs and split graphs.*

Note that the class of split graphs and the class of unit interval graphs are well-known subclasses of the class of *chordal graphs*. It would be interesting to determine whether PAU-VC can be solved in polynomial time on chordal graphs.

Proofs of statements marked with "$(*)$" are deferred to the appendix.

**Brief introduction on clique-width.** The notion of clique-width is closely related to the concept of *tree-width*. Tree-width is a well-studied graph parameter which measures how close a graph is to being a tree (Robertson and Seymour 2004). (Courcelle 1990) showed that every problem expressible in $\text{MSO}_2$-logic is fixed-parameter tractable when parameterized by the tree-width of a graph. However, classes of bounded tree-width must be sparse. To address this limitation, (Courcelle and Olariu 2000) introduced clique-width to extend properties of classes of bounded tree-width to dense graph classes, such as the class of complete graphs.

Every class of bounded tree-width has bounded clique-width (Courcelle and Olariu 2000; Corneil and Rotics 2005), but there are classes of bounded clique-width and unbounded tree-width, such as the class of complete graphs or complete bipartite graphs. (Courcelle, Makowsky, and Rotics 2000) showed that every problem expressible in $\text{MSO}_1$-logic is fixed-parameter tractable when parameterized by the clique-width of a graph. It is not difficult to see that PAU-VC cannot be expressible in $\text{MSO}_1$-logic, as we cannot represent the property that a set is a unique minimum vertex cover. So, the algorithmic meta theorem by Courcelle, Makowsky, and Rotics cannot be adapted for PAU-VC. The parameterized complexity of problems cannot be expressible by $\text{MSO}_1$-logic, such as HAMILTONAIN CYCLE and Graph Coloring, have been studied (Kobler and Rotics 2003; Fomin et al. 2010, 2014, 2019; Bergougnoux, Kanté, and Kwon 2020).

## Preliminary

For every positive integer $n$, let $[n]$ denote the set of positive integers at most $n$. All graphs in this paper are simple and finite. For a graph $G$ we denote by $V(G)$ and $E(G)$ the vertex set and edge set of $G$, respectively. For graphs $G$ and $H$, let $G \cup H$ be the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H)$.

Let $G$ be a graph. For a vertex $v$ of a graph $G$, let $N_G(v)$ denote the set of *neighbors* of $v$ in $G$. For $X \subseteq V(G)$, let $G[X]$ denote the subgraph of $G$ induced by $X$. We denote by $G - X$ the graph $G[V(G) \setminus X]$, and for a single vertex $x \in V(G)$, we use the shorthand '$G - x$' for '$G - \{x\}$'. For two sets $X, Y \subseteq V(G)$, let $G[X, Y]$ be the graph $(X \cup Y, \{xy \in E(G) : x \in X, y \in Y\})$.

A set $X \subseteq V(G)$ is a *clique* if any two vertices of $X$ are adjacent in $G$, and it is an *independent set* if any two vertices of $X$ are not adjacent in $G$.

## Trees

Before diving into our main theorems, we present an idea for having a simpler polynomial time algorithm for PAU-VC on trees. Recall that this significantly improves the best-known algorithm for this problem (Horiyama et al. 2024), which runs in exponential time. Let $G$ be a tree. We choose an arbitrary vertex as the root of $G$. For each node $v \in V(G)$, we use $G_v$ to denote the subtree of $G$ rooted at $v$.

For each vertex $v$, there are two types of vertex covers of $G_v$; one is a vertex cover of $G_v$ containing $v$ and the other is

a vertex cover of $G_v$ not containing $v$. We want to find a set $S$ which forces the number of minimum vertex covers of each type to satisfy a certain condition. This naturally suggests the following definition. For a function $\beta : \{0,1\} \to \{0,1,2\}$, a set $S \subseteq V(G_v)$ is a *$\beta$-set* in $G_v$ if the following hold:

- If $\beta(0) \in \{0,1\}$, then there is exactly $\beta(0)$ many minimum vertex covers of $G_v$ not containing $v$ and containing $S$.
- If $\beta(0) = 2$, then there are at least two minimum vertex covers of $G_v$ not containing $v$ and containing $S$.
- If $\beta(1) \in \{0,1\}$, then there is exactly $\beta(1)$ many minimum vertex covers of $G_v$ containing $v$ and containing $S$.
- If $\beta(1) = 2$, then there are at least two minimum vertex covers of $G_v$ containing $v$ and containing $S$.

We will recursively compute a minimum $\beta$-set in $G_v$ for every possible function $\beta$ and every vertex $v \in V$, if one exists.

It is not difficult to observe that if we have a minimum $\beta$-set of $G_r = G$ for every possible function $\beta$, then we can find an optimal solution of PAU-VC. That would be a minimum set among minimum $\beta$-sets of $G_r$ for which $\beta(0)+\beta(1) = 1$.

Therefore, it suffices to recursively compute a minimum $\beta$-set of $G_v$ for every vertex $v \in V$. The idea is straightforward. We need to propagate the information to children of $v$. Assume $\beta : \{0,1\} \to \{0,1,2\}$ is a given function. For example, if $\beta(0) = 1$, then the $\beta$-set in $G_v$ should force a unique minimum vertex cover of $G_v$ not containing $v$. Then for each child $w$ of $v$, we have to determine a set forcing a unique minimum vertex cover of $G_w$ that contains $w$. This suggests how to split $\beta$ into functions $\beta_w$ for each child $w$, and we can find the corresponding $\beta$-set by taking the union of $\beta_w$-sets for children $w$ of $v$.

This idea is generalized into graphs of bounded clique-width in the next section. We will provide the dynamic programming algorithm and prove the correctness.

## Graphs of Bounded Clique-width

In this section, we prove Theorem 1. Before we describe our strategy, we provide a formal definition of the clique-width and introduce some necessary notations. Let $k$ be a positive integer. A *$k$-labeled graph* is a pair $(G, \mathrm{lab}_G)$ of a graph $G$ and a function $\mathrm{lab}_G : V(G) \to [k]$, called the *labeling function*. We denote by $\mathrm{lab}_G^{-1}(i)$ the set of vertices in $G$ with label $i$.

### Definitions of Clique-Width and NLC-Width

We first define the *clique-width* of graphs. For a $k$-labeled graph $(G, \mathrm{lab}_G)$ and $i, j \in [k]$ with $i \neq j$, let $\eta_{i,j}(G, \mathrm{lab}_G)$ be the $k$-labeled graph obtained from $(G, \mathrm{lab}_G)$ by adding an edge between every vertex of label $i$ and every vertex of label $j$, and let $\rho_{i\to j}(G, \mathrm{lab}_G)$ be the $k$-labeled graph obtained from $(G, \mathrm{lab}_G)$ by relabeling every vertex of $i$ to $j$. For two vertex-disjoint $k$-labeled graphs $(G, \mathrm{lab}_G)$ and $(H, \mathrm{lab}_H)$, let $(G, \mathrm{lab}_G) \oplus (H, \mathrm{lab}_H)$ be the disjoint union of them.

The class $\mathrm{CW}_k$ of $k$-labeled graphs is recursively defined as follows. (1) The single vertex graph $i(x)$, with a vertex $x$ labeled with $i \in [k]$, is in $\mathrm{CW}_k$. (2) Let $(G, \mathrm{lab}_G)$ and $(H, \mathrm{lab}_H)$ be two vertex-disjoint $k$-labeled graphs in $\mathrm{CW}_k$. Then $(G, \mathrm{lab}_G) \oplus (H, \mathrm{lab}_H) \in \mathrm{CW}_k$. (3) Let $(G, \mathrm{lab}_G) \in \mathrm{CW}_k$ and $i, j \in [k]$ with $i \neq j$. Then $\eta_{i,j}(G, \mathrm{lab}_G) \in \mathrm{CW}_k$. (4) Let $(G, \mathrm{lab}_G) \in \mathrm{CW}_k$ and $i, j \in [k]$. Then $\rho_{i\to j}(G, \mathrm{lab}_G) \in \mathrm{CW}_k$. A *clique-width $k$-expression* is a finite term built with the four operations above and using at most $k$ labels. The *clique-width* of a graph, denoted by $\mathrm{cw}(G)$, is the minimum $k$ such that $(G, \mathrm{lab}_G) \in \mathrm{CW}_k$ for some labeling $\mathrm{lab}_G$.

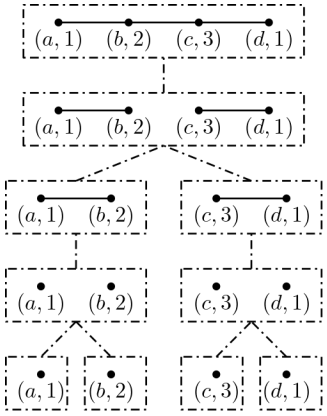


Figure 1: Illustration of a clique-width 3-expression of $P_4$.

For example,

$$\eta_{2,3}\Big(\eta_{1,2}(1(a) \oplus 2(b)) \oplus \eta_{1,3}(3(c) \oplus 1(d)))\Big)$$

is a clique-width 3-expression of a path $P_4$ on 4 vertices. See Figure 1. Thus, $P_4$ has clique-width at most 3.

Now, we define the *NLC-width* of graphs introduced by (Wanke 1994). (Johansson 1998) showed that every graph of clique-width at most $k$ has NLC-width at most $k$, and one can in polynomial time transform an expression for clique-width to an expression for NLC-width. For two vertex-disjoint $k$-labeled graphs $(G, \mathrm{lab}_G)$ and $(H, \mathrm{lab}_H)$ and a set $M \subseteq [k]^2$ of label pairs, we define $(G, \mathrm{lab}_G) \times_M (H, \mathrm{lab}_H) := ((V', E'), \mathrm{lab}')$ where

- $V' = V(G) \cup V(H)$,
- $E' = E(G) \cup E(H) \cup \{uv : u \in V(G), v \in V(H), (\mathrm{lab}_G(u), \mathrm{lab}_H(v)) \in M\}$,
- $\mathrm{lab}'(u) = \mathrm{lab}_G(u)$ if $u \in V(G)$ and $\mathrm{lab}'(u) = \mathrm{lab}_H(u)$ otherwise.

In other words, $(G, \mathrm{lab}_G) \times_M (H, \mathrm{lab}_H)$ is obtained from the disjoint union of $(G, \mathrm{lab}_G)$ and $(H, \mathrm{lab}_H)$ by, for every $(i, j) \in M$, adding all edges between vertices of label $i$ in $G$ and vertices of label $j$ in $H$. For a $k$-labeled graph $(G, \mathrm{lab}_G)$ and a function $R : [k] \to [k]$, let $\rho_R(G, \mathrm{lab}_G) = (G, \mathrm{lab}')$ where $\mathrm{lab}'(u) = R(\mathrm{lab}_G(u))$ for all $u \in V(G)$.

The class $\mathrm{NLC}_k$ of $k$-labeled graphs is recursively defined as follows. (1) The single vertex graph $i(x)$, with a vertex $x$ labeled with $i \in [k]$, is in $\mathrm{NLC}_k$. (2) Let $(G, \mathrm{lab}_G) \in \mathrm{NLC}_k$ and $R : [k] \to [k]$ be a function. Then $\rho_R(G, \mathrm{lab}) \in \mathrm{NLC}_k$. (3) Let $(G, \mathrm{lab}_G)$ and $(H, \mathrm{lab}_H)$ be

two vertex-disjoint labeled graphs in $\mathrm{NLC}_k$, and $M \subseteq [k]^2$. Then $(G, \mathrm{lab}_G) \times_M (H, \mathrm{lab}_H) \in \mathrm{NLC}_k$.

An *NLC-width $k$-expression* is a finite term built with the three operations above and using at most $k$ labels. The *NLC-width* of a graph $G$, denoted by $\mathrm{nlcw}(G)$, is the minimum $k$ such that $(G, \mathrm{lab}_G) \in \mathrm{NLC}_k$ for some labeling $\mathrm{lab}_G$.

**Theorem 3** ((Johansson 1998)). *Let $k$ be a positive integer. Every graph of clique-width at most $k$ has NLC-width at most $k$, and one can in polynomial time transform a clique-width $k$-expression to an NLC-width $k$-expression.*

We remark about algorithms to find a clique-width expression when it is not given. (Fellows et al. 2006) proved that computing clique-width is NP-hard. (Oum and Seymour 2006) first obtained an approximation algorithm that computes a clique-width $(2^{3k+2}-1)$-expression of a given graph $G$ of clique-width at most $k$, which runs in time $\mathcal{O}(8^k n^9 \log n)$. (Oum 2009) later improved this by providing two algorithms; one is an algorithm that computes a clique-width $(8^k-1)$-expression in time $\mathcal{O}(g(k) \cdot n^3)$ for some function $g$, and the other one is an algorithm that computes a clique-width $(2^{3k+2}-1)$-expression in time $\mathcal{O}(8^k n^4)$. We may use one of these algorithms to produce a clique-width expression, when it is not given as input.

## FPT Algorithm Parameterized by Clique-Width

Now we are ready to prove Theorem 1. Let $(H, \mathrm{lab}_H)$ be a $k$-labeled graph. For a set $X$ of vertices in $H$, we denote by $\mathrm{full}_H(X)$ the set of integers $i \in [k]$ where $\mathrm{lab}_H^{-1}(i) \subseteq X$. For $I \subseteq [k]$, a set $T \subseteq V(H)$ is a *minimum vertex cover of $H$ with respect to $I$* if it is a minimum set among all vertex covers $X$ of $H$ with $\mathrm{full}_H(X) = I$. Note that $T$ is not necessarily a minimum vertex cover of $H$. Let $\mu_H(I)$ be the size of a minimum vertex cover of $H$ with respect to $I$.

Assume that $(F, \mathrm{lab}_F) = (G, \mathrm{lab}_G) \times_M (H, \mathrm{lab}_H)$ for some $k$-labeled graphs $(G, \mathrm{lab}_G)$, $(H, \mathrm{lab}_H)$, and $M \subseteq [k]^2$. Observe that for every $(i,j) \in M$, every vertex cover of $F$ either contains all vertices of $\mathrm{lab}_G^{-1}(i)$ or contains all vertices of $\mathrm{lab}_H^{-1}(j)$. Thus, in each side, it is necessary to consider vertex covers that fully contain vertex sets of certain labels. This is the reason why we define minimum vertex covers with respect to $I \subseteq [k]$.

Now, to find sets $S \subseteq V(F)$ that force to have a unique minimum vertex cover, in each of $G$ and $H$, we need to know whether a set forces to have a unique minimum vertex cover with respect to some $I \subseteq [k]$. For each $I \subseteq [k]$, we need to distinguish three statuses: (1) $S$ does not force any minimum vertex cover with respect to $I$, (2) $S$ forces a unique minimum vertex cover with respect to $I$, or (3) $S$ forces at least two minimum vertex covers with respect to $I$. This property will be captured by the notion of characteristic, defined below.

A function $\beta : 2^{[k]} \to \{0,1,2\}$ is the *characteristic* of a set $S \subseteq V(H)$ in $H$ if for every $J \subseteq [k]$,

- if $\beta(J) \in \{0,1\}$, then there is exactly $\beta(J)$ many minimum vertex covers of $H$ with respect to $J$ and containing $S$, and
- if $\beta(J) = 2$, then there are at least two minimum vertex covers of $H$ with respect to $J$ and containing $S$.

Such a set $S \subseteq V(H)$ is called a *$\beta$-set in $H$*. Let $\Pi(H)$ be the collection of functions $\beta : 2^{[k]} \to \{0,1,2\}$ such that there is a $\beta$-set in $H$.

In the following lemma, we explain how we can solve PAU-VC on a $k$-labeled graph $H$ if we know the set $\Pi(H)$ and the function $\mu_H$ and a collection of minimum $\beta$-sets for each $\beta \in \Pi(H)$.

**Lemma 1.** *Let $k$ be a positive integer. Given a $k$-labeled graph $(G, \mathrm{lab}_G)$ with $\Pi(G)$, $\mu_G$ and a collection of minimum $\beta$-sets for each $\beta \in \Pi(G)$, one can solve* PAU-VC *for $G$ in time $\mathcal{O}(3^{2^k}|V(G)|)$.*

*Proof.* Let $\mu = \min_{I \subseteq [k]} \mu_G(I)$, and $\Gamma = \{J \subseteq [k] : \mu_G(J) = \mu\}$. Then $\mu$ is the size of a minimum vertex cover of $G$. We say that a function $\beta : 2^{[k]} \to \{0,1,2\}$ is *valid* if $\sum_{J \in \Gamma} \beta(J) = 1$. A $\beta$-set with a valid function $\beta$ in $\Pi(G)$ is a set forcing a unique minimum vertex set in $G$. Thus, the minimum $\beta$-set with a valid function $\beta$ in $\Pi(G)$ is a required solution for PAU-VC. □

By Lemma 1, it is sufficient to compute $\Pi(H)$ and $\mu_H$ and a collection of minimum $\beta$-sets. We will compute them in a bottom-up way, along a given NLC-width $k$-expression.

In the next lemma, we describe how to merge information for $(G, \mathrm{lab}_G)$ and $(H, \mathrm{lab}_H)$ to obtain information for $(G, \mathrm{lab}_G) \times_M (H, \mathrm{lab}_H)$.

**Lemma 2.** *Let $k$ be a positive integer, and let $(G, \mathrm{lab}_G)$ and $(H, \mathrm{lab}_H)$ be vertex-disjoint $k$-labeled non-empty graphs. Let $M \subseteq [k]^2$ and let $(F, \mathrm{lab}_F) = (G, \mathrm{lab}_G) \times_M (H, \mathrm{lab}_H)$.*

*Given $\Pi(G), \Pi(H)$ and $\mu_G, \mu_H$ and a collection $\mathcal{I}_G$ of minimum $\beta$-sets for $\beta \in \Pi(G)$ and a collection $\mathcal{I}_H$ of minimum $\beta$-sets for $\beta \in \Pi(H)$, one can compute $\Pi(F)$, $\mu_F$ and a collection $\mathcal{I}_F$ of minimum $\beta$-sets for $\beta \in \Pi(F)$ in time $\mathcal{O}(27^{2^k} \cdot |V(G)|)$.*

*Proof.* We construct an auxiliary bipartite graph $Q$ with bipartition $(\{a_i : i \in [k]\}, \{b_i : i \in [k]\})$ such that $a_i$ is adjacent to $b_j$ if and only if $(i,j) \in M$. Let $A = \{a_i : i \in [k]\}$, $B = \{b_i : i \in [k]\}$, and let $g(a_i) = g(b_i) = i$ for all $i \in [k]$. Let $\mathcal{Z}$ be the collection of all vertex covers of $Q$. Note that the number of all vertex covers of $Q$ is at most $2^{2k}$.

We first compute $\mu_F(I)$ for each $I \subseteq [k]$, which is the size of a minimum vertex cover of $F$ with respect to $I$. Note that for any $(i,j) \in M$, every vertex cover of $F$ contains either all vertices of $\mathrm{lab}_G^{-1}(i)$ or all vertices of $\mathrm{lab}_H^{-1}(j)$. We guess a vertex cover of $Q$ corresponding to parts whose all vertices are contained in a vertex cover of $F$.

We construct a function $\mu^*$ as below.

- Let $I \subseteq [k]$. For each $Z \in \mathcal{Z}$ with $(g(Z \cap A) \cap g(Z \cap B)) \setminus I = \emptyset$, let $I_G = I \cup g(Z \cap A)$ and $I_H = I \cup g(Z \cap B)$, and let $\alpha(Z) := \mu_G(I_G) + \mu_H(I_H)$.
- We define $\mu^*(I)$ as the minimum such $\alpha(Z)$ over all $Z \in \mathcal{Z}$ with $(g(Z \cap A) \cap g(Z \cap B)) \setminus I = \emptyset$. Note that such a set $Z$ exists as $A$ is such a vertex cover.

**Claim 1** (∗). *The above procedure correctly computes $\mu_F$, that is, $\mu^*(I) = \mu_F(I)$ for every $I \subseteq [k]$.*

Since the number of vertex covers of $Q$ is at most $2^{2k}$, for each $I \subseteq [k]$, we can determine $\mu^*(I)$ in time $\mathcal{O}(4^k)$. So, we can output $\mu^*$ in time $\mathcal{O}(4^k)$.

Now, we compute $\Pi(F)$ and $\mathcal{I}_F$. We construct sets $\Pi^*$ and $\mathcal{I}^*$ and will show that $\Pi^* = \Pi(F)$ and $\mathcal{I}^*$ is a collection of minimum $\beta$-sets for $\beta \in \Pi(F)$. For a function $\beta : 2^{[k]} \to \{0, 1, 2\}$, we need to determine whether there is a $\beta$-set in $F$. Let $\beta : 2^{[k]} \to \{0, 1, 2\}$ be a function.

1. Let $I \subseteq [k]$. We say that a pair $(I_G, I_H)$ of subsets of $[k]$ is a *split of $I$ with respect to $M$* if
   - $I_G \cap I_H = I$, and
   - for every $(i, j) \in M$, $i \in I_G$ or $j \in I_H$.

   A split $(I_G, I_H)$ of $I$ is *proper* if $\mu_G(I_G) + \mu_H(I_H) = \mu_F(I)$.
2. A pair $(\beta_G, \beta_H)$ of functions $\beta_G, \beta_H : 2^{[k]} \to \{0, 1, 2\}$ is *legitimate for $\beta$* if for all subsets $I \subseteq [k]$, we have

$$\beta(I) = \min \left( 2, \sum_{\substack{(I_G, I_H): \\ \text{proper split of } I}} \Big( \beta_G(I_G) \times \beta_H(I_H) \Big) \right).$$

3. Assume there is a legitimate pair $(\beta_G, \beta_H)$ for $\beta$ where $\beta_G \in \Pi(G)$ and $\beta_H \in \Pi(H)$ and $S_G$ is a minimum $\beta_G$-set in $\mathcal{I}_G$ and $S_H$ is a minimum $\beta_H$-set in $\mathcal{I}_H$. Then we add $\beta$ to $\Pi^*$ and add $S_G \cup S_H$ to $\mathcal{I}^*$. Otherwise, we do not add.

**Claim 2** ($*$)**.** *The above procedure correctly computes $\Pi(F)$, that is, $\Pi^* = \Pi(F)$. Also, $\mathcal{I}^*$ is a collection of $\beta$-sets for $\beta \in \Pi(F)$.*

Observe that the number of possible functions $\beta : 2^{[k]} \to \{0, 1, 2\}$ is at most $3^{2^k}$. Let $\beta$ be such a function. For each $I \subseteq [k]$, there are at most $3^{k-|I|} \leq 3^k$ splits of $I$ with respect to $M$. Thus, for a fixed pair $(\beta_G, \beta_H)$ of functions, one can test whether $(\beta_G, \beta_H)$ is legitimate for $\beta$ in time $\mathcal{O}(2^k \cdot 3^k)$. Thus, we can determine $\Pi^*$ and $\mathcal{I}^*$ in time $\mathcal{O}(3^{3 \cdot 2^k} \cdot 2^k \cdot 3^k \cdot |V(G)|) = \mathcal{O}(27^{2^k}|V(G)|)$. This concludes the proof. □

**Lemma 3** ($*$)**.** *Let $k$ be a positive integer, and let $(G, \mathrm{lab}_G)$ be a $k$-labeled graph. Let $R : [k] \to [k]$ be a function and let $(F, \mathrm{lab}_F) = \rho_R(G, \mathrm{lab}_G)$.*

*Given $\Pi(G)$, $\mu_G$, and a collection $\mathcal{I}_G$ of minimum $\beta$-sets for $\beta \in \Pi(G)$, one can compute $\Pi(F)$, $\mu_F$ and a collection $\mathcal{I}_F$ of minimum $\beta$-sets for $\beta \in \Pi(F)$ in time $\mathcal{O}(9^{2^k} \cdot |V(G)|)$.*

Now, we are ready to prove Theorem 1.

*Proof.* (Proof of Theorem 1) Using an algorithm by (Oum 2009), we can compute a clique-width $(2^{3t+2}-1)$-expression of a graph of clique-width $t$ in time $\mathcal{O}(8^t|V(G)|^4)$ as explained in the preliminary section. In the rest, we discuss how to obtain an algorithm if a clique-width expression is given.

Let $G$ be a graph and assume that its clique-width $k$-expression is given. By Theorem 3, we can transform it into an NLC-width $k$-expression $\phi$ in polynomial time.

We design a bottom-up dynamic programming along the NLC-width $k$-expression. At each $k$-labeled graph $(F, \mathrm{lab}_F)$ arising in $\phi$, we compute sets $\Pi(F)$, $\mu_F$, and a collection $\mathcal{I}_F$ of minimum $\beta$-sets for $\beta \in \Pi(F)$ as follows.

1. Assume $(F, \mathrm{lab}_F) = i(x)$, that is, $F$ is a graph on a vertex $x$ with label $i$.
   - Observe that $\mu_F(\{i\}) = 1$ because $\{x\}$ is the unique minimum vertex cover of $F$ with respect to $\{x\}$. Also, $\mu_F(\emptyset) = 1$ because $\emptyset$ is the unique minimum vertex cover of $F$ with respect to $\emptyset$. For other subsets $I$ of $[k]$, $\mu_F(I) = 0$, as there is no vertex cover of $F$ with respect to $I$.
   - Note that the empty set has the characteristic $\beta_0$ where $\beta_0(J) = 1$ for $J = \{i\}$ or $\emptyset$, and $\beta_0(J) = 0$ otherwise. The set $\{x\}$ has characteristic $\beta_1$ where $\beta_1(J) = 1$ for $J = \{i\}$, and $\beta_1(J) = 0$ otherwise. Let $\Pi(F) = \{\beta_0, \beta_1\}$. We store the empty set as a minimum $\beta_0$-set, and $\{x\}$ as a minimum $\beta_1$-set.
   - These can be computed in time $\mathcal{O}(k)$.
2. Assume that $(F, \mathrm{lab}_F) = \rho_R(F_1, \mathrm{lab}_1)$ for some function $R : [k] \to [k]$. By Lemma 3, we can in time $\mathcal{O}(9^{2^k} \cdot |V(G)|)$ compute $\Pi(F)$, $\mu_F$, and a collection $\mathcal{I}_F$ of minimum $\beta$-sets for $\beta \in \Pi(F)$.
3. Assume that $(F, \mathrm{lab}_F) = (F_1, \mathrm{lab}_1) \times_M (F_2, \mathrm{lab}_2)$ for some $M \subseteq [k]^2$. By Lemma 2, we can in time $\mathcal{O}(27^{2^k} \cdot |V(F)|)$ compute $\Pi(F)$, $\mu_F$, and a collection $\mathcal{I}_F$ of minimum $\beta$-sets for $\beta \in \Pi(F)$.

At the end, by Lemma 1, we can solve PAU-VC in time $\mathcal{O}(3^{2^k}|V(G)|)$. Note that there are at most $\mathcal{O}(k^2|V(G)|)$ operations in the NLC-width $k$-expression. Thus, in total, we can solve PAU-VC in time $\mathcal{O}(27^{2^k}|V(G)|^2)$. □

## Unit Interval Graphs

In this section, we give a linear time algorithm of PAU-VC for a unit interval graph $G$. A unit interval graph $G$ is a graph that has a set $\mathcal{I}$ of intervals of length one on the real line so that $G$ is an intersection graph of $\mathcal{I}$, refer to Figure 2.

**Theorem 4.** PAU-VC *can be solved in linear time for a unit interval graph.*

*Proof.* Let $G$ be a given unit interval graph. We can find a set $\mathcal{I}$ of unit intervals representing $G$ in linear time (Corneil et al. 1995). Furthermore, the obtained $\mathcal{I}$ is sorted by the left end points. For clarity, we refer to the intervals in $\mathcal{I}$ as the vertices of $G$. By perturbing if necessary, we may assume that all intervals are pairwise distinct.

First, we find a maximum independent set $\{\tilde{I}_1, \ldots, \tilde{I}_m\}$ of $G$ as follows: $\tilde{I}_1$ is the leftmost interval in $\mathcal{I}$ and $\tilde{I}_{i+1}$ is the leftmost interval in $\mathcal{I}$ disjoint to $\tilde{I}_j$ for all $j \in [i]$. It is easy to see that the obtained set is a maximum independent set of $G$. Then, for each $\tilde{I}_i$, let $\mathcal{I}_i$ be the set of intervals in $\mathcal{I}$ that start after $\tilde{I}_i$ and intersect with $\tilde{I}_i$, along with $\tilde{I}_i$ itself. Then $\{\mathcal{I}_1, \ldots, \mathcal{I}_m\}$ gives a partition of $\mathcal{I}$, and each $\mathcal{I}_i$ forms a clique in $G$. Note that if two intervals $I \in \mathcal{I}_i$ and $J \in \mathcal{I}_j$ are intersecting, then $|i - j| \leq 1$ since every interval has a unit length.

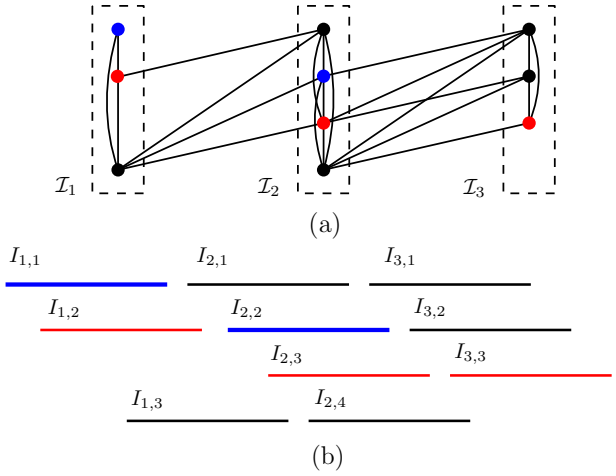


Figure 2: Illustration of the algorithm for a unit interval graph $G$. Figures (a) and (b) represent the unit interval graph $G$ and its representation, respectively. The algorithm returns $\{I_{1,1}, I_{2,2}\}$ as an optimal solution of PAU-VC of $G$, where $\{I_{1,1}, I_{2,2}\} = S[3,3]$ as $S[1,2] = \{I_{1,1}\}, A_{1,2,3} = \{I_{2,2}\}$, and $A_{2,3,3} = \emptyset$. Precisely, $S[3,3] = S[1,2] \cup A_{1,2,3} \cup A_{2,3,3}$.

**Claim 3.** *Every minimum vertex cover of $G$ excludes exactly one vertex in $\mathcal{I}_i$ for each $i \in [m]$.*

*Proof.* It is well known that the complement of a maximum independent set is a minimum vertex cover, and vice versa. Since $m$ is the size of the maximum independent set of $G$, every minimum vertex cover should exclude $m$ vertices. Note that each $\mathcal{I}_i$ forms a clique in $G$, it cannot exclude more than one vertex from the same $\mathcal{I}_i$. Thus, exactly one vertex for each $\mathcal{I}_i$ is excluded. $\diamondsuit$

Now we describe a dynamic programming to solve PAU-VC for $G$.

Let $i \in [m]$ and $j \in [|\mathcal{I}_i|]$. Let $I_{i,j}$ be the $j$th leftmost interval in $\mathcal{I}_i$, and let $G_{i,j}$ be the subgraph of $G$ induced by the intervals starting before $I_{i,j}$ along with $I_{i,j}$.

For $j \in [|\mathcal{I}_1|]$, let $A_{1,j} := \{I_{1,i} : i < j\}$. For $2 \leq i \leq m-1$ and $a \in [|\mathcal{I}_i|]$ and $b \in [|\mathcal{I}_{i+1}|]$, let

$$A_{i,a,b} := \{I_{i,x} : a < x \text{ and } I_{i,x} \cap I_{i+1,b} = \emptyset\} \cup \{I_{i+1,y} : y < b \text{ and } I_{i+1,y} \cap I_{i,a} = \emptyset\},$$

Intuitively, $A_{1,j}$ and $A_{i,a,b}$ are sets of vertices that have to be included in a set forcing a unique minimum vertex cover $S$ when we want that $I_{i,a}$ and $I_{i+1,b}$ are not in $S$.

Now, for each $I_{i,j} \in \mathcal{I}$, we denote $S[i,j]$ as the smallest vertex set in $G_{i,j}$ that forces the unique minimum vertex cover in $G_{i,j}$ and the vertex cover excludes $I_{i,j}$.

We compute $S[i,j]$ in a lexicographic order. As the base case, we set $S[1,j] := A_{1,j}$ for each $j \in [|\mathcal{I}_1|]$. By assuming that every $S[i,j']$ has been computed for $j' \in [|\mathcal{I}_i|]$, we set $S[i+1,j]$ as the smallest vertex set among

$$S[i,j'] \cup A_{i,j',j}$$

where $I_{i,j'} \in \mathcal{I}_i$ is disjoint from $I_{i+1,j}$. Note that $S[i+1,j]$ is well-defined, because $I_{i,1} = \tilde{I}_i$ is disjoint from $I_{i+1,1} = \tilde{I}_{i+1}$ along with $I_{i+1,j}$.

**Claim 4.** *For every $i \in [m]$ and $j \in [|\mathcal{I}_i|]$, $S[i,j]$ is a smallest vertex set in $G_{i,j}$ that forces a unique minimum vertex cover in $G_{i,j}$ and the vertex cover excludes $I_{i,j}$.*

*Proof.* We prove this by induction on $i+j$. First, assume that $i = 1$. Then $G_{1,j}$ is a complete graph. Thus, there is only one vertex cover excluding $I_{1,j}$, namely $V(G_{1,j}) \setminus \{I_{1,j}\}$. If we do not preassign a vertex of $V(G_{1,j}) \setminus \{I_{1,j}\}$, we may take $I_{i,j}$ instead of this vertex, to make another minimum vertex cover of $G_{1,j}$. Thus, $S[1,j]$ should be exactly $V(G_{1,j}) \setminus \{I_{1,j}\}$.

Suppose that $i \geq 2$. Let $T$ be a minimum vertex set in $G_{i,j}$ that forces a unique minimum vertex cover in $G_{i,j}$ and the vertex cover excludes $I_{i,j}$. We will show that $T$ is of the form $S[i-1,j'] \cup A_{i-1,j',j}$ in the definition. Let $U$ be the unique vertex cover in $G_{i,j}$ containing $T$, and let $W = V(G_{i,j}) \setminus U$. Observe that $W$ contains exactly one vertex from each of $\mathcal{I}_1, \ldots, \mathcal{I}_i$. Let $j_1, \ldots, j_{i-1}$ be integers such that $W = \{I_{1,j_1}, I_{2,j_2}, \ldots, I_{i-1,j_{i-1}}, I_{i,j}\}$.

We claim that

$$A^* := A_{1,j_1} \cup \left( \bigcup_{x \in [i-1]} A_{x,j_x,j_{x+1}} \right) \subseteq T.$$

Suppose for contradiction that this is not true. First assume that $A_{1,j_1}$ contains a vertex $I$ that is not in $T$. Then $(U \setminus \{I\}) \cup \{I_{1,j_1}\}$ is also a minimum vertex cover of $G_{i,j}$ containing $T$, a contradiction.

We assume that for some $x \in [i-1]$, $A_{x,j_x,j_{x+1}}$ contains a vertex $I$ that is not in $T$. If

$$I \in \{I_{x,z} : j_x < z \text{ and } I_{x,z} \cap I_{x+1,j_{x+1}} = \emptyset\},$$

then $(U \setminus \{I\}) \cup \{I_{x,j_x}\}$ is a minimum vertex cover of $G_{i,j}$, a contradiction. Otherwise, if

$$I \in \{I_{x+1,z} : z < j_{x+1} \text{ and } I_{x+1,z} \cap I_{x,j_x} = \emptyset\},$$

then $(U \setminus \{I\}) \cup \{I_{x+1,j_{x+1}}\}$ is a minimum vertex cover of $G_{i,j}$, a contradiction. Therefore, $A^* \subseteq T$.

Now, we verify that $A^*$ already forces that $U$ is a unique minimum vertex cover containing $A^*$. Suppose there is another minimum vertex cover $U'$ containing $A^*$. As $U' \neq U$, $U'$ does not contain a vertex of $U \setminus A^*$. Then $G_{i,j} - U'$ cannot contain an independent set of size $i$, a contradiction.

By our construction, $T$ is $S[i-1,j'] \cup A_{i-1,j',j}$ for some $j'$ where $I_{i-1,j'} \in \mathcal{I}_{i-1}$ disjoint from $I_{i,j}$. On the other hand, we compute $S[i,j]$ as a smallest vertex set among all possible $S[i-1,j'] \cup A_{i-1,j',j}$. Thus, $S[i,j]$ is a smallest vertex set in $G_{i,j}$ that forces a unique minimum vertex cover in $G_{i,j}$ and the vertex cover excludes $I_{i,j}$. $\diamondsuit$

Furthermore, the smallest vertex set among

$$S[m,j] \cup \{I_{m,k} \in \mathcal{I}_m : j < k\}$$

for $j \in [|\mathcal{I}_m|]$ is an optimal solution of PAU-VC for $G$.

This algorithm returns a solution in polynomial time. In the following, we slightly modify it as a linear time algorithm.

**Linear time algorithm.** We compute the interval set $\mathcal{I}$ corresponds to the given unit interval graph $G$, and its decomposition $\mathcal{I}_1, \ldots, \mathcal{I}_m$ as described above. It takes a linear time. To obtain a linear time algorithm for PAU-VC, we compute and store the size $s[i+1, j]$ of the set $S[i+1, j]$ for each $(i+1, j)$ with $i \in [m-1]$ and $j \in [|\mathcal{I}_{i+1}|]$ instead of explicitly constructing the set $S[i+1, j]$. Since the set $S[i+1, j]$ is the union of disjoint sets $S[i, j']$ and $A_{i,j',j}$, we can compute $s[i+1, j]$ without explicitly constructing $S[i, j]$. Furthermore, we also store the index $j' \in [|\mathcal{I}_i|]$ at the pair $(i+1, j)$ so that $S[i+1, j] = S[i, j'] \cup A_{i,j',j}$. We can compute all values of $s[\cdot, \cdot]$ in $\mathcal{O}(|\mathcal{I}|)$ time.

**Claim 5.** *We can compute all $s[\cdot, \cdot]$ in $\mathcal{O}(|\mathcal{I}|)$ time.*

*Proof.* We set $s[1, j] = j - 1$ by the definition. For an index $i \in [m-1]$, we assume that every $s[i, \cdot]$ is computed already, and describe how to compute all $s[i+1, \cdot]$ in $\mathcal{O}(|\mathcal{I}_i \cup \mathcal{I}_{i+1}|)$ time which directly implies the claim. For this, we first compute an index $k(j) \in [|\mathcal{I}_i|]$ for each $j \in [|\mathcal{I}_{i+1}|]$ so that the interval $I_{i,k(j)}$ is the rightmost interval in $\mathcal{I}_i$ disjoint from $I_{i+1,j}$. Since $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$ are sorted, the indices $k(j)$'s are monotonic increasing. Furthermore, we can compute all $k(j)$'s in $\mathcal{O}(|\mathcal{I}_i \cup \mathcal{I}_{i+1}|)$ time. For clarity, we set $k(0) = 0$ in the following. Note that $I_{i,j'} \cap I_{i+1,j} = \emptyset$ if and only if $j' \leq k(j)$ for $j \in [|\mathcal{I}_{i+1}|]$.

Recall that $s[i+1, j]$ is the smallest value among $s[i, j'] + |A_{i,j',j}|$ with $j' \leq k(j)$. Furthermore, if $j' \leq k(j-1)$, then $A_{i,j',j}$ is same with

$$A_{i,j',j-1} \cup \{I_{i,x} : k(j-1) < x \leq k(j)\} \cup \{I_{i+1,j-1}\}.$$

If an index $j' \leq k(j)$ gives the smallest set $S[i, j'] \cup A_{i,j',j}$, then either $j' > k(j-1)$ or it gives a smallest set among $S[i, j'] \cup A_{i,j',j-1}$. Thus, we can compute the size $s[i+1, j]$ of $S[i+1, j]$ by comparing $k(j) - k(j-1) + 1$ values. Totally, computing all $s[i+1, \cdot]$ requires $\mathcal{O}(|\mathcal{I}_i \cup \mathcal{I}_{i+1}|)$ time, and thus, computing all $s[\cdot, \cdot]$ takes $\mathcal{O}(|\mathcal{I}|)$ time. ◊

After we compute every $s[\cdot, \cdot]$, we find out the index $j^* \in [|\mathcal{I}_m|]$ minimizing the value $s[m, j^*] + |\mathcal{I}_m| - j^*$. Then we define $j_m = j^*$ and $j_i$ as the index with $S[i+1, j_{i+1}] = S[i, j_i] \cup A_{i,j_i,j_{i+1}}$ for $i \in [m-1]$. By the definition of the sets $S[\cdot, \cdot]$, the following set is same with $S[m, j^*] \cup \{I_{m,k} \in \mathcal{I}_m : j_m < k\}$ that is an optimal solution of PAU-VC

$$A_{1,j_1} \cup \left( \bigcup_{i \in [m-1]} A_{i,j_i,j_{i+1}} \right) \cup \{I_{m,k} \in \mathcal{I}_m : j_m < k\}.$$

In conclusion, our algorithm returns a solution of PAU-VC for $G$ in linear time, and thus, Theorem 4 holds. □

## Split Graphs

In this section, we describe a linear time algorithm for split graphs. A split graph $G$ is a graph in which there exist disjoint subsets $A, B \subseteq V(G)$ such that $V(G) = A \cup B$, $A$ is a clique and $B$ is an independent set.

**Theorem 5.** PAU-VC *can be solved in linear time for a split graph.*

*Proof.* Let $G$ be a given split graph, of which the vertex set consists of a clique $A$ and an independent set $B$. Observe that a minimum vertex set excludes at most one vertex from $A$, and furthermore, $A$ is a vertex cover of $G$. We claim that we can safely remove all vertices in $A$ from $G$ which has at least two adjacent vertices in $B$ and also remove isolated vertices.

**Claim 6** (∗)**.** *If $v \in A$ has at least two adjacent vertices in $B$, then every minimum vertex cover of $G$ includes $v$.*

In the following, suppose that every vertex in $A$ has at most one adjacent vertex in $B$ and there is no isolated vertex. Let $A_0$ be the set of vertices in $A$ which has no adjacent vertex in $B$. We first consider the case that $A_0$ is not empty. In such a case, a minimum vertex cover of $G$ has size $|A| - 1$, furthermore, $A \setminus \{v\}$ is a minimum vertex cover of $G$ for every vertex $v \in A_0$. Thus, $A_0 \setminus \{v\}$ is an optimal solution of PAU-VC for an arbitrary vertex $v \in A_0$.

In the following, we consider the other case that $A_0 = \emptyset$. In this case, the size of a minimum vertex cover of $G$ is $|A|$.

For each $a \in A$, let $v_a$ be the vertex of $A$ that is adjacent to $a$. Observe that for each $a \in A$, $(A \setminus \{a\}) \cup \{v_a\}$ is also a minimum vertex cover. We find a vertex $b^* \in B$ minimizing $N_G(b^*)$, and return $(N_G(b^*) \setminus \{v\}) \cup \{b^*\}$ as a solution of PAU-VC, where $v$ is an arbitrary vertex in $N_G(b^*)$.

**Claim 7** (∗)**.** *Let $b^* \in B$ such that $|N_G(b^*)|$ is minimum, and let $v \in N_G(B^*)$. Then $(N_G(b^*) \setminus \{v\}) \cup \{b^*\}$ is a solution of* PAU-VC.

In conclusion, we can find a solution of PAU-VC of a split graph $G$ by checking all neighbors for each vertex in $B$. Thus, it takes $\mathcal{O}(|V(G)|)$ time, because every vertex in $A$ has at most one neighbor in $B$. □

## Conclusion

In this paper, our main contributions are three-fold: a fixed-parameter tractable algorithm for PAU-VC parameterized by clique-width, and linear-time algorithms for unit interval graphs and split graphs. In particular, the first algorithm improves the best-known algorithm for PAU-VC on trees significantly. We believe that these algorithms can be used to generate benchmark datasets for evaluating the performances of AI algorithms on the unique vertex cover problem.

There are still lots of open problems in this topic. Can we design polynomial-time algorithms for interval graphs, chordal graphs, or perfect graphs? It is known that these graph classes admit polynomial-time algorithms for the minimum vertex cover problem (Grötschel, Lovász, and Schrijver 1981). Can we reduce the dependency on clique-width to be single-exponential, or can we show that our algorithm is optimal? Recall that our algorithm runs in time double exponential in the clique-width of a graph. Although the running time seems large, it is still possible that our algorithms are optimal; there are several problems with lower bounds that are double exponential in the tree-width or clique-width (Marx and Mitsou 2016; Golovach et al. 2018; Foucaud et al. 2024; Bliznets and Hecher 2024). Last but not least, can we design approximation algorithms for PAU-VC on general graphs or bipartite graphs?

## Acknowledgments

Y. Chang, O. Kwon, and M. Lee are supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (No. NRF-2021K2A9A2A11101617 and No. RS-2023-00211670). O. Kwon is also supported by the National Research Foundation of Korea (NRF) grant funded by Institute for Basic Science (IBS-R029-C1).

S. An, K. Cho, E. Oh, and H. Shin are supported by Institute of Information Communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2024-00440239) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2024-00358505). K. Cho is also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2024-00410835).

## References

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

Bergougnoux, B.; Kanté, M. M.; and Kwon, O. 2020. An optimal XP algorithm for Hamiltonian cycle on graphs of bounded clique-width. *Algorithmica*, 82(6): 1654–1674.

Bertossi, A. A. 1984. Dominating sets for split and bipartite graphs. *Information processing letters*, 19(1): 37–40.

Bliznets, I.; and Hecher, M. 2024. Tight double exponential lower bounds. In *Theory and applications of models of computation*, volume 14637 of *Lecture Notes in Comput. Sci.*, 124–136. Springer, Singapore. ISBN 978-981-97-2339-3; 978-981-97-2340-9.

Bozeman, C.; Brimkov, B.; Erickson, C.; Ferrero, D.; Flagg, M.; and Hogben, L. 2019. Restricted power domination and zero forcing problems. *Journal of Combinatorial Optimization*, 37: 935–956.

Calabro, C.; Impagliazzo, R.; Kabanets, V.; and Paturi, R. 2008. The complexity of unique $k$-SAT: An isolation lemma for $k$-CNFs. *Journal of Computer and System Sciences*, 74(3): 386–393.

Chartrand, G.; Gavlas, H.; Vandell, R. C.; and Harary, F. 1997. The forcing domination number of a graph. *J. Combin. Math. Combin. Comput.*, 25: 161–174.

Corneil, D. G.; Kim, H.; Natarajan, S.; Olariu, S.; and Sprague, A. P. 1995. Simple linear time recognition of unit interval graphs. *Information processing letters*, 55(2): 99–104.

Corneil, D. G.; and Rotics, U. 2005. On the relationship between clique-width and treewidth. *SIAM J. Comput.*, 34(4): 825–847.

Courcelle, B. 1990. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inform. and Comput.*, 85(1): 12–75.

Courcelle, B.; Makowsky, J. A.; and Rotics, U. 2000. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2): 125–150.

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

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

Fellows, M. R.; Rosamond, F. A.; Rotics, U.; and Szeider, S. 2006. Clique-width minimization is NP-hard (extended abstract). In *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, 354–362. ACM, New York. ISBN 1-59593-134-1.

Ferrero, D.; Hogben, L.; Kenter, F. H.; and Young, M. 2018. The relationship between $k$-forcing and $k$-power domination. *Discrete Mathematics*, 341(6): 1789–1797.

Fomin, F. V.; Golovach, P. A.; Lokshtanov, D.; and Saurabh, S. 2010. Intractability of clique-width parameterizations. *SIAM J. Comput.*, 39(5): 1941–1956.

Fomin, F. V.; Golovach, P. A.; Lokshtanov, D.; and Saurabh, S. 2014. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM J. Comput.*, 43(5): 1541–1563.

Fomin, F. V.; Golovach, P. A.; Lokshtanov, D.; Saurabh, S.; and Zehavi, M. 2019. Clique-width III: Hamiltonian cycle and the odd case of graph coloring. *ACM Trans. Algorithms*, 15(1): Art. 9, 27.

Foucaud, F.; Galby, E.; Khazaliya, L.; Li, S.; Mc Inerney, F.; Sharma, R.; and Tale, P. 2024. Problems in NP can admit double-exponential lower bounds when parameterized by treewidth or vertex cover. In *51st International Colloquium on Automata, Languages, and Programming*, volume 297 of *LIPIcs. Leibniz Int. Proc. Inform.*, Art. No. 66, 19. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern. ISBN 978-3-95977-322-5.

Gabow, H. N.; Kaplan, H.; and Tarjan, R. E. 1999. Unique maximum matching algorithms. In *Proceedings of the thirty-first annual ACM symposium on Theory of Computing*, 70–78.

Golovach, P. A.; Lokshtanov, D.; Saurabh, S.; and Zehavi, M. 2018. Cliquewidth III: the odd case of graph coloring parameterized by cliquewidth. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 262–273. SIAM.

Golumbic, M. C.; and Rotics, U. 2000. On the clique-width of some perfect graph classes. volume 11, 423–443. Selected papers from the Workshop on Theoretical Aspects of Computer Science (WG 99), Part 1 (Ascona).

Grötschel, M.; Lovász, L.; and Schrijver, A. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1: 169–197.

Harary, F.; Slany, W.; and Verbitsky, O. 2007. On the computational complexity of the forcing chromatic number. *SIAM Journal on Computing*, 37(1): 1–19.

Hell, P.; Shamir, R.; and Sharan, R. 2001. A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM Journal on Computing*, 31(1): 289–305.

Hertli, T. 2014a. 3-SAT faster and simpler—Unique-SAT bounds for PPSZ hold in general. *SIAM Journal on Computing*, 43(2): 718–729.

Hertli, T. 2014b. Breaking the PPSZ barrier for unique 3-SAT. In *International Colloquium on Automata, Languages, and Programming*, 600–611. Springer.

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.

Johansson, O. 1998. Clique-decomposition, NLC-decomposition, and modular decomposition—relationships and results for random graphs. In *Proceedings of the Twenty-ninth Southeastern International Conference on Combinatorics, Graph Theory and Computing (Boca Raton, FL, 1998)*, volume 132, 39–60.

Kimura, K.; Kamehashi, T.; and Fujito, T. 2018. The Fewest Clues Problem of Picross 3D. In *9th International Conference on Fun with Algorithms (FUN 2018)*, volume 100, 25:1–25:13.

Kobler, D.; and Rotics, U. 2003. Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Appl. Math.*, 126(2-3): 197–221.

Makowsky, J. A.; and Rotics, U. 1999. On the clique-width of graph with few $P_4$'s. *International Journal of Foundations of Computer Science*, 10(03): 329–348.

Marx, D.; and Mitsou, V. 2016. Double-Exponential and Triple-Exponential Bounds for Choosability Problems Parameterized by Treewidth. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Oum, S. 2009. Approximating rank-width and clique-width quickly. *ACM Trans. Algorithms*, 5(1): Art. 10, 20.

Oum, S.; and Seymour, P. 2006. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4): 514–528.

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

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

Scheder, D.; and Steinberger, J. P. 2017. PPSZ for general $k$-SAT-making Hertli's analysis simpler and 3-SAT faster. In *32nd Computational Complexity Conference (CCC 2017)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

Thomason, A. 1978. Hamiltonian Cycles and Uniquely Edge Colourable Graphs. In *Advances in Graph Theory*, volume 3 of *Annals of Discrete Mathematics*, 259–268. Elsevier.

Tjusila, G.; Besan¸con, M.; Turner, M.; and Koch, T. 2024. How many clues to give? A bilevel formulation for the minimum Sudoku clue problem. *Oper. Res. Lett.*, 54: Paper No. 107105, 6.

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