Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 29, no. 1, pp. 63–90 (2025)
DOI: 10.7155/jgaa.v29i1.3029

Correcting a Graph Into a Linegraph Minimizing Hamming Distance Edition Is NP-Complete and FPT by Treewidth

Dominique Barth[1] *Dimitri Watel*[2,3] *Marc-Antoine Weisser*[4]

[1]DAVID, University of Versailles-St Quentin, Versailles, France
[2]ensIIE, Evry, France
[3]SAMOVAR, Evry, France
[4]LISN, CentraleSupelec, Paris-Saclay University, France

Submitted: January 2024 Accepted: April 2025 Published: April 2025

Article type: Regular paper Communicated by: Meirav Zehavi

Abstract. Since Beineke's work in 1968 on linegraphs, attention has focused on the classification of graphs as linegraphs. It is known that every graph G is the linegraph of an hypergraph, and the question is to characterize that root graph. We introduce the $C_{p,q}$ classes, defined as sets of graphs where each vertex can be covered by at most p cliques, and each edge belongs to at most q cliques. These classes provide a comprehensive classification of linegraphs through a unified and parameterized approach. They describe previously known graph classes - such as linegraphs of simple graphs, p-uniform hypergraphs and p-uniform 1-linear hypergraphs - while being capable of generalization. We study the complexity of determining the membership and edit distance of a graph to one of these classes. We prove the first Fixed Parameter Tractable algorithm with respect to treewidth to compute the edit distance.

1 Introduction

The linegraph $L(G)$ of a (hyper) graph $G = (V, E)$ has a set of nodes E and two nodes of $L(G)$ are connected if the intersection of the two corresponding (hyper) edges in G is not empty [21]. The link between a (hyper) graph and its linegraph has been widely studied. Various works, for example, have been devoted to determining whether a graph is a linegraph and, if so, to discovering the graph of which it is the linegraph (referred to as the root graph of the linegraph) [2, 12, 28, 34, 39].

In this work, a characterization of linegraphs is provided. It gathers and uniformizes, among others, various classes of linegraphs including those of simple graphs, multigraphs, and hypergraphs. We also address the question of the edit distance from any graph to any graph belonging to this class. These two subjects are related, but for the sake of clarity, we will present them one after the other, introducing the state of the art and our results each time.

E-mail addresses: dominique.barth@uvsq.fr (Dominique Barth) dimitri.watel@ensiie.fr (Dimitri Watel) marc-antoine.weisser@centralesupelec.fr (Marc-Antoine Weisser)

1.1 Characterization

A characterization of the linegraphs of simple graphs was introduced by Beineke [2, 3]. It relies on a on a finite set of forbidden induced subgraphs. By using different sets of forbidden subgraphs, this method has been extended to characterize the classes of linegraphs of multigraphs [8], multigraphs with restricted multiplicities of edges [41], bipartite graphs [20] or bipartite multigraphs [42]. An algorithm [46] has been developed to generate a set of forbidden induced subgraphs for characterizing linegraphs belonging to any hereditary class[1].

The situation is more intricate for linegraphs of hypergraphs. There is no finite set of forbidden induced subgraphs that can be used to characterize linegraphs of 3-uniform[2] hypergraphs [7]. However there exists a finite set for linegraphs of 3-uniform linear[3] hypergraphs whose vertex degree it at least 19 [32]. The complexity of determining whether a graph G belongs to linegraphs of r-uniform hypergaphs is NP-complete [29], where both G and r are inputs. For 3-uniform hypergaphs, the question is open[4] but it is known it is NP-complete for the linegraphs of a 3-uniform 1-linear hypergraphs [33].

Another approach to characterize linegraphs is based on clique covering. A graph H is a linegraph of a simple graph[5] if and only if the edges of H can be parted into a set of cliques, where each vertex of the graph belongs to at most two of these cliques [27, 28, 43]. We can generalize this characterization by allowing a vertex to appear in an arbitrary number p of cliques. If $p = 1$, we characterize the cluster graphs, the union of complete graphs, which are the linegraphs of union of stars. Determining whether a graph belongs to this class is trivially polynomial. If $p \geq 3$, we characterize the linegraphs of 1-linear p-uniform hypergraphs [5]. Determining if a graph belongs to this class is NP-complete [36].

If the edge-constraint is relaxed, that is if we still search for a covering of the edges with cliques but an edge may belong to more than one clique, then the linegraphs of all p-uniform hypergraphs can be characterized, including those that are not 1-linear. For $p = 2$, it has been demonstrated that determining the membership of a graph in this class is polynomial problem [23, 24, 44]. However, for $p \geq 4$, it becomes an NP-complete problem [36]. The question of membership for $p = 3$ remains an open problem, and its complexity is yet to be determined.

In these classifications, the parameter p can vary gradually but either the cliques are required to be edge-disjoint, meaning each edge is covered by a unique clique, or no constraint is imposed, allowing an edge to be covered by up to p cliques. We propose introducing a new class denoted as $\mathcal{C}_{p,q}$ $(1 \leq q \leq p)$

Definition 1. *A graph $G = (V, E)$ belongs to the class $\mathcal{C}_{p,q}$ if and only if E can be covered by a set of cliques*

- *each vertex appears in at most p cliques*
- *each edge appears in at most q cliques*

[1] A class of graph X is hereditary if and only if $G \in X \Rightarrow G - v \in X$ for any vertex v of G where $G - v$ denotes the graph obtained from G by removing v and all edges incident to it.

[2] a hypergraph $G = (V, \mathcal{E})$ is p-uniform if $\forall E \in \mathcal{E}, |E| = p$.

[3] a hypergraph $G = (V, \mathcal{E})$ is l-linear if $\forall E_i, E_j \in \mathcal{E}, |E_i \cap E_j| \leq l$. A 1-linear hypergraph is simply called linear.

[4] The reduction of [29] is based on the partition of a linegraph into r cliques. One might be tempted to fix $r = 3$, since in the general case, partitioning an arbitrary graph into 3 cliques is NP-complete. However, in our case, the question is to partition a linegraph, and, to the best of our knowledge, no result addresses this specific class of graphs.

[5] A simple graph is a 2-uniform 1-linear hypergraph

$\mathcal{C}_{p,q}$	$p=1$	$p=2$	$p=3$	$p\geq 4$
$q=1$	P (trivial)	P [2, 3]	NP-c [36]	NP-c [36]
$q=2$		P [23, 24, 44]	***NP*-c** (Th. 1)	***NP*-c** (Th. 2)
$q=3$			?	?
$q\in[\![4,p]\!]$				NP-c [36]

Table 1: Complexity of determining if a graph belongs to a given $\mathcal{C}_{p,q}$ class. The results in bold are proven in this paper. For $\mathcal{C}_{p,3}$ with $p\geq 3$, problems are in NP (Lemma 1) but the question remains open for the NP-completeness.

This class corresponds to the linegraphs of p-uniform q-linear hypergraphs. This classification of graphs has proven to be valuable in various fields. One application is the determination of the minimum value of p for a graph to belong to $\mathcal{C}_{p,p}$, which is known as the local clique problem. This problem is equivalent to finding the intersection number of an intersection graph [37] which is NP-complete [35, 26].

The complexity of two classes remains an open question: $\mathcal{C}_{p,2}$ for all $p\geq 3$ and $\mathcal{C}_{p,3}$ for all $p\geq 3$. In this paper, we provide a proof for the complexity of the first class. The complexity results known to date, as well as those introduced in this paper, are summarized in Table 1.

Extending clique covering to allow each edge to appear in a maximum of q cliques is a natural extension. A similar approach is presented in [16], but in that case, the cliques must necessarily be maximum cliques. Other variants can also be found. For example, in [22], the cliques in the covering can be partially incomplete. These variations demonstrate the flexibility and applicability of clique covering in different contexts.

1.2 Edit Distance

While the characterization of graphs is significant, numerous applications focus on the proximity of a given graph to a specific class of graphs. The main metric for such proximity is the edit distance [13]. For two graphs G and G', it is denoted as $\Delta(G,G')$, and defined as the minimum number of edge deletions or additions needed to transform G into G' (since G and G' are undirected, it is half of the Hamming distance between their adjacency matrices). In this context, the problem we consider is the following:

Problem 1 (ED-$\mathcal{C}_{p,q}$)**.** *Given a graph G and two positive integers $p\in\mathbb{N}$ and $q\in[\![1,p]\!]$, find a graph $G'\in\mathcal{C}_{p,q}$ such that the edit distance $\Delta(G,G')$ is minimum.*

If $p=1$ and $q=1$, this is the cluster editing problem. It has applications in machine learning [1], data mining [11] and bioinformatics [31]. The problem has been shown to be NP-complete [4, 40] but admits a polynomial time algorithm if the distance is bounded by a parameter k [9].

If $p=2$ and $q=1$, this is the linegraph reconstruction problem. It has applications in networks. For instance, in the context of energy distribution networks, operators can infer the underlying topology of the network from electrical flow measurements by deducing correlations between links. This inference allows them to construct a linegraph L, which, in turn, helps in inferring the root graph G representing the distribution network [10, 19]. Challenges arise when measurement errors occur, causing L to deviate from being a valid linegraph. In such cases, it becomes necessary to correct L in order to obtain a valid linegraph [14, 15].

The linegraph reconstruction problem has been previously studied with slightly different formulations. In [18, 45], only edge deletions are allowed, while in [14], only edge additions are

$\mathcal{C}_{p,q}$	$p=1$	$p=2$	$p=3$	$p\geq 4$
$q=1$	NP-c [4, 40]	?	NP-c^*	NP-c^*
$q=2$		?	?	?
$q=3$			?	?
$q\in[\![4,p]\!]$				NP-c^*

Table 2: Complexity of the editing problem ED-$\mathcal{C}_{p,q}$ restricted to instances where p and q are fixed. Prior to our work, most of the cases were not addressed, except the case $p=q=1$ and the consequences of known results for the corresponding characterization problems (see Table 1). We demonstrate, with a unified perspective, that the problem is NP-complete for all cases Th. 3.

considered. In both cases, the problems have been shown to be NP-complete. However, an open question remains: Is the problem still NP-complete when both edge deletions and additions are allowed simultaneously? In this paper, we provide a proof that the problem is indeed NP-complete in this extended setting.

Finally, the editing problem is NP-complete for a given class $\mathcal{C}_{p,q}$ if the recognition problem for that class is also NP-complete. This was known for $\mathcal{C}_{3,1}$, $\mathcal{C}_{4,1}$ and $\mathcal{C}_{4,4}$ [36] but most of the results were missing (see Table 2). In this paper, we demonstrate, with a unified perspective, that the problem is NP-complete for all cases, including the open question of linegraphs: $\mathcal{C}_{2,1}$.

1.3 Parameterized Complexity

Given the complexity of the edit distance problems, it is natural to explore whether the problem becomes tractable for certain fixed parameter values.

A first parameter is k, the number of editions. For the $\mathcal{C}_{1,1}$ class, an FPT-algorithm [17] exists based on bounded tree search. The time complexity of this algorithm is $\mathcal{O}(2.77^k+|V|^3)$. However, to the best of our knowledge, there are no known FPT-algorithms for the $\mathcal{C}_{2,1}$ and $\mathcal{C}_{2,2}$ classes. It is worth noting that, although specific FPT-algorithms may not be known for these classes, the fact that they can be characterized by forbidden graphs allows for the construction of straightforward FPT-algorithms using bounded tree search. As a consequence of the NP-completeness of characterization problems, there are no FPT-algorithms with respect to k, p and q for $\mathcal{C}_{p,q}$ with $p\geq 3$, except possibly for the case where $q=3$ which remains an open question.

Another parameter is the size of maximum cliques. However, the cliques involved in reductions to prove NP-completeness results for the corresponding characterization problems are generally small. As a result, the size of maximum cliques is not a useful parameter for creating XP or FPT-algorithms.

The last classical parameter that can make the problem tractable is the treewidth [38]. To the best of our knowledge, there are no existing results using it. A natural question is whether Courcelles's theorem can be used to deduce the existence of an FPT algorithm with respect to the treewidth. In other words, is it possible to rewrite the problem as an MSO formula? Such a formulation would allow for the recognition of a linegraph (of a simple graph, that is a graph in $\mathcal{C}_{2,1}$) with an MSO formula. To our knowledge, no such formula has ever been given. A naive way to build such a formula would be to determine these cliques. We would then have a formula starting with $\exists V1\subset V, V2\subset V\ldots$. However, the number of cliques is not bounded by the treewidth. Then, it does not seem straightforward to write an n-independent size MSO formula that captures the fact that a graph is a linegraph of a graph.

This is why, in this paper, we construct a dynamic programming algorithm to answer the question. We first demonstrate that for given values of p and the treewidth, the size of the maximum cliques in the edited graph is bounded. This allows us to construct a dynamic programming algorithm that is the first FPT-algorithm with respect to the treewidth.

1.4 Paper Organization

As we have just presented, this paper compiles and completes the results concerning the complexity of characterizing linegraphs, editing distance and the associated FPT-algorithms. In the remaining sections of the paper, we will establish the following new results.

In Section 2, we will demonstrate that characterizing graphs belonging to the class $\mathcal{C}_{3,2}$ is an NP-complete problem. In Section 3, we provide further insights into the complexity of editing problems, thereby completing the existing knowledge in this area. Specifically, we establish the NP-completeness of the edit distance problem for $\mathcal{C}_{2,1}$ and subsequently extend this result to the classes $\mathcal{C}_{2,2}$ and $\mathcal{C}_{p,3}$ for $p \geq 3$. Finally, Section 4 focuses on parameterized complexity, where we present a dynamic programming algorithm that depends on both p and the treewidth.

2 NP-Completeness of Recognizing a Graph in $\mathcal{C}_{3,2}$

To address the NP-completeness of the recognition problem, we need to prove that determining if a graph belongs to $\mathcal{C}_{3,2}$ is in NP. We prove a more general result that we will use for other cases.

Lemma 1. *Given two integers p and q, determining if a graph G belongs to $\mathcal{C}_{p,q}$ is in NP.*

Proof: If a graph $G = (V, E)$ belongs to a class $\mathcal{C}_{p,q}$, then there exists a set of labels L and a function λ that associates each vertex with a nonempty subset of L, such that

1. $\forall l \in L$, the subgraph induced by $\{v \in V / l \in \lambda(v)\}$ is a clique;
2. $\forall v \in V$, $|\lambda(v)| \leq p$;
3. $\forall \{u, v\} \in E$, $1 \leq |\lambda(u) \cap \lambda(v)| \leq q$.

If such a labeling exists, then each vertex is covered by at most p cliques (conditions 1 and 2), and each edge belongs to at most q cliques (condition 3). This labeling is a certificate that can be verified in polynomial time. Note that, as each vertex cannot appear in more than p cliques, the size of L can be bounded by $n \cdot p$, and thus has a polynomial size. □

Consider now the case $p = 3$ and $q = 2$. We present a polynomial reduction from 2P2N-3-SAT, a variant of 3-SAT where each variable appears twice positively and twice negatively, to the problem of recognizing if a graph belongs to $\mathcal{C}_{3,2}$, that is if we can cover all the edges with cliques such that every node is in at most three cliques and every edge in at most two. 2P2N-3-SAT is NP-Complete [6]. Let $\phi = (X, C)$ be a 2P2N-3-SAT formula with a given set of variables $X = (x_1, x_2, \ldots, x_n)$ and a given set of clauses $C = (c_1, c_2, \ldots, c_m)$. We build a graph G as follows.

We first consider, as done in [36], the wheel W_6 *i.e.* a cycle of size 6 in which all nodes are connected to a central seventh node. This node must be covered by three non-consecutive triangles. We build, for each variable $x_i \in X$ a gadget by overlapping many wheels and three cliques of size 4 as pictured in figure 1. There are two possible coverings of the gadget, that are used to encode the truth value of the variable. The gadget contains four boundary nodes v^i_j corresponding to each of

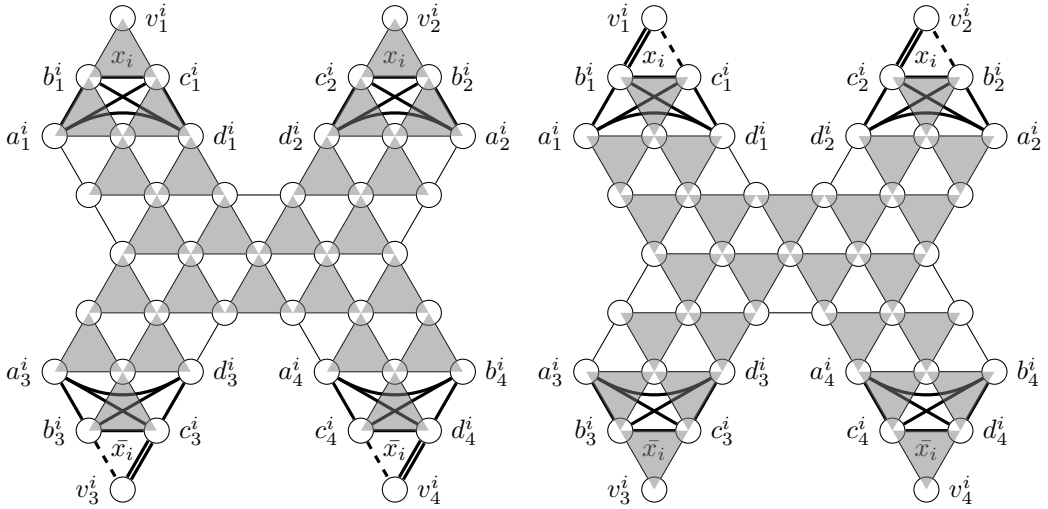


Figure 1: Gadget for a variable x_i in the reduction from 2P2N-3-SAT to the recognition of a graph in $C_{3,2}$. This variable appears twice positively in c_1 and c_2 and twice negatively in c_3 and c_4. The triangles in the two possible coverings of the gadgets are drawn in gray. To those triangles, we must add the three cliques of size 4 $(a_j^i, b_j^i, c_j^i, d_j^i)$ for $j \leq 3$. If (b_j^i, c_j^i) is already covered by a gray triangle, then we must also cover it with the clique (otherwise d_3^i will be covered by four cliques). Thus, we cannot cover that edge with the triangle (b_j^i, c_j^i, v_j^i). Thus the edges (b_j^i, v_j^i) and (c_j^i, v_j^i) may be covered by a triangle if (b_j^i, c_j^i) is not already covered by another triangle in the wheel to which it belongs, otherwise we must cover the two edges by two independent cliques.

the four literals of the variable in the formula, where j is the index of the clause containing that literal.

For each clause c_j, we create a linking gadget that connects the variable gadgets together. It consists of two vertices y_j and y'_j connected as shown in Figure 2.

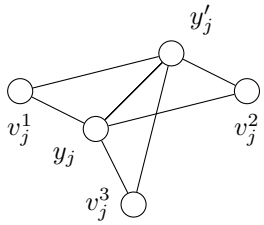


Figure 2: Gadget for a clause $C_j = (x_1 \vee x_2 \vee x_3)$ in the reduction from 3SAT to the recognition of a graph in $C_{3,2}$.

Theorem 1. *Determining if a graph G belongs to $\mathcal{C}_{3,2}$ is NP-Complete.*

Proof: The problem belongs to NP, as stated in Lemma 1. The previously described reduction is done in polynomial time, we now prove its correctness.

Note first that the variable gadget imposes that in any feasible covering, for any two clauses c_{j_1} and c_{j_2} containing respectively the literals x_i and $\bar{x_i}$, the node $v^i_{j_1}$ may be covered by the triangle $(b^i_{j_1}, c^i_{j_1}, v^i_{j_1})$ if and only if $v^i_{j_2}$ cannot be covered by $(b^i_{j_2}, c^i_{j_2}, v^i_{j_2})$. It must instead be covered by the two edges $(b^i_{j_2}, v^i_{j_2})$ and $(c^i_{j_2}, v^i_{j_2})$.

Assuming there exist truth values for the variables in X that satisfy all the clauses. We build a valid covering the following way. For each true variable, we use the version on the left of Figure 1 to cover the corresponding gadget so that the triangles (b^i_j, c^i_j, v^i_j) labeled with x_i on the figure belong to the covering. The right version is used if the variable is false. Let c_j be a clause of the formula. At least one literal x_i or $\bar{x_i}$ in c_j is true, meaning that the corresponding node v^i_j is covered by only one triangle. We add the edges (v^i_j, y_j) and (v^i_j, y'_j) to the covering. For each the two other literals x_i or $\bar{x_i}$ in c_j, we add the triangle (v^i_j, y_j, y'_j) to the covering whatever the truth value of that literal is. Every node is in at most three cliques and every edge is in at most two. Then G is in $\mathcal{C}_{3,2}$.

We now assume that there exists a valid covering of G. We set x_i to true if for at least one clause c_j containing the literal x_i, the triangle (b^i_j, c^i_j, v^i_j) belongs to the covering. Note that the variable gadget is such that, if the literal $\bar{x_i}$ is in c_j and if the covering contains (b^i_j, c^i_j, v^i_j), then x_i is necessarily false. Let c_j be a clause, we want to prove that one of the three literals of c_j is true. Without loss of generality, we assume that $c_j = (\bar{x_1} \vee x_2 \vee x_3)$. If $\bar{x_1}, x_2$ and x_3 are all false, then the covering must contain the cliques (b^i_j, v^i_j) and (c^i_j, v^i_j) for $i \leq 3$. As v^i_j is already covered by two cliques, the covering cannot contain the edges (v^i_j, y_j) and (v^i_j, y'_j) and must instead contain the triangle (v^i_j, y_j, y'_j) for $i \leq 3$. The edge (y_j, y'_j) is then covered with three cliques which is a contradiction. Thus either $\bar{x_1}, x_2$ or x_3 is true and formula is satisfied. □

Remark 1. *We cannot use a similar proof in order to prove that the recognition of $\mathcal{C}_{3,3}$ is NP-Complete because there is no longer constraint on the edges of the graph (if we satisfy the constraint on the nodes). In this reduction, the constraint on the edges $\{b^i_j, c^i_j\}$ and $\{y_j, y'_j\}$ is essential.*

The reduction cannot be modified to handle cases where $q \geq 3$. On the contrary, it can be extended for $\mathcal{C}_{p,2}$ with $p \geq 4$ by artificially increasing the number of cliques required to cover each node.

Theorem 2. *Determining if a graph G belongs to $\mathcal{C}_{p,2}$ is NP-Complete for $p \geq 4$.*

Proof: The reduction used for $\mathcal{C}_{3,2}$ can be modified by adding one final step. For each vertex v of the graph, we create $p-3$ new vertices v_i and connect them to v with $p-3$ edges $\{v, v_i\}$. Each new edge must be covered by a new clique. The number of cliques covering v is increased by $p-3$. □

3 NP-Completeness of Edit Distance to $\mathcal{C}_{p,1}$ and $\mathcal{C}_{p,q}$

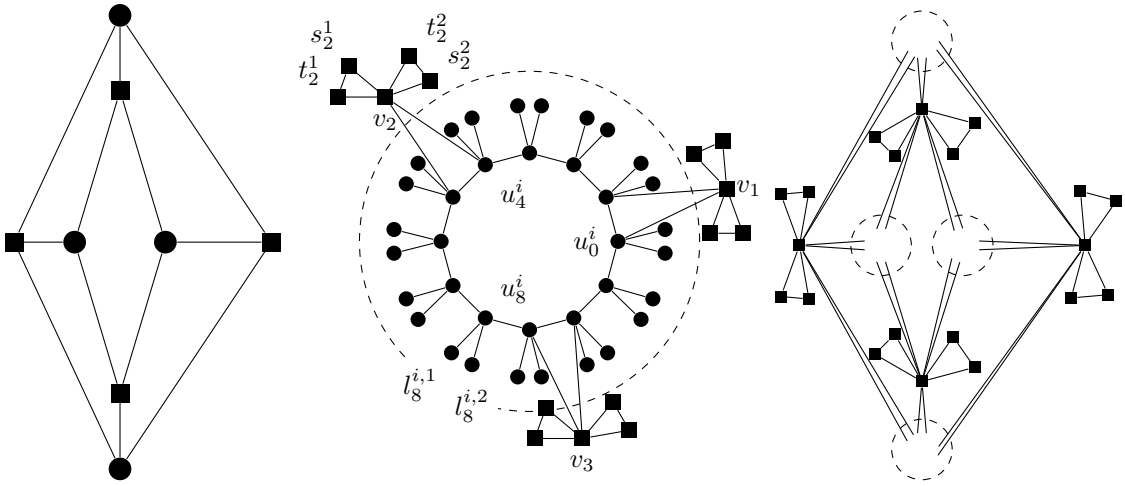
This section is dedicated to the NP-completeness of ED-$\mathcal{C}_{p,q}$, the edit distance to a class $\mathcal{C}_{p,q}$. We introduce the Planar 3-SAT problem, which is NP-complete, and a polynomial time reduction from Planar 3-SAT to ED-$\mathcal{C}_{p,1}$ for $p \geq 1$. We then generalize this result for larger values of q.

Let $\phi = (X, C)$ be a 3-SAT formula with a given set of variables X and a given set of clauses C. Let G_ϕ be a graph whose vertices are $V_\phi = X \cup C$ and the set of edges E_ϕ is such that for each variable $x \in X$ appearing in a clause $c \in C$ (negated or not) there exists an edge $\{x, c\}$. The graph G_ϕ is called the incidence graph of ϕ. The planar 3-SAT problem is, given a formula ϕ and an

incidence graph G_ϕ with a planar embedding, to determine whether ϕ is satisfiable. This problem is NP-complete [30].

We present a reduction from an instance $\phi = (X, C)$ of Planar 3-SAT to an instance of ED-$\mathcal{C}_{p,1}$, a graph $G = (V, E)$ that is built as follows. For each clause $c_j \in C$, we create a node v_j connecting $p-1$ triangles of nodes: $v_j, s_j^k, t_j^k \ \forall k \in [\![1, p-1]\!]$ (this means no triangle for $p = 1$).

By definition, a Planar 3-SAT instance can be represented with a planar embedding. Given an embedding, for a given variable x_i, we denote by $C(x_i)$ the clauses containing x_i, sorted by cyclic ordering of the neighbours in the embedding. Let $m_i = |C(x_i)|$. We create in G a cycle of size $4m_i$: $u_0^i, u_1^i, \ldots, u_{4m_i-1}^i, u_0^i$. Let c_j be the r-th clause containing the variable x_i. If it contains the literal x_i, we add two edges (u_{4r}^i, v_j) and (u_{4r+1}^i, v_j), otherwise, if the clause c_j contains the literal $\overline{x_i}$, we add two edges (u_{4r+1}^i, v_j) and (u_{4r+2}^i, v_j). Finally, for each node u_j^i of the cycle, we create $p-1$ leaves $l_j^{i,1}, \ldots, l_j^{i,p-1}$ and add $p-1$ edges $(u_j^i, l_j^{i,1}), \ldots, (u_j^i, l_j^{i,p-1})$ (no leaves for $p = 1$). Finally we write $m = |C|$ and we define $k = 10m$. The gadgets of this reduction are represented on Figure 3.



(a) An embedding of a Planar 3-SAT instance. Clauses are square nodes and variables are circle nodes.

(b) Inside the dashed circle, the gadget for a variable x_i belonging to three clauses c_1, c_2 and c_3. The literals are x_i in c_1, c_2 and $\bar{x}_i$ in c_3. The square nodes are part of clause gadgets (Fig. 3c).

(c) The instance of ED-$\mathcal{C}3, 1$ obtained by reduction. Each dashed circle represents a variable gadget (Fig. 3b). The squares are the clause gadgets.

Figure 3: Example of a reduction for $p = 3$.

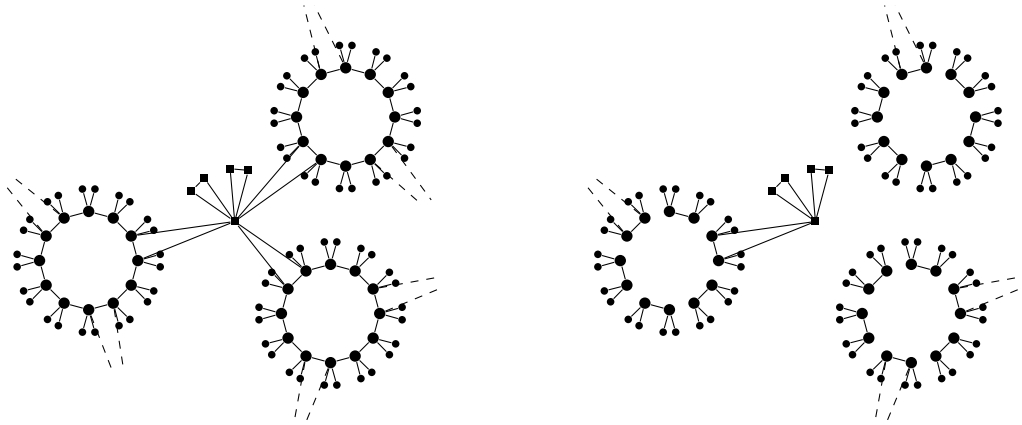
By definition, a Planar 3-SAT instance has a planar embedding. The reduction satisfies the cyclic ordering of such embedding, it only consists in replacing the clause nodes by triangle graphs and the variable nodes by cycles with leaves. Thus, the graph produced by the reduction also has a planar embedding.

Suppose there exists an assignment of variables satisfying ϕ. We can edit G with the following algorithm to obtain a new graph G' with edit distance $\Delta(G, G') \leq 10m$.

1. For each gadget corresponding to a variable x_i assigned to true, we delete all the edges $(u_{2j+1}, u_{2j+2}), \forall j \in [\![0, 2m_i - 1]\!]$.
2. For each gadget corresponding to a variable x_i assigned to false, we delete all the edges $(u_{2j}, u_{2j+1}), \forall j \in [\![0, 2m_i - 1]\!]$.

3. As ϕ is satisfied, for each clause c_j, at least one of the three literals is true (if there are more, we choose one arbitrarily). We delete the four edges connecting v_j to the gadgets of the two other literals.

We illustrate this algorithm on Figure 4. At steps one and two of this algorithm, for each variable x_i, we delete $2m_i$ edges of the corresponding gadget. As each clause contains exactly three literals, $\sum_i m_i = 3m$. The total number of edges deleted during these two steps is $6m$. At step three, $4m$ edges are deleted, 4 edges for each clause. Thus, the edit distance $\Delta(G, G') = 10m$.



(a) A part of the graph G: the clause gadget is connected to three variable gadgets. The dashed edges indicate that they may be connected to other clauses.

(b) A possible correction of the graph G. Only one variable gadget remains connected to the clause gadget. One edge out of two of the cycle of a variable gadget is deleted.

Figure 4: Example of correction for $p = 3$.

The graph G' belongs to $\mathcal{C}_{p,1}$. Each node v_j belongs to p triangles (the triangle of the clause gadget $\{v_j, s_j^k, t_j^k\}$ for $k \in [\![1, p-1]\!]$ and the triangle connecting v_j to one variable gadget: $\{v_j, u_{4r+1}^i, u_{4r+2}^i\}$ or $\{v_j, u_{2r+1}^i, u_{2r+2}^i\}$ depending on the literal of x^i contained in c_j). Each node u_j^i either belongs to a triangle and $p-1$ edges, or it belongs to p edges. Each edge belongs to a unique clique: a triangle or an edge.

Suppose now that it is possible to correct G by editing at most $10m$ edges. Each gadget of a variable x_i contains a cycle of size $4m_i$. Each node u_j^i of this cycle is covered by $p+1$ cliques: two cliques containing $\{u_{j-1}^i, u_j^i\}$ or $\{u_j^i, u_{j+1}^i\}$ and possibly a vertex $v_{j'}$ for some clause $c_{j'}$, and $p-1$ cliques containing $\{u_j, l_j^{i,k}\}$ $\forall k \in [\![1, p-1]\!]$. We need to remove $4m_i$ cliques. Adding an edge or deleting an edge not in the cycle deletes at most one clique, while deleting an edge of the cycle deletes two cliques. Therefore, $2m_i$ editions are required, and the only solution with that many edits consists in deleting one edge out of two from the cycle. In total, we thus need at least $6m$ editions on the variable gadgets. For those editions, we build an assignment of the variables as follows: if the edges (u_{2j+1}^i, u_{2j+2}^i) have been deleted for every $j \in [\![0, m_i - 1]\!]$ then x_i is set to true, else it is set to false.

As our solution corrects the graph with at most $10m$ editions, it remains $4m$ editions. Each vertex v_j belongs to $p-1$ triangles containing $\{s_j^k, t_j^k\}$ $\forall k \in [\![1, p-1]\!]$. For each literal x_i in c_j, if x_i is set to true, the vertex v_j belongs to one clique, the triangle $\{v_j, u_{4j}^i, u_{4j+1}^i\}$. Otherwise, it belongs to two cliques, the two edges $\{v_j, u_{4j}^i\}$ and $\{v_j, u_{4j+1}^i\}$. Similarly for a negative literal. As a consequence, first, at least 4 deletions are needed to keep v_j in exactly p cliques. Since we are only allowed to remove $4m$ edges, we must remove exactly 4 incident edges to v_j. Secondly, for each false literal, we must remove the two edges connecting v_j to the corresponding cycle. As a consequence, at most two literals of the clause are set to false by the assignment and the formula ϕ is satisfied.

This concludes our polynomial time reduction. Notice that the maximum degree of node is $2p+4$ (for nodes v_j).

Theorem 3. *Given two integers $p \geq 1$ and $q \in [\![1, p]\!]$, ED-$\mathcal{C}_{p,q}$ is NP-Complete even if G is planar and the degree of node is less than $2p+4$.*

Proof: As stated by Lemma 1, given $p \geq 1$, ED-$\mathcal{C}_{p,1}$ is in NP. Moreover, there exists a polynomial time reduction from Planar 3-SAT to ED-$\mathcal{C}_{p,1}$ restricted to planar graphs with degree maximum $2p+4$.

It can be observed that in the reduction for ED-$\mathcal{C}_{p,1}$, an edge always belongs to at most one clique. Therefore, this reduction remains valid for the problem ED-$\mathcal{C}_{p,q}$. □

4 FPT with Respect to Treewidth

This section is dedicated to proving that ED-$\mathcal{C}_{p,q}$ is FPT with respect to p and the treewidth [38] of the input graph.

4.1 Notations

Given two values $p \geq 1$ and $q \in [\![1, p]\!]$, let $G = (V, E)$ be an instance of ED-$\mathcal{C}_{p,q}$. Let τ be a *tree decomposition* of G. In order to avoid any confusion, a node of τ will be called a *bag*. We recall that τ is a tree, that every node of V belongs to at least one bag, that for each edge $(v, w) \in E$, there exists a bag of τ containing v and w, and that the subgraph of τ induced by all the bags containing a same node v is connected. For each bag u of τ, we define X_u as the set of nodes of V contained in the bag.

Without loss of generality, we consider that τ is a *nice tree decomposition* [25], *i.e.* it is a rooted binary tree with the following types of nodes.

- If r is the root or a leaf of τ, then $|X_r| = 0$.
- If u has exactly two children u_1 and u_2, then $X_u = X_{u_1} = X_{u_2}$. We say u is a *join bag*.
- If u has exactly one child u' then
  - either there exists $v \in V$ such that $X_u = X_{u'} \cup \{v\}$ (u is a *introduce bag*),
  - or there exists $v \in V$ such that $X_{u'} = X_u \cup \{v\}$ (u is a *forget bag*).

The width of a tree decomposition is the size of its largest bag minus one. The treewidth of a graph G is the minimum width among all possible tree decompositions of G. It is possible to

build, from an optimal decomposition, a nice decomposition that is also optimal, with $O(|V|)$ bags in linear time [25].

Given a rooted nice decomposition, we define Y_u as the union of the nodes of G in all the bags descendant from bag u in τ (including u) and Z_u as the nodes in all ancestor bags (including u). The treewidth of G is $tw = \max_{u \in \tau} |X_u| - 1$.

4.2 Labeling

We define another way of representing the set of feasible solutions. Each node $v \in V$ can be associated with a nonempty set $\lambda(v)$ of at most p labels in $[\![1, p \cdot n]\!]$ that identifies the cliques that should cover v after correcting G. Figure 5 provides an example. Two nodes that share the same label belong to the same clique and must be connected by an edge. Conversely, if they do not share a label, they never appear in the same clique and should not be connected by an edge. An edge (u, v) belongs to c cliques if nodes u and v share c labels.

Given a labeling of the graph G, we can deduce which edges should be added or removed to transform G into a graph composed of cliques. Then, we can check if no node belongs to more than p cliques and if no edge belongs to more than q cliques.

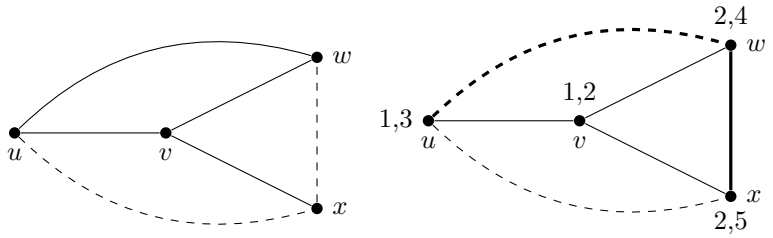


Figure 5: Labeling example with $p = 2$ and $q = 1$. One edge is removed (as there is no common label for u and w) and one edge is added to complete the clique numbered with the label 2.

Definition 2. *Let $X \subset V$, a pq-labeling of X is a function λ associating each vertice of X to a nonempty subset of at most p integers of $[\![1, p \cdot |X|]\!]$ (called* labels*). We write $\lambda^{-1}(l)$ the set of nodes of X labeled with l (in other words, $l \in \lambda(v)$) and $\lambda^{-1}(t) = \bigcap_{l \in t} \lambda^{-1}(l)$ for any tuple t of labels.*

This labeling is feasible *if, for every $(q+1)$-tuple t of labels, $\left|\lambda^{-1}(t)\right| \leq 1$. The weight $\omega(\lambda)$ of λ is*

$$\begin{aligned}\omega(\lambda) = &|\{u, v \in X^2, u \neq v \text{ such that } (u, v) \in E \text{ and } \lambda(u) \cap \lambda(v) = \emptyset\}| \\ &+ |\{u, v \in X^2, u \neq v \text{ such that } (u, v) \notin E \text{ and } \lambda(u) \cap \lambda(v) \neq \emptyset\}|\end{aligned}$$

If λ is not feasible then $\omega(\lambda) = +\infty$.

Property 1. *The edit distance of $G = (V, E)$ to $\mathcal{C}_{p,q}$ is $\min_{\text{feasible } \lambda} \omega(\lambda)$.*

If no ambiguity is possible, we will use the word labeling instead of *pq*-labeling. For readability, we also use the following notations: given two (possibly equal) sets X and Y, we define

- $\omega(\lambda, X, Y)$ as $\omega(\lambda)$ restricted to the couples of $X \times Y$

- $a(X,Y)$ as $|\{(u,v) \in X \times Y, u \neq v \text{ such that } (u,v) \notin E\}|$
- $r(X,Y)$ as $|\{(u,v) \in X \times Y, u \neq v \text{ such that } (u,v) \in E\}|$

Note that, if $X \cap Y = \emptyset$, we have $\omega(\lambda, X, Y) \leq |X| \cdot |Y|$ and $a(X,Y) + r(X,Y) = |X| \cdot |Y|$.

4.3 Main Idea of the Algorithm

The dynamic programming algorithm searches for a minimum weight labeling of V. Let u be a bag of a tree decomposition of G. Given a labeling λ of X_u, we recursively compute the optimal labeling of Y_u that extends λ. By enumerating all ways of labeling X_u, we get the minimum weight labeling of Y_u. This enumeration is FPT with respect to tw and p as there are at most $tw+1$ nodes in X_u and $p \cdot (tw+1)$ possible labels for each node. By applying this technique on the root bag, we obtain an optimal labeling of G.

However, the labeling of X_u is not sufficient for the recursive algorithm. Let's say we are given a tree decomposition with a bag u_1, the sole child bag u_2 of u_1, with $X_{u_1} = \{v, w\}$, $X_{u_2} = \{w\}$ and $Y_{u_2} = \{w, x, y\}$ (and thus $Y_{u_1} = \{v, w, x, y\}$). Assuming $q = 1, p = 2$ and we want to compute the minimum weight labeling of Y_{u_1} extending $\lambda(v) = \{1, 2\}$ and $\lambda(w) = \{1, 3\}$. A recursive algorithm would compute the minimum weight labelings of Y_{u_2} extending λ' for some labeling λ' of X_{u_2}. However, by losing the memory of $\lambda(v) = \{1, 2\}$, we may accidentally propose to label x or y with $\{1, 2\}$ and this would not result in a feasible labeling of Y_{u_1} as $q = 1$, which implies that $|\lambda^{-1}(1, 2)|$ should be lower than 1. Thus, in addition to λ, we must propagate all the $(q+1)$-tuples t of labels with $\left|\lambda^{-1}(t)\right| = 1$.

Moreover, assuming v and x are given the same label 1 but are not linked, we have to add an edge and increase the weight of the label by one. This means that, in some way, we have to propagate upward the fact that x is labeled with 1.

The next part provides an important result proving that the quantity of information that should be propagated is FPT in tw and p.

4.4 Preliminary Results and Definitions

4.4.1 Renumbering Labelings

We first reduce the number of labels of a set $X \subset V$. Given a labeling λ of V, we can assume, without loss of generality, that the labels in X belong to $[\![1, p|X|]\!]$. Indeed, we can renumber all the labels without changing the corresponding set of added and removed edges. Two labelings λ and λ' of V are *equal* if there exists a permutation σ of $[\![1, p|V|]\!]$ with $\lambda'(v) = \{\sigma(l) | l \in \lambda(v)\}$ for every node $v \in V$.

4.4.2 Continuous Labelings

We prove here that there exists an optimal labeling in which no label skips a bag.

Definition 3. *A labeling λ of V is said* discontinuous *if there exist a bag u and a label l such that $\lambda^{-1}(l) \cap X_u = \emptyset$, $\lambda^{-1}(l) \cap (Y_u \backslash X_u) \neq \emptyset$ and $\lambda^{-1}(l) \cap (Z_u \backslash X_u) \neq \emptyset$. Otherwise, the labeling is said* continuous.

Lemma 2. *Given a discontinuous optimal labeling λ of V, there exists a continuous optimal labeling λ' of V such that* $\max_{l \in [\![1, p|V|]\!]} |\lambda'^{-1}(l)| \leq \max_{l \in [\![1, p|V|]\!]} |\lambda^{-1}(l)|$.

Proof: Let λ be a discontinuous labeling such that there exist k couples of bag and label satisfying the property in Definition 3. Let l and u be such a couple. As at least two nodes are labeled with l, and as there exist $n \cdot p$ labels, there exists an unused label l'. We consider the labeling λ' equal to λ except that the label l of every node in $v \in Y_u$ is replaced with l'. Note that $\max_{l_a \in [\![1,p|V|]\!]} |\lambda'^{-1}(l_a)| \leq \max_{l_a \in [\![1,p|V|]\!]} |\lambda^{-1}(l_a)|$.

For readability, we write $X = X_u, Y = Y_u \backslash X_u$ and $Z = Z_u \backslash X_u$. As no label is changed in any node of X, Z and $Y \backslash \lambda^{-1}(l)$ and as no node of X is labeled with l, the replacement of l to l' only affects the appearance and disappearance of edges between $Y \cap \lambda^{-1}(l)$ and $Z \cap \lambda^{-1}(l)$. Consequently, $\omega(\lambda') - \omega(\lambda) \leq \omega(\lambda', Y \cap \lambda^{-1}(l), Z \cap \lambda^{-1}(l)) - \omega(\lambda, Y \cap \lambda^{-1}(l), Z \cap \lambda^{-1}(l))$.

By definition of the tree decomposition, there is no edge linking Y and Z, otherwise at least one node of those sets would be in X. Thus $\omega(\lambda, Y \cap \lambda^{-1}(l), Z \cap \lambda^{-1}(l)) = |Y \cap \lambda^{-1}(l)| \cdot |Z \cap \lambda^{-1}(l)|$. As $\omega(\lambda', Y \cap \lambda^{-1}(l), Z \cap \lambda^{-1}(l)) \leq |Y \cap \lambda^{-1}(l)| \cdot |Z \cap \lambda^{-1}(l)|$, we have $\omega(\lambda') - \omega(\lambda) \leq 0$. By optimality of λ, then λ' is also optimal.

Note that the labeling λ' contains only $k - 1$ couples of bag and label satisfying the property in Definition 3. We operate again this transformation until this number falls to 0. □

4.4.3 Size of $\lambda^{-1}(l)$

We now consider an optimal labeling and prove that the number of nodes labeled with a same label l depends only on tw and p. To do so, we demonstrate some intermediate lemmas. Lemma 3 identifies for each label l a special node y_l labelled with l that is one of the deepest nodes in the tree decomposition. The labels are renumbered to be ordered from the deepest to the highest. The main argument in this section is that we can derive a recurrence relation between the number of nodes sharing a label with y_l, and the number of nodes sharing a label with $y_{l'}$ with $l' < l$. To do so, we split these sets of nodes into a partition, as done in Definition 4. In Lemmas 4, 5 we prove the recurrence relation and we use it in Lemma 5 to show the upper bound.

Lemma 3. *There exists an optimal labeling λ such that we can partition the labels of G into ρ consecutive intervals $L_1 = [\![1, l_1]\!], L_2 = [\![l_1 + 1, l_2]\!], \ldots, L_\rho = [\![l_{\rho-1} + 1, l_\rho]\!]$, with $\rho \leq n \cdot p$, such that for every $i \in [\![1, \rho]\!]$, there exist a node y_i and a bag $u_i \in \tau$ satisfying the following properties*

1. *y_i is labeled with every label in L_i but no label in L_j for $j > i$,*
2. *for all $l \in L_i$, $(Y_{u_i} \backslash X_{u_i}) \cap \lambda^{-1}(l) = \{y_i\}$,*

Proof: Let λ be an optimal labeling. Let l be a label such that $\lambda^{-1}(l) \neq \emptyset$, we first show that there exist couples (y, u) containing a node y and a bag $u \in \tau$ such that $Y_u \backslash X_u \cap \lambda^{-1}(l) = \{y\}$.

There exists at least one forget bag u that forgets a node y labeled with l. Indeed, the root bag is empty and every node must appear in at least one bag of the decomposition. From all the couples (u, y) satisfying that property, we keep a couple (u, y) maximizing the depth d_l of u in the decomposition. Then no descendant of u is a bag that forgets a node labeled with l. As a consequence, a node in $Y_u \backslash \{y\}$ is either not labeled with l or is in X_u. Then $Y_u \backslash X_u \cap \lambda^{-1}(l) = \{y\}$. Let S be the set of nodes y in the kept couples.

We now build the intervals L_i. We sort S by decreasing value of the depth d_l. We then remove from S every node y such that $\lambda(y)$ is covered by the labels of the preceding nodes in S. We set y_i as the i-th node in S and $L_i = \lambda(y_i) \backslash \bigcup_{j=1}^{i-1} \lambda(y_j)$. Finally, we renumber the labels so that each L_i becomes an interval. □

Definition 4. *Let $i \in [\![1, \rho]\!]$. We write $\bar{Z}_i$ the set of nodes in $Z_{u_i} \setminus X_{u_i}$ labeled with at least one label in L_i, that is: $\bar{Z}_i = Z_{u_i} \setminus X_{u_i} \cap \bigcup_{l \in L_i} \lambda^{-1}(l)$. Let $t \subset \lambda(y_i)$ with $t \cap L_i \neq \emptyset$ and $k \in [\![|t|, p]\!]$. We define with $\bar{Z}_{(i,t,k)}$ the nodes in $\bar{Z}_i$ labeled with every label in t and no label in $\lambda(y_i) \setminus t$, and labeled with exactly k labels lower than* $\max(L_i)$.

Remark 2. *Note that k cannot be less than $|t|$. Indeed, by Lemma 3, y_i is not labeled with any label in L_j for $j > i$. As $L_1, L_2, \ldots, L_\rho$ are consecutive intervals, we have that every label in t is lower than* $\max(L_i)$.

Figure 6 gives an illustration of y_i, $\bar{Z}_i$ and $\bar{Z}_{(i,t,k)}$.

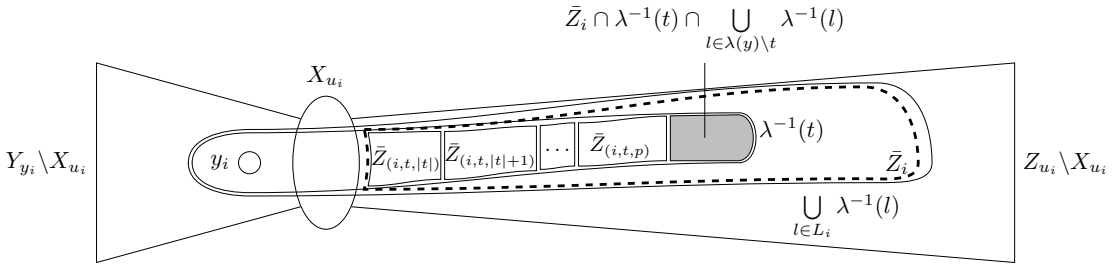


Figure 6: Illustration of the definition of y_i, $\bar{Z}_i$ and $\bar{Z}_{(i,t,k)}$. The node y_i is the sole node in $Y_{u_i} \setminus X_{u_i}$ labeled with any label in L_i. The gray set contains the nodes labeled with t and at least one label of $\lambda(y)$ not in t. By definition, this set and $\bar{Z}_{(i,t,k)}$ are disjoint. It contains instead all the sets $\bar{Z}_{(i,t',k)}$ where $t \subsetneq t'$.

By Lemma 3, any node labeled with $l \in L_i$ is either y_i, in X_{u_i} or in $\bar{Z}_i$. The purpose of the rest of this section is to prove that $|\bar{Z}_i|$ depends only on p and tw. This way, we deduce that the size of $\lambda^{-1}(l)$ depends only on p and tw.

Given a fixed subset t, the sets $\bar{Z}_{(i,t,k)}$ form a partition of $\bar{Z}_i \cap \lambda^{-1}(t) \setminus \bigcup_{l \in \lambda(y) \setminus t} \lambda^{-1}(l)$ that consists of the nodes of $\bar{Z}_i$ labeled with t and no label in $\lambda(y) \setminus t$. Thus, the sets $\bar{Z}_{(i,t,k)}$ for all $t \subset \lambda(y)$, are a partition of $\bar{Z}_i$.

$$\bar{Z}_i = \biguplus_{\substack{t \subset \lambda(y_i) \\ t \cap L_i \neq \emptyset}} \biguplus_{k=|t|}^{p} \bar{Z}_{(i,t,k)}$$

Consequently, we focus on proving that the size of $|\bar{Z}_{(i,t,k)}|$ is bounded by a function depending on p and tw.

Note that, according to the definition of y_i in Lemma 3, t cannot contain a label in L_j for any $j > i$. Thus t is either included in L_i or contains some label in L_j for some $j < i$, in which case $\min(t) \in L_j$ for some $j < i$. This is why we consider these two possible cases with Lemmas 4 and 5.

Lemma 4.

$$\sum_{\substack{t \subset L_i \\ t \neq \emptyset}} \sum_{k=|t|}^{p} |\bar{Z}_{(i,t,k)}| \leq tw + 1$$

Proof: For readability, let $X = X_{u_i}$, $Z = Z_{u_i} \backslash X_{u_i}$ and $Y = Y_{u_i} \backslash (X_{u_i} \cup \{y_i\})$. We build a secondary labeling λ' by changing the labels in L_i of y_i to unused labels.

The weights of the two labelings are

$$\omega(\lambda) = \omega(\lambda, X \cup Y \cup Z) + \omega(\lambda, \{y_i\}, Y) + \omega(\lambda, \{y_i\}, X \cup Z)$$
$$\omega(\lambda') = \omega(\lambda', X \cup Y \cup Z) + \omega(\lambda', \{y_i\}, Y) + \omega(\lambda', \{y_i\}, X \cup Z)$$

As no label in X, Y or Z is changed, $\omega(\lambda, X \cup Y \cup Z) = \omega(\lambda', X \cup Y \cup Z)$. By definition of y_i, no node of Y is labeled with any label in L_i, thus $\omega(\lambda, \{y_i\}, Y) = \omega(\lambda', \{y_i\}, Y)$.

Let X_i and Z_i be respectively the nodes of X and Z labeled (in λ) with some label in L_i but no label in $\lambda(y_i) \backslash L_i$. Note that $|Z_i| = \sum_{\substack{t \subset L_i \\ t \neq \emptyset}} \sum_{k=|t|}^{p} |\bar{Z}_{(i,t,k)}|$.

Let v be a node in $(X \cup Z) \backslash (X_i \cup Z_i)$. Then either v is labeled (in λ) with some label in $\lambda(y_i) \backslash L_i$, or v is not labeled with any label in L_i. Only the labels of y_i in L_i are replaced then, in the two cases, $\lambda(y_i) \cap \lambda(v) \neq \emptyset \iff \lambda'(y_i) \cap \lambda'(v) \neq \emptyset$. Thus, $\omega(\lambda, \{y_i\}, X \cup Z) - \omega(\lambda', \{y_i\}, X \cup Z) = \omega(\lambda, \{y_i\}, X_i \cup Z_i) - \omega(\lambda', \{y_i\}, X_i \cup Z_i)$.

By optimality of λ:

$$\omega(\lambda) \leq \omega(\lambda')$$
$$\omega(\lambda, \{y_i\}, X_i \cup Z_i) \leq \omega(\lambda', \{y_i\}, X_i \cup Z_i)$$

For any node $v \in X_i \cup Z_i$, $\lambda(y) \cap \lambda(v_i) \subset L_i$ and is not empty, thus $\omega(\lambda, \{y_i\}, X_i \cup Z_i) = a(\{y_i\}, X_i \cup Z_i)$. As, in λ', all the labels of y in L_i are replaced, $\lambda'(y) \cap \lambda'(v_i) = \emptyset$. Consequently, $\omega(\lambda', \{y_i\}, X_i \cup Z_i) = r(\{y_i\}, X_i \cup Z_i)$.

$$a(\{y_i\}, X_i \cup Z_i) \leq r(\{y_i\}, X_i \cup Z_i)$$
$$a(\{y_i\}, Z_i) + a(\{y_i\}, X_i) \leq r(\{y_i\}, Z_i) + r(\{y_i\}, X_i)$$

As $a(\{y_i\}, X_i) + r(\{y_i\}, X_i) \leq |X_i|$

$$a(\{y_i\}, Z_i) + 2 \cdot a(\{y_i\}, X_i) \leq r(\{y_i\}, Z_i) + |X_i|$$

As τ is a tree decomposition, as $Z_i \subset Z_{u_i} \backslash X_{u_i}$ and $y_i \in Y_{u_i} \backslash X_{u_i}$, there is no edge between Z_i and y_i

$$|Z_i| + 2 \cdot a(\{y_i\}, X_i) \leq |X_i|$$
$$|Z_i| \leq |X_i|$$

Finally, as $X_i \subset X$ and $|X| \leq tw + 1$, the lemma follows. □

Lemma 5. *Let $t \subset \lambda(y_i)$ with $t \cap L_i \neq \emptyset$ and $t \not\subset L_i$, let $k \in [\![|t|, p]\!]$ and let $j \in [\![1, i-1]\!]$ such that $\min(t) \in L_j$.*

$$|\bar{Z}_{(i,t,k)}| \leq \sum_{t' \subset \lambda(y_j)} \sum_{k'=|t'|}^{k-1} |\bar{Z}_{(j,t',k')}|$$

Proof: Let $v \in \bar{Z}_{(i,t,k)}$. Then v is labeled with some label in L_j. As a consequence, $v \in \bar{Z}_j$.

As $\lambda(v)$ contains k labels lower than $\max(L_i)$ and at least one label in L_i (not in L_j), it contains at most $k-1$ labels lower than $\max(L_j)$. Then, there exists $t' \subset \lambda(y_j)$ and $k' \leq k-1$ such that $v \in \bar{Z}_{(j,t',k')}$. □

With the recurrence relation proved in Lemmas 4 and 5, we prove our upper bound.

Lemma 6. *For every label $l \in L_i$, $\lambda^{-1}(l) \leq (2^p)^p \cdot p! \cdot (tw+1) + tw + 2$.*

Proof: By Lemma 3, there are three sets of nodes labeled with l: the node y_i, the nodes in X_{u_i} and the nodes in $\bar{Z}_i$. There are at most $tw+1$ nodes in X_{u_i}, we need to prove that $|\bar{Z}_i| \leq (2^p)^p \cdot p! \cdot (tw+1)$.

Recall that, as y_i is not labeled with any label in L_j for $j > i$, then we are either in the case of Lemma 4 or 5.

We prove by induction on k that, for every $i \in [\![1, \rho]\!]$ and $t \subset \lambda(y_i)$, $|\bar{Z}_{(i,t,k)}| \leq (2^p)^{k-1} \cdot (k-1)! \cdot (tw+1)$.

If $k = 1$, as $k \geq |t|$ then $|t| = k = 1$. Consequently, by Lemma 4, $|\bar{Z}_{(i,t,k)}| \leq tw + 1$.

We now prove the property for $k+1$ assuming it is true for k. Either $t \subset L_i$ and $|\bar{Z}_{(i,t,k+1)}| \leq tw+1$, by Lemma 4, in which case the property is proven. On the other hand, we can use Lemma 5. Let $j < i$ be such that $\min(t) \in L_j$.

$$|\bar{Z}_{(i,t,k+1)}| \leq \sum_{t' \subset \lambda(y_j)} \sum_{k'=|t'|}^{k} |\bar{Z}_{(j,t',k')}|$$

By induction, for every $t' \subset \lambda(y_j)$ and $k' \leq k$.

$$|\bar{Z}_{(j,t',k')}| \leq (2^p)^{k-1} \cdot (k-1)! \cdot (tw+1)$$

$$|\bar{Z}_{(i,t,k+1)}| \leq \sum_{t' \subset \lambda(y_j)} \sum_{k'=|t'|}^{k} (2^p)^{k-1} \cdot (k-1)! \cdot (tw+1)$$

There are at most 2^p subsets of $\lambda(y_j)$ and k values for k'. Then

$$|\bar{Z}_{(i,t,k+1)}| \leq 2^p \cdot k \cdot (2^p)^{k-1} \cdot (k-1)! \cdot (tw+1)$$
$$|\bar{Z}_{(i,t,k+1)}| \leq (2^p)^k \cdot k! \cdot (tw+1)$$

Thus the property is proven for all value of k. As $k \leq p$, then

$$|\bar{Z}_{(i,t,k)}| \leq (2^p)^{p-1} \cdot (p-1)! \cdot (tw+1)$$

Finally, as $\bar{Z}_i = \biguplus_{\substack{t \subset \lambda(y_i) \\ t \cap L_i \neq \emptyset}} \biguplus_{k=|t|}^{p} \bar{Z}_{(i,t,k)}$

$$|\bar{Z}_i| \leq (2^p)^p \cdot p! \cdot (tw+1)$$

Then, the lemma follows. □

With Lemmas 2 and 6, we are now able to build a dynamic programming algorithm working in time FPT with respect to tw and p.

4.5 Dynamic Programming Algorithm

4.5.1 State of the Dynamic Programming Algorithm

We define the following state containing all the necessary information that should be stored and propagated to find an optimal labeling. For readability, let $\sigma = (2^p)^p \cdot p! \cdot (tw+1) + tw + 2$.

Definition 5 (States of a bag)**.** *Given a bag u, we define with $S(u)$ the set of tuples containing:*

- *a pq-labeling λ of X_u;*
- *a mapping κ from $[\![1, p|X_u|]\!]$ to $[\![0, \sigma]\!]$;*
- *a boolean mapping δ from $[\![1, p|X_u|]\!]^{q+1}$ to $[\![0, 1]\!]$.*

The mapping κ corresponds to the number of nodes in Y_u that should be labeled with the same labels as in X_u. The boolean mapping δ determines whether, for every $(q+1)$-tuple of labels, there is a node in Y_u labeled with this tuple.

Lemma 7. *Let $S = \cup_{Bag\ u} S(u)$, then the number of states in S is FPT with respect to tw and p.*

Proof: There at $O(n)$ bags in the tree decomposition, then $|S| = n \cdot (tw+1)^{p\cdot(tw+1)} \cdot (\sigma+1)^{p\cdot(tw+1)} \cdot 2^{(p\cdot(tw+1))^{(q+1)}}$. □

We now define the weight of a state.

Definition 6. *Let $(\lambda, \kappa, \delta) \in S(u)$, then $L(u, \lambda, \kappa, \delta)$ is the set of labelings Λ of Y_u such that*

1. *Λ is feasible, continuous and for every label l, $|\Lambda^{-1}(l)| \leq \sigma$;*
2. *if $w \in X_u$, $\Lambda(w) = \lambda(w)$;*
3. *for any label l of a node in X_u, $|\Lambda^{-1}(l)| = \kappa(l)$;*
4. *for any $(q+1)$-tuple t of labels of the nodes in X_u, $|\Lambda^{-1}(t)| = \delta(t)$.*

Let $L^*(u, \lambda, \kappa, \delta)$ be the set of labelings $L(u, \lambda, \kappa, \delta)$ with minimum weight $\omega^*(u, \lambda, \kappa, \delta)$ (as stated in Definition 2). We set this value to $+\infty$ if $L(u, \lambda, \kappa, \delta)$ is empty.

First, we prove that λ, κ and δ should be coherent.

Lemma 8. *Let $(\lambda, \kappa, \delta) \in S(u)$. If there exists l such that $\kappa(l) < |\lambda^{-1}(l)|$ or a $(q+1)-$tuple t of labels in X_u such that $\delta(t) < |\lambda^{-1}(t)|$ then $\omega^*(u, \lambda, \kappa, \delta) = +\infty$.*

Proof: Obviously, there is no labeling in $L(u, \lambda, \kappa, \delta)$. □

We consider in the following subsections only states of $S(u)$ such that the assumptions of Lemma 8 are not satisfied.

4.5.2 Root and Leaves Cases

In this part, we write $\lambda_\emptyset$, $\kappa_\emptyset$ and $\delta_\emptyset$ the functions λ, κ and δ when $|X_u| = 0$. We recall that τ is a nice decomposition, and then that the root and the leaves are empty.

Lemma 9. *Let r be the root of τ, the edit distance of G to $\mathcal{C}_{p,q}$ is $\omega^*(r, \lambda_\emptyset, \kappa_\emptyset, \delta_\emptyset)$.*

Proof: As $Y_r = V$ and as $X_r = \emptyset$, any feasible continuous labeling Λ of G satisfying, for every label l, $|\Lambda^{-1}(l)| \leq \sigma$ belongs to $L(r, \lambda_\emptyset, \kappa_\emptyset, \delta_\emptyset)$. By Lemmas 2 and 6, there exists an optimal labeling in $L(r, \lambda_\emptyset, \kappa_\emptyset, \delta_\emptyset)$, consequently, the optimal weight is $\omega^*(r, \lambda_\emptyset, \kappa_\emptyset, \delta_\emptyset)$. □

Lemma 10. *Let u be a leaf of τ, then $\omega^*(u, \lambda_\emptyset, \kappa_\emptyset, \delta_\emptyset) = 0$.*

Proof: X_u and Y_u are empty, thus the empty labeling is in $L(u, \lambda_\emptyset, \kappa_\emptyset, \delta_\emptyset)$ and has a weight 0. □

4.5.3 Forget Bag

Let u be a forget bag, with a child u' in τ and v such that $X_{u'} = X_u \cup \{v\}$. We consider the state $(u, \lambda, \kappa, \delta)$ not satisfying the assumptions of Lemma 8.

Let S' be the set of states $(u', \lambda', \kappa', \delta')$ where λ' coincides with λ on X_u, where κ' coincides with κ on the set of labels $\bigcup_{w \in X_u} \lambda(w)$ and where δ' coincides with δ on the same set of labels.

Lemma 11. $\omega^*(u, \lambda, \kappa, \delta) = \min_{s' \in S'} \omega^*(s')$.

Proof: Let Λ be a labeling of $L^*(u, \lambda, \kappa, \delta)$. We show there exists a state $s' \in S'$ such that $\Lambda \in \Lambda(s')$. Indeed, we consider $s' = (u', \lambda', \kappa', \delta') \in S'$ where $\lambda'(v) = \Lambda(v)$, where, for every label $l \in \Lambda(v) \setminus \bigcup_{w \in X_u} \Lambda(w)$, $\kappa'(l) = |\Lambda^{-1}(l)|$ and where, for every $(q+1)$-tuple t of labels of $X_u \cup \{v\}$ containing at least one label in $\Lambda(v) \setminus \bigcup_{w \in X_u} \Lambda(w)$, then $\delta'(t) = |\Lambda^{-1}(t)|$. By definition of S' and s' and as $\Lambda \in L^*(u, \lambda, \kappa, \delta)$ then Λ satisfies the constraints of Definition 6 for the state s', and then $\Lambda \in \Lambda(s')$. Consequently $\omega^*(u, \lambda, \kappa, \delta) = \omega(\Lambda) \geq \omega^*(s') \geq \min_{s' \in S'} \omega^*(s')$. This shows also that if, for all $s \in S'$, $\omega^*(s') = +\infty$ then $\omega^*(u, \lambda, \kappa, \delta) = +\infty$.

On the other hand, we now consider the state $s' \in S'$ such that $\omega^*(s') = \min_{s' \in S'} \omega^*(s')$. Let Λ' be a labeling of $\Lambda^*(s')$. We show that $\Lambda' \in L(u, \lambda, \kappa, \delta)$. This labeling is feasible and continuous and no label is associated to more than σ nodes. Let $w \in X_u$, then $\Lambda'(w) = \lambda'(w) = \lambda(w)$. Considering $l \in \Lambda'(w)$, then $|\Lambda^{-1}(l)| = \kappa'(l) = \kappa(l)$. Finally, given a $(q+1)$-tuple t of labels in $\bigcup_{w \in X_u} \lambda(w)$, then $|\Lambda^{-1}(t)| = \delta'(t) = \delta(t)$. Then $\Lambda' \in L(u, \lambda, \kappa, \delta)$ and $\omega^*(u, \lambda, \kappa, \delta) \leq \omega(\Lambda') = \omega^*(s') = \min_{s' \in S'} \omega^*(s')$. This also shows that, if $\omega^*(u, \lambda, \kappa, \delta) = +\infty$ then, for all $s \in S'$, $\omega^*(s') = +\infty$. □

4.5.4 Introduce Bag

Let u be an introduce bag with a child u' in τ and v such that $X_u = X_{u'} \cup \{v\}$. We consider the state $(u, \lambda, \kappa, \delta)$ not satisfying the assumptions of Lemma 8.

Lemma 12. *Assuming there exists $l \in \lambda(v)$ such that $\lambda^{-1}(l) = \{v\}$ and $\kappa(l) > 1$ then $\omega^*(u, \lambda, \kappa, \delta) = +\infty$.*

Proof: Let $\Lambda \in L(u, \lambda, \kappa, \delta)$, then $|\Lambda^{-1}(l)| = \kappa(l) > 1$. Thus, there exists a node $w \neq v$ of Y_u labeled with l. This node is not in $X_{u'}$, indeed $\Lambda(w) = \lambda(w)$ and $\lambda^{-1}(l) = \{v\}$. Thus $w \in Y_u \setminus X_u = Y_{u'} \setminus X_{u'}$. Since $v \in Z_{u'} \setminus X_{u'}$, we deduce that Λ is discontinuous and there is a contradiction. As a consequence $L(u, \lambda, \kappa, \delta)$ is empty and $\omega^*(u, \lambda, \kappa, \delta) = +\infty$. □

Lemma 13. *Assuming there exists $l \in \lambda(v)$ with $\kappa(l) = 1$ and a $(q+1)$-tuple t of labels in $\bigcup_{w \in X_u} \lambda(w)$ with $l \in t$, $t \not\subset \lambda(v)$ and $\delta(t) = 1$ then $\omega^*(u, \lambda, \kappa, \delta) = +\infty$.*

Proof: Let $\Lambda \in L(u, \lambda, \kappa, \delta)$. Then $\Lambda^{-1}(l) = \kappa(l) = 1$. As $l \in \lambda(v) = \Lambda(v)$, then v is the sole node labeled with l. In addition $\delta(t) = 1$, thus, there exists a node labeled with t. As $t \not\subset \lambda(v)$, that node is not v. Finally, because $l \in t$, there is a contradiction. As a consequence $L(u, \lambda, \kappa, \delta)$ is empty and $\omega^*(u, \lambda, \kappa, \delta) = +\infty$. □

In this part, we need to manage the labels of v carefully. Contrary to the forget bag case, we cannot simply remove v from the labeling of X_u to get a labeling of $X_{u'}$. We assume none of the hypotheses of Lemma 12 or 13 are satisfied (in addition to Lemma 8). We renumber λ so that, first, for every $w \in X_{u'}$, $l \in \lambda(v)$ and $l' \in \lambda(w)$, then $l \geq l'$ and, secondly, there are no labels l and l' such that $l < l'$, $\lambda^{-1}(l) = \emptyset$ and $\lambda^{-1}(l') \neq \emptyset$. In other words, we use the smallest labels for X_u first and then for v. We now define two transformations between labeling of Y_u and $Y_{u'}$.

Let $(u', \lambda', \kappa', \delta')$ be the following state of $S(u')$:

- for every $w \in X_{u'}$, $\lambda'(w) = \lambda(w)$
- for every $l \in \bigcup_{w \in X_{u'}} \lambda(w)$, $\kappa'(l) = \begin{cases} \kappa(l) - 1 \text{ if } l \in \lambda(v) \\ \kappa(l) \text{ otherwise} \end{cases}$
- for every $(q+1)$-tuple t of labels in $\bigcup_{w \in X_{u'}} \lambda(w)$, $\delta'(t) = \begin{cases} 0 \text{ if } t \subset \lambda(v) \\ \delta(t) \text{ otherwise} \end{cases}$

Definition 7 (From Y_u to $Y_{u'}$)**.** *Given a labeling $\Lambda \in L^*(u, \lambda, \kappa, \delta)$, we build a labeling Λ' of $Y_{u'}$ by removing v from Λ and by renumbering the labels of $Y_{u'}$ so that they vary between 1 and $p \cdot |Y_u|$. We write $f(\Lambda) = \Lambda'$.*

More formally, f is the following procedure

- *If $l \leq p \cdot |Y_{u'}|$ then $\Lambda'^{-1}(l) = \Lambda^{-1}(l) \cap Y_{u'}$.*
- *If $l \in [\![p \cdot |Y_{u'}| + 1, p \cdot |Y_{u'}| + p]\!]$ and $\Lambda^{-1}(l) \cap Y_{u'} \neq \emptyset$, then there exists an unused label $l' \leq p \cdot |Y_{u'}|$ in $Y_{u'}$ (because a node can have at most p labels), then, $\Lambda'^{-1}(l')$ is currently empty. We then set $\Lambda'^{-1}(l') = \Lambda^{-1}(l) \cap Y_{u'}$.*

Definition 8 (From $Y_{u'}$ to Y_u)**.** *Given a labeling $\Lambda' \in L^*(u', \lambda', \kappa', \delta')$, we build a labeling Λ of Y_u by adding v, labeled with $\lambda(v)$ by paying attention to the special case where there exists $l \in \lambda(v)$ such that $\Lambda'^{-1}(l) \neq \emptyset$. We write $g(\Lambda') = \Lambda'$.*

More formally, g is the following procedure

- *For every l such that $l \notin \lambda(v)$ then $\Lambda^{-1}(l) = \Lambda'^{-1}(l)$*
- *For every l such that $l \in \lambda(v)$ and $\kappa(l) \neq 1$ then $\Lambda^{-1}(l) = \Lambda'^{-1}(l) \cup \{v\}$*
- *If $\lambda(v) = (l_1, l_2, \ldots, l_k)$ and $\kappa(l_i) = 1$, then we set $\Lambda^{-1}(p \cdot |Y_{u'}| + i) = \Lambda'^{-1}(l_i)$ and $\Lambda^{-1}(l_i) = \{v\}$*

Example 1. *We consider the example of Figure 7. On that example, with $p = 2$ and $q = 1$, $\lambda = \{v \to \{2, 3\}, v_2 \to \{1, 2\}\}$ and $\kappa = \{1 \to 1, 2 \to 3, 3 \to 1\}$, we have $\Lambda' = f(\Lambda)$ and $\Lambda = g(\Lambda')$.*

Note that, when transforming Λ' back to Λ, we have to label v with 2 and 3 as λ must coincide with Λ on X_u. Then the label 3 on v_3 and v_4 must be renumbered.

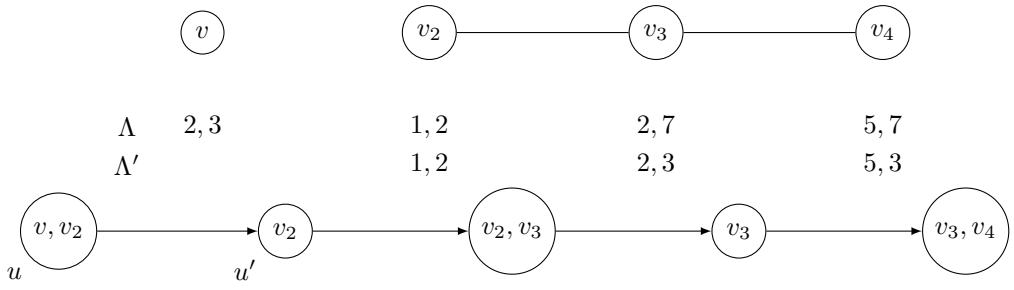


Figure 7: From top to bottom, this figure contains a graph G, a $(2,1)$-labeling Λ of v_1, v_2, v_3 and v_4, a $(2,1)$-labeling Λ' of v_2, v_3 and v_4, and a decomposition τ of G with the introduce bag u on the left, with $v = v_1$ and $X_{u'} = \{v_2\}$.

The six following lemmas from Lemma 14 to 20 are used to prove that $f(\Lambda) \in L(u', \lambda', \kappa', \delta')$ if $\Lambda \in L(u, \lambda, \kappa, \delta)$ and, conversely, $g(\Lambda') \in L(u, \lambda, \kappa, \delta)$ if $\Lambda' \in L(u', \lambda', \kappa', \delta')$.

Lemma 14. *If $\Lambda \in L(u, \lambda, \kappa, \delta)$ and $\Lambda' = f(\Lambda)$ or $\Lambda' \in L(u', \lambda', \kappa', \delta')$ and $\Lambda = g(\Lambda')$ then Λ is feasible and continuous, with $\Lambda^{-1}(l) \leq \sigma$ for every label $l \notin \bigcup_{w \in X_u} \Lambda(w)$ if and only if Λ' is feasible and continuous, with $\Lambda'^{-1}(l) \leq \sigma$ for every label $l \notin \bigcup_{w \in X_{u'}} \Lambda'(w)$.*

Proof: We deduce that result from the fact that, in the two cases, Λ' consists in Λ renumbered after the deletion of v. □

Lemma 15. *If $\Lambda \in L(u, \lambda, \kappa, \delta)$ and $\Lambda' = f(\Lambda)$ or $\Lambda' \in L(u', \lambda', \kappa', \delta')$ and $\Lambda = g(\Lambda')$ then for every $w \in X_{u'}$, $\Lambda(w) = \Lambda'(w) = \lambda(w) = \lambda'(w)$ and $\Lambda(v) = \lambda(v)$.*

Proof: We consider on the first hand that $\Lambda' = f(\Lambda)$. Let $w \in X_{u'}$ and $l \in \Lambda(w)$. Then $\Lambda \in L^*(u, \lambda, \kappa, \delta)$ and $\Lambda(w) = \lambda(w)$. Recall that we renumbered λ so that we use the smallest possible labels in X_u: if $l \in \lambda(w)$, $l \in [\![1, p|X_{u'}|]\!]$. As $|X_{u'}| \leq |Y_{u'}|$. We have $l \leq p|Y_{u'}|$. Due to the first rule of Definition 7, $\Lambda'^{-1}(l) = \Lambda^{-1}(l) \cap Y_{u'}$, then $l \in \Lambda'(w)$. This implies that $\Lambda(w) \subset \Lambda'(w)$. Given now $l' \in \Lambda'(w)$ with $l' \notin \Lambda(w)$, we show there is a contradiction. If $\Lambda'^{-1}(l')$ was defined with the first rule of Definition 7, then $\Lambda'^{-1}(l') = \Lambda^{-1}(l') \cap Y_{u'}$. Then $w \in \Lambda^{-1}(l')$ which is a contradiction. Then it was defined with the second rule of Definition 7. There exists a label l that greater than $p|Y_{u'}|+1$ such that $\Lambda'^{-1}(l') = \Lambda^{-1}(l) \cap Y_{u'}$. Then $w \in \Lambda^{-1}(l)$. As explained previously, if $w \in \Lambda^{-1}(l)$ then $l \leq p|Y_{u'}|$ and there is again a contradiction. Consequently, $\Lambda(w) = \Lambda'(w)$. Finally, as $\Lambda(w) = \lambda(w)$ and $\lambda(w) = \lambda'(w)$, the desired result is proved.

On the second hand, we consider the case where $\Lambda = g(\Lambda')$. Due to Definition 8, we have $\Lambda(v) = \lambda(v)$. Let $w \in X_{u'}$ and $l \in \Lambda'(w)$. If $l \notin \lambda(v)$, or $l \in \lambda(v)$ and $\kappa(l) \neq 1$, then $\Lambda^{-1}(l) \subset \Lambda'^{-1}(l)$ then $l \in \Lambda(w)$. Otherwise $l \in \lambda(v)$ et $\kappa(l) = 1$. As $\Lambda' \in L(u', \lambda', \kappa', \delta')$ then $\Lambda'(w) = \lambda'(w)$, then $l \in \lambda'(w) = \lambda(w)$. However $l \in \lambda(v)$ then $\kappa(l) = 1 < |\lambda^{-1}(l)|$. This is a contradiction as the hypotheses of Lemma 8 are not satisfied. This implies that $\Lambda'(w) \subset \Lambda(w)$. Given now $l \in \Lambda(w)$. Then $l \leq p|Y_{u'}|$. Indeed, on the contrary, $\Lambda^{-1}(l)$ is defined with the third rule of Definition 8. In that case, there exists a label $l' \in \lambda(v)$ with $\kappa(l') = 1$ and $\Lambda^{-1}(l) = \Lambda'^{-1}(l')$, then $l' \in \Lambda'(w) = \lambda'(w)$ which is a contradiction (using the same previous argument). Consequently $l \leq p|Y_{u'}|$. As $w \in \Lambda^{-1}(l)$ then $\Lambda^{-1}(l) \neq \{v\}$ which implies that $\Lambda^{-1}(l)$ is defined with the first or second rule of Definition 8. In the two cases $l \in \Lambda'(w)$. Consequently, $\Lambda(w) = \Lambda'(w)$. Finally, as $\Lambda(w) = \lambda(w)$ and $\lambda(w) = \lambda'(w)$, the desired result is proved. □

Lemma 16. *If $\Lambda \in L(u, \lambda, \kappa, \delta)$ and $\Lambda' = f(\Lambda)$ then, for every label l in $\bigcup_{w \in X_{u'}} \Lambda'(w)$, $\Lambda'^{-1}(l) = \Lambda^{-1}(l) \cap Y_{u'}$.*

Proof: Otherwise, $\Lambda'^{-1}(l)$ is defined with the second rule of Definition 7. In other words, there exists $i \in [\![1, p]\!]$ such that $\Lambda'^{-1}(l) = \Lambda^{-1}(p|Y_{u'}| + i) \cap Y_{u'}$. As $l \in \bigcup_{w \in X_{u'}} \Lambda'(w)$, there exists a node in $w \in X_{u'}$ such that $w \in \Lambda^{-1}(p|Y_{u'}| + i)$. By Lemma 15, $\Lambda(w) = \lambda(w)$, therefore $p|Y_{u'}| + i \in \lambda(w)$. Given the fact that λ uses the smallest possible labels, $p|Y_{u'}| + i \in [\![1, p|X_{u'}|]\!]$. As $|X_{u'}| \leq |Y_{u'}|$, there is a contradiction. □

Lemma 17. *If $\Lambda \in L(u, \lambda, \kappa, \delta)$ and $\Lambda' = f(\Lambda)$ then, for every label l of $\bigcup_{w \in X_{u'}} \Lambda'(w)$, $|\Lambda'^{-1}(l)| = \kappa'(l)$.*

Proof: By Lemma 16, $\Lambda'^{-1}(l) = \Lambda^{-1}(l) \cap Y_{u'}$. Moreover, as $\Lambda \in L(u, \lambda, \kappa, \delta)$, $|\Lambda^{-1}(l)| = \kappa(l)$

- Case 1 : If $l \in \lambda(v)$ then $v \in \Lambda^{-1}(l)$. Because $v \notin Y_{u'}$ then $|\Lambda'^{-1}(l)| = |\Lambda^{-1}(l) \cap Y_{u'}| = |\Lambda^{-1}(l)| - 1 = \kappa(l) - 1 = \kappa'(l)$.
- Case 2 : Otherwise, $|\Lambda'^{-1}(l)| = |\Lambda^{-1}(l) \cap Y_{u'}| = |\Lambda^{-1}(l)| = \kappa(l) = \kappa'(l)$.

□

Lemma 18. *If $\Lambda' \in L(u', \lambda', \kappa', \delta')$ and $\Lambda = g(\Lambda')$ then for every label l of $\bigcup_{w \in X_u} \Lambda(w)$, $|\Lambda^{-1}(l)| = \kappa(l)$.*

Proof: As $\Lambda' \in L(u', \lambda', \kappa', \delta')$, if $l \in \bigcup_{w \in X_u} \Lambda'(w)$, $|\Lambda'^{-1}(l)| = \kappa'(l)$.
We can consider the three following cases

- Case 1 : If $l \notin \lambda(v)$ then, $\kappa'(l) = \kappa(l)$ and, by Definition 8, $|\Lambda^{-1}(l)| = |\Lambda'^{-1}(l)|$. Then $|\Lambda^{-1}(l)| = \kappa(l)$.
- Case 2 : If $l \in \lambda(v)$ and $\kappa(l) = 1$ then, by Definition 8, $\Lambda^{-1}(l) = \{v\}$. Consequently $|\Lambda^{-1}(l)| = \kappa(l)$.
- Case 3 : If $l \in \lambda(v)$ and $\kappa(l) \neq 1$ then $\lambda^{-1}(l) \neq \{v\}$ (as we do not satisfy the hypotheses of Lemma 12). Then $l \in \bigcup_{w \in X_{u'}} \lambda(w) = \bigcup_{w \in X_{u'}} \Lambda'(w)$ by Lemma 15. Consequently $|\Lambda'^{-1}(l)| = \kappa'(l)$. Therefore $|\Lambda'^{-1}(l) \cup \{v\}| = |\Lambda'^{-1}(l)| + 1$. Finally, by Definition 8, $\Lambda^{-1}(l) = \Lambda'^{-1}(l) \cup \{v\}$. Then $|\Lambda^{-1}(l)| = \kappa'(l) + 1 = \kappa(l)$.

□

Lemma 19. *If $\Lambda \in L(u, \lambda, \kappa, \delta)$ and $\Lambda' = f(\Lambda)$ then, for every $(q+1)$-tuple t of labels in $\bigcup_{w \in X_{u'}} \Lambda'(w)$, $|\Lambda'^{-1}(t)| = \delta'(t)$.*

Proof: By Lemma 16, for $l \in t$, $\Lambda'^{-1}(l) = \Lambda^{-1}(l) \cap Y_{u'}$. Then $\Lambda'^{-1}(t) = \Lambda^{-1}(t) \cap Y_{u'}$.
Moreover $\Lambda \in L(u, \lambda, \kappa, \delta)$, then $|\Lambda^{-1}(t)| = \delta(t)$. We can then consider the two following cases

- Case 1 : if $t \not\subset \lambda(v)$ then $\delta'(t) = \delta(t)$. As $\Lambda \in L(u, \lambda, \kappa, \delta)$ then $\Lambda(v) = \lambda(v) \not\supset t$, thus $v \notin \Lambda^{-1}(t)$, which implies that $\Lambda^{-1}(t) \cap Y_{u'} = \Lambda^{-1}(t)$. Consequently $|\Lambda^{-1}(t) \cap Y_{u'}| = |\Lambda^{-1}(t)| = \delta(t) = \delta'(t)$.
- Case 2 : if $t \subset \lambda(v)$, then $\delta'(t) = 0$. In addition, as we do not satisfy the hypotheses of Lemma 8, $\delta(t) = 1$. Then $|\Lambda^{-1}(t)| = 1$ and thus $\Lambda^{-1}(t) = \{v\}$. Consequently $|\Lambda^{-1}(t) \cap Y_{u'}| = 0 = \delta'(t)$.

□

Lemma 20. *If* $\Lambda' \in L(u', \lambda', \kappa', \delta')$ *and* $\Lambda = g(\Lambda')$ *then, for every* $(q+1)$*-tuple* t *of labels in* $\bigcup_{w \in X_u} \Lambda(w)$, $|\Lambda^{-1}(t)| = \delta(t)$.

Proof: Note that $\delta'(t)$ is defined if and only if $l \in \bigcup_{w \in X_{u'}} \lambda(w)$ for all $l \in t$. In that case, as $\Lambda' \in L(u', \lambda', \kappa', \delta')$, $|\Lambda'^{-1}(t)| = \delta'(t)$.

We consider the following cases:

- Case 1 : if $t \subset \lambda(v)$, as we do not satisfy the hypotheses of Lemma 8, $\delta(t) = 1$.
  - Case 1.1 : we first assume that $\lambda^{-1}(l) = \{v\}$ for some $l \in t$. Note that $\delta'(t)$ is not defined. As we do not satisfy the hypotheses of Lemma 12, $\kappa(l) = 1$. By Lemma 18, $|\Lambda^{-1}(l)| = 1$. Consequently $|\Lambda^{-1}(t)| = |\{v\}| = 1 = \delta(t)$.
  - Case 1.2: we now assume that $\lambda^{-1}(l) \neq \{v\}$ for every $l \in t$. Then $t \subset \bigcup_{w \in X_{u'}} \lambda(w)$. Consequently, $\delta'(t)$ is defined and set to 0 and $\Lambda'^{-1}(t) = \emptyset$. In addition, as the hypotheses of Lemma 8 are not satisfied, $\kappa(l) \geq |\lambda^{-1}(l)| > 1$ for every $l \in t$. Thus, $\Lambda^{-1}(l)$ is defined with the second rule of Definition 8, $|\Lambda^{-1}(t)| = |(\Lambda'^{-1}(t) \cup \{v\}| = |\{v\}| = 1$. Thus $|\Lambda^{-1}(t)| = \delta(t)$.
- Case 2 : in this case, we assume that $t \not\subset \lambda(v)$.
  - Case 2.1 : If $l \in \lambda(v)$ and $\lambda^{-1}(l) = \{v\}$ for some $l \in t$ then, as we do not satisfy the hypotheses of Lemma 12, $\kappa(l) = 1$. In addition, the hypotheses of Lemma 13 are also not satisfied, $\delta(t) = 0$. By Lemma 18, $|\Lambda^{-1}(l)| = 1$ then $\Lambda^{-1}(l) = \{v\}$. Due to the fact that $v \notin \Lambda^{-1}(l')$ for some $l' \in t$, $|\Lambda^{-1}(t)| = 0 = \delta(t)$.
  - Case 2.2 : If $l \notin \lambda(v)$, or $l \in \lambda(v)$ and $\lambda^{-1}(l) \neq \{v\}$ for all $l \in t$, we deduce with the same reasoning used in Case 1.2 that $\delta'(t)$ is defined. As $t \not\subset \lambda(v)$ then $\delta(t) = \delta'(t)$. Moreover, by Definition 8, $\Lambda^{-1}(l) = \Lambda'^{-1}(l) \cup \{v\}$ or $\Lambda'^{-1}(l)$ (depending on whether $l \in \lambda(v)$ or not). As $l' \notin \lambda(v)$ for some $l' \in t$ then $|\bigcap_{l \in t} \Lambda^{-1}(l)| = |\Lambda^{-1}(t)| = |\Lambda'^{-1}(t))| = \delta'(t) = \delta(t)$.

□

Lemma 21. $\omega^*(u, \lambda, \kappa, \delta) = \omega^*(u', \lambda', \kappa', \delta') + \omega(\lambda, \{v\}, X_{u'}) + \sum_{l \in \lambda(v)} (\kappa(l) - |(\lambda^{-1}(l)|)$.

Proof:

By Lemmas 14, 15, 17, 18, 19, 20 and 21, if $\Lambda \in L(u, \lambda, \kappa, \delta)$ and $\Lambda' = f(\Lambda)$, then $\Lambda' \in L(u', \lambda', \kappa', \delta')$. On the other hand $\Lambda' \in L(u', \lambda', \kappa', \delta')$ and $\Lambda = g(\Lambda')$ implies that $\Lambda \in L(u, \lambda, \kappa, \delta)$.

In the two cases, the weight of Λ equals the weight of Λ' plus:

- $\omega(\lambda, \{v\}, X_{u'})$
- $\omega(\lambda, \{v\}, Y_u \backslash X_u)$. This weight may be calculated with λ and κ. There are $\kappa(l)$ nodes labeled with l in Λ. We must remove from this set the nodes in X_u labeled with l (including v), in other words $|\lambda^{-1}(l)|$ nodes. Moreover, by definition of a decomposition, G contains no edge from v to a node in $Y_u \backslash X_u$. Thus, if they have a common label, we must add each of those edges. Thus this weight equals $\sum_{l \in \lambda(v)} (\kappa(l) - |\lambda^{-1}(l)|)$.

Consequently

$$\omega(\Lambda) = \omega(\Lambda') + \omega(\lambda, \{v\}, X_{u'}) + \sum_{l \in \lambda(v)} (\kappa(l) - |(\lambda^{-1}(l)|)$$

Assuming $\Lambda' = f(\Lambda)$, for some $\Lambda \in L^*(u, \lambda, \kappa, \delta)$

$$\begin{aligned}\omega^*(u, \lambda, \kappa, \delta) = \omega(\Lambda) &= \omega(\Lambda') + \omega(\lambda, \{v\}, X_{u'}) + \sum_{l \in \lambda(v)} (\kappa(l) - |(\lambda^{-1}(l)|) \\ &\geq \omega^*(u', \lambda', \kappa', \delta') + \omega(\lambda, \{v\}, X_{u'}) + \sum_{l \in \lambda(v)} (\kappa(l) - |(\lambda^{-1}(l)|)\end{aligned}$$

Assuming $\Lambda = g(\Lambda')$, for some, $\Lambda' \in L^*(u', \lambda', \kappa', \delta')$

$$\begin{aligned}\omega^*(u, \lambda, \kappa, \delta) \leq \omega(\Lambda) &= \omega(\Lambda') + \omega(\lambda, \{v\}, X_{u'}) + \sum_{l \in \lambda(v)} (\kappa(l) - |(\lambda^{-1}(l)|) \\ &= \omega^*(u', \lambda', \kappa', \delta') + \omega(\lambda, \{v\}, X_{u'}) + \sum_{l \in \lambda(v)} (\kappa(l) - |(\lambda^{-1}(l)|)\end{aligned}$$

□

4.5.5 Join Bag

In this section, let u be a join bag in τ with children u' and $u"$. We recall that $X_u = X_{u'} = X_{u"}$. We consider the state $(u, \lambda, \kappa, \delta)$ not satisfying the assumptions of Lemma 8.

Let S' be the set of couple of states $(u', \lambda, \kappa', \delta')$ and $(u", \lambda, \kappa", \delta")$ such that:

- given $l \in \bigcup_{w \in X_u} \lambda(w)$, $\kappa'(l) + \kappa"(l) - |\lambda^{-1}(l)| = \kappa(l)$
- given a $(q+1)$-tuple t in $\bigcup_{w \in X_u} \lambda(w)$, $\delta'(t) + \delta"(t) - |\lambda^{-1}(t)| = \delta(t)$

Lemma 22. $\omega^*(u, \lambda, \kappa, \delta) = (\min_{(s', s") \in S'} \omega^*(s') + \omega^*(s")) - \omega(\lambda)$.

Proof: Let Λ be a labeling of $L^*(u, \lambda, \kappa, \delta)$. We consider the states $s' = (u', \lambda, \kappa', \delta')$ and $s" = (u", \lambda, \kappa", \delta")$ such that, for every label $l \in \bigcup_{w \in X_u} \lambda(w)$, $\kappa'(l) = |\Lambda^{-1}(l) \cap Y_{u'}|$ and $\kappa"(l) = |\Lambda^{-1}(l) \cap Y_{u"}|$ and where, for every $(q+1)$-tuple t of labels in $\bigcup_{w \in X_u} \lambda(w)$ then $\delta'(t) = |\Lambda^{-1}(t) \cap Y_{u'}|$ and $\delta"(t) = |\Lambda^{-1}(t) \cap Y_{u"}|$. We easily prove that $(s', s") \in S"$.

We consider $\Lambda' = \Lambda_{|Y_{u'}}$ and $\Lambda" = \Lambda_{|Y_{u"}}$. We renumber the labels of Λ' (respectively $\Lambda"$) in the range from 1 to $2|Y_{u'}|$ (respectively $2|Y_{u"}|$). Note that the labels of X_u are not renumbered as Λ and λ coincide on X_u: they vary from 1 to $2|X_u|$ and $|X_u| \leq |Y_{u'}|$ and $|X_u| \leq |Y_{u"}|$. Due to the definitions of $\kappa', \delta', \kappa"$ and $\delta"$, we check that $\Lambda' \in L(u', \lambda, \kappa', \delta')$ and $\Lambda" \in L(u", \lambda, \kappa", \delta")$.

Note that, due to the properties of the decomposition, $Y_{u'} \cap Y_{u"} = X_u$. Thus weights $\omega(\Lambda)$, $\omega(\Lambda')$ and $\omega(\Lambda")$ equal the following sums

$$\begin{aligned}\omega(\Lambda) &= \omega(\Lambda, X_u) + \omega(\Lambda, X_u, Y_{u'} \setminus X_u) + \omega(\Lambda, Y_{u'} \setminus X_u) + \omega(\Lambda, X_u, Y_{u"} \setminus X_u) + \omega(\Lambda, Y_{u"} \setminus X_u) + \omega(\Lambda, Y_{u'}, Y_{u"}) \\ \omega(\Lambda') &= \omega(\Lambda', X_u) + \omega(\Lambda', X_u, Y_{u'} \setminus X_u) + \omega(\Lambda', Y_{u'} \setminus X_u) \\ \omega(\Lambda") &= \omega(\Lambda", X_u) + \omega(\Lambda", X_u, Y_{u"} \setminus X_u) + \omega(\Lambda", Y_{u'"} \setminus X_u)\end{aligned}$$

First Λ, Λ' and Λ" coincide on X_u with λ, secondly, Λ and Λ' coincide on $Y_{u'}$ and, thirdly, Λ and Λ" coincide on $Y_{u"}$. Finally, $\omega(\Lambda, Y_{u'}, Y_{u"}) \geq 0$, then

$$\begin{aligned}\omega^*(u,\lambda,\kappa,\delta) &= \omega(\Lambda)\\ &\geq \omega(\Lambda') + \omega(\Lambda") - \omega(\lambda)\\ &\geq (\min_{(s',s")\in S'} \omega^*(s') + \omega^*(s")) - \omega(\lambda)\end{aligned}$$

We now consider two states $(s', s") \in S'$ such that $\omega^*(s') + \omega^*(s") = (\min_{(s',s")\in S"} \omega^*(s') + \omega^*(s"))$. Let $\Lambda' \in L^*(u', \lambda, \kappa', \delta')$ and $\Lambda" \in L^*(u", \lambda, \kappa", \delta")$. We build a labeling Λ of Y_u as follows: for every $l \in \bigcup_{w\in Y_{u'}} \Lambda'(w)$, $\Lambda^{-1}(l) = \Lambda'^{-1}(l)$ and for every $l \in \bigcup_{w\in Y_{u"}\setminus X_u} \Lambda"(w) \setminus \bigcup_{w\in X_u} \Lambda"(w)$, $\Lambda^{-1}(l + p|Y_{u'}| - p|X_u|) = \Lambda"^{-1}(l)$. Note that the labels of Λ belong to $[\![1, p|Y_u|]\!]$ and that there is no common label between nodes in $Y_{u'}\setminus X_u$ and $Y_{u"}\setminus X_u$. We show that $\Lambda \in L(u, \lambda, \kappa, \delta)$.

This labeling is feasible and continuous and no label is associated to more than σ nodes. Indeed, if we restrict the labeling on $Y_{u'}$ and $Y_{u"}$, the property is true, and there is no common label in $Y_{u'}\setminus X_u$ and $Y_{u"}\setminus X_u$.

The constraint 2 of Definition 6 is satisfied as Λ' and Λ" coincide with λ and Λ on X_u.

As $\Lambda' \in L^*(u', \lambda, \kappa', \delta')$ and $\Lambda" \in L^*(u", \lambda, \kappa", \delta")$ then, for every $l \in \bigcup_{w\in X_u} \lambda(w)$, the number of nodes in $Y_{u'}$ labeled with l in Λ' (respectively Λ") is $\kappa'(l)$ (respectively $\kappa"(l)$). Thus the number of nodes in Y_u labeled with l in Λ is then $\kappa'(l) + \kappa"(l) - |\Lambda^{-1}(l) \cap X_u| = \kappa'(l) + \kappa"(l) - |\lambda^{-1}(l)| = \kappa(l)$ by definition of S'. Thus Constraint 3 is satisfied. Similarly, the last constraint is satisfied. Thus $\Lambda \in L(u, \lambda, \kappa, \delta)$.

And then $\omega^*(u, \lambda, \kappa, \delta) \leq \omega(\Lambda) = \omega(\Lambda') + \omega(\Lambda") - \omega(\lambda) = (\min_{(s',s")\in S'} \omega^*(s') + \omega^*(s")) - \omega(\lambda)$. □

4.6 Main Result

Theorem 4. *ED-$\mathcal{C}_{p,q}$ is FPT with respect to p and tw.*

Proof: By Lemma 7, the size of S is FPT with respect to p and tw. Assuming we can compute $\omega^*(s)$ for every state $s \in S$, by Lemma 9, we can deduce the edit distance from G to $\mathcal{C}_{p,q}$.

For every state $s \in S$, we can check Lemma 8 in time $O((p \cdot (tw + 1))^{q+1})$.

If u is a leaf and $s \in S(u)$ then we can compute $\omega^*(s)$ in constant time by Lemma 10.

If u is a forget bag and $s \in S(u)$ such that u' is the child of u, and, assuming $\omega^*(s')$ is computed for every $s' \in S(u')$, then, by Lemma 11, we can compute $\omega^*(s)$ in time $O(|S'|)$ where S' is the set defined in that Lemma. We can check if $s \in S'$ in time $O((p \cdot (tw + 1))^{q+1})$. As $S' \subset S$, we can build S' in time $O(|S| \cdot (p \cdot (tw + 1))^{q+1})$.

If u is a join bag and $s \in S(u)$ such that u' and u" are the children of u, and assuming $\omega^*(s')$ and $\omega^*(s")$ are computed for every $s' \in S(u')$ and $S" \in S(u")$, then, by Lemma 22, we can compute $\omega^*(s)$ in time $O(|S'|)$ where S' is the set defined in that Lemma. We can check if $(s', s") \in S'$ in time $O((p \cdot (tw + 1))^{q+1})$. As $S' \subset S^2$, we can build S' in time $O(|S|^2 \cdot (p \cdot (tw + 1))^{q+1})$.

If u is an introduce bag and $s \in S(u)$ such that u' is the child of u. We can check Lemma 12 and 13 respectively in time $O(p)$ and $O((p \cdot (tw + 1))^{q+1})$. If the hypotheses are not satisfied and, assuming $\omega^*(s')$ is computed for every $s' \in S(u')$, we can build the state $(u', \lambda', \kappa', \delta')$ considered in Lemma 21 in time $O((p \cdot (tw + 1))^{q+1})$. We can then compute $\omega^*(s)$ in constant time.

Recall that, we built a decomposition with $O(|V|)$ bags. Thus, the complexity of the calculation of $\omega^*(s)$ for every state s is, on the worst case, $O(|V||S|^2 \cdot (p \cdot (tw + 1))^{q+1})$ which is FPT with respect to tw and p. □

5 Conclusion

We present in this paper an approach to categorize the known results regarding graph membership within a class of linegraphs. This classification allows us to identify the lack of results for specific classes of linegraphs, notably the linegraphs of p-uniform and q-linear hypergraphs with $p \geq 3$ and $q \in [\![1; p]\!]$. We establish that the membership problems are NP-complete for $p \geq 3$ and $q = 2$.

Our classification allows us to address the graph editing distance problem concerning the classes of linegraphs. While some results were known for certain classes, there were gaps in our understanding. Specifically, for class $C_{2,1}$, the linegraphs of simple graphs, the NP-completeness of the editing problem had only been established when either adding or deleting edges was allowed, but not in the general case. We prove that the editing problem for any class $C_{p,q}$ is NP-complete in the general case, for any values of p and q.

Finally, we proposed the first FPT-algorithm with respect to p and treewidth. This algorithm is important because there cannot be an FPT-algorithm depending only on p. However, some questions remain open.

- The complexity of determining whether a graph belongs to class $C_{p,3}$ for $p \geq 3$ and the existence of an FPT-algorithm for the associated editing problems with respect to the number of allowed editions k are still unknown.

- Our algorithm does not address the case where k and tw are fixed but where p and q are not. One can note that the maximum size of a clique after k editions is bounded if the two parameters are. However, our dynamic programming algorithm is not polynomial with respect to p and q, thus, we cannot reused it in this case. Possibly a first question that could be explored is determining whether a graph with fixed treewidth belongs to $\mathcal{C}_{p,q}$ is an NP-Hard problem. In that case, there is no XP algorithm with respect to tw and k. If the recognition problem is polynomial, there is at least an XP algorithm consisting in enumerating the $O(n^{2k})$ possible editions of the graph and applying the recognition algorithm on each such graph. Possibly an FPT algorithm could be derived from an algorithm solving the recognition problem with fixed treewidth.

References

[1] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine learning*, 56(1):89–113, 2004. doi:10.1023/B:MACH.0000033116.57574.95.

[2] L. W. Beineke. Derived graphs and digraphs. *Beiträge zur Graphentheorie*, pages 17–33, 1968.

[3] L. W. Beineke. Characterizations of derived graphs. *Journal of Combinatorial Theory*, 9(2):129–135, 1970. doi:10.1016/S0021-9800(70)80019-9.

[4] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of computational biology*, 6(3-4):281–297, 1999. doi:10.1089/106652799318274.

[5] C. Berge. *Graphes and Hypergraphes.* DUNOD, Paris, 1970.

[6] P. Berman, M. Karpinski, and A. Scott. Approximation hardness of short symmetric instances of max-3sat. *Electronic Colloquium on Computational Complexity (ECCC)*, 01 2003.

[7] J. Bermond, M. Heydemann, and D. Sotteau. Line graphs of hypergraphs i. *Discrete Mathematics*, 18(3):235–241, 1977. doi:10.1016/0012-365X(77)90127-3.

[8] J.-C. Bermond and J.-C. Meyer. Graphe représentaif des arêtes d'un multigraphe. *Journal de Mathématiques Pures et Appliquées*, 52(9):299–308, 1973. URL: https://inria.hal.science/hal-02373326.

[9] J. Chen and J. Meng. A 2k kernel for the cluster editing problem. *Journal of Computer and System Sciences*, 78(1):211–220, 2012. JCSS Knowledge Representation and Reasoning. doi:10.1016/j.jcss.2011.04.001.

[10] D. Cvetkovic, P. Rowlinson, and S. Simic. *Spectral Generalizations of Line Graphs: On Graphs with Least Eigenvalue -2*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2004. doi:10.1017/CBO9780511751752.

[11] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. *Computer Networks*, 31(11):1467–1479, 1999. doi:10.1016/S1389-1286(99)00022-5.

[12] D. G. Degiorgi and K. Simon. A dynamic algorithm for line graph recognition. In M. Nagl, editor, *Graph-Theoretic Concepts in Computer Science*, pages 37–48, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. doi:10.1007/3-540-60618-1_64.

[13] M. Deza and E. Deza. *Encyclopedia of Distances, 4-th ed.* Springer-Verlag, 2016. doi:10.1007/978-3-642-00234-2.

[14] W. Ehounou, D. Barth, A. de Moissac, D. Watel, and M.-A. Weisser. Minimizing the hamming distance between a graph and a line-graph to discover the topology of an electrical network. *Journal of Graph Algorithms and Applications*, 24(3):133–153, Mar. 2020. doi:10.7155/jgaa.00522.

[15] W. J. Ehounou, D. Barth, and A. De Moissac. Discovery of energy network topology from uncertain flow measurements. In J. Rothe, editor, *Algorithmic Decision Theory*, pages 355–360, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-67504-6_27.

[16] M. R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Graph-based data clustering with overlaps. *Discrete Optimization*, 8(1):2–17, 2011. Parameterized Complexity of Discrete Optimization. doi:10.1016/j.disopt.2010.09.006.

[17] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Fixed-parameter algorithms for clique generation. In R. Petreschi, G. Persiano, and R. Silvestri, editors, *Algorithms and Complexity*, pages 108–119, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. doi:10.1007/3-540-44849-7_17.

[18] B. V. Halldórsson, D. Blokh, and R. Sharan. Estimating population size via line graph reconstruction. *Algorithms for molecular biology : AMB*, 8(1):17, jul 2013. doi:10.1186/1748-7188-8-17.

[19] F. Harary. Line Graphs. In *Graph Theory*, chapter 8. Addison Wesley, 1972.

[20] F. Harary and C. Holzmann. Line graphs of bipartite graphs. *Rev. Soc. Mat. Chile*, 1(1):19–22, 1974.

[21] F. Harary and R. Z. Norman. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo*, 9(2):161–168, 1960.

[22] P. Heggernes, D. Lokshtanov, J. Nederlof, C. Paul, and J. A. Telle. Generalized graph clustering: Recognizing (p,q)-cluster graphs. In D. M. Thilikos, editor, *Graph Theoretic Concepts in Computer Science*, pages 171–183, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-16926-7_17.

[23] R. Javadi, Z. Maleki, and B. Omoomi. Local clique covering of graphs, 2012. URL: https://arxiv.org/abs/1210.6965, arXiv:1210.6965.

[24] R. Javadi, Z. Maleki, and B. Omoomi. Local clique covering of claw-free graphs. *Journal of Graph Theory*, 81(1):92–104, 2016. doi:10.1002/jgt.21864.

[25] T. Kloks. *Treewidth: computations and approximations*. Springer Berlin, Heidelberg, 1994. doi:10.1007/BFb0045375.

[26] L. T. Kou, L. J. Stockmeyer, and C. K. Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Commun. ACM*, 21(2):135–139, Feb. 1978. doi:10.1145/359340.359346.

[27] J. Krausz. Démonstration nouvelle d'une théoreme de whitney sur les réseaux. *Mat. Fiz. Lapok*, 50(1):75–85, 1943.

[28] P. G. H. Lehot. An optimal algorithm to detect a line graph and output its root graph. *J. ACM*, 21(4):569–575, Oct. 1974. doi:10.1145/321850.321853.

[29] A. Levin and R. Tyshkevich. Line hypergraphs. *Discrete Mathematics and Applications*, 3(4):407–428, 1993. doi:10.1515/dma-1993-0406.

[30] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. doi:10.1137/0211025.

[31] G.-H. Lin, P. E. Kearney, and T. Jiang. Phylogenetic k-root and steiner k-root. In G. Goos, J. Hartmanis, J. van Leeuwen, D. T. Lee, and S.-H. Teng, editors, *Algorithms and Computation*, pages 539–551, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. doi:10.1007/3-540-40996-3_46.

[32] Y. Metelsky and R. Tyshkevich. On line graphs of linear 3-uniform hypergraphs. *Journal of Graph Theory*, 25(4):243–251, 1997. doi:10.1002/(SICI)1097-0118(199708)25:4<243::AID-JGT1>3.0.CO;2-K.

[33] R. N. Naik, S. Rao, S. Shrikhande, and N. Singhi. Intersection graphs of k-uniform linear hypergraphs. *European Journal of Combinatorics*, 3(2):159–172, 1982. doi:10.1016/S0195-6698(82)80029-2.

[34] J. Naor and M. B. Novick. An efficient reconstruction of a graph from its line graph in parallel. *Journal of Algorithms*, 11(1):132–143, mar 1990. doi:10.1016/0196-6774(90)90034-C.

[35] J. Orlin. Contentment in graph theory: Covering graphs with cliques. volume 80, pages 406–424, 1977. doi:10.1016/1385-7258(77)90055-5.

[36] S. Poljak, V. Rödl, and D. Turzík. Complexity of representation of graphs by set systems. *Discrete Applied Mathematics*, 3(4):301–312, 1981. Special Copy. doi:10.1016/0166-218X(81)90007-X.

[37] F. S. Roberts. Applications of edge coverings by cliques. *Discrete Applied Mathematics*, 10(1):93–109, 1985. doi:10.1016/0166-218X(85)90061-7.

[38] N. Robertson and P. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

[39] N. D. Roussopoulos. A max m,n algorithm for determining the graph h from its line graph g. *Information Processing Letters*, 2(4):108–112, 1973. doi:10.1016/0020-0190(73)90029-X.

[40] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1):173–182, 2004. Discrete Mathematics and Data Mining. doi:10.1016/j.dam.2004.01.007.

[41] V. Tashkinov. A characterization of line graphs of p graphs. In *All Union Conf. on Problems of Theoret. Cybern*, pages 135–137, 1980.

[42] R. Tyshkevich, O. Urbanovich, and I. Zverovich. Matroidal decomposition of a graph. *Banach Center Publications*, 25(1):195–205, 1989.

[43] R. I. Tyshkevich and V. E. Zverovich. Line hypergraphs: A survey. *Acta Applicandae Mathematica*, 52(1):209–222, 1998. doi:10.1023/A:1005963110362.

[44] H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54(1):150–168, 1932. doi:10.2307/2371086.

[45] M. Yannakakis. Node-and edge-deletion np-complete problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, page 253–264, New York, NY, USA, 1978. Association for Computing Machinery. doi:10.1145/800133.804355.

[46] I. Zverovich. An extension of hereditary classes and line graphs. *Izvest. AN Belar. Ser. Fiz. Mat. Nauk*, 3, 1994.