Rank Based Approach on Graphs with Structured Neighborhood

Benjamin Bergougnoux, Mamadou Moustapha Kanté, Mamadou Moustapha Kanté

RANK BASED APPROACH ON GRAPHS WITH STRUCTURED NEIGHBORHOOD

BENJAMIN BERGOUGNOUX AND MAMADOU MOUSTAPHA KANTÉ

ABSTRACT. In this paper, we combine the rank-based approach with the notion of d-neighbor equivalence to obtain efficient algorithms for several connectivity problems such as CONNECTED DOMINATING SET, NODE WEIGHTED STEINER TREE, MAXIMUM INDUCED TREE, LONGEST INDUCED PATH, and FEEDBACK VERTEX SET. For all these problems, we obtain $2^{O(k)} \cdot n^{O(1)}$, $2^{O(k \log(k))} \cdot n^{O(1)}$, $2^{O(k^2)} \cdot n^{O(1)}$ and $n^{O(k)}$ time algorithms parameterized respectively by clique-width, $\mathbb{Q}$-rank-width, rank-width and maximum induced matching width. Our approach simplifies and unifies the known algorithms for each of the parameters and match asymptotically also the best time complexity for VERTEX COVER and DOMINATING SET.

1. INTRODUCTION

Connectivity problems such as CONNECTED DOMINATING SET, FEEDBACK VERTEX SET or HAMILTONIAN CYCLE were for a long time a curiosity in FPT world as they admit trivial $k^{O(k)} \cdot n^{O(1)}$ time algorithms parameterized by tree-width, but no lower-bounds were known. Indeed, for good reason, Cygan et al. presented in [9] Monte Carlo algorithms for a wide range of connectivity problems running in time $2^{O(k)} \cdot n^{O(1)}$ with k the tree-width of the input graph. Later, Bodlaender et al. proposed in [4] a general toolkit called *rank-based approach* to design deterministic $2^{O(k)} \cdot n^{O(1)}$ time algorithms, with k the tree-width of the input graph, to solve a wider range of connectivity problems. The idea is to boost up basic dynamic programming algorithms, by reducing at each step, the size of the set of partial solutions. For doing so, they associate each set of partial solutions with a binary matrix and then show that this matrix admits a basis of size $2^{O(k)}$ which *represents* the set of optimal solutions and is computable by a greedy algorithm. Informally, a set $\mathcal{S}'$ represents a set of partial solutions $\mathcal{S}$, if whenever there exits $S \in \mathcal{S}$ that leads to an optimal solution, there exists $S' \in \mathcal{S}'$ leading to an optimal solution.

Nevertheless, despite the broad interest on tree-width, only sparse graphs can have bounded tree-width. But, many NP-hard problems are tractable on dense graph classes. Most of the time, this tractability can be explained by the ability of these graphs to be recursively decomposed along vertex bipartitions (A, B) where the adjacency between A and B is simple to describe, *i.e.*, they have a *structured neighborhood*. A lot of graph parameters have been defined to characterize this ability, the most remarkable ones are certainly clique-width [8], rank-width [20], and maximum induced matching width (called mim-width) [25].

Introduced by Courcelle and Olariu [8], the modeling power of clique-width is strictly stronger than the modeling power of tree-width. In other words, if a graph class has bounded tree-width, then it has bounded clique-width [8], but the converse is false as cliques have clique-width at most 2 and unbounded tree-width. While rank-width has the same modeling power as clique-width, mim-width has the strongest one among all these complexity measures and is even bounded on interval graphs [1]. Despite their generality, a lot of NP-hard problems admit polynomial time algorithms when one of these parameters is fixed. But, dealing with these parameters is known to be harder than manipulating tree-width.

We obtain most of these parameters through the notion of *layout*. A layout of a graph G is a tree T whose leaves are in bijection with the vertices of G. Every edge e of the layout is

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

Key words and phrases. connectivity problem, feedback vertex set, d-neighbor equivalence, σ, ρ-domination, clique-width, rank-width, mim-width.

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

associated with a vertex bipartition of G through the two connected components obtained by the removal of e. Given a symmetric function $f : 2^{V(G)} \times 2^{V(G)} \to \mathbb{N}$, one can associate with each layout T a measure, called usually f-width, defined as the maximum $f(X, \overline{X})$ over all the vertex bipartitions $(X, \overline{X})$ associated with the edges of T. For instance, rank-width is defined from the function $f(X, \overline{X})$ which corresponds to the rank over $GF(2)$ of the adjacency matrix between the vertex sets X and $\overline{X}$; if we take the rank over $\mathbb{Q}$, we obtain a useful variant of rank-width introduced in [21], called $\mathbb{Q}$-rank-with. For mim-width, $f(X, \overline{X})$ is the maximum size of an induced matching in a bipartite graph associated with $(X, \overline{X})$.

All the algorithms parameterized by clique-width (or mim-width) require that a layout of bounded clique-width (mim-width) is given as input. Indeed, it is not known whether the clique-width (respectively mim-width) of a graph can be approximated within a constant factor in time $f(k) \cdot n^{O(1)}$ (resp. $n^{f(k)}$) for some function f. The same assumption is not necessary for rank-width and $\mathbb{Q}$-rank-width thanks to the following result of Oum and Seymour [22].

Theorem 1.1 (Oum and Seymour [22])**.** *There is a $2^{3k} \cdot n^{O(1)}$ time algorithm that, given a graph G as input and $k \in \mathbb{N}$, either outputs a layout for G of ($\mathbb{Q}$-)rank-width at most $3k + 1$ or confirms that the ($\mathbb{Q}$-)rank-width of G is more than k.*

Unlike tree-width, algorithms parameterized by clique-width, rank-width and mim-width for connectivity problems, were not investigated, except for some special cases such as Feedback Vertex Set which is proved to admit a $2^{O(k)} \cdot n^{O(1)}$ time algorithm parameterized by clique-width in [2], a $2^{O(k^2)} \cdot n^{O(1)}$ time algorithm parameterized by rank-width [12], and an $n^{O(k)}$ time algorithm parameterized by mim-width [17].

One successful way to work with these different parameters is through the notion of *d-neighbor equivalence* introduced in [7]. A *module* of a graph is a subset of vertices $A \subseteq V(G)$ with the same neighbors in $V(G) \setminus A$. The notion of d-neighbor equivalence generalizes the module notion. Formally, given $A \subseteq V(G)$ and $d \in \mathbb{N} \setminus \{0\}$, two sets $X, Y \subseteq A$ are d-neighbor equivalent *w.r.t.* A if for all $v \in V(G) \setminus A$, we have $\min(d, |N(v) \cap X|) = \min(d, |N(v) \cap Y|)$, where $N(v)$ is the set of neighbors of v in G. One easily checks that it is an equivalence relation, and if $d = 1$, then it measures the number of subsets of A with different neighborhoods in $V(G) \setminus A$. The d-neighbor equivalence relation was the key in the design of efficient algorithms to solve some well-studied and well-known difficult problems such as Dominating Set [7, 14, 21] or Feedback Vertex Set [2].

In this paper, we use a parameter related to the d-neighbor equivalence relation that we call *d-neighbor-width*. The d-neighbor-width of a layout is the maximum number of equivalence classes over the vertex bipartitions of the layout *w.r.t.* to the d-neighbor equivalence relation. It is worth noticing that the boolean-width of a layout introduced in [6] corresponds to the binary logarithm of the 1-neighbor-width.

Our Contributions and approach. One of our main contribution is the modification of the rank-based approach to fit with the d-neighbor-width (presented in Section 3) in order to get fast polynomial time algorithms for connectivity constraints problems. The resulting framework simplifies and generalizes the original rank-based approach from [4]. The rank-based approach is a general toolkit used for connectivity problems in order to reduce the size of the set of partial solutions at every step of a dynamic programming algorithm. In our case, the steps of our algorithms are associated with the vertex bipartitions $(S, \overline{S})$ induced by the edges of a layout and the sets of partial solutions are collections of subsets of S. Let us explain the idea of our framework with respect to a graph G and a vertex bipartition $(S, \overline{S})$ of G induced by an edge of a layout of G. One of the main differences with [4] is that our notion of representativity is defined with respect to a 1-neighbor equivalence class C of $\overline{S}$. More precisely, a set of partial solutions $\mathcal{A}'$ *C-represents* a set $\mathcal{A}$, if for all sets $Y \in C$, the best solution[1] we get by the union of Y and a set in $\mathcal{A}$ has the same weight as the best solution we can get from the union of Y and a set in $\mathcal{A}'$. The main tool of our framework is a function that, given a set of partial solutions

[1]The solution of optimum weight (maximum or minimum) that induces a connected graph.

$\mathcal{A}$ of S and a 1-neighbor equivalence class C of $\overline{S}$, outputs a subset of $\mathcal{A}$ that C-represents $\mathcal{A}$ and whose size is polynomial in the number of 1-neighbor equivalence classes of S. As in [4], this function is obtained by computing a basis of optimum weight of the row space of a matrix associated with $\mathcal{A}$. In order to compute an optimum solution from the partial solutions of $G[S]$, it is enough to keep, for each 1-neighbor equivalence class C of $\overline{S}$, a set that C-represents the set of all the partial solutions of $G[S]$. Consequently, for each vertex bipartition induced by an edge of a layout, it is enough to keep a set of partial solutions whose size is polynomial in the number of 1-neighbor equivalence classes of S and $\overline{S}$.

In Section 4, we apply our framework to connectivity problems with locally checkable properties, such as Connected Dominating Set, Connected Vertex Cover or Node Weighted Steiner Tree. Each of this problem is a connected variant of a problem in the family of problems called (σ, ρ)-Dominating Set problems. This family of problems was introduced in [24] and studied in graphs through the notion of d-neighbor equivalence in [7, 14, 21]. Given two non-empty finite or co-finite subsets of $\mathbb{N}$ and a graph G, a (σ, ρ)-*dominating set* of G is a subset D of $V(G)$ such that, for each vertex $x \in V(G)$, the number of neighbors of x in X is in σ if $x \in X$ and in ρ otherwise. We provide an algorithm that, given an n-vertex graph G and a layout of G, finds an optimum (minimum or maximum) (σ, ρ)-dominating set which induces a connected graph. The running time of this algorithm is polynomial in n and the d-neighbor-width of the given layout, with d a constant depending on σ and ρ. For doing so, we use our framework and the algorithm from [7] that finds an optimum (σ, ρ)-dominating set. Let us explain how we modify this algorithm. Let G be a graph and $(S, \overline{S})$ a vertex bipartition of G induced by an edge of a layout of G. In order to find an optimum (σ, ρ)-dominating set, Bui-Xuan et al. [7] proved that it is enough to keep a partial solution X of optimum weight "coherent" with (R, R'), for each pair (R, R') where R (resp. R') is a d-neighbor equivalence class of S (resp. $\overline{S}$). We say that a set $X \subseteq S$ is coherent with (R, R') if X belongs to R and $X \cup Y$ (σ, ρ)-dominates S in the graph G for every $Y \in R'$. To solve the connected variant of the same problem, we prove that it is enough to keep, for each such pair (R, R'), a set that R'-represents the set of all the partial solutions that are coherent with (R, R'). We, consequently, obtain, efficient algorithms to solve any Connected (σ, ρ)-Dominating Set problem, with parameters clique-width, ($\mathbb{Q}$)-rank-width, and mim-width. The running times of these algorithms are given in Table 1. Up to a constant in the exponent, these running times match those known for basic problems such as Vertex Cover and Dominating Set [7, 21]. Moreover, our algorithms simplify the known ones and highlight the importance of the d-neighbor equivalence relation for these parameters.

Clique-width	Rank-width	$\mathbb{Q}$-rank-width	Mim-width
$2^{O(k)} \cdot n^{O(1)}$	$2^{O(k^2)} \cdot n^{O(1)}$	$2^{O(k \log(k))} \cdot n^{O(1)}$	$n^{O(k)}$

Table 1. Running times of our algorithms for the different parameters, where n is the number of vertices of the given graph.

In Section 5, we integrate the notion of acyclicity to our framework. As a result, we obtain efficient algorithms for any acyclic variant of a Connected (σ, ρ)-Dominating Set problem whose parameters and running times are exactly the same as those described in Table 1. Both Maximum Induced Tree (MIT for short) and Longest Induced Path are the acyclic variant of a Connected (σ, ρ)-Dominating Set problem. We also obtain the same algorithmic results for any acyclic variant of a (σ, ρ)-Dominating Set problem by polynomially reducing this latter problem to its connected variant. Consequently, we can use the algorithm for MIT to solve the Feedback Vertex Set problem. Let us explain the idea of our algorithm for MIT with respect to an n-vertex graph G and a vertex bipartition $(S, \overline{S})$ of $V(G)$. In order to solve MIT, we need a new notion of representativity that takes into account the acyclicity. We define this new notion with respect to a 2-neighbor equivalence class C of $\overline{S}$. We say that a set $\mathcal{A}'$ C-*ac-represents* a set $\mathcal{A}$, if for every $Y \in C$, the maximum weight of a set $X \in \mathcal{A}$ such that $G[X \cup Y]$ is a tree equals the maximum weight of a set $X' \in \mathcal{A}'$ such that $G[X' \cup Y]$ is a tree. Similarly to Section 3, we provide a function that, given a set of partial solutions $\mathcal{A}$ and a 2-neighbor equivalence class C of $\overline{S}$, outputs a small subset $\mathcal{A}'$ of $\mathcal{A}$ that C-ac-represents $\mathcal{A}$. However, we were not able to upper bound the size of $\mathcal{A}'$ by a polynomial in the number of 2-neighbor equivalence classes and n. Instead, we prove that, for clique-width, rank-width, $\mathbb{Q}$-rank-width, and mim-width, the size of $\mathcal{A}'$ can be upper bounded by, respectively, $2^{O(k)} \cdot n$, $2^{O(k^2)} \cdot n$, $2^{O(k\log(k))} \cdot n$, and $n^{O(k)}$. The key to compute $\mathcal{A}'$ is to decompose $\mathcal{A}$ into a small number of sets $\mathcal{A}_1 \dots, \mathcal{A}_\ell$, said C-consistent, where the notion of C-ac-representativity matches the notion of C-representativity. More precisely, any C-representative set of a C-consistent set is also a C-ac-representative set. To compute a C-ac-representative set of $\mathcal{A}$ it is then enough to compute a C-representative set for each C-consistent set in the decomposition of $\mathcal{A}$. The union of these C-representative sets is a C-ac-representative set of $\mathcal{A}$. Besides the notion of representativity, the algorithm for MIT is very similar to the one for finding a connected (σ,ρ)-dominating set.

Observe that we can not use the same trick as in [4] to ensure the acyclicity, that is counting the number of edges induced by the partial solutions. Indeed, we would need to differentiate at least n^k partial solutions (for any parameter k considered in Table 1) in order to update this information. We give more explanation on this statement at the beginning of Section 5.

Relation to previous works. This work follows a previous paper by the authors [2] where they have generalized the rank-based approach to clique-width in order to solve any Connected (σ,ρ)-Dominating Set problem and Feedback Vertex Set in time $2^{O(k)} \cdot n$ given a clique-width k-expression. The algorithms parameterized by clique-width from [2] have a better running time than the algorithms from this paper. However, our approach generalizes and simplifies the results from [2], in particular for Feedback Vertex Set where the use of weighted partitions to represent the partial solutions implies to take care of many technical details concerning the acyclicity.

Moreover, the results we obtain for Feedback Vertex Set simplifies also the $2^{O(k^2)} \cdot n^{O(1)}$ time algorithm parameterized by rank-width from [12], and the $n^{O(k)}$ time algorithm parameterized by mim-width from [17].

Our algorithm for Longest Induced Path generalizes and simplifies the $n^{O(k)}$-time algorithm parameterized by mim-width from [16].

Notice that our results generalize also the original rank-based approach of [4] and can be used to obtain $2^{O(k)} \cdot n^{O(1)}$ time algorithms parameterized by tree-width for connectivity problems. This is due to the fact that if S is a vertex separator of size k, then the number of different neighborhoods in S is upper-bounded by 2^k. Contrary to [4], we do not use weighted partitions to represent the partial solutions. Consequently, the definitions of the dynamic programming tables and the computational steps of our algorithms are simpler than those in [4].

It is worth noticing that the approach used in [9] called "cut and count" can also be generalized to the d-neighbor-width for any Connected (σ,ρ)-dominating set problem with more or less the same arguments used in this paper.

However, it is not clear how to generalize the "cut and count" approach to solve the acyclic variants of the Connected (σ,ρ)-dominating set problems, with parameters considered in this paper.

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. We often write x to denote the singleton set $\{x\}$. We denote by $\mathbb{N}$ the set of non-negative integers and by $\mathbb{N}^+$ the set $\mathbb{N} \setminus \{0\}$. We let $\min(\emptyset) := +\infty$ and $\max(\emptyset) := -\infty$. For two sets A and B, we define the merging of A and B, denoted by $A \bigotimes B$, as

$$A \bigotimes B := \begin{cases} \emptyset & \text{if } A = \emptyset \text{ or } B = \emptyset, \\ \{X \cup Y \,:\, X \in A \text{ and } Y \in B\} & \text{otherwise.} \end{cases}$$

Let V be a finite set. A set function $f : 2^V \to \mathbb{N}$ is *symmetric* if for all $S \subseteq V$, we have $f(S) = f(V \setminus S)$.

Graphs. Our graph terminology is standard, and we refer to [10]. The vertex set of a graph G is denoted by $V(G)$ and its edge set by $E(G)$. For every vertex set $X \subseteq V(G)$, when the underlying graph is clear from context, we denote by $\overline{X}$, the set $V(G) \setminus X$. An edge between two vertices x and y is denoted by xy or yx. The set of vertices that is adjacent to x is denoted by $N_G(x)$. For a set $U \subseteq V(G)$, we define $N_G(U) := \bigcup_{x \in U} N_G(x)$. If the underlying graph is clear, then we may remove G from the subscript.

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 $G[V(G) \setminus X]$. For $X, Y \subseteq V(G)$, we denote by $G[X, Y]$ the bipartite graph with vertex set $X \cup Y$ and edge set $\{xy \in E(G) \,:\, x \in X \text{ and } y \in Y\}$. Moreover, we denote by $M_{X,Y}$ the adjacency matrix between X and Y, *i.e.*, the (X, Y)-matrix such that $M_{X,Y}[x, y] = 1$ if $y \in N(x)$ and 0 otherwise.

For a graph G, we denote by $\mathsf{cc}(G)$ the partition $\{V(C) \,:\, C \text{ is a connected component of } G\}$. Let $X \subseteq V(G)$. A *consistent cut* of X is an ordered bipartition (X_1, X_2) of X such that $N(X_1) \cap X_2 = \emptyset$. We denote by $\mathsf{ccut}(X)$ the set of all consistent cuts of X. In our proofs, we use the following facts.

Fact 2.1. *Let $X \subseteq V(G)$. For every $C \in \mathsf{cc}(G[X])$ and every $(X_1, X_2) \in \mathsf{ccut}(X)$, we have either $C \subseteq X_1$ or $C \subseteq X_2$.*

We deduce from the above fact that $|\mathsf{ccut}(X)| = 2^{|\mathsf{cc}(G[X])|}$.

Fact 2.2. *Let X and Y be two disjoint subsets of $V(G)$. We have $(W_1, W_2) \in \mathsf{ccut}(X \cup Y)$ if and only if the following conditions are satisfied*

(1) $(W_1 \cap X, W_2 \cap X) \in \mathsf{ccut}(X)$,

(2) $(W_1 \cap Y, W_2 \cap Y) \in \mathsf{ccut}(Y)$, and

(3) $N(W_1 \cap X) \cap (W_2 \cap Y) = \emptyset$ and $N(W_2 \cap X) \cap (W_1 \cap Y) = \emptyset$.

d-neighbor-equivalence. Let G be a graph. The following definition is from [7]. Let $A \subseteq V(G)$ and $d \in \mathbb{N}^+$. Two subsets X and Y of A are *d-neighbor equivalent* w.r.t. A, denoted by $X \equiv^d_A Y$, if $\min(d, |X \cap N(u)|) = \min(d, |Y \cap N(u)|)$ for all $u \in \overline{A}$. It is not hard to check that $\equiv^d_A$ is an equivalence relation. See Figure 1 for an example of 2-neighbor equivalent sets.

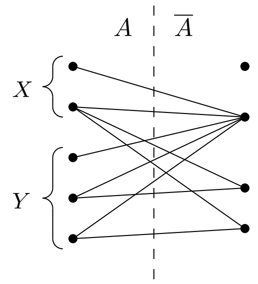

FIGURE 1. We have $X \equiv^2_A Y$, but it is not the case that $X \equiv^3_A Y$.

For all $d \in \mathbb{N}^+$, we let $\mathsf{nec}_d : 2^{V(G)} \to \mathbb{N}$ where for all $A \subseteq V(G)$, $\mathsf{nec}_d(A)$ is the number of equivalence classes of $\equiv^d_A$. Notice that while nec_1 is a symmetric function [18, Theorem 1.2.3], nec_d is not necessarily symmetric for $d \geq 2$. For example, if a vertex x of G has c neighbors, then for every $d \in \mathbb{N}^+$, we have $\mathsf{nec}_d(\{x\}) = 2$ and $\mathsf{nec}_d(\overline{\{x\}}) = 1 + \min(d, c)$. It is worth noticing that, for every $d \in \mathbb{N}^+$, $\mathsf{nec}_d(A)$ and $\mathsf{nec}_d(\overline{A})$ are at most $\mathsf{nec}_1(A)^{d \log_2(\mathsf{nec}_1(A))}$, for each $A \subseteq V(G)$ [7].

The following fact follows directly from the definition of the d-neighbor equivalence relation. We use it several times in our proofs.

Fact 2.3. *Let $A, B \subseteq V(G)$ such that $A \subseteq B$, and let $d \in \mathbb{N}^+$. For all $X, Y \subseteq A$, if $X \equiv_A^d Y$, then $X \equiv_B^d Y$.*

In order to manipulate the equivalence classes of $\equiv_A^d$, one needs to compute a representative for each equivalence class in polynomial time. This is achieved with the following notion of a representative. Let G be a graph with an arbitrary ordering of $V(G)$ and let $A \subseteq V(G)$. For each $X \subseteq A$, let us denote by $\mathsf{rep}_A^d(X)$ the lexicographically smallest set $R \subseteq A$ such that $|R|$ is minimized and $R \equiv_A^d X$. Moreover, we denote by $\mathcal{R}_A^d$ the set $\{\mathsf{rep}_A^d(X) : X \subseteq A\}$. It is worth noticing that the empty set always belongs to $\mathcal{R}_A^d$, for all $A \subseteq V(G)$ and $d \in \mathbb{N}^+$. Moreover, we have $\mathcal{R}_{V(G)}^d = \mathcal{R}_\emptyset^d = \{\emptyset\}$ for all $d \in \mathbb{N}^+$. In order to compute $\mathcal{R}_A^d$, we use the following lemma.

Lemma 2.4 ([7])**.** *For every $A \subseteq V(G)$ and $d \in \mathbb{N}^+$, one can compute in time $O(\mathsf{nec}_d(A) \cdot \log(\mathsf{nec}_d(A)) \cdot |V(G)|^2)$, the sets $\mathcal{R}_A^d$ and a data structure, that given a set $X \subseteq A$, computes $\mathsf{rep}_A^d(X)$ in time $O(\log(\mathsf{nec}_d(A)) \cdot |A| \cdot |V(G)|)$.*

Rooted Layout. A *rooted binary tree* is a binary tree with a distinguished vertex called the *root*. Since we manipulate at the same time graphs and trees representing them, the vertices of trees will be called *nodes*.

A *rooted layout* of G is a pair $\mathcal{L} = (T, \delta)$ of a rooted binary tree T and a bijective function δ between $V(G)$ and the leaves of T. For each node x of T, let L_x be the set of all the leaves l of T such that the path from the root of T to l contains x. We denote by $V_x^{\mathcal{L}}$ the set of vertices that are in bijection with L_x, *i.e.*, $V_x^{\mathcal{L}} := \{v \in V(G) : \delta(v) \in L_x\}$. When $\mathcal{L}$ is clear from the context, we may remove $\mathcal{L}$ from the superscript.

All the structural parameters dealt with in this paper are special cases of the following one, the difference being in each case the used set function. Given a set function $f : 2^{V(G)} \to \mathbb{N}$ and a rooted layout $\mathcal{L} = (T, \delta)$, the f-width of a node x of T is $f(V_x^{\mathcal{L}})$ and the f-width of (T, δ), denoted by $f(T, \delta)$ (or $f(\mathcal{L})$), is $\max\{f(V_x^{\mathcal{L}}) : x \in V(T)\}$. Finally, the f-width of G is the minimum f-width over all rooted layouts of G.

d-neighbor-width. For every graph G and $d \in \mathbb{N}^+$, the *d-neighbor-width* is the parameter obtained through the symmetric function $\mathsf{s\text{-}nec}_d : 2^{V(G)} \to \mathbb{N}$ such that

$$\mathsf{s\text{-}nec}_d(A) = \max(\mathsf{nec}_d(A), \mathsf{nec}_d(\overline{A})).$$

Clique-width / Module-width. We won't define *clique-width*, but its equivalent measure *module-width* [23]. The module-width of a graph G is the mw-width where $\mathsf{mw}(A)$ is the cardinal of $\{N(v) \cap \overline{A} : v \in A\}$ for all $A \subseteq V(G)$. One also observes that $\mathsf{mw}(A)$ is the number of different rows in $M_{A,\overline{A}}$. The following theorem shows the link between module-width and clique-width.

Theorem 2.5 ([23, Theorem 6.6])**.** *For every n-vertex graph G, $\mathsf{mw}(G) \leq \mathsf{cw}(G) \leq 2\mathsf{mw}(G)$, where $\mathsf{cw}(G)$ denotes the clique-width of G. One can moreover translate, in time at most $O(n^2)$, a given decomposition into the other one with width at most the given bounds.*

($\mathbb{Q}$)-Rank-width. The rank-width and $\mathbb{Q}$-rank-width are, respectively, the rw-width and $\mathsf{rw}_{\mathbb{Q}}$-with where $\mathsf{rw}(A)$ (resp. $\mathsf{rw}_{\mathbb{Q}}(A)$) is the rank over $GF(2)$ (resp. $\mathbb{Q}$) of the matrix $M_{A,\overline{A}}$ for all $A \subseteq V(G)$.

Mim-width. The mim-width of a graph G is the mim-width of G where $\mathsf{mim}(A)$ is the size of a maximum induced matching of the graph $G[A, \overline{A}]$ for all $A \subseteq V(G)$.

It is worth noticing that Module-width is the only parameter associated with a set function that is not symmetric.

The following lemma provides some upper bounds between mim-width and the other parameters that we use in Section 5. All of these upper bounds are proved in [25].

Lemma 2.6 ([25])**.** *Let G be a graph. For every $A \subseteq V(G)$, $\mathsf{mim}(A)$ is upper bounded by $\mathsf{rw}(A), \mathsf{rw}_{\mathbb{Q}}(A)$ and $\log_2(\mathsf{nec}_1(A))$.*

Proof. Let S be the vertex set of a maximum induced matching of the graph $G[A, \overline{A}]$. Observe that the restriction of the matrix $M_{A,\overline{A}}$ to rows and columns in S is the identity matrix. Hence, $\mathsf{mim}(A)$ is upper bounded both by $\mathsf{rw}(A)$ and $\mathsf{rw}_{\mathbb{Q}}(A)$. It is clear that every pair of subsets of $S \cap A$ have a different neighborhood in $\overline{A}$. Thus, we have $2^{\mathsf{mim}(A)} \leq \mathsf{nec}_1(A)$. We deduce that $\mathsf{mim}(A) \leq \log_2(\mathsf{nec}_1(A))$. $\square$

The following lemma shows how the d-neighbor-width is upper bounded by the other parameters, most of the upper bounds were already proved in [1, 21].

Lemma 2.7 ([1, 21, 25])**.** *Let G be a graph. For every $A \subseteq V(G)$ and $d \in \mathbb{N}^+$, we have the following upper bounds on $\mathsf{nec}_d(A)$ and $\mathsf{nec}_d(\overline{A})$:*

- $(d+1)^{\mathsf{mw}(A)}$,
- $2^{d \cdot \mathsf{rw}(A)^2}$,
- $2^{\mathsf{rw}_{\mathbb{Q}}(A) \log_2(d \cdot \mathsf{rw}_{\mathbb{Q}}(A)+1)}$,
- $n^{d \cdot \mathsf{mim}(A)}$.

Proof. The first upper bound was proved in [25, Lemma 5.2.2]. The second upper bound was implicitly proved in [25] and is due to the fact that $\mathsf{nec}_d(A) \leq \mathsf{mw}(A)^{\mathsf{mim}(A)}$ [25, Lemma 5.2.3]. Since $\mathsf{mim}(A) \leq \mathsf{rw}(A)$ by Theorem 2.6 and $\mathsf{mw}(A) \leq 2^{\mathsf{rw}(A)}$ [22], we deduce that $\mathsf{nec}_d(A) \leq 2^{d \cdot \mathsf{rw}(A)^2}$. The third upper bound was proved in [21, Theorem 4.2]. The fourth was proved in [1, Lemma 2]. $\square$

We use the upper bounds of Lemma 2.7 to obtain from an $\mathsf{s\text{-}nec}_c(T,\delta)^{O(1)} \cdot n^{O(1)}$ time algorithm, with c a constant, the parameterized algorithms with parameters and running times given in Table 1.

In the following, we fix G an n-vertex graph, (T, δ) a rooted layout of G, and $\mathsf{w} : V(G) \to \mathbb{Q}$ a weight function over the vertices of G. We also assume that $V(G)$ is ordered.

3. Representative sets

In this section, we define a notion of representativity similar to the one defined in [4] and adapted to the notion of 1-neighbor-equivalence. Our notion of representativity is defined *w.r.t.* some node x of T and the 1-neighbor equivalence class of some set $R' \subseteq \overline{V_x}$. In our algorithm, R' will always belong to $\mathcal{R}^d_{\overline{V_x}}$ for some $d \in \mathbb{N}^+$. For the connectivity, $d = 1$ is enough but if we need more information for some reasons (for example the (σ, ρ)-domination or the acyclicity), we may take $d > 1$. This is not a problem as the d-neighbor equivalence class of R' is included in the 1-neighbor equivalence class of R'. Hence, in this section, we fix a node x of T and a set $R' \subseteq \overline{V_x}$ to avoid to overload the statements by the sentence "let x be a node of T and $R' \subseteq \overline{V_x}$".

We let $\mathsf{opt} \in \{\min, \max\}$; if we want to solve a maximization (or minimization) problem, we use $\mathsf{opt} = \max$ (or $\mathsf{opt} = \min$).

Definition 3.1 ((x, R')-representativity)**.** *For every $\mathcal{A} \subseteq 2^{V(G)}$ and $Y \subseteq V(G)$, we define*

$$\mathsf{best}(\mathcal{A}, Y) := \mathsf{opt}\{\mathsf{w}(X) \,:\, X \in \mathcal{A} \text{ and } G[X \cup Y] \text{ is connected }\}.$$

Let $\mathcal{A}, \mathcal{B} \subseteq 2^{V_x}$. We say that $\mathcal{B}$ (x, R')-represents $\mathcal{A}$ if for every $Y \subseteq \overline{V_x}$ such that $Y \equiv^1_{\overline{V_x}} R'$, we have $\mathsf{best}(\mathcal{A}, Y) = \mathsf{best}(\mathcal{B}, Y)$.

Notice that the (x, R')-representativity is an equivalence relation. The set $\mathcal{A}$ is meant to represent a set of partial solutions of $G[V_x]$ which have been computed. We expect to complete these partial solutions with partial solutions of $G[\overline{V_x}]$ which are equivalent to R' *w.r.t.* $\equiv^1_{\overline{V_x}}$. If $\mathcal{B}$ (x, R')-represents $\mathcal{A}$, then we can safely substitute $\mathcal{B}$ to $\mathcal{A}$ because the quality of the output of the dynamic programming algorithm will remain the same. Indeed, for every subset Y of $\overline{V_x}$ such that $Y \equiv^1_{\overline{V_x}} R'$, the optimum solutions obtained by the union of a partial solution in $\mathcal{A}$ and Y will have the same weight as the optimum solution obtained from the union of a set in $\mathcal{B}$ and Y.

We will show that given a set $\mathcal{A} \subseteq 2^{V_x}$, we can compute efficiently a small subset of $\mathcal{A}$ that (x, R')-represents $\mathcal{A}$. Similarly to [4], the small representative set we want to compute corresponds to a basis of maximum weight of some matrix. To compute this basis, we use the following lemma. The constant ω denotes the matrix multiplication exponent, which is known to be strictly less than 2.3727 due to [26].

Lemma 3.2 ([4])**.** *Let M be a binary $n \times m$-matrix with $m \leq n$ and let $\mathsf{w} : \{1, \dots, n\} \to \mathbb{Q}$ be a weight function on the rows of M. Then, one can find a basis of maximum (or minimum) weight of the row space of M in time $O(nm^{\omega-1})$.*

In order to compute a small (x, R')-representative set of a set $\mathcal{A} \subseteq 2^{V_x}$, the following theorem requires that the sets in $\mathcal{A}$ are pairwise equivalent *w.r.t.* $\equiv^1_{V_x}$. This is useful since in our algorithm we classify our sets of partial solutions with respect to this property. However, if one wants to compute a small (x, R')-representative set of a set $\mathcal{A}$ that does not respect this property, then it is enough to compute an (x, R')-representative set for each 1-neighbor equivalence class of $\mathcal{A}$. The union of these (x, R')-representative sets is an (x, R')-representative set of $\mathcal{A}$.

Theorem 3.3. *Let $R \in \mathcal{R}^1_{V_x}$. Then, there exists an algorithm* reduce *that, given $\mathcal{A} \subseteq 2^{V_x}$ such that $X \equiv^1_{V_x} R$ for all $X \in \mathcal{A}$, outputs in time $O(|\mathcal{A}| \cdot \mathsf{nec}_1(V_x)^{2(\omega-1)} \cdot n^2)$ a subset $\mathcal{B} \subseteq \mathcal{A}$ such that $\mathcal{B}$ (x, R')-represents $\mathcal{A}$ and $|\mathcal{B}| \leq \mathsf{nec}_1(V_x)^2$.*

Proof. We assume w.l.o.g. that $\mathsf{opt} = \max$, the proof is symmetric for $\mathsf{opt} = \min$. First, we suppose that $R' = \emptyset$. Observe that for every $Y \equiv^1_{\overline{V_x}} \emptyset$, we have $N(Y) \cap V_x = N(\emptyset) \cap V_x = \emptyset$. It follows that for every $Y \subseteq \overline{V_x}$ such that $Y \equiv^1_{\overline{V_x}} \emptyset$ and $Y \neq \emptyset$, we have $\mathsf{best}(\mathcal{A}, Y) = -\infty$. Moreover, by definition of best, we have $\mathsf{best}(\mathcal{A}, \emptyset) = \max\{\mathsf{w}(X) : X \in \mathcal{A} \text{ and } G[X] \text{ is connected}\}$. Hence, if $R' = \emptyset$, then it is sufficient to return $\mathcal{B} = \{X\}$, where X is an element of $\mathcal{A}$ of maximum weight that induces a connected graph.

Assume from now that $R' \neq \emptyset$. Let $X \in \mathcal{A}$. If there exists $C \in \mathsf{cc}(G[X])$ such that $N(C) \cap R' = \emptyset$, then for all $Y \equiv^1_{\overline{V_x}} R'$, we have $N(C) \cap Y = \emptyset$. Moreover, as $R' \neq \emptyset$, we have $Y \neq \emptyset$. Consequently, for every $Y \equiv^1_{\overline{V_x}} R'$, the graph $G[X \cup Y]$ is not connected. We can conclude that $\mathcal{A} \setminus \{X\}$ (x, R') represents $\mathcal{A}$. Thus, we can safely remove from $\mathcal{A}$ all such sets, and this can be done in time $|\mathcal{A}| \cdot n^2$. From now on, we may assume that for all $X \in \mathcal{A}$ and for all $C \in \mathsf{cc}(G[X])$, we have $N(C) \cap R' \neq \emptyset$. It is worth noticing that if $R = \emptyset$ or more generally $N(R) \cap R' = \emptyset$, then by assumption, $\mathcal{A} = \emptyset$.

Indeed, if $N(R) \cap R' = \emptyset$, then for every $X \in \mathcal{A}$, we have $N(X) \cap R' = N(R) \cap R' = \emptyset$ and in particular, for every $C \in \mathsf{cc}(G[X])$, we have $N(C) \cap R' = \emptyset$ (and we have assumed that no such set exists in $\mathcal{A}$).

Symmetrically, if for some $Y \subseteq \overline{V_x}$ there exists $C \in \mathsf{cc}(G[Y])$ such that $N(C) \cap R = \emptyset$, then for every $X \in \mathcal{A}$, the graph $G[X \cup Y]$ is not connected. Let $\mathcal{D}$ be the set of all subsets Y of $\overline{V_x}$ such that $Y \equiv^1_{\overline{V_x}} R'$ and, for all $C \in \mathsf{cc}(G[Y])$, we have $N(C) \cap R \neq \emptyset$. Notice that the sets in $2^{\overline{V_x}} \setminus \mathcal{D}$ do not matter for the (x, R')-representativity.

For every $Y \in \mathcal{D}$, we let v_Y be one fixed vertex of Y. In the following, we denote by $\mathcal{F}$ the set $\{(R'_1, R'_2) \in \mathcal{R}^1_{\overline{V_x}} \times \mathcal{R}^1_{\overline{V_x}}\}$. Let $\mathcal{M}$, $\mathcal{C}$, and $\overline{\mathcal{C}}$ be, respectively, an $(\mathcal{A}, \mathcal{D})$-matrix, an $(\mathcal{A}, \mathcal{F})$-matrix, and an $(\mathcal{F}, \mathcal{D})$-matrix such that

$$\mathcal{M}[X, Y] := \begin{cases} 1 & \text{if } G[X \cup Y] \text{ is connected,} \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathcal{C}[X, (R'_1, R'_2)] := \begin{cases} 1 & \text{if } \exists (X_1, X_2) \in \mathsf{ccut}(X) \text{ such that } N(X_1) \cap R'_2 = \emptyset \text{ and } N(X_2) \cap R'_1 = \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

$$\overline{\mathcal{C}}[(R'_1, R'_2), Y] := \begin{cases} 1 & \text{if } \exists (Y_1, Y_2) \in \mathsf{ccut}(Y) \text{ such that } v_Y \in Y_1,\ Y_1 \equiv^1_{\overline{V_x}} R'_1, \text{ and } Y_2 \equiv^1_{\overline{V_x}} R'_2, \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, $\mathcal{M}$ contains all the information we need. Indeed, it is easy to see that a basis of maximum weight of the row space of $\mathcal{M}$ in $GF(2)$ is an (x, R')-representative set of $\mathcal{A}$. But,

$\mathcal{M}$ is too big to be computable efficiently. Instead, we prove that a basis of maximum weight of the row space of $\mathcal{C}$ is an (x, R')-representative set of $\mathcal{A}$. This follows from the fact that $(\mathcal{C} \cdot \overline{\mathcal{C}})[X, Y]$ equals the number of consistent cuts (W_1, W_2) in $\mathsf{ccut}(X \cup Y)$ such that $v_Y \in W_1$. That is $(\mathcal{C} \cdot \overline{\mathcal{C}})[X, Y] = 2^{|\mathsf{cc}(G[X \cup Y])|-1}$ and thus $(\mathcal{C} \cdot \overline{\mathcal{C}})[X, Y]$ is odd if and only if $G[X \cup Y]$ is connected. We deduce the running time of reduce and the size of $\mathsf{reduce}(\mathcal{A})$ from the size of $\mathcal{C}$ (*i.e.* $|\mathcal{A}| \cdot \mathsf{nec}_1(V_x)^2$) and the fact that $\mathcal{C}$ is easy to compute.

We start by proving that $\mathcal{M} =_2 \mathcal{C} \cdot \overline{\mathcal{C}}$, where $=_2$ denotes the equality in $GF(2)$. Let $X \in \mathcal{A}$ and $Y \in \mathcal{D}$. We want to prove the following equality

$$(\mathcal{C} \cdot \overline{\mathcal{C}})[X, Y] = \sum_{(R'_1, R'_2) \in \mathcal{F}} \mathcal{C}[X, (R'_1, R'_2)] \cdot \overline{\mathcal{C}}[(R'_1, R'_2), Y] = 2^{|\mathsf{cc}(G[X \cup Y])|-1}.$$

We prove this equality with the following two claims.

Claim 3.3.1. *We have* $\mathcal{C}[X, (R'_1, R'_2)] \cdot \overline{\mathcal{C}}[(R'_1, R'_2), Y] = 1$ *if and only if there exists* $(W_1, W_2) \in \mathsf{ccut}(X \cup Y)$ *such that* $v_Y \in W_1$, $W_1 \cap Y \equiv^1_{V_x} R'_1$ *and* $W_2 \cap Y \equiv^1_{V_x} R'_2$.

Proof. By definition, we have $\mathcal{C}[X, (R'_1, R'_2)] \cdot \overline{\mathcal{C}}[(R'_1, R'_2), Y] = 1$, if and only if

(a) $\exists (Y_1, Y_2) \in \mathsf{ccut}(Y)$ such that $v_Y \in Y_1$, $Y_1 \equiv^1_{V_x} R'_1$, $Y_2 \equiv^1_{V_x} R'_2$, and

(b) $\exists (X_1, X_2) \in \mathsf{ccut}(X)$ such that $N(X_1) \cap R'_2 = \emptyset$ and $N(X_2) \cap R'_1 = \emptyset$.

Let $(Y_1, Y_2) \in \mathsf{ccut}(Y)$ and $(X_1, X_2) \in \mathsf{ccut}(X)$ that satisfy, respectively, Properties (a) and (b). By definition of $\equiv^1_{V_x}$, we have $N(X_1) \cap Y_2 = \emptyset$ because $N(X_1) \cap R'_2 = \emptyset$ and $Y_2 \equiv^1_{V_x} R'_2$. Symmetrically, we have $N(X_2) \cap Y_1 = \emptyset$. By Fact 2.2, we deduce that $(X_1 \cup Y_1, X_2 \cup Y_2) \in \mathsf{ccut}(X \cup Y)$. This proves the claim. □

Claim 3.3.2. *Let* (W_1, W_2) *and* $(W'_1, W'_2) \in \mathsf{ccut}(X \cup Y)$. *We have* $W_1 \cap Y \equiv^1_{V_x} W'_1 \cap Y$ *and* $W_2 \cap Y \equiv^1_{V_x} W'_2 \cap Y$ *if and only if* $W_1 = W'_1$ *and* $W_2 = W'_2$.

Proof. We start by an observation about the connected components of $X \cup Y$. As $Y \in \mathcal{D}$, for all $C \in \mathsf{cc}(G[Y])$, we have $N(C) \cap R \neq \emptyset$. Moreover, by assumption, for all $C \in \mathsf{cc}(G[X])$, we have $N(C) \cap R' \neq \emptyset$. Since $X \equiv^1_{V_x} R$ and $Y \equiv^1_{V_x} R'$, every connected component of $G[X \cup Y]$ contains at least one vertex of X and one vertex of Y.

Suppose that $W_1 \cap Y \equiv^1_{V_x} W'_1 \cap Y$ and $W_2 \cap Y \equiv^1_{V_x} W'_2 \cap Y$. Assume towards a contradiction that $W_1 \neq W'_1$ and $W_2 \neq W'_2$. Since $W_1 \neq W'_1$, by Fact 2.1, we deduce that there exists $C \in \mathsf{cc}(G[X \cup Y])$ such that either (1) $C \subseteq W_1$ and $C \subseteq W'_2$ or (2) $C \subseteq W'_1$ and $C \subseteq W_2$. We can assume w.l.o.g. that there exits $C \in \mathsf{cc}(G[X \cup Y])$ such that $C \subseteq W_1$ and $C \subseteq W'_2$. From the above observation, C contains at least one vertex of X and one of Y, and we have $N(C \cap X) \cap (W_1 \cap Y) \neq \emptyset$ and $N(C \cap X) \cap (W'_2 \cap Y) \neq \emptyset$. But, since $W_2 \cap Y \equiv^1_{V_x} W'_2 \cap Y$, we have $N(C \cap X) \cap (W_2 \cap Y) \neq \emptyset$. This implies in particular that $N(W_1) \cap W_2 \neq \emptyset$. It is a contradiction with the fact that $(W_1, W_2) \in \mathsf{ccut}(X \cup Y)$. □

Notice that Claim 3.3.2 implies that for every $(R'_1, R'_2) \in \mathcal{F}$, there exists at most one consistent cut $(W_1, W_2) \in \mathsf{ccut}(X \cup Y)$ such that $v_Y \in W_1$, $W_1 \cap Y \equiv^1_{V_x} R'_1$, and $W_2 \cap Y \equiv^1_{V_x} R'_2$. We can thus conclude from these two claims that

$$(\mathcal{C} \cdot \overline{\mathcal{C}})[X, Y] = |\{(W_1, W_2) \in \mathsf{ccut}(X \cup Y) \,:\, v_Y \in W_1\}|.$$

By Fact 2.1, we deduce that $(\mathcal{C} \cdot \overline{\mathcal{C}})[X, Y] = 2^{|\mathsf{cc}(G[X \cup Y])|-1}$ since every connected component of $G[X \cup Y]$ can be in both sides of a consistent cut at the exception of the connected component containing v_Y. Hence, $(\mathcal{C} \cdot \overline{\mathcal{C}})[X, Y]$ is odd if and only if $|\mathsf{cc}(G[X \cup Y])| = 1$. We conclude that $\mathcal{M} =_2 \mathcal{C} \cdot \overline{\mathcal{C}}$.

Let $\mathcal{B} \subseteq \mathcal{A}$ be a basis of maximum weight of the row space of $\mathcal{C}$ over $GF(2)$. We claim that $\mathcal{B}$ (x, R')-represents $\mathcal{A}$.

Let $Y \subseteq \overline{V_x}$ such that $Y \equiv^1_{V_x} R'$. Observe that, by definition of $\mathcal{D}$, if $Y \notin \mathcal{D}$, then $\mathsf{best}(\mathcal{A}, Y) = \mathsf{best}(\mathcal{B}, Y) = -\infty$. Thus it is sufficient to prove that for every $Y \in \mathcal{D}$, we have $\mathsf{best}(\mathcal{A}, Y) = \mathsf{best}(\mathcal{B}, Y)$.

Let $X \in \mathcal{A}$ and $Y \in \mathcal{D}$. Recall that we have proved that $M[X,Y] =_2 (\mathcal{C} \cdot \overline{\mathcal{C}})[X,Y]$. Since $\mathcal{B}$ is a basis of $\mathcal{C}$, there exists $\mathcal{B}' \subseteq \mathcal{B}$ such that for each $(R_1', R_2') \in \mathcal{F}$, we have $\mathcal{C}[X,(R_1',R_2')] =_2 \sum_{W \in \mathcal{B}'} \mathcal{C}[W,(R_1',R_2')]$. Thus, we have the following equalities

$$\begin{aligned}
\mathcal{M}[X,Y] &=_2 \sum_{(R_1',R_2') \in \mathcal{F}} \mathcal{C}[X,(R_1',R_2')] \cdot \overline{\mathcal{C}}[(R_1',R_2'),Y] \\
&=_2 \sum_{(R_1',R_2') \in \mathcal{F}} \left(\sum_{W \in \mathcal{B}'} \mathcal{C}[W,(R_1',R_2')] \right) \cdot \overline{\mathcal{C}}[(R_1',R_2'),Y] \\
&=_2 \sum_{W \in \mathcal{B}'} \left(\sum_{(R_1',R_2') \in \mathcal{F}} \mathcal{C}[W,(R_1',R_2')] \cdot \overline{\mathcal{C}}[(R_1',R_2'),Y] \right) \\
&=_2 \sum_{W \in \mathcal{B}'} (\mathcal{C} \cdot \overline{\mathcal{C}})[W,Y] =_2 \sum_{W \in \mathcal{B}'} \mathcal{M}[W,Y].
\end{aligned}$$

If $\mathcal{M}[X,Y] = 1$ (*i.e.* $G[X \cup Y]$ is connected), then there is an odd number of sets W in $\mathcal{B}'$ such that $\mathcal{M}[W,Y] = 1$ (*i.e.* $G[W \cup Y]$ is connected). Hence, there exists at least one $W \in \mathcal{B}'$ such that $G[W \cup Y]$ is connected. Let $W \in \mathcal{B}'$ such that $\mathcal{M}[W,Y] = 1$ and $\mathsf{w}(W)$ is maximum. Assume towards a contradiction that $\mathsf{w}(W) < \mathsf{w}(X)$. Notice that $(\mathcal{B} \setminus \{W\}) \cup \{X\}$ is also a basis of $\mathcal{C}$ since the set of independent row sets of a matrix forms a matroid. Since $\mathsf{w}(W) < \mathsf{w}(X)$, the weight of the basis $(\mathcal{B} \setminus \{W\}) \cup \{X\}$ is strictly greater than the weight of the basis $\mathcal{B}$, yielding a contradiction. Thus $\mathsf{w}(X) \leq \mathsf{w}(W)$. Hence, for all $Y \in \mathcal{D}$ and all $X \in \mathcal{A}$, if $G[X \cup Y]$ is connected, then there exists $W \in \mathcal{B}$ such that $G[W \cup Y]$ is connected and $\mathsf{w}(X) \leq \mathsf{w}(W)$. This is sufficient to prove that $\mathcal{B}$ (x, R')-represents $\mathcal{A}$. Since $\mathcal{B}$ is a basis, the size of $\mathcal{B}$ is at most the number of columns of $\mathcal{C}$, thus, $|\mathcal{B}| \leq \mathsf{nec}_1(V_x)^2$.

It remains to prove the running time. We claim that $\mathcal{C}$ is easy to compute.

By Fact 2.1, $\mathcal{C}[X,(R_1',R_2')] = 1$ if and only if for each $C \in \mathsf{cc}(G[X])$, we have either $N(C) \cap R_1' = \emptyset$ or $N(C) \cap R_2' = \emptyset$. Thus, each entry of $\mathcal{C}$ is computable in time $O(n^2)$. Since $\mathcal{C}$ has $|\mathcal{A}| \cdot |\mathcal{R}^1_{\overline{V_x}}|^2 = |\mathcal{A}| \cdot \mathsf{nec}_1(V_x)^2$ entries, we can compute $\mathcal{C}$ in time $O(|\mathcal{A}| \cdot \mathsf{nec}_1(V_x)^2 \cdot n^2)$. Furthermore, by Lemma 3.2, a basis of maximum weight of $\mathcal{C}$ can be computed in time $O(|\mathcal{A}| \cdot \mathsf{nec}_1(V_x)^{2(\omega-1)})$. We conclude that $\mathcal{B}$ can be computed in time $O(|\mathcal{A}| \cdot \mathsf{nec}_1(V_x)^{2(\omega-1)} \cdot n^2)$. ☐

Now to boot up a dynamic programming algorithm P on some rooted layout (T, δ) of G, we can use the function reduce to keep the size of the sets of partial solutions bounded by $\mathsf{s\text{-}nec}_1(T,\delta)^2$. We call P' the algorithm obtained from P by calling the function reduce at every step of computation. We can assume that the set of partial solutions $\mathcal{A}_r$ computed by P and associated with the root r of (T,δ) contains an optimal solution (this will be the cases for our algorithms). To prove the correctness of P', we need to prove that $\mathcal{A}_r'$ $(r, \emptyset)$-represents $\mathcal{A}_r$ where $\mathcal{A}_r'$ is the set of partial solutions computed by P' and associated with r. For doing so, we need to prove that at each step of the algorithm the operations we use preserve the representativity. The following fact states that we can use the union without restriction, it follows directly from Definition 3.1 of (x,R')-representativity.

Fact 3.4. *If $\mathcal{B}$ and $\mathcal{D}$ (x,R')-represents, respectively, $\mathcal{A}$ and $\mathcal{C}$, then $\mathcal{B} \cup \mathcal{D}$ (x,R')-represents $\mathcal{A} \cup \mathcal{C}$.*

The second operation we use in our dynamic programming algorithms is the merging operator $\otimes$. In order to safely use it, we need the following notion of compatibility.

Definition 3.5 (d-(R,R')-compatibility)**.** *Suppose that x is an internal node of T with a and b as children. Let $d \in \mathbb{N}^+$ and $R \in \mathcal{R}^d_{V_x}$. We say that $(A, A') \in \mathcal{R}^d_{V_a} \times \mathcal{R}^d_{\overline{V_a}}$ and $(B, B') \in \mathcal{R}^d_{V_b} \times \mathcal{R}^d_{\overline{V_b}}$ are d-(R,R')-compatible if we have:*

- $A \cup B \equiv^d_{V_x} R$,
- $A' \equiv^d_{\overline{V_a}} B \cup R'$, *and*

- $B' \equiv^d_{\overline{V_b}} A \cup R'$.

The d-(R, R')-compatibility just tells which partial solutions from V_a and V_b can be joined to possibly form a partial solution in V_x.

Lemma 3.6. *Suppose that x is an internal node of T with a and b as children. Let $d \in \mathbb{N}^+$ and $R \in \mathcal{R}^d_{V_x}$. Let $(A, A') \in \mathcal{R}^d_{V_a} \times \mathcal{R}^d_{\overline{V_a}}$ and $(B, B') \in \mathcal{R}^d_{V_b} \times \mathcal{R}^d_{\overline{V_b}}$ that are d-(R, R')-compatible. Let $\mathcal{A} \subseteq 2^{V_a}$ such that for all $X \in \mathcal{A}$, we have $X \equiv^d_{V_a} A$, and let $\mathcal{B} \subseteq 2^{V_b}$ such that for all $W \in \mathcal{B}$, we have $W \equiv^d_{V_b} B$. If $\mathcal{A}' \subseteq \mathcal{A}$ (a, A')-represents $\mathcal{A}$ and $\mathcal{B}' \subseteq \mathcal{B}$ (b, B')-represents $\mathcal{B}$, then $\mathcal{A}' \bigotimes \mathcal{B}'$ (x, R')-represents $\mathcal{A} \bigotimes \mathcal{B}$.*

Proof. We assume w.l.o.g. that $\mathsf{opt} = \max$, the proof is symmetric for $\mathsf{opt} = \min$. Suppose that $\mathcal{A}' \subseteq \mathcal{A}$ (a, A')-represents $\mathcal{A}$ and $\mathcal{B}' \subseteq \mathcal{B}$ (b, B')-represents $\mathcal{B}$. To prove the lemma, it is sufficient to prove that $\mathsf{best}(\mathcal{A}' \bigotimes \mathcal{B}', Y) = \mathsf{best}(\mathcal{A} \bigotimes \mathcal{B}, Y)$ for every $Y \equiv^1_{\overline{V_x}} R'$.

Let $Y \subseteq \overline{V_x}$ such that $Y \equiv^1_{\overline{V_x}} R'$. We start by proving the following facts

(a) for every $W \in \mathcal{B}$, we have $W \cup Y \equiv^1_{\overline{V_a}} A'$,
(b) for every $X \in \mathcal{A}$, we have $X \cup Y \equiv^1_{\overline{V_b}} B'$.

Let $W \in \mathcal{B}$. Owing to the d-(R, R')-compatibility, we have $B \cup R' \equiv^d_{\overline{V_a}} A'$. Since $W \equiv^d_{V_b} B$ and $V_b \subseteq \overline{V_a}$, by Fact 2.3, we deduce also that $W \equiv^d_{\overline{V_a}} B$ and thus $W \cup R' \equiv^d_{\overline{V_a}} A'$. In particular, we have $W \cup R' \equiv^1_{\overline{V_a}} A'$. Similarly, we deduce from Fact 2.3 that $W \cup Y \equiv^1_{\overline{V_a}} A'$ because $Y \equiv^1_{\overline{V_x}} R'$ and $\overline{V_x} \subseteq \overline{V_a}$. This proves Fact (a). The proof for Fact (b) is symmetric.

Now observe that, by the definitions of best and of the merging operator $\bigotimes$, we have (even if $\mathcal{A} = \emptyset$ or $\mathcal{B} = \emptyset$)

$$\mathsf{best}(\mathcal{A} \bigotimes \mathcal{B}, Y) = \max\{\mathsf{w}(X) + \mathsf{w}(W) \ : \ X \in \mathcal{A} \wedge W \in \mathcal{B} \wedge G[X \cup W \cup Y] \text{ is connected}\}.$$

Since $\mathsf{best}(\mathcal{A}, W \cup Y) = \max\{\mathsf{w}(X) \ : \ X \in \mathcal{A} \wedge G[X \cup W \cup Y] \text{ is connected}\}$, we deduce that

$$\mathsf{best}\big(\mathcal{A} \bigotimes \mathcal{B}, Y\big) = \max\{\mathsf{best}(\mathcal{A}, W \cup Y) + \mathsf{w}(W) \ : \ W \in \mathcal{B}\}.$$

Since $\mathcal{A}'$ (a, A')-represents $\mathcal{A}$, by Fact (a), we have

$$\begin{aligned} \mathsf{best}\big(\mathcal{A} \bigotimes \mathcal{B}, Y\big) &= \max\{\mathsf{best}(\mathcal{A}', W \cup Y) + \mathsf{w}(W) \ : \ W \in \mathcal{B}\} \\ &= \mathsf{best}\big(\mathcal{A}' \bigotimes \mathcal{B}, Y\big). \end{aligned}$$

Symmetrically, we deduce from Fact (b) that $\mathsf{best}\big(\mathcal{A}' \bigotimes \mathcal{B}, Y\big) = \mathsf{best}\big(\mathcal{A}' \bigotimes \mathcal{B}', Y\big)$. This stands for every $Y \subseteq \overline{V_x}$ such that $Y \equiv^1_{\overline{V_x}} R'$. Thus, we conclude that $\mathcal{A}' \bigotimes \mathcal{B}'$ (x, R')-represents $\mathcal{A} \bigotimes \mathcal{B}$. □

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

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 $u \in U \cap D$, we have $|N(u) \cap D| \in \sigma$, and for every vertex $u \in U \setminus D$, we have $|N(u) \cap D| \in \rho$. A subset D of $V(G)$ is a (σ, ρ)-*dominating set* (resp. co-(σ, ρ)-*dominating set*) if D (resp. $V(G) \setminus D$) (σ, ρ)-*dominates* $V(G)$. The Connected (σ, ρ)-Dominating Set problem asks, given a weighted graph G, a maximum or minimum (σ, ρ)-dominating set which induces a connected graph. Similarly, one can define Connected Co-(σ, ρ)-Dominating Set. Examples of some Connected (co)-(σ, ρ)-Dominating Set problems are shown in Table 2.

Let $d(\mathbb{N}) := 0$, and for a finite or co-finite subset μ of $\mathbb{N}$, let

$$d(\mu) := 1 + \min(\max(\mu), \max(\mathbb{N} \setminus \mu)).$$

Let $d := \max\{1, d(\sigma), d(\rho)\}$. 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 $\mathbb{N} \setminus \{1, \ldots, d(\mu)\}$.

Fact 4.1. *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* $\min(d(\mu), a + b) \in \mu$.

As in [7], we use the *d-neighbor equivalence* relation to characterize the (σ, ρ)-domination of the partial solutions.

σ	ρ	d	Version	Standard name
$\mathbb{N}$	$\mathbb{N}^+$	1	Connected	Connected Dominating Set
$\{q\}$	$\mathbb{N}$	$q+1$	Connected	Connected Induced q-Regular Subgraph
$\mathbb{N}$	$\{1\}$	2	Connected	Connected Perfect Dominating Set
$\{0\}$	$\mathbb{N}$	1	Connected Co	Connected Vertex Cover

Table 2. Examples of (Co)-(σ, ρ)-Dominating Set problems. To solve these problems, we use the d-neighbor equivalence relation with $d := \max\{1, d(\sigma), d(\rho)\}$. Column d shows the value of d for each problem.

We will need the following lemma in our proof.

Lemma 4.2 ([7]). *Let $A \subseteq V(G)$. Let $X \subseteq A$ and $Y, Y' \subseteq \overline{A}$ such that $Y \equiv^d_A Y'$. Then $(X \cup Y)$ (σ, ρ)-dominates A if and only if $(X \cup Y')$ (σ, ρ)-dominates A.*

In this section, we present an algorithm that computes a maximum (or minimum) connected (σ, ρ)-dominating set with the graph G and the layout (T, δ) as inputs.

Its running time is $O(\mathsf{s\text{-}nec}_d(T, \delta)^3 \cdot \mathsf{s\text{-}nec}_1(T, \delta)^{2(\omega+1)} \cdot \log(\mathsf{s\text{-}nec}_d(T, \delta)) \cdot n^3)$. The same algorithm, with some little modifications, will be able to find a minimum Steiner tree or a maximum or minimum connected co-(σ, ρ)-dominating set as well.

For each node x of T and for each pair $(R, R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$, we will compute a set of partial solutions $\mathcal{D}_x[R, R']$ coherent with (R, R') that (x, R')-represents the set of all partials solutions coherent with (R, R'). We say that a set $X \subseteq V_x$ is coherent with (R, R') if $X \equiv^d_{V_x} R$ and $X \cup R'$ (σ, ρ) dominates V_x. Observe that by Lemma 4.2, we have $X \cup Y$ (σ, ρ)-dominates V_x for all $Y \equiv^d_{\overline{V_x}} R'$ and for all $X \subseteq V_x$ coherent with (R, R'). We compute these sets by a bottom-up dynamic programming algorithm, starting at the leaves of T. By calling the function reduce defined in Section 3, each set $\mathcal{D}_x[R, R']$ contains at most $\mathsf{s\text{-}nec}_1(T, \delta)^2$ partial solutions. If we want to compute a maximum (resp. minimum) connected (σ, ρ)-dominating set, we use the framework of Section 3 with $\mathsf{opt} = \max$ (resp. $\mathsf{opt} = \min$). If G admits a connected (σ, ρ)-dominating set, then a maximum (or minimum) connected (σ, ρ)-dominating set can be found by looking at the entry $\mathcal{D}_r[\emptyset, \emptyset]$ with r the root of T.

We begin by defining the sets of partial solutions for which we will compute representative sets.

Definition 4.3. *Let $x \in V(T)$. For all pairs $(R, R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$, we let $\mathcal{A}_x[R, R'] := \{X \subseteq V_x : X \equiv^d_{V_x} R$ and $X \cup R'$ (σ, ρ)-dominates $V_x\}$.*

For each node x of $V(T)$, our algorithm will compute a table $\mathcal{D}_x$ that satisfies the following invariant.

Invariant. For every $(R, R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$, the set $\mathcal{D}_x[R, R']$ is a subset of $\mathcal{A}_x[R, R']$ of size at most $\mathsf{s\text{-}nec}_1(T, \delta)^2$ that (x, R')-represents $\mathcal{A}_x[R, R']$.

Notice that, by the definition of $\mathcal{A}_r[\emptyset, \emptyset]$ and Definition 3.1, if G admits a connected (σ, ρ)-dominating set, then $\mathcal{D}_r[\emptyset, \emptyset]$ must contain a maximum (or minimum) connected (σ, ρ)-dominating set.

The following lemma provides an equality between the entries of the table $\mathcal{A}_x$ and the entries of the tables $\mathcal{A}_a$ and $\mathcal{A}_b$ for each internal node $x \in V(T)$ with a and b as children. We use this lemma to prove, by induction, that the entry $\mathcal{D}_x[R,R']$ (x,R')-represents $\mathcal{A}_x[R,R']$ for every $(R,R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$. Note that this lemma can be deduced from [7].

Lemma 4.4. *For all $(R,R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$, we have*

$$\mathcal{A}_x[R,R'] = \bigcup_{(A,A'),\ (B,B')\ d\text{-}(R,R')\text{-compatible}} \mathcal{A}_a[A,A'] \bigotimes \mathcal{A}_b[B,B'].$$

Proof. The lemma is implied by the two following claims.

Claim 4.4.1. *For all $X \in \mathcal{A}_x[R,R']$, there exist d-(R,R')-compatible pairs (A,A') and (B,B') such that $X \cap V_a \in \mathcal{A}_a[A,A']$ and $X \cap V_b \in \mathcal{A}_b[B,B']$.*

Proof. Let $X \in \mathcal{A}_x[R,R']$, $X_a := X \cap V_a$ and $X_b := X \cap V_b$. Let $A := \mathsf{rep}^d_{V_a}(X_a)$ and $A' := \mathsf{rep}^d_{\overline{V_a}}(X_b \cup R')$. Symmetrically, we define $B := \mathsf{rep}^d_{V_b}(X_b)$ and $B' := \mathsf{rep}^d_{\overline{V_b}}(X_a \cup R')$.

We claim that $X_a \in \mathcal{A}_a[A,A']$. As $X \in \mathcal{A}_x[R,R']$, we know, by Definition 2, that $X \cup R' = X_a \cup X_b \cup R'$ is a (σ,ρ)-dominating set of V_x. In particular, $X_a \cup (X_b \cup R')$ (σ,ρ)-dominates V_a. Since $A' \equiv^d_{\overline{V_a}} X_b \cup R'$, by Lemma 4.2, we conclude that $X_a \cup A'$ (σ,ρ)-dominates V_a. As $A \equiv^d_{V_a} X_a$, we have $X_a \in \mathcal{A}_a[A,A']$. By symmetry, we have $B \in \mathcal{A}_b[B,B']$.

It remains to prove that (A,A') and (B,B') are d-(R,R')-compatible.

- By construction, we have $X_a \cup X_b \equiv^d_{V_x} R$. As $A \equiv^d_{V_a} X_a$ and from Fact 2.3, we have $A \cup X_b \equiv^d_{V_x} R$. Since $B \equiv^d_{V_b} X_b$, we deduce that $A \cup B \equiv^d_{V_x} R$.
- By definition, we have $A' \equiv^d_{\overline{V_a}} X_b \cup R'$. As $B \equiv^d_{V_b} X_b$ and by Fact 2.3, we have $A' \equiv^d_{\overline{V_a}} B \cup R'$. Symmetrically, we deduce that $B' \equiv^d_{\overline{V_b}} R' \cup A$.

Thus, (A,A') and (B,B') are d-(R,R')-compatible. ☐

Claim 4.4.2. *For every $X_a \in \mathcal{A}_a[A,A']$ and $X_b \in \mathcal{A}_b[B,B']$ such that (A,A') and (B,B') are d-(R,R')-compatible, we have $X_a \cup X_b \in \mathcal{A}_x[R,R']$.*

Proof. Since $X_a \equiv^d_{V_a} A$ and $X_b \equiv^d_{V_b} B$, by Fact 2.3, we deduce that $X_a \cup X_b \equiv^d_{V_x} A \cup B$. Thus, by the definition of d-(R,R')-compatibility, we have $X_a \cup X_b \equiv^d_{V_x} R$.

It remains to prove that $X_a \cup X_b \cup R'$ (σ,ρ)-dominates V_x. As before, one can check that Fact 2.3 implies that $X_b \cup R' \equiv^d_{\overline{V_a}} B \cup R'$. From Lemma 4.2, we conclude that $X_a \cup X_b \cup R'$ (σ,ρ)-dominates V_a. Symmetrically, we prove that $X_a \cup X_b \cup R'$ (σ,ρ)-dominates V_b. As $V_x = V_a \cup V_b$, we deduce that $X_a \cup X_b \cup R'$ (σ,ρ)-dominates V_x. Hence, we have $X_a \cup X_b \in \mathcal{A}_x[R,R']$. ☐

☐

We are now ready to prove the main theorem of this section.

Theorem 4.5. *There exists an algorithm that, given an n-vertex graph G and a rooted layout (T,δ) of G, computes a maximum (or minimum) connected (σ,ρ)-dominating set in time $O(\mathsf{s\text{-}nec}_d(T,\delta)^3 \cdot \mathsf{s\text{-}nec}_1(T,\delta)^{2(\omega+1)} \cdot \log(\mathsf{s\text{-}nec}_d(T,\delta)) \cdot n^3)$ with $d := \max\{1, d(\sigma), d(\rho)\}$.*

Proof. The algorithm is a usual bottom-up dynamic programming algorithm and computes for each node x of T the table $\mathcal{D}_x$.

The first step of our algorithm is to compute, for each $x \in V(T)$, the sets $\mathcal{R}^d_{V_x}$, $\mathcal{R}^d_{\overline{V_x}}$ and a data structure to compute $\mathsf{rep}^d_{V_x}(X)$ and $\mathsf{rep}^d_{\overline{V_x}}(X)$ for each $X \subseteq V_x$ in time $O(\log(\mathsf{s\text{-}nec}_d(T,\delta)) \cdot n^2)$. As T has $2n-1$ nodes, by Lemma 2.4, we can compute these sets and data structures in time $O(\mathsf{s\text{-}nec}_d(T,\delta) \cdot \log(\mathsf{s\text{-}nec}_d(T,\delta)) \cdot n^3)$.

Let x be a leaf of T with $V_x = \{v\}$. Observe that for all $(R,R') \in \mathcal{R}^{V_x}_d \times \mathcal{R}^{\overline{V_x}}_d$, we have $\mathcal{A}_x[R,R'] \subseteq 2^{V_x} = \{\emptyset, \{v\}\}$. Thus, our algorithm can directly compute $\mathcal{A}_x[R,R']$ and set $\mathcal{D}_x[R,R'] := \mathcal{A}_x[R,R']$. In this case, the invariant trivially holds.

Now let x be an internal node with a and b as children such that the invariant holds for a and b. For each $(R, R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$, the algorithm computes $\mathcal{D}_x[R, R'] := \mathsf{reduce}(\mathcal{B}_x[R, R'])$, where the set $\mathcal{B}_x[R, R']$ is defined as follows

$$\mathcal{B}_x[R, R'] := \bigcup_{(A,A'),\ (B,B')\ d\text{-}(R,R')\text{-compatible}} \mathcal{D}_a[A, A'] \bigotimes \mathcal{D}_b[B, B'].$$

We claim that the invariant holds for x. Let $(R, R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$.

We start by proving that the set $\mathcal{B}_x[R, R']$ is an (x, R')-representative set of $\mathcal{A}_x[R, R']$. By Lemma 3.6, for all d-(R, R')-compatible pairs (A, A') and (B, B'), we have

$$\mathcal{D}_a[A, A'] \bigotimes \mathcal{D}_b[B, B'] \ (x, R')\text{-represents } \mathcal{A}_a[A, A'] \bigotimes \mathcal{A}_b[B, B'].$$

By Lemma 4.4 and by construction of $\mathcal{D}_x[R, R']$ and from Fact 3.4, we conclude that $\mathcal{B}_x[R, R']$ (x, R')-represents $\mathcal{A}_x[R, R']$.

From the invariant, we have $\mathcal{D}_a[A, A'] \subseteq \mathcal{A}_a[A, A']$ and $\mathcal{D}_b[B, B'] \subseteq \mathcal{A}_b[B, B']$, for all d-(R, R')-compatible pairs (A, A') and (B, B'). Thus, from Lemma 4.4, it is clear that by construction, we have $\mathcal{B}_x[R, R'] \subseteq \mathcal{A}_x[R, R']$. Hence, $\mathcal{B}_x[R, R']$ is a subset and an (x, R')-representative set of $\mathcal{A}_x[R, R']$.

Notice that for each $X \in \mathcal{B}_x[R, R']$, we have $X \equiv^d_{V_x} R$. Thus, we can apply Theorem 3.3 and the function reduce on $\mathcal{B}_x[R, R']$. By Theorem 3.3, $\mathcal{D}_x[R, R']$ is a subset and an (x, R')-representative set of $\mathcal{B}_x[R, R']$. Thus $\mathcal{D}_x[R, R']$ is a subset of $\mathcal{A}_x[R, R']$. Notice that the (x, R')-representativity is an equivalence relation and in particular it is transitive. Consequently, $\mathcal{D}_x[R, R']$ (x, R')-represents $\mathcal{A}_x[R, R']$. From Theorem 3.3, the size of $\mathcal{D}_x[R, R']$ is at most $\mathsf{nec}_1(V_x)^2$ and that $\mathcal{D}_x[R, R'] \subseteq \mathcal{B}_x[R, R']$. As $\mathsf{nec}_1(V_x) \leq \mathsf{s\text{-}nec}_1(T, \delta)$ and $\mathcal{B}_x[R, R'] \subseteq \mathcal{A}_x[R, R']$, we conclude that the invariant holds for x.

By induction, the invariant holds for all nodes of T. The correctness of the algorithm follows from the fact that $\mathcal{D}_r[\emptyset, \emptyset]$ $(r, \emptyset)$-represents $\mathcal{A}_r[\emptyset, \emptyset]$.

Running Time. Let x be a node of T. Suppose first that x is a leaf of T. Then $|\mathcal{R}^d_{V_x}| \leq 2$ and $|\mathcal{R}^d_{\overline{V_x}}| \leq d$. Thus, $\mathcal{D}_x$ can be computed in time $O(d \cdot n)$.

Assume now that x is an internal node of T with a and b as children.

Notice that, by Definition 3.5, for every $(A, B, R') \in \mathcal{R}^d_{V_a} \times \mathcal{R}^d_{V_b} \times \mathcal{R}^d_{\overline{V_x}}$, there exists only one tuple $(A', B', R) \in \mathcal{R}^d_{\overline{V_a}} \times \mathcal{R}^d_{\overline{V_b}} \times \mathcal{R}^d_{V_x}$ such that (A, A') and (B, B') are d-(R, R')-compatible. More precisely, you have to take $R = \mathsf{rep}^d_{V_x}(A \cup B)$, $A' = \mathsf{rep}^d_{\overline{V_a}}(R' \cup B)$, and $B' = \mathsf{rep}^d_{\overline{V_b}}(R' \cup A)$. Thus, there are at most $\mathsf{s\text{-}nec}_d(T, \delta)^3$ tuples (A, A', B, B', R, R') such that (A, A') and (B, B') are d-(R, R')-compatible. It follows that we can compute the intermediary table $\mathcal{B}_x$ by doing the following.

- Initialize each entry of $\mathcal{B}_x$ to $\emptyset$.
- For each $(A, B, R') \in \mathcal{R}^d_{V_a} \times \mathcal{R}^d_{V_b} \times \mathcal{R}^d_{\overline{V_x}}$, compute $R' := \mathsf{rep}^d_{V_x}(A \cup B)$, $A' = \mathsf{rep}^d_{\overline{V_a}}(R' \cup B)$, and $B' = \mathsf{rep}^d_{\overline{V_b}}(R' \cup A)$. Then, update $\mathcal{B}_x[R, R'] := \mathcal{B}_x[R, R'] \cup (\mathcal{D}_a[A, A'] \bigotimes \mathcal{D}_b[B, B'])$.

Each call to the functions $\mathsf{rep}^d_{V_x}$, $\mathsf{rep}^d_{\overline{V_a}}$, and $\mathsf{rep}^d_{\overline{V_b}}$ takes $\log(\mathsf{s\text{-}nec}_d(T, \delta) \cdot n^2)$ time. We deduce that the running time to compute the entries of $\mathcal{B}_x$ is

$$O\left(\mathsf{s\text{-}nec}_d(T, \delta)^3 \log(\mathsf{s\text{-}nec}_d(T, \delta)) \cdot n^2 + \sum_{(R,R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}} |\mathcal{B}_x[R, R']| \cdot n^2\right).$$

Observe that for each $(R, R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$, by Theorem 3.3, the running time to compute $\mathsf{reduce}(\mathcal{B}_x[R, R'])$ from $\mathcal{B}_x[R, R']$ is $O(|\mathcal{B}_x[R, R']| \cdot \mathsf{s\text{-}nec}_1(T, \delta)^{2(\omega-1)} \cdot n^2)$. Thus, the total running time to compute the table $\mathcal{D}_x$ from the table $\mathcal{B}_x$ is

$$O\left(\sum_{(R,R')\in \mathcal{R}^d_{V_x}\times \mathcal{R}^d_{\overline{V_x}}} |\mathcal{B}_x[R,R']| \cdot \log(\mathsf{s\text{-}nec}_d(T,\delta)) \cdot \mathsf{s\text{-}nec}_1(T,\delta)^{2(\omega-1)} \cdot n^2 \right). \tag{1}$$

For each (A,A') and (B,B'), the size of $\mathcal{D}_a[A,A'] \bigotimes \mathcal{D}_b[B,B']$ is at most $|\mathcal{D}_a[A,A']|\cdot|\mathcal{D}_b[B,B']| \leq \mathsf{s\text{-}nec}_1(T,\delta)^4$. Since there are at most $\mathsf{s\text{-}nec}_d(T,\delta)^3$ pairs d-(R,R')-compatible, we can conclude that

$$\sum_{(R,R')\in \mathcal{R}^d_{V_x}\times \mathcal{R}^d_{\overline{V_x}}} |\mathcal{B}_x[R,R']| \leq \mathsf{s\text{-}nec}_d(T,\delta)^3 \cdot \mathsf{s\text{-}nec}_1(T,\delta)^4.$$

From Equation (1), we deduce that the entries of $\mathcal{D}_x$ are computable in time

$$O(\mathsf{s\text{-}nec}_d(T,\delta)^3 \cdot \mathsf{s\text{-}nec}_1(T,\delta)^{2(\omega+1)} \cdot \log(\mathsf{s\text{-}nec}_d(T,\delta)) \cdot n^2).$$

Since T has $2n-1$ nodes, the running time of our algorithm is $O(\mathsf{s\text{-}nec}_d(T,\delta)^3 \cdot \mathsf{s\text{-}nec}_1(T,\delta)^{2(\omega+1)} \cdot \log(\mathsf{s\text{-}nec}_d(T,\delta)) \cdot n^3)$. ☐

As a corollary, we can solve in time $\mathsf{s\text{-}nec}_1(T,\delta)^{(2\omega+5)} \cdot \log(\mathsf{s\text{-}nec}_1(T,\delta)) \cdot n^3$ the Node-Weighted Steiner Tree problem that asks, given a subset of vertices $K \subseteq V(G)$ called *terminals*, a subset T of minimal weight such that $K \subseteq T \subseteq V(G)$ and $G[T]$ is connected.

Corollary 4.6. *There exists an algorithm that, given an n-vertex graph G, a subset $K \subseteq V(G)$, and a rooted layout (T,δ) of G, computes a minimum node-weighted Steiner tree for (G,K) in time $O(\mathsf{s\text{-}nec}_1(T,\delta)^{(2\omega+5)} \cdot \log(\mathsf{s\text{-}nec}_1(T,\delta)) \cdot n^3)$.*

Proof. Observe that a Steiner tree is a minimum connected $(\mathbb{N},\mathbb{N})$-dominating set of G that contains K. Thus, it is sufficient to change the definition of the table $\mathcal{A}_x$ as follows. Let $x \in V(T)$. For all $(R,R') \in \mathcal{R}^1_{V_x} \times \mathcal{R}^1_{\overline{V_x}}$, we define $\mathcal{A}_x[R,R'] \subseteq V_x$ as follows

$$\mathcal{A}_x[R,R'] := \{X \subseteq V_x \,:\, X \equiv^d_{V_x} R,\ K \cap V_x \subseteq X, \text{ and } X \cup R' \ (\mathbb{N},\mathbb{N})\text{-dominates } V_x\}.$$

Notice that this modification will just modify the way we compute the table $\mathcal{D}_x$ when x is a leaf of T associated with a vertex in K. With this definition of $\mathcal{A}_x$ and by Definition 3.1 of (x,R')-representativity, if G contains a Steiner tree, then $\mathcal{D}_r[\emptyset,\emptyset]$ contains a minimum Steiner tree of G. The running time comes from the running time of Theorem 4.5 with $d = 1$. ☐

With few modifications, we can easily deduce an algorithm to compute a maximum (or minimum) connected co-(σ,ρ)-dominating set.

Corollary 4.7. *There exists an algorithm that, given an n-vertex graph G and a rooted layout (T,δ) of G, computes a maximum (or minimum) connected co-(σ,ρ)-dominating set in time $O(\mathsf{s\text{-}nec}_d(T,\delta)^3 \cdot \mathsf{s\text{-}nec}_1(T,\delta)^{(2\omega+5)} \cdot \log(\mathsf{s\text{-}nec}_d(T,\delta)) \cdot n^3)$ with $d := \max\{1, d(\sigma), d(\rho)\}$.*

Proof. To find a maximum (or minimum) co-(σ,ρ)-dominating set, we need to modify the definition of the table $\mathcal{A}_x$, the invariant and the computational steps of the algorithm from Theorem 4.5. For each vertex $x \in V(T)$, we define the set of indices of our table $\mathcal{D}_x$ as $\mathbb{I}_x := \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}} \times \mathcal{R}^1_{V_x} \times \mathcal{R}^1_{\overline{V_x}}$.

For all $(R,R',\overline{R},\overline{R}') \in \mathbb{I}_x$, we define $\mathcal{A}_x[R,R',\overline{R},\overline{R}'] \subseteq 2^{V_x}$ as the following set

$$\{X \subseteq V_x \,:\, X \equiv^1_{V_x} \overline{R},\ (V_x \setminus X) \equiv^d_{V_x} R, \text{ and } (V_x \setminus X) \cup R' \ (\sigma,\rho)\text{-dominates } V_x\}.$$

It is worth noticing that the definition of $\mathcal{A}_x$ does not depend on $\overline{R}'$, it is just more convenient to write the proof this way in order to obtain an algorithm similar to the one from Theorem 4.5.

Similarly to Theorem 4.5, for each node x of $V(T)$, our algorithm will compute a table $\mathcal{D}_x$ that satisfies the following invariant.

Invariant. For every $(R, R', \overline{R}, \overline{R}') \in \mathbb{I}_x$, the set $\mathcal{D}_x[R, R', \overline{R}, \overline{R}']$ is a subset of $\mathcal{A}_x[R, R', \overline{R}, \overline{R}']$ of size at most $\mathsf{s\text{-}nec}_1(T, \delta)^2$ that $(x, \overline{R}')$-represents $\mathcal{A}_x[R, R', \overline{R}, \overline{R}']$.

Intuitively, we use $\overline{R}$ and $\overline{R}'$ for the connectivity of the co-(σ, ρ)-dominating set and R and R' for the (σ, ρ)-domination.

The following claim adapts Lemma 4.4 to the co-(σ, ρ)-dominating set case.

Claim 4.7.1. *Let x be an internal node of T with a and b as children. For all $(R, R', \overline{R}, \overline{R}') \in \mathbb{I}_x$, we have*

$$\mathcal{A}_x[R, R', \overline{R}, \overline{R}'] := \bigcup_{\substack{(A,A'),\ (B,B')\ d\text{-}(R,R')\text{-compatible} \\ (\overline{A},\overline{A'}),\ (\overline{B},\overline{B'})\ 1\text{-}(\overline{R},\overline{R}')\text{-compatible}}} \mathcal{A}_a[A, A', \overline{A}, \overline{A'}] \bigotimes \mathcal{A}_b[B, B', \overline{A}, \overline{A'}].$$

The proof of this claim follows from the proof of Lemma 4.4. With these modifications, it is straightforward to check that the algorithm of Theorem 4.5 can be adapted straightforwardly to compute a minimum or maximum connected co-(σ, ρ)-dominating set of $V(G)$. With the same analysis as in Theorem 4.5, one easily deduces that the running time of this modified algorithm is $O(\mathsf{s\text{-}nec}_d(T, \delta)^3 \cdot \mathsf{s\text{-}nec}_1(T, \delta)^{(2\omega+5)} \cdot \log(\mathsf{s\text{-}nec}_d(T, \delta)) \cdot n^3)$. □

5. Acyclic variants of (Connected) (σ, ρ)-Dominating Set

The acyclic variant of a Connected (σ, ρ)-Dominating Set (resp. (σ, ρ)-Dominating Set) problem, consists in finding a subset $T \subseteq V(G)$ of maximum (or minimum) weight such that T is a (σ, ρ)-dominating set of G and $G[T]$ is a tree (resp. a forest). We call AC-(σ, ρ)-Dominating Set (resp. Acyclic (σ, ρ)-Dominating Set) the family of problems that are the acyclic variant of a Connected (σ, ρ)-Dominating Set (resp. (σ, ρ)-Dominating Set) problem. See Table 3 for some examples of AC-(σ, ρ)-Dominating Set problems and Acyclic (σ, ρ)-Dominating Set problems. Observe that Maximum Induced Forest is equivalent to the problem Feedback Vertex Set as the complementary of a minimum feedback vertex set is a maximum induced forest.

σ	ρ	d	Version	Standard name
$\mathbb{N}$	$\mathbb{N}$	2	AC	Maximum Induced Tree
$\mathbb{N}$	$\mathbb{N}$	2	Acyclic	Maximum Induced Forest
$\{1, 2\}$	$\mathbb{N}$	3	AC	Longest Induced Path
$\{1, 2\}$	$\mathbb{N}$	3	Acyclic	Maximum Induced Linear Forest

Table 3. Examples of AC-(σ, ρ)-Dominating Set problems and Acyclic (σ, ρ)-Dominating Set problems. To solve these problems, we use the d-neighbor equivalence with $d := \max\{2, d(\sigma), d(\rho)\}$. Column d shows the value of d for each problem.

In this section, we present an algorithm that solves any AC-(σ, ρ)-Dominating Set problem. Unfortunately, we were not able to obtain an algorithm whose running time is polynomial in n and the d-neighbor-width of the given layout (for some constant d). But, for the other parameters, by using their respective properties, we get the running time presented in Table 4 which are roughly the same as those in the previous section. Moreover, we show, via a polynomial reduction, that we can use our algorithm for AC-(σ, ρ)-Dominating Set problems (with some modifications) to solve any Acyclic (σ, ρ)-Dominating Set problem.

Let us explain why we cannot use the same trick as in [4] to ensure the acyclicity, that is classifying the partial solutions X – associated with a node $x \in V(T)$ – with respect to $|X|$ and $|E(G[X])|$. Indeed, for two sets $X, W \subseteq V_x$ with $|X| = |W|$ and $|E(G[X])| = |E(G[W])|$, we have $|E(G[X \cup Y])| = |E(G[W \cup Y])|$ for all $Y \subseteq \overline{V_x}$ if and only if $X \equiv^n_{V_x} W$. Hence, the trick used in

Parameter	Running time
Neighbor-width	$O(\mathsf{s\text{-}nec}_d(\mathcal{L})^3 \cdot \mathsf{s\text{-}nec}_1(\mathcal{L})^{2(\log_2(\mathsf{s\text{-}nec}_1(\mathcal{L}))+\omega+1)} \cdot \log(\mathsf{s\text{-}nec}_2(\mathcal{L})) \cdot n^4)$
Mim-width	$O(n^{(2\omega+3d+4)\mathsf{mim}(\mathcal{L})+4} \cdot \mathsf{mim}(\mathcal{L}))$
Module-width	$O((d+1)^{3\mathsf{mw}(\mathcal{L})} \cdot 2^{(2\omega+3)\mathsf{mw}(\mathcal{L})} \cdot \mathsf{mw}(\mathcal{L}) \cdot n^4)$
Rank-width	$O(2^{(2\omega+3d+4)\mathsf{rw}(\mathcal{L})^2} \cdot \mathsf{rw}(\mathcal{L}) \cdot n^4)$
$\mathbb{Q}$-rank-width	$O(2^{(2\omega+5)\mathsf{rw}_{\mathbb{Q}}(\mathcal{L})\log_2(d\cdot \mathsf{rw}_{\mathbb{Q}}(\mathcal{L}))} \cdot \mathsf{rw}_{\mathbb{Q}}(\mathcal{L}) \cdot n^4)$

TABLE 4. Upper bounds on the running time of our algorithms for an AC-(σ,ρ)-DOMINATING SET problem with $\mathcal{L}=(T,\delta)$ and $d := \max\{2, d(\sigma), d(\rho)\}$.

[4] would imply to classify the partial solutions with respect to their n-neighbor equivalence class. But, we do not have good upper bounds on $\mathsf{nec}_n(V_x)$ with respect to clique-width, ($\mathbb{Q}$-)rank-width and mim-width. In fact, one can construct an n-vertex bipartite graph $H[A,\overline{A}]$ where $\mathsf{nec}_n(A) \in (n/\mathsf{mw}(A))^{\Omega(\mathsf{mw}(A))}$. Since both $\mathsf{rw}(A)$ and $\mathsf{rw}_{\mathbb{Q}}(A)$ are upper-bounded by $\mathsf{mw}(A)$, we deduce that using the trick of [4] would give, for each $f \in \{\mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}\}$, an $n^{\Omega(f(T,\delta))}$ time algorithm.

The rest of this section is organized as follows. We start by defining a new notion of representativity for connected and acyclic problems. Then, we explain the ideas of our algorithm with a concrete example: MAXIMUM INDUCED TREE. Subsequently, we introduced some new concepts that extends the framework designed in Section 3 in order to manage acyclicity. Finally, we present the algorithms for the AC-(σ,ρ)-DOMINATING SET problems and the algorithms for ACYCLIC (σ,ρ)-DOMINATING SET problems.

In order to manage the acyclicity, we need to extend the framework of Section 3. In particular, we define a new notion of representativity which is defined *w.r.t.* to the 2-neighbor equivalence class of a set $R' \subseteq \overline{V_x}$. We consider 2-neighbor equivalence classes instead of 1-neighbor equivalence classes in order to manage the acyclicity (see the explanations after the following definition). Similarly to Section 3, every concept introduced in this section is defined with respect to a node x of T and a set $R' \subseteq \overline{V_x}$. To simplify this section, we fix a node x of T and $R' \subseteq \overline{V_x}$. In our algorithm, R' will always belong to $\mathcal{R}^d_{\overline{V_x}}$ for some $d \in \mathbb{N}^+$ with $d \geq 2$. For MAXIMUM INDUCED TREE $d = 2$ is enough and in general, we use $d := \max\{2, d(\sigma), d(\rho)\}$.

The following definition extends Definition 3.1 of Section 3 to deal with the acyclicity.

Definition 5.1 ($(x,R')^{\mathsf{acy}}$-representativity)**.** *For every $\mathcal{A} \subseteq 2^{V(G)}$ and $Y \subseteq V(G)$, we define*

$$\mathsf{best}(\mathcal{A}, Y)^{\mathsf{acy}} := \mathsf{opt}\{\mathsf{w}(X) \,:\, X \in \mathcal{A} \text{ and } G[X \cup Y] \text{ is a tree}\}.$$

Let $\mathcal{A},\mathcal{B} \subseteq 2^{V_x}$. We say that $\mathcal{B}$ $(x,R')^{\mathsf{acy}}$-represents $\mathcal{A}$ if for every $Y \subseteq \overline{V_x}$ such that $Y \equiv^2_{\overline{V_x}} R'$, we have $\mathsf{best}^{\mathsf{acy}}(\mathcal{A}, Y) = \mathsf{best}^{\mathsf{acy}}(\mathcal{B}, Y)$.

Let us explain the ideas of our algorithm with a concrete example: MAXIMUM INDUCED TREE. To solve this problem, we design an algorithm similar to those of Section 4. For each $(R,R') \in \mathcal{R}^2_{V_x} \times \mathcal{R}^2_{\overline{V_x}}$, this algorithm will compute a set $\mathcal{D}_x[R,R'] \subseteq 2^{V_x}$ that is an $(x,R')^{\mathsf{acy}}$-representative set of small size of the set $\mathcal{A}_x[R]$ of all partial solutions: the sets $X \subseteq V_x$ such that $X \equiv^2_{V_x} R$. This is sufficient to compute a maximum induced tree of G since we have $\mathcal{A}_r[\emptyset] = 2^{V(G)}$ with r the root of T. Thus, by Definition 5.1, any $(r,\emptyset)^{\mathsf{acy}}$-representative set of $\mathcal{A}_r[\emptyset]$ contains a maximum induced tree.

The key to compute the tables of our algorithm is a function that, given $\mathcal{A} \subseteq 2^{V_x}$, computes a small subset of $\mathcal{A}$ that $(x,R')^{\mathsf{acy}}$-represents $\mathcal{A}$. This function starts by removing from $\mathcal{A}$ some sets that will never give a tree with a set $Y \equiv^2_{\overline{V_x}} R'$. For doing so, we characterize the sets $X \in \mathcal{A}$ such that $G[X \cup Y]$ is a tree for some $Y \equiv^2_{\overline{V_x}} R'$. We call these sets R'-important. We show that every R'-important set X satisfies the following conditions:

- for every pair (a, b) of distinct vertices of X such that a and b have at least two neighbors in R', we have $N(a) \cap \overline{V_x} \neq N(b) \cap \overline{V_x}$,
- the number of vertices in X having at least two neighbors in R' is upper bounded by $2\mathsf{mim}(V_x)$.

In order to prove these two necessary conditions, we need the properties of the 2-neighbor equivalence relation. More precisely, we use the fact that, for all $X \subseteq V_x$ and $Y \equiv^2_{\overline{V_x}} R'$, the set of vertices in X having at least two neighbors in Y corresponds to the set of vertices in X having at least two neighbors in R'. By removing from $\mathcal{A}$ the sets that do not respect the two above properties, we are able to decompose $\mathcal{A}$ into a small number of sets $\mathcal{A}_1, \ldots, \mathcal{A}_t$ such that an (x, R')-representative set of $\mathcal{A}_i$ is an $(x, R')^{\mathsf{acy}}$-representative set of $\mathcal{A}_i$ for each $i \in \{1, \ldots, t\}$. We find an $(x, R')^{\mathsf{acy}}$-representative set of $\mathcal{A}$, by computing an (x, R')-representative set $\mathcal{B}_i$ for each $\mathcal{A}_i$ with the function reduce. This is sufficient because $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_t$ is an $(x, R')^{\mathsf{acy}}$-representative set of $\mathcal{A}$.

The following definition characterizes the sets $\mathcal{A} \subseteq 2^{V_x}$ for which an (x, R')-representative set is also an $(x, R')^{\mathsf{acy}}$-representative set.

Definition 5.2. *We say that $\mathcal{A} \subseteq 2^{V_x}$ is R'-consistent, if for each $Y \subseteq \overline{V_x}$ satisfying*

(1) $Y \equiv^2_{\overline{V_x}} R'$

(2) there exists $W \in \mathcal{A}$ such that $G[W \cup Y]$ is a tree

we have that, for each $X \in \mathcal{A}$, either $G[X \cup Y]$ is a tree or $G[X \cup Y]$ is not connected.

The following lemma proves that an (x, R')-representative set of an R'-consistent set is also an $(x, R')^{\mathsf{acy}}$-representative set of this later.

Lemma 5.3. *Let $\mathcal{A} \subseteq 2^{V_x}$. For all $\mathcal{D} \subseteq \mathcal{A}$, if $\mathcal{A}$ is R'-consistent and $\mathcal{D}$ (x, R')-represents $\mathcal{A}$, then $\mathcal{D}$ $(x, R')^{\mathsf{acy}}$-represents $\mathcal{A}$.*

Proof. We assume that $\mathsf{opt} = \max$, the proof for $\mathsf{opt} = \min$ is similar. Let $Y \equiv^2_{\overline{V_x}} R'$. If $\mathsf{best}^{\mathsf{acy}}(\mathcal{A}, Y) = -\infty$, then we also have $\mathsf{best}^{\mathsf{acy}}(\mathcal{D}, Y) = -\infty$ because $\mathcal{D} \subseteq \mathcal{A}$.

Assume now that $\mathsf{best}^{\mathsf{acy}}(\mathcal{A}, Y) \neq -\infty$. Thus, there exists $W \in \mathcal{A}$ such that $G[W \cup Y]$ is a tree. Since $\mathcal{A}$ is R'-consistent, for all $X \in \mathcal{A}$, the graph $G[X \cup Y]$ is either a tree or is not connected. Thus, by Definition 3.1 of best, we have $\mathsf{best}^{\mathsf{acy}}(\mathcal{A}, Y) = \mathsf{best}(\mathcal{A}, Y)$. As $\mathcal{D} \subseteq \mathcal{A}$, we have also $\mathsf{best}^{\mathsf{acy}}(\mathcal{D}, Y) = \mathsf{best}(\mathcal{D}, Y)$. We conclude by observing that if $\mathcal{D}$ (x, R')-represents $\mathcal{A}$, then $\mathsf{best}^{\mathsf{acy}}(\mathcal{D}, Y) = \mathsf{best}^{\mathsf{acy}}(\mathcal{A}, Y)$. □

As for the (x, R')-representativity, we need to prove that the operations we use in our algorithm preserve the $(x, R')^{\mathsf{acy}}$-representativity. The following fact follows from Definition 5.1 of $(x, R')^{\mathsf{acy}}$-representativity. We refer to Definition 3.5 for the notion of d-(R, R')-compatibility.

Fact 5.4. *If $\mathcal{B}$ and $\mathcal{D}$ $(x, R')^{\mathsf{acy}}$-represents, respectively, $\mathcal{A}$ and $\mathcal{C}$, then $\mathcal{B} \cup \mathcal{D}$ $(x, R')^{\mathsf{acy}}$-represents $\mathcal{A} \cup \mathcal{C}$.*

The following lemma is an adaptation of Lemma 3.6 to the notion of $(x, R')^{\mathsf{acy}}$-representativity. The proof is almost the same as the one of Lemma 3.6.

Lemma 5.5. *Let $d \in \mathbb{N}^+$ such that $d \geq 2$. Suppose that x is an internal node of T with a and b as children. Let $R \in \mathcal{R}^d_{V_x}$. Let $(A, A') \in \mathcal{R}^d_{V_a} \times \mathcal{R}^d_{\overline{V_a}}$ and $(B, B') \in \mathcal{R}^d_{V_b} \times \mathcal{R}^d_{\overline{V_b}}$ that are d-(R, R')-compatible. Let $\mathcal{A} \subseteq 2^{V_a}$ such that for all $X \in \mathcal{A}$, we have $X \equiv^d_{V_a} A$, and let $\mathcal{B} \subseteq 2^{V_b}$ such that for all $W \in \mathcal{B}$, we have $W \equiv^d_{V_b} B$.*

If $\mathcal{A}' \subseteq \mathcal{A}$ $(a, A')^{\mathsf{acy}}$-represents $\mathcal{A}$ and $\mathcal{B}' \subseteq \mathcal{B}$ $(b, B')^{\mathsf{acy}}$-represents $\mathcal{B}$, then

$$\mathcal{A}' \bigotimes \mathcal{B}' \ (x, R')^{\mathsf{acy}}\text{-represents } \mathcal{A} \bigotimes \mathcal{B}.$$

Proof. Suppose that $\mathcal{A}' \subseteq \mathcal{A}$ $(a, A')^{\mathsf{acy}}$-represents $\mathcal{A}$ and $\mathcal{B}' \subseteq \mathcal{B}$ $(b, B')^{\mathsf{acy}}$-represents $\mathcal{B}$. In order to prove this lemma, it is sufficient to prove that, for each $Y \equiv^2_{\overline{V_x}} R'$, we have $\mathsf{best}^{\mathsf{acy}}(\mathcal{A}' \bigotimes \mathcal{B}', Y) = \mathsf{best}^{\mathsf{acy}}(\mathcal{A} \bigotimes \mathcal{B}, Y)$.

Let $Y \subseteq \overline{V_x}$ such that $Y \equiv^2_{\overline{V_x}} R'$. We claim the following facts

(a) for every $W \in \mathcal{B}$, we have $W \cup Y \equiv^2_{\overline{V_a}} A'$,

(b) for every $X \in \mathcal{A}$, we have $X \cup Y \equiv^2_{\overline{V_b}} B'$.

Let $W \in \mathcal{B}$. By Fact 2.3, we have that $W \equiv^d_{\overline{V_a}} B$ because $V_b \subseteq \overline{V_a}$ and $W \equiv^d_{V_b} B$. Since $d \geq 2$, we have $W \equiv^2_{\overline{V_a}} B$. By Fact 2.3, we deduce also that $Y \equiv^2_{\overline{V_a}} R'$. Since (A, A') and (B, B') are d-(R, R')-compatible, we have $A' \equiv^d_{\overline{V_a}} R' \cup B$. In particular, we have $A' \equiv^2_{\overline{V_a}} R' \cup B$ because $d \geq 2$. We can conclude that $W \cup Y \equiv^2_{\overline{V_a}} A'$ for every $W \in \mathcal{B}$. The proof for Fact (b) is symmetric.

Now observe that, by the definitions of $\mathsf{best}^{\mathsf{acy}}$ and of the merging operator $\bigotimes$, we have

$$\mathsf{best}^{\mathsf{acy}}\big(\mathcal{A} \bigotimes \mathcal{B}, Y\big) = \mathsf{opt}\{\mathsf{w}(X) + \mathsf{w}(W) \,:\, X \in \mathcal{A} \wedge W \in \mathcal{B} \wedge G[X \cup W \cup Y] \text{ is a tree}\}.$$

Since $\mathsf{best}^{\mathsf{acy}}(\mathcal{A}, W \cup Y) = \mathsf{opt}\{\mathsf{w}(X) \,:\, X \in \mathcal{A} \wedge G[X \cup W \cup Y] \text{ is a tree}\}$, we deduce that

$$\mathsf{best}^{\mathsf{acy}}\big(\mathcal{A} \bigotimes \mathcal{B}, Y\big) = \mathsf{opt}\{\mathsf{best}^{\mathsf{acy}}(\mathcal{A}, W \cup Y) + \mathsf{w}(W) \,:\, W \in \mathcal{B}\}.$$

Since $\mathcal{A}'$ (a, A')-represents $\mathcal{A}$ and by Fact (a), we have

$$\begin{aligned} \mathsf{best}^{\mathsf{acy}}\big(\mathcal{A} \bigotimes \mathcal{B}, Y\big) &= \mathsf{opt}\{\mathsf{best}^{\mathsf{acy}}(\mathcal{A}', W \cup Y) + \mathsf{w}(W) \,:\, W \in \mathcal{B}\} \\ &= \mathsf{best}^{\mathsf{acy}}\big(\mathcal{A}' \bigotimes \mathcal{B}, Y\big). \end{aligned}$$

Symmetrically, we deduce from Fact (b) that $\mathsf{best}^{\mathsf{acy}}\big(\mathcal{A}' \bigotimes \mathcal{B}, Y\big) = \mathsf{best}^{\mathsf{acy}}\big(\mathcal{A}' \bigotimes \mathcal{B}', Y\big)$. This stands for every $Y \subseteq \overline{V_x}$ such that $Y \equiv^2_{\overline{V_x}} R'$. Thus, we conclude that $\mathcal{A}' \bigotimes \mathcal{B}'$ $(x, R')^{\mathsf{acy}}$-represents $\mathcal{A} \bigotimes \mathcal{B}$. □

In order to decompose a set $\mathcal{A} \subseteq 2^{V_x}$ into a small number of R'-consistent sets, we need to remove unimportant partial solutions from $\mathcal{A}$. By unimportant, we mean the partial solutions in $\mathcal{A}$ that would never induce a tree with a set equivalent to R' *w.r.t.* $\equiv^2_{\overline{V_x}}$. The following gives a formal definition of these important and unimportant partial solutions.

Definition 5.6 (R'-important)**.** *We say that $X \subseteq V_x$ is R'-important if there exists $Y \subseteq \overline{V_x}$ such that $Y \equiv^2_{\overline{V_x}} R'$ and $G[X \cup Y]$ is a tree, otherwise, we say that X is R'-unimportant.*

By definition, any set obtained from a set $\mathcal{A}$ by removing R'-unimportant sets is an $(x, R')^{\mathsf{acy}}$-representative set of $\mathcal{A}$. The following lemma gives some necessary conditions on R'-important sets. It follows that any set which does not respect one of these conditions can safely be removed from $\mathcal{A}$. These conditions are the key to obtain the running times of Table 4. At this point, we need to introduce the following notations. For every $X \subseteq V_x$, we define $X^0 := \{v \in X \,:\, N(v) \cap R' = \emptyset\}$, $X^1 := \{v \in X \,:\, |N(v) \cap R'| = 1\}$, and $X^{2+} := \{v \in X \,:\, |N(v) \cap R'| \geq 2\}$. Notice that for every $Y \equiv^2_{\overline{V_x}} R'$ and $X \subseteq V_x$, the vertices in X^0 have no neighbor in Y, those in X^1 have exactly one neighbor in Y and those in X^{2+} have at least 2 neighbors in Y. We remind the reader that $R' \subseteq \overline{V_x}$.

Lemma 5.7. *If $X \subseteq V_x$ is R'-important, then $G[X]$ is a forest and the following properties are satisfied:*

(1) for every pair of distinct vertices a and b in X^{2+}, we have $N(a) \cap \overline{V_x} \neq N(b) \cap \overline{V_x}$,

(2) $|X^{2+}|$ is upper bounded by $2\mathsf{mim}(V_x)$, $2\mathsf{rw}(V_x)$, $2\mathsf{rw}_{\mathbb{Q}}(V_x)$, and $2\log_2(\mathsf{nec}_1(V_x))$.

Proof. Obviously, any R'-important set must induce a forest.

Let $X \subseteq V_x$ be an R'-important set. Since X is R'-important, there exists $Y \subseteq \overline{V_x}$ such that $Y \equiv^2_{\overline{V_x}} R'$ and $G[X \cup Y]$ is a tree.

Assume towards a contradiction that there exist two distinct vertices a and b in X^{2+} such that $N(a) \cap \overline{V_x} = N(b) \cap \overline{V_x}$. Since a and b belong to X^{2+} and $Y \equiv^2_{\overline{V_x}} R'$, both a and b have at least two neighbors in Y. Thus, a and b have at least two common neighbors in Y. We conclude that $G[X \cup Y]$ admits a cycle of length four, yielding a contradiction. We conclude that Property (1) holds for every R'-important set.

Now, we prove that Property (2) holds for X. Observe that, by Lemma 2.6, $\mathsf{mim}(V_x)$ is upper bounded by $\mathsf{rw}(V_x)$, $\mathsf{rw}_{\mathbb{Q}}(V_x)$, and $\log_2(\mathsf{nec}_1(V_x))$. Thus, in order to prove Property (2), it is sufficient to prove that $|X^{2+}| \leq 2\mathsf{mim}(V_x)$.

We claim that $|X^{2+}| \leq 2k$ where k is the size of a maximum induced matching of $F := G[X^{2+}, Y]$. Since F is an induced subgraph of $G[V_x, \overline{V_x}]$, we have $k \leq \mathsf{mim}(V_x)$ and this is enough to prove Property (2). Notice that F is a forest because F is a subgraph of $G[X \cup Y]$, which is a tree.

We say that an induced subforest F' of F admits a *good bipartition* if there exists a bipartition $\{X_1, X_2\}$ of $X^{2+} \cap V(F')$ such that for each $i \in \{1, 2\}$ and for each $v \in X_i$, there exists $y \in Y \cap V(F')$ such that $N_F(y) \cap X_i = \{v\}$.

In the following, we prove that F admits a *good bipartition*. Observe that this is enough to prove Property (2) since if F admits a good bipartition $\{X_1, X_2\}$, then $|X_1| \leq k$ and $|X_2| \leq k$.

In order to prove that F admits a good bipartition it is sufficient to prove that each connected component of F admits a good bipartition. Let $C \in \mathsf{cc}(F)$ and $x \in C \cap X^{2+}$. As F is a forest, $F[C]$ is a tree. Observe that the distance in F between each vertex $v \in C \cap X^{2+}$ and $x \in F$ is even because $F := G[X^{2+}, Y]$. Let C_1 (resp. C_2) be the set of all vertices $v \in C \cap X^{2+}$ such that there exists an odd (resp. even) integer $i \in \mathbb{N}$ so that the distance between v and x in F is $2i$. We claim that $\{C_1, C_2\}$ is a good bipartition of $F[C]$. Let $v \in C_1$ and $i \in \mathbb{N}$ such that the distance between c and x in F is $2i$. We want to prove that there exists $y \in Y$ such that $N_F(y) \cap C_1 = \{v\}$. Let $P := \{v \in C_1 : N_F(v) \cap N_F(v') \neq \emptyset\}$. If a vertex $v' \in C \cap X^{2+}$ has a common neighbor with v in F, then the distance between v' and x in F is either $2i - 2$, $2i$ or $2i + 2$. By construction of C_1 and C_2, every vertex at distance $2i - 2$ and $2i + 2$ from x must belong to C_2 because v belongs to C_1 and v is at distance $2i$ from x. Thus, every vertex in P is at distance $2i$ from x. As $F[C]$ is a tree, v has only one neighbor y at distance $2i - 1$ from x in F. Because $F[C]$ is a tree, we deduce that $N_F(P) \cap N_F(v) = \{y\}$. Since $v \in X^{2+}$, v has at least two neighbors in Y (because $Y \equiv^2_{V_x} R'$), thus there exists $y' \in Y$ such that $y \neq y'$. By definition of P, we have $N_F(y') \cap C_1 = \{v\}$. Symmetrically, we can prove that for every vertex $v \in C_2$, there exits $y \in V(F[C]) \cap Y$ such that $N_F(y) \cap C_2 = \{v\}$. Hence, we deduce that $\{C_1, C_2\}$ is a good bipartition of $F[C]$.

We deduce that every connected component of F admits a good bipartition and thus F admits a good bipartition. Thus, $|X^{2+}| \leq 2\mathsf{mim}(V_x)$. ☐

The next lemma proves that, for each $f \in \{\mathsf{nec}_1, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{mim}\}$, we can decompose a set $\mathcal{A} \subseteq 2^{V_x}$ into a small collection $\{\mathcal{A}_1, \ldots, \mathcal{A}_t\}$ of pairwise disjoint subsets of $\mathcal{A}$ such that each $\mathcal{A}_i$ is R'-consistent. Even though some parts of the proof are specific to each parameter, the ideas are roughly the same. Intuitively, we compute this decomposition by removing the sets X in $\mathcal{A}$ that do not induce a forest. If $f = \mathsf{mw}$, we remove the sets in $\mathcal{A}$ that do not respect Condition (1) of Lemma 5.7, otherwise, we remove the sets that do not respect the upper bound associated with f from Condition (2) of Lemma 5.7. These sets can be safely removed as, by Lemma 5.7, they are R'-unimportant. After removing these sets, we obtain the decomposition of $\mathcal{A}$ by taking the equivalence classes of some equivalence relation. Owing to the set of R'-unimportant sets we have removed from $\mathcal{A}$, we prove that the number of equivalence classes of this latter equivalence relation respects the upper bound associated with f that is described in Table 5.

Observe that, for $f \in \{\mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{nec}_1\}$, it is tempting to remove the sets $X \in \mathcal{A}$ such that $|X^{2+}| > 2\mathsf{mim}(V_x)$ since $\mathsf{mim}(V_x)$ is upper bounded by $\mathsf{rw}(V_x)$, $\mathsf{rw}_{\mathbb{Q}}(V_x)$, and $\log_2(\mathsf{nec}_1(V_x))$. Doing so implies to compute $\mathsf{mim}(V_x)$. But, if $f \neq \mathsf{mim}$, then this is not necessary and can drastically increase the running time to compute the decomposition of $\mathcal{A}$.

Lemma 5.8. *Let $\mathcal{A} \subseteq 2^{V_x}$. For each $f \in \{\mathsf{nec}_1, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{mim}\}$, there exists a collection $\{\mathcal{A}_1, \ldots, \mathcal{A}_t\}$ of pairwise disjoint subsets of $\mathcal{A}$ computable in time $O(|\mathcal{A}| \cdot \mathcal{N}_f(T, \delta) \cdot n^2)$ such that*

- *$\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_t$ $(x, R')^{\mathsf{acy}}$-represents $\mathcal{A}$,*

- $\mathcal{A}_i$ *is* R'*-consistent for each* $i \in \{1, \dots, t\}$ *and*
- $t \leq \mathcal{N}_f(T, \delta)$,

where $\mathcal{N}_f(T, \delta)$ *is the term defined in Table 5.*

f	$\mathcal{N}_f(T, \delta)$
nec_1	$\mathsf{s\text{-}nec}_1(T,\delta)^{2\log_2(\mathsf{s\text{-}nec}_1(T,\delta))} \cdot 2n$
mim	$2n^{2\mathsf{mim}(T,\delta)+1}$
mw	$2^{\mathsf{mw}(T,\delta)} \cdot 2n$
rw	$2^{2\mathsf{rw}(T,\delta)^2} \cdot 2n$
$\mathsf{rw}_{\mathbb{Q}}$	$2^{\mathsf{rw}_{\mathbb{Q}}(T,\delta)\log_2(2\mathsf{rw}_{\mathbb{Q}}(T,\delta)+1)} \cdot 2n$

TABLE 5. Upper bounds $\mathcal{N}_f(T, \delta)$ on the cardinal of the decomposition of Lemma 5.8 for each $f \in \{\mathsf{nec}_1, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{mim}\}$.

Proof. We begin by defining an equivalence relation $\sim$ on 2^{V_x} such that each equivalence class of $\sim$ over 2^{V_x} is an R'-consistent set.

For $X \subseteq V_x$, let $\sigma(X)$ be the vector corresponding to the sum (over $\mathbb{Q}$) of the row vectors of $M_{V_x,\overline{V_x}}$ corresponding to X. Notice that if $\sigma(X) = \sigma(X')$, then $X \equiv^d_{V_x} X'$ for all $d \in \mathbb{N}^+$, because the entries of $\sigma(X)$ represent the number of neighbors in X for each vertex in $\overline{V_x}$. Moreover, if $\sigma(X) = \sigma(X')$, then $|E(G[X, Y])| = |E(G[X', Y])|$ for every $Y \subseteq \overline{V_x}$.

We define the equivalence relation $\sim$ on 2^{V_x} such that $X \sim W$ if we have $\sigma(X^{2+}) = \sigma(W^{2+})$ and $|E(G[X])| - |X \setminus X^1| = |E(G[W])| - |W \setminus W^1|$.

Intuitively, if $X \sim W$, then for all $Y \equiv^2_{\overline{V_x}} R'$, we have $|E(G[X \cup Y])| = |X \cup Y| - 1$ if and only if $|E(G[W \cup Y])| = |W \cup Y| - 1$. Thus, if $X \sim W$ and both sets induce with Y a connected graph, then both sets induce with Y a tree (because any connected graph with t vertices and $t - 1$ edges is a tree). Consequently, an equivalence class of $\sim$ is an R'-consistent set.

Claim 5.8.1. *Let* $\mathcal{A}' \subseteq \mathcal{A}$*. If, for all* $X, W \in \mathcal{A}'$*, we have* $X \sim W$*, then* $\mathcal{A}'$ *is* R'*-consistent.*

Proof. Suppose that $X \sim W$ for all $X, W \in \mathcal{A}'$. In order to prove that $\mathcal{A}'$ is R'-consistent, it is enough to prove that, for each $X, W \in \mathcal{A}'$ and $Y \equiv^2_{\overline{V_x}} R'$, if $G[X \cup Y]$ is a tree and $G[W \cup Y]$ is connected, then $G[W \cup Y]$ is a tree.

Let $Y \equiv^2_{\overline{V_x}} R'$ and $X, W \in \mathcal{A}'$. Assume that $G[X \cup Y]$ is a tree and that $G[W \cup Y]$ is connected. Observe that $G[X \cup Y]$ is a tree if and only if $G[X \cup Y]$ is connected and $|E(G[X \cup Y])| = |X \cup Y| - 1$.

Since the vertices in X^0 have no neighbors in Y, we can decompose $|E(G[X \cup Y])| = |X \cup Y| - 1$ to obtain the following equation

$$|E(G[Y])| + |E(G[X^{2+}, Y])| + |E(G[X^1, Y])| + |E(G[X])| = |X \cup Y| - 1. \tag{2}$$

Since every vertex in X^1 has exactly one neighbor in Y (because $Y \equiv^2_{\overline{V_x}} R'$) , we have $|E(G[X^1, Y])| = |X^1|$. Thus, Equation (1) is equivalent to

$$|E(G[Y])| + |E(G[X^{2+}, Y])| + |E(G[X])| = |X \setminus X^1| + |Y| - 1. \tag{3}$$

Since $X \sim W$, we have $|E(G[X])| - |X \setminus X^1| = |E(G[W])| - |W \setminus W^1|$. Moreover, owing to $\sigma(X^{2+}) = \sigma(W^{2+})$, we have $|E(G(X^{2+}, Y))| = |E(G(W^{2+}, Y))|$. We conclude that Equation (2) is equivalent to

$$|E(G[Y])| + |E(G[W^{2+}, Y])| + |E(G[W])| = |W \setminus W^1| + |Y| - 1. \tag{4}$$

With the same arguments to prove that (3) is equivalent to $|E(G[X \cup Y])| = |X \cup Y| - 1$, we can show that (3) is equivalent to $|E(G(W \cup Y))| = |W \cup Y| - 1$. By assumption, $G[W \cup Y]$ is connected and thus we conclude that $G[W \cup Y]$ is a tree. $\square$

We are now ready to decompose $\mathcal{A}$. We start by removing from $\mathcal{A}$ all the sets that do not induce a forest. Trivially, this can be done in time $O(|\mathcal{A}| \cdot n)$. Moreover, these sets are R'-unimportant and thus we keep an $(x, R')^{\mathsf{acy}}$-representative set of $\mathcal{A}$. Before explaining how we proceed separately for each parameter, we need the following observation which follows from the removal of all the sets in $\mathcal{A}$ that do not induce a forest.

Observation 5.8.2. *For all $X \in \mathcal{A}$, we have $-n \leq |E(G[X])| - |X \setminus X^1| < n$.*

Concerning module-width. We remove all the sets X in $\mathcal{A}$ that do not respect Condition (1) of Lemma 5.7. By Lemma 5.7, these sets are R'-unimportant and thus we keep an $(x, R')^{\mathsf{acy}}$-representative set of $\mathcal{A}$. After removing these sets, for each $X \in \mathcal{A}$, every pair (a, b) of distinct vertices in X^{2+} have a different neighborhood in $\overline{V_x}$. Observe that, by definition of module-width, we have

- $\mathsf{mw}(V_x) = |\{N(v) \cap \overline{V_x} : v \in V_x\}|$ and
- for every $a, b \in V_x$, if $N(a) \cap \overline{V_x} = N(b) \cap \overline{V_x}$, then $\sigma(\{a\}) = \sigma(\{b\})$.

We deduce from these observations that $|\{\sigma(X^{2+}) : X \in \mathcal{A}\}| \leq 2^{\mathsf{mw}(V_x)}$. Thus, the number of equivalence classes of $\sim$ over $\mathcal{A}$ is at most $2^{\mathsf{mw}(V_x)} \cdot 2n \leq \mathcal{N}_{\mathsf{mw}}(T, \delta)$. The factor $2n$ comes from Observation 5.8.2 and appears also in all subsequent upper-bounds.

Concerning mim-width. We remove from $\mathcal{A}$ all the sets X such that $|X^{2+}| > 2\mathsf{mim}(V_x)$. By Lemma 5.7, these sets are R'-unimportant and thus we keep an $(x, R')^{\mathsf{acy}}$-representative set of $\mathcal{A}$. Observe that this can be done in time $O(n^{\mathsf{mim}(V_x)+1} + |\mathcal{A}| \cdot n^2)$ because $\mathsf{mim}(V_x)$ can be computed in time $O(n^{\mathsf{mim}(V_x)} + 1)$. Since $|X^{2+}| \leq 2\mathsf{mim}(V_x)$, for every $X \in \mathcal{A}$, we have $|\{\sigma(X^{2+}) : X \in \mathcal{A}\}| \leq n^{2\mathsf{mim}(V_x)}$.

Hence, the number of equivalence classes of $\sim$ over $\mathcal{A}$ is at most $2n^{2\mathsf{mim}(V_x)+1} \leq \mathcal{N}_{\mathsf{mim}}(T, \delta)$.

Concerning 1-neighbor-width. We remove all the sets $X \in \mathcal{A}$ such that $|X^{2+}| > 2\log_2(\mathsf{nec}_1(V_x))$. By Lemma 5.7, we keep an $(x, R')^{\mathsf{acy}}$-representative set of $\mathcal{A}$. Since there are at most $\mathsf{nec}_1(V_x)$ different rows in $M_{V_x, \overline{V_x}}$, we deduce that $|\{\sigma(X^{2+}) : X \in \mathcal{A}\}| \leq \mathsf{nec}_1(V_x)^{2\log_2(\mathsf{nec}_1(V_x))}$ values.

Hence, the number of equivalence classes of $\sim$ over $\mathcal{A}$ is at most $\mathsf{nec}_1(V_x)^{2\log_2(\mathsf{nec}_1(V_x))} \cdot 2n \leq \mathcal{N}_{\mathsf{nec}_1}(T, \delta)$.

Concerning rank-width. We remove from $\mathcal{A}$ all the sets X such that $|X^{2+}| > 2\mathsf{rw}(V_x)$ because they are R'-unimportant by Lemma 5.7. We know that there are at most $2^{\mathsf{rw}(V_x)}$ different rows in M. Thus, we have $|\{\sigma(X^{2+}) : X \in \mathcal{A}\}| \leq (2^{\mathsf{rw}(V_x)})^{2\mathsf{rw}(V_x)}$.

We can therefore conclude that the number of equivalence classes of $\sim$ over $\mathcal{A}$ is at most $2^{2\mathsf{rw}(V_x)^2} \cdot 2n \leq \mathcal{N}_{\mathsf{rw}}(T, \delta)$.

Concerning $\mathbb{Q}$-rank-width. We remove all the sets $X \in \mathcal{A}$ such that $|X^{2+}| > 2\mathsf{rw}_{\mathbb{Q}}(V_x)$. By Lemma 5.7, we keep an $(x, R')^{\mathsf{acy}}$-representative set of $\mathcal{A}$. We claim that $|\{\sigma(X^{2+}) : X \in \mathcal{A}\}| \leq 2^{\mathsf{rw}_{\mathbb{Q}}(V_x) \cdot \log_2(2\mathsf{rw}_{\mathbb{Q}}(V_x)+1)}$. Notice that the proof can be deduced from [21, Theorem 4.2].

Let C be a set of $\mathsf{rw}_{\mathbb{Q}}(V_x)$ columns of $M_{V_x, \overline{V_x}}$. Since the rank over $\mathbb{Q}$ of $M_{V_x, \overline{V_x}}$ is $\mathsf{rw}_{\mathbb{Q}}(V_x)$, every linear combination of row vectors of M is completely determined by its entries in C. Since $|X^{2+}| \leq 2\mathsf{rw}_{\mathbb{Q}}(V_x)$ for every $X \in \mathcal{A}$, the values in $\sigma(X^{2+})$ are between 0 and $2\mathsf{rw}_{\mathbb{Q}}(V_x)$. Hence, the number of possible values for $\sigma(X^{2+})$ is at most $(2\mathsf{rw}_{\mathbb{Q}}(V_x)+1)^{\mathsf{rw}_{\mathbb{Q}}(V_x)} = 2^{\mathsf{rw}_{\mathbb{Q}}(V_x)\log_2(2\mathsf{rw}_{\mathbb{Q}}(V_x)+1)}$.

We conclude that the number of equivalence classes of $\sim$ over $\mathcal{A}$ is at most $2^{\mathsf{rw}_{\mathbb{Q}}(V_x)\log_2(2\mathsf{rw}_{\mathbb{Q}}(V_x)+1)} \cdot 2n \leq \mathcal{N}_{\mathsf{rw}_{\mathbb{Q}}}(T, \delta)$.

It remains to prove the running time. Observe that, for module-width, ($\mathbb{Q}$-)rank-width and 1-neighbor-width, the removal of R'-unimportant sets can be done in time $O(|\mathcal{A}| \cdot n^2)$. Indeed, $\mathsf{mw}(V_x), \mathsf{rw}(V_x)$ and $\mathsf{rw}_{\mathbb{Q}}(V_x)$ can be computed in time $O(n^2)$. For 1-neighbor-width, we can assume that the size of $\mathsf{nec}_1(V_x)$ is given because the first step of our algorithm for AC-(σ,ρ)-Dominating Set problems is to compute $\mathcal{R}^d_{V_x}$ for some $d \in \mathbb{N}^+$ and one can easily compute $\mathsf{nec}_1(V_x)$ while computing $\mathcal{R}^d_{V_x}$. Notice that we can decide whether $X \sim W$ in time $O(n^2)$. Therefore, for each $f \in \{\mathsf{nec}_1, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{mim}\}$, we can therefore compute the equivalence classes of $\mathcal{A}$ in time $O(|\mathcal{A}| \cdot \mathcal{N}_f(T,\delta) \cdot n^2)$. $\square$

We are now ready to give an adaptation of Theorem 3.3 to the notion of $(x, R')^{\mathsf{acy}}$-representativity.

Theorem 5.9. *Let $\mathcal{A} \subseteq 2^{V_x}$ such that, for each $X \in \mathcal{A}$, we have $X \equiv^2_{V_x} R$. For each $f \in \{\mathsf{nec}_1, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{mim}\}$, there exists an algorithm $\mathsf{reduce}^{\mathsf{acy}}_f$ that outputs in time $O((\mathsf{nec}_1(V_x)^{2(\omega-1)} + \mathcal{N}_f(T,\delta)) \cdot |\mathcal{A}| \cdot n^2)$, a subset $\mathcal{B} \subseteq \mathcal{A}$ such that $\mathcal{B}$ $(x,R')^{\mathsf{acy}}$-represents $\mathcal{A}$ and $|\mathcal{B}| \leq \mathcal{N}_f(T,\delta) \cdot \mathsf{nec}_1(V_x)^2$.*

Proof. Let $f \in \{\mathsf{nec}_1, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{mim}\}$. By Lemma 5.8, we can compute in time $O(|\mathcal{A}| \cdot \mathcal{N}_f(T,\delta) \cdot n^2)$ a collection $\{\mathcal{A}_1, \ldots, \mathcal{A}_t\}$ of pairwise disjoint subsets of $\mathcal{A}$ such that

- $\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_t$ $(x,R')^{\mathsf{acy}}$-represents $\mathcal{A}$,
- $\mathcal{A}_i$ is R'-consistent for each $i \in \{1,\ldots,t\}$,
- $t \leq \mathcal{N}_f(T,\delta)$.

For each $X \in \mathcal{A}$, we have $X \equiv^1_{V_x} R$ because $X \equiv^2_{V_x} R$. Since $\mathcal{A}_1,\ldots,\mathcal{A}_t \subseteq \mathcal{A}$, we can apply Theorem 3.3 to compute, for each $i \in \{1,\ldots,t\}$, the set $\mathcal{B}_i := \mathsf{reduce}(\mathcal{A}_i)$. By Theorem 3.3, for each $i \in \{1,\ldots,t\}$, the set $\mathcal{B}_i$ is a subset and an (x,R')-representative set of $\mathcal{A}_i$ whose size is bounded by $\mathsf{nec}_1(V_x)^2$. Moreover, as $\mathcal{A}_i$ is R'-consistent, we have $\mathcal{B}_i$ $(x,R')^{\mathsf{acy}}$-represents $\mathcal{A}_i$ by Lemma 5.3.

Let $\mathcal{B} := \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_t$. Since $\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_t$ $(x,R')^{\mathsf{acy}}$-represents $\mathcal{A}$, we deduce from Fact 5.4 that $\mathcal{B}$ $(x,R')^{\mathsf{acy}}$-represents $\mathcal{A}$. Furthermore, we have $|\mathcal{B}| \leq \mathcal{N}_f(T,\delta) \cdot \mathsf{nec}_1(V_x)^2$ owing to $t \leq \mathcal{N}_f(T,\delta)$ and $|\mathcal{B}_i| \leq \mathsf{nec}_1(V_x)^2$ for all $i \in \{1,\ldots,t\}$.

It remains to prove the running time. By Theorem 3.3, we can compute $\mathcal{B}_1,\ldots,\mathcal{B}_t$ in time $O(|\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_t| \cdot \mathsf{nec}_1(V_x)^{2(\omega-1)} \cdot n^2)$. Since the sets $\mathcal{A}_1,\ldots,\mathcal{A}_t$ are subsets of $\mathcal{A}$ and pairwise disjoint, we have $|\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_t| \leq |\mathcal{A}|$. That proves the running time and concludes the theorem. $\square$

We are now ready to present an algorithm that solves any AC-(σ,ρ)-Dominating Set problem. This algorithm follows the same ideas as the algorithm from Theorem 4.5, except that we use $\mathsf{reduce}^{\mathsf{acy}}_f$ instead of reduce.

Theorem 5.10. *For each $f \in \{\mathsf{nec}_1, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{mim}\}$, there exists an algorithm that, given an n-vertex graph G and a rooted layout (T,δ) of G, solves any* AC-(σ,ρ)-Dominating Set *problem, in time*

$$O(\mathsf{s\text{-}nec}_d(T,\delta)^3 \cdot \mathsf{s\text{-}nec}_1(T,\delta)^{2(\omega+1)} \cdot \mathcal{N}_f(T,\delta)^2 \cdot \log(\mathsf{s\text{-}nec}_d(T,\delta)) \cdot n^3),$$

with $d := \{2, d(\sigma), d(\rho)\}$.

Proof. Let $f \in \{\mathsf{nec}_1, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{mim}\}$. If we want to compute a solution of maximum (resp. minimum) weight, then we use the framework of Section 3 with $\mathsf{opt} = \max$ (resp. $\mathsf{opt} = \min$).

The first step of our algorithm is to compute, for each $x \in V(T)$, the sets $\mathcal{R}^d_{V_x}$, $\mathcal{R}^d_{\overline{V_x}}$ and a data structure to compute $\mathsf{rep}^d_{V_x}(X)$ and $\mathsf{rep}^d_{\overline{V_x}}(X)$ for $X \subseteq V_x$ in time $O(\log(\mathsf{s\text{-}nec}_d(T,\delta)) \cdot n^2)$. As T has $2n-1$ nodes, by Lemma 2.4, we can compute these sets and data structures in time $O(\mathsf{s\text{-}nec}_d(T,\delta) \cdot \log(\mathsf{s\text{-}nec}_d(T,\delta)) \cdot n^3)$.

For each node $x \in T$ and for each $(R,R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$, we define $\mathcal{A}_x[R,R'] \subseteq 2^{V_x}$ as follows

$$\mathcal{A}_x[R,R'] := \{X \subseteq V_x \,:\, X \equiv^d_{V_x} R \text{ and } X \cup R' \ (\sigma,\rho)\text{-dominates } V_x\}.$$

We deduce the following claim from the proof of Claim 4.4.

Claim 5.10.1. *For every internal node $x \in V(T)$ with a and b as children and $(R, R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$, we have*

$$\mathcal{A}_x[R, R'] = \bigcup_{(A,A'),\ (B,B')\ d\text{-}(R,R')\text{-compatible}} \mathcal{A}_a[A, A'] \bigotimes \mathcal{A}_b[B, B'].$$

For each node x of $V(T)$, our algorithm will compute a table $\mathcal{D}_x$ that satisfies the following invariant.

Invariant. For every $(R, R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$, the set $\mathcal{D}_x[R, R']$ is a subset of $\mathcal{A}_x[R, R']$ of size at most $\mathcal{N}_f(T, \delta) \cdot \mathsf{nec}_1(V_x)^2$ that $(x, R')^{\mathsf{acy}}$-represents $\mathcal{A}_x[R, R']$.

Notice that by Definition of $(x, R')^{\mathsf{acy}}$-representativity, if the invariant holds for r, then $\mathcal{D}_r[\emptyset, \emptyset]$ contains a set X of maximum (or minimum) weight such that X is a (σ, ρ)-dominating set of G and $G[X]$ is a tree.

The algorithm is a usual bottom-up dynamic programming algorithm and computes for each node x of T the table $\mathcal{D}_x$.

Let x be a leaf of T with $V_x = \{v\}$. Observe that $\mathcal{A}_x[R, R'] \subseteq 2^{V_x} = \{\emptyset, \{v\}\}$. Thus, our algorithm can directly compute $\mathcal{A}_x[R, R']$ and set $\mathcal{D}_x[R, R'] := \mathcal{A}_x[R, R']$. In this case, the invariant trivially holds.

Now, take x an internal node of T with a and b as children such that the invariant holds for a and b. For each $(R, R') \in \mathcal{R}^d_{V_x} \times \mathcal{R}^d_{\overline{V_x}}$, the algorithm computes $\mathcal{D}_x[R, R'] := \mathsf{reduce}^{\mathsf{acy}}_f(\mathcal{B}_x[R, R'])$, where the set $\mathcal{B}_x[R, R']$ is defined as follows

$$\mathcal{B}_x[R, R'] := \bigcup_{(A,A'),\ (B,B')\ d\text{-}(R,R')\text{-compatible}} \mathcal{D}_a[A, A'] \bigotimes \mathcal{D}_b[B, B'].$$

Similarly to the proof of Theorem 4.5, we deduce from Fact 5.4, Lemma 5.5, and Claim 5.10.1, that $\mathcal{D}_x[R, R']$ is a subset and an $(x, R')^{\mathsf{acy}}$-representative set of $\mathcal{A}_x[R, R']$. By Theorem 5.9, we have $|\mathcal{D}_x[R, R']| \leq \mathcal{N}_f(T, \delta) \cdot \mathsf{s\text{-}nec}_1(T, \delta)^2$.

Consequently, the invariant holds for x and by induction, it holds for all the nodes of T. The correctness of the algorithm follows.

Running Time. The running time of our algorithm is almost the same as the running time given in Theorem 5.10. The only difference is the factor $\mathcal{N}_f(T, \delta)^2$ which is due to the following fact: by the invariant condition, for each (A, A') and (B, B'), the size of $\mathcal{D}_a[A, A'] \bigotimes \mathcal{D}_b[B, B']$ is at most $\mathcal{N}_f(T, \delta)^2 \cdot \mathsf{s\text{-}nec}_1(T, \delta)^4$. ☐

The following theorem shows how to use the algorithm for AC-(σ, ρ)-Dominating Set problems in order to solve any Acyclic (σ, ρ)-Dominating Set problem.

Theorem 5.11. *For each $f \in \{\mathsf{nec}_1, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{mim}\}$, there exists an algorithm that, given an n-vertex graph G and a rooted layout (T, δ) of G, solves any* Acyclic (σ, ρ)-Dominating Set *problem in time*

$$O(\mathsf{s\text{-}nec}_2(T, \delta)^3 \cdot \mathsf{s\text{-}nec}_1(T, \delta)^{2(\omega+1)} \cdot \mathcal{N}_f(T, \delta)^{O(1)} \cdot n^3),$$

with $d := \max\{2, d(\sigma), d(\rho)\}$.

Proof. Let $f \in \{\mathsf{nec}_1, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}, \mathsf{mim}\}$. Suppose that we want to compute a maximum acyclic (σ, ρ)-dominating set. The proof for computing a minimum acyclic (σ, ρ)-dominating set is symmetric.

The first step of this proof is to construct a $2n + 1$-vertex graph G' from G and a layout $(T^\star, \delta^\star)$ of G' from (T, δ) in time $O(n^2)$ such that $(T^\star, \delta^\star)$ respect the following inequalities:

(1) for every $d \in \mathbb{N}^+$, $\mathsf{s\text{-}nec}_d(T^\star, \delta^\star) \leq (d + 1) \cdot \mathsf{s\text{-}nec}_d(T, \delta)$,
(2) for every $f \in \{\mathsf{mim}, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}\}$, $f(T^\star, \delta^\star) \leq f(T, \delta) + 1$.

The second step of this proof consists in showing how the algorithm of Theorem 5.10 can be modified to find a maximum acyclic (σ, ρ)-dominating set of G by running this modified algorithm on G' and $(T^\star, \delta^\star)$.

We construct G' as follows. Let β be a bijection from $V(G)$ to a set V^+ disjoint from $V(G)$. The vertex set of G' is $V(G) \cup V^+ \cup \{v_0\}$ with v_0 a vertex distinct from those in $V(G) \cup V^+$. We extend the weight function w of G to G' such that the vertices of $V(G)$ have the same weight as in G and the weight of the vertices in $V^+ \cup \{v_0\}$ is 0. Finally, the edge set of G' is defined as follows

$$E(G') := E(G) \cup \{\{v, \beta(v)\}, \{v_0, \beta(v)\} \,:\, v \in V(G)\}.$$

We now construct $\mathscr{L} = (T^\star, \delta^\star)$ from $\mathcal{L} := (T, \delta)$. We obtain $T^\star$ and $\delta^\star$ from T and δ, by doing the following transformations.

- For each leaf ℓ of T with $\delta(\ell) = \{v\}$, we transform ℓ into an internal node by adding two new nodes a_ℓ and b_ℓ as its children such that $\delta^\star(a_\ell) = v$ and $\delta^\star(b_\ell) = \beta(v)$.
- The root of $T^\star$ is a new node r whose children are the root of T and a new node a_r with $\delta^\star(a_r) = v_0$.

In order to simplify the proof, we use the following notations.

For each node $x \in V(T^\star)$, we let $\overline{V_x^{\mathscr{L}}} := V(G') \setminus V_x^{\mathscr{L}}$ and for each node $x \in V(T)$, we let $\overline{V_x^{\mathcal{L}}} := V(G) \setminus V_x^{\mathcal{L}}$.

Now, we prove that $(T^\star, \delta^\star)$ respects Inequalities (1) and (2). Let x be a node of $T^\star$. Observe that if $x \in V(T^\star) \setminus V(T)$, then the set $V_x^{\mathscr{L}}$ either contains one vertex or equals $V(G')$. Hence, in this case, the inequalities hold because $\mathsf{s\text{-}nec}_d(V_x^{\mathscr{L}}) \leq d$ for each $d \in \mathbb{N}^+$ and $f(V_x^{\mathscr{L}}) \leq 1$ for each $f \in \{\mathsf{mim}, \mathsf{mw}, \mathsf{rw}, \mathsf{rw}_{\mathbb{Q}}\}$.

Now, assume that x is also a node of T. Hence, by construction, we have

$$V_x^{\mathscr{L}} = V_x^{\mathcal{L}} \cup \{\beta(v) \,:\, v \in V_x^{\mathcal{L}}\}.$$
$$\overline{V_x^{\mathscr{L}}} = \overline{V_x^{\mathcal{L}}} \cup \{\beta(v) \,:\, v \in \overline{V_x^{\mathcal{L}}}\} \cup \{v_0\}$$

Now, we prove Inequality (1). Let $d \in \mathbb{N}^+$. By construction of G' and $\mathscr{L}$, for each vertex $v \in V_x^{\mathcal{L}}$, we have $\beta(v) \in V_x^{\mathscr{L}}$ and

$$N_{G'}(v) \cap \overline{V_x^{\mathscr{L}}} = N_G(v) \cap \overline{V_x^{\mathcal{L}}}, \tag{5}$$
$$N_{G'}(\beta(v)) \cap \overline{V_x^{\mathscr{L}}} = \{v_0\}. \tag{6}$$

We deduce that, for every $X, Y \subseteq V_x^{\mathscr{L}}$, we have $X \equiv^d_{V_x^{\mathscr{L}}} Y$ if and only

- $X \cap V(G) \equiv^d_{V_x^{\mathcal{L}}} Y \cap V(G)$ and
- $\min(d, |N(v_0) \cap X|) = \min(d, |N(v_0) \cap Y|)$.

Similarly, we deduce that, for every $X, Y \subseteq \overline{V_x^{\mathscr{L}}}$, we have $X \equiv^d_{\overline{V_x^{\mathscr{L}}}} Y$ if and only if

- $X \cap V(G) \equiv^d_{\overline{V_x^{\mathcal{L}}}} Y \cap V(G)$ and
- $X \cap \{v_0\} = Y \cap \{v_0\}$.

Thus, we can conclude that $\mathsf{s\text{-}nec}_d(V_x^{\mathscr{L}}) \leq (d+1) \cdot \mathsf{s\text{-}nec}_d(V_x^{\mathcal{L}})$. Consequently, Inequality (1) holds.

We deduce Inequality (2) from Figure 2 describing the adjacency matrix between $V_x^{\mathscr{L}}$ and $\overline{V_x^{\mathscr{L}}}$.

Now, we explain how we modify the algorithm of Theorem 5.10 in order to find a maximum acyclic (σ, ρ)-dominating set of G by calling this algorithm on G'. For doing so, we modify the definition of the table $\mathcal{A}_x$, the invariant, and the computational steps of the algorithm of Theorem 5.10. The purpose of these modifications is to restrict the (σ, ρ)-domination to the vertices of $V(G)$. For doing so, we consider the set of nodes $S := V(T) \cup \{r, a_r\}$. Observe that, for every node x in S, there are no edges in $G[V_x^{\mathscr{L}}, \overline{V_x^{\mathscr{L}}}]$ between a vertex in $V(G)$ and a vertex in $V(G') \setminus V(G)$. This is not true for the nodes of $V(T^\star) \setminus S$. For this reason, our algorithm ignores the nodes in $V(T^\star) \setminus S$ and computes a table only for the nodes in S.

$$
\begin{array}{cc|c|c|c|}
 & & \multicolumn{3}{c}{\overbrace{\hphantom{MMMMMMMMMMMMMMMMMMMM}}^{\overline{V_x^{\mathscr{L}}}}} \\
 & & \overline{V_x^{\mathcal{L}}} & v_0 & V^+ \cap \overline{V_x^{\mathscr{L}}} \\ \cline{3-5}
\multirow{2}{*}{$V_x^{\mathscr{L}}$} & V^+ \cap V_x^{\mathscr{L}} & 0 & 1 & 0 \\ \cline{3-5}
 & V_x^{\mathcal{L}} & M_{V_x^{\mathcal{L}},\overline{V_x^{\mathcal{L}}}} & \multicolumn{2}{c|}{0} \\ \cline{3-5}
\end{array}
$$

FIGURE 2. The adjacency matrix between $V_x^{\mathscr{L}}$ and $\overline{V_x^{\mathscr{L}}}$.

For every $x \in S$ and every $(R, R') \in \mathcal{R}^d_{V_x^{\mathscr{L}}} \times \mathcal{R}^d_{\overline{V_x^{\mathscr{L}}}}$ we define $\mathcal{A}_x[R, R'] \subseteq 2^{V_x^{\mathscr{L}}}$ as follows

$$\mathcal{A}_x[R, R'] := \{X \subseteq V_x^{\mathscr{L}} \,:\, X \equiv^d_{V_x^{\mathscr{L}}} R \text{ and } (X \cup R') \cap V(G) \ (\sigma, \rho)\text{-dominates } V_x^{\mathscr{L}} \cap V(G)\}.$$

We claim that if G admits an acyclic (σ, ρ)-dominating set D, then there exists $D' \in \mathcal{A}_r[\emptyset, \emptyset]$ such that $D' \cap V(G) = D$ and $G'[D']$ is a tree. Let D be an acyclic (σ, ρ)-dominating set of G with $\mathsf{cc}(G[D]) = \{C_1, \ldots, C_t\}$. For each $i \in \{1, \ldots, t\}$, let v_i be a vertex in C_i. One easily checks that $G'[D \cup \{\beta(v_i) \,:\, 1 \leq i \leq t\} \cup v_0]$ is a tree. Moreover, by definition of $\mathcal{A}_r[\emptyset, \emptyset]$, for every $X \in \mathcal{A}_r[\emptyset, \emptyset]$, if $G[X]$ is a tree, then $X \cap V(G)$ is an acyclic (σ, ρ)-dominating set of G. Hence, if G admits an acyclic (σ, ρ)-dominating set, any $(r, \emptyset)^{\mathsf{acy}}$-representative set of $\mathcal{A}_r[\emptyset, \emptyset]$ contains a set X such that $X \cap V(G)$ is a maximum acyclic (σ, ρ)-dominating set of G.

For every node $x \in S$, we compute a table $\mathcal{D}_x$ satisfying the following invariant.

Invariant. For each node $x \in S$ and each $(R, R') \in \mathcal{R}^d_{V_x^{\mathscr{L}}} \times \mathcal{R}^d_{\overline{V_x^{\mathscr{L}}}}$, the set $\mathcal{D}_x[R, R']$ is a subset of $\mathcal{A}_x[R, R']$ of size at most $\mathcal{N}_f(T^\star, \delta^\star) \cdot \mathsf{nec}_1(V_x^{\mathscr{L}})^2$ that $(x, R')^{\mathsf{acy}}$-represents $\mathcal{A}_x[R, R']$.

Before we explain how to compute the table $\mathcal{D}_x$ for each $x \in S$, we need the following fact and claim. We deduce the following fact from Lemma 4.2 and the fact that, for every node x in S, there are no edges in $G[V_x^{\mathscr{L}}, \overline{V_x^{\mathscr{L}}}]$ between a vertex in $V(G)$ and a vertex in $V(G') \setminus V(G)$.

Fact 5.12. *Let $x \in S$.*

Let $X \subseteq V_x^{\mathscr{L}}$ and $Y, R' \subseteq \overline{V_x^{\mathscr{L}}}$ such that $Y \equiv^d_{\overline{V_x^{\mathscr{L}}}} R'$. Then $(X \cup R') \cap V(G)$ (σ, ρ)-dominates $V_x^{\mathscr{L}} \cap V(G)$ if and only if $(X \cup Y) \cap V(G)$ (σ, ρ)-dominates $V_x^{\mathscr{L}} \cap V(G)$.

We deduce the following claim from Fact 5.12 and Lemma 4.4.

Claim 5.12.1. *Let $x \in S \setminus \{a_r\}$ such that x is not a leaf in T. Let a and b be the children of x in $T^\star$. For every $(R, R') \in \mathcal{R}^d_{V_x^{\mathscr{L}}} \times \mathcal{R}^d_{\overline{V_x^{\mathscr{L}}}}$, we have*

$$\mathcal{A}_x[R, R'] = \bigcup_{(A,A'),\ (B,B')\ d\text{-}(R,R')\text{-compatible}} \mathcal{A}_a[A, A'] \bigotimes \mathcal{A}_b[B, B'].$$

The algorithm starts by computing the table $\mathcal{D}_x$ for each node $x \in S$ such that $x = a_r$ or x is a leaf of T. Since $|V_x^{\mathscr{L}}| \leq 2$, our algorithm directly computes $\mathcal{A}_x[R, R']$ and set $\mathcal{D}_x[R, R'] := \mathcal{A}_x[R, R']$ for every $(R, R') \in \mathcal{R}^d_{V_x^{\mathscr{L}}} \times \mathcal{R}^d_{\overline{V_x^{\mathscr{L}}}}$.

For the other nodes of $T^\star$ our algorithm computes the table $\mathcal{D}_x$ exactly as the algorithm of Theorem 5.10.

The correctness of this algorithm follows from Theorem 5.10. By Theorem 5.10, the running time of this algorithm is

$$O(\mathsf{s\text{-}nec}_2(\mathscr{L})^3 \cdot \mathsf{s\text{-}nec}_1(\mathscr{L})^{2(\omega+1)} \cdot \mathcal{N}_f(\mathscr{L})^2 \cdot n^3).$$

We deduce the running time in function of $\mathcal{L}$ from Inequalities (1) and (2).

$\square$

6. Conclusion

We have generalized the rank-based approach of [4] to work with the d-neighbor-equivalence relation of [7]. As a result, we provide deterministic algorithms for a wide range of connectivity problems running in time $\mathsf{s\text{-}nec}_d(T,\delta)^{O(1)} \cdot n^{O(1)}$ where (T,δ) is a given layout and d is a constant which depends on the problem. From the upper-bounds presented in Lemma 2.7, we obtain parameterized algorithms with parameters and running times given in Table 1.

We have extended our framework in order to solve Maximum Induced Tree, Longest Induced Path, and Feedback Vertex set. We obtain algorithms whose running times match those described in Table 1. But, even if our algorithms rely heavily on the d-neighbor-width on the input rooted layout, some specific parts of the running time analysis use the properties of other parameters. We leave open the existence of $\mathsf{s\text{-}nec}_c(T,\delta)^{O(1)} \cdot n^{O(1)}$ time algorithms for these problems, for some constant c.

Concerning mim-width, we provide unified polynomial-time algorithms for the considered problems for all well-known graph classes having bounded mim-width and for which a layout of bounded mim-width can be computed in polynomial time [1]. This is true for all graph classes studied in [1]. It includes in particular the following classes of graphs: circular arc graphs, permutation graphs, circular k-trapezoid graphs, Dilworth-k graphs and k-polygon graphs for all fixed k. Notice that we also generalize one of the results from [19] proving that the Connected Vertex Cover problem is solvable in polynomial time for circular arc graphs.

This paper highlights the importance of the d-neighbor-equivalence relation in the design of algorithms parameterized by clique-width, ($\mathbb{Q}$-)rank-width, and mim-width. Prior to this work, the d-neighbor-equivalence relation was used only for problems with a locally checkable property like Dominating Set. We prove that the d-neighbor-equivalence relation can also be useful for problems with a connectivity constraint and an acyclicity constaint. Is this notion also useful for other kinds of problems ? Can we use it for the problems which are known to not admit FPT algorithms parameterized by clique-width, $\mathbb{Q}$-rank-width or rank-width ? This is the case for well-known problems such as Hamiltonian Cycle, Graph Coloring, and Max Cut. The complexity of these problems parameterized by clique-width is well-known. Indeed, for each of these problems, we have an algorithm that is optimal under ETH [3, 15]. On the other hand, little is known concerning rank-width and $\mathbb{Q}$-rank-width.

It is worth noticing that Hamiltonian Cycle is NP-complete on graphs of mim-width 1, even when given a layout [17]. Moreover, Graph Coloring is NP-complete on graphs of mim-width 2 because this latter is known to be NP-complete on circular arc graphs [13]. Thus, from Ladner's Theorem, we deduce that, unless $\mathsf{P} = \mathsf{NP}$, there is no $n^{O(\log(\mathsf{nec}_c(T,\delta)))}$ time algorithm for Hamiltonian Cycle and Graph Coloring as this would imply an $n^{O(\log(n))}$ time algorithm for an NP-complete problem. To the best of our knowledge, we do not know if Max Cut is solvable in time $n^{O(\mathsf{mim}(\mathcal{L}))}$ on an n-vertex graph given with a rooted layout $\mathcal{L}$. In fact, we do not even know if Max-Cut is solvable in polynomial time on interval graphs (which are known to have mim-width 1 [1]). On the other hand, it is known that Max Cut is solvable in polynomial time on proper interval graphs [5].

As explained in [2], the algorithmic results we obtain for clique-width are asymptotically optimal under the Exponential Time Hypothesis (ETH). However, this is not the case for the other parameters. It would be particularly interesting to have tight upper bounds for rank-width or $\mathbb{Q}$-rank-width since we know how to compute efficiently these parameters. To the best of our knowledge, there is no algorithm parameterized by rank-width (resp. $\mathbb{Q}$-rank-width) that is known to be optimal under ETH. Even for "basic" problems such as Vertex Cover or Dominating Set, the best algorithms [7] run in time $2^{O(k^2)} \cdot n^{O(1)}$ (resp. $2^{O(k \cdot \log(k))} \cdot n^{O(1)}$) and the best lower bounds state that, unless ETH fails, there are no $2^{o(k)} \cdot n^{O(1)}$ time algorithms parameterized by rank-width or $\mathbb{Q}$-rank-width for these problems.

Finally, Fomin et al. [11] have shown that we can use fast computation of representative sets in matroids to obtain deterministic $2^{O(k)} \cdot n^{O(1)}$ time algorithms parameterized by tree-width for many connectivity problems. Is this approach also generalizable to d-neighbor-width ? Can it be of any help for obtaining $2^{O(\mathsf{rw}(G))} \cdot n^{O(1)}$ time algorithm for problems like Vertex Cover or Dominating Set ?

References

[1] Rémy Belmonte and Martin Vatshelle. Graph classes with structured neighborhoods and algorithmic applications. *Theoret. Comput. Sci.*, 511:54–65, 2013. URL: `http://dx.doi.org/10.1016/j.tcs.2013.01.011`, `doi:10.1016/j.tcs.2013.01.011`.

[2] Benjamin Bergougnoux and Mamadou Moustapha Kanté. Fast exact algorithms for some connectivity problems parametrized by clique-width. Submitted to TCS, 2017. URL: `https://hal.archives-ouvertes.fr/hal-01560555`.

[3] Benjamin Bergougnoux, Mamadou Moustapha Kanté, and O-joung Kwon. An optimal XP algorithm for hamiltonian cycle on graphs of bounded clique-width. In *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, pages 121–132, 2017. URL: `https://doi.org/10.1007/978-3-319-62127-2_11`, `doi:10.1007/978-3-319-62127-2\_11`.

[4] 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. URL: `http://dx.doi.org/10.1016/j.ic.2014.12.008`, `doi:10.1016/j.ic.2014.12.008`.

[5] Arman Boyaci, Tinaz Ekim, and Mordechai Shalom. A polynomial-time algorithm for the maximum cardinality cut problem in proper interval graphs. *Inf. Process. Lett.*, 121:29–33, 2017. URL: `https://doi.org/10.1016/j.ipl.2017.01.007`, `doi:10.1016/j.ipl.2017.01.007`.

[6] Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Boolean-width of graphs. In Jianer Chen and Fedor V. Fomin, editors, *IWPEC*, volume 5917 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 2009.

[7] 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. URL: `http://dx.doi.org/10.1016/j.tcs.2013.01.009`, `doi:10.1016/j.tcs.2013.01.009`.

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

[9] 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. URL: `http://dx.doi.org/10.1109/FOCS.2011.23`, `doi:10.1109/FOCS.2011.23`.

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

[11] 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. URL: `http://doi.acm.org/10.1145/2886094`, `doi:10.1145/2886094`.

[12] 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. URL: `http://dx.doi.org/10.1016/j.dam.2009.10.018`, `doi:10.1016/j.dam.2009.10.018`.

[13] M. R. Garey, David S. Johnson, G. L. Miller, and Christos H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM J. Matrix Analysis Applications*, 1(2):216–227, 1980. URL: `https://doi.org/10.1137/0601025`, `doi:10.1137/0601025`.

[14] Petr A. Golovach, Pinar Heggernes, Mamadou Moustapha Kanté, Dieter Kratsch, Sigve Hortemo Sæther, and Yngve Villanger. Output-polynomial enumeration on graphs of bounded (local) linear mim-width. *Algorithmica*, 80(2):714–741, 2018. URL: `https://doi.org/10.1007/s00453-017-0289-1`, `doi:10.1007/s00453-017-0289-1`.

[15] Petr A. Golovach, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Cliquewidth III: the odd case of graph coloring parameterized by cliquewidth. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 262–273, 2018. URL: `https://doi.org/10.1137/1.9781611975031.19`, `doi:10.1137/1.9781611975031.19`.

[16] Lars Jaffke, O-joung Kwon, and Jan Arne Telle. Polynomial-time algorithms for the longest induced path and induced disjoint paths problems on graphs of bounded mim-width. In *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, pages 21:1–21:13, 2017. URL: `https://doi.org/10.4230/LIPIcs.IPEC.2017.21`, `doi:10.4230/LIPIcs.IPEC.2017.21`.

[17] Lars Jaffke, O-joung Kwon, and Jan Arne Telle. A unified polynomial-time algorithm for feedback vertex set on graphs of bounded mim-width. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, pages 42:1–42:14, 2018. URL: `https://doi.org/10.4230/LIPIcs.STACS.2018.42`, `doi:10.4230/LIPIcs.STACS.2018.42`.

[18] Ki Hang Kim. *Boolean matrix theory and applications*, volume 70. Dekker, 1982.
[19] Pedro Montealegre and Ioan Todinca. On distance-d independent set and other problems in graphs with "few" minimal separators. In *Graph-Theoretic Concepts in Computer Science - 42nd International Workshop, WG 2016, Istanbul, Turkey, June 22-24, 2016, Revised Selected Papers*, pages 183–194, 2016. URL: `https://doi.org/10.1007/978-3-662-53536-3_16`, `doi:10.1007/978-3-662-53536-3\_16`.
[20] Sang-Il Oum. *Graphs of Bounded Rank Width*. PhD thesis, Princeton University, 2005.
[21] 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. URL: `http://dx.doi.org/10.1016/j.tcs.2014.03.024`, `doi:10.1016/j.tcs.2014.03.024`.
[22] Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006. URL: `http://dx.doi.org/10.1016/j.jctb.2005.10.006`, `doi:10.1016/j.jctb.2005.10.006`.
[23] Michaël Rao. *Décompositions de Graphes et Algorithmes Efficaces*. PhD thesis, Université Paul Verlaine, Metz, 2006.
[24] Jan Arne Telle and Andrzej Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discrete Math.*, 10(4):529–550, 1997. URL: `http://dx.doi.org/10.1137/S0895480194275825`, `doi:10.1137/S0895480194275825`.
[25] Martin Vatshelle. *New width parameters of graphs*. PhD thesis, University of Bergen, Bergen, Norway, 2012.
[26] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 887–898, 2012. URL: `http://doi.acm.org/10.1145/2213977.2214056`, `doi:10.1145/2213977.2214056`.

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