Electronic Journal of Graph Theory and Applications **9** (2) (2021), 277-300

Electronic Journal of
Graph Theory and Applications

On the k-rainbow domination in graphs with bounded tree-width

M. Alambardar Meybodi[a], M.R. Hooshmandasl[b], P. Sharifani[c], A. Shakiba[d]

[a] *Department of Applied Mathematics and Computer Science, Faculty of Mathematics and Statistics, University of Isfahan, Iran*
[b]*Department of Computer Science, University of Mohaghegh Ardabili, Iran*
[c]*Institute for Research in Fundamental Sciences(IPM), Tehran, Iran*
[d]*Department of Computer Science, Vali-e-Asr University of Rafsanjan, Iran*

m.alambardar@sci.ui.ac.ir, hooshmandasl@uma.ac.ir, pouyeh.sharifani@gmail.com, ali.shakiba@vru.ac.ir

Abstract

Given a positive integer k and a graph $G = (V, E)$, a function f from V to the power set of I_k is called a k-rainbow function if for each vertex $v \in V$, $f(v) = \emptyset$ implies $\cup_{u \in N(v)} f(u) = I_k$ where $N(v)$ is the set of all neighbors of vertex v and $I_k = \{1, \ldots, k\}$. Finding a k-rainbow function of minimum weight of $\sum_{v \in V} |f(v)|$, which is called the k-rainbow domination problem, is known to be NP-complete for arbitrary graphs and values of k. In this paper, we propose a dynamic programming algorithm to solve the k-rainbow domination problem for graphs with bounded tree-width tw in $\mathcal{O}\left(\left(2^{k+1}+1\right)^{\mathrm{tw}} n\right)$ time, where G has n vertices. Moreover, we also show that the same approach is applicable to solve the weighted k-rainbow domination problem with the same complexity. Therefore, both problems of k-rainbow and weighted k-rainbow domination belong to the class FPT, or fixed parameter tractable, with respect to tree-width. In addition to formally showing the correctness of our algorithms, we also implemented these algorithms to illustrate some examples.

Keywords: domination, k-rainbow domination, weighted k-rainbow domination, bounded tree-width graphs
Mathematics Subject Classification : 05C69, 11Y16
DOI: 10.5614/ejgta.2021.9.2.4

Received: 12 May 2018, Revised: 30 December 2020, Accepted: 19 March 2021.

1. Introduction

Different variations of dominating sets are defined and studied in graph theory and solving these problems algorithmically is an active field of study both among mathematicians and computer scientists [16]. The concept of k-rainbow domination for a graph G was initially proposed and studied in [7, 8]. It is shown that it coincides with the standard domination of the Cartesian product of G with K_k, the complete graph with k vertices.

It is known that rainbow domination problem is NP-complete for arbitrary graphs and values of k. More interestingly, the k-rainbow domination is still NP-complete for chordal and bipartite graphs [11]. The same thing holds for split graphs [15] or planar graphs [26]. Even, when restricted to chordal or bipartite graphs, the 2-rainbow domination problem is known to be NP-complete [9]. By the way, the k-rainbow domination problem can be solved in linear time for trees [11, 26] and for cographs [15]. In addition, polynomial time algorithms are proposed to solve 2-rainbow domination for interval graphs [15] and k-rainbow domination for strongly chordal graphs [10].

Beside algorithmic and computational complexity approaches to the k-rainbow domination problem, several scholars tried to provide bounds on k-rainbow domination number for some classes of graphs and/or for specific values of k. For example, the 2-rainbow domination number for a variety of graph classes such as paths, cycles and suns in [9], generalized Peterson graphs $P(n, 2)$ in [20] is determined. In addition to these exact values, upper or lower bounds for the 2-rainbow domination number in some other kinds of graphs such as generalized Peterson graphs $P(n, 3)$ in [25], and a tight upper bound for generalized Peterson graphs $P(n, k)$ for $n \geq 4k + 1$ in [22] are obtained. Another interesting topic studied for rainbow domination problem is bounding the 2-rainbow domination number of lexicographic product of two graphs G and H in terms of the 2-rainbow domination of G and H [19].

There is also a line of research comparing rainbow domination with other kinds of domination, e.g. Roman domination number in [24, 14] and perfect domination in [13].

One trivial way to compute the k-rainbow domination number exactly is to use the fact that $\gamma_{rk}(G) = \gamma\left(G \Box K_k\right)$ [7, 8]. This way, one can compute the exact value of $\gamma_{rk}(G)$ by iterative computing the $\gamma\left(G \Box K_k\right)$. Note that this is feasible since $\gamma_{rk}(G) \leq k \times \gamma(G)$ by Vizing's conjecture. The time complexity of this approach is $\mathcal{O}\left(1.4969^{n \times k}\right)$ for G with n vertices [21, 15].

It has been shown that k-rainbow domination problem belongs to FPT, i.e. fixed parameter tractable, in [15]. Their proof uses the fact that every graph property which can be expressed in monadic second-order logic is fixed-parameter tractable with respect to tree-width or rank-width [12] and then expresses the k-rainbow domination problem in monadic second-order logic.

In this paper, we will show that k-rainbow domination problem belongs to FPT by proposing a dynamic programming approach and give its runtime as $\mathcal{O}\left(\left(2^{k+1}+1\right)^{\mathrm{tw}} n\right)$ for arbitrary graphs of fixed tree-width tw with n vertices. To enhance the algorithm, we use a certain type of monotonicity in our algorithm. The idea of using monotonicity in dynamic programming algorithms for domination like problems of fixed tree-width is due to [1]. In [1], several algorithms are proposed to solve ordinary, independent, total, perfect, total perfect domination and perfect code problems for graphs of fixed tree-width tw with time complexity of $\mathcal{O}\left(q^{\mathrm{tw}} n\right)$ for $q = 2, 4, 5, 4, 5$ and 4, respectively. We also show that our proposed algorithm is applicable to weighted k-rainbow domination problem with some minor changes.

The rest of the paper is organized as follows. Some necessary topics in graph theory and k-rainbow domination are reviewed in Section 2. In Section 3, we first illustrate our fixed-parameter algorithm for 2-rainbow domination problem and is then followed by a rigorous correctness proof and runtime enhancement by a modification in handling join bags. In Section 4, we will generalize the algorithm of Section 3 for arbitrary values of k and obtain its runtime. Section 5 solves the wighted k-rainbow problem by the same algorithmic approach. A conclusion is drawn in Section 6. Finally, the algorithms of this paper are illustrated in several examples in Appendix A by implementing them in `Python 2.7.1`.

2. Preliminaries

In this section, we will review some vital topics and set our notation. Throughout the paper, graph $G = (V, E)$ is a simple undirected graph with the set of the vertices V and the set of edges E. The *open neighborhood*, or simply *neighborhood*, of a vertex $v \in V$ in G is denoted by $N(v)$ and is equal to $N(v) = \{u \in V \mid \{u, v\} \in E\}$. We define $N_G^A(v) = N_G(v) \cap A$ for any set $A \subseteq V$. A *k-rainbow dominating function*, or kRDF for short, is a function f of the form $f : V \to \mathcal{P}(\{1, \ldots, k\})$ such that $f(v) = \emptyset$ implies $\cup_{u \in N(v)} f(u) = \{1, \ldots, k\}$. Note that $\mathcal{P}(A)$ denotes the power set of A. The *weight of a kRDF f* is defined as $wt(f) = \sum_{v \in V} |f(v)|$. The *$k$-rainbow domination problem* for a graph G is finding a kRDF function f with minimum weight. The minimum weight of any kRDF on G is called the *k-rainbow domination number* of G and is denoted by $\gamma_{rk}(G)$. It is easy to see that 1-rainbow domination problem coincides with the ordinary domination problem. More interestingly, it is known that $\gamma_{rk}(G) = \gamma(G \square K_k)$ where $\gamma(G)$, $\square$ and K_k denote the ordinary domination number of graph G, the Cartesian product of two graphs, and the complete graph with k vertices, respectively [8].

It is well-known that the k-rainbow domination problem is NP-complete. A main road of attack against NP-complete problems is studying their complexity with respect to some fixed parameter such as tree-width [1]. A problem is FPT if its running time with respect to a fixed parameter t and complexity parameter n is $\mathcal{O}\left(f(t) n^{\mathcal{O}(1)}\right)$ for an arbitrary function f which depends only on parameter t.

The notion of tree-width is proposed in [17, 18] and is a famous fixed-parameter. Our presentation is taken from [3]. Given a graph $G = (V, E)$, a *tree decomposition of* G is a pair $\mathcal{T} = (\mathcal{X}, T = (I, F))$ where $\mathcal{X} = \{X_i \mid i \in I\} \subseteq \mathcal{P}(V)$ and tree T satisfies:

1. $\cup_{i \in I} X_i = V$,
2. for all edges $\{u, v\} \in E$, there exists an $i \in I$ such that $\{u, v\} \subseteq X_i$,
3. for all $i, j, k \in I$, if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

Note that the last condition can be substituted by:

3. for each $v \in V$, the set of bags $\{i \in I \mid v \in X_i\}$ is a subtree of T.

The elements of set $\mathcal{X}$ which correspond the nodes of tree T are called *bags*. Note that we reserve the term vertex for elements of V and bags for elements of $\mathcal{X}$. Also, the set X_i is the corresponding bag to node $i \in I$ in tree T. Note that we will use the term bag for node $i \in I$ and $X_i \in \mathcal{X}$ interchangeably when there is no chance of confusion. The *width of a tree decomposition* is defined as $\max_{i \in I} |X_i| - 1$ and the *tree-width of a graph G* equals the minimum width over all tree decompositions of G and is denoted as $\mathrm{tw}(G)$, or simple tw whenever the G is clear from the context. Computing the tree-width of arbitrary graphs is an NP-complete problem [2]. However, finding exact or an approximate solution is an active field of research, e.g. [6, 5]. Recently, it is shown that tree-width is computable in $\mathcal{O}^\star(2.9512^n)$ for a graph with n vertices in polynomial space [6]. Note that $\mathcal{O}^\star(\cdot)$ is used to suppress all other polynomially bounded terms dependent on the complexity parameter [23].

A tree decomposition $\mathcal{T}$ for graph G can be simply rooted by designating a bag as the root bag. A rooted tree decomposition is called *nice* whenever each bag $i \in I$ is one of the following types:

- *leaf bag*: i has no child.
- *forget bag*: i has exactly one child j where $X_i \subseteq X_j$ and $|X_i| = |X_j| - 1$.
- *introduce bag*: i has exactly one child j such that $X_j \subseteq X_i$ and $|X_i| = |X_j| + 1$.
- *join bag*: i has exactly two children j and j' where $X_i = X_j = X_{j'}$.

In addition to being nice, if all leaf bags i satisfy $|X_i| = 1$, then it is called *very nice* [4]. The input tree decomposition to our algorithm is supposed to be very nice, however, as Lemma 2.1 shows, any tree decomposition can be converted in polynomial time to a corresponding very nice one with the same tree-width.

Lemma 2.1. *Let G be a graph with n vertices. Given a tree decomposition $\mathcal{T}$ of G of width tw with $\mathcal{O}(n)$ bags, then a very nice tree decomposition of width tw can be obtained with at most $\mathcal{O}(4n)$ bags in $\mathcal{O}(cn)$ time for some constant c.*

Note that Lemma 2.1 stated here is a corollary of Lemma 6 in [4], which was stated for nice tree decomposition rather very nice ones. It is easy to see that a very nice tree decomposition can be easily obtained from a nice tree decomposition by adding extra bags to leaves.

For a graph G and its corresponding tree decomposition $\mathcal{T} = (\{X_i\}_{i \in I}, (I, F))$, we associate a subgraph of G as $G_i = (V_i, E_i)$ to each bag X_i where V_i is the union of all $X_j \in \mathcal{X}$ such that either $j = i$ or j is a descendant of bag i in tree T and E_i is the set of all edges in E which has both of its endpoints in V_i.

3. Algorithm for 2-Rainbow Domination

In this section, we illustrate the idea of our proposed algorithm for $k = 2$ and consider its correctness and running time. The input to this algorithm is a graph $\mathcal{G} = (V, E)$ and its corresponding very nice tree decomposition $\mathcal{T} = (\mathcal{X}, T = (I, F))$ where $\mathcal{X} = \{X_i \mid i \in I \text{ and } X_i \subseteq V\}$. We assume that for every $i \in I$, there is a fixed ordering on X_i, i.e. $X_i = (x_{i_1}, \cdots, x_{i_{n_i}})$ where $i_1 \leq i_2 \leq \cdots \leq i_{n_i}$.

For each bag X_i, $5^{|X_i|}$ different color labellings like $L_i = (\ell_{i_1}, \ldots, \ell_{i_{n_i}})$ are defined where $|X_i| = n_i$ and $\ell_{i_j} \in \mathcal{P}(\{1,2\}) \cup \{*\}$. For $\ell \in \mathcal{P}(\{1,2\}) \cup \{*\}$, $|\ell|$ is defined as

$$|\ell| = \begin{cases} 0, & \text{if } \ \ell = \{*\}, \\ \mathrm{card}(\ell), & \text{otherwise}, \end{cases} \tag{1}$$

where $\mathrm{card}(\ell)$ denotes the cardinality of set ℓ. It is worth noting that for each bag, the maximum number of color labels equals $5^{\mathrm{tw}+1}$ where the tree-width of graph is tw. For each bag X_i, the cost and validation functions are of the form $C_i : (\mathcal{P}(\{1,2\}) \cup \{*\})^{|X_i|} \to \mathbb{N} \cup \{+\infty\}$ and $E_i : (\mathcal{P}(\{1,2\}) \cup \{*\})^{|X_i|} \to \{1, +\infty\}$, respectively. The first function denotes the cost of each coloring and the last one denotes the validity of the colorings. Finally, for coloring L_m, $s_n(L_m)$ is defined as

$$s_n(L_m) = \begin{cases} \{1,2\} \setminus W_i^n, & \text{if } \ell_n = \{*\}, \\ \text{undefined}, & \text{otherwise}, \end{cases} \tag{2}$$

where $W_i^n = \cup_{v_o \in N_G^{X_i}(v_n)} \ell_o$ such that ℓ_o is the label of vertex v_o and $n \in \{1, \ldots, n_i\}$.

A key ingredient to our algorithm is a kind of monotonicity introduced in [1] for dynamic programming for domination like problems. Let $\prec$ be a partial ordering on $\mathcal{P}(\{1,2\}) \cup \{*\}$ such that

1. $* \prec \emptyset$,
2. $|\ell| \leq |\ell'|$ implies $\ell \prec \ell'$,

for all $\ell, \ell' \in \mathcal{P}(\{1,2\}) \cup \{*\}$. This ordering is naturally extensible to colorings $L_i, L_j \in (\mathcal{P}(\{1,2\}) \cup \{*\})^n$, that is $L_i \prec L_j$ if and only if $\ell_{i_k} \prec \ell_{j_k}$ for $k = 1, \ldots, n$ where $L_i = (\ell_{i_1}, \ldots, \ell_{i_n})$ and $L_j = (\ell_{j_1}, \ldots, \ell_{j_n})$. Mapping $C : (\mathcal{P}(\{1,2\}) \cup \{*\})^n \to \mathbb{N} \cup \{+\infty\}$ is called monotonous if only if $L_i \prec L_j$ implies $C(L_i) \leq C(L_j)$ for all $L_i, L_j \in (\mathcal{P}(\{1,2\}) \cup \{*\})^n$.

1- Initialization Step:. For all leaf bags X_i where $i \in I$, the cost and validation functions are defined as

$$C_i(L_m) = |\ell|, \tag{3}$$

and

$$E_i(L_m) = \begin{cases} \infty, & \text{if } \ \ell = \emptyset, \\ 1, & \text{otherwise}, \end{cases} \tag{4}$$

respectively, for all possible colorings $L_m = (\ell)$. Finally, $s_1(L_m)$ is defined as

$$s_1(L_m) = \begin{cases} \{1,2\}, & \text{if } \ell = \{*\}, \\ \text{undefined}, & \text{otherwise}. \end{cases} \tag{5}$$

Lemma 3.1. *The initialization step consisting of evaluating the cost and validation functions for a leaf node can be carried out in $\mathcal{O}(1)$ time.*

Proof. The proof is clear considering the fact that the input tree decomposition is very nice. Therefore, every leaf bag has exactly one member and each of the cost and validation functions have at most 5 possible inputs. □

Lemma 3.2. *The mapping C_i defined in Equation (3) is monotonous.*

Proof. It is clear. □

2- Iterative Updating:. In a bottom-up traversal of the tree decomposition, from the leaves to the root, after initialization step and based on the bag type, the cost and validation functions are calculated according to the following rules.

- INTRODUCE: Let i be an introduce bag with child j such that $X_i = X_j \cup \{v\}$ for some vertex like $v \in V$. Suppose that $L_m = \left(\ell_1, \ldots, \ell_{n_j}\right)$ is a coloring for bag X_j and $t \in \mathcal{P}\left(\{1,2\}\right) \cup \{*\}$. Let $\phi_t(L_m) = \left(\ell'_1, \ldots, \ell'_{n_j}\right)$ such then

$$\ell'_n = \begin{cases} *, & \text{if } v_n \in N_G^{X_i}(v),\ \ell_n = \emptyset \text{ and } s_n\left(L_{\overline{m}}\right) \subseteq t, \\ \ell_n, & \text{otherwise,} \end{cases} \tag{6}$$

where $L_{\overline{m}} = \left(\ell''_1, \ldots, \ell''_{n_j}\right)$ where

$$\ell''_n = \begin{cases} *, & \text{if } \ell_n = \emptyset, \\ \ell_n, & \text{otherwise.} \end{cases} \tag{7}$$

Lemma 3.3. *For all $t \in \mathcal{P}(\{1,2\}) \cup \{*\}$, it is the case that $C_j(L_m) = C_j(\phi_t(L_m))$.*

Proof. As we just replace the $\emptyset$ by $\{*\}$ in function Φ_t and the cost of both is zero, then the cost function does not change.

□

Lemma 3.4. *The cost and validation functions for introduce bag X_i with child bag X_j are defined as*

$$C_i(L_m \times \ell) = \begin{cases} C_j(L_m), & \textit{if} \quad \ell = \{*\}, \\ C_j(L_m), & \textit{if} \quad \ell = \emptyset, \\ C_j(\phi_{\{1\}}\left(L_m\right)) + 1, & \textit{if} \quad \ell = \{1\}, \\ C_j(\phi_{\{2\}}\left(L_m\right)) + 1, & \textit{if} \quad \ell = \{2\}, \\ C_j(\phi_{\{1,2\}}\left(L_m\right)) + 2, & \textit{if} \quad \ell = \{1,2\}, \end{cases} \tag{8}$$

and

$$E_i(L_m \times \ell) = \begin{cases} \infty, & \textit{if } \ell = \emptyset \textit{ and } \cup_{x_n \in N_G^{X_j}(v)} \ell_n \neq \{1,2\}, \\ E(L_m), & \textit{otherwise,} \end{cases} \tag{9}$$

respectively, where $L_{m'} = L_m \times \ell = \left(\ell_1, \ldots, \ell_{n_j}, \ell\right)$ is a coloring for bag X_i, $L_m = \left(\ell_1, \ldots, \ell_{n_j}\right)$ is a coloring for bag X_j and $\ell \in \mathcal{P}\left(\{1,2\}\right) \cup \{*\}$. If $\ell \neq \{*\}$, then we set $s_n(L_{m'})$ to $s_n(L_m) \setminus \ell$ for all $v_n \in N_G^{X_j}(v)$ which satisfies $\ell_n = \{*\}$. Otherwise, i.e. $\ell = \{*\}$, we set $s_n(L_{m'})$ to $\{1,2\} \setminus \cup_{v_n \in N_G^{X_i}(v)} \ell_n$.

Proof. In Equation (8), we consider the following cases for ℓ:

Case 1. $\ell = \{*\}$ or $\ell = \emptyset$. The value of cost function does not change if the label of the introduced vertex ℓ is $\{*\}$ or $\emptyset$.

Case 2. $\ell = \{1\}$ or $\ell = \{2\}$. Without loss of generality, we assume that $\ell = \{1\}$. So, $\{1\}$ appears in

$$\bigcup_{x_k \in N_G^{X_j}(v)} \ell_k,$$

for $L_m = \left(\ell_1, \ldots, \ell_{n_j}\right)$. To evaluate $C_i(L_m \times \{1\})$, suppose $\ell_k = \emptyset$ for some vertices $v_k \in X_j$. Since v is labeled by $\{1\}$ and we have $s_k(L_{\overline{m}}) \subseteq \{1\}$ for some $x_k \in N(v)$ which is labeled by $\emptyset$ in L_m. So, by the Lemma 3.3, we obtain $C_i(L_m \times \ell) = C_j(\phi_{\{1\}}(L_m)) + 1$.

Case 3. $\ell = \{1, 2\}$. This case can be shown with the same approach for the previous case.

For correctness of Equation (9), it is enough to verify its validity for color $L_m \times \emptyset$. Since X_i is the first node with vertex v present, then all of the neighbors of v appear in X_i and its ascendant bags in T. So we set $E(L_m \times \emptyset) = \infty$ if $\cup_{v_k \in N(v)} \ell_k \neq \{1, 2\}$. In other cases, the validity $L_m \times \emptyset$ is similar to the validity of L_m. □

It is not hard to verify the following statement.

Lemma 3.5. *The cost function defined in Equation (8) is monotonous if the cost function for X_j is also monotonous.*

Lemma 3.6. *The cost and validation functions defined in Equations (8) and (9) can be calculated in $\mathcal{O}\left(5^{n_i} n_i\right)$ time.*

Proof. For bag X_i, there are $5^{|X_i|}$ different color labels that we need to compute. Also, to compute the cost and validation functions for every color label, we need to spend $\mathcal{O}\left(|X_i|\right)$ time. Therefore, the cost and validation functions are computable in $\mathcal{O}\left(5^{|X_i|}|X_i|\right)$ time. □

- FORGET: Let i is a forget bag with child j such that $X_i = X_j \setminus \{v\}$. Suppose that $L_m = \left(\ell_1, \ldots, \ell_{n_j}\right)$ be a coloring for bag X_j.

Lemma 3.7. *The cost and validation functions for forget bag X_i with child bag X_j are defined as*

$$C_i(L_{m'}) = \min_{m \in \mathcal{M}} \left\{C_j\left(L_m\right) \times E_j\left(L_m\right)\right\}, \tag{10}$$

and

$$E_i(L_{m'}) = \begin{cases} \infty, & \textit{if } \; E_j\left(L_m\right) = \infty, \\ 1, & \textit{otherwise,} \end{cases} \tag{11}$$

where $L_{m'} = \left(\ell_1, \ldots, \ell_{n_j - 1}\right)$ is a coloring for bag X_i and

$$\mathcal{M} = \left\{m'' \mid L_{m''} = L_{m'} \times \ell, \ell \in \mathcal{P}\left(\{1, 2\}\right)\right\}. \tag{12}$$

Also, we have considered ℓ_{n_j} as the corresponding color for vertex v. Finally, for $n = 1, \ldots, n_j - 1$, we set $s_n(L_{m'}) = s_n(L_m)$.

Proof. For each color L_m of bag X_i, there are 5 different color labels $L_m \times (\{*\}, \emptyset, \{1\}, \{2\}, \{1,2\})$ for bag X_j. Since X_i is a forget bag, then there are no other ascendant bags of bag X_i which contains v. So, we consider minimum value of all $C(L_m \times \ell)$ needs to be computed, where $L_m \times \ell$ is a valid labeling. If all possible color labels of $L_m \times \ell$ are invalid, then label L_m cannot lead to a valid color label for bag X_i. □

The following lemmas are easy to verify.

Lemma 3.8. *The cost function defined in Equation (10) is monotonous if the cost function for X_j is also monotonous.*

Lemma 3.9. *The cost and validation functions defined in Equations (10) and (11) can be calculated in $\mathcal{O}(5^{n_i})$ time.*

- JOIN: Let i is a join bag with two children j and j'. Since X_i is a join bag, we know that $X_i = X_j = X_{j'}$. Suppose that $L_m = \left(\ell_1^m, \ldots, \ell_{n_i}^m\right)$, $L_{m'} = \left(\ell_1^{m'}, \ldots, \ell_{n_j}^{m'}\right)$ and $L_{m''} = \left(\ell_1^{m''}, \ldots, \ell_{n_{j'}}^{m''}\right)$ be colorings for bags X_i, X_j and $X_{j'}$, respectively. Colorings $L_{m'}$ and $L_{m''}$ divide coloring L_m whenever for all $n = 1, \ldots, n_i$ we have
  1. $\ell_n^m = \{*\}$ implies $\ell_n^{m'} = \ell_n^{m''} = \{*\}$,
  2. $\ell_n^m = \emptyset$ implies either $\ell_n^{m'} = \ell_n^{m''} = \emptyset$, $\ell_n^{m'} = \{*\} \wedge \ell_n^{m''} = \emptyset$ or $\ell_n^{m'} = \emptyset \wedge \ell_n^{m''} = \{*\}$,
  3. $\ell_n^m = \{1\}$ implies either $\ell_n^{m'} = \ell_n^{m''} = \{1\}$, $\ell_n^{m'} \in \{*, \emptyset\} \wedge \ell_n^{m''} = \{1\}$ or $\ell_n^{m'} = \{1\} \wedge \ell_n^{m''} \in \{*, \emptyset\}$,
  4. $\ell_n^m = \{2\}$ implies either $\ell_n^{m'} = \ell_n^{m''} = \{2\}$, $\ell_n^{m'} \in \{*, \emptyset\} \wedge \ell_n^{m''} = \{2\}$ or $\ell_n^{m'} = \{2\} \wedge \ell_n^{m''} \in \{*, \emptyset\}$,
  5. $\ell_n^m = \{1,2\}$ implies either $\ell_n^{m'} = \{1,2\}$ or $\ell_n^{m''} = \{1,2\}$.

These five cases partition the state space for possible colorings of bag X_i and hence can be used to construct the cost and validation of colorings for X_i.

Lemma 3.10. *The cost and validation function for join bag X_i with children X_j and $X_{j'}$ are defined as:*

$$\begin{aligned} C_i(L_m) = \min_{m',m'' \in \mathcal{M}_m} \Big\{ & C_j(L_{m'}) \times E_j(L_{m'}) + \\ & C_{j'}(L_{m''}) \times E_{j'}(L_{m''}) - \sum_{n=1}^{n_i} \left| \ell_n^{m'} \cap \ell_n^{m''} \right| \Big\}, \end{aligned} \tag{13}$$

and

$$E_i(L_m) = \begin{cases} \infty, & \textit{if} \quad C_j(L_m) = \infty \textit{ and } C_{j'}(L_m) = \infty, \\ 1, & \textit{otherwise,} \end{cases} \tag{14}$$

respectively, where

$$\mathcal{M}_m = \{(m', m'') \mid L_{m'} \textit{ and } L_{m''} \textit{ divide } L_m\}. \tag{15}$$

We set $s_n(L_m) = s_n(L_{m'}) \cap s_n(L_{m''})$ for all $n = 1, \ldots, n_i$ and $(m', m'') \in \mathcal{M}_m$.

Proof. We determine color labels of bag X_i using labels of bags X_j and $X_{j'}$. Suppose $v \in X_i$ and nodes X_j and $X_{j'}$ are children of X_i. Also, assume that $L_{m'} = (\ell_{i_1}^{m'}, \ell_{i_2}^{m'}, \cdots, \ell_{i_{n_i}}^{m'})$ and $L_{m''} = (\ell_{i_1}^{m''}, \ell_{i_2}^{m''}, \cdots, \ell_{i_{n_i}}^{m''})$ are color labels for X_j and $X_{j'}$. If $\ell_v^{m'} = \ell_v^{m''} = \{*\}$, node v will be labeled by $\{*\}$ in L_v^m. If $\ell_v^{m'} = \{*\}$ and $\ell_v^{m''} = d \neq \{*\}$ or vice versa, then node v will be labeled by d in L_v^m. If $\ell_v^{m'} = d_1 \neq \{*\}$ and $\ell_v^{m''} = d_2 \neq \{*\}$, then node v will be labeled by $d_1 \cup d_2$ in L_v^m. It is easy to see

$$C(L_m) = C(L_{m'}) \times E(L_{m'}) + C(L_{m''}) \times E(L_{m''}) - \sum_{x_t \in X_i} |\ell_t^{m'} \cap \ell_t^{m''}|. \tag{16}$$

If there are no any valid color labels $L_{m'}$ or $L_{m''}$ to compute L_m, then we set $C(L_m) = \infty$ and $E(L_m) = \infty$, otherwise $E(L_i^r) = 1$. □

Lemma 3.11. *The cost function in Equation (13) is monotonous if the cost functions for X_j and $X_{j'}$ are also monotonous.*

Lemma 3.12. *The cost and validation functions defined in Equations (13) and (14) can be calculated in $\mathcal{O}(25^{n_i} n_i)$.*

Proof. These values can be calculated by combining the colorings for bags j and j', which are totally $\mathcal{O}(25^{n_i} n_i)$ pairs. □

3- Final Step:. Suppose that X_i is the root of the tree decomposition $\mathcal{T}$. Then, we have

$$\gamma_{r,2}(G) = \min_{r=1,\dots,5^{n_i}} \{C_i(L_i^r) \times E_i(L_i^r) \mid L_i^r \text{ does not contain } *\}. \tag{17}$$

3.1. Correctness and Time Complexity

By the above steps, we have the following results.

Theorem 3.1. *The value computed in Equation (17) is the 2-rainbow domination number of graph G.*

Proof. It is obvious that the domination computed in Equation (17) is the 2-rainbow domination as we just considered valid color labels in the root of the tree. The property of tree decomposition guarantees that every vertex in graph is labeled along the algorithm. In the leaf bags, we have initialized with the possible color labels corresponding to that leaf bag. For introduce bags, we have considered and computed the cost and validation of all color labels after addition of the new vertex. Also, for forget and join bags, we have chosen the minimum cost for among color labels. So, as all the vertices have labels and we have selected valid color labels in the tree root and also, minimum color label is preserved for each color label in the bottom-up approach of the algorithm, then the result is easily concluded. □

Corollary 3.1. *The value in Equation (17) can be computed in $\mathcal{O}(25^{\mathrm{tw}} N)$ time for a graph G with N nodes and treewidth of tw given a very nice tree decomposition for it.*

Proof. This is directly follows from Lemmas 3.1, 3.6, 3.9, and 3.12. □

3.2. Improving the Time Complexity

The time complexity of the proposed algorithm for determining the 2-rainbow domination number of a graph G can be improved to $\mathcal{O}\left(9^{\mathrm{tw}} N\right)$ if the division condition of join bags is replaced by the followings. Suppose that $L_m = \left(\ell_1^m, \ldots, \ell_{n_i}^m\right)$, $L_{m'} = \left(\ell_1^{m'}, \ldots, \ell_{n_j}^{m'}\right)$ and $L_{m''} = \left(\ell_1^{m''}, \ldots, \ell_{n_{j'}}^{m''}\right)$ are colorings for bags X_i, X_j, and $X_{j'}$, respectively. Colorings $L_{m'}$ and $L_{m''}$ divide coloring L_m whenever for all $n = 1, \ldots, n_i$ we have

1. $\ell_n^m = \{*\}$ implies $\ell_n^{m'} = \ell_n^{m''} = \{*\}$,
2. for label $\ell \in \mathcal{P}(\{1,2\})$, it is the case that $\ell_n^m = \ell$ implies that $\ell_n^{m'}, \ell_n^{m''} \in \{\ell, \{*\}\} \wedge \ell_n^{m'} \neq \ell_n^{m''}$.

Lemma 3.13. *The cost and validation functions defined in Equations (13) and (14) can be calculated in $\mathcal{O}\left(9^{n_i} n_i\right)$ if we use the improved division condition in place of the original one.*

Proof. The running time of step 2 is given by

$$T = \sum_{L_m \in (\mathcal{P}(\{1,2\}) \cup \{*\})^{n_i}} \left|\{(L_{m'}, L_{m''}) : L_{m'} \textit{ and } L_{m''} \textit{ divide } L_m\}\right|.$$

Let $L_m = (\ell_{i_1}, \ell_{i_2}, \cdots, \ell_{i_{n_i}})$ be a labeling for bag X_i, and

$$z_d = \left|\{t \in \{i_1, i_2, \cdots, i_{n_i}\} : \ell_t = d \text{ and } L_m \text{ is a color label for bag } X_i\}\right|,$$

for $d \in \mathcal{P}(\{1,2\}) \cup \{*\}$, we have

$$\begin{aligned}
T &= \sum_{z_\emptyset=0}^{n_i} \sum_{z_{\{1\}}=0}^{n_i - z_\emptyset} \sum_{z_{\{2\}}=0}^{n_i - z_\emptyset - z_{\{1\}}} \sum_{z_{\{1,2\}}=0}^{n_i - z_\emptyset - z_{\{1\}} - z_{\{2\}}} \binom{n_i}{z_\emptyset} 2^{z_\emptyset} \binom{n_i - z_\emptyset}{z_{\{1\}}} 2^{z_{\{1\}}} \\
&\qquad \binom{n_i - z_\emptyset - z_{\{1\}}}{z_{\{2\}}} 2^{z_{\{2\}}} \binom{n_i - z_\emptyset - z_{\{1\}} - z_{\{2\}}}{z_{\{1,2\}}} 2^{z_{\{1,2\}}} \\
&= \sum_{z_\emptyset=0}^{n_i} \sum_{z_{\{1\}}=0}^{n_i - z_\emptyset} \sum_{z_{\{2\}}=0}^{n_i - z_\emptyset - z_{\{1\}}} \binom{n_i}{z_\emptyset} 2^{z_\emptyset} \binom{n_i - z_\emptyset}{z_{\{1\}}} 2^{z_{\{1\}}} \\
&\qquad \binom{n_i - z_\emptyset - z_{\{1\}}}{z_{\{2\}}} 2^{z_{\{2\}}} 3^{n_i - z_\emptyset - z_{\{1\}} - z_{\{2\}}} \\
&= 3^{n_i} \sum_{z_\emptyset=0}^{n_i} \sum_{z_{\{1\}}=0}^{n_i - z_\emptyset} \sum_{z_{\{2\}}=0}^{n_i - z_\emptyset - z_{\{1\}}} \binom{n_i}{z_\emptyset} (2/3)^{z_\emptyset} \binom{n_i - z_\emptyset}{z_{\{1\}}} (2/3)^{z_{\{1\}}} \binom{n_i - z_\emptyset - z_{\{1\}}}{z_{\{2\}}} (2/3)^{z_{\{2\}}} \\
&= 3^{n_i} \sum_{z_\emptyset=0}^{n_i} \sum_{z_{\{1\}}=0}^{n_i - z_\emptyset} \binom{n_i}{z_\emptyset} (2/3)^{z_\emptyset} \binom{n_i - z_\emptyset}{z_{\{1\}}} (2/3)^{z_{\{1\}}} (5/3)^{n_i - z_\emptyset - z_{\{1\}}} \\
&= 5^{n_i} \sum_{z_\emptyset=0}^{n_i} \sum_{z_{\{1\}}=0}^{n_i - z_\emptyset} \binom{n_i}{z_\emptyset} (2/5)^{z_\emptyset} \binom{n_i - z_\emptyset}{z_{\{1\}}} (2/5)^{z_{\{1\}}} \\
&= 5^{n_i} \sum_{z_\emptyset=0}^{n_i} \binom{n_i}{z_\emptyset} (2/5)^{z_\emptyset} (7/5)^{n_i - z_\emptyset}
\end{aligned}$$

$$= 7^{n_i} \sum_{z_\emptyset=0}^{n_i} \binom{n_i}{z_\emptyset} (2/7)^{z_\emptyset}$$
$$= 7^{n_i}(9/7)^{n_i} = 9^{n_i}$$

□

Theorem 3.2. *The value computed in Equation (17) is the 2-rainbow domination number of graph G if the division condition is changed as is described in this section.*

Proof. The proof is similar to the proof of Theorem 3.1. □

Theorem 3.3. *The value in Equation (17) can be computed in $\mathcal{O}\left(9^{\mathrm{tw}}N\right)$ time for a graph G with N nodes and treewidth of tw given a very nice tree decomposition for it and that the improved division condition is used for join bags.*

Proof. The proof is similar to the proof of Corollary 3.1, except that the division condition for join bags is different with time complexity of Lemma 3.13. □

4. Generalization for arbitrary *k*

The algorithm stated in Section 3 can be easily generalized to compute the k-rainbow domination number of a graph G with N vertices in $\mathcal{O}\left(\left(2^{k+1}+1\right)^{\mathrm{tw}}N\right)$. This is easily done by substituting the set $\{1,2\}$ by the set $\{1,\dots,k\}$, using the improved division condition stated in Section 3.2 and changing the way we compute the cost function of an introduce bag X_i with child X_j such that $X_i = X_j \cup \{v\}$, Equation (8), as

$$C_i(L_m \times \ell) = \begin{cases} C_j(L_m), & \text{if } \ \ell \in \{\emptyset, *\}, \\ C_j\left(\phi_\ell\left(L_m\right)\right) + |\ell|, & \text{otherwise,} \end{cases} \tag{18}$$

where bag X_j is labeled by $L_m = \left(\ell_1,\dots,\ell_{n_j}\right)$ and $\ell \in \mathcal{P}\left(\{1,\dots,k\}\right) \cup \{*\}$.

Theorem 4.1. *Given that X_i is the root of the tree decomposition, then the value of*

$$\min_{m=1,\dots,\left(2^k+1\right)^{n_i}} \left\{C_i(L_m) \times E_i(L_m) \mid L_m \textit{ does not contain } *\right\}, \tag{19}$$

equals the k-rainbow domination number of a graph G and can be computed in $\mathcal{O}\left(\left(2^{k+1}+1\right)^{\mathrm{tw}}N\right)$ time given a very nice tree decomposition of it, where N is the number of its nodes and its tree-width equals tw.

Proof. By changing $\{1,2\}$ to $\{1,\dots,k\}$, it is easy to see that

- the initialization step can be done in $\mathcal{O}(1)$ time,
- the cost and validation functions for an introduce bag X_i can be determined in $\mathcal{O}\left(\left(2^k+1\right)^{n_i} n_i\right)$,

- the cost and validation functions for a forget bag X_i can be evaluated in in $\mathcal{O}\left(\left(2^k+1\right)^{n_i} n_i\right)$, too.

The time complexity of calculating the cost and validation functions for a join bag X_i equals

$$T = \sum_{L_m \in (\mathcal{P}(\{1,\dots,k\}) \cup \{*\})^{n_i}} \left|\{(L_{m'}, L_{m''} \mid L_{m'} \text{ and } L_{m''} \text{ divide } L_m)\}\right|. \tag{20}$$

By induction on the number of Σs in T, denoted as $\#_\Sigma$, we show $T = \left(2^{k+1}+1\right)^{n_i}$. If $\#_\Sigma = 1$, then $T = \sum_{z=0}^{n_i} \binom{n_i}{z}(2)^z = 3^{n_i}$. Let $p \geq 2$ be given and suppose (18) is true for $\#_\Sigma = p$. Then

$$\begin{aligned}
T &= \sum_{z_{p+1}=0}^{n_i} \binom{n_i}{z_{p+1}} (2)^{z_{p+1}} \sum_{z_p=0}^{n_i - z_{p+1}} \binom{n_i - z_{p+1}}{z_p} (2)^{z_p} \sum_{z_{p-1}=0}^{n_i - z_p} \binom{n_i - z_p}{z_{p-1}} (2)^{z_{p-1}} \\
&\quad \cdots \sum_{z_0=0}^{n_i - \sum_{i=p}^{2} z_i} \binom{n_i - \sum_{i=p}^{2} z_i}{z_1} (2)^{z_1} \\
&= \sum_{z_{p+1}=0}^{n_i} \binom{n_i}{z_{p+1}} (2)^{z_{p+1}} (2p+1)^{n_i - z_{p+1}} \\
&= (2p+1)^{n_i}((2p+3)/(2p+1))^{n_i} \\
&= (2(p+1)+1)^{n_i}.
\end{aligned}$$

Thus, (18) holds for $\#_\Sigma = p+1$, and the proof of the induction step is complete.

For 2-rainbow domination, we have $\#_\Sigma = 4$ so $T = 9^{n_i}$. Also for k-rainbow domination number, it is easy to see that $\#_\Sigma = 2^k$, therefore we have $T = (2^{k+1}+1)^{n_i}$.

□

5. Weighted k-Rainbow Domination

In this section, we extend the approach discussed in Section 4 in order to find the weighted domination of a graph $G = (V, E)$. Each vertex $v \in V$ has k positive weights which are denoted by $w_j(v)$ for $j = 1, \dots, k$. Then, for a rainbow function $f : V \to (\mathcal{P}(\{1,\dots,k\}) \cup \{*\})$, its cost $w(f)$ equals

$$w(f) = \sum_{v \in V} \sum_{j \in f(v)} w_j(v). \tag{21}$$

Note that $f(v)$ is the same as the label of v. The only modification we need to make in the approach of Section 4 is to change the cost functions. Everything else remains the same. To be precise, we will use the following modified cost functions:

1. for a leaf bag $X_i = \{v\}$ with coloring $L_m = (\ell_1)$, its cost function is modified as

$$C_i(L_m) = \begin{cases} 0, & \text{if } \ell_1 \in \{\emptyset, *\}, \\ \sum_{j \in \ell_1} w_j(v), & \text{otherwise.} \end{cases} \tag{22}$$

2. for an introduce bag X_i with child X_j such that $X_i = X_j \cup \{v\}$, we use the following cost function
$$C_i(L_m \times \ell) = \begin{cases} C_j(L_m), & \text{if} \quad \ell \in \{\emptyset, *\}, \\ C_j\left(\phi_\ell\left(L_m\right)\right) + \sum_{j\in\ell} w_j(v), & \text{otherwise}, \end{cases} \tag{23}$$
where bag X_j is labeled by $L_m = \left(\ell_1, \ldots, \ell_{n_j}\right)$ and $\ell \in \mathcal{P}\left(\{1, \ldots, k\}\right) \cup \{*\}$.
3. for a forget bag, we use exactly the same approach.
4. for a join node X_i with children X_j and $X_{j'}$, we will change Equation (13) to
$$\begin{aligned} C_i(L_m) = \min_{m',m''\in\mathcal{M}_m} \Big\{ & C_j(L_{m'}) \times E_j(L_{m'}) \\ & + C_j(L_{m''}) \times E_j(L_{m''}) - \sum_{n=1}^{n_i} w_{\ell_n^{m'} \cap \ell_n^{m''}}(v_n) \Big\}, \end{aligned} \tag{24}$$
where $X_i = (v_1, \ldots, v_{n_i})$ and $w_t(v)$ for $t \subseteq \{1, \ldots, k\}$ and vertex $v \in V$ is defined as
$$w_t(v) = \begin{cases} \sum_{j\in t} w_j(v), & \text{if} \quad t \notin \{\emptyset, *\}, \\ 0, & \text{if} \quad t \in \{\emptyset, *\}. \end{cases} \tag{25}$$

Theorem 5.1. *Given that X_i is the root of the tree decomposition, then the value of*
$$\gamma_{wr,k} = \min_{m=1,\ldots,\left(2^k+1\right)^{n_i}} \left\{ C_i(L_m) \times E_i(L_m) \mid L_m \textit{ does not contain } * \right\}, \tag{26}$$
equals the weighted k-rainbow domination of a graph G and can be computed in $\mathcal{O}\left(\left(2^{k+1}+1\right)^{\mathrm{tw}} N\right)$ time given a very nice tree decomposition of it, where N is the number of its nodes and its treewidth equals tw.

Proof. The proof is similar to the proof of Theorem 4.1. □

6. Conclusion

In this paper, we presented a novel dynamic programming algorithm to solve the k-rainbow domination for graphs with bounded tree-width tw in $\mathcal{O}\left(\left(2^{k+1}+1\right)^{\mathrm{tw}} n\right)$ time. This problem is known to be NP-hard for general graphs and arbitrary values of k. We also used the same approach to solve the weighted k-rainbow domination problem with the same complexity. So, we have shown that both problems belong to the class FPT with respect to tree-width. We have also illustrated our algorithms by implementing them in `Python`.

References

[1] J. Alber and R. Niedermeier, Improved tree decomposition based algorithms for domination-like problems, *Theoretical Informatics*, Springer, (2002) 613–627.

[2] S. Arnborg, D.G. Corneil, and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM Journal on Algebraic Discrete Methods* **8** (2) (1987), 277–284.

[3] H.L. Bodlaender, A tourist guide through treewidth, *Acta cybernetica* **11** (1-2) (1993), 1–21.

[4] H.L. Bodlaender, P. Bonsma, and D. Lokshtanov, The fine details of fast dynamic programming over tree decompositions, In: International Symposium on Parameterized and Exact Computation, Springer (2013), 41–53.

[5] H.L. Bodlaender, F.V. Fomin, A.M.C.A. Koster, D. Kratsch, and D.M. Thilikos (2006) On Exact Algorithms for Treewidth, In: Azar Y., Erlebach T. (eds) Algorithms – ESA 2006. *Lecture Notes in Computer Science*, Vol 4168, Springer.

[6] H.L. Bodlaender, F.V. Fomin, A.M. Koster, D. Kratsch, and D.M. Thilikos, On exact algorithms for treewidth, *ACM Transactions on Algorithms* **9** (1) 12 (2012).

[7] B. Brešar, M.A. Henning, and D.F. Rall, Paired-domination of Cartesian products of graphs and rainbow domination, *Electronic Notes in Discrete Mathematics* **22** (2005), 233–237.

[8] B. Brešar, M.A. Henning, and D.F. Rall, Rainbow domination in graphs, *Taiwanese Journal of Mathematics* **12** (1) (2008), 213–225.

[9] B. Brešar and T.K. Šumenjak, On the 2-rainbow domination in graphs, *Discrete Appl. Math.* **155** (17) (2007), 2394–2400.

[10] G.J. Chang, B.J. Li, and J. Wu, Rainbow domination and related problems on strongly chordal graphs, *Discrete Appl. Math.* **161** (10) (2013), 1395–1401.

[11] G.J. Chang, J. Wu, and X. Zhu, Rainbow domination on trees, *Discrete Appl. Math.* **158** (1) (2010), 8–12.

[12] B. Courcelle, The expression of graph properties and graph transformations in monadic second-order logic, *Handbook of Graph Grammars* **1** (1997), 313–400.

[13] L.R. Fuentes, I.J. Dejter, and C.A. Araujo, Rainbow perfect domination in lattice graphs, *Electronic J. Graph Theory Appl.* **6** (1) (2018), 95–112.

[14] S. Fujita and M. Furuya: Difference between 2-rainbow domination and Roman domination in graphs, *Discrete Appl. Math.* **161** (6) (2013), 806–812.

[15] W.K. Hon, T. Kloks, H.H. Liu, and H.L. Wang, Rainbow domination and related problems on some classes of perfect graphs, Springer International Publishing, Cham (2016), 121–134.

[16] N.J. Rad and E. Shabani, On the complexity of some hop domination parameters, *Electronic J. Graph Theory Appl.* **7** (1), (2019), 77–89.

[17] N. Robertson and P.D. Seymour, Graph minors. I. Excluding a forest. *J. Combin. Theory Ser. B* **35** (1) (1983), 39–61.

[18] N. Robertson and P.D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, *Journal of Algorithms* **7** (3) (1986), 309–322.

[19] T.K. Šumenjak, D.F. Rall, and A. Tepeh, Rainbow domination in the lexicographic product of graphs, *Discrete Appl. Math.* **161** (13) (2013), 2133–2141.

[20] C. Tong, X. Lin, Y. Yang, and M. Luo, 2-rainbow domination of generalized Petersen graphs $p(n, 2)$, *Discrete Appl. Math.* **157** (8) (2009), 1932–1937.

[21] J.M. Van Rooij and H.L. Bodlaender, Exact algorithms for dominating set, *Discrete Appl. Math.* **159** (17) (2011), 2147–2164.

[22] Y. Wang and K. Wu, A tight upper bound for 2-rainbow domination in generalized Petersen graphs, *Discrete Appl. Math.* **161** (13) (2013), 2178–2188.

[23] G.J. Woeginger, Exact algorithms for NP-hard problems: A survey, In: Combinatorial Optimization—Eureka, You Shrink!, Springer (2003), 185–207.

[24] Y. Wu and H. Xing, Note on 2-rainbow domination and Roman domination in graphs. *Appl. Math. Lett.* **23** (6) (2010), 706–709.

[25] G. Xu, 2-rainbow domination in generalized Petersen graphs $p(n, 3)$, *Discrete Appl. Math.* **157** (11) (2009), 2570 – 2573.

[26] C.K. Yen, 2-rainbow domination and its practical variation on weighted graphs. In: Advances in Intelligent Systems and Applications-Volume 1, Springer (2013), 59–68.

Appendix A: Illustrative Examples

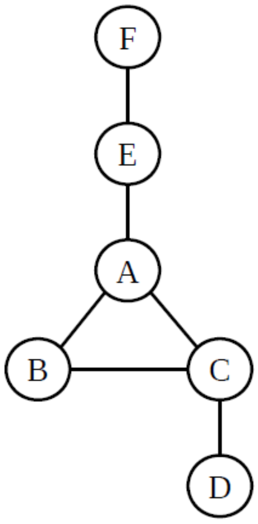
The algorithms in this paper are implemented in `Python 2.7.11`. Moreover, we have used `networkx 1.11` to represent graph structures as well as using some basic algorithms. Throughout the examples in this section, we are going to compute the (weighted) k-rainbow domination of the graph in Figure 1a with the corresponding tree decomposition in Figure 1b.

Computing the cost and validation functions for a tree decomposition can be viewed as a table-filling algorithms. So, we use a hash table with color labels as keys for each bag. A tricky issue is computing the cost and validations for join bags. To accommodate with the proposed running time, we need to take a generative approach, that is for each color labels of a join bag, generate the set of possible color labels.

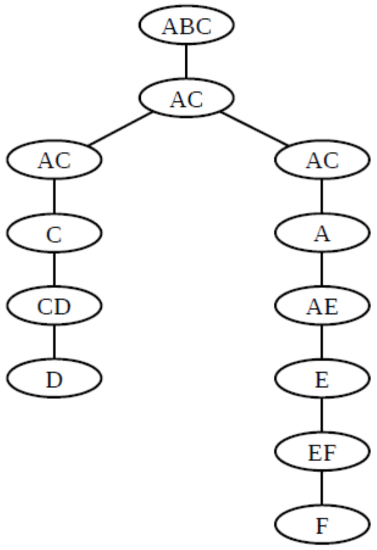
Running the algorithm to compute the 2-rainbow domination for the graph G gives $\gamma_{r,2} = 4$ with 2-rainbow function f_2 defined as follows

$$\begin{aligned} &f_2(A) = \{1, 2\}, \quad f_2(B) = \emptyset, \quad f_2(C) = \emptyset, \\ &f_2(D) = \{2\}, \qquad f_2(E) = \emptyset, \quad f_2(F) = \{1\}. \end{aligned} \tag{27}$$

The values of cost, validation and symbol functions for leaf bag $\{F\}$, forget bag $\{E\}$, introduce bag $\{E, F\}$ and join bag $\{A, C\}$ are given in Tables 1, 2, 3 and 4 respectively.



(a) Main Graph G.



(b) Very Nice Tree Decomposition of Graph G.

Table 1: Cost, validation and symbol functions for leaf bag $\{F\}$ for 2-rainbow domination

F	s	C	E
$\{1,2\}$	undefined	2	1
$\{\}$	undefined	0	∞
$\{1\}$	undefined	1	1
$\{2\}$	undefined	1	1
$\{*\}$	$\{1,2\}$	0	1

We also run the algorithm to compute the 3-rainbow domination for the graph G which gives $\gamma_{r,3} = 5$ with 3-rainbow function f_3 defined as follows

$$\begin{array}{lll} f_3(A) = \{1,2\}, & f_3(B) = \emptyset, & f_3(C) = \{3\}, \\ f_3(D) = \{1\}, & f_3(E) = \emptyset, & f_3(F) = \{3\}. \end{array} \tag{28}$$

The values of cost, validation and symbol functions for leaf bag $\{D\}$, forget bag $\{C\}$, introduce bag $\{C, D\}$ and join bag $\{A, C\}$ are given in Tables 5, 6, 7 and 8 respectively.

Finally, we run the algorithm to compute the weighted 2-rainbow domination for the graph G with respect to the weights in Table 9. The algorithm returns $\gamma_{wr,2} = 8$ with weighted 2-rainbow function $f_{w,2}$ defined as follows

$$\begin{array}{lll} f(A) = \{1,2\}, & f(B) = \emptyset, & f(C) = \emptyset, \\ f(D) = \{1\}, & f(E) = \emptyset, & f(F) = \{1\}. \end{array} \tag{29}$$

The values of cost, validation and symbol functions for leaf bag $\{F\}$, forget bag $\{E\}$, introduce bag $\{E, F\}$ and join bag $\{A, C\}$ are given in Tables 10, 11, 12 and 13 respectively.

Table 2: Cost, validation and symbol functions for forget bag $\{E\}$ for 2-rainbow domination

E	s	C	E
$\{1,2\}$	undefined	3	1
$\{\}$	undefined	2	1
$\{1\}$	undefined	2	1
$\{2\}$	undefined	2	1
$\{*\}$	$\{1\}$	1	1

Table 3: Cost, validation and symbol functions for introduce bag $\{E, F\}$ for 2-rainbow domination

E	s	F	s	C	E
$\{1\}$	undefined	$\{2\}$	undefined	2	1
$\{1,2\}$	undefined	$\{1,2\}$	undefined	4	1
$\{\}$	undefined	$\{\}$	undefined	0	∞
$\{\}$	undefined	$\{1\}$	undefined	1	∞
$\{*\}$	$\{1,2\}$	$\{*\}$	undefined	0	1
$\{1\}$	undefined	$\{1,2\}$	undefined	3	1
$\{*\}$	$\{1,2\}$	$\{\}$	undefined	0	∞
$\{\}$	undefined	$\{*\}$	undefined	0	∞
$\{2\}$	undefined	$\{*\}$	undefined	1	1
$\{1\}$	undefined	$\{*\}$	undefined	1	1
$\{2\}$	undefined	$\{1\}$	undefined	2	1
$\{1,2\}$	undefined	$\{*\}$	undefined	2	1
$\{*\}$	undefined	$\{1,2\}$	undefined	2	1
$\{1,2\}$	undefined	$\{1\}$	undefined	3	1
$\{1\}$	undefined	$\{1\}$	undefined	2	1
$\{\}$	undefined	$\{1,2\}$	undefined	2	1
$\{1\}$	undefined	$\{\}$	undefined	1	∞
$\{2\}$	undefined	$\{1,2\}$	undefined	3	1
$\{\}$	undefined	$\{2\}$	undefined	1	∞
$\{1,2\}$	undefined	$\{\}$	undefined	2	∞
$\{2\}$	undefined	$\{2\}$	undefined	2	1
$\{*\}$	$\{1\}$	$\{2\}$	undefined	1	1
$\{*\}$	$\{2\}$	$\{1\}$	undefined	1	1
$\{1,2\}$	undefined	$\{2\}$	undefined	3	1
$\{2\}$	undefined	$\{\}$	undefined	1	∞

Table 4: Cost, validation and symbol functions for join bag $\{A, C\}$ for 2-rainbow domination

A	s	C	s	C	E
$\{1\}$	undefined	$\{2\}$	undefined	4	1
$\{1,2\}$	undefined	$\{1,2\}$	undefined	6	1
$\{\}$	undefined	$\{\}$	undefined	5	1
$\{\}$	undefined	$\{1\}$	undefined	5	1
$\{*\}$	undefined	$\{*\}$	undefined	3	1
$\{1\}$	undefined	$\{1,2\}$	undefined	5	1
$\{*\}$	undefined	$\{\}$	undefined	4	1
$\{2\}$	undefined	$\{1\}$	undefined	5	1
$\{2\}$	undefined	$\{*\}$	undefined	4	1
$\{1\}$	undefined	$\{*\}$	undefined	3	1
$\{\}$	undefined	$\{*\}$	undefined	4	1
$\{1\}$	undefined	$\{1\}$	undefined	4	1
$\{*\}$	undefined	$\{1,2\}$	undefined	5	1
$\{1,2\}$	undefined	$\{1\}$	undefined	5	1
$\{1,2\}$	undefined	$\{*\}$	undefined	4	1
$\{\}$	undefined	$\{1,2\}$	undefined	5	1
$\{1\}$	undefined	$\{\}$	undefined	4	1
$\{2\}$	undefined	$\{1,2\}$	undefined	6	1
$\{\}$	undefined	$\{2\}$	undefined	5	1
$\{1,2\}$	undefined	$\{\}$	undefined	4	1
$\{2\}$	undefined	$\{2\}$	undefined	5	1
$\{*\}$	undefined	$\{2\}$	undefined	4	1
$\{*\}$	undefined	$\{1\}$	undefined	4	1
$\{1,2\}$	undefined	$\{2\}$	undefined	5	1
$\{2\}$	undefined	$\{\}$	undefined	5	1

Table 5: Cost, validation and symbol functions for leaf bag $\{D\}$ for 3-rainbow domination

D	s	C	E
$\{1,2,3\}$	undefined	3	1
$\{1,2\}$	undefined	2	1
$\{1,3\}$	undefined	2	1
$\{2\}$	undefined	1	1
$\{1\}$	undefined	1	1
$\{3\}$	undefined	1	1
$\{2,3\}$	undefined	2	1
$\{\}$	undefined	0	∞
$\{*\}$	$\{1,2,3\}$	0	1

Table 6: Cost, validation and symbol functions for forget bag $\{C\}$ for 3-rainbow domination

C	s	C	E
$\{1,2,3\}$	undefined	4	1
$\{1,2\}$	undefined	3	1
$\{1,3\}$	undefined	3	1
$\{2\}$	undefined	2	1
$\{1\}$	undefined	2	1
$\{3\}$	undefined	2	1
$\{*\}$	$\{1,2\}$	1	1
$\{\}$	undefined	3	1
$\{2,3\}$	undefined	3	1

Table 7: Cost, validation and symbol functions for introduce bag $\{C, D\}$ for 3-rainbow domination

C	s	D	s	C	E
$\{1\}$	undefined	$\{2\}$	undefined	2	1
$\{1,2\}$	undefined	$\{1,2\}$	undefined	4	1
$\{*\}$	$\{1,2,3\}$	$\{\}$	undefined	0	∞
$\{1,2\}$	undefined	$\{1,2,3\}$	undefined	5	1
$\{3\}$	undefined	$\{2,3\}$	undefined	3	1
$\{1,2,3\}$	undefined	$\{1,3\}$	undefined	5	1
$\{3\}$	undefined	$\{3\}$	undefined	2	1
$\{1,3\}$	undefined	$\{*\}$	undefined	2	1
$\{1\}$	undefined	$\{1,2\}$	undefined	3	1
$\{2,3\}$	undefined	$\{1,2\}$	undefined	4	1
$\{2\}$	undefined	$\{*\}$	undefined	1	1
$\{1,2,3\}$	undefined	$\{1\}$	undefined	4	1
$\{2,3\}$	undefined	$\{3\}$	undefined	3	1
$\{\}$	undefined	$\{1,3\}$	undefined	2	∞
$\{1,3\}$	undefined	$\{1\}$	undefined	3	1
$\{*\}$	$\{1,2,3\}$	$\{*\}$	undefined	0	1
$\{1\}$	undefined	$\{3\}$	undefined	2	1
$\{1\}$	undefined	$\{1\}$	undefined	2	1
$\{1,2\}$	undefined	$\{*\}$	undefined	2	1
$\{1\}$	undefined	$\{1,2,3\}$	undefined	4	1
$\{\}$	undefined	$\{\}$	undefined	0	∞
$\{*\}$	$\{1,2\}$	$\{3\}$	undefined	1	1
$\{1,3\}$	undefined	$\{1,3\}$	undefined	4	1
$\{*\}$	$\{2\}$	$\{1,3\}$	undefined	2	1
$\{1,2\}$	undefined	$\{1\}$	undefined	3	1
$\{1,2,3\}$	undefined	$\{*\}$	undefined	3	1
$\{2\}$	undefined	$\{1\}$	undefined	2	1
$\{2\}$	undefined	$\{1,3\}$	undefined	3	1
$\{3\}$	undefined	$\{1\}$	undefined	2	1
$\{1,2,3\}$	undefined	$\{2,3\}$	undefined	5	1
$\{*\}$	$\{3\}$	$\{1,2\}$	undefined	2	1
$\{\}$	undefined	$\{1,2,3\}$	undefined	3	1
$\{\}$	undefined	$\{3\}$	undefined	1	∞
$\{\}$	undefined	$\{1\}$	undefined	1	∞
$\{1,2\}$	undefined	$\{2,3\}$	undefined	4	1
$\{1\}$	undefined	$\{*\}$	undefined	1	1
$\{1,2\}$	undefined	$\{1,3\}$	undefined	4	1
$\{1,2,3\}$	undefined	$\{1,2,3\}$	undefined	6	1
$\{2\}$	undefined	$\{3\}$	undefined	2	1
$\{1,3\}$	undefined	$\{\}$	undefined	2	∞
$\{1,2,3\}$	undefined	$\{3\}$	undefined	4	1
$\{1,3\}$	undefined	$\{1,2,3\}$	undefined	5	1
$\{3\}$	undefined	$\{1,2,3\}$	undefined	4	1
$\{2,3\}$	undefined	$\{2,3\}$	undefined	4	1
$\{1,3\}$	undefined	$\{3\}$	undefined	3	1
$\{2,3\}$	undefined	$\{1,3\}$	undefined	4	1
$\{2\}$	undefined	$\{1,2,3\}$	undefined	4	1
$\{2,3\}$	undefined	$\{*\}$	undefined	2	1
$\{1,2,3\}$	undefined	$\{1,2\}$	undefined	5	1
$\{1\}$	undefined	$\{1,3\}$	undefined	3	1
$\{1\}$	undefined	$\{2,3\}$	undefined	3	1
$\{3\}$	undefined	$\{*\}$	undefined	1	1
$\{2,3\}$	undefined	$\{2\}$	undefined	3	1
$\{2,3\}$	undefined	$\{1\}$	undefined	3	1
$\{1,2,3\}$	undefined	$\{2\}$	undefined	4	1
$\{\}$	undefined	$\{1,2\}$	undefined	2	∞
$\{*\}$	undefined	$\{1,2,3\}$	undefined	3	1
$\{1,2\}$	undefined	$\{3\}$	undefined	3	1
$\{3\}$	undefined	$\{2\}$	undefined	2	1
$\{3\}$	undefined	$\{1,3\}$	undefined	3	1
$\{1\}$	undefined	$\{\}$	undefined	1	∞
$\{\}$	undefined	$\{*\}$	undefined	0	∞
$\{2\}$	undefined	$\{1,2\}$	undefined	3	1
$\{\}$	undefined	$\{2\}$	undefined	1	∞
$\{2\}$	undefined	$\{2,3\}$	undefined	3	1
$\{3\}$	undefined	$\{1,2\}$	undefined	3	1
$\{1,3\}$	undefined	$\{1,2\}$	undefined	4	1
$\{1,2\}$	undefined	$\{\}$	undefined	2	∞
$\{*\}$	$\{1\}$	$\{2,3\}$	undefined	2	1
$\{1,3\}$	undefined	$\{2,3\}$	undefined	4	1
$\{2\}$	undefined	$\{2\}$	undefined	2	1
$\{1,3\}$	undefined	$\{2\}$	undefined	3	1
$\{*\}$	$\{1,3\}$	$\{2\}$	undefined	1	1
$\{*\}$	$\{2,3\}$	$\{1\}$	undefined	1	1
$\{2,3\}$	undefined	$\{\}$	undefined	2	∞
$\{3\}$	undefined	$\{\}$	undefined	1	∞
$\{1,2\}$	undefined	$\{2\}$	undefined	3	1
$\{1,2,3\}$	undefined	$\{\}$	undefined	3	∞
$\{2,3\}$	undefined	$\{1,2,3\}$	undefined	5	1
$\{2\}$	undefined	$\{\}$	undefined	1	∞
$\{\}$	undefined	$\{2,3\}$	undefined	2	∞

Table 8: Cost, validation and symbol functions for join bag $\{A, C\}$ for 3-rainbow domination

A	s	C	s	C	E
$\{1\}$	undefined	$\{2\}$	undefined	5	1
$\{3\}$	undefined	$\{2,3\}$	undefined	6	1
$\{*\}$	undefined	$\{\}$	undefined	5	1
$\{1,2\}$	undefined	$\{1,2,3\}$	undefined	7	1
$\{1,2,3\}$	undefined	$\{2,3\}$	undefined	7	1
$\{1,2,3\}$	undefined	$\{1,3\}$	undefined	7	1
$\{3\}$	undefined	$\{3\}$	undefined	5	1
$\{1,3\}$	undefined	$\{*\}$	undefined	5	1
$\{1\}$	undefined	$\{1,2\}$	undefined	6	1
$\{2,3\}$	undefined	$\{1,2\}$	undefined	7	1
$\{\}$	undefined	$\{1,2\}$	undefined	5	1
$\{1,2,3\}$	undefined	$\{1\}$	undefined	6	1
$\{1,2\}$	undefined	$\{1,2\}$	undefined	6	1
$\{\}$	undefined	$\{1,3\}$	undefined	7	1
$\{1,3\}$	undefined	$\{1\}$	undefined	6	1
$\{1\}$	undefined	$\{3\}$	undefined	5	1
$\{1,2\}$	undefined	$\{*\}$	undefined	4	1
$\{1\}$	undefined	$\{1,2,3\}$	undefined	7	1
$\{*\}$	undefined	$\{1,2,3\}$	undefined	6	1
$\{\}$	undefined	$\{\}$	undefined	7	1
$\{*\}$	undefined	$\{3\}$	undefined	4	1
$\{1,3\}$	undefined	$\{1,3\}$	undefined	7	1
$\{*\}$	undefined	$\{1,3\}$	undefined	5	1
$\{\}$	undefined	$\{1,2,3\}$	undefined	6	1
$\{1,2,3\}$	undefined	$\{*\}$	undefined	5	1
$\{2\}$	undefined	$\{1\}$	undefined	5	1
$\{2\}$	undefined	$\{1,3\}$	undefined	6	1
$\{3\}$	undefined	$\{1\}$	undefined	5	1
$\{3\}$	undefined	$\{\}$	undefined	6	1
$\{*\}$	undefined	$\{1,2\}$	undefined	5	1
$\{1,2\}$	undefined	$\{1\}$	undefined	5	1
$\{\}$	undefined	$\{3\}$	undefined	6	1
$\{\}$	undefined	$\{1\}$	undefined	6	1
$\{1\}$	undefined	$\{*\}$	undefined	4	1
$\{1,2\}$	undefined	$\{1,3\}$	undefined	6	1
$\{1,2,3\}$	undefined	$\{1,2,3\}$	undefined	8	1
$\{2\}$	undefined	$\{3\}$	undefined	5	1
$\{*\}$	undefined	$\{1\}$	undefined	4	1
$\{1,2,3\}$	undefined	$\{3\}$	undefined	6	1
$\{1,3\}$	undefined	$\{1,2,3\}$	undefined	8	1
$\{3\}$	undefined	$\{1,2,3\}$	undefined	7	1
$\{*\}$	undefined	$\{*\}$	undefined	3	1
$\{1,3\}$	undefined	$\{3\}$	undefined	6	1
$\{2\}$	undefined	$\{*\}$	undefined	4	1
$\{2,3\}$	undefined	$\{1,3\}$	undefined	7	1
$\{1\}$	undefined	$\{1\}$	undefined	5	1
$\{2,3\}$	undefined	$\{*\}$	undefined	5	1
$\{1,2,3\}$	undefined	$\{1,2\}$	undefined	7	1
$\{1\}$	undefined	$\{1,3\}$	undefined	6	1
$\{1\}$	undefined	$\{2,3\}$	undefined	6	1
$\{3\}$	undefined	$\{*\}$	undefined	4	1
$\{2,3\}$	undefined	$\{2\}$	undefined	6	1
$\{2,3\}$	undefined	$\{1\}$	undefined	6	1
$\{1,2,3\}$	undefined	$\{2\}$	undefined	6	1
$\{2,3\}$	undefined	$\{3\}$	undefined	6	1
$\{2\}$	undefined	$\{1,2,3\}$	undefined	7	1
$\{1,2\}$	undefined	$\{3\}$	undefined	5	1
$\{3\}$	undefined	$\{2\}$	undefined	5	1
$\{2,3\}$	undefined	$\{1,2,3\}$	undefined	8	1
$\{3\}$	undefined	$\{1,3\}$	undefined	6	1
$\{1\}$	undefined	$\{\}$	undefined	6	1
$\{\}$	undefined	$\{*\}$	undefined	5	1
$\{2\}$	undefined	$\{1,2\}$	undefined	6	1
$\{\}$	undefined	$\{2\}$	undefined	6	1
$\{2\}$	undefined	$\{2,3\}$	undefined	6	1
$\{1,2\}$	undefined	$\{2,3\}$	undefined	6	1
$\{1,3\}$	undefined	$\{1,2\}$	undefined	7	1
$\{1,2\}$	undefined	$\{\}$	undefined	5	1
$\{*\}$	undefined	$\{2,3\}$	undefined	5	1
$\{1,3\}$	undefined	$\{2,3\}$	undefined	7	1
$\{2\}$	undefined	$\{2\}$	undefined	5	1
$\{1,3\}$	undefined	$\{2\}$	undefined	6	1
$\{*\}$	undefined	$\{2\}$	undefined	4	1
$\{1,3\}$	undefined	$\{\}$	undefined	7	1
$\{2,3\}$	undefined	$\{\}$	undefined	7	1
$\{1,2\}$	undefined	$\{2\}$	undefined	5	1
$\{1,2,3\}$	undefined	$\{\}$	undefined	5	1
$\{3\}$	undefined	$\{1,2\}$	undefined	6	1
$\{2\}$	undefined	$\{\}$	undefined	6	1
$\{\}$	undefined	$\{2,3\}$	undefined	7	1
$\{2,3\}$	undefined	$\{2,3\}$	undefined	7	1

Table 9: Weights for vertices of graph G where the i^{th} row corresponds to the weight of color i for the vertex of corresponding column

	A	B	C	D	E	F
1	2	3	2	2	3	1
2	3	1	1	2	2	3

Table 10: Cost, validation and symbol functions for leaf bag $\{F\}$ for weighted 2-rainbow domination

F	s	C	E
$\{1,2\}$	undefined	4	1
$\{\}$	undefined	0	∞
$\{1\}$	undefined	1	1
$\{2\}$	undefined	3	1
$\{*\}$	$\{1,2\}$	0	1

Table 11: Cost, validation and symbol functions for forget bag $\{E\}$ for weighted 2-rainbow domination

E	s	C	E
$\{1,2\}$	undefined	6	1
$\{\}$	undefined	4	1
$\{1\}$	undefined	4	1
$\{2\}$	undefined	3	1
$\{*\}$	$\{2\}$	1	1

Table 12: Cost, validation and symbol functions for introduce bag $\{E, F\}$ for weighted 2-rainbow domination

E	s	F	s	C	E
$\{1\}$	undefined	$\{2\}$	undefined	6	1
$\{1,2\}$	undefined	$\{1,2\}$	undefined	9	1
$\{\}$	undefined	$\{\}$	undefined	0	∞
$\{\}$	undefined	$\{1\}$	undefined	1	∞
$\{*\}$	$\{1,2\}$	$\{*\}$	undefined	0	1
$\{1\}$	undefined	$\{1,2\}$	undefined	7	1
$\{*\}$	$\{1,2\}$	$\{\}$	undefined	0	∞
$\{\}$	undefined	$\{*\}$	undefined	0	∞
$\{2\}$	undefined	$\{*\}$	undefined	2	1
$\{1\}$	undefined	$\{*\}$	undefined	3	1
$\{2\}$	undefined	$\{1\}$	undefined	3	1
$\{1,2\}$	undefined	$\{*\}$	undefined	5	1
$\{*\}$	undefined	$\{1,2\}$	undefined	4	1
$\{1,2\}$	undefined	$\{1\}$	undefined	6	1
$\{1\}$	undefined	$\{1\}$	undefined	4	1
$\{\}$	undefined	$\{1,2\}$	undefined	4	1
$\{1\}$	undefined	$\{\}$	undefined	3	∞
$\{2\}$	undefined	$\{1,2\}$	undefined	6	1
$\{\}$	undefined	$\{2\}$	undefined	3	∞
$\{1,2\}$	undefined	$\{\}$	undefined	5	∞
$\{2\}$	undefined	$\{2\}$	undefined	5	1
$\{*\}$	$\{1\}$	$\{2\}$	undefined	3	1
$\{*\}$	$\{2\}$	$\{1\}$	undefined	1	1
$\{1,2\}$	undefined	$\{2\}$	undefined	8	1
$\{2\}$	undefined	$\{\}$	undefined	2	∞

Table 13: Cost, validation and symbol functions for join bag $\{A, C\}$ for weighted 2-rainbow domination

A	s	C	s	C	E
$\{1\}$	undefined	$\{2\}$	undefined	8	1
$\{1,2\}$	undefined	$\{1,2\}$	undefined	11	1
$\{\}$	undefined	$\{\}$	undefined	10	1
$\{\}$	undefined	$\{1\}$	undefined	7	1
$\{*\}$	undefined	$\{*\}$	undefined	5	1
$\{1\}$	undefined	$\{1,2\}$	undefined	10	1
$\{*\}$	undefined	$\{\}$	undefined	7	1
$\{2\}$	undefined	$\{1\}$	undefined	8	1
$\{2\}$	undefined	$\{*\}$	undefined	6	1
$\{1\}$	undefined	$\{*\}$	undefined	7	1
$\{\}$	undefined	$\{*\}$	undefined	8	1
$\{1\}$	undefined	$\{1\}$	undefined	9	1
$\{*\}$	undefined	$\{1,2\}$	undefined	8	1
$\{1,2\}$	undefined	$\{1\}$	undefined	10	1
$\{1,2\}$	undefined	$\{*\}$	undefined	8	1
$\{\}$	undefined	$\{1,2\}$	undefined	8	1
$\{1\}$	undefined	$\{\}$	undefined	7	1
$\{2\}$	undefined	$\{1,2\}$	undefined	9	1
$\{\}$	undefined	$\{2\}$	undefined	9	1
$\{1,2\}$	undefined	$\{\}$	undefined	8	1
$\{2\}$	undefined	$\{2\}$	undefined	7	1
$\{*\}$	undefined	$\{2\}$	undefined	6	1
$\{*\}$	undefined	$\{1\}$	undefined	7	1
$\{1,2\}$	undefined	$\{2\}$	undefined	9	1
$\{2\}$	undefined	$\{\}$	undefined	8	1