Fast exact algorithms for some connectivity problems parametrized by clique-width

Benjamin Bergougnoux, Mamadou Moustapha Kanté, Mamadou Kanté

FAST EXACT ALGORITHMS FOR SOME CONNECTIVITY PROBLEMS PARAMETERIZED BY CLIQUE-WIDTH

BENJAMIN BERGOUGNOUX AND MAMADOU KANTÉ

Abstract. Given a clique-width k-expression of a graph G, we provide $2^{O(k)} \cdot n$ time algorithms for connectivity constraints on locally checkable properties such as Node-Weighted Steiner Tree, Connected Dominating Set, or Connected Vertex Cover. We also propose a $2^{O(k)} \cdot n$ time algorithm for Feedback Vertex Set. The best running times for all the considered cases were either $2^{O(k \cdot \log(k))} \cdot n^{O(1)}$ or worse.

1. Introduction

Tree-width [20] is probably the most well-studied graph parameter in the graph algorithm community, and particularly by people working in *Fixed Parameter Tractable* (FPT for short) algorithms, due partly to its numerous structural and algorithmic properties [5, 10]. For a while, people used to think that for many connectivity constraints problems, *e.g.*, Hamiltonian Cycle, Steiner Tree, the naive $k^{O(k)} \cdot n^{O(1)}$ time algorithm, k the tree-width of the input graph, cannot be improved. Indeed, it seems necessary to know the connected components of the partial solutions in order to be able to extend them and also certify that the given solution is really connected. But, quite surprisingly, Cygan et al showed in [8] that some of these connectivity constraints problems admit randomized $2^{O(k)} \cdot n^{O(1)}$ time algorithms. The first deterministic $2^{O(k)} \cdot n^{O(1)}$ time algorithms for these problems was due to Bodlaender et al. in [2]. First, let us say that $\mathcal{S}'$ *represents* the set of partial solutions $\mathcal{S}$ if whenever there is $S \in \mathcal{S}$ such that S can be completed into an optimum solution, there is $S' \in \mathcal{S}'$ that can be also completed into an optimum solution. Most of the dynamic programming algorithms are based on this notion of representativity, and proposing a fast algorithm is usually reduced to defining the adequate set $\mathcal{S}'$. For instance, one can define, for each node u of a tree-decomposition, a matrix M_u over the binary field where $M_u[F, F'] = 1$ if and only if the partial solutions F and F' can be "joined" into a valid solution of the instance (F is usually intended to be a subset of a solution, subset included in the already processed part of the graph, and F' the remaining part of the solution, that is not yet known).

The main contribution of [2] was to show that, for some connectivity constraints problems, for each node u of a tree-decomposition of width k, the rank of M_u is at most $2^{O(k)}$; moreover, a maximum weighted basis - such that the solutions indexing its rows represent the solutions indexing the rows of M_u - can be computed in time $2^{O(k)}$.

1991 *Mathematics Subject Classification.* F.2.2, G.2.1, G.2.2.

Key words and phrases. clique-width, module-width, single exponential algorithm, feedback vertex set, connected σ, ρ-domination.

This work is supported by French Agency for Research under the GraphEN project (ANR-15-CE40-0009).

Nevertheless, despite the broad interest on tree-width, only sparse graphs can have bounded tree-width. But, on many dense graph classes, some NP-hard problems admit polynomial time algorithms, and many of these algorithms can be explained by the boundedness of their *clique-width*, a graph parameter introduced by Courcelle and Olariu [7] and that emerges from the theory of graph grammars.

Clique-width is defined in terms of the following graph operations: (1) addition of a single vertex labeled $i \in \mathbb{N}$, (3) renaming label i into j ($ren_{i\to j}$), (3) addition of edges between vertices labeled i and those labeled j ($add_{i,j}$), (4) disjoint union ($\oplus$). The *clique-width* of a graph is the minimum number of labels needed to construct it, and the expression constructing it is called *k-expression*, k the number of used labels. Clique-width generalizes tree-width in the sense that if a graph class has bounded tree-width, then it has bounded clique-width [7], but the converse is false as cliques have clique-width at most 2 and unbounded tree-width. Furthermore, clique-width appears also to be of big importance in FPT algorithms [5]. While it is still open whether there exists an FPT algorithm to compute an optimal k-expression of a given graph, one can ask when clique-width behaves similarly as tree-width. It is known that clique-width is far from behaving similarly as tree-width on some well-studied and well-known difficult problems such as Hamiltonicity [11]. On the other hand, Bui-Xuan et al. [4], and Ganian et al. [13, 14] managed to prove, after more substantial work than for tree-width, that for locally checkable properties and some sparse problems, one can get $2^{k^{O(1)}} \cdot n^{O(1)}$ time algorithms, k the clique-width of the input graph. For some of these algorithms, the exponential in their running times are proved to have a linear dependence on the clique-width, while for others only a polynomial dependence is known.

However, nothing is known on connectivity constraints problems, except some special cases such as Feedback Vertex Set which was proved to admit a $k^{O(k)} \cdot n^{O(1)}$ time algorithm in [3], provided the graph is given with a k-expression.

Our Contributions. We investigate connectivity constraints on locally checkable properties, such as Connected Dominating Set, or Connected Vertex Cover. All these problems are the connected variant of a problem in the family of problems called (σ,ρ)-Dominating Set problems. The family of (σ,ρ)-Dominating Set problems was introduced in [21] and studied in graphs of bounded clique-width in [4, 18]. We recall its definition at the end of Section 2. It is not hard to modify the dynamic programming algorithm from [4] that computes a minimum (σ,ρ)-dominating set in order to compute a minimum connected (σ,ρ)-dominating set in time $k^{O(k)} \cdot n^{O(1)}$, since it suffices to keep track, for each family of partial solutions, the possible partitions of the label classes induced by them. We modify slightly this naive algorithm and prove that one can define representative sets of size $2^{O(k)}$, yielding $2^{O(k)} \cdot n$ time algorithms resolving this family of problems.

We also consider the Feedback Vertex Set problem, which asks to compute a minimum set of vertices to delete so that the resulting graph is acyclic, and propose similarly a $2^{O(k)} \cdot n$ time algorithm. But, the algorithm, even in the same spirit as the one for connected (σ,ρ)-dominating set, is less trivial since one has to check also the acyclicity, a task that is not trivial when dealing with clique-width operations as a bunch of edges can be added at the same time. Indeed, at each step of the dynamic programming algorithm, when dealing with tree-width the number of vertices that have a neighbor in the rest of the graph is bounded, but for clique-width they can be only classified into a bounded number of equivalence classes (with respect to having the same neighborhood in the rest of the graph); these equivalence classes are the *label classes* of the given *k-expression*.

In both cases, we use the same rank-based approach as in [2], but we need to adapt in the Feedback Vertex Set's case the operations on partitions to fit with

clique-width operations. The main difficulty is to deal with the acyclicity, which the authors of [2] also encountered but solved by counting the number of edges. In our case, counting the number of edges would yield an $n^{O(k)}$ time algorithm. Let's explain the ideas of the algorithms with two examples: (1) Connected Dominating Set, asking to compute a minimum connected set D such that each vertex in $V(G) \setminus D$ has a neighbor in D and (2) Maximum Induced Tree, that consists in computing a maximum induced tree.

Let G be a graph, for which a k-expression is given, and let (X, Y) be a bipartition of its vertex set induced by a subexpression of the k-expression. Let us denote the labels of vertices in X by $\{1, 2, \ldots, k\}$. It is worth noticing that at the time we are processing the set X, may be not all the edges between the vertices of X are known (those edges may be added in the forthcoming $add_{i,j}$ operations). To facilitate the steps of the dynamic programming algorithm, we first assume that either all the edges between the vertices of X labeled i and the vertices of X labeled j are already known, or none of them, for all distinct i, j. A k-expressions fulfilling this constraint can be computed from any k-expression in linear time [7].

(1) Let D be a connected dominating set of G and let $D_X := D \cap X$. First, D is not necessarily connected, neither a dominating set of $G[X]$. So, the usual dynamic programming algorithm keeps, for each such D_X, the pair (R, R') of sequences over $\{0, 1\}^k$, where $R_i := 1$ if and only if D_X contains a vertex labeled i, and $R'_i := 1$ if and only if X has a vertex labeled i not dominated by D_X. One first observes that if D_X and D'_X have the same pair of sequences (R, R'), then $D_X \cup D_Y$ is a dominating set of G if and only if $D'_X \cup D_Y$ is a dominating set. Therefore, it is sufficient to keep for each pair (R, R') of sequences in $\{0, 1\}^k$ the possible partitions of $\{1, \ldots, k\}$ corresponding, informally, to the connected components of the graphs induced by the D_X's, and for each possible partition, the maximum weight among all corresponding D_X's. Notice that the graphs induced by the D_X's are not necessarily induced subgraphs of G. One easily checks that these tables can be updated without difficulty following the clique-width operations in time $k^{O(k)} \cdot n^{O(1)}$.

In order to obtain $2^{O(k)} \cdot n$ time algorithms, we modify this algorithm so that the partitions instead of corresponding to the connected components of the graphs induced by the D_X's, do correspond to the connected components of the induced subgraphs $G[D_X]$. For doing so, we do not guess the existence of vertices labeled i that are not dominated, but rather the existence of a vertex that will dominate the vertices labeled i (if not already dominated). With this modification, the steps of our dynamic programming algorithms can be described in terms of the operations on partitions defined in [2]. We can therefore use the same notion of representativity in order to reduce the time complexity.

(2) We consider this example because we reduce the computation of a minimum feedback vertex set to that of a maximum tree. We first observe that we cannot use the same trick as in [2] to ensure the acyclicity, that is counting the number of edges induced by the partial solutions. Indeed, whenever an $add_{i,j}$ operation is used, many edges can be added at the same time. Hence, counting the edges induced by a partial solution would imply to know, for each partial solution, the number of vertices labeled i, for each i. But, this automatically leads to an $n^{O(k)}$ time algorithm. We overcome this difficulty by first defining a binary relation $\mathsf{acyclic}$ on partitions where $\mathsf{acyclic}(p, q)$ holds whenever there are forests E and F, on the same vertex set, such that $E \cup F$ is a forest, and p and q correspond, respectively, to the

connected components of E and F. In a second step, we redefine some of the operations on partitions defined in [2] in order to deal with the acyclicity. These operations are used to describe the steps of the algorithm. They informally help updating the partitions after each clique-width operation by detecting partial solutions that may contain cycles. We also define a new notion of representativity, *ac-representativity*, where $\mathcal{S}'$ ac-represents $\mathcal{S}$ if, whenever there is $S \in \mathcal{S}$ that can be completed into an acyclic connected set, there is $S' \in \mathcal{S}'$ that can be completed into a connected acyclic set. We then prove that one can also compute an ac-representative set of size $2^{O(k)}$, assuming the partitions are on $\{1, \dots, k\}$.

It remains now to describe the steps of the dynamic programming algorithm in terms of the new operations on partitions. First, we are tempted to keep for each forest F of X, the partition induced by the transitive closure of $\sim$ where $i \sim j$ whenever there is a vertex x labeled i, connected in F, to a vertex y labeled j. However, this is not sufficient because we may have in a same connected component two vertices labeled i, and any forthcoming $add_{i,j}$ operation will create a cycle in F if there is a vertex labeled j in F. To overcome this difficulty we index our dynamic programming tables with functions s that inform, for each i, whether there is a vertex labeled i in the partial solutions, and if yes, in exactly 1 vertex, or in at least 2 vertices. Indeed, knowing the existence of at least two vertices is sufficient to detect some cycles when we encounter an $add_{i,j}$ operation. We need also to make a difference when we have all the vertices labeled i in different connected components, or when at least two are in a same connected component. This will allow to detect all the partial solutions F that may contain triangles or cycles on 4 vertices with a forthcoming $add_{i,j}$ operation. Such cycles cannot be detected by the $\mathsf{acyclic}$ binary relation on partitions since this latter does not keep track of the number of vertices in each label. However, the other kinds of cycles are detected through the $\mathsf{acyclic}$ binary relation. We refer the reader to Section 4 for a more detailed description of the algorithm.

One might notice that our algorithms are optimal under the well-know Exponential Time Hypothesis (ETH) [16]. Indeed, from known Karp-reductions, there are no $2^{o(n)}$ time algorithms for the considered problems. Since the clique-width of a graph is always smaller than its number of vertices, it follows that, unless ETH fails, there are no $2^{o(k)} \cdot n^{O(1)}$ time algorithms, for these problems.

The remainder of the paper is organised as follows. The next section is devoted to the main notations, and for the definition of clique-width and of the considered problems. The notion of *ac-representativity* and the modified operations on partitions are given in Section 3. We also propose the algorithm for computing an ac-representative set of size $2^{O(|V|)}$, for sets of weighted partitions on a finite set V. The algorithms for computing a minimum feedback vertex set and connected (σ, ρ)-dominating sets are given in, respectively, Sections 4 and 5.

2. Preliminaries

The size of a set V is denoted by $|V|$ and its power set is denoted by 2^V. We write $A \setminus B$ for the set difference of A from B, and we write $A \uplus B$ for the disjoint union of A and B. We often write x to denote the singleton set $\{x\}$. For a mapping $f : A \to B$, we let $f^{-1}(b) := \{a \in A : b = f(a)\}$ for $b \in B$. We let $\min(\emptyset) := +\infty$ and $\max(\emptyset) := -\infty$. We let $[k] := \{1, \dots, k\}$. We denote by $\mathbb{N}$ the set of non-negative integers and by $\mathbb{F}_2$ the binary field.

Partitions. A partition p of a set V is a collection of non-empty subsets of V that are pairwise non-intersecting and such that $\bigcup_{p_i \in p} p_i = V$; each set in p is called a *block* of p. The set of partitions of a finite set V is denoted by $\Pi(V)$, and $(\Pi(V), \sqsubseteq)$ forms a lattice where $p \sqsubseteq q$ if for each block p_i of p there is a block q_j of q with $p_i \subseteq q_j$. The join operation of this lattice is denoted by $\sqcup$. For example, we have

$$\{\{1,2\},\{3,4\},\{5\}\} \sqcup \{\{1\},\{2,3\},\{4\},\{5\}\} = \{\{1,2,3,4\},\{5\}\}.$$

Let $\sharp\text{block}(p)$ denote the number of blocks of a partition p. Observe that $\emptyset$ is the only partition of the empty set. A *weighted partition* is an element of $\Pi(V) \times \mathbb{N}$ for some finite set V.

For $p \in \Pi(V)$ and $X \subseteq V$, let $p_{\downarrow X} \in \Pi(X)$ be the partition $\{p_i \cap X : p_i \in p\} \setminus \{\emptyset\}$, and for $Y \supseteq V$, let $p_{\uparrow Y} \in \Pi(Y)$ be the partition $p \cup \left(\bigcup_{y \in Y \setminus V} \{\{y\}\}\right)$.

Graphs. Our graph terminology is standard, and we refer to [9]. The vertex set of a graph G is denoted by $V(G)$ and its edge set by $E(G)$. An edge between two vertices x and y is denoted by xy (or yx). The subgraph of G induced by a subset X of its vertex set is denoted by $G[X]$, and we write $G \setminus X$ to denote the induced subgraph $G[V(G) \setminus X]$. The set of vertices that are adjacent to x is denoted by $N_G(x)$, and for $U \subseteq V(G)$, we let $N_G(U) := (\cup_{v \in U} N_G(v)) \setminus U$. For a graph G, we denote by $\mathsf{cc}(G)$ the partition $\{V(C) : C \text{ is a connected component of } G\}$ of $V(G)$.

Clique-Width. A *k-labeled graph* is a pair (G, lab_G) with G a graph and lab_G a function from V_G to $[k]$, called the *labeling function*; each set $lab_G^{-1}(i)$ is called a *label class* and vertices in $lab_G^{-1}(i)$ are called *i-vertices*. The notion of clique-width is defined by Courcelle et al. [6] and is based on the following operations.

(1) Create a graph, denoted by $\mathbf{1}(x)$, with a single vertex x labeled with 1.
(2) For a labeled graph G and distinct labels $i, j \in [k]$, relabel the i-vertices of G with j (denoted by $ren_{i \to j}(G)$). Notice that there are no more i-vertices in $ren_{i \to j}(G)$.
(3) For a labeled graph G and distinct labels $i, j \in [k]$, add all the non-existent edges between the i-vertices and the j-vertices (denoted by $add_{i,j}(G)$).
(4) Take the disjoint union of two labeled graphs G and H, denoted by $G \oplus H$, with $lab_{G \oplus H}(v) := \begin{cases} lab_G(v) & \text{if } x \in V(G), \\ lab_H(v) & \text{otherwise.} \end{cases}$

A *k-expression* is a finite well-formed term built with the four operations above. Each k-expression t evaluates into a k-labeled graph $(val(t), lab(t)$. The *clique-width* of a graph G, denoted by $\text{cwd}(G)$, is the minimum k such that G is isomorphic to $val(t)$ for some k-expression t. We can assume without loss of generality that any k-expression defining a graph G uses $O(n)$ disjoint union operations and $O(nk^2)$ unary operations [7].

It is worth noticing that from the recursive definition of k-expressions, one can compute in time linear in $|t|$ the labeling function $lab(t)$ of $val(t)$, and hence we will always assume that it is given.

To simplify our algorithms, we will use *irredundant k-expressions* (see for instance Section 4).

Definition 2.1 (Irredundant k-expressions [7])**.** A k-expression is *irredundant* if whenever the operation $add_{i,j}$ is applied on a graph G, there is no edge between an i-vertex and a j-vertex in G.

It is proved in [7] that any k-expression can be transformed in linear time into an irredundant k-expression.

Considered Connectivity Problems. For all the problems in this article, we consider the weight function to be on the vertices. Observe that putting weights on the edges would make some of the considered problems such as Steiner Tree NP-hard even on graphs of clique-width two (since cliques have clique-width two).

A subset $X \subseteq V(G)$ of the vertex set of a graph G is a *feedback vertex set* if $G \setminus X$ is a forest. The problem Feedback Vertex Set consists in finding a minimum feedback vertex set. It is not hard to verify that X is a minimum feedback vertex set of G if and only if $G \setminus X$ is a maximum induced forest.

The problem Node-weighted Steiner Tree asks, given a subset of vertices $K \subseteq V(G)$ called *terminals*, a subset T of minimum weight such that $K \subseteq T \subseteq V(G)$ and $G[T]$ is connected.

Let σ and ρ be two (non-empty) finite or co-finite subsets of $\mathbb{N}$. We say that a subset D of $V(G)$ (σ, ρ)-*dominates* a subset $U \subseteq V(G)$ if for every vertex $u \in U$, $|N_G(u) \cap D| \in \sigma$ if $u \in D$, otherwise $|N_G(u) \cap D| \in \rho$. We say that a set X is a (σ, ρ)-dominating set (resp. *co-*(σ, ρ)*-dominating set*) of a graph G, if $V(G)$ is (σ, ρ)-dominated by X (resp. $V(G) \setminus X$). In the connected version, we are moreover asking X to be connected.

Examples of Connected (Co-)(σ, ρ)-Dominating Set problems are shown in Table 1.

σ	ρ	Version	Standard name
$\mathbb{N}$	$\mathbb{N}^+$	*Normal*	Connected Dominating Set
$\mathbb{N}^+$	$\mathbb{N}^+$	*Normal*	Connected Total Dominating Set
$\{d\}$	$\mathbb{N}$	*Normal*	Connected Induced d-Regular Subgraph
$\mathbb{N}$	$\{1\}$	*Normal*	Connected Perfect Dominating Set
$\{0\}$	$\mathbb{N}$	*Co*	Connected Vertex Cover

Figure 1. Examples of Connected (Co-)(σ, ρ)-Dominating Set problems, $\mathbb{N}^+ := \mathbb{N} \setminus \{0\}$.

3. Representing Sets of Acyclic Weighted Partitions by Matrices

We recall that a *weighted partition* is an element of $\Pi(V) \times \mathbb{N}$ for some finite set V. Our algorithms compute a set of weighted partitions $\mathcal{A} \subseteq \Pi(V) \times \mathbb{N}$, for each labeled graph H used in the k-expression of the given graph G and for every subset $V \subseteq lab_H^{-1}(V(H))$. Each weighted partition $(p, w) \in \mathcal{A} \subseteq \Pi(V) \times \mathbb{N}$ is intended to mean the following: there is a solution $S \subseteq V(H)$ of weight w such that p is the transitive closure of the following equivalence relation $\sim$ on V : $i \sim j$ if there exist an i-vertex and a j-vertex in the same component of $H[S]$. Moreover, for every label i in V, there is at least one i-vertex in S. For each label i in V, we expect the i-vertices of S to have an additional neighbor in any extension of S into an optimum solution. This way, for each label $i \in V$, we can consider the i-vertices of S as one vertex in terms of connectivity, since they will have a common neighbor in any extension of S. On the other hand, the labels $j \in [k] \setminus V$ such that S contains at least one j-vertex are expected to have no additional neighbor in any extension of S into an optimum solution. Consequently, the vertices in S with a label in $[k] \setminus V$ do no longer play a role in the connectivity. These expectations allow us to represent the connected components of $H[S]$ by p. Our algorithms will guarantee that the weighted partitions computed from (p, w) are computed accordingly to these expectations.

When considering the Feedback Vertex Set problem, as said in the introduction, the trick used in [2] to deal with acyclicity and that consists in counting the number of edges induced by the partial solutions would yield an $n^{O(k)}$ time algorithm in the case of clique-width. Since the partial solutions for the Feedback Vertex Set problem are represented by weighted partitions, we need to certify that whenever we join two weighted partitions and keep it as a partial solution, it does not correspond to a partial solution with cycles. We introduce in the following a notion of acyclicity between two partitions so that we can identify the joins of partitions which do not produce cycles.

Definition 3.1. Let V be a finite set. We let $\mathsf{acyclic}$ be the relation on $\Pi(V)\times\Pi(V)$ where $\mathsf{acyclic}(p,q)$ holds exactly when $|V|+\sharp\mathrm{block}(p\sqcup q)-(\sharp\mathrm{block}(p)+\sharp\mathrm{block}(q))=0$.

Observe that, if $F_p := (V, E_p)$ and $F_q := (V, E_q)$ are forests with components $p = \mathsf{cc}(F_p)$ and $q = \mathsf{cc}(F_q)$ respectively, then $\mathsf{acyclic}(p,q)$ holds if and only if $E_p \cap E_q = \emptyset$ and $(V, E_p \uplus E_q)$ is a forest. The following is then quite easy to deduce.

Fact 3.2. Let V be a finite set. For all partitions $p, q, r \in \Pi(V)$,

$$\mathsf{acyclic}(p,q) \wedge \mathsf{acyclic}(p\sqcup q, r) \Leftrightarrow \mathsf{acyclic}(q,r) \wedge \mathsf{acyclic}(p, q\sqcup r).$$

Proof. For a partition $p \in \Pi(V)$, let $f(p) := |V| - \sharp\mathrm{block}(p)$. One easily checks that $\mathsf{acyclic}(p,q)$ holds if and only if $f(p\sqcup q)$ equals $f(p)+f(q)$. One can therefore deduce, by an easy calculation from this equivalence, that $\mathsf{acyclic}(p\sqcup q, r) \wedge \mathsf{acyclic}(p,q)$ is equivalent to saying that $f(p\sqcup q\sqcup r)$ equals $f(p)+f(q)+f(r)$. The same statement holds for $\mathsf{acyclic}(q,r) \wedge \mathsf{acyclic}(p, q\sqcup r)$. □

By definition of $\mathsf{acyclic}$ and of $\sqcup$, we can also observe the following.

Fact 3.3. Let V be a finite set. Let $q \in \Pi(V)$ and let $X \subseteq V$ such that no subset of X is a block of q. Then, for each $p \in \Pi(V \setminus X)$, we can observe the following equivalences

$$p_{\uparrow X} \sqcup q = \{V\} \Longleftrightarrow p \sqcup q_{\downarrow(V\setminus X)} = \{V \setminus X\} \qquad \text{and} \tag{1}$$

$$\mathsf{acyclic}(p_{\uparrow X}, q) \Longleftrightarrow \mathsf{acyclic}(p, q_{\downarrow(V\setminus X)}). \tag{2}$$

We modify in this section the operators on weighted partitions defined in [2] in order to express our dynamic programming algorithms in terms of these operators, and also to deal with acyclicity. Let V be a finite set. First, for $\mathcal{A} \subseteq \Pi(V)\times\mathbb{N}$, let

$$\mathsf{rmc}(\mathcal{A}) := \{(p,w) \in \mathcal{A} : \forall (p,w') \in \mathcal{A}, w' \leq w)\}.$$

This operator, defined in [2], is used to remove all the partial solutions whose weights are not maximum *w.r.t.* to their partitions.

Ac-Join. Let V' be a finite set. For $\mathcal{A} \subseteq \Pi(V)\times\mathbb{N}$ and $\mathcal{B} \subseteq \Pi(V')\times\mathbb{N}$, we define $\mathsf{acjoin}(\mathcal{A},\mathcal{B}) \subseteq \Pi(V\cup V')\times\mathbb{N}$ as

$$\mathsf{acjoin}(\mathcal{A},\mathcal{B}) := \{(p_{\uparrow V'} \sqcup q_{\uparrow V}, w_1+w_2) : (p,w_1)\in\mathcal{A}, (q,w_2)\in\mathcal{B} \text{ and } \mathsf{acyclic}(p_{\uparrow V'}, q_{\uparrow V})\}.$$

This operator is more or less the same as the one in [2], except that we incorporate the acyclicity condition. It is used to construct partial solutions while guaranteeing the acyclicity.

Project. For $X \subseteq V$ and $\mathcal{A} \subseteq \Pi(V)\times\mathbb{N}$, let $\mathsf{proj}(\mathcal{A}, X) \subseteq \Pi(V\setminus X)\times\mathbb{N}$ be

$$\mathsf{proj}(\mathcal{A}, X) := \{(p_{\downarrow(V\setminus X)}, w) : (p,w)\in\mathcal{A} \text{ and } \forall p_i \in p, (p_i \setminus X) \neq \emptyset\}.$$

This operator considers all the partitions such that no block is completely contained in X, and then remove X from those partitions. We index our dynamic programming tables with functions that informs on the label classes playing a role in the connectivity of partial solutions, and this operator is used to remove from

the partitions the label classes that are required to no longer play a role in the connectivity of the partial solutions. If a partition has a block fully contained in X, it means that this block will remain disconnected in the future steps of our dynamic programming algorithm, and that is why we remove such partitions (besides those with cycles).

One needs to perform the above operations efficiently, and this is guaranteed by the following, which assumes that $\log(|\mathcal{A}|) \leq |V|^{O(1)}$ for each $\mathcal{A} \subseteq \Pi(V) \times \mathbb{N}$ (this can be established by applying the operator rmc).

Proposition 3.4 (Folklore)**.** *The operator* acjoin *can be performed in time* $|\mathcal{A}| \cdot |\mathcal{B}| \cdot |V \cup V'|^{O(1)}$ *and the size of its output is upper-bounded by* $|\mathcal{A}| \cdot |\mathcal{B}|$*. The operators* rmc *and* proj *can be performed in time* $|\mathcal{A}| \cdot |V|^{O(1)}$*, and the sizes of their outputs are upper-bounded by* $|\mathcal{A}|$*.*

We now define the notion of representative sets of weighted partitions which is the same as the one in [2], except that we need to incorporate the acyclicity condition as for the acjoin operator above.

Definition 3.5. Let V be a finite set and let $\mathcal{A} \subseteq \Pi(V) \times \mathbb{N}$. For $q \in \Pi(V)$, let

$$\textbf{ac-opt}(\mathcal{A}, q) := \max\{w : (p, w) \in \mathcal{A}, p \sqcup q = \{V\} \text{ and } \mathsf{acyclic}(p, q)\}.$$

A set of weighted partitions $\mathcal{A}' \subseteq \Pi(V) \times \mathbb{N}$ *ac-represents* $\mathcal{A}$ if for each $q \in \Pi(V)$, it holds that $\textbf{ac-opt}(\mathcal{A}, q) = \textbf{ac-opt}(\mathcal{A}', q)$.

Let Z and V' be two finite sets. A function $f : 2^{\Pi(V)\times\mathbb{N}} \times Z \to 2^{\Pi(V')\times\mathbb{N}}$ is said to *preserve ac-representation* if for each $\mathcal{A}, \mathcal{A}' \subseteq \Pi(V) \times \mathbb{N}$ and $z \in Z$, it holds that $f(\mathcal{A}', z)$ ac-represents $f(\mathcal{A}, z)$ whenever $\mathcal{A}'$ ac-represents $\mathcal{A}$.

At each step of our algorithm, we will compute a small set $\mathcal{S}'$ that ac-represents the set $\mathcal{S}$ containing all the partial solutions. In order to prove that we compute an ac-representative set of $\mathcal{S}$, we show that $\mathcal{S} = f(\mathcal{R}_1, \ldots, \mathcal{R}_t)$ with f a composition of functions that preserve ac-representation, and $\mathcal{R}_1, \ldots, \mathcal{R}_t$ the sets of partials solutions associated with the previous steps of the algorithm. To compute $\mathcal{S}'$, it is sufficient to compute $f(\mathcal{R}'_1, \ldots, \mathcal{R}'_t)$, where each $\mathcal{R}'_i$ is an ac-representative set of $\mathcal{R}_i$. The following lemma guarantees that the operators we use preserve ac-representation.

Lemma 3.6. *The union of two sets in* $2^{\Pi(V)\times\mathbb{N}}$ *and the operators* rmc*,* proj*, and* acjoin *preserve ac-representation.*

Proof. Let V be a finite set and let $\mathcal{A}$ and $\mathcal{A}'$ be two subsets of $\Pi(V) \times \mathbb{N}$. The proof for the union follows directly from the definition of **ac-opt**.

Rmc. Let $q \in \Pi(V)$. By the definition of rmc, whenever $(p, w) \in \mathcal{A}$ is such that $p \sqcup q = \{V\}$, $\mathsf{acyclic}(p, q)$ and $\textbf{ac-opt}(\mathcal{A}, q) = w$, then $(p, w) \in \mathsf{rmc}(\mathcal{A})$, otherwise there would exist $(p, w') \in \mathcal{A}$ with $w' > w$ which would contradict $w = \textbf{ac-opt}(\mathcal{A}, q)$. Therefore, $\textbf{ac-opt}(\mathsf{rmc}(\mathcal{A}), q) = \textbf{ac-opt}(\mathcal{A}, q)$. We can then conclude that if $\mathcal{A}'$ ac-represents $\mathcal{A}$, it holds that $\mathsf{rmc}(\mathcal{A}')$ ac-represents $\mathsf{rmc}(\mathcal{A})$.

Projections. Because $\mathsf{proj}(\mathcal{A}, X) = \mathsf{proj}(\mathsf{proj}(\mathcal{A}, x), X \setminus \{x\})$ for all $X \subseteq V$ and $x \in X$, we can assume that $X = \{x\}$. Let $q \in \Pi(V \setminus \{x\})$. For every $(p, w) \in \mathcal{A}$, if $\{x\} \in p$, then $p \sqcup q_{\uparrow x} \neq \{V\}$, and $(p_{\downarrow V\setminus x}, w) \notin \mathsf{proj}(\mathcal{A}, \{x\})$. Otherwise, $(p_{\downarrow V\setminus x}, w) \in \mathsf{proj}(\mathcal{A}, \{x\})$, and by Fact 3.3 we have

$$p_{\downarrow V\setminus x} \sqcup q = \{V \setminus \{x\}\} \iff p \sqcup q_{\uparrow x} = \{V\} \qquad \text{and}$$
$$\mathsf{acyclic}(p_{\downarrow V\setminus x}, q) \iff \mathsf{acyclic}(p, q_{\uparrow x}).$$

Therefore, we have $\textbf{ac-opt}(\mathsf{proj}(\mathcal{A}, \{x\}), q) = \textbf{ac-opt}(\mathcal{A}, q_{\uparrow x})$. From this equality, we can conclude that $\mathsf{proj}(\mathcal{A}', \{x\})$ ac-represents $\mathsf{proj}(\mathcal{A}, \{x\})$, for all $\mathcal{A}' \subseteq \mathcal{A}$ such that $\mathcal{A}'$ ac-represents $\mathcal{A}$.

Ac-Join. Let V' be a finite set and let $\mathcal{B} \subseteq \Pi(V') \times \mathbb{N}$. Let $r \in \Pi(V \cup V')$.

Observe that for all $(q, w_2) \in \mathcal{B}$, if a subset of $V' \setminus V$ is a block of $q_{\uparrow V} \sqcup r$, then for all $p \in \Pi(V)$, we have $p_{\uparrow V'} \sqcup q_{\uparrow V} \sqcup r \neq \{V \cup V'\}$. Therefore, $\textbf{ac-opt}(\mathsf{acjoin}(\mathcal{A}, \mathcal{B}), r) = \textbf{ac-opt}(\mathsf{acjoin}(\mathcal{A}, \mathcal{B}'), r)$ where $\mathcal{B}'$ is the set of all $(q, w) \in \mathcal{B}$ such that no subset of $V' \setminus V$ is a block of $q_{\uparrow V} \sqcup r$.

By definition, $\textbf{ac-opt}(\mathsf{acjoin}(\mathcal{A}, \mathcal{B}'), r)$ equals

$$\max\{w_1 + w_2 \,:\, (p, w_1) \in \mathcal{A} \wedge (q, w_2) \in \mathcal{B}' \wedge (p_{\uparrow V'} \sqcup q_{\uparrow V} \sqcup r) = \{V \cup V'\} \\ \wedge \mathsf{acyclic}(p_{\uparrow V'}, q_{\uparrow V}) \wedge \mathsf{acyclic}(p_{\uparrow V'} \sqcup q_{\uparrow V}, r)\}.$$

By Fact 3.2, $\textbf{ac-opt}(\mathsf{acjoin}(\mathcal{A}, \mathcal{B}'), r)$ is then equal to

$$\max\{w_1 + w_2 \,:\, (p, w_1) \in \mathcal{A} \wedge (q, w_2) \in \mathcal{B}' \wedge (p_{\uparrow V'} \sqcup q_{\uparrow V} \sqcup r) = \{V \cup V'\} \\ \wedge \mathsf{acyclic}(q_{\uparrow V}, r) \wedge \mathsf{acyclic}(p_{\uparrow V'}, q_{\uparrow V} \sqcup r)\}.$$

We deduce, by Fact 3.3 and the definition of $\mathcal{B}'$, that $\textbf{ac-opt}(\mathsf{acjoin}(\mathcal{A}, \mathcal{B}'), r)$ equals

$$\max\{w_1 + w_2 \,:\, (p, w_1) \in \mathcal{A} \wedge (q, w_2) \in \mathcal{B}' \wedge (p \sqcup (q_{\uparrow V} \sqcup r)_{\downarrow V}) = \{V\} \\ \wedge \mathsf{acyclic}(q_{\uparrow V}, r) \wedge \mathsf{acyclic}(p, (q_{\uparrow V} \sqcup r)_{\downarrow V})\}.$$

Therefore, we can conclude that $\textbf{ac-opt}(\mathsf{acjoin}(\mathcal{A}, \mathcal{B}), r)$ equals

$$\max\{\textbf{ac-opt}(w_2 + \mathcal{A}, (q_{\uparrow V} \sqcup r)_{\downarrow V}) \,:\, (q, w_2) \in \mathcal{B}' \wedge \mathsf{acyclic}(q_{\uparrow V}, r)\}.$$

Therefore, if $\mathcal{A}'$ ac-represents $\mathcal{A}$, then we can conclude that $\textbf{ac-opt}(\mathsf{acjoin}(\mathcal{A}, \mathcal{B}), r)$ equals $\textbf{ac-opt}(\mathsf{acjoin}(\mathcal{A}', \mathcal{B}), r)$. As this statement is true for all $r \in \Pi(V \cup V')$, we can conclude that $\mathsf{acjoin}(\mathcal{A}', \mathcal{B})$ ac-represents $\mathsf{acjoin}(\mathcal{A}, \mathcal{B})$ whenever $\mathcal{A}'$ ac-represents $\mathcal{A}$. Symmetrically, we deduce that $\mathsf{acjoin}(\mathcal{A}, \mathcal{B}^\star)$ ac-represents $\mathsf{acjoin}(\mathcal{A}, \mathcal{B})$ whenever $\mathcal{B}^\star$ ac-represents $\mathcal{B}$. □

In the remaining, we will prove that, for every set $\mathcal{A} \subseteq \Pi(V) \times \mathbb{N}$, we can find, in time $|\mathcal{A}| \cdot 2^{O(|V|)}$, a subset $\mathcal{A}' \subseteq \mathcal{A}$ of size at most $|V| \cdot 2^{|V|}$ that ac-represents $\mathcal{A}$. As in [2], we will encode the ac-representativity by a matrix over $\mathbb{F}_2$ and show that this one has rank at most the desired bound. Next, we show that an optimum basis of this matrix ac-represents $\mathcal{A}$ and such a basis can be computed using the following lemma from [2]. The constant ω denotes the matrix multiplication exponent.

Lemma 3.7 ([2])**.** *Let M be an $n \times m$-matrix over $\mathbb{F}_2$ with $m \leq n$ and let $w : \{1, \ldots, n\} \to \mathbb{N}$ be a weight function. Then, one can find a basis of maximum weight of the row space of M in time $O(nm^{\omega-1})$.*

Theorem 3.8. *There exists an algorithm* ac-reduce *that, given a set of weighted partitions $\mathcal{A} \subseteq \Pi(V) \times \mathbb{N}$, outputs in time $|\mathcal{A}| \cdot 2^{(\omega-1)\cdot|V|} \cdot |V|^{O(1)}$ a subset $\mathcal{A}'$ of $\mathcal{A}$ that ac-represents $\mathcal{A}$ and such that $|\mathcal{A}'| \leq |V| \cdot 2^{|V|-1}$.*

Proof. If $V = \emptyset$, then it is enough to return $\mathcal{A}' := \{(\emptyset, w)\}$, where $(\emptyset, w) \in \mathcal{A}$ and w is maximum because $\emptyset$ is the only partition of the empty set.

Assume from now that $|V| \neq \emptyset$ (this will ensure that the following definitions exist). Let us first define the matrix that encodes the property that the join of two partitions corresponds to a partition arising from a connected solution.

Let v_0 be a fixed element of V and let $\mathsf{cuts}(V) := \{(V_1, V_2) : V_1 \uplus V_2 = V \text{ and } v_0 \in V_1\}$. Let M and C be, respectively, a $(\Pi(V), \Pi(V))$-matrix and a $(\Pi(V), \mathsf{cuts}(V))$-matrix, both over $\mathbb{F}_2$, such that

$$M[p,q] := \begin{cases} 0 & \text{if } p \sqcup q \neq \{V\}, \\ 1 & \text{otherwise.} \end{cases}$$

$$C[p,(V_1,V_2)] := \begin{cases} 0 & \text{if } p \not\sqsubseteq (V_1,V_2), \\ 1 & \text{otherwise.} \end{cases}$$

As in [2, 8], we fix an element v_0 to ensure that for all $p \in \Pi(V)$, the number of cuts (V_1, V_2) such that $p \sqcup q \sqsubseteq (V_1, V_2)$ is odd if and only if $p \sqcup q = \{V\}$. In fact, this number equals $2^{\sharp\text{block}(p)-1}$. This property is used in [2] to prove that $M = C \cdot C^t$.

Let $\mathcal{A}$ be a set of weighted partitions. In order to compute an ac-representative set of $\mathcal{A}$, we will decompose $\mathcal{A}$ into a small number of sets $\mathcal{A}_i$. Then, for each set $\mathcal{A}_i$, we compute a set $\mathcal{A}'_i \subseteq \mathcal{A}_i$ of $\mathcal{A}_i$ such that the union of the sets $\mathcal{A}'_i$ ac-represents $\mathcal{A}$. To compute $\mathcal{A}'_i$, we use Lemma 3.7 to find a maximum basis of the row space of C restricted to $\mathcal{A}_i$.

For each $0 \leq i \leq |V|-1$, let $\mathcal{A}_i$ be the set $\{p : (p,w) \in \mathcal{A} \text{ and } |V| - \sharp\text{block}(p) = i\}$, and let $C^i_{\mathcal{A}}$ be the restriction of C to rows in $\mathcal{A}_i$. Let $\mathcal{B}_i$ be a basis of the row space of $C^i_{\mathcal{A}}$ of maximum weight, where the weights are the weights[1] of the considered weighted partitions in $\mathcal{A}$. Observe that $|\mathcal{B}_i| \leq 2^{|V|-1}$ because the rank of $C^i_{\mathcal{A}}$ is bounded by $|\mathsf{cuts}(V)| = 2^{|V|-1}$. For $p \in \mathcal{A}_i$, let $\mathcal{B}_i(p)$ be the subset of $\mathcal{B}_i$ such that $C[p,(V_1,V_2)] = \sum_{q \in \mathcal{B}_i(p)} C[q,(V_1,V_2)]$ for all $(V_1,V_2) \in \mathsf{cuts}(V)$. Let $\mathcal{A}'_i$ be the subset of $\mathcal{A}$ corresponding to the rows in $\mathcal{B}_i$, and let $\mathcal{A}' := \mathcal{A}'_0 \uplus \cdots \uplus \mathcal{A}'_{|V|-1}$. Notice that $|\mathcal{A}'| \leq |V| \cdot 2^{|V|-1}$.

Since $C^i_{\mathcal{A}}$ has $|\mathcal{A}_i|$ rows and $2^{|V|-1}$ columns, $\mathcal{A}_i$ is computable in time $|\mathcal{A}_i| \cdot 2^{|V|-1} \cdot |V|^{O(1)}$. By Lemma 3.7, we can compute $\mathcal{B}_i$ in time $|\mathcal{A}_i| \cdot 2^{(\omega-1)\cdot|V|} \cdot |V|^{O(1)}$. Hence, we can compute $\mathcal{A}'$ in time $|\mathcal{A}| \cdot 2^{(\omega-1)\cdot|V|} \cdot |V|^{O(1)}$ because $\{\mathcal{A}_0, \ldots, \mathcal{A}_{|V|-1}\}$ is a partition of $\{p : (p,w) \in \mathcal{A}\}$.

Let us show that for all $p \in \mathcal{A}_i$ and $r \in \Pi(V)$, if $M[p,r] = 1$, then there is $q \in \mathcal{B}_i(p)$ such that $M[q,r] = 1$. Now, from the equality $M = C \cdot C^t$, we have

$$\begin{aligned} M[p,r] &= \sum_{(V_1,V_2) \in \mathsf{cuts}(V)} C[p,(V_1,V_2)] \cdot C^t[(V_1,V_2),r] \\ &= \sum_{(V_1,V_2) \in \mathsf{cuts}(V)} \left(\sum_{q \in \mathcal{B}_i(p)} C[q,(V_1,V_2)] \right) \cdot C^t[(V_1,V_2),r] \\ &= \sum_{q \in \mathcal{B}_i(p)} \left(\sum_{(V_1,V_2) \in \mathsf{cuts}(V)} C[q,(V_1,V_2)] \cdot C^t[(V_1,V_2),r] \right) \\ &= \sum_{q \in \mathcal{B}_i(p)} M[q,r]. \end{aligned}$$

So, $M[p,r] = 1$ if and only if there is an odd number of $q \in \mathcal{B}_i(p)$ such that $M[q,r] = 1$.

It remains now to show that $\mathcal{A}'$ ac-represents $\mathcal{A}$. Since by construction $\mathcal{A}' \subseteq \mathcal{A}$, for each $q \in \Pi(V)$, we have $\textbf{ac-opt}(\mathcal{A},q) \geq \textbf{ac-opt}(\mathcal{A}',q)$. Assume towards a contradiction that $\mathcal{A}'$ does not ac-represent $\mathcal{A}$. Thus, there is $q \in \Pi(V)$ such that $\textbf{ac-opt}(\mathcal{A},q) > \textbf{ac-opt}(\mathcal{A}',q)$, and hence there is $(p,w) \in \mathcal{A} \setminus \mathcal{A}'$ such that $p \sqcup q = \{V\}$, $\mathsf{acyclic}(p,q)$ holds and $w > \textbf{ac-opt}(\mathcal{A}',q)$. Let $i := |V| - \sharp\text{block}(p)$.

[1] We can assume w.l.o.g. that $\mathcal{A} = \mathsf{rmc}(\mathcal{A})$, and thus for each $p \in \mathcal{A}$, there is a unique $w \in \mathbb{N}$ such that $(p,w) \in \mathcal{A}$.

Hence, $p \in \mathcal{A}_i$ and there exists $p' \in \mathcal{B}_i(p)$ such that $M[p', q] = 1$ that is $p' \sqcup q = \{V\}$. Let $(p^\star, w^\star) \in \mathcal{A}'_i$ such that $p^\star \in \mathcal{B}_i(p)$, $p^\star \sqcup q = \{V\}$ and $w^\star$ is maximum.

Since $(p^\star, w^\star) \in \mathcal{A}'_i$, we have $|V| - \sharp\text{block}(p^\star) = |V| - \sharp\text{block}(p) = i$. We can conclude that $\mathsf{acyclic}(p^\star, q)$ holds because $\mathsf{acyclic}(p, q)$ holds. Indeed, by definition, $\mathsf{acyclic}(p, q)$ holds if and only if $|V|+\sharp\text{block}(p\sqcup q)-(\sharp\text{block}(p)+\sharp\text{block}(q)) = 0$. Since $p \sqcup q = \{V\}$, we deduce that $\mathsf{acyclic}(p, q)$ holds if and only if $i = |V| + 1 - \sharp\text{block}(q)$. Because $p^\star \sqcup q = \{V\}$ and $|V| - \sharp\text{block}(p^\star) = i$, we can conclude that $\mathsf{acyclic}(p^\star, q)$ holds.

Hence, we have $\textbf{ac-opt}(\mathcal{A}', q) \geq w^\star$. Since $w > \textbf{ac-opt}(\mathcal{A}', q)$, it must hold that $w > w^\star$. But, $(\mathcal{B}_i \setminus \{p^\star\}) \cup \{p\}$ is also a basis of $C^i_{\mathcal{A}}$ since the set of independent row sets of a matrix forms a matroid. Since $w > w^\star$, the weight of $(\mathcal{B}_i \setminus \{p^\star\}) \cup \{p\}$ is strictly greater than the weight of $\mathcal{B}_i$, yielding a contradiction. □

4. Feedback Vertex Set

We will use the weighted partitions defined in the previous section to represent the partial solutions. At each step of our algorithm, we will ensure that the stored weighted partitions correspond to acyclic partial solutions. However, the framework in the previous section deals only with connected acyclic solutions. So, instead of computing a maximum induced forest (the complementary of a minimum feedback vertex set), we will compute a maximum induced tree. As in [2], we introduce a hypothetical new vertex, denoted by v_0, that is universal and we compute a pair (F, E_0) so that F is a maximum induced forest of G, E_0 is a subset of edges incident to v_0, and $(V(F)\cup\{v_0\}, E(F)\cup E_0)$ is a tree. For a weight function w on the vertices of G and a subset $S \subseteq V(G)$, we denote by $\mathsf{w}(S) := \sum_{v\in S} \mathsf{w}(v)$. In order now to reduce the sizes of the dynamic programming tables, we will express the steps of the algorithm in terms of the operators on weighted partitions defined in the previous section.

Let us explain the idea of the algorithm before defining the dynamic programming tables and the steps. Let H be a k-labeled graph. We are interested in storing *ac-representative sets* of all induced forests of H that may produce a solution. If F is an induced forest of H, we would like to store the partition p corresponding to the quotient set of the transitive closure of the relation $\sim$ on $V(F)$ where $x \sim y$ if x and y have the same label or are in the same connected component. If $J \subseteq [k]$ is such that $\bigcup_{x\in V(F)} lab_H(x) = J$, then this is equivalent to storing the partition p of J where i and j are in the same block if there are, respectively, an i-vertex x and a j-vertex y in the same connected component of F.

Now, if H is used in a k-expression of a k-labeled graph G, then in the clique-width operations defining G we may add edges between the i-vertices and the j-vertices of H, for some $i, j \in J$. Now, this has no effect if there are exactly one i-vertex and one j-vertex at distance one in $H[F]$, otherwise cycles may be created, *e.g.*, whenever an i-vertex and a j-vertex are non-adjacent and belong to the same connected component, or the number of i-vertices and j-vertices are both at least 2. Nevertheless, we are not able to handle all these cases with the operators on weighted partitions. To resolve the situation where an i-vertex and a j-vertex are already adjacent, we consider *irredundant k-expressions*, *i.e.*, whenever an operation $add_{i,j}$ is used there are no edges between i-vertices and j-vertices. For the other cases, we index the dynamic programming tables with functions $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$ that tells, for each $i \in [k]$, if the label class $lab_H^{-1}(i)$ does not intersect F ($s(i) = \gamma_0$), or if it does, in one vertex ($s(i) = \gamma_1$), or in at least two ($s(i) \in \{\gamma_2, \gamma_{-2}\}$) vertices. We have two possible values for label classes intersecting $V(F)$ in at least two vertices because whenever two i-vertices belong to the same connected component of F, F does not produce a valid solution once an operation

$add_{i,\ell}$ is applied to H with $s(\ell) \neq \gamma_0$. So, if a label class $lab_H^{-1}(i)$ intersects $V(F)$ in at least two vertices, since we do not know whether a clique-width operation $add_{i,\ell}$ with $s(\ell) \neq \gamma_0$ will be applied to H, we guess it in the function s, and whenever $s(i)$ equals γ_{-2}, we throw p when we encounter an $add_{i,\ell}$ operation (with $s(\ell) \neq \gamma_0$), and if $s(i) = \gamma_2$, we force F to not having two i-vertices in the same connected component.

Nonetheless, even though we are able to detect the partitions corresponding to induced subgraphs with cycles, taking p as the transitive closure of the relation $\sim$ on $[k]$ describe above may detect false cycles. Indeed, let x_i, x_j and x_ℓ, x'_ℓ be, respectively, an i-vertex, a j-vertex and two ℓ-vertices, such that x_i and x_ℓ belong to the same connected component in F, and similarly, x_j and x'_ℓ in another connected component of F. Now, if we apply an operation $add_{i,j}$ on H, we may detect a cycle with p (through the acjoin operation), which may not exist when for instance there are only one i and one j-vertex in F, both in different connected components. We resolve this case with the functions s indexing the dynamic programming tables by forcing each label i in $s^{-1}(\gamma_2)$ to wait for exactly one clique-width operation $add_{i,t}$ for some $t \in [k]$. We, therefore, translate all the acyclicity tests to the acjoin operation. Indeed, the case explained will be no longer a false cycle as x_i and x_j will be adjacent (with the $add_{i,j}$ operation), and we know that x_ℓ and x'_ℓ will be connected to some other vertex in F, and since x_i is connected to x_ℓ and x_j to x'_ℓ, we have a cycle. The following notion of *certificate graph* formalizes this requirement.

Definition 4.1 (Certificate graph of a solution)**.** Let G be a k-labeled graph, F an induced forest of G, $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$, and E_0 a subset of edges incident to v_0. Let β be a bijection from $s^{-1}(\gamma_2)$ to a set V_s^+ disjoint from $V(G) \cup \{v_0\}$. The *certificate graph of* (F, E_0) *with respect to* s, denoted by $\mathbf{CG}(F, E_0, s)$, is the graph $(V(F) \cup V_s^+ \cup \{v_0\}, E(F) \cup E_0 \cup E_s^+)$ with

$$E_s^+ := \bigcup_{i \in s^{-1}(\gamma_2)} \{\{v, \beta(i)\} : v \in (V(F) \cap lab_G^{-1}(i))\}.$$

In a certificate graph of (F, E_0), the vertices in V_s^+ represent the expected future neighbors of all the vertices in $V(F) \cap lab_G^{-1}(s^{-1}(\gamma_2))$. For convenience, whenever we refer to a vertex v_i^+ of V_s^+, we mean the vertex of V_s^+ adjacent to the i-vertices in $\mathbf{CG}(F, E_0, s)$. See Figure 2 for an example of a certificate graph.

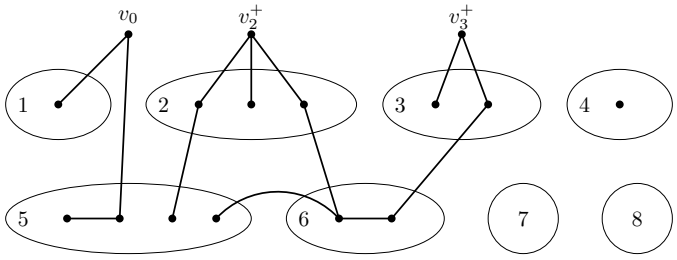

FIGURE 2. Example of a certificate graph; here $p = \{\{v_0, 1\}, \{2, 3\}, \{4\}\}$, $s^{-1}(\gamma_0) = \{7, 8\}$, $s^{-1}(\gamma_1) = \{1, 4\}$, $s^{-1}(\gamma_2) = \{2, 3\}$ and $s^{-1}(\gamma_{-2}) = \{5, 6\}$. The set V_s^+ is $\{v_2^+, v_3^+\}$ with v_2^+ mapped to 2 and v_3^+ mapped to 3.

We are now ready to define the sets of weighted partitions which representatives we manipulate in our dynamic programming tables.

Definition 4.2 (Weighted partitions in $\mathcal{A}_G[s]$)**.** Let G be a k-labeled graph and let $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$ be a total function. The entries of $\mathcal{A}_G[s]$ are all weighted partitions $(p, w) \in \Pi(s^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\}) \times \mathbb{N}$ such that there exist an induced forest F of G and $E_0 \subseteq \{v_0 v : v \in V(F)\}$ so that $\mathsf{w}(V(F)) = w$, and

(1) The sets $s^{-1}(\gamma_0) = \{i \in [k] : |V(F) \cap lab_G^{-1}(i)| = 0\}$ and $s^{-1}(\gamma_1) = \{i \in [k] : |V(F) \cap lab_G^{-1}(i)| = 1\}$.
(2) The certificate graph $\mathbf{CG}(F, E_0, s)$ is a forest.
(3) Each connected component C of $\mathbf{CG}(F, E_0, s)$ has at least one vertex in $lab_G^{-1}(s^{-1}(\{\gamma_1, \gamma_2\})) \cup \{v_0\}$.
(4) The partition p equals $(s^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\})/\sim$ where $i \sim j$ if and only if a vertex in $lab_G^{-1}(i) \cap V(F)$ is connected, in $\mathbf{CG}(F, E_0, s)$, to a vertex in $lab_G^{-1}(j) \cap V(F)$; we consider $lab_G^{-1}(v_0) = \{v_0\}$.

Conditions (1) and (3) are as explained above, and automatically imply that, for each i in $s^{-1}(\{\gamma_2, \gamma_{-2}\})$, we have $|V(F) \cap lab_G^{-1}(i)| \geq 2$. Conditions (2) and (4) guarantee that $(V(F) \cup \{v_0\}, E(F) \cup E_0)$ can be extended into a tree, if any. They also guarantee that cycles detected through the acjoin operation correspond to cycles, and each cycle can be detected with it.

In the sequel, we call any triplet $(F, E_0, (p, \mathsf{w}(F)))$ a *candidate solution* in $\mathcal{A}_G[s]$ if Condition (4) is satisfied, and if in addition Conditions (1)-(3) are satisfied, we call it a *solution* in $\mathcal{A}_G[s]$.

Given a k-labeled graph G, the size of a maximum induced tree of G corresponds to the maximum, over all $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$ with $s^{-1}(\gamma_2) = \emptyset$, of $\max\{w : (\{s^{-1}(\gamma_1) \cup \{v_0\}\}, w) \in \mathcal{A}_G[s]\}$. Indeed, by definition, if $(\{s^{-1}(\gamma_1) \cup \{v_0\}\}, w)$ belongs to $\mathcal{A}_G[s]$, then there exist an induced forest F of G with $\mathsf{w}(F) = w$ and a set E_0 of edges incident to v_0 such that $(V(F) \cup \{v_0\}, E(F) \cup E_0)$ is a tree. This follows from the fact that if $s^{-1}(\gamma_2) = \emptyset$, then we have $\mathbf{CG}(F, E_0, s) = (V(F) \cup \{v_0\}, E(F) \cup E_0)$.

Our algorithm will store, for each k-labeled graph G and each function $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$, an ac-representative set $tab_G[s]$ of $\mathcal{A}_G[s]$. We are now ready to give the different steps of the algorithm, depending on the clique-width operations.

Computing tab_G for $G = \mathbf{1}(x)$. For $s : \{1\} \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$, let

$$tab_G[s] := \begin{cases} \{(\{\{1, v_0\}\}, \mathsf{w}(x)), (\{\{1\}, \{v_0\}\}, \mathsf{w}(x))\} & \text{if } s(1) = \gamma_1, \\ \{(\{\{v_0\}\}, 0)\} & \text{if } s(1) = \gamma_0, \\ \emptyset & \text{if } s(1) \in \{\gamma_2, \gamma_{-2}\}. \end{cases}$$

Since $|V(G)| = 1$, there is no solution intersecting the label class $lab_G^{-1}(1)$ on at least two vertices, and so the set of weighted partitions satisfying Definition 4.2 equals the empty set for $s(1) \in \{\gamma_2, \gamma_{-2}\}$. If $s(1) = \gamma_1$, there are two possibilities, depending on whether $E_0 = \emptyset$ or $E_0 = \{x v_0\}$. We can thus conclude that $tab_G[s] = \mathcal{A}_G[s]$ is correctly computed.

Computing tab_G for $G = add_{i,j}(H)$. We can suppose that H is k-labeled. Let $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$.

(a) If $s(i) = \gamma_0$ or $s(j) = \gamma_0$, then let $tab_G[s] := tab_H[s]$. We just copy all the solutions not intersecting $lab_H^{-1}(i)$ or $lab_H^{-1}(j)$. In the following cases, we assume that $s(i) \neq \gamma_0$ and $s(j) \neq \gamma_0$. In this case, we do not need to use the operators ac-reduce and rmc since we update $tab_G[s]$ with one table from tab_H.
(b) If $s(i) = \gamma_2$ or $s(j) = \gamma_2$, then we let $tab_G[s] = \emptyset$. In this case $\mathcal{A}_G[s] = \emptyset$. Indeed, for every set $X \subseteq V(G)$ respecting Condition (1), the graph $\mathbf{CG}(X, E_0, s)$ contains a cycle, for all subsets $E_0 \subseteq \{x v_0 : x \in X\}$. For example, if $s(i) = \gamma_2$

and $s(j) = \gamma_1$, then the two i-vertices in X are adjacent in $\mathbf{CG}(X, \emptyset, s)$ to v_i^+ and to the j-vertex in X, thus $\mathbf{CG}(X, \emptyset, s)$ contains a cycle of length 4.

(c) If $s(i) = s(j) = \gamma_{-2}$, then we let $tab_G[s] = \emptyset$. Similarly to Case (b), we have $\mathcal{A}_G[s] = \emptyset$ because every vertex set with two i-vertices and two j-vertices induce a cycle of length 4 in G.

(d) Otherwise, we let $tab_G[s] := \mathsf{rmc}(\mathcal{A}))$ with

$$\mathcal{A} := \mathsf{proj}(s^{-1}(\gamma_{-2}) \cap \{i, j\}, \mathsf{acjoin}(tab_H[s_H], \{(\{\{i, j\}\}, 0)\})$$

where $s_H(\ell) := s(\ell)$, for $\ell \in [k] \setminus \{i, j\}$, and

$$(s_H(i), s_H(j)) := \begin{cases} (\gamma_1, \gamma_1) & \text{if } s(i) = s(j) = \gamma_1, \\ (\gamma_2, \gamma_1) & \text{if } (s(i), s(j)) = (\gamma_{-2}, \gamma_1), \\ (\gamma_1, \gamma_2) & \text{if } (s(i), s(j)) = (\gamma_1, \gamma_{-2}). \end{cases}$$

Observe that this case corresponds to $s(i), s(j) \in \{\gamma_1, \gamma_{-2}\}$ with $s(i) = \gamma_1$ or $s(j) = \gamma_1$. Intuitively, we consider the weighted partitions $(p, w) \in tab_H[s_H]$ such that i and j belong to different blocks of p, we merge the blocks containing i and j, remove the elements in $s^{-1}(\gamma_{-2}) \cap \{i, j\}$ from the resulting block, and add the resulting weighted partition to $\mathcal{A}$. Notice that it is not necessarily to call the operator ac-reduce in this case since we update $tab_G[s]$ with only one table from tab_H.

Lemma 4.3. *Let $G = add_{i,j}(H)$ be a k-labeled graph. For each function $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$, $tab_G[s]$ ac-represents $\mathcal{A}_G[s]$ assuming that $tab_H[s']$ ac-represents $\mathcal{A}_H[s']$ for all $s' : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$.*

Proof. We first recall that $lab_G(x) = lab_H(x)$ for all $x \in V(G) = V(H)$. If we are in Cases (b)-(c), then we are done since we clearly have $\mathcal{A}_G[s] = \emptyset$. Since the used operators preserve ac-representation, it is enough to prove that in Case (a), we have $\mathcal{A}_G[s] = \mathcal{A}_H[s]$, and in Case (d), we have $\mathcal{A}_G[s] = \mathcal{A}$ if we let $tab_H[s_H] = \mathcal{A}_H[s_H]$. If we are in Case (a), then we are done because one easily checks that $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_H[s]$ if and only if $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_G[s]$, *i.e.*, we have $\mathcal{A}_H[s] = \mathcal{A}_G[s]$.

Now, we assume that we are in Case (d), that is $s(i), s(j) \in \{\gamma_1, \gamma_{-2}\}$ with $s(i) = \gamma_1$ or $s(j) = \gamma_1$. We can assume w.l.o.g. that $s(j) = \gamma_1$. Let $(F, E_0, (p, w))$ be a solution in $\mathcal{A}_G[s]$. We prove that $(p, w) \in \mathcal{A}$. Let $\{x_j\} := V(F) \cap lab_G^{-1}(j)$ (by assumption $s(j) = \gamma_1$), $X_i := V(F) \cap lab_G^{-1}(i)$, $E_{i,j} := \{vx_j : v \in X_i\}$, and $F_H := (V(F), E(F) \setminus E_{i,j})$. Because we consider only irredundant k-expressions, we know that $E_{i,j} \cap E(H) = \emptyset$, *i.e.*, F_H is an induced forest of H. Let p' be the partition on $s_H^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\}$ such that $(F_H, E_0, (p', w))$ is a candidate solution in $\mathcal{A}_H[s_H]$. We claim that (F_H, E_0, s_H) is a solution in $\mathcal{A}_H[s_H]$. By the definition of s_H, Condition (1) is trivially satisfied. If $|X_i| = 1$, then $\mathbf{CG}(F_H, E_0, s_H)$ is a subgraph of $\mathbf{CG}(F, E_0, s)$ and so Condition (2) is satisfied. And if $|X_i| \geq 2$ and $\mathbf{CG}(F_H, E_0, s_H)$ contains a cycle, then the cycle should contain the vertex $v_i^+ \in V_{s_H}^+$, but this vertex may be replaced by x_j in $\mathbf{CG}(F, E_0, s)$, contradicting the fact this latter is acyclic. Condition (3) is also satisfied because $s_H(i), s_H(j) \in \{\gamma_1, \gamma_2\}$ and for every connected component C of $\mathbf{CG}(F_H, E_0, s_H)$, either C is a connected component of $\mathbf{CG}(F, E_0, s)$ or C contains an ℓ-vertex with $\ell \in \{i, j\}$. Therefore, $(F_H, E_0, (p', w))$ is a solution in $\mathcal{A}_H[s_H]$.

It remains to prove that (p, w) is added in $\mathcal{A}$. We claim that $\mathsf{acyclic}(p', \{\{i, j\}\}_{\uparrow V})$ holds with $V = s^{-1}(\gamma_1, \gamma_2) \cup \{v_0\}$. By definition of acyclic it is equivalent to prove that i and j cannot belong to a same block of p'. Assume towards a contradiction that i and j belong to a same block of p'. Then, there is a path, in $\mathbf{CG}(F_H, E_0, v_0)$, between an i-vertex x_i and the j-vertex x_j of F. Let us choose this path P to be the

smallest one. One first notices that P cannot contain the vertex v_i^+ of $V_{s_H}^+$, if any, because v_i^+ is only adjacent to i-vertices. Because $V(\mathbf{CG}(F_H, E_0, s_H)) \setminus \{v_i^+\} = V(\mathbf{CG}(F, E_0, s))$, we would conclude that $\mathbf{CG}(F, E_0, s)$ contains a cycle as $x_i x_j \in E(F) \setminus E(F_H)$, contradicting that $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_G[s]$. Therefore, $\mathsf{acyclic}(p', \{\{i,j\}\}_{\uparrow V})$ holds. By assumption $s(j) = \gamma_1$, thus $s^{-1}(\gamma_{-2}) \cap \{i,j\} \subseteq \{i\}$. Since i and j are in the same block of the partition $p \sqcup \{\{i,j\}\}_{\uparrow V}$, we conclude that $(p, w) \in \mathsf{proj}(s^{-1}(\gamma_{-2}) \cap \{i,j\}, \mathsf{acjoin}(\{(p', w)\}, \{(\{\{i,j\}\}, 0)\}))$.

It remains to prove that each weighted partition $(p, w) \in \mathcal{A}$ belongs to $\mathcal{A}_G[s]$. Let $(F_H, E_0^H, (p', w))$ be a solution in $\mathcal{A}_H[s_H]$ so that

$$(p, w) \in \mathsf{proj}(s^{-1}(\gamma_{-2}) \cap \{i,j\}, \mathsf{acjoin}(\{(p', w)\}, \{(\{\{i,j\}\}, 0)\})).$$

Let $F = G[V(F_H)]$. By assumption $s(j) = \gamma_1$, and thus $s_H(j) = \gamma_1$. Let $\{x_j\} := V(F_H) \cap lab_H^{-1}(j)$, $X_i := V(F_H) \cap lab_H^{-1}(i)$, and $E_{i,j} := \{x_j v : v \in X_i\}$. Notice that $E(F) \setminus E(F_H) = E_{i,j}$. We claim that $(F, E_0^H, (p, w))$ is a solution in $\mathcal{A}_G[s]$.

- First, Condition (1) is trivially satisfied by the definition of s_H.
- Secondly, $\mathbf{CG}(F, E_0^H, s)$ is a forest. Indeed, $\{i,j\}$ cannot be a block of p', otherwise the acjoin operator would discard p'. If $|X_i| = 1$, then $\mathbf{CG}(F, E_0^H, s) = (V(\mathbf{CG}(F_H, E_0^H, s_H)), E(\mathbf{CG}(F_H, E_0^H, s_H)) \cup E_{i,j})$ and it is clearly a forest. Otherwise, if $|X_i| \geq 2$, then $\mathbf{CG}(F, E_0^H, s)$ can be obtained from $\mathbf{CG}(F_H, E_0^H, s_H)$ by fusing the vertex v_i^+ and the vertex x_j. Clearly, this operation keeps the graph acyclic since x_j and v_i^+ are not connected in $\mathbf{CG}(F_H, E_0^H, s_H)$. Thus $(F, E_0^H, (p, w))$ satisfies Condition (2).
- Each connected component of $\mathbf{CG}(F_H, E_0^H, s_H)$ is contained in a connected component of $\mathbf{CG}(F, E_0^H, s)$, and the i-vertices are in the same connected component, in $\mathbf{CG}(F, E_0^H, s)$, as x_j. Therefore, Condition (3) is satisfied by $(F, E_0^H, (p, w))$ as $s(j) = \gamma_1$ and $s(\ell) = s_H(\ell)$ for all $\ell \in [k] \setminus \{i,j\}$.
- Also, Condition (4) is satisfied as p is then obtained from p' by merging the blocks of p' which contains i and j, and by removing i if $s(i) = \gamma_{-2}$.

We can therefore conclude that $(F, E_0^H, (p, w))$ is a solution in $\mathcal{A}_G[s]$. □

Computing tab_G for $G = ren_{i \to j}(H)$. We can suppose that H is k-labeled. Let $s : [k] \setminus \{i\} \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$.

(a) Let $\mathcal{A}_1 := tab_H[s_1]$ where $s_1(i) = \gamma_0$ and $s_1(\ell) = s(\ell)$ for all $\ell \in [k] \setminus \{i\}$. This set contains all weighted partitions corresponding to solutions not intersecting $lab_H^{-1}(i)$. They are trivially solutions in $\mathcal{A}_G[s]$.

(b) If $s(j) = \gamma_0$, then let $\mathcal{A}_2 = \emptyset$, otherwise let $s_2 : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$ such that $s_2(j) = \gamma_0$, $s_2(i) = s(j)$ and $s_2(\ell) = s(\ell)$ for all $\ell \in [k] \setminus \{i,j\}$ and let

$$\mathcal{A}_2 := \begin{cases} tab_H[s_2] & \text{if } s(j) = \gamma_{-2}, \\ \mathsf{proj}(\{i\}, \mathsf{acjoin}(tab_H[s_2], \{(\{\{i,j\}\}, 0)\})) & \text{otherwise.} \end{cases}$$

This set contains all weighted partitions corresponding to solutions not intersecting $lab_H^{-1}(j)$. They are solutions in $\mathcal{A}_G[s]$ by replacing i by j with the acjoin operator, if necessary, in the corresponding weighted partitions. Notice that if $s(j) = \gamma_0$, then $s_1 = s_2$, this is why we let $\mathcal{A}_2 = \emptyset$ in this case.

(c) If $s(j) \neq \gamma_{-2}$, then let $\mathcal{A}_3 := \emptyset$, otherwise let

$$\mathcal{A}_3 := \bigcup_{s_3 \in \mathcal{S}_3} \mathsf{proj}(\{i,j\}, tab_H[s_3]),$$

where $\mathcal{S}_3$ is the set of functions s_3 with $s_3(i), s_3(j) \in \{\gamma_1, \gamma_{-2}\}$, and $s_3(\ell) = s(\ell)$ for all $\ell \in [k] \setminus \{i,j\}$. Intuitively, $\mathcal{S}_3$ is the set of functions coherent with s if $s(j) = \gamma_{-2}$. The set $\mathcal{A}_3$ corresponds to partial solutions intersecting $lab_H^{-1}(i)$

and $lab_H^{-1}(j)$ when $s(j) = \gamma_{-2}$. In this case, we have to ensure that the partial solutions in $\mathcal{A}_3$ respect Condition (3) of Definition 4.2. We do that by removing all the partitions with a block included in $\{i, j\}$.

(d) We now define the last set considering the other cases. If $s(j) \neq \gamma_2$, then let $\mathcal{A}_4 = \emptyset$, otherwise let

$$\mathcal{A}_4 := \bigcup_{s_4 \in \mathcal{S}_4} \mathsf{proj}(\{i\}, \mathsf{acjoin}(tab_H[s_4], \{(\{\{i, j\}\}, 0)\})),$$

where $\mathcal{S}_4$ is the set of functions s_4 with $s_4(i), s_4(j) \in \{\gamma_1, \gamma_2\}$ and $s_4(\ell) = s(\ell)$ for all $\ell \in [k] \setminus \{i, j\}$. Informally, $\mathcal{S}_4$ is the set of functions compatible with s if $s(j) = \gamma_2$. The set $\mathcal{A}_4$ corresponds to partial solutions intersecting $lab_H^{-1}(i)$ and $lab_H^{-1}(j)$ when $s(j) = \gamma_2$. We have to force that i-vertices and j-vertices belong to different connected components. We check this with the function $\mathsf{acyclic}$ in the operator acjoin.

We let $tab_G[s] := \mathsf{ac\text{-}reduce}(\mathsf{rmc}(\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4))$.

Lemma 4.4. *Let $G = ren_{i \to j}(H)$ with H a k-labeled graph. For each $s : [k] \setminus \{i\} \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$, the table $tab_G[s]$ ac-represents $\mathcal{A}_G[s]$ assuming that $tab_H[s']$ ac-represents $\mathcal{A}_H[s']$ for all $s' : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$.*

Proof. Since the used operators preserve ac-representation, it is enough to prove that $\mathcal{A}_G[s] = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$ if we let $tab_H[s'] = \mathcal{A}_H[s']$ for every $s' : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$.

Let $(F, E_0, (p, w))$ be a solution in $\mathcal{A}_G[s]$. We want to prove that $(p, w) \in \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$. If $V(F)$ does not intersect $lab_H^{-1}(i)$, then $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_H[s_1]$. Assume now that $V(F)$ intersects $lab_H^{-1}(i)$. If $V(F) \cap lab_H^{-1}(j) = \emptyset$ and $s(j) = \gamma_{-2}$, $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_H[s_2]$. If $V(F) \cap lab_H^{-1}(j) = \emptyset$ and $s(j) \in \{\gamma_1, \gamma_2\}$, then it is easy to check that $(F, E_0, (p', w))$ is a solution in $\mathcal{A}_H[s_2]$ where p' is obtained from p by replacing j by i.

We may assume now that $V(F) \cap lab_H^{-1}(i) \neq \emptyset$, $V(F) \cap lab_H^{-1}(j) \neq \emptyset$. Then, we have $s(j) \in \{\gamma_2, \gamma_{-2}\}$ as $|lab_G^{-1}(j) \cap V(F)| \geq 2$. Let $s_\star$ be a function from $[k]$ such that $s_\star(\ell) := s(\ell)$ for all $\ell \in [k] \setminus \{i, j\}$, and for $t \in \{i, j\}$,

$$s_\star(t) := \begin{cases} \gamma_1 & \text{if } |V(F) \cap lab_H^{-1}(t)| = 1, \\ s(j) & \text{if } |V(F) \cap lab_H^{-1}(t)| \geq 2. \end{cases}$$

By definition, if $s(j) = \gamma_{-2}$, then $s_\star$ belongs to $\mathcal{S}_3$ and if $s(j) = \gamma_2$, then $s_\star$ belongs to $\mathcal{S}_4$. Let p' be the partition on $s_\star^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\}$ such that $(F, E_0, (p', w))$ is a candidate solution in $\mathcal{A}_H[s_\star]$. We claim that $(F, E_0, (p', w))$ is a solution in $\mathcal{A}_H[s_\star]$. By Definition of $s_\star$ and of (F, E_0), Condition (1) is satisfied by $(F, E_0, (p', w))$.

Suppose first that $s(j) = \gamma_{-2}$. Observe that Condition (2) is satisfied because the certificate graphs of (F, E_0) with respect to s and $s_\star$ are the same. Condition (3) is also satisfied by definition of $s_\star$ and because $\mathbf{CG}(F, E_0, s) = \mathbf{CG}(F, E_0, s_\star)$. So, $(F, E_0, (p', w))$ is a solution in $\mathcal{A}_H[s_\star]$.

Assume now that $s(j) = \gamma_2$. Condition (2) is satisfied. Indeed, if $s_\star(i) = \gamma_1$, then $\mathbf{CG}(F, E_0, s)$ is a subgraph of $\mathbf{CG}(F, E_0, s_\star)$. Otherwise, if $s_\star(i) = \gamma_2$, then $\mathbf{CG}(F, E_0, s)$ can be obtained from $\mathbf{CG}(F, E_0, s_\star)$ by fusing v_i^+ with v_j^+. In both cases, it is easy to see that $\mathbf{CG}(F, E_0, s_\star)$ is acyclic as $\mathbf{CG}(F, E_0, s)$ is acyclic. Condition (3) is satisfied because each connected component C of $\mathbf{CG}(F, E_0, s)$ contains at least a vertex in $lab_G^{-1}(s^{-1}(\{\gamma_1, \gamma_2\})) \cup \{v_0\}$, and we have from the definition of $s_\star$

$$lab_H^{-1}(s_\star^{-1}(\{\gamma_1, \gamma_2\})) = lab_G^{-1}(s^{-1}(\{\gamma_1, \gamma_2\})).$$

In both cases, $(F, E_0, (p', w))$ is a solution in $\mathcal{A}_H[s_\star]$, and depending on $s(j)$, we can clearly conclude that (p, w) is obtained from (p', w).

Let us now prove that for any weighted partition $(p, w) \in \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$, there is a pair (F, E_0) such that $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_G[s]$. This is clear if $(p, w) \in \mathcal{A}_1 \cup \mathcal{A}_2$.

Assume that $s(j) = \gamma_{-2}$ and let $(p, w) \in \mathcal{A}_3$. Let $s_3 \in \mathcal{S}_3$ and (p', w) be the weighted partition from $tab_H[s_3]$ from which (p, w) is obtained. Let $(F, E_0, (p', w))$ be a solution in $\mathcal{A}_H[s_3]$. We claim that $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_G[s]$. By definition of $\mathcal{S}_3$, we clearly have $|V(F) \cap lab_H^{-1}(\{i, j\})| = |V(F) \cap lab_G^{-1}(j)| \geq 2$. We deduce that Condition (1) is satisfied. Condition (2) is also satisfied because $\mathbf{CG}(F, E_0, s_3)$ is the same as $\mathbf{CG}(F, E_0, s)$. We claim that Condition (3) is satisfied. Notice that $s^{-1}(\{\gamma_1, \gamma_2\}) = s_3^{-1}(\{\gamma_1, \gamma_2\}) \setminus \{i, j\}$. Moreover, if $i \in s_3^{-1}(\{\gamma_1, \gamma_2\})$, by definition of s_3, we have $s_3(i) = \gamma_1$, that is F has exactly one i-vertex (the same statement is true for j). Since we use the operator proj with $\{i, j\}$, there is no block of p' included in $\{i, j\}$. Hence, if F contains one i-vertex (one j-vertex), then, by Condition (4), this vertex is connected in $\mathbf{CG}(F, E_0, s_3)$ to either v_0 or an ℓ-vertex with $\ell \in s^{-1}(\{\gamma_1, \gamma_2\})$. We can conclude that each connected component of F must contain a vertex in $lab_G^{-1}(s^{-1}(\{\gamma_1, \gamma_2\})) \cup \{v_0\}$, *i.e.* Condition (3) is satisfied. Condition (4) is satisfied owing to the fact that p' is obtained from p by doing a projection on $\{i, j\}$.

Assume now that $s(j) = \gamma_2$ and let $(p, w) \in \mathcal{A}_4$. Let $s_4 \in \mathcal{S}_4$ and (p', w) be the weighted partition from $tab_H[s_4]$ from which (p, w) is obtained, and let $(F, E_0, (p', w))$ be a solution in $\mathcal{A}_H[s_4]$. By definition of $\mathcal{S}_4$, we deduce that Condition (1) is satisfied (see the case $s(j) = \gamma_{-2}$). If there is a cycle in $\mathbf{CG}(F, E_0, s)$, then it must be between an i-vertex and a j-vertex of H, but then we must have a path between them in $\mathbf{CG}(F, E_0, s_4)$, *i.e.*, i and j belong to a same block of p', contradicting that (p, w) is produced from (p', w) (because the acjoin operator would detect that $\mathsf{acyclic}(p, \{\{i, j\}\}_{\uparrow[k]})$ does not hold). So, Condition (2) is satisfied. We deduce that Condition (3) is satisfied from the fact that by definition of s_4, we have $lab_G^{-1}(s^{-1}(\{\gamma_1, \gamma_2\})) = lab_H^{-1}(s_4^{-1}(\{\gamma_1, \gamma_2\}))$. Also, as p is obtained from p' by merging the blocks containing i and j, and by removing i, we deduce that Condition (4) is satisfied.

In both cases, we can conclude that $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_G[s]$. □

Computing tab_G for $G = G_a \oplus G_b$. We can suppose w.l.o.g. that G_a and G_b are both k-labeled[2]. Let $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$.

We say that $s_a : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$ and $s_b : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$ *u-agree on* s if,

(u1) for each $i \in s_a^{-1}(\gamma_0)$, $s(i) = s_b(i)$. Similarly, for each $i \in s_b^{-1}(\gamma_0)$, $s(i) = s_a(i)$,
(u2) for each $i \in s^{-1}(\gamma_1)$, either $s_a(i) = \gamma_0$ or $s_b(i) = \gamma_0$,
(u3) for each $i \in [k] \setminus (s_a^{-1}(\gamma_0) \cup s_b^{-1}(\gamma_0))$, if $s(i) = \gamma_2$, then $s_a(i), s_b(i) \in \{\gamma_1, \gamma_2\}$,
(u4) for each $i \in [k] \setminus (s_a^{-1}(\gamma_0) \cup s_b^{-1}(\gamma_0))$, if $s(i) = \gamma_{-2}$, then $s_a(i), s_b(i) \in \{\gamma_1, \gamma_{-2}\}$.

The functions s_a and s_b inform about the indices to look at tab_{G_a} and tab_{G_b} in order to construct $tab_G[s]$. Let $(F, E_0, (p, w))$ be a solution in $\mathcal{A}_G[s]$, and assume that it is constructed from solutions $(F_a, E_a^0, (p_a, w_a))$ and $(F_b, E_b^0, (p_b, w_b))$, in respectively, $\mathcal{A}_{G_a}[s_a]$ and $\mathcal{A}_{G_b}[s_b]$. The first condition tells that if F_a (resp. F_b) does not intersect $lab_G^{-1}(i)$, then the intersection of F with $lab_G^{-1}(i)$ depends only on $V(F_b) \cap lab_G^{-1}(i)$ (resp. $V(F_a) \cap lab_G^{-1}(i)$), and so if $V(F)$ does not intersect $lab_G^{-1}(i)$, then F_a and F_b do not intersect $lab_G^{-1}(i)$. The second condition tells that

[2]If $J \subset [k]$ is the set of labels G_a (or G_b), we can extend the domain of any function $s' : J \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$ to $[k]$ by setting $s'(i) = \gamma_0$ for all $i \in [k] \setminus J$.

if $|F \cap lab_G^{-1}(i)| = 1$, then either $F \cap lab_G^{-1}(i) = F_a \cap lab_G^{-1}(i)$ or $F \cap lab_G^{-1}(i) = F_b \cap lab_G^{-1}(i)$. The other two conditions tell when F intersects both $lab_{G_a}^{-1}(i)$ and $lab_{G_b}^{-1}(i)$. Notice that we may have $s(i)$ set to γ_{-2} (or γ_2), while F_a and F_b each intersects $lab_G^{-1}(i)$ in exactly one vertex.

We let $tab_G[s] := \mathsf{ac\text{-}reduce}(\mathsf{rmc}(\mathcal{A}))$ where,

$$\mathcal{A} := \bigcup_{\substack{s_a, s_b \\ \text{u-agree on } s}} \mathsf{acjoin}\left(\mathsf{proj}(s^{-1}(\gamma_{-2}), tab_{G_a}[s_a]), \mathsf{proj}(s^{-1}(\gamma_{-2}), tab_{G_b}[s_b])\right).$$

The weighted partitions (p, w) added in $tab_G[s]$ are all the weighted partitions that are joins of weighted partitions (p_a, w_a) and (p_b, w_b) from $tab_{G_a}[s_a]$ and $tab_{G_b}[s_b]$, respectively. We need to do the projections before the join because we may have $s_a(i) = s_b(i) = \gamma_1$, $s(i) = \gamma_{-2}$, and there is j such that $s(j) = \gamma_2$ with j in the same block as i in both partitions p_a and p_b. If we do the projection after the acjoin operator, this latter will detect that $\mathsf{acyclic}(p_a, p_b)$ does not hold, and won't construct (p, w), which indeed corresponds to a solution in $\mathcal{A}_G[s]$.

Lemma 4.5. *Let $G = G_a \oplus G_b$ be a k-labeled graph. For each function $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$, the table $tab_G[s]$ ac-represents $\mathcal{A}_G[s]$ assuming that $tab_{G_a}[s']$ and $tab_{G_b}[s']$ ac-represent, respectively, $\mathcal{A}_{G_a}[s']$ and $\mathcal{A}_{G_b}[s']$, for each $s' : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$.*

Proof. Since the used operators preserve ac-representation, it is enough to prove that $\mathcal{A}_G[s] = \mathcal{A}$ if we let $tab_{G_t}[s'] = \mathcal{A}_{G_t}[s']$, for every $t \in \{a, b\}$ and $s' : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$.

Let $(F, E_0, (p, w))$ be a solution in $\mathcal{A}_G[s]$. We claim that $(p, w) \in \mathcal{A}$. For $t \in \{a, b\}$, let $F_t := G_t[V(F) \cap V(G_t)]$, $E_0^t := \{v_0 v \in E_0 : v \in V(F_t)\}$, and $w_t := \mathsf{w}(V(F_t))$, and let $s_t : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$ such that

$$s_t(i) := \begin{cases} \gamma_0 & \text{if } V(F_t) \cap lab_{G_t}^{-1}(i) = \emptyset, \\ \gamma_1 & \text{if } |V(F_t) \cap lab_{G_t}^{-1}(i)| = 1, \\ s(i) & \text{if } |V(F_t) \cap lab_{G_t}^{-1}(i)| \geq 2. \end{cases}$$

It is straightforward to verify that s_a and s_b u-agree on s. Observe that, by definition, $s_t^{-1}(\gamma_{-2}) \subseteq s^{-1}(\gamma_{-2})$ and $s_t^{-1}(\gamma_2) \subseteq s^{-1}(\gamma_2)$, for each $t \in \{a, b\}$. Let p_a and p_b be, respectively, partitions on $s_a^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\}$ and $s_b^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\}$ such that $(F_a, E_0^a, (p_a, w_a))$ and $(F_b, E_0^b, (p_b, w_b))$ are, respectively, candidate solutions in $\mathcal{A}_{G_a}[s_a]$ and $\mathcal{A}_{G_b}[s_b]$.

We claim that $(F_a, E_0^a, (p_a, w_a))$ is a solution in $\mathcal{A}_{G_a}[s_a]$. By definition of p_a, s_a, and of (F_a, E_0^a), Conditions (1) and (4) are clearly satisfied. Because $s_a^{-1}(\gamma_2) \subseteq s^{-1}(\gamma_2)$, and $F = F_a \oplus F_b$, we can conclude that $\mathbf{CG}(F_a, E_0^a, s_a)$ is an induced subgraph of $\mathbf{CG}(F, E_0, s)$, and because $\mathbf{CG}(F, E_0, s)$ is acyclic, we can conclude that $\mathbf{CG}(F_a, E_0^a, s_a)$ is acyclic, *i.e.*, Condition (2) is satisfied. If a connected component C of $\mathbf{CG}(F_a, E_0^a, s_a)$ does not intersect $s_a^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\}$, then C is entirely contained in $lab_{G_a}^{-1}(s_a^{-1}(\gamma_{-2}))$. But, this yields a contradiction with $(F, E_0, (p, w))$ satisfying Condition (3) because C is a connected component of $\mathbf{CG}(F, E_0, s)$, and $s_a^{-1}(\gamma_{-2}) \subseteq s^{-1}(\gamma_{-2})$. Therefore Condition (3) is also satisfied. We can thus conclude that $(F_a, E_0^a, (p_a, w_a))$ is a solution in $\mathcal{A}_{G_a}[s_a]$. Similarly, one can check that $(F_b, E_0^b, (p_b, w_b))$ is a solution in $\mathcal{A}_{G_b}[s_b]$.

It remains to prove that

$$(p, w) \in \mathsf{acjoin}(\mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_a, w_a)\}), \mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_b, w_b)\})).$$

First, recall that each connected component of F is either a connected component of F_a or of F_b. Then, because $(F, E_0, (p, w))$ satisfies Condition (3), we have that

$\mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_a, w_a)\})) \neq \emptyset$, and similarly $\mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_b, w_b)\})) \neq \emptyset$. We deduce that

$$p'_a := p_{a\downarrow(s^{-1}(\gamma_{-2}))} \in \mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_a, w_a)\}),$$
$$p'_b := p_{b\downarrow(s^{-1}(\gamma_{-2}))} \in \mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_b, w_b)\}).$$

Let $p''_a := p'_{a\uparrow([k]\setminus s^{-1}(\{\gamma_0,\gamma_{-2}\}))}$ and $p''_b := p'_{b\uparrow([k]\setminus s^{-1}(\{\gamma_0,\gamma_{-2}\}))}$. We claim that $\mathsf{acyclic}(p''_1, p''_2)$ holds. Assume towards a contradiction that it is not the case. We let $\sim_a$ (resp. $\sim_b$) be an equivalence relation on $[k] \setminus s^{-1}(\{\gamma_0, \gamma_{-2}\}) \cup \{v_0\}$ where $i \sim_a j$ (resp. $i \sim_b j$) if there is an i-vertex[3] and a j-vertex that are connected in $\mathbf{CG}(F_a, E_0^a, s_a)$ (resp. $\mathbf{CG}(F_b, E_0^b, s_b)$). By the graphical definition of $\mathsf{acyclic}$, we can easily see that if $\mathsf{acyclic}(p''_a, p''_b)$ does not hold, then there is a sequence $i_0, \ldots, i_{2r-1}$ of $s^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\}$ such that[4], for all $0 \leq \alpha < r-1$, we have $i_{2\alpha} \sim_b i_{2\alpha+1}$ and $i_{2\alpha+1} \sim_a i_{2\alpha+2}$. We can thus construct a cycle in $\mathbf{CG}(F, E_0, s)$ from this sequence since $V(F_a) \cap V(F_b) = \emptyset$, $\mathbf{CG}(F_a, E_0^a, s_a)$ and $\mathbf{CG}(F_b, E_0^b, s_b)$ are induced subgraphs of $\mathbf{CG}(F, E_0, s)$, and all the vertices labeled with a label from $s^{-1}(\gamma_2)$ are at distance at most two in $\mathbf{CG}(F, E_0, s)$. This yields a contradiction as $\mathbf{CG}(F, E_0, s)$ is acyclic by assumption. Therefore, $\mathsf{acyclic}(p''_1, p''_2)$ holds.

Finally, $p = p''_a \sqcup p''_b$ because one easily checks that there is an i-vertex x connected to a j-vertex y in $\mathbf{CG}(F, E_0, s)$ if and only if $i\mathcal{R}j$ where $\mathcal{R}$ is the transitive closure of ($i \sim_a j$ or $i \sim_b j$). This follows from the fact that for every $i \in [k] \setminus s^{-1}(\{\gamma_0, \gamma_{-2}\})$, either there is exactly one i-vertex in F or the i-vertices of F are all adjacent to v_i^+ in $\mathbf{CG}(F, E_0, s)$. In both cases, the i-vertices of F are in the same connected component of $\mathbf{CG}(F, E_0, s)$. Since, the equivalence classes of $\mathcal{R}$ correspond to the blocks of $p''_1 \sqcup p''_2$, and $w = w_a + w_b$, we can conclude that $(p, w) \in \mathsf{acjoin}(\mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_a, w_a)\}), \mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_b, w_b)\}))$.

We now prove that if (p, w) is added to $tab_G[s]$ from $(p_a, w_a) \in \mathcal{A}_{G_a}[s_a]$ and $(p_b, w_b) \in \mathcal{A}_{G_b}[s_b]$, then there exists a pair (F, E_0) such that $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_G[s]$. Let (F_a, E_0^a) and (F_b, E_0^b) such that $(F_a, E_0^a, (p_a, w_a))$ and $(F_b, E_0^b, (p_b, w_b))$ are solutions in, respectively, $\mathcal{A}_{G_a}[s_a]$ and $\mathcal{A}_{G_b}[s_b]$ with s_a and s_b u-agreeing on s. We claim that $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_G[s]$ with $F := (V(F_1) \cup V(F_2), E(F_1) \cup E(F_2))$ and $E_0 := E_0^a \cup E_0^b$. Because s_a and s_b u-agree on s, we clearly have that Condition (1) is satisfied.

Let $\sim_a$ and $\sim_b$ as defined above. Assume towards a contradiction that there exists a cycle C in $\mathbf{CG}(F, E_0, s)$. Since both $\mathbf{CG}(F_a, E_0^a, s_a)$ and $\mathbf{CG}(F_b, E_0^b, s_b)$ are acyclic, C must be a cycle alternating between paths in $\mathbf{CG}(F_a, E_0^a, s_a)$ and paths in $\mathbf{CG}(F_b, E_0^b, s_b)$. One can easily check that this implies the existence of a sequence $i_0, \ldots, i_{2r-1}$ of $s^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\}$ such that[4], for all $0 \leq \alpha < r-1$, we have $i_{2\alpha} \sim_b i_{2\alpha+1}$ and $i_{2\alpha+1} \sim_a i_{2\alpha+2}$. Moreover, it is easy to infer, from this sequence and the graphical definition of $\mathsf{acyclic}$, that $\mathsf{acyclic}\left(p'_{a\uparrow V}, p'_{b\uparrow V}\right)$ does not hold with $p'_a := p_{a\downarrow(s^{-1}(\gamma_{-2}))}$, $p'_b := p_{b\downarrow(s^{-1}(\gamma_{-2}))}$ and $V = ([k] \setminus s^{-1}(\{\gamma_0, \gamma_{-2}\}))$, contradicting the fact that $(p, w) = \mathsf{acjoin}(\mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_a, w_a)\}), \mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_b, w_b)\}))$. Therefore, $\mathbf{CG}(F, E_0, s)$ is acyclic and so Condition (2) is satisfied.

If we suppose that Condition (3) is not satisfied, then there is a connected component C of $\mathbf{CG}(F, E_0, s)$ that does not intersect $s^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\}$, *i.e.*, C is fully contained in $lab_G^{-1}(s^{-1}(\gamma_{-2}))$. Since $F = F_a \oplus F_b$, C is either a connected component of F_a or of F_b. Suppose *w.l.o.g.* that C is a connected component of F_a. Observe that C intersects $lab_{G_a}^{-1}(s_a^{-1}(\{\gamma_1, \gamma_2\}))$ because $(F_a, E_0^a, (p_a, w_a))$ is a solution in

[3]We consider v_0 as a v_0-vertex.

[4] The indexes are modulo $2r$.

$\mathcal{A}_{G_a}[s_a]$. Moreover, C does not intersect $lab_{G_a}^{-1}(s_a^{-1}(\gamma_2))$, otherwise C would intersect $lab_G^{-1}(s^{-1}(\gamma_2))$ since if $s_a(i) = \gamma_2$, then $s(i) = \gamma_2$, for all $i \in [k]$. Thus C is a connected component of $\mathbf{CG}(F_a, E_0^a, s_a)$ and $b_C := \{i \in s_a^{-1}(\gamma_1) : C \cap lab_{G_a}^{-1}(i) \neq \emptyset\}$ is a block of p_a because $(F_a, E_0^a, (p_a, w_a))$ is a candidate solution in $\mathcal{A}_{G_a}[s_a]$. Thus, by definition of proj, we have $\mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_a, w_a)\}) = \emptyset$, which contradicts the fact that $(p, w) = \mathsf{acjoin}(\mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_a, w_a)\}), \mathsf{proj}(s^{-1}(\gamma_{-2}), \{(p_b, w_b)\}))$. So, Condition (3) is also satisfied. We deduce that Condition (4) is satisfied by observing that, for $i, j \in s^{-1}(\{\gamma_1, \gamma_2\}) \cup \{v_0\}$, there is an i-vertex connected to a j-vertex in $\mathbf{CG}(F, E_0, s)$ if and only if $i\mathcal{R}j$ where $\mathcal{R}$ is the transitive closure of ($i \sim_a j$ or $i \sim_b j$). This concludes the proof that $(F, E_0, (p, w))$ is a solution in $\mathcal{A}_G[s]$. □

Theorem 4.6. *There is an algorithm that, given an n-vertex graph G and an irredundant k-expression of G, computes a minimum feedback vertex set in time $O(15^k \cdot 2^{(\omega+1)\cdot k} \cdot k^{O(1)} \cdot n)$.*

Proof. We do a bottom-up traversal of the k-expression and at each step we update the tables as indicated above. The correctness of the algorithm follows from Lemmas 4.3-4.5. From the definition of $\mathcal{A}_G[s]$, we conclude that the size of a maximum induced forest is the maximum over all $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$ with $s^{-1}(\gamma_2) = \emptyset$, of $\max\{w : (\{\{v_0\} \cup s^{-1}(\gamma_1)\}, w) \in tab_G[s]\}$ because $tab_G[s]$ ac-represents $\mathcal{A}_G[s]$ for all $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$.

Let us discuss the time complexity now. If $G = add_{i,j}(H)$ or $G = ren_{i\to j}(H)$, and $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$, then we update $tab_G[s]$ from a constant number of tables from tab_H, each identified in constant time from s. Since each table contains at most $2^{k-1} \cdot k$ entries, we call the function $\mathsf{ac\text{-}reduce}$ with a set of size at most $O(2^{k-1} \cdot k)$ as input. By Theorem 3.8, we can thus update tab_G in time $2^{\omega \cdot k} \cdot k^{O(1)}$. If $G = G_a \oplus G_b$, then we claim that the tables from tab_G are computable in time $O(15^k \cdot 2^{(\omega+1)\cdot k} \cdot k^{O(1)})$. For $s : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$, we let

$$\mathcal{A}[s] := \bigcup_{\substack{s_a, s_b \\ \text{u-agree on } s}} \mathsf{acjoin}\left(\mathsf{proj}(s^{-1}(\gamma_{-2}), tab_{G_a}[s_a]), \mathsf{proj}(s^{-1}(\gamma_{-2}), tab_{G_b}[s_b])\right).$$

By Theorem 3.8, computing $tab_G[s] := \mathsf{ac\text{-}reduce}(\mathsf{rmc}(\mathcal{A}[s]))$ can be done in time $|\mathcal{A}[s]| \cdot 2^{(\omega-1)\cdot k} \cdot k^{O(1)}$. Therefore, we can compute the tables from tab_G in time

$$\sum_{s:[k]\to\{\gamma_0,\gamma_1,\gamma_2,\gamma_{-2}\}} |\mathcal{A}[s]| \cdot 2^{(\omega-1)\cdot k} \cdot k^{O(1)}.$$

Now, observe that there are at most 15^k functions $s, s_a, s_b : [k] \to \{\gamma_0, \gamma_1, \gamma_2, \gamma_{-2}\}$ such that s_a and s_b u-agree on s. Indeed, for all $i \in [k]$, if s_a and s_b u-agree on s, then the tuple $(s_a(i), s_b(i), s(i))$ can take up to 15 values. See Table 1 for all the possible values.

	$s_b(i) = \gamma_0$	$s_b(i) = \gamma_1$	$s_b(i) = \gamma_2$	$s_b(i) = \gamma_{-2}$
$s_a(i) = \gamma_0$	γ_0	γ_1	γ_2	γ_{-2}
$s_a(i) = \gamma_1$	γ_1	γ_2, γ_{-2}	γ_2	γ_{-2}
$s_a(i) = \gamma_2$	γ_2	γ_2	γ_2	forbidden
$s_a(i) = \gamma_{-2}$	γ_{-2}	γ_{-2}	forbidden	γ_{-2}

TABLE 1. Possibles values of $s(i)$ depending on the value of $s_a(i)$ and $s_b(i)$ when s_a and s_b u-agree on s, there are 15 possible values for the tuple $(s_a(i), s_b(i), s(i))$.

Because each table of tab_{G_a} and tab_{G_b} contains at most $2^{k-1} \cdot k$ values, we have $|\mathsf{acjoin}\left(\mathsf{proj}(s^{-1}(\gamma_{-2}), tab_{G_a}[s_a]), \mathsf{proj}(s^{-1}(\gamma_{-2}), tab_{G_b}[s_b])\right)| \leq 2^{2k-2} \cdot k^2$. It follows that $\sum_{s:[k]\to\{\gamma_0,\gamma_1,\gamma_2,\gamma_{-2}\}} |\mathcal{A}[s]| \leq 15^k \cdot 2^{2k} \cdot k^2$. Hence, we can conclude that the tables from tab_G can be computed in time $O(15^k \cdot 2^{(\omega+1)\cdot k} \cdot k^{O(1)})$.

Because the size of a k-expression is $O(n \cdot k^2)$, we can conclude that a minimum feedback vertex set can be computed in the given time. □

5. Connected (Co-)(σ, ρ)-Dominating Sets

We will show here how to use the operators defined in [2] in order to obtain a $2^{O(d\cdot k)} \cdot n$ time algorithm for computing a minimum or a maximum connected (σ, ρ)-dominating set, given a k-expression, with d a constant that depends only on (σ, ρ). We deduce from this algorithm a $2^{O(k)} \cdot n^{O(1)}$ time algorithm for computing a minimum node-weighted Steiner tree, and a $2^{O(d\cdot k)} \cdot n^{O(1)}$ time algorithm for computing a maximum (or minimum) connected co-(σ, ρ)-dominating set.

We let $opt \in \{\min, \max\}$, *i.e.*, we are interested in computing a connected (σ, ρ)-dominating set of maximum (or minimum) weight if $opt = \max$ (or $opt = \min$). Let us first give some definitions. As defined in Section 3, rmc works only for the case $opt = \max$, we redefine it as follows in order to take into account minimization problems.

$$\mathsf{rmc}(\mathcal{A}) := \{(p, w) \in \mathcal{A} : \forall (p, w') \in \mathcal{A}, opt(w, w') = w\}.$$

Join. Let V' be a finite set. For $\mathcal{A} \subseteq \Pi(V) \times \mathbb{N}$ and $\mathcal{B} \subseteq \Pi(V') \times \mathbb{N}$, we define $\mathsf{join}(\mathcal{A}, \mathcal{B}) \subseteq \Pi(V \cup V') \times \mathbb{N}$ as

$$\mathsf{join}(\mathcal{A}, \mathcal{B}) := \{(p_{\uparrow V'} \sqcup q_{\uparrow V}, w_1 + w_2) : (p, w_1) \in \mathcal{A}, (q, w_2) \in \mathcal{B}\}.$$

This operator is the one from [2]. It is used mainly to construct partial solutions of $G \oplus H$ from partial solutions of G and H.

The following proposition assumes that $\log(|\mathcal{A}|) \leq |V|^{O(1)}$ for each $\mathcal{A} \subseteq \Pi(V) \times \mathbb{N}$ (this can be established by applying the operator rmc).

Proposition 5.1 (Folklore)**.** *The operator* join *can be performed in time* $|\mathcal{A}| \cdot |\mathcal{B}| \cdot |V \cup V'|^{O(1)}$ *and the size of its output is upper-bounded by* $|\mathcal{A}| \cdot |\mathcal{B}|$.

The following is the same as Definition 3.5, but does not require acyclicity.

Definition 5.2 ([2])**.** For $\mathcal{A} \subseteq \Pi(V) \times \mathbb{N}$, with V a finite set, and $q \in \Pi(V)$, let

$$\mathbf{opt}(\mathcal{A}, q) := opt\{w : (p, w) \in \mathcal{A}, p \sqcup q = \{V\}\}.$$

A set of weighted partitions $\mathcal{B} \subseteq \Pi(V) \times \mathbb{N}$ *represents* $\mathcal{A}$ if for each $q \in \Pi(V)$, it holds that $\mathbf{opt}(\mathcal{A}, q) = \mathbf{opt}(\mathcal{B}, q)$.

Let Z and V' be two finite sets. A function $f : 2^{\Pi(V)\times\mathbb{N}} \times Z \to 2^{\Pi(V')\times\mathbb{N}}$ is said to *preserve representation* if for each $\mathcal{A}, \mathcal{B} \subseteq \Pi(V) \times \mathbb{N}$ and $z \in Z$, it holds that $f(\mathcal{B}, z)$ represents $f(\mathcal{A}, z)$ whenever $\mathcal{B}$ represents $\mathcal{A}$.

Lemma 5.3 ([2])**.** *The operators* rmc, proj *and* join *preserve representation.*

Theorem 5.4 ([2])**.** *There exists an algorithm* reduce *that, given a set of weighted partitions* $\mathcal{A} \subseteq \Pi(V) \times \mathbb{N}$*, outputs in time* $|\mathcal{A}| \cdot 2^{(\omega-1)|V|} \cdot |V|^{O(1)}$ *a subset* $\mathcal{B}$ *of* $\mathcal{A}$ *that represents* $\mathcal{A}$*, and such that* $|\mathcal{B}| \leq 2^{|V|-1}$.

We use the following function to upper bound the amount of information we need to store in our dynamic programming tables concerning the (σ, ρ)-domination. For every non-empty finite or co-finite subset $\mu \subseteq \mathbb{N}$, we define $d(\mu)$ such as

$$d(\mu) := \begin{cases} 0 & \text{if } \mu = \mathbb{N}, \\ 1 + \min(\max(\mu), \max(\mathbb{N} \setminus \mu)) & \text{otherwise.} \end{cases}$$

For example, $d(\mathbb{N}^+) = 1$ and for every $c \in \mathbb{N}$, we have $d(\{0, \dots, c\}) = c + 1$.

The definition of d is motivated by the following observation which is due to the fact that, for all $\mu \subseteq \mathbb{N}$, if $d(\mu) \in \mu$, then μ is co-finite and contains $\mathbb{N} \setminus \{1, \dots, d(\mu)\}$.

Fact 5.5. For every $a, b \in \mathbb{N}$ and μ a finite or co-finite subset of $\mathbb{N}$, we have $a+b \in \mu$ if and only if $\min(d, a + b) \in \mu$.

Let us describe with a concrete example the information we need concerning the (σ, ρ)-domination. We say that a set $D \subseteq V(G)$ is a 2-dominating set if every vertex in $V(G)$ has at least 2 neighbors in D. It is worth noticing that a 2-dominating set is an $(\mathbb{N} \setminus \{0, 1\}, \mathbb{N} \setminus \{0, 1\})$-dominating set and $d(\mathbb{N} \setminus \{0, 1\}) = 2$. Let H be a k-labeled graph used in an irredundant k-expression of a graph G. Assuming $D_H \subseteq V(H)$ is a subset of a 2-dominating set D of G, we would like to characterize the sets $Y \subseteq V(G) \setminus V(H)$ such that $D \cup Y$ is a 2-dominating set of G. One first observes that D_H is not necessarily a 2-dominating set of H, and $D \setminus D_H$ also is not necessarily a 2-dominating set of $G[V(G) \setminus V(H)]$. Since we want to 2-dominate $V(H)$, we need to know for each vertex x in $V(H)$ how many neighbors it needs in addition to be 2-dominated by D_H. For doing so, we associate D_H with a sequence $R' = (r'_1, \dots, r'_k)$ over $\{0, 1, 2\}^k$ such that, for each $i \in [k]$, every vertex in $lab_H^{-1}(i)$ has at least $2 - r'_i$ neighbors in D_H. For example, if $r'_i = 1$, then every i-vertex has at least one neighbor in D_H. Notice that D_H can be associated with several such sequences. This sequence is enough to characterize what we need to 2-dominate $V(H)$ since the vertices with the same label in H have the same neighbors in the graph $(V(G), E(G) \setminus E(H))$ and each vertex needs at most 2 additional neighbors to be 2-dominated.

In order to update the sequence $R' = (r'_1, \dots, r'_k)$ associated with a set D_H, we associate with D_H another sequence $R = (r_1, \dots, r_k)$ over $\{0, 1, 2\}^k$ such that r_i corresponds to the minimum between 2 and the number of i-vertices in D_H, for each $i \in [k]$. This way, when we apply an operation $add_{i,j}$ on H, we know that every i-vertex has at least $2 - r'_i + r_j$ neighbors in D_H in the graph $add_{i,j}(H)$. For example, if $r_j = 1$ and every i-vertex has at least one neighbor in H that belongs D_H, then we know that every i-vertex is 2-dominated by D_H in the graph $add_{i,j}(H)$.

Let (σ, ρ) be a fixed pair of non-empty finite or co-finite subsets of $\mathbb{N}$. Let's first show how to compute an optimum connected (σ, ρ)-dominating set. We consider node-weighted Steiner tree and connected co-(σ, ρ)-dominating set at the end of the section. Let $d := \max\{d(\sigma), d(\rho)\}$.

The following definitions formalize the intuitions we give for 2-dominating set to the (σ, ρ)-domination.

Definition 5.6 (Certificate graph of a solution)**.** Let G be a k-labeled graph, and $R' := (r'_1, \dots, r'_k) \in \{0, \dots, d\}^k$. Let $V^+ := \{v_1^1, \dots, v_d^1, v_1^2, \dots, v_d^2, \dots, v_1^k, \dots, v_d^k\}$ be a set disjoint from $V(G)$ and of size $d \cdot k$. Let $V^+(R') := V_1^+(R') \cup \cdots \cup V_k^+(R')$ with

$$V_i^+(R') := \begin{cases} \emptyset & \text{if } r'_i = 0, \\ \{v_1^i, \dots, v_{r'_i}^i\} & \text{otherwise.} \end{cases}$$

The certificate graph of G with respect to R', denoted by $\mathbf{CG}(G, R')$, is the graph $(V(G) \cup V^+(R'), E(G) \cup E_1^+ \cup \cdots \cup E_k^+)$ with

$$E_i^+ = \{\{v, v_t^i\} \,:\, v_t^i \in V_i^+(R') \wedge v \in lab_G^{-1}(i)\}.$$

It is worth noticing that E_i^+ is empty if $lab_G^{-1}(i) = \emptyset$ or $V_i^+(R') = \emptyset$.

Definition 5.7. Let G be a k-labeled graph. For each $D \subseteq V(G)$ and $i \in [k]$, let $r_{i,G}^d(D) := \min(d, |lab_G^{-1}(i) \cap D|)$ and let $r_G^d(D) := (r_{1,G}^d(D), \dots, r_{k,G}^d(D))$.

The sequence $r_G^d(D)$ describes how each label class is intersected by D up to d vertices. Moreover, notice that $|\{r_G^d(D) : D \subseteq V(G)\}| \leq |\{0,\dots,d\}^k| \leq (d+1)^k$.

The motivation behind these two sequences is that for computing an optimum (σ,ρ)-dominating set, it is enough to compute, for any k labeled graph H used in an irredundant k-expression of a graph G and for each $R, R' \in \{0,\dots,d\}^k$, the optimum weight of a set $D \subseteq V(H)$ such that

- $r_H(D) = R$,
- $D \cup V^+(R')$ (σ,ρ)-dominates $V(H)$ in the graph $\mathbf{CG}(H, R')$.

It is worth noticing that the sequences $r_H^d(D)$ and R' are similar to the notion of *d-neighbor equivalence* introduced in [4].

We can assume w.l.o.g. that $d \neq 0$, that is $\sigma \neq \mathbb{N}$ or $\rho \neq \mathbb{N}$. Indeed, if $\sigma = \rho = \mathbb{N}$, then the problem of finding a minimum (or maximum) (co-)connected (σ,ρ)-dominating set is trivial. For computing an optimum connected (σ,ρ)-dominating set, we will as in Section 4 keep partitions of a subset of labels corresponding to the connected components of the sets D (that are candidates for the (σ,ρ)-domination). As $d \neq 0$, we know through $r_H(D)$ the label classes intersected by D. Moreover, we know through $R' = (r'_1,\dots,r'_k)$ whether the i-vertices in such a D will have a neighbor in any extension D' of D into a (σ,ρ)-dominating set. It is enough to keep the partition of the labels i with $r_{i,H}(D) \neq 0$ and $r'_i \neq 0$ that corresponds to the equivalence classes of the equivalence relation $\sim$ on $\{i \in [k] \setminus r_i \neq 0 \text{ and } r'_i \neq 0\}$ where $i \sim j$ if and only if an i-vertex is connected to a j-vertex in $\mathbf{CG}(G, R')[D \cup V^+]$.

We use the following definition to simplify the notations.

Definition 5.8. For $R = (r_1,\dots,r_k), R' = (r'_1,\dots,r'_k) \in \{0,\dots,d\}^k$, we define $\mathsf{active}(R, R') = \{i \in [k] : r_i \neq 0 \text{ and } r'_i \neq 0\}$.

We are now ready to define the sets of weighted partitions which representative sets we manipulate in our dynamic programming tables.

Definition 5.9 (Weighted partitions in $\mathcal{D}_G[R, R']$)**.** Let G be a k-labeled graph, and $R, R' \in \{0,\dots,d\}^k$. The entries of $\mathcal{D}_G[R, R']$ are all the weighted partitions $(p, w) \in \Pi(\mathsf{active}(R, R')) \times \mathbb{N}$ so that there exists a set $D \subseteq V(G)$ such that $\mathsf{w}(D) = w$, and

(1) $r_G^d(D) = R$,
(2) $D \cup V^+(R')$ (σ,ρ)-dominates $V(G)$ in $\mathbf{CG}(G, R')$,
(3) if $\mathsf{active}(R, R') = \emptyset$, then $G[D]$ is connected, otherwise for each connected component C of $G[D]$, we have $C \cap lab_G^{-1}(\mathsf{active}(R, R')) \neq \emptyset$,
(4) $p = \mathsf{active}(R, R')/\sim$ where $i \sim j$ if and only if an i-vertex is connected to a j-vertex in $\mathbf{CG}(G, R')[D \cup V^+(R')]$.

Conditions (1) and (2) guarantee that (p, w) corresponds to a set D that is coherent with R and R'. Condition (3) guarantees that each partial solution can be extended into a connected graph. Contrary to Section 4, the set of labels expected to play a role in the connectivity (*i.e.* $\mathsf{active}(R, R')$) can be empty. In this case, we have to make sure that the weighted partitions represent a connected solution. It is worth mentioning that $G[\emptyset]$, *i.e.* the empty graph, is considered as a connected graph. Observe that for each $(p, w) \in \mathcal{D}_G[R, R']$, the partition p has the same meaning as in Section 4. It is worth noticing that $\sim$ is an equivalence relation because if $i \in \mathsf{active}(R, R')$, then all the vertices in $lab_G^{-1}(i)$ are connected in $\mathbf{CG}(G, R')$ through the vertex in $V_i^+(R')$. In fact, the relation $\sim$ is equivalent to the transitive closure of the relation $\asymp$ where $i \asymp j$ if there exists an i-vertex and a j-vertex in the same connected component of $G[D]$.

In the sequel, we call a pair $(D,(p,\mathsf{w}(D)))$ a *candidate solution* in $\mathcal{D}_G[R,R']$ if $p=\mathsf{active}(R,R')/\sim$ where $i\sim j$ if and only if an i-vertex is connected to a j-vertex in $\mathbf{CG}(G,R')[D\cup V^+(R')]$. If in addition Conditions (1)-(3) are satisfied, we call $(D,(p,\mathsf{w}(D)))$ a *solution* in $\mathcal{D}_G[R,R']$.

It is straightforward to check that the weight of an optimum solution is the optimum over all $R\in\{0,\dots,d\}^k$ of $opt\{w:(\emptyset,w)\in\mathcal{D}_G[R,\{0\}^k]\}$ for a k-labeled graph G.

Analogously to Section 4 our dynamic programming algorithm will store a subset of $\mathcal{D}_G[R,R']$ of size 2^{k-1} that represents $\mathcal{D}_G[R,R']$. Recall that we suppose that any graph is given with an irredundant k-expression.

Computing tab_G for $G=\mathbf{1}(x)$. For $(r_1)\in\{0,\dots,d\},(r'_1)\in\{0,\dots,d\}$, let

$$tab_G[(r_1),(r'_1)]:=\begin{cases}\{(\emptyset,0)\} & \text{if } r_1=0 \text{ and } r'_1\in\rho,\\ \{(\{\{1\}\},\mathsf{w}(x))\} & \text{if } r_1=1 \text{ and } r'_1\in\sigma,\\ \emptyset & \text{otherwise.}\end{cases}$$

Since there is only one vertex in G labeled 1, $\mathcal{D}_G[(r_1),(r'_1)]$ is empty whenever $r_1\notin\{0,1\}$. Also, the possible solutions are either to put x in the solution ($r_1=1$) or to discard it ($r_1=0$); in both cases we should check that x is (σ,ρ)-dominated by $V_1^+(R')$. We deduce then that $tab_G[(r_1),(r'_1)]=\mathcal{D}_G[(r_1),(r'_1)]$.

Computing tab_G for $G=ren_{i\to j}(H)$. We can suppose that H is k-labeled and that $i=k$. Let $R=(r_1,\dots,r_{k-1}),R'=(r'_1,\dots,r'_{k-1})\in\{0,\dots,d\}^{k-1}$.

To compute $tab_G[R,R']$, we define $\mathcal{S}$, the set of tuples coherent with respect to R and Condition (1), as follows

$$\mathcal{S}:=\{(s_1,\dots,s_k)\in\{0,\dots,d\}^k:\ r_j=\min(d,s_k+s_j)\text{ and }\forall\ell\in[k]\setminus\{i,j\},s_\ell=r_\ell\}.$$

It is worth noticing that we always have $(r_1,\dots,r_{k-1},0)\in\mathcal{S}$. Moreover, if $r_j=0$, then $\mathcal{S}=\{(r_1,\dots,r_{k-1},0)\}$. We define also $S'=(s'_1,\dots,s'_k)\in\{0,\dots,d\}^k$ with $s'_k=r'_j$ and $s'_\ell=r'_\ell$ for all $\ell\in[k-1]$. Notice that S' is the only tuple compatible with R' and Condition (2) since for every $v\in lab_G^{-1}(j)$, the number of vertices in $V^+(R')$ adjacent to v in $\mathbf{CG}(G,R')$ is the same as the number of vertices in $V^+(S')$ adjacent to v in $\mathbf{CG}(H,S')$.

(a) If $r_j=0$ or $r'_j=0$, then we let

$$tab_G[R,R']:=\mathsf{reduce}\left(\mathsf{rmc}\left(\bigcup_{S\in\mathcal{S}}tab_H[S,S']\right)\right).$$

In this case, the vertices in $lab_G^{-1}(j)$ are not expected to play a role in the future as either we expect no neighbors for them in the future or they are not intersected by the partial solutions.

(b) Otherwise, we let $tab_G[R,R']:=\mathsf{reduce}(\mathsf{rmc}(\mathcal{A}))$ with

$$\mathcal{A}:=\mathsf{proj}\left(\{k\},\bigcup_{S\in\mathcal{S}}\mathsf{join}(tab_H[S,S'],\{(\{\{j,k\}\},0)\})\right).$$

Intuitively, we put in $tab_G[R,R']$ all the weighted partitions (p,w) from the tables $tab_H[S,S']$ with $S\in\mathcal{S}$, after merging the blocks in p containing k and j, and removing k from the resulting partition.

Lemma 5.10. *Let $G=ren_{k\to j}(H)$ with H a k-labeled graph. For all $R=(r_1,\dots,r_{k-1}),R'=(r'_1,\dots,r'_{k-1})\in\{0,\dots,d\}^{k-1}$, the table $tab_G[R,R']$ is a representative set of $\mathcal{D}_G[R,R']$ assuming that $tab_H[S,S']$ is a representative set of $\mathcal{D}_H[S,S']$ for all $S\in\{0,\dots,d\}^k$.*

Proof. Since the used operators preserve representation, it is enough to prove that each weighted partition added to $tab_G[R, R']$ belongs to $\mathcal{D}_G[R, R']$, and that

- in Case (a), we have $\mathcal{D}_G[R, R'] \subseteq \bigcup_{S \in \mathcal{S}} \mathcal{D}_H[S, S']$, and
- in Case (b), we have $\mathcal{D}_G[R, R'] \subseteq \mathcal{A}$ if we let $tab_H[S, S'] = \mathcal{D}_H[S, S']$ for every $S \in \{0, \ldots, d\}^k$.

Let $(D, (p, w))$ be a solution in $\mathcal{D}_G[R, R']$. We start by proving that we have $(p, w) \in \bigcup_{S \in \mathcal{S}} \mathcal{D}_H[S, S']$ if we are in Case (a), or $(p, w) \in \mathcal{A}$ if we are in Case (b). By the definition of $\mathcal{S}$, we deduce that $r_H(D) \in \mathcal{S}$. Indeed, $r_{j,G}(D) = \min(d, |D \cap lab_G^{-1}(j)|)$ equals $\min(d, r^d_{j,H}(D) + r^d_{k,H}(D))$ because $lab_G^{-1}(j) = lab_H^{-1}(\{j, k\})$.

Let $p' \in \Pi(\mathsf{active}(r^d_H(D), S'))$ such that $(D, (p', w))$ is a candidate solution in $\mathcal{D}_H[r^d_H(D), R']$. We claim that $(D, (p', w))$ is a solution in $\mathcal{D}_H[r^d_H(D), R']$. Condition (1) is trivially satisfied. We deduce from the definition of S' that Condition (2) is satisfied. We claim that Condition (3) is satisfied. If $R' = \{0\}^{k-1}$, then we have $S' = \{0\}^k$ and Condition (3) is satisfied because $H[D] = G[D]$ must be connected since $(D, (p, w))$ is a solution in $\mathcal{D}_G[R, R']$. Otherwise, every connected component C of $G[D] = H[D]$ intersects $lab_G^{-1}(\mathsf{active}(R, R'))$. Let C be a connected component of $H[D]$. If C contains a vertex labeled l in G with $l \in \mathsf{active}(R, R') \setminus \{j\}$, then by definition of S', we have $\ell \in \mathsf{active}(r^d_H(D), S')$. Suppose now, C contains a vertex v in $lab_G^{-1}(j)$ and $j \in \mathsf{active}(R, R')$. If v is labeled k in H, then $r^d_{k,H}(D) \neq 0$ and thus k belongs to $\mathsf{active}(r^d_H(D), S')$ because $s'_k = r'_j \neq 0$. Symmetrically, if v is labeled j in H, then $j \in \mathsf{active}(r^d_H(D), S')$. In both cases, C intersects $lab_H^{-1}(\mathsf{active}(r^d_H(D), S'))$. We can conclude that Condition (3) is satisfied. Hence, $(D, (p', w))$ is a solution in $\mathcal{D}_H[r^d_H(D), R']$.

If $r_j = 0$ or $r'_j = 0$ (Case (a)), then it is easy to see that $p = p'$ from Equation (3) and thus $(p, w) \in tab_H[S, S']$.

Assume now that $r_j \neq 0$ and $r'_j \neq 0$ (Case (b)). Let $D_k := D \cap lab_H^{-1}(k)$ and $D_j := D \cap lab_H^{-1}(j)$. Observe that the graph $\mathbf{CG}(G, R')[D \cup V^+(R')]$ can be obtained from the graph $\mathbf{CG}(H, S')[D \cup V^+(S')]$ by removing the vertices in $V_k^+(R')$ and by adding the edges between $V_j^+(R')$ and D_k. Hence, p is obtained from p' by merging the blocks containing j and k, and by removing k. Thus, we can conclude that $(p, w) \in \mathcal{A}$.

It remains to prove that each weighted partition (p, w) added to $tab_G[R, R']$ belongs to $\mathcal{D}_G[R, R']$. Let (p, w) be a weighted partition added to $tab_G[R, R']$ from $(p', w) \in tab_H[S, S']$, and let $D \subseteq V(H)$ such that $(D, (p', w))$ is a solution in $\mathcal{D}_H[S, S']$. We want to prove that $(D, (p, w))$ is a solution in $\mathcal{D}_G[R, R']$. From the definitions of $\mathcal{S}$ and S', we deduce that D satisfies Conditions (1) and (2). We deduce that D satisfies also Condition (3) from Equation (3), the fact that $lab_G^{-1}(j) = lab_H^{-1}(\{j, k\})$, and because $G[D] = H[D]$.

If $r_j = 0$ or $r'_j = 0$, then it is easy to see from Equation (3) that $p = p'$ and that $(D, (p, w))$ satisfies Condition (4). Otherwise, we deduce from the previous observations concerning the differences between $\mathbf{CG}(G, R')[D \cup V^+(R')]$ and $\mathbf{CG}(H, S')[D \cup V^+(S')]$, that $(D, (p, w))$ satisfies Condition (4). In both cases, we can conclude that $(D, (p, w))$ is a solution in $\mathcal{D}_G[R, R']$. □

Computing tab_G for $G = add_{i,j}(H)$. We can suppose that H is k-labeled. Let $R = (r_1, \ldots, r_k) \in \{0, \ldots, d\}^k, R' = (r'_1, \ldots, r'_k) \in \{0, \ldots, d\}^k$.

Let $S' := (s'_1, \ldots, s'_k) \in \{0, \ldots, d\}^k$ such that $s'_i := \min(d, r'_i + r_j)$, $s'_j := \min(d, r'_j + r_i)$, and $s'_\ell = r'_\ell$ for all $\ell \in [k] \setminus \{i, j\}$. It is easy to see that S' is the only tuple compatible with R' and Condition (2).

(a) If $\mathsf{active}(R, R') = \emptyset$, then we let

$$tab_G[R, R'] := \mathsf{rmc}\left(\{(\emptyset, w) \,:\, (p, w) \in tab_H[R, S']\}\right).$$

In this case, the partial solution in $tab_G[R, R']$ are associated with connected solutions by Condition (3). The partial solutions in $tab_H[R, S']$ trivially satisfy this condition in G. Notice that $tab_G[R, R']$ represents $\mathcal{D}_G[R, R']$ because the function $f : 2^{\Pi(V)\times\mathbb{N}} \to 2^{\{\emptyset\}\times\mathbb{N}}$ with $f(\mathcal{A}) := \{(\emptyset, w) \,:\, (p, w) \in \mathcal{A}\}$ preserves representation.

(b) If $r_i = 0$ or $r_j = 0$, we let $tab_G[R, R'] := tab_H[R, S']$. We just copy all the solutions not intersecting $lab_G^{-1}(i)$ or $lab_G^{-1}(j)$. In this case, the connectivity of the solutions is not affected by the $add_{i,j}$ operation.

(c) Otherwise, we let $tab_G[R, R'] := \mathsf{rmc}(\mathcal{A})$, where

$$\mathcal{A} := \mathsf{proj}(\{t \in \{i, j\} \,:\, r'_t = 0\}, \mathsf{join}(tab_H[R, S'], \{(\{\{i, j\}\}, 0)\})).$$

In this last case, we have $i, j \in \mathsf{active}(R, S')$. We put in $tab_G[R, R']$, the weighted partitions of $tab_H[R, S']$ after merging the blocks containing i and j, and removing i or j if, respectively, $r'_i = 0$ and $r'_j = 0$, *i.e.*, if they don't belong, respectively, to $\mathsf{active}(R, R')$.

It is worth noticing that if $|tab_H[R, S']| \leq 2^{k-1}$, then we have $|tab_G[R, S']| \leq 2^{k-1}$. Thus, we do not have to use the function reduce to compute tab_G.

Lemma 5.11. *Let $G = add_{i,j}(H)$ be a k-labeled graph. For all tuples $R = (r_1, \ldots, r_k), R' = (r'_1, \ldots, r'_k) \in \{0, \ldots, d\}^k$, the table $tab_G[R, R']$ is a representative set of $\mathcal{D}_G[R, R']$ assuming that $tab_H[R, S']$ is a representative set of $\mathcal{D}_H[R, S']$.*

Proof. Since the used operators preserve representation, it is enough to prove that every weighted partition added to $tab_G[R, R']$ belongs to $\mathcal{D}_G[R, R']$, and that

- in Case (a), we have $\mathcal{D}_G[R, R'] \subseteq \{(\emptyset, w) \,:\, (p, w) \in \mathcal{D}_H[R, S']\}$,
- in Case (b), we have $\mathcal{D}_G[R, R'] \subseteq \mathcal{D}_H[R, S']$, and
- in Case (c), we have $\mathcal{D}_G[R, R'] \subseteq \mathcal{A}$ if we let $tab_H[R, S'] = \mathcal{D}_H[R, S']$.

Let $(D, (p, w))$ be a solution in $\mathcal{D}_G[R, R']$. Let $p' \in \Pi(\mathsf{active}(R, S'))$ such that $(D, (p', w))$ is a candidate solution in $\mathcal{D}_H[R, S']$. We claim that $(D, (p', w))$ is a solution in $\mathcal{D}_H[R, S']$. Condition (1) is trivially satisfied because $lab_G(v) = lab_H(v)$ for all $v \in V(G)$. We claim that Condition (2) is satisfied, that is $D \cup V^+(S')$ (σ, ρ) dominates $V(H)$ in $\mathbf{CG}(H, S')$. It is quite easy to see that $D \cup V^+(S')$ (σ, ρ) dominates $V(H) \setminus lab_H^{-1}(\{i, j\})$. Let $v \in lab_H^{-1}(i)$. We claim that v is (σ, ρ) dominated by $D \cup V^+(S')$. Because we consider only irredundant k-expressions, there is no edge between an i-vertex and a j-vertex in H. Therefore, we have $|N_G(v) \cap D| = |N_H(v) \cap D| + |D \cap lab_G^{-1}(j)|$. Since $D \cup V^+(R')$ (σ, ρ)-dominates $V(G)$, if $v \in D$, then $|N_G(v) \cap D| + r'_i \in \sigma$, otherwise, $|N_G(v) \cap D| + r'_i \in \rho$. By Fact 5.5, we conclude that $\min(d, |N_H(v) \cap D| + |D \cap lab_G^{-1}(j)| + r'_i)$ belongs to σ if $v \in D$, otherwise it belongs to ρ. As $\min(d, r'_i + |D \cap lab_G^{-1}(j)|) = \min(d, r'_i + r_j) = s'_i$, we deduce that $D \cup V^+(S')$ (σ, ρ) dominates v. Thus, every i-vertex is (σ, ρ)-dominated by $D \cup V^+(S')$ in $\mathbf{CG}(H, S')$. Symmetrically, we deduce that every j-vertex is (σ, ρ)-dominated by $D \cup V^+(S')$ in $\mathbf{CG}(H, S')$. Hence, Condition (2) is satisfied.

In order to prove that Condition (3) is satisfied, we distinguish the following cases. First, suppose that $\mathsf{active}(R, R') = \emptyset$. By Condition (3), $G[D]$ is connected. As $G = add_{i,j}(H)$, the graph $H[D]$ is obtained from $G[D]$ by removing all edges between the i-vertices and the j-vertices. We deduce that either $H[D]$ is connected (if $r_i = 0$ or $r_j = 0$), or that every connected component of $H[D]$ contains at least one vertex whose label is i or j (otherwise $G[D]$ would not be connected). In both cases, Condition (3) is satisfied.

Assume now that $\mathsf{active}(R, R') \neq \emptyset$. If $r_i = 0$ (resp. $r_j = 0$), then, by definition of S', we have $s'_j = r'_j$ (resp. $s'_i = r'_i$). Therefore, if $r_i = 0$ or $r_j = 0$, then we have $\mathsf{active}(R, R') = \mathsf{active}(R, S')$, and Condition (3) is trivially satisfied. Otherwise, if $r_i \neq 0$ and $r_j \neq 0$, then, by definition of S', we have $s'_i \neq 0$, $s'_j \neq 0$, and thus $i, j \in \mathsf{active}(R, S')$. In this case, we conclude that Condition (3) is satisfied because, for every connected component C of $H[D]$, either C is a connected component of $G[D]$, or C contains at least one vertex whose label is i or j.

Hence, $(D, (p', w))$ is a solution in $\mathcal{D}_H[R, S']$. If $\mathsf{active}(R, R') = \emptyset$, then $p = \emptyset$ and (p, w) is added in $tab_G[R, R']$. Else if $r_i = 0$ or $r_j = 0$, then $p' = p$ and (p, w) is also added to $tab_G[R, R']$. Otherwise, it is easy to see that p is obtained from p' by merging the blocks of p' containing i and j, and by removing them if they belong to $\{\ell \in [k] : r'_\ell \neq 0\}$. Thus, we can conclude that $(p, w) \in \{(\emptyset, w) : (p, w) \in \mathcal{D}_H[R, S']\})$ in Case (a), $(p, w) \in \mathcal{D}_H[R, S']$ in Case (b), and $(p, w) \in \mathcal{A}$ in Case (c).

It remains to prove that every weighted partition added to $tab_G[R, R']$ belongs to $\mathcal{D}_G[R, R']$. Let (p, w) be a weighted partition added to $tab_G[R, R']$ from $(p', w) \in tab_H[R, S']$ and let $D \subseteq V(H)$ such that $(D, (p', w))$ is a solution in $\mathcal{D}_H[R, S']$. We claim that $(D, (p, w))$ is a solution in $\mathcal{D}_G[R, R']$. Obviously, Condition (1) is satisfied. We deduce that Condition (2) is satisfied from the definition of S' and the previous arguments. We claim that Condition (3) is satisfied. Suppose first that $R' = \{0\}^k$. Then either $S' = \{0\}^k$ or $\{i, j\} = \mathsf{active}(R, S')$. Indeed, we have $\mathsf{active}(R, S') \subseteq \{i, j\}$. Now, $i \in \mathsf{active}(R, S')$ if and only if $r_i \neq 0$ and $s'_i \neq 0$, and then, by definition of S', $s'_j = r_i \neq 0$ and $s'_i = r_j \neq 0$. Thus, $i \in \mathsf{active}(R, S')$ if and only if $j \in \mathsf{active}(R, S')$. Therefore, we conclude that either $H[D]$ is connected or every connected component C of $H[D]$ contains at least a vertex whose label is i or j. As $G[D]$ is obtained from $H[D]$ by adding all the edges between the i-vertices and the j-vertices, we conclude that, in both cases, $G[D]$ is connected.

Otherwise, if $R' \neq \{0\}^k$, then every connected component of $H[D]$ contains at least one vertex whose label is in $\mathsf{active}(R, S')$. If $r_i = 0$ or $r_j = 0$, then we are done because $\mathsf{active}(R, R') = \mathsf{active}(R, S')$ by definition of S'. Suppose now that $r_i \neq 0$ and $r_j \neq 0$. In this case, we have $i, j \in \mathsf{active}(R, S')$ from the definition of S'. Assume towards a contradiction that there exists a connected component C in $G[D]$ such that C does not intersect $lab_G^{-1}(\mathsf{active}(R, R'))$. Notice that C intersects $lab_G^{-1}(\mathsf{active}(R, S'))$ because C is a union of connected components of $H[D]$. As $\mathsf{active}(R, S') \setminus \mathsf{active}(R, R') \subseteq \{i, j\}$, we deduce that C contains at least one vertex whose label is i or j. Since the i-vertices and the j-vertices of D are in the same connected component of $G[D]$, we have $lab_G^{-1}(\{i, j\}) \cap D \subseteq C$. Therefore, we conclude that $\mathsf{active}(R, S') \setminus \mathsf{active}(R, R') = \{i, j\}$. It follows, that all the connected components of $H[D]$ that intersect $lab_G^{-1}(\{i, j\})$ does not intersect $lab_G^{-1}(\mathsf{active}(R, S') \setminus \{i, j\})$ because they are contained in C. We can conclude that $\{i, j\}$ is a block of $p'' := p' \sqcup \{\{i, j\}\}_{\uparrow \mathsf{active}(R, S')}$. Hence, we have $\mathsf{proj}(\{t \in \{i, j\} : r'_t = 0\}, \{(p'', w)\}) = \emptyset$ because $\{t \in \{i, j\} : r'_t = 0\} = \{i, j\}$. This contradicts the fact that (p, w) is obtained from (p', w'). Thus, Condition (3) is satisfied.

We deduce from the previous observations concerning Condition (4) that this condition is also satisfied. Thus, every solution (p, w) added to $tab_G[R, R']$ belongs to $\mathcal{D}_G[R, R']$. □

Computing tab_G for $G = G_a \oplus G_b$. We can suppose w.l.o.g. that G_a and G_b are k-labeled[5]. Let $R = (r_1, \ldots, r_k), R' = (r'_1, \ldots, r'_k) \in \{0, \ldots, d\}^k$.

[5]For example, if the set of labels of G_a (or G_b) is $[k] \setminus 1$, then we can extend any tuples $R = (r_2, \ldots, r_k), R' = (r'_2, \ldots, r'_k) \in \{0, \ldots, d\}^{k-1}$, to $\{0, \ldots, d\}^k$ by adding $r_1 = 0$ to R and $r'_1 = *$ to R' with $*$ a special value considered as equal to all the integers in $\{0, \ldots, d\}$.

Let $A = (a_1, \dots, a_k), B = (b_1, \dots, b_k) \in \{0, \dots, d\}^k$. The following notion characterizes the pairs (A, B) compatible with R with respect to Condition (1). We say that (A, B) is R-*compatible* if and only if for all $i \in [k]$, we have $r_i = \min(d, a_i + b_i)$.

(a) If $R' = \{0\}^k$, then we let $tab_G[R, R'] := \mathsf{reduce}(\mathsf{rmc}(tab_{G_a}[R, R'] \cup tab_{G_b}[R, R']))$ if $0 \in \rho$, otherwise we let $tab_G[R, R'] = \emptyset$. Condition (3) implies that the partial solutions in $\mathcal{D}_G[R, R']$ are either fully contained in $V(G_a)$ or in $V(G_b)$ since there are no edges between these vertex sets in G. Moreover, in order to satisfy Condition (2), we must have $0 \in \sigma$.

(b) Otherwise, we let $tab_G[R, R'] := \mathsf{reduce}(\mathsf{rmc}(\mathcal{A}))$ where

$$\mathcal{A} := \bigcup_{(A,B) \text{ is } R\text{-compatible}} \mathsf{join}(tab_{G_a}[A, R'], tab_{G_b}[B, R']).$$

Lemma 5.12. *Let $G = G_a \oplus G_b$ be a k-labeled graph. For all $R = (r_1, \dots, r_k), R' = (r'_1, \dots, r'_k) \in \{0, \dots, d\}^k$, the table $tab_G[R, R']$ is a representative set of $\mathcal{D}_G[R, R']$ assuming that $tab_{G_a}[A, R']$ and $tab_{G_a}[B, R']$ are representative sets of $\mathcal{D}_{G_a}[A, R']$ and $\mathcal{D}_{G_b}[B, R']$, respectively, for all $A, B \in \{0, \dots, d\}^k$.*

Proof. Since the used operators preserve representation, it is easy to see that if $R' = \{0\}^k$, then we are done as $\mathcal{D}_G[R, R'] = \mathcal{D}_{G_a}[R, R'] \cup \mathcal{D}_{G_b}[R, R']$ if $0 \in \rho$, otherwise $\mathcal{D}_G[R, R'] = \emptyset$. Indeed, by Condition (3), for all solutions $(D, (p, w))$ in $\mathcal{D}_G[R, R']$, the graph $G[D]$ must be connected. Since $G = G_a \oplus G_b$, there are no edges between $V(G_a)$ and $V(G_b)$ in $G[D]$. Thus, D is either included in $V(G_a)$ or in $V(G_b)$. Since $V(G_a) \neq \emptyset$ and $V(G_b) \neq \emptyset$, we have $V(G) \setminus D \neq \emptyset$. This implies that $0 \in \rho$, otherwise the vertices in $V(G) \setminus D$ will not be (σ, ρ)-dominated by $D \cup V^+(R') = D$ in $\mathbf{CG}(G, R')$ as $V^+(R') = \emptyset$.

In the following, we assume that $R' \neq \{0\}^k$. Since the used operators preserve representation, it is enough to prove that $\mathcal{D}_G[R, R'] = \mathcal{A}$ if we let $tab_{G_t}[S, R'] = \mathcal{D}_{G_t}[S, R']$ for all $S \in \{0, \dots, d\}^k$ and $t \in \{a, b\}$.

Let $(D, (p, w))$ be a solution in $\mathcal{D}_G[R, R']$. We start by proving that $(p, w) \in \mathcal{A}$. Let $D_a = D \cap V(G_a)$ and $D_b = D \cap V(G_b)$. From the definition of R-compatibility, we deduce that $r^d_{G_a}(D_a)$ and $r^d_{G_b}(D_b)$ are R-compatible. Indeed, we have $\min(d, |lab_G^{-1}(i) \cap D|) = \min(d, r^d_{G_a}(D_a) + r^d_{G_b}(D_b))$ for all $i \in [k]$.

Let $p_a \in \Pi(\mathsf{active}(r^d_{G_a}(D_a), R'))$ such that $(D_a, (p_a, \mathsf{w}(D_a)))$ is a candidate solution in $\mathcal{D}_{G_a}[r^d_{G_a}(D_a), R']$. We claim that the pair $(D_a, (p_a, \mathsf{w}(D_a)))$ is a solution in $\mathcal{D}_{G_a}[r^d_{G_a}(D_a), R']$. Condition (1) is trivially satisfied. By assumption, $D \cup V^+(R')$ (σ, ρ) dominates $V(G)$ in $\mathbf{CG}(G, R')$. Since there are no edges between the vertices in $V(G_a)$ and those in $V(G_b)$, we conclude that $D_a \cup V^+(R')$ (σ, ρ)-dominates $V(G_a)$ in $\mathbf{CG}(G_a, R')$. That is Condition (2) is satisfied. Observe that every connected component of $G[D]$ is either included in D_a or in D_b. Therefore, every connected component of $G_a[D_a]$ contains a vertex v with a label $j \in \mathsf{active}(R, R')$. Since $v \in D_a$, we conclude that $r^d_{j,G_a}(D_a) \neq 0$, thus $j \in \mathsf{active}(r^d_{G_a}(D_a), R')$. We can conclude that Condition (3) is satisfied. Thus, $(D_a, (p_a, \mathsf{w}(D_a)))$ is a solution in $\mathcal{D}_{G_a}[r^d_{G_a}(D_a), R']$. Symmetrically, we deduce that there exits $p_b \in \mathsf{active}(R^d_{G_b}(D_b), R')$ such that $(D_b, (p_b, \mathsf{w}(D_b)))$ is a solution in $\mathcal{D}_{G_b}[r^d_{G_b}(D_b), R']$.

It remains to prove that $p = p_{a\uparrow V} \sqcup p_{b\uparrow V}$ with $V = \mathsf{active}(R, R')$. First, observe that $\mathsf{active}(r^d_{G_a}(D_a), R')$ and $\mathsf{active}(r^d_{G_b}(D_b, R'))$ are both subset of $\mathsf{active}(R, R')$ and thus $p_{a\uparrow V} \sqcup p_{b\uparrow V}$ is a partition of $\mathsf{active}(R, R')$. Let $\sim_a$ (resp. $\sim_b$) be the equivalence relation such that $i \sim_a j$ (resp. $i \sim_b j$) if an i-vertex is connected to a j-vertex in the graph $\mathbf{CG}(G_a, R')[D_a \cup V^+(R')]$ (resp. $\mathbf{CG}(G_b, R')[D_b \cup V^+(R')]$). By Condition (4), two labels i, j are in the same block of p if and only if an i-vertex and a j-vertex are connected in $\mathbf{CG}(G, R')[D \cup V^+(R')]$. On the other hand, i and

j are in the same block of $p_{a\uparrow V} \sqcup p_{b\uparrow V}$ if and only if $i\mathcal{R}j$ where $\mathcal{R}$ is the transitive closure of the relation ($i \sim_a j$ or $i \sim_b j$). By definition of $\mathbf{CG}(G, R')$, for every label $i \in \mathsf{active}(R, R')$, the vertices in $lab_G^{-1}(i) \cap D$ are all adjacent to the vertices in $V_i^+(R')$. One can easily deduce from these observations that $p = p_{a\uparrow V} \sqcup p_{b\uparrow V}$. Hence, $(p, w) \in \mathcal{A}$.

We now prove that every weighted partition in $\mathcal{A}$ belongs to $\mathcal{D}_G[R, R']$. Let (p, w) be a weighted partition added to $tab_G[R, R']$ from a solution $(D_a, (p_a, w_a))$ in $\mathcal{D}_{G_a}[A, R']$ and a solution $(D_b, (p_b, w_b))$ in $\mathcal{D}_{G_b}[B, R']$. We claim that $(D_a \cup D_b, (p_{a\uparrow V} \sqcup p_{b\uparrow V}, w_a + w_b))$ is a solution in $\mathcal{D}_G[R, R']$ with $V = \mathsf{active}(R, R')$. We deduce that Condition (1) is satisfied from the definition of R-compatibility and because $\min(d, |lab_G^{-1}(i) \cap D|) = \min(d, r^d_{G_a}(D_a) + r^d_{G_b}(D_b))$ for all $i \in [k]$. With the same arguments given previously, one easily deduces that Conditions (2)-(4) are also satisfied. We conclude that $(D_a \cup D_b, (p_{a\uparrow V} \sqcup p_{b\uparrow V}, w_a + w_b))$ is a solution in $\mathcal{D}_G[R, R']$. □

Theorem 5.13. *There is an algorithm that, given an n-vertex graph G and an irredundant k-expression of G, computes a maximum (or a minimum) connected (σ, ρ)-dominating set in time $(d+1)^{3k} \cdot 2^{(\omega+1)\cdot k} \cdot k^{O(1)} \cdot n$ with $d = \max(d(\sigma), d(\rho))$.*

Proof. We do a bottom-up traversal of the k-expression and at each step we update the tables as indicated above. The correctness of the algorithm follows from Lemmas 5.10-5.12. From the definition of $\mathcal{D}_G[R, R']$, we deduce that the weight of an optimum connected (σ, ρ)-dominating set corresponds to the optimum over all $R \in \{0, \ldots, d\}^k$ of $\mathbf{opt}\{w : (\emptyset, w) \in tab_G[R, \{0\}^k]\}$ because $tab_G[R, \{0\}^k]$ represents $\mathcal{D}_G[R, \{0\}^k]$.

Let us discuss the time complexity now. We claim that the tables of tab_G can be computed in time $(d+1)^{3k} \cdot 2^{(\omega+1)\cdot k} \cdot k^{O(1)}$. We distinguish the following cases:

- If $G = \mathbf{1}(x)$, then it is easy to see that tab_G is computable in time $O(d)$.
- If $G = add_{i,j}(H)$, then we update $tab_G[R, R']$ from one entry $tab_H[R, S']$ for some fixed S' computable in constant time. The used join operation runs in time $2^{k-1} \cdot k^{O(1)}$ (from Fact 3.3). Thus, tab_G is computable in time $(d+1)^{2k} \cdot k^{O(1)}$.
- Now, if $G = ren_{i\to j}(H)$, then we update $tab_G[R, R']$ from at most $|\mathcal{S}| = (d+1)^2$ tables from tab_H, each identified in constant time from (R, R'). Since each table of tab_H contains at most 2^{k-1} entries, computing the call at the function reduce take $(d+1)^2 \cdot 2^{\omega\cdot k} \cdot k^{O(1)}$. Thus, we can compute tab_G in time $(d+1)^{2k+2} \cdot 2^{\omega\cdot k} \cdot k^{O(1)}$.
- If $G = G_a \oplus G_b$, then the bottleneck is when $R' \neq \{0\}^k$. Indeed, if $R' = \{0\}^k$, then $tab_G[R, R']$ can be computed in time $O(2^{\omega\cdot k})$ since $tab_G[R, R']$ is computed from two tables, each containing at most 2^{k-1} entries. Let $R' \neq \{0\}^k$. By Theorem 5.4, we can compute the tables $tab_G[R, R']$, for every $R \in \{1, \ldots, d\}^k$ in time

$$\sum_{R\in\{0,\ldots,d\}^k} \left(\sum_{\substack{(A,B) \text{ is} \\ R\text{-comptatible}}} |\mathsf{join}(tab_{G_a}[A, R'], tab_{G_b}[B, R'])| \cdot 2^{(\omega-1)\cdot k} \cdot k^{O(1)} \right).$$

  Observe that for all $A, B \in \{0, \ldots, d\}^k$:
  (1) There is only one $R \in \{0, \ldots, d\}^k$ such that (A, B) is R-compatible. This follows from the definition of R-compatibility. Hence, there are at most $(d+1)^{2k}$ tuples (A, B, R) such that (A, B) is R-compatible.
  (2) The size of $\mathsf{join}(tab_{G_a}[A, R'], tab_{G_b}[B, R'])$ is bounded by $2^{2(k-1)}$ and this set can be computed in time $2^{2(k-1)} \cdot k^{O(1)}$.

Since $2^{2(k-1)} \cdot 2^{(\omega-1)\cdot k} \leq 2^{(\omega+1)\cdot k}$, we conclude from Observations (1)-(2) that we can compute the tables $tab_G[R, R']$, for every $R \in \{1, \ldots, d\}^k$, in time $(d+1)^{2k} \cdot 2^{(\omega+1)\cdot k} \cdot k^{O(1)}$.

Hence, we can update tab_G in time $(d+1)^{3k} \cdot 2^{(\omega+1)\cdot k} \cdot k^{O(1)}$.

Hence, in the worst case, the tables of tab_G takes $(d+1)^{3k} \cdot 2^{(\omega+1)\cdot k} \cdot k^{O(1)}$ time to be computed. Because the size of a k-expression is $O(n \cdot k^2)$, we can conclude that a maximum (or minimum) (σ, ρ)-dominating set can be computed in the given time. □

As a consequence of Theorem 5.13, we have the following corollary.

Corollary 5.14. *There is an algorithm that, given an n-vertex graph G, a subset $K \subseteq V(G)$ and an irredundant k-expression of G, computes a minimum node-weighted Steiner tree for (G, K) in time $2^{(\omega+4)\cdot k} \cdot k^{O(1)} \cdot n$.*

Proof. We can assume w.l.o.g. that $|K| \geq 2$. We can reduce the problem Node-weighted Steiner Tree to a variant of (σ, ρ)-Dominating Set where $\sigma = \mathbb{N}^+$ and $\rho = \mathbb{N}$. This variant requires K to be included in the (σ, ρ)-dominating set. We can add this constraint, by modifying how we compute the table tab_G, when $G = \mathbf{1}(x)$ and $x \in K$. For $(r_1), (r'_1) \in \{0, 1\}$, we let

$$tab_G[R, R'] := \begin{cases} \{(\{\{1\}\}, \mathsf{w}(x))\} & \text{if } r_1 = 1 \text{ and } r'_1 = 1, \\ \emptyset & \text{otherwise.} \end{cases}$$

It is straightforward to check that this modification implements this constraint and our algorithm with this modification computes a minimum node-weighted Steiner tree. The running time follows from the running time of Theorem 5.13 with $d = 1$. □

More modifications are needed in order to compute a maximum (or minimum) connected co-(σ, ρ)-dominating set.

Corollary 5.15. *There is an algorithm that, given an n-vertex graph G and an irredundant k-expression of G, computes a maximum (or minimum) co-(σ, ρ)-dominating set in time $(d+2)^{3k} \cdot 2^{(\omega+1)\cdot k} \cdot k^{O(1)} \cdot n$.*

Proof. First, we need to modify the definition of the tables $\mathcal{D}_G$. Let H be a k-labeled graph, $R, R' \in \{0, \ldots, d\}^k$, and $\overline{R}, \overline{R}' \in \{0, 1\}^k$. The entries of $\mathcal{D}_H[R, R', \overline{R}, \overline{R}']$ are all the weighted partitions $(p, w) \in \Pi(\mathsf{active}(\overline{R}, \overline{R}')) \times \mathbb{N}$ such that there exists a set $X \subseteq V(G)$ so that $w = \mathsf{w}(X)$ and

(1) $r^d_H(V(H) \setminus X) = R$ and $r^1_H(X) = \overline{R}$,
(2) $(V(H) \setminus X) \cup V^+(R')$ (σ, ρ)-dominates $V(H)$ in $\mathbf{CG}(H, R')$,
(3) if $\mathsf{active}(\overline{R}, \overline{R}') = \emptyset$, then $H[X]$ is connected, otherwise every connected component of $H[X]$ intersects $lab^{-1}_H(\mathsf{active}(\overline{R}, \overline{R}'))$,
(4) $p = \mathsf{active}(\overline{R}, \overline{R}')/\sim$ where $i \sim j$ if and only if an i-vertex is connected to a j-vertex in $\mathbf{CG}(H, \overline{R}')[X \cup V^+(\overline{R}')]$.

As a solution is a set X such that $V(G) \setminus X$ is a (σ, ρ)-dominating set and $G[X]$ is a connected graph, we need information about $X \cap V(H)$ and $(V(H) \setminus X)$. Intuitively, R, R' are the information we need to guarantee the (σ, ρ)-domination, and $\overline{R}, \overline{R}'$ are the information we need to guarantee the connectedness. In particular, $\overline{R}$ specifies which label classes are intersected and $\overline{R}'$ tells which label classes are expected to have at least one neighbor in the future.

These modifications imply in particular to change the notion of R-compatibility. For each $t \in \{a, b, c\}$, let $R_t = (r^t_1, \ldots, r^t_k) \in \{0, \ldots, d\}^k$, and $\overline{R_t} = (\overline{r}^t_1, \ldots, \overline{r}^t_k) \in$

$\{0,1\}^k$, we say that $(R_a, \overline{R_a}, R_b, \overline{R_b})$ is $(R_c, \overline{R_c})$-compatible if for all $i \in [k]$, we have $r_i^c = \min(d, r_i^a + r_i^b)$ and $\overline{r}_i^c = \min(1, \overline{r}_i^a + \overline{r}_i^b)$.

It is now an exercise to modify the algorithm of Theorem 5.13 in order to update the tables tab_H through the different clique-width operations. The weight of an optimum solution corresponds to the optimum over all $R \in \{0,\ldots,d\}^k, \overline{R} \in \{0,1\}^k$ of $\mathbf{opt}\{w : (\emptyset, w) \in tab_G[R, \{0\}^k, \overline{R}, \{0\}^k]\}$, since $tab_G[R, \{0\}^k, \overline{R}, \{0\}^k]$ represents $\mathcal{D}_G[R, \{0\}^k, \overline{R}, \{0\}^k]$.

Let us discuss the time complexity now. Let H be a k-labeled graph that is used in the k-expression of G. First, observe that we do not need to compute $tab_H[R, R', \overline{R}, \overline{R}']$ for all $R, R' \in \{0,\ldots,d\}^k$, and $\overline{R}, \overline{R}' \in \{0,1\}^k$. Indeed, for all $X \subseteq V(H)$ and $i \in [k]$, if we have $r^d_{i,H}(V(H) \setminus X) < d$, then

$$r^1_{i,H}(X) = \begin{cases} 0 & \text{if } |lab_H^{-1}(i)| = r^d_{i,H}(V(H) \setminus X), \\ 1 & \text{otherwise.} \end{cases}$$

Hence, we deduce that there are at most $(d+2)^k$ pairs $(R, \overline{R}) \in \{0,\ldots,d\}^k \times \{0,1\}^k$ such that $R = r^d_H(V(H) \setminus X)$ and $\overline{R} = r^1_H(X)$. Indeed, whenever $r^d_{i,H}(V(H) \setminus X) < d$, there is only one possible value for $r^1_{i,H}(X)$, and when $r^d_{i,H}(V(H) \setminus X) = d$, there are at most 2 possible values for $r^1_{i,H}(X)$. With the same arguments used to prove the running time of Theorem 5.13, one easily deduces that there are at most $(d+2)^{2k}$ tuples $(R_a, \overline{R_a}, R_b, \overline{R_b}, R_c, \overline{R_c})$ such that $(R_a, \overline{R_a}, R_b, \overline{R_b})$ is $(R_c, \overline{R}_c)$-compatible.

Moreover, it is sufficient to consider $(d+2)^k$ pairs $(R', \overline{R}') \in \{0,\ldots,d\}^k \times \{0,1\}^k$ when we update tab_H. For every $i \in [k]$, we let $c^i_H := |N_G(lab_H^{-1}(i)) \setminus N_H(lab_H^{-1}(i))|$. Notice that for every vertex $v \in lab_H^{-1}(i)$, we have $|N_G(v)| = |N_H(v)| + c^i_H$. Informally, we cannot expect more than c^i_H neighbors in the future for the i-vertices of H. Hence, it is enough to consider the pairs $(R', \overline{R}') \in \{0,\ldots,d\}^k \times \{0,1\}^k$, with $R' = (r'_1,\ldots,r'_k)$ and $\overline{R}' = (\overline{r}'_1,\ldots,\overline{r}'_k)$, such that for all $i \in [k]$, if $r'_i < d$, then

$$\overline{r}'_i = \begin{cases} 0 & \text{if } r'_i \geq c^H_i, \\ 1 & \text{otherwise.} \end{cases}$$

That is, for every $i \in [k]$, if $r'_i < d$, then there is one possible value for $\overline{r}'_i$ because if we expect $r'_i < d$ neighbors for the i-vertices in $V(H) \setminus X$, then we must expect $\min(0, c^i_H - r'_i)$ neighbors for the i-vertices in X. If $r'_i = d$, then there are no restrictions on the value of $\overline{r}'_i$. Thus, the pairs $(r'_i, \overline{r}'_i)$ can take up to $(d+2)$ values. We conclude that there are at most $(d+2)^k$ pairs $(R', \overline{R}') \in \{0,\ldots,d\}^k \times \{0,1\}^k$ worth to looking at. With these observations and the arguments used in the running time proof of Theorem 5.13, we conclude that we can compute a maximum (or minimum) co-(σ,ρ)-dominating set in the given time. □

6. Concluding Remarks

We combine the techniques introduced in [4] and the rank-based approach from [2] to obtain $2^{O(k)} \cdot n$ time algorithms for several connectivity constraints problems such as Connected Dominating Set, Node-Weighted Steiner Tree, Feedback Vertex Set, Connected Vertex Cover, etc. While we did not consider connectivity constraints on locally vertex partitioning problems [4], it seems clear that we can adapt our algorithms from the paper to consider such connectivity constraints: if the solution is $\{D_1,\ldots,D_q\}$, each block D_i is connected or a proper subset of the blocks form a connected graph. We did not consider counting versions and it would be interesting to know if we can adapt the approach in [2] based on the determinant to the clique-width.

In [12] Fomin et al. use fast computation of representative sets in matroids to provide deterministic $2^{O(k)} \cdot n^{O(1)}$ time algorithms parameterized by tree-width for many connectivity problems. Is this approach also generalizable to clique-width ?

The main drawback of clique-width is that, for fixed k, there is no known FPT polynomial time algorithm that produces a k-expression, that even approximates within a constant factor the clique-width of the given graph. We can avoid this major open question, by using other parameters as powerful as clique-width. Oum and Seymour introduced the notion of *rank-width* and its associated *rank-decomposition* [19] such that $\mathsf{rank\text{-}width}(G) \leq \mathsf{clique\text{-}width}(G) \leq 2^{\mathsf{rank\text{-}width}(G)+1} - 1$. Furthermore there is a $2^{O(k)} \cdot n^3$ time algorithm for computing it [15, 17]. If we use our approach with a rank-decomposition, the number of twin-classes is bounded by 2^k, and so we will get a $2^{2^{O(k)}} \cdot n^{O(1)}$ time algorithm. We can circle it and obtain $2^{O(k^2)} \cdot n^{O(1)}$ as done for example in [13] for Feedback Vertex Set by using Myhill-Nerode congruences.

In [1], the authors adapted the rank-based approach to the notion of *d-neighbor equivalence* from [4]. They deduce, in particular, $2^{O(k)} \cdot n^{O(1)}$, $2^{O(k \cdot \log(k))} \cdot n^{O(1)}$, and $2^{O(k^2)} \cdot n^{O(1)}$ time algorithms parameterized respectively by clique-width, $\mathbb{Q}$-rank-width (a variant of rank-width [18]), and rank-width for all the problems considered in this paper. The results in [1] generalize, simplify, and unify several results including [2, 13] and those from this paper. However, for each considered problem in this article, the algorithms parameterized by clique-width from [1] have a worse running time than the algorithms from this paper.

We recall that, it is still open whether we can obtain a $2^{O(k)} \cdot n^{O(1)}$ time algorithm parameterized by rank-with (or $\mathbb{Q}$-rank-width) to solve the problems considered in this paper, or more basic problems such as Independent Set or Dominating Set.

References

[1] Benjamin Bergougnoux and Mamadou Moustapha Kanté. Rank based approach on graphs with structured neighborhood. *CoRR*, abs/1805.11275, 2018.

[2] Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inform. and Comput.*, 243:86–111, 2015.

[3] Binh-Minh Bui-Xuan, Ondřej Suchý, Jan Arne Telle, and Martin Vatshelle. Feedback vertex set on graphs of low clique-width. *European J. Combin.*, 34(3):666–679, 2013.

[4] Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theoret. Comput. Sci.*, 511:66–76, 2013.

[5] Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic*, volume 138 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2012. A language-theoretic approach, With a foreword by Maurice Nivat.

[6] Bruno Courcelle, Joost Engelfriet, and Grzegorz Rozenberg. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2):218–270, 1993.

[7] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

[8] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time (extended abstract). In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science—FOCS 2011*, pages 150–159. IEEE Computer Soc., Los Alamitos, CA, 2011.

[9] Reinhard Diestel. *Graph Theory*. Number 173 in Graduate Texts in Mathematics. Springer, third edition, 2005.

[10] Rodney G. Downey and Michael R. Fellows. *Fundamentals of parameterized complexity*. Texts in Computer Science. Springer, London, 2013.

[11] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM J. Comput.*, 43(5):1541–1563, 2014.

[12] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016.

[13] Robert Ganian and Petr Hliněný. On parse trees and Myhill-Nerode-type tools for handling graphs of bounded rank-width. *Discrete Appl. Math.*, 158(7):851–867, 2010.

[14] Robert Ganian, Petr Hliněný, and Jan Obdržálek. Clique-width: when hard does not mean impossible. In *28th International Symposium on Theoretical Aspects of Computer Science*, volume 9 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 404–415. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2011.

[15] Petr Hliněný and Sang-il Oum. Finding branch-decompositions and rank-decompositions. *SIAM J. Comput.*, 38(3):1012–1032, 2008.

[16] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 653–663, 1998.

[17] Sang-Il Oum. Approximating rank-width and clique-width quickly. *ACM Trans. Algorithms*, 5(1):Art. 10, 20, 2009.

[18] Sang-il Oum, Sigve Hortemo Sæther, and Martin Vatshelle. Faster algorithms for vertex partitioning problems parameterized by clique-width. *Theoret. Comput. Sci.*, 535:16–24, 2014.

[19] Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.

[20] Neil Robertson and Paul D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[21] Jan Arne Telle and Andrzej Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discrete Math.*, 10(4):529–550, 1997.

Université Clermont Auvergne, LIMOS, CNRS, Aubière, France.
Email address: {mamadou.kante,benjamin.bergougnoux}@uca.fr