# Inclusive and Exclusive Vertex Splitting into Specific Graph Classes: NP-Hardness and Algorithms

Ajinkya Gaikwad[1], Hitendra Kumar[2], Soumen Maity[2], S Padmapriya[2], Praneet Kumar Patra[2] and Harsh Sanklecha[2]

[1]Czech Technical University, Prague
[2]Indian Institute of Science Education and Research, Pune

**Abstract**

Graph modification problems ask whether a given graph can be transformed into a target graph class using a bounded number of operations. We study such problems under the vertex splitting operation. Given a graph class $\mathcal{F}$, the problem $\mathcal{F}$-Vertex Splitting asks whether a graph can be transformed into a graph in $\mathcal{F}$ using at most $k$ vertex splits. We consider two variants of the operation: *inclusive* and *exclusive* vertex splitting.

We investigate the problem for three fundamental graph classes: cycle union, linear forests, and constellations. We show that Cycle union-Vertex Splitting and Linear Forest-Vertex Splitting are polynomial-time solvable for both variants. Notably, the latter contrasts with the corresponding vertex- and edge-deletion problems, which are NP-complete. In contrast, we prove that Constellation-Vertex Splitting is NP-complete for both variants, even on cubic graphs and on planar graphs of maximum degree three.

Our results reveal that the complexity of vertex splitting can differ substantially from that of classical graph modification operations, highlighting the distinctive algorithmic behavior of this operation and enriching the understanding of its complexity landscape.

**Keywords**

Vertex Splitting, Cycle, Linear Forest, Constellation, Graph Modification

## 1. Introduction

Graph modification problems constitute a fundamental area of research in both classical and parameterized complexity theory. Given a graph and a target graph class, the objective is typically to determine whether the graph can be transformed into the target class using at most $k$ allowed operations. A variety of graph operations, including vertex deletion, edge deletion, edge editing, and edge contraction, have been extensively studied from both algorithmic and complexity-theoretic perspectives [1, 2, 3, 4].

In recent years, a relatively new and intriguing graph operation, known as vertex splitting, has attracted growing attention [5, 6, 7, 8, 9, 10, 11] in the algorithmic and complexity-theoretic community. A vertex splitting operation replaces a vertex $v$ with two new non-adjacent vertices $v_1$ and $v_2$ such that $N(v) = N(v_1) \cup N(v_2)$, where $N(v)$ denotes the neighborhood of $v$. If the neighborhoods of $v_1$ and $v_2$ are disjoint, that is, $N(v_1) \cap N(v_2) = \emptyset$, then the operation is called an *exclusive vertex splitting*. Otherwise, the operation is referred to as an *inclusive vertex splitting*.

The vertex splitting operation was first introduced by Tutte [12]. Historically, the operation was initially defined so that the two newly created vertices were adjacent. Later, a variant in which the two new vertices are non-adjacent emerged. Vertex splitting was subsequently studied in the context of graph drawing and planarization [13, 11], where it was used as a technique for reducing edge crossings in dense graphs. Recent research has focused on the computational complexity of vertex-splitting problems under various graph classes and different variants of the operation. Beyond its applications in graph drawing and planarization, vertex splitting has also found relevance in other domains, including circuit design [14] and statistical data analysis [15].

*ICTCS 2026: Italian Conference on Theoretical Computer Science, September 7–9, 2026, Udine, Italy*
ajinkyag.iiser@gmail.com (A. Gaikwad); hitendra.kumar1729@gmail.com (H. Kumar); soumen@iiserpune.ac.in (S. Maity); padmapriya@ucsc.edu (S. Padmapriya); praneet03res@gmail.com (P. K. Patra); harshsanklecha4348@gmail.com (H. Sanklecha)

For each fixed graph class $\mathscr{F}$, we define a distinct vertex splitting problem, denoted as $\mathscr{F}$-Inclusive (respectively Exclusive) Vertex Splitting, which asks whether a given graph $G$ can be transformed into a graph satisfying $\mathscr{F}$ by performing at most k inclusive (respectively exclusive) vertex splits. While the inclusive variant allows greater flexibility and can simulate the exclusive one, the exclusive variant is often more restrictive and structurally cleaner. Understanding how these variants differ in their expressive and algorithmic power remains a central challenge in the theory of vertex splitting.

A variety of graph classes have been studied in the context of vertex splitting, including cluster graphs [9, 10], bicluster graphs [16, 5], claw-free graphs [6], bipartite graphs [10], pathwidth-one graphs [8], uniform cluster graphs [17], $s$-club cluster graphs [18], as well as forest, split, and threshold graphs [9]. In contrast, comparatively simpler graph classes such as cycle union, linear forests, and constellations have been overlooked in this setting, despite the fact that these classes have been studied extensively under other graph modification operations [19, 20]. Motivated by this gap, we investigate the complexity of vertex splitting problems for cycle union, linear forests, and constellations.

Several of our results are particularly striking. For example, the problem of transforming a graph into a linear forest through vertex deletion, known as Co-Path Packing, is NP-complete [21, 22]. Likewise, the corresponding edge deletion variant, called Co-Path Set, and edge contraction variant, known as Path-Contraction are also NP-complete [4, 23, 24]. In contrast, we prove that the vertex splitting variant for obtaining a linear forest is polynomial-time solvable. This contrast highlights that, unlike many classical graph modification operations, the complexity of vertex splitting problems can vary significantly depending on the target graph class. Consequently, studying individual graph classes, including structurally simple ones, is essential for better understanding the algorithmic and complexity-theoretic landscape of vertex splitting problems.

### 1.1. Vertex Splitting Operation and Problem Definitions

A *vertex split* is a graph modification in which a vertex $v$ is replaced by two new vertices, $v_1$ and $v_2$. Each edge originally incident to $v$ is reassigned to either $v_1$ or $v_2$ or both. Abu-Khzam et al. [25] proposed two types of vertex splitting operations: *Inclusive Vertex Splitting* and *Exclusive Vertex Splitting*. We now formally define both notions below.

**Definition 1** (Inclusive Vertex Splitting)**.** *Let $G = (V, E)$ be a graph and $v \in V$ be a vertex. An inclusive vertex split replaces $v$ with two new vertices $v_1$ and $v_2$, such that $N(v_1) \cup N(v_2) = N(v)$.*

**Definition 2** (Exclusive Vertex Splitting)**.** *Let $G = (V, E)$ be a graph and $v \in V$ be a vertex. An exclusive vertex split replaces $v$ with two new vertices $v_1$ and $v_2$, such that:*

- $N(v_1) \cup N(v_2) = N(v)$*;*
- $N(v_1) \cap N(v_2) = \emptyset$*, ensuring that $v_1$ and $v_2$ have disjoint neighborhoods.*

By definition, it is clear that an exclusive vertex split is a special case of an inclusive vertex split. If $v$ is split into $v_1$ and $v_2$, then $v_1$ and $v_2$ are called the *successors* of $v$. Every vertex split increases the number of vertices in a graph by exactly one. Typically, we are interested in carrying out more than one vertex split. To facilitate this, we introduce the concept of a splitting sequence.

**Definition 3.** *A splitting sequence of $k$ splits is a sequence of graphs $G_0, \ldots, G_k$, such that $G_{i+1}$ is obtainable from $G_i$ via a vertex split for $i \in \{0, \ldots, k-1\}$.*

The notion of *successor vertices* is extended in a transitive and reflexive way to splitting sequences. For each fixed graph class $\mathscr{F}$, we define distinct vertex splitting problems as follows.

| $\mathscr{F}$-Inclusive Vertex Splitting |
| --- |
| **Input:** An undirected graph $G = (V, E)$ and a positive integer $k$. |
| **Question:** Is there a sequence of at most $k$ inclusive vertex splits that transform $G$ into a graph $G'$ such that $G' \in \mathscr{F}$? |

$\mathcal{F}$-Exclusive Vertex Splitting
**Input:** An undirected graph $G = (V, E)$ and a positive integer $k$.
**Question:** Is there a sequence of at most $k$ exclusive vertex splits that transform $G$ into a graph $G'$ such that $G' \in \mathcal{F}$?

In this work, we investigate the computational complexity of $\mathcal{F}$-Inclusive Vertex Splitting and $\mathcal{F}$-Exclusive Vertex Splitting, where $\mathcal{F}$ corresponds to the class of cycle union, linear forests, or constellations. When we consider inclusive vertex splitting and $\mathcal{F}$ corresponds to the class of cycle union, linear forests, and constellations, the resulting problems are referred to as Cycle Union-Inclusive Vertex Splitting, Linear Forest-Inclusive Vertex Splitting, and Constellation-Inclusive Vertex Splitting, respectively. An analogous naming convention is adopted for the exclusive variant, with the term "Inclusive" replaced by "Exclusive".

## 1.2. Our Results and Methods

This work systematically investigates the computational complexity of two variants of vertex splitting: inclusive and exclusive vertex splitting—for three fundamental graph classes: cycle union, linear forests, and constellations.

### 1.2.1. Polynomial-Time Case

We show that both Cycle Union-Inclusive Vertex Splitting and Cycle Union-Exclusive Vertex Splitting are polynomial-time solvable. For the inclusive variant, we reduce the problem to finding a minimum closed walk that traverses every edge of the input graph to an instance of the well-known Chinese Postman Problem, which is solvable in polynomial time. A key intermediate result shows that converting a graph into a disjoint union of cycles using at most $k$ inclusive vertex splits is equivalent to converting it into a single cycle with the same number of splits. Combined with a structural observation linking the number of required vertex splits to the length of the covering closed walk, we obtain a polynomial-time algorithm.

**Theorem 1.** Cycle Union-Inclusive Vertex Splitting *is solvable in polynomial time.*

For the exclusive variant, we derive a necessary condition: any yes-instance requires the input graph to be Eulerian (all vertices have even degree). If the graph contains vertices of odd degree, no sequence of exclusive vertex splits can produce cycle union. For Eulerian graphs, we leverage Eulerian circuits—closed walks traversing each edge exactly once—and adapt the walk-based splitting procedure from the inclusive case to construct cycle union via exclusive splits in polynomial time.

**Theorem 2.** Cycle Union-Exclusive Vertex Splitting *is solvable in polynomial time.*

We also show that both Linear Forest-Inclusive Vertex Splitting and Linear Forest-Exclusive Vertex Splitting are polynomial-time solvable. Analogous to the cycle case (which relies on closed walks), linear forest transformation corresponds to decomposing the graph into open edge-disjoint trails.

**Theorem 3.** *For any graph $G$, the minimum number of exclusive vertex splits required to obtain a linear forest $G_P$ can be computed in polynomial time.*

We give a lemma (See Lemma 7) that says the minimum number of exclusive splits required is the same as the minimum number of inclusive splits required to convert a graph $G$ to a linear forest. This gives us the following theorem.

**Theorem 4.** *For any graph $G$, the minimum number of inclusive vertex splits required to obtain a linear forest $G_P$ can be computed in polynomial time.*

Abu-Khzam et al. [26] recently established a linear-time algorithm for computing the minimum number of inclusive or exclusive vertex splits required to transform a graph into a linear forest. Theorem 3 and Theorem 4 follow as a direct corollary of their result by applying the Handshaking Lemma and distinguishing between Eulerian and non-Eulerian graphs. We note that the work of Abu-Khzam et al. [26] appeared only recently, and our results were obtained independently.

For all the polynomial-time algorithms, we assume that the input graph is connected because if the graph is disconnected, then the problems can be solved in each connected component independently.

#### 1.2.2. NP-hardness results

In sharp contrast, we prove that Constellation-Inclusive Vertex Splitting and Constellation-Exclusive Vertex Splitting are both NP-complete. We establish NP-completeness via a chain of polynomial-time reductions originating from the classic NP-complete Vertex Cover problem.

**Theorem 5.** Constellation-Inclusive Vertex Splitting *is NP-complete on cubic graphs and on planar graphs of maximum degree three.*

**Theorem 6.** Constellation-Exclusive Vertex Splitting *is NP-complete on cubic graphs and on planar graphs of maximum degree three.*

Theorem 5 and Theorem 6 strengthen a result of Abu-Khzam et al. [26], who proved that transforming a graph into a caterpillar is NP-complete. Since every star is a caterpillar, transforming a graph into a star is a more restrictive requirement. Consequently, our results establish stronger NP-completeness results than those presented in [26].

## 2. Preliminaries

Throughout this paper, $G = (V, E)$ denotes a finite, simple, undirected graph with vertex set $V$ and edge set $E \subseteq V \times V$. An edge $(u, v) \in E$ is written as $uv$. For $u \in V$, the *open neighborhood* is $N(u) = \{v \in V : uv \in E\}$ and the *closed neighborhood* is $N[u] = N(u) \cup \{u\}$. The degree of $u$ in $G$ is $d_G(u) = |N(u)|$. For $U \subseteq V$ and $F \subseteq E$, the *induced subgraphs* are $G[U]$ and $G[F]$, respectively. We write $G - U = G[V \setminus U]$, and $G - F = (V, E \setminus F)$.

A *walk* in a graph $G = (V, E)$ is a sequence of vertices $v_1, v_2, \ldots, v_m$ such that $v_i v_{i+1} \in E$ for all $i \in [m-1]$; in a walk, both vertices and edges may repeat. A *path* is a walk in which all vertices are distinct. A *trail* is a walk in which no edge is repeated, although vertices may repeat. A closed walk is a walk where the first and the last vertices are identical. A closed trail is a closed walk that is also a trail. A cycle is a closed trail where all the vertices except the first and the last are different. An *Eulerian trail* is a trail that uses every edge of the graph exactly once. A graph is called *Eulerian* if it contains an *Eulerian circuit*, that is, an Eulerian trail that starts and ends at the same vertex. A graph is called *non-Eulerian* if it is not Eulerian. It is very well known that a connected graph is Eulerian if and only if every vertex has even degree.

A graph is *connected* if every pair of vertices is joined by a path; otherwise, it is *disconnected*. Maximal connected induced subgraphs of a disconnected graph are called *components*. We denote the number of connected components of a graph $G$ as $\mathscr{C}(G)$. A *linear forest* is a graph whose components are paths. A *cycle* is a connected graph in which every vertex has degree 2 (equivalently, traditionally it is defined as a path whose first and last vertices are equivalent). A disjoint union of cycles is called *cycle union*. A *cubic graph* has all vertices of degree 3. A *tree* is a connected acyclic graph, a *forest* is a graph whose components are trees, and a *star* is a graph with one central vertex adjacent to all others. A disjoint union of stars is called *constellation*. For a graph $G$, a *vertex cover* is a set $X \subseteq V(G)$ such that every edge of $G$ has at least one endpoint in $X$. We use $\tau(G)$ to denote the size of a minimum vertex cover of $G$.

## 3. Cycle Union-Inclusive/Exclusive Vertex Splitting

We formally define the problems considered in this section.

| Cycle Union-Inclusive/Exclusive Vertex Splitting |
|---|
| **Input:** An undirected graph $G = (V, E)$, and a positive integer $k$. |
| **Question:** Is there a sequence of at most $k$ inclusive/exclusive vertex splittings that transform $G$ into cycle union? |

**Observation 1.** *Let $G$ be a graph and $G'$ be a cycle union obtained using a sequence of inclusive vertex splits from $G$. Then the number of splits required to obtain $G'$ from $G$ is exactly $|E(G')| - |V(G)|$.*

*Proof.* Since $G'$ is a cycle union, the number of vertices and edges of $G'$ is the same. As each of the splits introduces exactly one new vertex, the number of inclusive vertex splits required to obtain $G'$ from $G$ is exactly $|V(G')| - |V(G)|$, which is the same as $|E(G')| - |V(G)|$. □

**Lemma 2.** *Let $G = (V, E)$ be a connected graph and let $W$ be a closed walk that visits all vertices and covers all the edges in $G$. If $|W|$ denotes the length of the walk $W$, then there exists a sequence of exactly $|W| - |V(G)|$ inclusive vertex splits such that the resulting graph belongs to the class of cycle union (or a single cycle) with the only exception being an edge graph.*

*Proof.* Let $W = (w_0, w_1 \cdots w_{l-1}, w_l)$ be a closed walk in a graph $G$, where $w_l = w_0$. Note that the walk may traverse the same edge multiple times. Algorithm 1 provides a procedure for generating a cycle union using a sequence of inclusive vertex splits. Here, for a vertex $v \in V(G)$, $v^b$ denotes the copy of $v$ that is black and $v^g$ denotes the copy of $v$ that is grey. Note that, at an intermediate step of the Algorithm 1, some vertices are colored grey while others are colored black. Also note that whenever the walk $W$ in $G$ returns to a previously visited vertex, our algorithm considers the unique black successor of that vertex.

**Algorithm 1** Constructing cycle union from a walk $W$

**Input:** Closed walk $W = (w_0, w_1, \ldots, w_{l-1}, w_l)$ in graph $G$
**Initialize:** Color all vertices of $G$ black
Construct multigraph $G_M = (V, E_M)$ where multiplicity of edge $uv$ equals its number of occurrences in $W$
$w_i^s$ represents the unique black successor of $w_i$
$\texttt{prev} \leftarrow w_{l-1}$
$j \leftarrow 0$
**while** walk not exhausted **do**
  **if** $j = l - 1$ **or** $\deg(w_j^s) = 2$ **then**
    Change color of $w_j^s$ from black to grey and call it $w_j^g$
    $\texttt{prev} \leftarrow w_j^g$
  **else if** $j \neq l - 1$ **and** $\deg(w_j^s) \geq 4$ **then**
    Do an exclusive vertex splitting of $w_j^s$ into a black and grey copy, $w_j^b$ and $w_j^g$ respectively, making $w_j^g$ adjacent to prev and $w^s_{(j+1) \bmod l}$, satisfying the exclusive vertex split conditions, we make $w_j^b$ adjacent to the original neighbors of $w_j^s$ using the same edges (and parallel edges) except an edge to prev and an edge to $w^s_{(j+1) \bmod l}$.
    $\texttt{prev} \leftarrow w_j^g$
  **end if**
  $j \leftarrow (j + 1) \bmod l$
**end while**
**Output:** Graph $G'$ with $|W|$ edges, which is a cycle union.

Algorithm 1 constructs the multigraph $G_M = (E_M, V)$ from $G$ where the number of parallel edges between two vertices $u, v \in V(G_M)$ is the number of times the edge $uv$ has appeared in the walk $W$. The

output graph $G'$ in Algorithm 1 has $|W|$ edges. Hence, by Observation 1, the number of vertex splits required to get $G'$ is exactly $|W| - |V(G)|$.

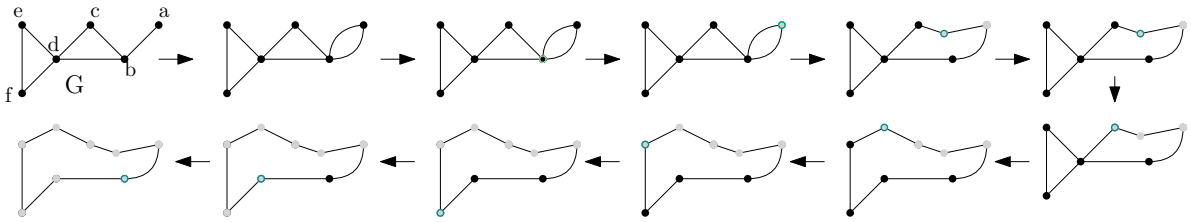


**Figure 1:** We run the algorithm on the graph $G$ with the walk-abcdefdba. Note that, the vertices with the green circle denote the prev vertex (variable) at that step and the vertices denoted as grey and black are the grey and black vertices as explained in the algorithm.

Call the modified graph returned by this algorithm $G'$. One can check that the number of edges in $G'$ is $|W|$, and by construction, the resultant connected graph has all vertices as degree 2 vertices. This is because all the vertices in the end are grey vertices, and every time a grey vertex is constructed, it is made adjacent to two vertices, one of which is a previous grey vertex. Thus, the final graph is a cycle generated by a sequence of exclusive vertex splits on $G$. Figure 1 demonstrates a run of the algorithm to "open" a walk.

Note that the exclusive split sequence on the multigraph $G_M$, which ultimately produces the cycle $G'$, can also be viewed as an inclusive vertex split sequence on the graph $G$ that yields the same final graph $G'$. In particular, consider an exclusive split of a vertex $v$ adjacent to $u$, producing vertices $v_1$ and $v_2$, where both $v_1$ and $v_2$ are made adjacent to $u$ because $u$ was connected to $v$ by parallel edges before the split. In the inclusive version, we can likewise make both $v_1$ and $v_2$ adjacent to $u$. Here, parallel edges are not required beforehand; the adjacency of $u$ and $v$ alone is sufficient to make the step valid. This therefore ensures that the adjacency list of all the vertices in the exclusive split variant (from $G_M$) is exactly the same as that of the inclusive split variant (from $G$) at every step. Furthermore, in any step, the resulting graph does not have any parallel edge that was not already present in $G$ (as it was never constructed). This completes the proof. □

An important observation is that, after performing the exclusive splits, the resulting graph admits a minimum closed walk covering all vertices and edges in which no vertex is visited more than once, except for the starting and ending vertex. Consequently, for any connected graph $G$ with more than two vertices, the resulting graph is a cycle. However, consider the graph $G_e$ consisting of two vertices joined by two parallel edges. In this case, the required property is already satisfied without performing any split operation. Consequently, the expression $|W| - |V(G)|$ incorrectly suggests that no splits are required. The reason is that $G_e$ is the unique boundary case: it is the largest graph satisfying the property while not itself being a cycle. Hence, the formula does not apply to this exceptional case.

We now state an observation, together with Lemma 2, that highlights the similarity between closed edge-covering walks and cycles obtained through inclusive vertex splits.

**Observation 3.** *If an inclusive vertex split sequence transforms a graph $G$ into a single cycle, say $(c_0, c_1, \cdots, c_l)$ with $c_0 = c_l$ in $k$ splits. Let $c_i$ be the successor of $v_i \in V(G)$. Then, the sequence of vertices $(v_0, v_1, \cdots, v_l)$ is a closed walk in $G$ covering all the edges at least once.*

*Proof Sketch.* Since an inclusive vertex split never decreases the number of successors of a vertex in $G$, the sequence $(v_0, v_1, \cdots, v_l)$ covers (possibly multiple times) all the vertices of $G$. Therefore, we just need to show that each $v_iv_{i+1}$ corresponds to an edge in $G$ and that each edge $v_jv_k$ of $G$ appears as one of the edges of the form $v_iv_{i+1}$ for some $i \in [l-1] \cup \{0\}$.

For the first part, suppose for contradiction that $v_jv_k \notin E(G)$, but $c_ic_{i+1}$ is an edge in the final cycle where $c_i$ and $c_{i+1}$ are successors of $v_j$ and $v_k$, respectively. Let $s$ be the first split after which an edge appears between successors of $v_j$ and $v_k$. During this split, some vertex $x$ (without loss of generality, say

a successor of $v_j$) is replaced by $x_1$ and $x_2$, and one of them becomes adjacent to a successor of $v_k$ that was not adjacent to $x$. This step violates the definition of inclusive split, hence, $v_jv_k \in E(G)$.

For the second part, note that whenever a vertex $v$ adjacent to $u$ is split, one of its successors remains adjacent to $u$. This shows that an edge $vu$ in $G$ appears as an edge $c_ic_{i+1}$ in the final cycle where $c_i$ is a successor of $v$ and $c_{i+1}$ is a successor of $u$.

Note that the first argument essentially shows that the sequence $(v_0, v_1, \cdots, v_l)$ is a closed walk and the second argument shows that the closed walk traverses each edge of $G$ at least once. Therefore, the statement of the observation holds.

Now, we observe that if a certain number of inclusive vertex splits is enough to get a cycle union from a graph, then the same number of inclusive vertex splits is also enough to obtain a single cycle.

**Lemma 4.** *Given a graph $G = (V, E)$. There exists a sequence of $k$ inclusive vertex splits that transforms $G$ into a disjoint union of cycles if and only if there exists a sequence of $k$ inclusive vertex splits that transforms $G$ into a single cycle.*

*Proof.* Backward direction is trivial. For the forward direction, let us assume that there exists a sequence of $k$ inclusive vertex splits that transforms $G$ into a disjoint union of cycles, say $G'$. By Observation 1, $k = |E(G')| - |V(G)|$. If $G'$ is a single cycle, then we are done. So, we assume that $G$ is a disjoint union of at least two cycles, say $C_1, C_2, \ldots, C_r$ ( $r > 1$). Each cycle in $G'$ corresponds to a closed walk in the graph $G$. For each $i \in [r]$, let $T_i$ denote the closed walk corresponding to the cycle $C_i$ in $G$. These walks may traverse the same edge multiple times, and different walks may share edges or vertices.

Construct a new multigraph $\tilde{G} = (V, \tilde{E})$ as follows: for every edge $e \in E(G)$ and for every occurrence of $e$ as an edge in some walk $T_i$, include a distinct parallel copy of $e$ in $\tilde{E}$. Then each $T_i$ becomes a trail in $\tilde{G}$ and the trails corresponding to different $T_i$ are edge-disjoint in $\tilde{G}$. This implies that every vertex in the graph $\tilde{G}$ has even degree. Hence $\tilde{G}$ is an Eulerian graph.

We find an Eulerian trail in the graph $\tilde{G}$. By using the trail, we obtain a single closed walk, say $W$, in the original graph, where each parallel edge in $\tilde{G}$ is replaced by its corresponding original edge in $G$. $W$ has exactly as many edges (counted with multiplicity) as there are edges in $G'$. By Lemma 2, we can obtain a single cycle, say $C'$, by $|W| - |V(G)|$ number of inclusive vertex splits. We have already observed that $|W| = |E(G')|$. Hence, we need exactly $|E(G')| - |V(G)|$, which is equal to $k$, the number of splits to get a single cycle. □

As a consequence of Observation 1, Observation 3 and Lemma 4, we conclude that for a graph $G$, the problem of determining the minimum number of inclusive vertex splits required to obtain a cycle union reduces to the problem of finding the smallest closed walk in $G$ that traverses every edge at least once. The problem of finding the smallest closed walk that visits every edge is precisely the well-studied Chinese Postman problem, which is polynomial-time solvable [27]. Therefore we have that the Cycle Union-Inclusive Vertex Splitting is solvable in polynomial time.

**Theorem 1.** Cycle Union-Inclusive Vertex Splitting *is solvable in polynomial time.*

Note that there is no way to obtain a cycle union from an isolated edge by performing a sequence of exclusive vertex splits. However, the situation changes if we allow inclusive vertex splits. To see this, consider an isolated edge $(x, y)$. By first splitting $x$ inclusively, where both $x_1$ and $x_2$ are adjacent to $y$, then splitting $y$ similarly, making $y_1$ and $y_2$ adjacent to both $x_1$ and $x_2$, we obtain a cycle of length 4.

We now note the properties of the graphs that should hold for instances that admit an exclusive vertex split sequence resulting in a cycle union.

**Lemma 5.** *If $(G, k)$ is a yes-instance of* Cycle Union-Exclusive Vertex Splitting, *then every vertex of $G$ has even degree.*

*Proof.* Let $G_0, \ldots, G_l$ be a splitting sequence such that $G_0 = G$ and $G_l$ is a cycle union. Since we perform only exclusive vertex splitting, we have

$$d_G(v) = \sum_{v' : v' \text{ is a successor of } v \text{ in } G_l} d_{G_l}(v').$$

Since $G_l$ is a cycle union, every vertex in $G_l$ has even degree. Therefore, every vertex in $G$ has even degree. □

By applying Lemma 5, we obtain the following corollary.

**Corollary 6.** *If $G$ has a vertex with odd degree, then $(G, k)$ is a no-instance of* Cycle Union-Exclusive Vertex Splitting *for any $k \geq 0$.*

If all vertices of a connected graph have even degree, that is, if the graph is Eulerian, then there exists an Eulerian circuit, a closed walk that traverses every edge exactly once and returns to its starting vertex. Such a walk can be computed in polynomial time, equivalently as a special case of the Chinese Postman Problem in the undirected Eulerian setting. In this case, we may apply the same procedure as in the inclusive vertex split construction; since the Eulerian trail uses each edge with multiplicity one, the transformation does not increase the number of edges. Consequently, after a sequence of exclusive vertex splits, we obtain a cycle union. This establishes the stronger result:

**Theorem 2.** Cycle Union-Exclusive Vertex Splitting *is solvable in polynomial time.*

## 4. Linear Forest-Inclusive/Exclusive Vertex Splitting

The aim of this section is to give a polynomial-time algorithm for Linear Forest-Inclusive Vertex Splitting and Linear Forest-Exclusive Vertex Splitting. Drawing an analogy with the previous section, we note that the focus of attention now shifts from searching for closed walks to searching for trails. We first define our problems formally.

| Linear Forest-Inclusive/Exclusive Vertex Splitting |
| --- |
| **Input:** An undirected graph $G = (V, E)$, and a positive integer $k$. |
| **Question:** Is there a sequence of at most $k$ inclusive/exclusive vertex splittings that transform $G$ into a linear forest? |

**Lemma 7.** *The minimum number of exclusive splits required to convert a graph $G$ to a linear forest is the same as the minimum number of inclusive splits required to convert a graph $G$ to a linear forest.*

*Proof.* Let $k^{\text{in}}$ and $k^{\text{ex}}$ be the minimum number of inclusive and exclusive vertex splits, respectively, required to convert $G$ to a linear forest. Since every exclusive split is a special case of an inclusive split, any valid sequence of exclusive splits is also a valid sequence of inclusive splits. Thus, $k^{\text{in}} \leq k^{\text{ex}}$.

Now we aim to prove $k^{\text{ex}} \leq k^{\text{in}}$. Suppose a sequence of $k^{\text{in}}$ inclusive splits transforms $G$ into a linear forest $G^{\text{in}}$. We now construct a new graph $G^{\text{ex}}$ using the corresponding sequence of $k^{\text{in}}$ exclusive splits. Whenever an inclusive split replaces a vertex $x$ with new vertices $x_1$ and $x_2$, we apply an exclusive split to $x$ by distributing its neighbors as follows:

- If a neighbor $y$ is assigned to only $x_1$ or only $x_2$ in the inclusive split, we preserve that assignment.
- If a neighbor $y$ is assigned to both $x_1$ and $x_2$, we arbitrarily assign $y$ to exactly one of them.

By definition, this ensures the new neighborhoods are disjoint, making every split exclusive. Because this process perfectly mirrors the inclusive splits but simply drops redundant edge assignments, the resulting graph $G^{\text{ex}}$ is a subgraph of $G^{\text{in}}$. Since $G^{\text{in}}$ is a linear forest, its subgraph $G^{\text{ex}}$ must be a linear forest. Therefore, $G$ can be converted into a linear forest using $k^{\text{in}}$ exclusive splits, which implies $k^{\text{ex}} \leq k^{\text{in}}$. Combining both inequalities yields $k^{\text{in}} = k^{\text{ex}}$. □

To prove the main result of this section, we make use of the following proposition.

**Proposition 8** ([28, 29]). *For a nontrivial connected graph $G = (V, E)$ with $\alpha$ odd-vertices, the minimum number of trails (denoted as $\mathrm{MNT}(G)$) that decompose $E(G)$ is $\max(\frac{\alpha}{2}, 1)$, i.e., $\mathrm{MNT}(G) = \max(\frac{\alpha}{2}, 1)$.*

**Theorem 3.** *For any graph $G$, the minimum number of exclusive vertex splits required to obtain a linear forest $G_P$ can be computed in polynomial time.*

*Proof.* Let $k$ be a number of splits required to obtain linear forest $G_P$ from $G$. Since we use only exclusive vertex splits, the number of edges is preserved, that is, $|E(G_P)| = |E(G)|$. Note that each split increases the number of vertices by exactly one. Hence, $|V(G_P)| - \mathscr{C}(G_P) = |V(G)| + k - \mathscr{C}(G_P) = |E(G_P)| = |E(G)|$, where $\mathscr{C}(G_P)$ is the number of connected components in $G_P$, which implies $k = |E(G)| - |V(G)| + \mathscr{C}(G_P)$. Thus, minimizing $k$ is equivalent to minimizing the number of connected components of $G_P$. As in the cycle case, the paths of the resulting linear forest $G_P$ correspond to trails in $G$. Hence, minimizing $k$ is equivalent to minimizing the number of edge-disjoint trails covering all edges of $G$, which gives a lower bound to $k$ of $|E(G)| - |V(G)| + \mathrm{MNT}(G)$.

Let $\alpha$ be the number of odd-degree vertices in $G$. We introduce $\frac{\alpha}{2}$ false edges between paired odd-degree vertices, making the graph Eulerian. Now we use Algorithm 1 to get a single cycle. Since the length of cycle is $|E(G)| + \frac{\alpha}{2}$, the number of splits required to a single cycle is $|E(G)| - |V(G)| + \frac{\alpha}{2}$. Now we remove the false edges, which gives us a linear forest. If there are no false edges, i.e., $G$ is Eulerian, an additional split would be required to convert the cycle into a path. Hence an Eulerian graph will take $|E(G)| - |V(G)| + 1$ splits. In general, the number of exclusive splits used in the process is $|E(G)| - |V(G)| + \mathrm{MNT}(G)$. Since this is the lower bound on $k$, the theorem follows. □

From Lemma 7, we have the following theorem.

**Theorem 4.** *For any graph $G$, the minimum number of inclusive vertex splits required to obtain a linear forest $G_P$ can be computed in polynomial time.*

## 5. Constellation-Inclusive/Exclusive Vertex Splitting

A disjoint union of stars is known as a *constellation*. In this section, we prove that both Constellation-Inclusive Vertex Splitting and Constellation-Exclusive Vertex Splitting are NP-hard. To do so, we establish the following chain of polynomial-time reductions, starting from the classical NP-hard Vertex Cover problem:

$$\text{Vertex Cover} \leq_P \text{Weighted Star Decomposition}$$

$$\leq_P \text{Constellation-Inclusive Vertex Splitting}$$

We formally define the Constellation-Inclusive/Exclusive Vertex Splitting and Weighted Star-Decomposition below.

| Constellation-Inclusive/Exclusive Vertex Splitting |
|---|
| **Input:** An undirected graph $G = (V, E)$, and a positive integer $k$. |
| **Question:** Is there a sequence of at most $k$ inclusive/exclusive vertex splitting that transform $G$ into a constellation? |

**Definition 4.** *A star-decomposition $\mathscr{P}$ of $G$ is a set $\{S_1, S_2, \ldots, S_l\}$ of subgraphs of $G$ such that $E(S_1), \ldots, E(S_l)$ form a partition of $E(G)$, and each $S_i$ is a star graph. The weight of $\mathscr{P}$ is defined as*

$$\mathrm{wgt}_G(\mathscr{P}) = \sum_{v \in V} \#\mathscr{P}(v),$$

*where $\#\mathscr{P}(v) := |\{S_i \mid v \in V(S_i), S_i \in \mathscr{P}\}|$ denotes the number of stars in $\mathscr{P}$ that contain $v$.*

Now, we formulate the associated decision problem:

| Weighted Star-Decomposition |
| --- |
| **Input:** An undirected graph $G = (V, E)$, and a positive integer $k$. |
| **Question:** Does there exist a star-decomposition $\mathscr{P}$ of $G$ such that $\mathtt{wgt}(\mathscr{P}) \leq k$? |

**Lemma 9.** Weighted Star Decomposition *is NP-complete on cubic graphs and on planar graphs of maximum degree three.*

*Proof.* It is easy to see that Weighted Star-Decomposition is in NP. It is known that Vertex Cover is NP-complete on cubic graphs and also on planar graphs of maximum degree three [30]. To prove that Weighted Star-Decomposition is NP-hard on cubic graphs and also on planar graphs of maximum degree three, we give a reduction from Vertex Cover. We prove that $(G, k)$ is a yes-instance of Vertex Cover if and only if $(G, m + k)$ is a yes-instance of Weighted Star-Decomposition.

For the forward direction, we assume that $(G, k)$ is a yes-instance of Vertex Cover. Suppose $X = \{v_1, v_2, \ldots, v_l\}$ is a vertex cover of $G$ with $l \leq k$. Now, we define $E_1 = \{v_1 w : w \in N(v_1)\}$ and $E_i = \{v_i w : w \in N(v_i) \setminus \{v_1, v_2, \ldots, v_{i-1}\}\}$ for $2 \leq i \leq l$. We claim that $\mathscr{P} = \{G[E_1], G[E_2], \ldots, G[E_l]\}$ is a star decomposition with $\mathtt{wgt}(\mathscr{P}) \leq m + k$.

Clearly, each $E_i$ is a star with center vertex $v_i$. Note that $G - X$ is an independent set, so all the edges of graph $G$ are either between $X$ and $G - X$ or between the vertices of $X$. The way the construction of $E_i$ is done and $X$ is a vertex cover makes it clear that $\mathscr{P}$ covers all edges of $G$. It is also clear that $E_i \cap E_j = \emptyset$ whenever $i \neq j$.

Note that if $E_i$ has $p_i$ edges, then the number of vertices in $E_i$ is $p_i + 1$. Hence,

$$\mathtt{wgt}(\mathscr{P}) = \sum_{1 \leq i \leq l} |V(E_i)| = \sum_{1 \leq i \leq l} (p_i + 1) = \sum_{1 \leq i \leq l} p_i + l = m + l \leq m + k.$$

For the reverse direction, we assume that there exists a star decomposition, say $\mathscr{S}$ of $G$ with $\mathtt{wgt}(\mathscr{S}) \leq m + k$. Suppose $\mathscr{S} = \{S_1, S_2, \ldots, S_q\}$.

Define $C = \{c_1, c_2, \ldots, c_q\}$ where $c_i$ is center of star $S_i$. Suppose a star with center $c_i$ contains $p_i$ number of edges, then the number of vertices is $p_i + 1$ in that star. Now we see that $\mathtt{wgt}(\mathscr{S}) = \sum_{1 \leq i \leq q}(p_i + 1) = \sum_{1 \leq i \leq q} p_i + q = m + q$. Since $\mathtt{wgt}(\mathscr{S}) \leq m + k$, we have $q \leq k$.

We claim that $C = \{c_1, c_2, \ldots, c_q\}$ is a vertex cover of $G$. Let $uv$ be an edge of the graph $G$. Since $\mathscr{S}$ is a sigma star decomposition of $G$, $uv$ must be in $S_i$ for some $i$. Since $S_i$ is a star graph, either $u$ or $v$ must be $c_i$. The edge $uv$ is covered by $C$ because $C$ contains $c_i$. So we have proved that $C$ is a vertex cover of size at most $k$, hence $(G, k)$ is a yes-instance of the Vertex Cover. □

Now we aim to prove that Constellation-Vertex Splitting is NP-hard by giving a polynomial-time reduction from Weighted Star Decomposition. But we first give a few lemmas. The statements and proof ideas of Lemma 10 and Lemma 11 are influenced by the work in [9].

**Lemma 10.** *Let $G = (V, E)$ be a graph without isolated vertices, and $\mathscr{P}$ be a star-decomposition of $G$ such that $\mathtt{wgt}_G(\mathscr{P}) \leq |V(G)| + \alpha$. Then, either $G$ is already a constellation graph or there exists a vertex $u \in V$ such that $u$ can be exclusively split in $G$ to obtain a graph $G'$ satisfying:*

1. *$G'$ admits a star-decomposition, say $\mathscr{P}'$,*
2. *$\mathtt{wgt}_{G'}(\mathscr{P}') \leq |V(G')| + \alpha - 1$*
3. *$G'$ has no isolated vertices*

*Proof.* Suppose $\mathscr{P} = \{S_1, S_2, \ldots, S_l\}$ is a star-decomposition of $G$. If $G$ is not already a constellation, there must exist $S_i$ and $S_j$ such that $V(S_i) \cap V(S_j) \neq \emptyset$. Without loss of generality, we assume $S_i = S_1$ and $S_j = S_2$. In this case, let $u \in V(S_1) \cap V(S_2)$. We define $G'$ as the graph obtained by splitting $u$ into two vertices $u_{\text{in}}$ and $u_{\text{out}}$ as follows:

$$N_{G'}(u_{\text{in}}) := N_G(u) \cap V(S_1),$$
$$N_{G'}(u_{\text{out}}) := N_G(u) \setminus V(S_1).$$

We construct a star-decomposition of $G'$, denoted by $\mathscr{P}' = \{S'_1, S'_2, \dots, S'_l\}$, where:

$$S'_i := \begin{cases} S_{\text{in}} & \text{if } S_i = S_1, \\ S_{\text{out},i} & \text{if } u \in V(S_i) \wedge (S_i \neq S_1), \\ S_i & \text{otherwise.} \end{cases}$$

Here, we define

$$V(S_{\text{in}}) = (V(S_1) \setminus \{u\}) \cup \{u_{\text{in}}\}, \qquad E(S_{\text{in}}) = \{xy \in E(S_1) \,:\, u \notin \{x, y\}\} \cup \{u_{\text{in}}v \,:\, uv \in E(S_1)\},$$
$$V(S_{\text{out},i}) = (V(S_i) \setminus \{u\}) \cup \{u_{\text{out}}\}, \quad E(S_{\text{out},i}) = \{xy \in E(S_i) \,:\, u \notin \{x, y\}\} \cup \{u_{\text{out}}v \,:\, uv \in E(S_i)\}.$$

We now verify that $\mathscr{P}'$ is indeed a star-decomposition of $G'$. Clearly $S'_i \cong S_i$, since $S'_i$ are either identical to $S_i$ or are formed by relabeling of a vertex.

Now we have to verify that all the edges of $G'$ are covered by $\mathscr{P}'$. Since each $S'_i \cong S_i$, we have $|E(S_i)| = |E(S'_i)|$. The construction ensures that $S'_i$ are edge-disjoint. Therefore, the total number of edges covered by $\mathscr{P}'$ is $|E(G)| = |E(G')|$. Thus, $\mathscr{P}'$ covers all edges of $G'$ and is a star-decomposition of $G'$.

We now examine the weight of the decomposition $\mathscr{P}'$ and prove the condition 2. We have

$$\begin{aligned} \mathtt{wgt}(\mathscr{P}') &= \sum_{v \in V(G')} \#\mathscr{P}'(v) = \sum_{v \in V(G') \setminus \{u_{\text{in}}, u_{\text{out}}\}} \#\mathscr{P}'(v) + \#\mathscr{P}'(u_{\text{in}}) + \#\mathscr{P}'(u_{\text{out}}) \\ &= \sum_{v \in V(G) \setminus \{u_{\text{in}}, u_{\text{out}}\}} \#\mathscr{P}(v) + 1 + (\#\mathscr{P}(u) - 1) = \sum_{v \in V(G)} \#\mathscr{P}(v) \\ &\leq |V(G)| + \alpha = |V(G')| + \alpha - 1. \end{aligned}$$

Finally, since $G'$ is obtained from $G$ by splitting a vertex, it follows that if $G$ has no isolated vertices, then $G'$ does not either. □

**Lemma 11.** *Let $G = (V, E)$ be a graph and $G' = (V', E')$ be obtained from $G$ by inclusive/exclusive vertex splitting of a vertex $u \in V$ into two vertices $v, w \in V'$. If $\mathscr{P}'$ is a star decomposition of $G'$, then there exists a star decomposition $\mathscr{P}$ of $G$ with $\mathtt{wgt}(\mathscr{P}) \leq \mathtt{wgt}(\mathscr{P}')$.*

*Proof.* By the proof of Lemma 9, it is clear that the minimum weight of a star decomposition of graph $G'$ is $|E(G')| + \tau(G')$. Let $\mathscr{P}'$ be a star decomposition of $G'$. Then $\mathtt{wgt}(\mathscr{P}') \geq |E(G')| + \tau(G')$. An inclusive/exclusive vertex split does not decrease the size of a minimum vertex cover. Hence, $\tau(G) \leq \tau(G')$. Moreover, an inclusive vertex split cannot decrease the number of edges, that is, $|E(G)| \leq |E(G')|$. Therefore, $|E(G)| + \tau(G) \leq |E(G')| + \tau(G') \leq \mathtt{wgt}(\mathscr{P}')$.

Again by the proof of Lemma 9, graph $G$ admits a star decomposition $\mathscr{P}$ of weight exactly $|E(G)| + \tau(G)$. Consequently, $\mathtt{wgt}(\mathscr{P}) = |E(G)| + \tau(G) \leq \mathtt{wgt}(\mathscr{P}')$, which completes the proof. □

**Lemma 12.** *Let $G = (V, E)$ be a graph without any isolated vertices. Then $(G, k)$ is a yes-instance of* Constellation-Inclusive Vertex Splitting *if and only if $(G, |V(G)| + k)$ is a yes-instance of* Weighted Star Decomposition.

*Proof.* Let $(G, k)$ be a yes-instance of Constellation-Inclusive Vertex Splitting, that is, there exists a sequence of at most $k$ inclusive vertex splitting in $G$ such that the resulting graph $G'$ is a disjoint union of stars. Let $G_0, \dots, G_l$ be a sequence of graphs with $G_0 = G$ and $l \leq k$, such that for each $i \in [l]$, the graph $G_i$ is obtained from $G_{i-1}$ via an inclusive vertex split, and $G_l$ is a disjoint union of stars. By identifying each connected component of $G_l$ with its vertex set, we obtain a star decomposition $\mathscr{P}_l$ of $G_l$ with $\mathtt{wgt}(\mathscr{P}_l) = |V(G_l)|$. Each split used in the construction of $G_1, \dots, G_l$ introduces exactly one new vertex, therefore $|V(G_l)| = |V(G)| + l$. Combining this with the fact that $l \leq k$, we derive $\mathtt{wgt}(\mathscr{P}_l) \leq |V| + k$. Using the sequence $G_0, \dots, G_l$ in reverse order, we iteratively apply Lemma 11 $l$ times taking $\mathscr{P}_l$ and $G_l$ as base case and obtain $\mathscr{P}_0, \dots, \mathscr{P}_l$. In particular, it follows that $\mathscr{P}_0$ is a star decomposition of $G$ satisfying $\mathtt{wgt}(\mathscr{P}_0) \leq |V(G)| + k$. Thus, $(G, |V(G)| + k)$ is a yes-instance.

For the backward direction, we assume that $(G, |V(G)| + k)$ is a yes-instance of Weighted Star Decomposition, that is, there exists a star decomposition, say $\mathscr{P} = \{H_1, H_2, \ldots, H_p\}$ with $\mathtt{wgt}(\mathscr{P}) \leq |V(G)| + k$. By iteratively applying Lemma 10 a number of times, say $l$, either until a constellation is obtained as a direct result of the lemma, or alternatively, stopping after $k$ iterations, we obtain the sequences $G_0, \ldots, G_k$ and $\mathscr{P}_0, \ldots, \mathscr{P}_k$, where each graph $G_i \forall 0 \leq i \leq k$ has no isolated vertices.

We now show that $G_k$ must be a disjoint union of stars. As a consequence of the repeated applications of Lemma 10, we get $\mathtt{wgt}(\mathscr{P}_k) \leq |V(G_k)|$. The fact that for each vertex $v \in V(G_k)$ ensures that there exists a subgraph, say $H_i^k \in \mathscr{P}_k$ with $v \in V(H_i^k)$. Since $G_k$ has no isolated vertices, we derive $\mathtt{wgt}(\mathscr{P}_k) \geq |V(G_k)|$. Thus, we have that $\mathtt{wgt}(\mathscr{P}_k) = |V(G_k)|$ and it follows that $G_k$ is a disjoint union of stars. Hence $(G, k)$ is a yes-instance of Constellation-Inclusive Vertex Splitting. □

**Theorem 5.** Constellation-Inclusive Vertex Splitting *is NP-complete on cubic graphs and on planar graphs of maximum degree three.*

*Proof.* Clearly, Constellation-Inclusive Vertex Splitting belongs to NP. By Lemma 12, an instance $(G, k)$ of Constellation-Inclusive Vertex Splitting is equivalent to the instance $(G, |V(G)| + k)$ of Weighted Star Decomposition. As Weighted Star Decomposition is NP-complete on cubic graphs and on planar graphs of maximum degree three, it follows that Constellation-Inclusive Vertex Splitting is NP-hard on these graph classes. Hence, it is NP-complete on cubic graphs and planar graphs of maximum degree three. □

The proof of Lemma 12 extends to exclusive vertex splitting. The forward and backward implications rely on Lemmas 11 and 10, respectively, both of which hold for exclusive vertex splitting. Hence the above observation gives us the following two results.

**Corollary 13.** *Let $G = (V, E)$ be a graph. Then $(G, k)$ is a yes-instance of* Constellation-Exclusive Vertex Splitting *if and only if $(G, |V(G)| + k)$ is a yes-instance of* Weighted Star Decomposition

**Theorem 6.** Constellation-Exclusive Vertex Splitting *is NP-complete on cubic graphs and on planar graphs of maximum degree three.*

## 6. Conclusion

We proved that Constellation-Inclusive/Exclusive Vertex Splitting is NP-complete on cubic graphs and planar graphs of maximum degree 3. In contrast, we showed that both variants of Cycle Union-Vertex Splitting and Linear Forest-Vertex Splitting are solvable in polynomial time.


## Acknowledgements

Research supported by the Czech Science Foundation Grant no. 24-12046S.


## Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

## References


[1] L. Cai, Fixed-parameter tractability of graph modification problems for hereditary properties, Inf. Process. Lett. 58 (1996) 171–176. doi:10.1016/0020-0190(96)00050-6.

[2] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Springer, 2015. URL: https://doi.org/10.1007/978-3-319-21275-3. doi:10.1007/978-3-319-21275-3.

[3] T. Asano, T. Hirata, Edge-contraction problems, Journal of Computer and System Sciences 26 (1983) 197–208. URL: https://www.sciencedirect.com/science/article/pii/0022000083900120. doi:https://doi.org/10.1016/0022-0000(83)90012-0.

[4] T. Watanabe, T. Ae, A. Nakamura, On the removal of forbidden graphs by edge-deletion or by edge-contraction, Discrete Applied Mathematics 3 (1981) 151–153. URL: https://www.sciencedirect.com/science/article/pii/0166218X81900391. doi:https://doi.org/10.1016/0166-218X(81)90039-1.

[5] F. N. Abu-Khzam, L. Isenmann, Z. Merchad, Bicluster editing with overlaps: A vertex splitting approach, 2025. URL: https://arxiv.org/abs/2505.03959. arXiv:2505.03959.

[6] F. N. Abu-Khzam, S. Thoumi, On the complexity of claw-free vertex splitting, 2025. URL: https://arxiv.org/abs/2506.06044. arXiv:2506.06044.

[7] R. Ahmed, S. Kobourov, M. Kryven, An fpt algorithm for bipartite vertex splitting, 2022. URL: https://arxiv.org/abs/2208.12898. arXiv:2208.12898.

[8] J. Baumann, M. Pfretzschner, I. Rutter, Parameterized complexity of vertex splitting to pathwidth at most 1, Theoretical Computer Science 1021 (2024) 114928. doi:https://doi.org/10.1016/j.tcs.2024.114928.

[9] A. Firbas, Establishing hereditary graph properties via vertex splitting, Ph.D. thesis, Technische Universität Wien, 2023.

[10] A. Firbas, M. Sorge, On the Complexity of Establishing Hereditary Graph Properties via Vertex Splitting, in: 35th International Symposium on Algorithms and Computation (ISAAC 2024), volume 322 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2024, pp. 30:1–30:15. doi:10.4230/LIPIcs.ISAAC.2024.30.

[11] M. Nöllenburg, M. Sorge, S. Terziadis, A. Villedieu, H.-Y. Wu, J. Wulms, Planarizing graphs and their drawings by vertex splitting, Journal of Computational Geometry 16 (2025) 333–372. URL: https://jocg.org/index.php/jocg/article/view/4103. doi:10.20382/jocg.v16i1a10.

[12] W. Tutte, A theory of 3-connected graphs, Indagationes Mathematicae (Proceedings) 64 (1961) 441–455. URL: https://www.sciencedirect.com/science/article/pii/S1385725861500455. doi:https://doi.org/10.1016/S1385-7258(61)50045-5.

[13] L. Faria, C. de Figueiredo, C. Mendonça, splitting number is np-complete, Discrete Applied Mathematics 108 (2001) 65–83. doi:https://doi.org/10.1016/S0166-218X(00)00220-1.

[14] D. Paik, S. Reddy, S. Sahni, Vertex splitting in dags and applications to partial scan designs and lossy circuits, International Journal of Foundations of Computer Science 09 (1998) 377–398. URL: https://doi.org/10.1142/S0129054198000301. doi:10.1142/S0129054198000301. arXiv:https://doi.org/10.1142/S0129054198000301.

[15] J. Gramm, J. Guo, F. Hüffner, R. Niedermeier, H.-P. Piepho, R. Schmid, Algorithms for compact letter displays: Comparison and evaluation, Computational Statistics & Data Analysis 52 (2007) 725–736. doi:https://doi.org/10.1016/j.csda.2006.09.035.

[16] M. Bentert, P. G. Drange, E. Haugen, Overlapping biclustering, 2025. URL: https://arxiv.org/abs/2505.05213. arXiv:2505.05213.

[17] A. Gaikwad, H. Kumar, S. Maity, Parameterized algorithms for editing to uniform cluster graph, 2025. URL: https://arxiv.org/abs/2404.10023. arXiv:2404.10023.

[18] F. N. Abu-Khzam, T. Davot, L. Isenmann, S. Thoumi, On the complexity of 2-club cluster editing with vertex splitting, in: Computing and Combinatorics, Springer Nature Singapore, Singapore, 2026, pp. 3–14.

[19] M. R. Garey, D. S. Johnson, L. Stockmeyer, Some simplified np-complete problems, in: Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, STOC '74, Association for Computing Machinery, New York, NY, USA, 1974, p. 47–63. URL: https://doi.org/10.1145/800119.803884. doi:10.1145/800119.803884.

[20] I. Koch, N. Pardal, V. F. dos Santos, Edge deletion to tree-like graph classes, Discrete Applied Mathematics 348 (2024) 122–131. doi:https://doi.org/10.1016/j.dam.2024.01.028.

[21] J. M. Lewis, M. Yannakakis, The node-deletion problem for hereditary properties is np-complete, Journal of Computer and System Sciences 20 (1980) 219–230. URL: https://www.sciencedirect.com/science/article/pii/0022000080900604. doi:https://doi.org/10.1016/0022-0000(80)90060-4.

[22] Y. Liu, M. Xiao, Solving co-path/cycle packing and co-path packing faster than 3k, Information and Computation 309 (2026) 105406. URL: https://www.sciencedirect.com/science/article/pii/S0890540126000040. doi:https://doi.org/10.1016/j.ic.2026.105406.

[23] T. Watanabe, T. Ae, A. Nakamura, On the np-hardness of edge-deletion and -contraction problems, Discrete Applied Mathematics 6 (1983) 63–78. URL: https://www.sciencedirect.com/science/article/pii/0166218X83901014. doi:https://doi.org/10.1016/0166-218X(83)90101-4.

[24] A. Levin, D. Paulusma, G. J. Woeginger, The complexity of graph contractions, in: H. L. Bodlaender (Ed.), Graph-Theoretic Concepts in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 322–333.

[25] F. N. Abu-Khzam, J. Egan, S. Gaspers, A. Shaw, P. Shaw, Cluster editing with vertex splitting, in: Combinatorial Optimization, Springer International Publishing, Cham, 2018, pp. 1–13.

[26] F. N. Abu-Khzam, D. Chakraborty, L. Isenmann, N. Oijid, On the complexity of vertex-splitting into an interval graph, in: F. Foucaud, A. Parreau (Eds.), Combinatorial Algorithms, Springer Nature Switzerland, Cham, 2026, pp. 1–15.

[27] J. Edmonds, E. L. Johnson, Matching, euler tours and the chinese postman, Mathematical programming 5 (1973) 88–124.

[28] D. B. West, Introduction to graph theory, 2001.

[29] R. Diestel, Graph theory 5th ed, Graduate texts in mathematics 173 (2017).

[30] D. Lichtenstein, Planar formulae and their uses, SIAM Journal on Computing 11 (1982) 329–343. URL: https://doi.org/10.1137/0211025. doi:10.1137/0211025. arXiv:https://doi.org/10.1137/0211025.