

# Minimal Roman Dominating Functions: Extensions and Enumeration

Faisal N. Abu-Khizam<sup>1</sup>[0000-0001-5221-8421], Henning  
Fernau<sup>2</sup>[0000-0002-4444-3220], and Kevin Mann<sup>2</sup>[0000-0002-0880-2513]

<sup>1</sup> Department of Computer Science and Mathematics  
Lebanese American University, Beirut, Lebanon.

faisal.abukhizam@lau.edu.lb

<sup>2</sup> Universität Trier, Fachber. 4 – Abteilung Informatikwissenschaften  
54286 Trier, Germany.

{fernau,mann}@uni-trier.de

**Abstract** Roman domination is one of the many variants of domination that keeps most of the complexity features of the classical domination problem. We prove that Roman domination behaves differently in two aspects: enumeration and extension. We develop non-trivial enumeration algorithms for minimal Roman domination functions with polynomial delay and polynomial space. Recall that the existence of a similar enumeration result for minimal dominating sets is open for decades. Our result is based on a polynomial-time algorithm for EXTENSION ROMAN DOMINATION: Given a graph  $G = (V, E)$  and a function  $f : V \rightarrow \{0, 1, 2\}$ , is there a minimal Roman domination function  $\tilde{f}$  with  $f \leq \tilde{f}$ ? Here,  $\leq$  lifts  $0 < 1 < 2$  pointwise; minimality is understood in this order. Our enumeration algorithm is also analyzed from an input-sensitive viewpoint, leading to a run-time estimate of  $\mathcal{O}(1.9332^n)$  for graphs of order  $n$ ; this is complemented by a lower bound example of  $\Omega(1.7441^n)$ .

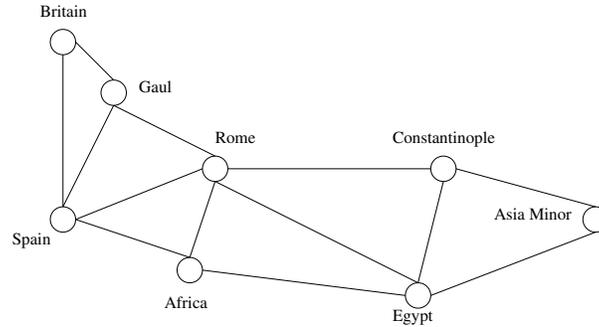
**Keywords:** Roman domination · Extension problems · Enumeration.

## 1 Introduction

This paper combines four lines of research: (a) studying variations of domination problems, here the Roman domination [17,21,28]; (b) input-sensitive enumeration of minimal solutions, a topic that has drawn attention in particular from people also interested in domination problems [2,18,19,26,27]; (c) related to (and motivated by) enumeration, extension problems have been introduced and studied in particular in the context of domination problems<sup>3</sup> in [3,9,11,12,32,33,40]: is a given set a subset of any minimal dominating set?; (d) the HITTING SET TRANSVERSAL PROBLEM is the question if all minimal hitting sets of a hypergraph can be enumerated with polynomial delay (or even output-polynomial)

<sup>3</sup> Historically, a logical extension problem [10] should be mentioned, as it has led to [40, Théorème 2.16], dealing with an extension variant of 3-HITTING SET; also see [40, Proposition 3.39] concerning implications for EXTENSION DOMINATING SET.

only: this question is open for four decades by now and is equivalent to several enumeration problems in logic, database theory and also to enumerating minimal dominating sets in graphs, see [20,22,25,31]. By way of contrast, we show that enumerating all minimal Roman domination functions is possible with polynomial delay, a result which is quite surprising in view of the general similarities between the complexities of domination and Roman domination problems.



**Figure 1.** The Roman Empire in the times of Constantine

ROMAN DOMINATION comes with a nice (hi)story: namely, it should reflect the idea of how to secure the Roman Empire by positioning the armies (legions) on the various parts of the Empire in a way that either (1) a specific region  $r$  is also the location of at least one army or (2) one region  $r'$  neighboring  $r$  has two armies, so that  $r'$  can afford sending off one army to the region  $r$  (in case of an attack) without diminishing self-defense capabilities. More specifically, Emperor Constantine had a look at a map of his empire (as discussed in [49], also see Fig. 1).<sup>4</sup> Related is the island hopping strategy pursued by General MacArthur in World War II in the Pacific theater to gradually increase the US-secured areas.

ROMAN DOMINATION has received a lot of attention from the algorithmic community in the past 15 years [4,15,21,24,35,36,39,43,44,47]. Relevant to our paper is the development of exact algorithms for ROMAN DOMINATION: combining ideas from [35,46], an  $\mathcal{O}(1.5014^n)$  exponential-time and -space algorithm (making use of known SET COVER algorithms via a transformation to PARTIAL DOMINATING SET) was presented in [48]. In [14,16,23,30,34,38,37,42,50,51,52], more combinatorial studies can be found. This culminated in a chapter on Roman domination, stretching over nearly 50 pages in the monograph [29]. There is also an interesting link to the notion of a *differential* of a graph, introduced in [41], see [7], also adding further algorithmic thoughts, as expressed in [1,5,6]. For instance, in [5] an exponential-time algorithm was published, based on a direct Measure-and-Conquer approach.

<sup>4</sup> The historical background is also nicely described in the online Johns Hopkins Magazine, visit <http://www.jhu.edu/~jhumag/0497web/locate3.html> to pre-view [45].

One of the ideas leading to the development of the area of *extension problems* (as described in [12]) was to cut branches of search trees as early as possible, in the following sense: to each node of the search tree, a so-called pre-solution  $U$  can be associated, and it is asked if it is possible to extend  $U$  to a meaningful solution  $S$ . In the case of DOMINATING SET, this means that  $U$  is a set of vertices and a ‘meaningful solution’ is an inclusion-wise minimal dominating set. Notice that such a strategy would work not only for computing smallest dominating sets, but also for computing largest minimal dominating set, or for counting minimal solutions, or for enumerating them. Alas, as it has been shown by many examples, extension problems turn out to be quite hard problems. Even for combinatorial problems whose standard decision version is solvable in polynomial time (for instance, EDGE COVER), its extension variation is NP-hard. In such a case, the approach might still be viable, as possibly parameterized algorithms exist with respect to the parameter ‘pre-solution size’. This would be interesting, as this parameter is small when a big gain can be expected in terms of an early abort of a search tree branch. In particular for EXTENSION DOMINATING SET, this hope is not fulfilled. To the contrary, with this parameterization  $|U|$ , EXTENSION DOMINATING SET is one of the few problems known to be complete for the parameterized complexity class W[3], as shown in [8].

With an appropriate definition of the notion of minimality, ROMAN DOMINATION becomes one of the few examples where the hope seeing extension variants being efficiently solvable turns out to be true, as we will show in this paper. This is quite a surprising result, as in nearly any other way, ROMAN DOMINATION behaves most similar to DOMINATING SET. Together with its combinatorial foundations (a characterization of minimal Roman domination functions), this constitutes the first main result of this paper. The main algorithmic exploit of this result is a non-trivial polynomial-space enumeration algorithm for minimal Roman domination functions that guarantees polynomial delay only, which is the second main result of the paper. As mentioned above, the corresponding question for enumerating minimal dominating sets is open since decades, and we are not aware of any other modification of the concept of domination that seems to preserve any other of the difficulties of DOMINATING SET, like classical or parameterized or approximation complexities, apart from the complexity of extension and enumeration. Our enumeration algorithm is a branching algorithm that we analyzed with a simple Measure & Conquer approach, yielding a running time of  $\mathcal{O}(1.9332^n)$ , which also gives an upper bound on the number of minimal Roman dominating functions of an  $n$ -vertex graph. This result is complemented by a simple example that proves a lower bound of  $\Omega(1.7441^n)$  for the number of minimal Roman dominating functions on graphs of order  $n$ .

## 2 Definitions

Let  $\mathbb{N} = \{1, 2, 3, \dots\}$  be the set of positive integers. For  $n \in \mathbb{N}$ , let  $[n] = \{m \in \mathbb{N} \mid m \leq n\}$ . We only consider undirected simple graphs. Let  $G = (V, E)$  be a graph. For  $U \subseteq V$ ,  $G[U]$  denotes the graph induced by  $U$ . For  $v \in V$ ,  $N_G(v) := \{u \in V \mid$

$\{u, v\} \in E\}$  denotes the *open neighborhood* of  $v$ , while  $N_G[v] := N_G(v) \cup \{v\}$  is the *closed neighborhood* of  $v$ . We extend such set-valued functions  $X : V \rightarrow 2^V$  to  $X : 2^V \rightarrow 2^V$  by setting  $X(U) = \bigcup_{u \in U} X(u)$ . Subset  $D \subseteq V$  is a *dominating set*, or ds for short, if  $N_G[D] = V$ . For  $D \subseteq V$  and  $v \in D$ , define the *private neighborhood* of  $v \in V$  with respect to  $D$  as  $P_{G,D}(v) := N_G[v] \setminus N_G[D \setminus \{v\}]$ . A function  $f : V \rightarrow \{0, 1, 2\}$  is called a *Roman dominating function*, or rdf for short, if for each  $v \in V$  with  $f(v) = 0$ , there exists a  $u \in N_G(v)$  with  $f(u) = 2$ . To simplify the notation, we define  $V_i(f) := \{v \in V \mid f(v) = i\}$  for  $i \in \{0, 1, 2\}$ . The *weight*  $w_f$  of a function  $f : V \rightarrow \{0, 1, 2\}$  equals  $|V_1(f)| + 2|V_2(f)|$ . The classical ROMAN DOMINATION problem asks, given  $G$  and an integer  $k$ , if there exists an rdf for  $G$  of weight at most  $k$ . Connecting to the original motivation,  $G$  models a map of regions, and if the region vertex  $v$  belongs to  $V_i$ , then we place  $i$  armies on  $v$ .

For the definition of the problem EXTENSION ROMAN DOMINATION, we need to define the order  $\leq$  on  $\{0, 1, 2\}^V$  first: for  $f, g \in \{0, 1, 2\}^V$ , let  $f \leq g$  if and only if  $f(v) \leq g(v)$  for all  $v \in V$ . In other words, we extend the usual linear ordering  $\leq$  on  $\{0, 1, 2\}$  to functions mapping to  $\{0, 1, 2\}$  in a pointwise manner. We call a function  $f \in \{0, 1, 2\}^V$  a *minimal Roman dominating function* if and only if  $f$  is a rdf and there exists no rdf  $g$ ,  $g \neq f$ , with  $g \leq f$ .<sup>5</sup> The weights of minimal rdf can vary considerably. Consider for example a star  $K_{1,n}$  with center  $c$ . Then,  $f_1(c) = 2$ ,  $f_1(v) = 0$  otherwise;  $f_2(v) = 1$  for all vertices  $v$ ;  $f_3(c) = 0$ ,  $f_3(u) = 2$  for one  $u \neq c$ ,  $f_3(v) = 1$  otherwise, define three minimal rdf with weights  $w_{f_1} = 2$ , and  $w_{f_2} = w_{f_3} = n + 1$ .

**Problem name:** EXTENSION ROMAN DOMINATION, or EXTRD for short  
**Given:** A graph  $G = (V, E)$  and a function  $f \in \{0, 1, 2\}^V$ .  
**Question:** Is there a minimal rdf  $\tilde{f} \in \{0, 1, 2\}^V$  with  $f \leq \tilde{f}$ ?

As our first main result, we are going to show that EXTRD can be solved in polynomial time in section 4. To this end, we need some understanding of the combinatorial nature of this problem, which we provide in section 3.

The second problem that we consider is that of enumeration, both from an output-sensitive and from an input-sensitive perspective.

**Problem name:** ROMAN DOMINATION ENUMERATION, or RDENUM for short  
**Given:** A graph  $G = (V, E)$ .  
**Task:** Enumerate all minimal rdf  $f \in \{0, 1, 2\}^V$  of  $G$ !

From an output-sensitive perspective, it is interesting to perform this enumeration without repetitions and with polynomial delay, which means that there is a polynomial  $p$  such that between the consecutive outputs of any two minimal rdf of a graph of order  $n$  that are enumerated, no more than  $p(n)$  time elapses,

<sup>5</sup> According to [29], this notion of minimality for rdf was coined by Cockayne but then dismissed, as it does not give a proper notion of *upper Roman domination* number. However, in our context, this definition seems to be the most natural one, as it also perfectly fits the extension framework proposed in [13]. We will propose in section 7 yet another notion of minimal rdf that also fits the mentioned extension framework.

including the corner-cases at the beginning and at the end of the algorithm. From an input-sensitive perspective, we want to upper-bound the running time of the algorithm, measured against the order of the input graph. The obtained run-time bound should not be too different from known lower bounds, given by graph families where one can prove that a certain number of minimal rdf must exist. Our algorithm will be analyzed from both perspectives and achieves both goals. This is explained in section 5 and in section 6.

### 3 Properties of Minimal Roman Dominating Functions

**Theorem 1.** *Let  $G = (V, E)$  be a graph and  $f : V \rightarrow \{0, 1, 2\}$  be a minimal rdf. Then  $N_G[V_2(f)] \cap V_1(f) = \emptyset$  holds.*

*Proof.* Assume that there exists a  $\{u, v\} \in E$  with  $f(v) = 2$  and  $f(u) = 1$ . Let

$$\tilde{f} : V \rightarrow \{0, 1, 2\}, w \mapsto \begin{cases} f(w), & w \neq u \\ 0, & w = u \end{cases}$$

We show that  $\tilde{f}$  is a rdf, which contradicts the minimality of  $f$ , as  $\tilde{f} \leq f$  and  $\tilde{f}(u) < f(u)$  are given by construction. Consider  $w \in V_0(\tilde{f})$ . If  $w = u$ ,  $w$  is dominated by  $v$ , as  $\{u, v\} \in E$ . Consider  $w \neq u$ . Since  $f$  is a rdf and  $V_0(f) \cup \{u\} = V_0(\tilde{f})$ , there exists a  $t \in N_G[w] \cap V_2(f)$ . By construction of  $\tilde{f}$ ,  $V_2(f) = V_2(\tilde{f})$  holds. This implies  $N_G[w] \cap V_2(\tilde{f}) \neq \emptyset$ . Hence,  $\tilde{f}$  is a rdf.  $\square$

**Theorem 2.** *Let  $G = (V, E)$  be a graph and  $f : V \rightarrow \{0, 1, 2\}$  be a minimal rdf. Then for all  $v \in V_2(f)$ ,  $P_{G[V_0(f) \cup V_2(f)], V_2(f)}(v) \not\subseteq \{v\}$  holds.*

*Proof.* Define  $G' := G[V \setminus V_1(f)] = G[V_0(f) \cup V_2(f)]$ . In contrast to the claim, assume that there exists a  $v \in V_2(f)$  with  $P_{G', V_2(f)}(v) \subseteq \{v\}$ . Define

$$\tilde{f} : V \rightarrow \{0, 1, 2\}, w \mapsto \begin{cases} f(w), & w \neq v \\ 1, & w = v \end{cases}$$

We show that  $\tilde{f}$  is a rdf, which contradicts the minimality of  $f$ , as  $\tilde{f} \leq f$  and  $\tilde{f}(v) < f(v)$  are given by construction. Let  $u \in V_0(\tilde{f}) = V_0(f)$ . We must show that some neighbor of  $u$  belongs to  $V_2(\tilde{f}) = V_2(f) \setminus \{v\}$ . Then,  $\tilde{f}$  is a rdf.

First, assume that  $u$  is a neighbor of  $v$ . By the choice of  $v$ ,  $u$  is not a private neighbor of  $v$ . Hence, there exists a  $w \in N_G[u] \cap (V_2(f) \setminus \{v\}) = N_G[u] \cap V_2(\tilde{f})$ . Secondly, if  $u \in V_0(\tilde{f})$  is not a neighbor of  $v$ , then there exists a  $w \in V_2(f) \setminus \{v\}$  that dominates  $u$ , i.e.,  $w \in N_G[u] \cap (V_2(f) \setminus \{v\}) = N_G[u] \cap V_2(\tilde{f})$ .  $\square$

As each  $v \in V_0(f)$  has to be dominated by a  $w \in V_2(f)$ , the next claim follows.

**Corollary 1.** *Let  $G = (V, E)$  be a graph and  $f \in \{0, 1, 2\}^V$  be a minimal rdf. Then,  $V_2 := V_2(f)$  is a minimal ds of  $G[N_G[V_2]]$ , with  $N_G[V_2] = V_0(f) \cup V_2$ .*

*Remark 1.* We can generalize the last statement as follows: Let  $G = (V, E)$  be a graph and  $f : V \rightarrow \{0, 1, 2\}$  be a minimal rdf. Let  $I \subseteq V_1(f)$  be an independent set in  $G$ . Then,  $V_2(f) \cup I$  is a minimal ds of  $G[V_0(f) \cup V_2(f) \cup I]$ . If  $I$  is a maximal independent set in  $G[V_1(f)]$ , then  $V_2(f) \cup I$  is a minimal ds of  $G[V_0(f) \cup V_2(f) \cup V_1(f)]$ .

This allows us to deduce the following characterization result.

**Theorem 3.** *Let  $G = (V, E)$  be a graph,  $f : V \rightarrow \{0, 1, 2\}$  and abbreviate  $G' := G[V_0(f) \cup V_2(f)]$ . Then,  $f$  is a minimal rdf if and only if the following conditions hold:*

1.  $N_G[V_2(f)] \cap V_1(f) = \emptyset$ ,
2.  $\forall v \in V_2(f) : P_{G', V_2(f)}(v) \not\subseteq \{v\}$ , also called privacy condition, and
3.  $V_2(f)$  is a minimal dominating set of  $G'$ .

*Proof.* The “only if” follows by Theorem 1, Theorem 2 and Corollary 1.

Let  $f$  be a function that fulfills the three conditions. Since  $V_2(f)$  is a dominating set on  $G'$ , for each  $u \in V_0(f)$ , there exists a  $v \in V_2(f) \cap N_G[u]$ . Therefore,  $f$  is a rdf. Let  $\tilde{f} : V \rightarrow \{0, 1, 2\}$  be a minimal rdf with  $\tilde{f} \leq f$ . Therefore,  $\tilde{f}$  (also) satisfies the three conditions by Theorem 1, Theorem 2 and Corollary 1. Assume that there exists a  $v \in V$  with  $\tilde{f}(v) < f(v)$ . Hence,  $V_2(\tilde{f}) \subseteq V_2(f) \setminus \{v\}$ .

**Case 1:**  $\tilde{f}(v) = 0, f(v) = 1$ . Therefore, there exists a  $u \in N_G(v)$  with  $f(u) \geq \tilde{f}(u) = 2$ . This contradicts Condition 1.

**Case 2:**  $\tilde{f}(v) \in \{0, 1\}, f(v) = 2$ . Let  $u \in N_G(v)$  with  $f(u) = 0$ . This implies  $\tilde{f}(u) = 0$  and

$$\emptyset \neq N_G[u] \cap V_2(\tilde{f}) \subseteq N_G[u] \cap V_2(f) \setminus \{v\}$$

holds. Therefore,  $N_G(v) \subseteq N_G[V_2(f) \setminus \{v\}]$ . This contradicts Condition 2.

Thus,  $\tilde{f} = f$  holds and  $f$  is minimal.  $\square$

We conclude this section with an upper bound on the size of  $V_2(f)$ .

**Lemma 1.** *Let  $G = (V, E)$  be a graph and  $f : V \rightarrow \{0, 1, 2\}$  be a minimal rdf. Then  $2|V_2(f)| \leq |V|$  holds.*

*Proof.* Consider a graph  $G = (V, E)$  and a minimal rdf  $f : V \rightarrow \{0, 1, 2\}$ . For each  $v \in V_2(f)$ , let  $P_f(v) = P_{G[V_0(f) \cup V_2(f)], V_2(f)}(v) \setminus \{v\} \subseteq V \setminus V_2(f)$ . By Theorem 2, these sets are not empty and, by definition, they do not intersect. Hence, we get:

$$|V| = |V_2(f)| + |V \setminus V_2(f)| \geq |V_2(f)| + \left| \bigcup_{v \in V_2(f)} P_f(v) \right| \geq 2|V_2(f)|.$$

Therefore, the claim is true.  $\square$

## 4 A Polynomial-time Algorithm for EXTRD

With Theorem 6, we can construct an algorithm that solves the problem EXTENSION ROMAN DOMINATION in polynomial time.

---

### Algorithm 1 Solving instances of EXTRD

---

```

1: procedure EXTRD SOLVER( $G, f$ )
   Input: A graph  $G = (V, E)$  and a function  $f: V \rightarrow \{0, 1, 2\}$ .
   Output: Is there a minimal Roman dominating function  $\tilde{f}$  with  $f \leq \tilde{f}$ ?
2:    $f := f$ .
3:    $M_2 := V_2(f)$ . { Invariant:  $M_2 = V_2(\tilde{f})$  }
4:    $M := M_2$ . { All  $v \in V_2(\tilde{f})$  are considered below; invariant:  $M \subseteq M_2$ . }
5:   while  $M \neq \emptyset$  do
6:     Choose  $v \in M$ . { Hence,  $\tilde{f}(v) = 2$ . }
7:     for  $u \in N(v)$  do
8:       if  $\tilde{f}(u) = 1$  then
9:          $\tilde{f}(u) := 2$ .
10:      Add  $u$  to  $M$  and to  $M_2$ .
11:     Delete  $v$  from  $M$ .
12:   for  $v \in M_2$  do
13:     if  $N_G(v) \subseteq N_G[M_2 \setminus \{v\}]$  then
14:       Return No.
15:   for  $v \in V \setminus N_G[M_2]$  do
16:      $\tilde{f}(v) := 1$ .
17:   Return Yes.

```

---

**Theorem 4.** *Let  $G = (V, E)$  be a graph and  $f: V \rightarrow \{0, 1, 2\}$ . For the inputs  $G, f$ , Algorithm 1 returns yes if and only if  $(G, f)$  is a yes-instance of EXTRD. In this case, the function  $\tilde{f}$  computed by Algorithm 1 is a minimal rdf.*

*Proof.* First observe that the invariants stated in Lines 3 and 4 of Algorithm 1 are true whenever entering or leaving the while-loop.

Let the answer of the algorithm be *yes* and  $\tilde{f}$  be the function computed by the algorithm. We will show that  $\tilde{f}$  satisfies the conditions formulated in Theorem 6.

Observing the if-condition in Line 8, clearly after the while-loop, no neighbor  $u$  of  $v \in V_2(\tilde{f})$  fulfills  $\tilde{f}(u) = 1$ . Hence,  $\tilde{f}$  satisfies Condition 1. If the function  $\tilde{f}$  would contradict Condition 2 of Theorem 6, then we would get to Line 14 and the algorithm would answer *no*. As we are considering a *yes*-answer of our algorithm, we can assume that this privacy condition holds after the for-loop of Line 12. We also can assume that  $M_2 = V_2(\tilde{f})$  is a minimal ds of the graph  $G[N_G[M_2]]$ . Otherwise, for such a  $v \in M_2$  and each  $u \in N_G(v)$ , there

would exist a  $w \in N_G[u] \cap (M_2 \setminus \{v\})$ . In this case, the algorithm would return *no* in Line 14. In the for-loop of Line 15, we update for all  $v \in V \setminus N_G[M_2]$  the value  $\tilde{f}(v)$  to 1. With the while-loop, this implies  $N_G[M_2] = V_0(\tilde{f}) \cup V_2(\tilde{f})$ . Therefore,  $V_2(\tilde{f})$  is a minimal ds of  $G[V_0(\tilde{f}) \cup V_2(\tilde{f})]$ . Since we do not update the values of  $\tilde{f}$  to two in this last for-loop, Condition 2 from Theorem 6 holds. By the while-loop and the for-loop starting in Line 15, it is trivial to see that Condition 1 also holds for the final  $\tilde{f}$ . We can now use Theorem 6 to see that  $\tilde{f}$  is a minimal rdf.

Since we never decrease  $\tilde{f}$  in this algorithm, starting with  $\tilde{f} = f$  in Line 2, we get  $f \leq \tilde{f}$ . Therefore,  $(G, f)$  is a *yes*-instance of EXTRD.

Now we assume that  $(G, f)$  is a *yes*-instance, but the algorithm returns *no*. Therefore, there exists a minimal rdf  $\bar{f}$  with  $f \leq \bar{f}$ . Since  $N_G[V_2(\bar{f})] \cap V_1(\bar{f}) = \emptyset$ ,  $\tilde{f} \leq \bar{f}$  holds for the function  $\tilde{f}$  in Line 12. This implies  $M_2 = V_2(\tilde{f}) \subseteq V_2(\bar{f})$ . The algorithm returns *no* if and only if there exists a  $v \in M_2$  with

$$N_G(v) \subseteq N_G[M_2 \setminus \{v\}] \subseteq N_G[V_2(\bar{f}) \setminus \{v\}].$$

Applying again Theorem 6, we see that  $\bar{f}$  cannot be a minimal rdf, contradicting our assumption.  $\square$

In Proposition 1, we prove that our algorithm needs polynomial time only.

**Proposition 1.** *Algorithm 1 runs in time cubic in the order of the input graph.*

*Proof.* Let  $G = (V, E)$  be the input graph. Define  $n = |V|$ . Up to Line 4, the algorithm can run in linear time. As each vertex can only be once in  $M$  and we look at the neighbors of each element in  $M$ , the while-loop runs in time  $\mathcal{O}(n^2)$ . In the for-loop starting in Line 12, we build for all  $v \in M_2$  the set  $N_G[M_2 \setminus \{v\}]$ . This needs  $\mathcal{O}(n^3)$  time. The other steps of this loop run in time  $\mathcal{O}(n^2)$ . The last for-loop requires linear time. Hence, the algorithm runs in time  $\mathcal{O}(n^3)$ .  $\square$

## 5 Enumerating Minimal RDF for General Graphs

For general graphs, our general combinatorial observations allow us to strengthen the (trivial)  $\mathcal{O}^*(3^n)$ -algorithm for enumerating all minimal rdf for graphs of order  $n$  down to  $\mathcal{O}^*(2^n)$ , as displayed in Algorithm 2. To understand the correctness of this enumeration algorithm, the following lemma is crucial.

**Lemma 2.** *Let  $G = (V, E)$  be a graph with  $V_2 \subseteq V$  such that  $P_{G, V_2}(v) \not\subseteq \{v\}$  for each  $v \in V_2$  holds. Then there exists exactly one minimal rdf  $f \in \{0, 1, 2\}^V$  with  $V_2 = V_2(f)$ . Algorithm 1 can calculate  $f$ .*

*Proof.* Define

$$f : V \rightarrow \{0, 1, 2\}, v \mapsto \begin{cases} 2, & v \in V_2 \\ 1, & v \notin N[V_2] \\ 0, & \text{otherwise} \end{cases}$$

---

**Algorithm 2** A simple enumeration algorithm for minimal rdf
 

---

```

1: procedure RD ENUMERATION( $G$ )
   Input: A graph  $G = (V, E)$ .
   Output: Enumeration of all minimal rdf  $f : V \rightarrow \{0, 1, 2\}$ .
2:   for all functions  $f : V \rightarrow \{1, 2\}$  do
3:     for all  $v \in V$  with  $f(v) = 1$  do
4:       if  $\exists u \in N_G(v) : f(u) = 2$  then
5:          $f(v) := 0$ .
6:       Build graph  $G'$  induced by  $f^{-1}(\{0, 2\}) = V_0(f) \cup V_2(f)$ .
7:       private-test := 1.
8:       for all  $v \in V$  with  $f(v) = 2$  do
9:         if  $P_{G', V_2(f)}(v) \subseteq \{v\}$  then
10:          private-test := 0.
11:      if private-test = 1 and if  $f^{-1}(2) = V_2(f)$  is a minimal ds of  $G'$  then
12:        Output the current function  $f : V \rightarrow \{0, 1, 2\}$ .
    
```

---

Hence,  $N_G[V_2] = V_2 \cup V_0(f)$ . With the assumption  $P_{G, V_2}(v) \not\subseteq \{v\}$ ,  $V_2$  is a minimal ds of  $G[V_2 \cup V_0(f)]$ . Furthermore,  $N_G[V_2] \cap V_1(f) = \emptyset$ . As  $V_2 = V_2(f)$ , all conditions of Theorem 6 hold and  $f$  is a minimal rdf.

Let  $\tilde{f} \in \{0, 1, 2\}^V$  be a minimal rdf with  $V_2 = V_2(\tilde{f})$ . If there exists some  $v \in V_0(f) \cap V_1(\tilde{f})$ , this contradicts Condition 1, as  $v \in N_G[V_2] = N_G[V_2(\tilde{f})]$ . Therefore,  $V_0(f) \subseteq V_0(\tilde{f})$  holds. By the assumption that  $\tilde{f}$  is a rdf, for each  $v \in V_0(\tilde{f})$  there exists a  $u \in V_2(\tilde{f}) \cap N[v] = V_2 \cap N[v]$ . This implies  $v \in N_G[V_2] \setminus V_2 = V_0(f)$ . Therefore,  $V_0(f) = V_0(\tilde{f})$  holds. This implies  $f = \tilde{f}$ .

Define:

$$\hat{f} : V \rightarrow \{0, 1, 2\}, v \mapsto \begin{cases} 2, & v \in V_2 \\ 0, & v \notin V_2 \end{cases}.$$

It is trivial to see that  $\hat{f} \leq f$ . By Theorem 4, Algorithm 1 returns *yes* for the input  $\hat{f}$ . Let  $\bar{f}$  be the minimal rdf produced by Algorithm 1, given  $\hat{f}$ . We want to show that  $V_2 = V_2(\bar{f})$ . We do this by looking at the steps of the algorithm. Since  $V_1(\hat{f}) = \emptyset$ , the algorithm never gets into the If-clause in Line 8. This is the only way to update a vertex to the value 2. Therefore,  $V_2 = V_2(\bar{f})$ .  $\square$

**Proposition 2.** *Let  $G = (V, E)$  be a graph. For minimal rdf  $f, g \in \{0, 1, 2\}^V$  with  $V_2(f) = V_2(g)$ , it holds  $f = g$ .*

*Proof.* By Theorem 2,  $V_2(f)$  fulfills the conditions of Lemma 2. Therefore, there exists a unique minimal rdf  $h \in \{0, 1, 2\}^V$  with  $V_2(h) = V_2(f) = V_2(g)$ . Thus,  $f = g = h$  holds.  $\square$

Hence, there is a bijection between the minimal rdf of a graph  $G = (V, E)$  and subsets  $V_2 \subseteq V$  that satisfy the condition of Lemma 2.

**Proposition 3.** *All minimal rdf of a graph of order  $n$  can be enumerated in time  $\mathcal{O}^*(2^n)$ .*

*Proof.* Consider Algorithm 2. The running time claim is obvious. The correctness of the algorithm is clear due to Theorem 6 and Lemma 2.  $\square$

The presented algorithm clearly needs polynomial space only, but it is less clear if it has polynomial delay. Below, we will present a branching algorithm that has both of these desirable properties, and moreover, its running time is below  $2^n$ . How good or bad such an enumeration is, clearly also depends on examples that provide a lower bound on the number of objects that are enumerated. The next lemma explains why the upper bounds for enumerating minimal rdf must be bigger than those for enumerating minimal dominating sets.

**Lemma 3.** *A disjoint collection of  $c$  cycles on five vertices yields a graph of order  $n = 5c$  that has  $(16)^c$  many minimal rdf.*

*Proof.* Let  $C_5$  be a cycle of length 5 with  $V(C_5) = \{v_1, \dots, v_5\}$  and  $E(C_5) = \{\{v_i, v_{i+1}\} \mid i \in [4]\} \cup \{\{v_1, v_5\}\}$ . For a  $f \in \{0, 1, 2\}^{V(C_5)}$  there are at least the following sixteen possibilities for  $(f(v_1), \dots, f(v_5))$ :

- zero occurrences of 2:  $(1, 1, 1, 1, 1)$ ;
- one occurrence of 2:  $(2, 0, 1, 1, 0)$  and four more cyclic shifts;
- two adjacent occurrences of 2:  $(2, 2, 0, 1, 0)$  and four more cyclic shifts;
- two non-adjacent occurrences of 2:  $(2, 0, 2, 0, 0)$  and four more cyclic shifts.

Therefore, there are at least 16 minimal rdf on  $C_5$ . To prove that these are all the minimal rdf, we use Lemma 1, which implies  $|V_2(f)| \leq \frac{|V(C_5)|}{2} < 3$ . Hence, the number of minimal rdf on  $C_5$  is at most  $\binom{5}{0} + \binom{5}{1} + \binom{5}{2} = 16$ .  $\square$

**Corollary 2.** *There are graphs of order  $n$  that have at least  $\sqrt[5]{16}^n \in \Omega(1.7441^n)$  many minimal rdf.*

We checked with the help of a computer program that there are no other connected graphs of order at most eight that yield (by taking disjoint unions) a bigger lower bound.

## 6 A Refined Enumeration Algorithm

In this section, we are going to prove the following result, which can be considered as the second main result of this paper.

**Theorem 5.** *There is a polynomial-space algorithm that enumerates all minimal rdf of a given graph of order  $n$  with polynomial delay and in time  $\mathcal{O}^*(1.9332^n)$ .*

Notice that this is in stark contrast to what is known about the enumeration of minimal dominating sets, or, equivalently, of minimal hitting sets in hypergraphs. Here, it is a long-standing open problem if minimal hitting sets in hypergraphs can be enumerated with polynomial delay.

The remainder of this section is dedicated to describing the proof of this theorem.

### 6.1 A bird's eye view on the algorithm

As all along the search tree, from inner nodes we branch into the two cases if a certain vertex is assigned 2 or not, it is clear that (with some care concerning the final processing in leaf nodes) no minimal rdf is output twice. Hence, there is no need for the branching algorithm to store intermediate results to test (in a final step) if any solution was generated twice. Therefore, our algorithm needs only polynomial space, as detailed in Proposition 7 and Corollary 4.

Because we have a polynomial-time procedure that can test if a certain given pre-solution can be extended to a minimal rdf, we can build (a slightly modified version of) this test into an enumeration procedure, hence avoiding unnecessary branchings. Therefore, whenever we start with our binary branching, we know that at least one of the search tree branches will return at least one new minimal rdf. Hence, we will not move to more than  $N$  nodes in the search tree before outputting a new minimal rdf, where  $N$  is upper-bounded by twice the order of the input graph. This is the basic explanation for the claimed polynomial delay, as detailed in Proposition 5.

Let  $G = (V, E)$  be a graph. Let us call a(ny partial) function

$$f : V \longrightarrow \{0, 1, 2, \bar{1}, \bar{2}\}$$

a *generalized Roman domination function*, or grdf for short. Extending previously introduced notation, let  $\bar{V}_1(f) = \{x \in V \mid f(x) = \bar{1}\}$ , and  $\bar{V}_2(f) = \{x \in V \mid f(x) = \bar{2}\}$ . A vertex is said to be *active* if it has not been assigned a value (yet) under  $f$ ; these vertices are collected in the set  $A(f)$ . Hence, for any grdf  $f$ , we have the partition  $V = A(f) \cup V_0(f) \cup V_1(f) \cup V_2(f) \cup \bar{V}_1(f) \cup \bar{V}_2(f)$ .

After performing a branching step, followed by an exhaustive application of the reduction rules, any grdf  $f$  considered in our algorithm always satisfies the following **(grdf) invariants**:

1.  $\forall x \in \bar{V}_1(f) \cup V_0(f) \exists y \in N_G(x) : y \in V_2(f)$ ,
2.  $\forall x \in V_2(f) : N_G(x) \subseteq \bar{V}_1(f) \cup V_0(f) \cup V_2(f)$ ,
3.  $\forall x \in V_1(f) : N_G(x) \subseteq \bar{V}_2(f) \cup V_0(f) \cup V_1(f)$ ,
4. if  $\bar{V}_2(f) \neq \emptyset$ , then  $A(f) \cup \bar{V}_1(f) \neq \emptyset$ .<sup>6</sup>

<sup>6</sup> This condition assumes that our graphs have non-empty vertex sets.

For the extension test, we will therefore consider the function  $\hat{f} : V \rightarrow \{0, 1, 2\}$  that is derived from a grdf  $f$  as follows:

$$\hat{f}(v) = \begin{cases} 0, & \text{if } v \in A(f) \cup V_0(f) \cup \overline{V_1}(f) \cup \overline{V_2}(f) \\ 1, & \text{if } v \in V_1(f) \\ 2, & \text{if } v \in V_2(f) \end{cases}$$

The enumeration algorithm uses a combination of reduction and branching rules, starting with the nowhere defined function  $f_\perp$ , so that  $A(f_\perp) = V$ . The schematics of the algorithm is shown in Algorithm 3. To understand the algorithm, call an rdf  $g$  as *consistent* with a grdf  $f$  if  $g(v) = 2$  implies  $v \in A(f) \cup V_2(f) \cup \overline{V_1}(f)$  and  $g(v) = 1$  implies  $v \in A(f) \cup V_1(f) \cup \overline{V_2}(f)$  and  $g(v) = 0$  implies  $v \in A(f) \cup V_0(f) \cup \overline{V_1}(f) \cup \overline{V_2}(f)$ . Below, we start with presenting some reduction rules, which also serve as (automatically applied) actions at each branching step, whenever applicable. The branching itself always considers a most attractive vertex  $v$  and either gets assigned 2 or not. The running time analysis will be performed with a measure-and-conquer approach. Our simple measure is defined by  $\mu(G, f) = |A(f)| + \omega_1|\overline{V_1}(f)| + \omega_2|\overline{V_2}(f)| \leq |V|$  for some constants  $\omega_1$  and  $\omega_2$  that have to be specified later.

The measure never increases when applying a reduction rule.

---

**Algorithm 3** A refined enumeration algorithm for minimal rdf

---

```

1: procedure REFINED RD ENUMERATION( $G, f$ )
   Input: A graph  $G = (V, E)$ , a grdf  $f : V \rightarrow \{0, 1, 2, \bar{1}, \bar{2}\}$ .
   Assumption: There exists at least one minimal rdf consistent with  $f$ .
   Output: Enumeration of all minimal rdf consistent with  $f$ .
2:   if  $f$  is everywhere defined and  $f(V) \subseteq \{0, 1, 2\}$  then
3:     Output  $f$  and return.
4:   { We know that  $A(f) \cup \overline{V_1}(f) \neq \emptyset$ . }
5:   Pick a vertex  $v \in A(f) \cup \overline{V_1}(f)$  of highest priority for branching.
6:    $f_2 := f$ ;  $f_2(v) := 2$ .
7:   Exhaustively apply reduction rules to  $f_2$ . { Invariants are valid for  $f_2$ . }
8:   if GENEXTRD SOLVER  $(G, \hat{f}_2, \overline{V_2}(f_2))$  then
9:     REFINED RD ENUMERATION  $(G, f_2)$ .
10:   $f_{\bar{2}} := f$ ; if  $v \in A(f)$  then  $f_{\bar{2}}(v) := \bar{2}$  else  $f_{\bar{2}}(v) := 0$ .
11:  Exhaustively apply reduction rules to  $f_{\bar{2}}$ . { Invariants are valid for  $f_{\bar{2}}$ . }
12:  if GENEXTRD SOLVER  $(G, \hat{f}_{\bar{2}}, \overline{V_2}(f_{\bar{2}}))$  then
13:    REFINED RD ENUMERATION  $(G, f_{\bar{2}})$ .

```

---

We are now presenting details of the algorithm and its analysis.

## 6.2 How to achieve polynomial delay and polynomial space

In this section, we need a slight modification of the problem EXTRD in order to cope with pre-solutions. In this version, we add to an instance, usually specified by  $G = (V, E)$  and  $f : V \rightarrow \{0, 1, 2\}$ , a set  $\bar{V}_2 \subseteq V$  with  $V_2(f) \cap \bar{V}_2 = \emptyset$ . The question is if there exists a minimal RDF  $\hat{f}$  with  $f \leq \hat{f}$  and  $V_2(\hat{f}) \cap \bar{V}_2 = \emptyset$ . We call this problem a *generalized rdf extension problem*, or GENEXTRD for short. In order to solve this problem, we modify Algorithm 1 to cope with GENEXTRD by adding an if-clause after Line 8 that asks if  $u \in \bar{V}_2$ . If this is true, then the algorithm returns *no*, because it is prohibited that  $\hat{f}(u)$  is set to 2, while this is necessary for minimal rdf, as there is a vertex  $v$  in the neighborhood of  $u$  such that  $\hat{f}(v)$  has been set to 1. We call this algorithm GENEXTRD SOLVER.

**Lemma 4.** *Let  $G = (V, E)$  be a graph,  $f : V \rightarrow \{0, 1, 2\}$  be a function and  $\bar{V}_2 \subseteq V$  be a set with  $V_2(f) \cap \bar{V}_2 = \emptyset$ . GENEXTRD SOLVER gives the correct answer when given the GENEXTRD instance  $(G, f, \bar{V}_2)$ .*

*Proof.* In Algorithm 1, the only statement where we give a vertex the value 2 is in the if-clause of Line 8. The modified version would first check if the vertex is in  $\bar{V}_2$ . If this is true, there will be no minimal RDF solving this problem. Namely, if we give the vertex the value 2, this would contradict  $V_2(\hat{f}) \cap \bar{V}_2 = \emptyset$ . If the value stays 1, this would contradict Condition 1. By Theorem 4,  $\hat{f}$  will be a minimal rdf with  $V_2(\hat{f}) \cap \bar{V}_2 = \emptyset$  if the algorithm returns *yes*.

Assume there exists a minimal RDF  $\bar{f}$  with  $V_2(\bar{f}) \cap \bar{V}_2 = \emptyset$  but the algorithm returns *no*. First we assume that *no* is returned by the new if-clause. This implies that a vertex  $u \in V_1(f)$  is in the neighborhood of a vertex  $v \in V$  that has to have the value 2 in any minimal rdf that is bigger than  $f$  (because Theorem 4). But this would lead to a similar contradiction as above.

Therefore, the answer *no* has to be returned in Line 14. That would contradict Condition 2 or Condition 3. Thus the algorithm would correctly return *yes*.  $\square$

Let  $f$  be a generalized rdf at any moment of the branching algorithm. The next goal is to show that GENEXTRD SOLVER could tell us in polynomial time if there exists a minimal rdf that could be enumerated by the branching algorithm from this point on.

**Proposition 4.** *Let  $G = (V, E)$  be a graph,  $f : V \rightarrow \{0, 1, 2, \bar{1}, \bar{2}\}$  be a partial function. Then, GENEXTRD SOLVER correctly answers if there exists some minimal rdf  $g : V \rightarrow \{0, 1, 2\}$  that is consistent with  $f$  when GENEXTRD SOLVER is given the instance  $(G, \hat{f}, \bar{V}_2(f))$ .*

The following proof makes use of the grdf invariants presented above, which are only formally proved to hold in the next subsection, in Proposition 8.

*Proof.* We have to show two assertions: (1) If GENEXTRD SOLVER answers *yes* on the instance  $(G, \hat{f}, \bar{V}_2(f))$ , then there exists a minimal rdf  $g : V \rightarrow \{0, 1, 2\}$

that is consistent with  $f$ . (2) If there exists a minimal rdf  $g : V \rightarrow \{0, 1, 2\}$  that is consistent with  $f$ , then GENEXTRD SOLVER answers *yes* on the instance  $(G, \hat{f}, \overline{V_2}(f))$ .

ad (1): Assume GENEXTRD SOLVER found a minimal rdf  $g$  such that  $\hat{f} \leq g$  and  $\overline{V_2}(g) \cap \overline{V_2}(f) = \emptyset$ . Let  $v \notin A(f)$ . First assume that  $g(v) = 2$ . Clearly, vertices in  $V_2(f) = V_2(\hat{f})$  do not get changed, as they cannot be made bigger. Hence, assume  $v \notin V_2(f)$  exists with  $g(v) = 2$ . As GENEXTRD SOLVER will only explicitly set the value 2 for vertices originally set to 1 (by their  $f$ -assignment) that are in the neighborhood of vertices already set to value 2 and that do not belong to  $\overline{V_2}(f)$ , we have to reason about a possible  $v \in V_1(f) = V_1(\hat{f})$ . By the third grdf invariant, the neighborhood of  $v$  contains no vertex from  $V_2(f) = V_2(\hat{f})$ , so that the case of some  $v \notin V_2(f)$  with  $g(v) = 2$  can be excluded.

Secondly, assume that  $g(v) = 1$ . The case  $v \in V_1(f)$  is not critical, and  $v \in V_2(f)$  is not possible, as reasoned above. Notice that  $g(v) = 1$  was set in the last lines of the algorithm. In particular,  $N_G(v) \cap V_2(g) = \emptyset$ . As  $V_2(f) \subseteq V_2(g)$ , also  $N_G(v) \cap V_2(f) = \emptyset$ . By the first grdf invariant,  $v \notin \overline{V_1}(f) \cup V_0(f)$ . Hence, only  $v \in \overline{V_2}(f)$  remains as a possibility.

Thirdly, assume that  $g(v) = 0$ . As  $f(v) \in \{1, 2\}$  is clearly impossible,  $v \in V_0(f) \cup \overline{V_1}(f) \cup \overline{V_2}(f)$  must follow. Hence,  $g$  is consistent with  $f$ .

ad (2): Assume that there exists a minimal rdf  $g : V \rightarrow \{0, 1, 2\}$  that is consistent with  $f$ . We have to prove that  $\hat{f} \leq g$  and that  $V_2(g) \cap \overline{V_2}(f) = \emptyset$ , because then GENEXTRD SOLVER will correctly answer *yes* by Lemma 4. For  $g(v) = 2$ , then consistency implies  $f(v) \neq \bar{2}$ , and trivially  $\hat{f}(v) \leq g(v)$ . For  $g(v) = 1$ ,  $v \in A(f) \cup V_1(f) \cup \overline{V_2}(f)$ , and hence  $\hat{f}(v) \in \{0, 1\}$ , so that  $\hat{f}(v) \leq g(v)$ . If  $g(v) = 0$ , then  $v \in A(f) \cup V_0(f) \cup \overline{V_1}(f) \cup \overline{V_2}(f) = V_0(\hat{f})$ , so that again  $\hat{f}(v) \leq g(v)$ .  $\square$

An important consequence of the previous proposition is stated next. Notice that our algorithm behaves quite differently from what is known about algorithms the enumerate minimal ds.

**Proposition 5.** *Procedure REFINED RD ENUMERATION, on input  $G = (V, E)$ , outputs functions  $f : V \rightarrow \{0, 1, 2\}$  with polynomial delay.*

*Proof.* Although the reduction rules are only stated in the next subsection, it is not hard to see by quickly browsing through them that they can be implemented to run in polynomial time. Moreover, GENEXTRD SOLVER runs in polynomial time. Hence, all work done in an inner node of the search tree needs polynomial time only. By the properties of GENEXTRD SOLVER, the search tree will never continue branching if no outputs are to be expected that are consistent with the current grdf (that is associated to that inner node). Hence, a run of the procedure REFINED RD ENUMERATION dives straight through setting more and more values of a grdf, until it is everywhere defined with values from  $\{0, 1, 2\}$ , and then it returns from the recursion and dives down the next promising branch. Clearly, the length of any search tree branch is bounded by  $|V|$ , so that at most  $2^{|V|}$  many inner nodes are visited between any two outputs. This also holds at

the very beginning (i.e., only polynomial time will elapse until the first function is output) and at the very end (i.e., only polynomial time will be spent after outputting the last function). This proves the claimed polynomial delay.  $\square$

**Proposition 6.** *Procedure REFINED RD ENUMERATION correctly enumerates all minimal rdf that are consistent with the input grdf, assuming that at least one consistent rdf exists.*

*Proof.* As there exists a consistent rdf, outputting the input function is correct if the input function is already an rdf, which is checked, as we test if the given grdf is everywhere defined and has only images in  $\{0, 1, 2\}$ . Also, before REFINED RD ENUMERATION is called recursively, we explicitly check if at least one consistent rdf exists.

If the input grdf  $f$  is not everywhere defined or if  $\overline{V}_2(f) \cup \overline{V}_1(f) \neq \emptyset$ , then  $A(f) \cup \overline{V}_1(f) \neq \emptyset$  by the fourth grdf invariant. Hence, whenever REFINED RD ENUMERATION is called recursively,  $A(f) \cup \overline{V}_1(f) \neq \emptyset$  holds, as these calls are immediately after applying all reduction rules exhaustively.

Hence, by induction and based on the previous propositions, REFINED RD ENUMERATION correctly enumerates all minimal rdf that are consistent with the input grdf.  $\square$

**Corollary 3.** *Procedure REFINED RD ENUMERATION correctly enumerates all minimal rdf of a given graph  $G = (V, E)$  when provided with the nowhere defined grdf  $f_\perp$ .*

*Proof.* Due to the previous proposition, it is sufficient to notice that all minimal rdf are consistent with  $f_\perp$  and that the function that is constant 1 is a minimal rdf consistent with  $f_\perp$ .  $\square$

**Proposition 7.** *Procedure REFINED RD ENUMERATION never enumerates any minimal rdf consistent with the given grdf on the input graph  $G = (V, E)$  twice.*

*Proof.* Notice that the enumeration algorithm always branches by deciding for a vertex  $v$  from  $A(f) \cup \overline{V}_1(f)$ , where  $f$  is the current grdf, if  $f(v)$  is updated to 2 or not, which means that either  $v \in A(f)$  is set to  $\overline{2}$ , or  $v \in \overline{V}_1$  is set to 0. Then, reduction rules may apply, but they never change the decision if, in a certain branch of the search tree,  $f(v) = 2$  is either true or false. Moreover, they never set any vertex to 2. As any minimal rdf that is ever output in a certain branch will be consistent with the grdf  $f$  associated to an inner node of the search tree, Procedure REFINED RD ENUMERATION never enumerates any minimal rdf twice.  $\square$

An important consequence of the last claim is that there is no need to store all output functions in order to finally parse them to see into enumerating any of them only once.

**Corollary 4.** *Algorithm REFINED RD ENUMERATION lists all minimal rdf consistent with the given grdf on the input graph  $G = (V, E)$  without repetitions and in polynomial space.*

### 6.3 Details on reductions and branchings

For the presentation of the following rules, we assume that  $G = (V, E)$  and a grdf  $f$  is given. We also assume that the rules are executed exhaustively in the given order.

**Reduction Rule LPN (Last Potential Private Neighbor).** If  $v \in V_2(f)$  satisfies  $|N_G(v) \cap (\overline{V_2}(f) \cup A(f))| = 1$ , then set  $f(x) = 0$  for  $\{x\} = N_G(v) \cap (\overline{V_2}(f) \cup A(f))$ .

**Reduction Rule  $V_0$ .** Let  $v \in V_0(f)$ . Assume there exists a unique  $u \in V_2(f) \cap N_G(v)$ . Moreover, assume that for all  $x \in N_G(u) \cap (V_0(f) \cup \overline{V_1}(f) \cup \overline{V_2}(f))$ ,  $|N_G(x) \cap V_2(f)| \geq 2$  if  $x \neq v$ . Then, for any  $w \in N_G(v) \cap A(f)$ , set  $f(w) = \overline{2}$  and for any  $w \in N_G(v) \cap \overline{V_1}(f)$ , set  $f(w) = 0$ .

**Reduction Rule  $V_1$ .** Let  $v \in V_1(f)$ . For any  $w \in N_G(v) \cap A(f)$ , set  $f(w) = \overline{2}$ . For any  $w \in N_G(v) \cap \overline{V_1}(f)$ , set  $f(w) = 0$ .

**Reduction Rule  $V_2$ .** Let  $v \in V_2(f)$ . For any  $w \in N_G(v) \cap A(f)$ , set  $f(w) = \overline{1}$ . For any  $w \in N_G(v) \cap \overline{V_2}(f)$ , set  $f(w) = 0$ .

**Reduction Rule NPD (No Potential Domination).** If  $v \in \overline{V_2}(f)$  satisfies  $N_G(v) \subseteq \overline{V_2}(f) \cup V_0(f) \cup V_1(f)$ , then set  $f(v) = 1$  (this also applies to isolated vertices in  $\overline{V_2}(f)$ ).

**Reduction Rule NPN (No Private Neighbor).** If  $v \in A(f)$  satisfies  $N_G(v) \subseteq V_0 \cup \overline{V_1}(f)$ , then set  $f(v) = \overline{2}$  (this also applies to isolated vertices in  $A(f)$ ).

**Reduction Rule Isolate.** If  $A(f) = \emptyset$  and if  $v \in \overline{V_1}(f)$  satisfies  $N_G(v) \cap \overline{V_2}(f) = \emptyset$ , then set  $f(v) = 0$ .

**Reduction Rule Edges.** If  $u, v \in \overline{V_2}(f) \cup V_0(f) \cup V_1(f)$  and  $e = uv \in E$ , then remove the edge  $e$  from  $G$ .

In the following, we first take care of the claimed grdf invariants.

**Proposition 8.** *After exhaustively executing the proposed reduction rules, as indicated in Algorithm 3, the claimed grdf invariants are maintained.*

*Proof.* We argue for the correctness of the grdf invariants by induction one by one. Notice that (trivially) all invariants hold if we start the algorithm with the nowhere defined grdf.

1.  $\forall x \in \overline{V_1}(f) \cup V_0(f) \exists y \in N_G(x) : y \in V_2(f)$ .  
 We need to show that  $N_G(x) \cap V_2(f) \neq \emptyset$  holds for each  $x \in V_0(f) \cup \overline{V_1}(f)$ . For the inductive step, we only have to look at the reduction rules, since the branching rules only change the value to 0 if the vertex was already in  $\overline{V_1}(f)$ . For each reduction rule where we set a value of a vertex to 0 or to  $\overline{1}$ , there exists a vertex in the neighborhood with value 2, which is seen as follows.
  - LPN: We explicitly consider  $x \in N_G(V_2(f))$  only to be set by  $f(x) = 0$ .
  - $V_0$  &  $V_1$ : We only set  $w$  to 0 if it has been in  $\overline{V_1}$ . By induction hypothesis,  $w$  has a neighbor in  $V_2(f)$ .
  - $V_2$ : We explicitly consider  $w \in N_G(V_2(f))$  only to be set to 0 or to  $\overline{1}$ .
  - Isolate: Only vertices from  $\overline{V_1}(f)$  are set to 0; apply induction hypothesis.

2.  $\forall x \in V_2(f) : N_G(x) \subseteq \overline{V_1}(f) \cup V_0(f) \cup V_2(f)$ .

This property can only be invalidated if new vertices get the value 2 or if vertices from  $\overline{V_1}(f) \cup V_0(f) \cup V_2(f)$  are changed to a value other than this or if edges are deleted (as vertices are never deleted). The only way in which a vertex gets the value 2 is by branching. Immediately afterwards, the reduction rules are executed: LPN and  $V_2$  will install the invariant for the neighborhood of  $v$ . No reduction rule ever changes the value of a vertex from  $V_0(f) \cup V_2(f)$ , while vertices from  $\overline{V_1}(f)$  might be set to 0 or 2. The Reduction Rule Edges deletes no edges incident to vertices from  $V_2(f)$ .

3.  $\forall x \in V_1(f) : N_G(x) \subseteq \overline{V_2}(f) \cup V_0(f) \cup V_1(f)$ .

The invariant is equivalent to the following three conditions: (a)  $N(V_1(f)) \cap A(f) = \emptyset$ , (b)  $N(V_1(f)) \cap \overline{V_1}(f) = \emptyset$  and (c)  $N(V_1(f)) \cap V_2(f) = \emptyset$ . Conditions (a) and (b) are taken care of by Reduction Rule  $V_1$ . Condition (c) immediately follows by the already proven second invariant.

4. If  $\overline{V_2}(f) \neq \emptyset$ , then  $A(f) \cup \overline{V_1}(f) \neq \emptyset$ .

Consider some  $x \in \overline{V_2}(f)$ . By the second invariant,  $N_G(x) \cap V_2(f) = \emptyset$ . By the Reduction Rule Edges,  $N_G(x) \cap (\overline{V_2}(f) \cup V_0(f) \cup V_1(f)) = \emptyset$ . As the Reduction Rule NPD did not apply, the only possible neighbors of  $x$  are in  $A(f) \cup \overline{V_1}(f)$ .  $\square$

We have now to show the *soundness* of the proposed reduction rules. In the context of enumerating minimal rdf, this means the following: if  $f, f'$  are grdf of  $G = (V, E)$  before or after applying any of the reduction rules, then  $g$  is a minimal rdf that is consistent with  $f$  if and only if it is consistent with  $f'$ .

**Proposition 9.** *All proposed reduction rules are sound.*

*Proof.* For the soundness of the reduction rules, we also need the invariants proven to be correct in Proposition 8. We now prove the soundness of each reduction rule, one at a time.

If possible, we apply Reduction Rule LPN first. Consider  $v \in V_2(f)$  with  $\{x\} = N_G(v) \cap (\overline{V_2}(f) \cup A(f))$ . Before the branching step, due to the second invariant, neighbors of  $V_2(f)$ -vertices are either in  $V_0(f)$ ,  $V_2(f)$  or in  $\overline{V_1}(f)$ . As no reduction rule adds a vertex to  $V_2(f)$ ,  $v$  must have been put into  $V_2(f)$  by the last branching step. By the first invariant, we know that all  $y \in N_G(v) \cap (\overline{V_1}(f) \cup V_0(f))$  are dominated by vertices different from  $v$ . As  $v \in V_2(f)$ , it still needs a private neighbor to dominate. As  $N_G(v) \cap (\overline{V_1}(f) \cup A(f))$  contains one element  $x$  only, setting  $f(x) = 0$  is enforced for any minimal rdf (see Theorem 2).

Next, we prove Reduction Rule  $V_0$ . We consider  $v \in V_0(f)$  and  $u \in V$  with  $\{u\} = V_2(f) \cap N_G(v)$ . We can use the rule, since  $u$  needs a private neighbor which can only be  $v$ , by the assumption that every other neighbor of  $u$  is dominated at least twice. To maintain the property that  $v$  is a private neighbor of  $u$ , each  $A(f)$ -neighbor of  $v$  is set to  $\bar{2}$  and each  $\overline{V_1}(f)$ -neighbor of  $v$  is set to 0. This annotates the fact that any minimal rdf  $g$  compatible with  $f$  will satisfy  $(g(x) = 2 \implies x = u)$  for each  $x \in N_G(v)$ .

The soundness of Reduction Rule  $V_1$  and Reduction Rule  $V_2$  mainly follows from Theorem 1.

Coming to the Reduction Rule NPD, notice that setting  $f(v) = 0$  would necessitate  $f(u) = 2$  for some neighbor  $u$  of  $v$ , which is impossible.

For the Reduction Rule NPN, we use the fact that  $N_G(v) \cap V_2(f) \neq \emptyset$  holds for each  $v \in V_0(f) \cup \overline{V}_1(f)$ , which is the first invariant.<sup>7</sup> This implies that there is no element left for  $v$  to dominate (therefore it has no private neighbor except itself). Thus, if  $v$  has the value 2, then it would contradict with Theorem 2.

For the soundness of Reduction Rule Isolate, we note that, since  $A(f)$  is empty,  $v \in \overline{V}_1(f)$  can only have neighbors in  $V_1(f) \cup V_2(f) \cup \overline{V}_1(f)$ , as  $\overline{V}_2(f)$ -neighbors are prohibited. As Reduction Rule  $V_1$  was (if possible) executed before,  $N_G(v) \cap V_1(f) = \emptyset$  holds. Therefore,  $v \in \overline{V}_1(f)$  would not have a private neighbor if  $f(v) = 2$ , cf. the first invariant.<sup>8</sup>

Finally, the soundness of Reduction Rule Edges follows trivially from the fact that an element of  $\overline{V}_2(f) \cup V_0(f) \cup V_1(f)$  cannot dominate any vertex in  $\overline{V}_2(f) \cup V_0(f) \cup V_1(f)$ . Hence, a minimal rdf  $g$  is consistent with  $f$  if and only if it is consistent with  $f'$ , obtained by applying the Reduction Rule Edges to  $f$ .  $\square$

In order to fully understand Algorithm 3, we need to describe priorities for branching. We describe these priorities in the following in decreasing order for a vertex  $v \in A(f) \cup \overline{V}_1(f)$ .

1.  $v \in A(f)$  and  $|N_G(v) \cap (A(f) \cup \overline{V}_2(f))| \geq 2$ ;
2. any  $v \in A(f)$ ;
3. any  $v \in \overline{V}_1(f)$ , preferably if  $|N_G(v) \cap \overline{V}_2(f)| \neq 2$ .

These priorities also split the run of our algorithm into phases, as whenever the algorithm was once forced to pick a vertex according to some lower priority, there will be never again the chance to pick a vertex of higher priority thereafter. It is useful to collect some **phase properties** that instances must satisfy after leaving Phase  $i$ , determined by applying the  $i^{\text{th}}$  branching priority.

- Before entering any phase, there are no edges between vertices  $u, v$  if  $u, v \in V_0(f) \cup V_1(f) \cup \overline{V}_2(f)$  or if  $u \in V_2(f)$  and  $v \in \overline{V}_2(f) \cup A(f)$  or if  $u \in V_1(f)$  and  $v \in \overline{V}_1(f) \cup A(f)$ , as we can assume that the reduction rules have been exhaustively applied.
- After leaving the first phase, any active vertex with an active neighbor is either pendant or has only further neighbors from  $\overline{V}_1(f) \cup V_0(f)$ .
- After leaving the second phase,  $A(f) = \emptyset$  and  $N_G(\overline{V}_2(f)) \subseteq \overline{V}_1(f)$ . Moreover, any vertex  $x \in \overline{V}_2(f)$  has neighbors in  $\overline{V}_1(f)$ .
- After leaving the third phase,  $A(f) = \overline{V}_2(f) = \overline{V}_1(f) = \emptyset$ , so that  $f$  is a Roman dominating function.

<sup>7</sup> More precisely, we also have to check that the possibly newly introduced vertices in  $V_0(f)$  or  $\overline{V}_1(f)$  by the branching or by the reduction rules up to this point do maintain the invariant, but this is nothing else then re-checking the induction step of the correctness proof of this invariant, see the proof of Proposition 8.

<sup>8</sup> Again, one has to partially follow the induction step of the proof of Proposition 8.

Phase #	Branching vector
1.1	$(1 - \omega_2, 3 - 2\omega_1)$
1.2	$(1 - \omega_2, 1 + 2\omega_2)$
1.3	$(1 - \omega_2, 2 + \omega_2 - \omega_1)$
2.1 & 2.2.b	$(1 - \omega_2, 2)$
2.2.a	$(1 - \omega_2, 1 + \omega_2 + \omega_1)$
2.2.c	$(1 + \omega_2, 1)$
3.1	$(\omega_1, \omega_1 + 3\omega_2)$
3.2.a	$(\omega_1, 2\omega_1 + \omega_2)$
3.2.b & 3.3.a	$(\omega_1 + \omega_2, \omega_1 + \omega_2)$
3.3.b	$(2\omega_1 + 2\omega_2, 2\omega_1 + 2\omega_2, 2\omega_1 + 2\omega_2, 2\omega_1 + 2\omega_2)$

**Table 1.** The branching vectors of different branching scenarios of the enumeration algorithm for listing all minimal Roman domination functions of a given graph

**Proposition 10.** *The phase properties hold.*

*Proof.* We are considering the items on the list separately.

- Reduction Rule Edges shows the first claim. Reduction Rules  $V_2$  and  $V_1$  show the other two claims.
- By the branching condition, we know that after leaving the first phase,  $|N_G(v) \cap (A(f) \cup \overline{V_2}(f))| < 2$  for any active vertex  $v$ . Since  $v$  has a neighbor in  $A(f)$  (say  $u$ ) this implies that there cannot be any other neighbor in  $A(f) \cup \overline{V_2}(f)$ . Moreover, by the Reduction Rule  $V_1$ ,  $N_G(v) \cap V_1(f) = \emptyset$ , and by the Reduction Rule  $V_2$ ,  $N_G(v) \cap V_2(f) = \emptyset$ . Hence,  $N_G(v) \setminus \{u\} \subseteq \overline{V_1}(f) \cup V_0(f)$ .
- The second phase branches on each  $v \in A(f)$ . Therefore, it ends if  $A(f) = \emptyset$ . Let  $v \in \overline{V_2}(f)$ . By Reduction Rule Edge, we get  $N_G(v) \cap (\overline{V_2}(f) \cup V_0 \cup V_1) = \emptyset$ . Reduction Rule  $V_2$  implies that  $v$  does not have a neighbor in  $V_2(f)$ . Therefore we get  $N_G(v) \subseteq \overline{V_1}(f)$ . If  $N_G(v)$  is empty, Reduction Rule NPD will be triggered. Therefore,  $N_G(v)$  has at least one element.
- The third phase runs on the vertices in  $\overline{V_1}(f)$ . Thus,  $\overline{V_1}(f) = \emptyset$  holds at the end of this phase. Since we never put a vertex into  $A(f)$  again,  $A(f)$  is empty. To get  $\overline{V_2}(f) = \emptyset$ , we can use the same argumentation as in the property before, since a vertex goes only from  $A(f)$  to  $\overline{V_2}(f)$ .  $\square$

#### 6.4 A Measure & Conquer Approach

We now present the branching analysis, classified by the described branching priorities. We summarize a list of all resulting branching vectors in Table 1.

**Branching in Phase 1.** We are always branching on an active vertex  $v$ . In the first branch, we set  $f(v) = 2$ . In the second branch, we set  $f(v) = \bar{2}$ . In the first branch, in addition the Reduction Rule  $V_2$  triggers at least twice. In order to

determine a lower bound on the branching vector, we describe three worst-case scenarios; all other reductions of the measure can be only better.

1.  $|N_G(v) \cap A(f)| = 2$ , i.e.,  $v$  has two active neighbors  $x$  and  $y$ . The corresponding recurrence is:  $T(\mu) = T(\mu - (1 - \omega_2)) + T(\mu - (1 + 2(1 - \omega_1)))$ , as either  $v$  moves from  $A(f)$  to  $V_2(f)$  and  $x, y$  move from  $A(f)$  to  $\bar{V}_1(f)$ , or  $v$  itself moves from  $A(f)$  to  $\bar{V}_2(f)$ . The branching vector is hence:  $(1 - \omega_2, 3 - 2\omega_1)$ , as noted in the first row of Table 1.
2.  $|N_G(v) \cap \bar{V}_2(f)| = 2$ . The corresponding recurrence is:  $T(\mu) = T(\mu - (1 - \omega_2)) + T(\mu - (1 + 2\omega_2))$ , see the second row of Table 1.
3.  $|N_G(v) \cap A(f)| = 1$  and  $|N_G(v) \cap \bar{V}_2(f)| = 1$ , leading to  $T(\mu) = T(\mu - (1 - \omega_2)) + T(\mu - (1 + (1 - \omega_1) + \omega_2))$ , see Table 1, third row.

**Branching in Phase 2.** We are again branching on an active vertex  $v$ . By Reduction Rule NPN, we can assume that  $N_G(v) \neq \emptyset$ . In the first branch, we set  $f(v) = 2$ . In the second branch, we set  $f(v) = \bar{2}$ .

1. If  $N_G(v) \cap A(f) = \{x\}$ , then  $N_G(v) \cap \bar{V}_2(f) = \emptyset$  in this phase. Therefore, in the first branch,  $f(x) = 0$  is enforced by Reduction Rule LPN. Notice that this might further trigger Reduction Rule  $V_0$  if  $N_G(x) \cap (A(f) \cup \bar{V}_1(f))$  contains vertices other than  $v$ . The corresponding worst-case recurrence is:  $T(\mu) = T(\mu - (1 - \omega_2)) + T(\mu - (1 + 1))$ , see Table 1, fourth row.
2. If  $N_G(v) \cap \bar{V}_2(f) = \{x\}$ , then  $N_G(v) \cap A(f) = \emptyset$  in this phase. Therefore, in the first branch,  $f(x) = 0$  is enforced by Reduction Rule LPN. We consider several sub-cases now.
  - (a)  $N_G(x) \cap \bar{V}_1(f) \neq \emptyset$ . Reduction Rule  $V_0$  will put all these vertices into  $V_0(f)$ . The corresponding worst-case recurrence is:  $T(\mu) = T(\mu - (1 - \omega_2)) + T(\mu - (1 + \omega_2 + \omega_1))$ , see Table 1, fifth row.
  - (b)  $|N_G(x) \cap A(f)| \geq 2$ . Reduction Rule  $V_0$  will put all these vertices into  $\bar{V}_2(f)$  (except for  $v$ ). The corresponding worst-case recurrence is:  $T(\mu) = T(\mu - (1 - \omega_2)) + T(\mu - (1 + \omega_2 + (1 - \omega_2)))$ , see Table 1, fourth row.
  - (c) Recall that by Reduction Rule Edges,  $N_G(x) \cap (\bar{V}_2(f) \cup V_0(f) \cup V_1(f)) = \emptyset$ , so that (if the first two cases do not apply) now we have  $N_G(x) \setminus \{v\} \subseteq V_2(f)$ . By the properties listed above, also  $N_G(x) \cap V_2(f) = \emptyset$  is clear, so that now  $|N_G(x)| = 1$ , i.e.,  $x$  is a pendant vertex. In this situation, we do not gain anymore from the first branch, but when  $f(v) = \bar{2}$  is set, Reduction Rule NPD triggers and sets  $f(x) = 1$ . The corresponding worst-case recurrence is:  $T(\mu) = T(\mu - (1 + \omega_2)) + T(\mu - (1 - \omega_2 + \omega_2))$ , see Table 1, sixth row.

**Branching in Phase 3.** As  $A(f) = \emptyset$ , we are now branching on a vertex  $v \in \bar{V}_1(f)$ . Due to Reduction Rule Isolate, we know that  $N_G(v) \cap \bar{V}_2(f) \neq \emptyset$ . In the first branch, we consider setting  $f(v) = 2$ , while in the second branch, we set  $f(v) = 0$ . Again, we discuss several scenarios in the following.

1. Assume that  $|N_G(v) \cap \overline{V}_2(f)| \geq 3$ . If we set  $f(v) = 2$ , then Reduction Rule  $V_2$  triggers at least thrice. The corresponding worst-case recurrence is:  $T(\mu) = T(\mu - \omega_1) + T(\mu - (\omega_1 + 3\omega_2))$ , with a branching vector of  $(\omega_1, \omega_1 + 3\omega_2)$ , see Table 1, seventh row.
2. Assume that  $|N_G(v) \cap \overline{V}_2(f)| = 1$ , i.e., there is some (unique)  $u \in \overline{V}_2(f)$  such that  $N_G(v) \cap \overline{V}_2(f) = \{u\}$ . We consider two sub-cases:
  - (a) If  $|N_G(u) \cap \overline{V}_1(f)| \geq 2$ , then if we set  $f(v) = 2$ , then first Reduction Rule LPN triggers  $f(u) = 0$ , which in turn sets  $f(w) = 0$  for all  $w \in N_G(u) \cap \overline{V}_1(f)$ ,  $w \neq u$ , by Reduction Rule  $V_0$ . The corresponding worst-case recurrence is:  $T(\mu) = T(\mu - \omega_1) + T(\mu - (\omega_2 + 2\omega_1))$ , see Table 1, eighth row.
  - (b) If  $|N_G(u) \cap \overline{V}_1(f)| = 1$ , then  $u$  is a pendant vertex. Hence, in the first branch, we have (as above)  $f(v) = 2$  and  $f(u) = 0$ , while in the second branch, we have  $f(v) = 0$  and  $f(u) = 1$  by Reduction Rule NPD. This decreases the measure by  $\omega_1 + \omega_2$  in both branches, see Table 1, ninth row. This scenario happens in particular if the graph  $G'$  induced by  $\overline{V}_1(f) \cup \overline{V}_2(f)$  contains a connected component which is a  $P_2$ . Therefore, we refer to this (also) as a  $P_2$ -branching below.
3. Assume that  $|N_G(v) \cap \overline{V}_2(f)| = 2$ , i.e., there are some  $u_1, u_2 \in \overline{V}_2(f)$  such that  $N_G(v) \cap \overline{V}_2(f) = \{u_1, u_2\}$ . Notice that in the first branch, when  $f(v) = 2$ , Reduction Rule  $V_2$  triggers twice, already reducing the measure by  $\omega_1 + 2\omega_2$ . We consider further sub-cases:
  - (a) If  $|N_G(u_1) \cap \overline{V}_1(f)| = 1$ , then  $u_1$  is a pendant vertex. As in the previous sub-case, this helps us reduce the measure in the second branch by  $\omega_1 + \omega_2$  due to Reduction Rule NPD, which obviously puts us in a better branching than Table 1, ninth row. Similarly, we can discuss the case  $|N_G(u_2) \cap \overline{V}_1(f)| = 1$ .
  - (b) If  $|N_G(u_1) \cap \overline{V}_1(f)| = 2$ , then we know now that the graph  $G'$  induced by  $\overline{V}_1(f) \cup \overline{V}_2(f)$  is bipartite after removing edges between vertices from  $\overline{V}_1(f)$ , and vertices from  $\overline{V}_1(f)$  all have degree two and vertices from  $\overline{V}_2(f)$  all have degree at least two. The worst case for the following branching is hence given by a  $K_{2,2}$  as a connected component in  $G'$ : Testing now all possibilities of setting the  $\overline{V}_2(f)$ -vertices to 0 or to 1 will determine all values of the  $\overline{V}_1(f)$ -vertices by reduction rules. Hence, we have in particular for the  $K_{2,2}$  a scenario with four branches, and in each branch, the measure is reduced by  $2\omega_2 + 2\omega_1$  ( $K_{2,2}$ -branching).

**Proposition 11.** *On input graphs of order  $n$ , Algorithm REFINED RD ENUMERATION runs in time  $\mathcal{O}^*(1.9332^n)$ .*

*Proof.* We follow the run-time analysis that led us to the branching vectors listed in Table 1. The claim follows by choosing as weights  $\omega_1 = \frac{2}{3}$  and  $\omega_2 = 0.38488$ .  $\square$

The two worst-case branchings (with the chosen weights  $\omega_1 = \frac{2}{3}$  and  $\omega_2 = 0.38488$ ) are 1.1, 3.2.b and 3.3. If we want to further improve on our figures, we would have to work on a deeper analysis in these cases. For the  $P_2$ -branching,

it might be an idea to combine it with the branchings where it could ever originate from. Notice that adjacent  $\overline{V_1}$ - $\overline{V_2}$ -vertices can be only produced in the first branching phase. But we would then have to improve also on Phase 3.3, the worst case being a  $K_{2,2}$ -branching in Case 3.3 (b).

Let us finally summarize the corner-stones of our reasoning.

*Proof (Theorem 5).* Several important properties have been claimed and proved about Algorithm 3 that show the claim of our second main theorem.

- The algorithm correctly enumerates all minimal rdf; see Corollary 3.
- The algorithm needs polynomial space only; see Corollary 4.
- The algorithm achieves polynomial delay; see Proposition 5.
- The algorithm runs in time  $\mathcal{O}^*(1.9332^n)$  on input graphs of order  $n$ ; see Proposition 11. □

## 7 An Alternative Notion of Minimal RDF

So far, we focused on an ordering of the functions  $V \rightarrow \{0, 1, 2\}$  that was derived from the linear ordering  $0 < 1 < 2$ . Due to the different functionalities, it might be not that clear if 2 should be bigger than 1. If we rather choose as a basic partial ordering  $0 < 1, 2$ , with 1, 2 being incomparable, this yields another ordering for the functions  $V \rightarrow \{0, 1, 2\}$ , again lifted pointwise. Being reminiscent of partial orderings, let us call the resulting notion of minimality PO-minimal rdf. Recall that the notion of minimality for Roman dominating functions that we considered so far and that we also view as the most natural interpretation of this notion has been refuted in the literature, because it leads to a trivial notion of UPPER ROMAN DOMINATION, because the minimal rdf  $f : V \rightarrow \{0, 1, 2\}$  with biggest sum  $\sum_{v \in V} f(v)$  is achieved by the constant function  $f = 1$ . This is no longer true for the (new) problem UPPER PO-ROMAN DOMINATION.

Also, this can be seen as a natural pointwise lifting of the inclusion ordering, keeping in mind that  $f \leq_{PO} g$  iff  $V_1(f) \subseteq V_1(g)$  and  $V_2(f) \subseteq V_2(g)$ .

More interesting for the storyline of this paper are the following results:

**Theorem 6.** *Let  $G = (V, E)$  be a graph,  $f : V \rightarrow \{0, 1, 2\}$  and abbreviate  $G' := G[V_0(f) \cup V_2(f)]$ . Then,  $f$  is a PO-minimal rdf if and only if the following conditions hold:*

1.  $N_G[V_2(f)] \cap V_1(f) = \emptyset$ ,
2.  $V_2(f)$  is a minimal dominating set of  $G'$ .

*Proof.* First we look into the “only if”-part. The first condition follows analogously from Theorem 1. For the other condition, we assume that there exists a graph  $G = (V, E)$  and a PO-minimal-rdf  $f : V \rightarrow \{0, 1, 2\}$  such that  $V_2(f)$  is not a minimal dominating set in  $G'$ . Since  $f$  is a rdf,  $V_2(f)$  is a dominating set in  $G'$ . Thus,  $V_2(f)$  is not irredundant in  $G'$ . Hence, there exists a  $v \in V_2(f)$  such that  $N[v] \subseteq N_G[V_2(f) \setminus \{v\}]$ . Define

$$\tilde{f} : V \rightarrow \{0, 1, 2\}, w \mapsto \begin{cases} f(w), & w \neq v \\ 0, & w = v \end{cases}.$$

Clearly, vertices  $w \in (V_0(f) \cup V_1(f)) \setminus N_G[v]$  are dominated by  $\tilde{f}$ . But  $N_G[v]$  is also dominated, since  $N_G[v] \subseteq N_G[V_2(f) \setminus \{v\}]$  holds. This would contradict the PO-minimality of  $f$ .

Let  $f$  be a function that fulfills the two conditions. Since  $V_2(f)$  is a dominating set in  $G'$ , for each  $u \in V_0(f)$ , there exists a  $v \in V_2(f) \cap N_G[u]$ . Therefore,  $f$  is a rdf. Let  $\tilde{f} : V \rightarrow \{0, 1, 2\}$  be a PO-minimal rdf such that  $\tilde{f}$  is smaller than  $f$  with respect to the partial ordering. Therefore,  $\tilde{f}$  (also) satisfies the two conditions. Assume that there exists a  $v \in V$  with  $\tilde{f}(v) < f(v)$ . Hence,  $V_2(\tilde{f}) \subseteq V_2(f) \setminus \{v\}$ .

**Case 1:**  $\tilde{f}(v) = 0$  and  $f(v) = 1$ . Therefore, there exists a vertex  $u \in N_G(v)$  with  $f(u) \geq \tilde{f}(u) = 2$ . This contradicts Condition 1.

**Case 2:**  $\tilde{f}(v) = 0$  and  $f(v) = 2$ . Thus, for each  $u \in V_0(f) \cap N_G[v] \subseteq V_0(\tilde{f}) \cap N_G[v]$  there exists a  $w \in V_2(\tilde{f}) \cap N_G(u) \subseteq V_2(f) \cap N_G(u)$ . This implies, that  $V_2(f)$  is not irredundant in  $G'$ , which contradicts the second condition.

Therefore,  $\tilde{f} = f$  holds and  $f$  is PO-minimal.  $\square$

Based on this characterization of PO-minimality, we can again derive a positive algorithmic results for the corresponding extension problem.

**Theorem 7.** *The extension problem EXTPO-RDF can be solved in polynomial time.*

*Proof.* For this problem, we have to modify Algorithm 1 again. This time, we have to modify Line 13 to **if**  $N_G[v] \subseteq N[M_2 \setminus \{v\}]$  **do**. The rest of the proof is analogous to the proof of Theorem 4.  $\square$

Furthermore, we can show that for PO-minimal rdf, the simple enumeration algorithm is already provably optimal.

**Theorem 8.** *There is a polynomial-space algorithm that enumerates all PO-minimal rdf of a given graph of order  $n$  in time  $\mathcal{O}^*(2^n)$  with polynomial delay. Moreover, there is a family of graphs  $G_n$ , with  $G_n$  being of order  $n$ , such that  $G_n$  has  $2^n$  many PO-minimal rdf.*

*Proof.* The algorithm itself works similar to Algorithm 2, but we have to integrate the extension tests as in Algorithm 3. Therefore, we need to combine our two modifications for Algorithm 1. This new version would solve the GENEXTPO-RDF, where a graph  $G = (V, E)$ , a function  $f : V \rightarrow \{0, 1, 2\}$  and a set  $\bar{V}_2$  are given and we need to find a PO-minimal rdf  $\tilde{f}$  with  $f \leq \tilde{f}$  and  $\bar{V}_2 \cap V_2(f) = \emptyset$  (to prove this, combine the proofs of Lemma 4 and Theorem 7). To see optimality of the enumeration algorithm, notice that the null graph (edge-less graph) of order  $n$  has any mapping  $f : V \rightarrow \{1, 2\}$  as a PO-minimal rdf.  $\square$

It follows that the (relatively simple) enumeration algorithm is optimal for PO-minimal rdf. If one dislikes the fact that our graph family is disconnected, consider the star  $K_{1,n}$  that has  $2^n + 1$  many different PO-minimal rdf: If  $V(K_{1,n}) =$

$\{0, 1, \dots, n\}$ , with 0 being the center and  $i \in \{1, \dots, n\}$  being the ‘ray vertices’ of this star, then either put  $f(0) = 2$  and  $f(i) = 0$  for  $i \in \{1, \dots, n\}$ , or  $f(j) = 1$  for  $j \in \{0, 1, \dots, n\}$ , or  $f(0) = 0$  and  $f(i) \in \{1, 2\}$  is arbitrary for  $i \in \{1, \dots, n\}$  (except for  $f(j) = 1$  for  $j \in \{1, \dots, n\}$ ). This example proves that there cannot be any general enumeration algorithm running in time  $\mathcal{O}((2-\varepsilon)^n)$  for any  $\varepsilon > 0$ , even for connected graphs of order  $n$ .

## 8 Conclusions

While the combinatorial concept of Roman domination leads to a number of complexity results that are completely analogous to what is known about the combinatorial concept of domination, the two concepts lead to distinctively different results when it comes to enumeration and extension problems. These are the main messages and results of the present paper.

We are currently working on improved enumeration and also on counting of minimal rdf in special graph classes. Our first results are very promising; for instance, there are good chances to completely close the gap between lower and upper bounds for enumerating minimal rdf for some graph classes.

Another line of research is looking into problems that are similar to Roman domination, in order to better understand the specialties of Roman domination in contrast to the classical domination problem. What makes Roman domination behave different from classical domination when it comes to finding extensions or to enumeration?

Finally, let us mention that our main branching algorithm also gives an input-sensitive enumeration algorithm for minimal Roman dominating functions in the sense of Chellali *et al.* [16]. However, we do not know of a polynomial-delay enumeration algorithm in that case. This is another interesting line of research. Here, the best lower bound we could find was a repetition of a  $C_4$ , leading to  $\sqrt[4]{8} \geq 1.68179$  as the basis.

## References

1. Abu-Khzam, F.N., Bazgan, C., Chopin, M., Fernau, H.: Data reductions and combinatorial bounds for improved approximation algorithms. *Journal of Computer and System Sciences* **82**(3), 503–520 (2016)
2. Abu-Khzam, F.N., Heggernes, P.: Enumerating minimal dominating sets in chordal graphs. *Information Processing Letters* **116**(12), 739–743 (2016)
3. Bazgan, C., Brankovic, L., Casel, K., Fernau, H., Jansen, K., Klein, K.M., Lampis, M., Liedloff, M., Monnot, J., Paschos, V.: The many facets of upper domination. *Theoretical Computer Science* **717**, 2–25 (2018)
4. Benecke, S.: Higher Order Domination of Graphs. Master’s thesis, Department of Applied Mathematics of the University of Stellenbosch, South Africa, <http://dip.sun.ac.za/~vuuren/Theses/Benecke.pdf> (2004)
5. Bermudo, S., Fernau, H.: Computing the differential of a graph: hardness, approximability and exact algorithms. *Discrete Applied Mathematics* **165**, 69–82 (2014)

6. Bermudo, S., Fernau, H.: Combinatorics for smaller kernels: The differential of a graph. *Theoretical Computer Science* **562**, 330–345 (2015)
7. Bermudo, S., Fernau, H., Sigarreta, J.M.: The differential and the Roman domination number of a graph. *Applicable Analysis and Discrete Mathematics* **8**, 155–171 (2014)
8. Bläsius, T., Friedrich, T., Lischeid, J., Meeks, K., Schirneck, M.: Efficiently enumerating hitting sets of hypergraphs arising in data profiling. In: *Algorithm Engineering and Experiments (ALENEX)*. pp. 130–143. SIAM (2019)
9. Bonamy, M., Defrain, O., Heinrich, M., Raymond, J.F.: Enumerating minimal dominating sets in triangle-free graphs. In: Niedermeier, R., Paul, C. (eds.) *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*. LIPIcs, vol. 126, pp. 16:1–16:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
10. Boros, E., Gurvich, V., Hammer, P.L.: Dual subimplicants of positive Boolean functions. *Optimization Methods and Software* **10**(2), 147–156 (1998)
11. Casel, K., Fernau, H., Ghadikolaei, M.K., Monnot, J., Sikora, F.: Extension of some edge graph problems: Standard and parameterized complexity. In: Gasieniec, L.A., Jansson, J., Levkopoulos, C. (eds.) *Fundamentals of Computation Theory - 22nd International Symposium, FCT*. LNCS, vol. 11651, pp. 185–200. Springer (2019)
12. Casel, K., Fernau, H., Ghadikolaei, M.K., Monnot, J., Sikora, F.: Abundant extensions. In: Calamoneri, T., Corò, F. (eds.) *Algorithms and Complexity - 12th International Conference, CIAC*. LNCS, vol. 12701, pp. 3–17. Springer (2021). [https://doi.org/10.1007/978-3-030-75242-2\\_1](https://doi.org/10.1007/978-3-030-75242-2_1)
13. Casel, K., Fernau, H., Ghadikolaei, M.K., Monnot, J., Sikora, F.: On the complexity of solution extension of optimization problems. *Theoretical Computer Science* **904**, 48–65 (2022). <https://doi.org/https://doi.org/10.1016/j.tcs.2021.10.017>
14. Chambers, E.W., Kinnnersley, B., Prince, N., West, D.B.: Extremal problems for Roman domination. *SIAM Journal of Discrete Mathematics* **23**, 1575–1586 (2009)
15. Chapelle, M., Cochefert, M., Couturier, J., Kratsch, D., Liedloff, M., Perez, A.: Exact algorithms for weak Roman domination. In: Lecroq, T., Mouchard, L. (eds.) *Combinatorial Algorithms - 24th International Workshop, IWOCA*. LNCS, vol. 8288, pp. 81–93. Springer (2013)
16. Chellali, M., Haynes, T.W., Hedetniemi, S.M., Hedetniemi, S.T., McRae, A.A.: A Roman domination chain. *Graphs and Combinatorics* **32**(1), 79–92 (2016)
17. Cockayne, E.J., Dreyer Jr., P., Hedetniemi, S.M., Hedetniemi, S.T.: Roman domination in graphs. *Discrete Mathematics* **278**, 11–22 (2004)
18. Couturier, J., Hegernes, P., van 't Hof, P., Kratsch, D.: Minimal dominating sets in graph classes: Combinatorial bounds and enumeration. *Theoretical Computer Science* **487**, 82–94 (2013)
19. Couturier, J., Letourneur, R., Liedloff, M.: On the number of minimal dominating sets on some graph classes. *Theoretical Computer Science* **562**, 634–642 (2015)
20. Creignou, N., Kröll, M., Pichler, R., Skritek, S., Vollmer, H.: A complexity theory for hard enumeration problems. *Discrete Applied Mathematics* **268**, 191–209 (2019)
21. Dreyer, P.A.: *Applications and Variations of Domination in Graphs*. Ph.D. thesis, Rutgers University, New Jersey, USA, PhD Thesis (2000)
22. Eiter, T., Gottlob, G.: Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing* **24**(6), 1278–1304 (1995)
23. Favaron, O., Karami, H., Khoeilar, R., Sheikholeslami, S.M.: On the Roman domination number of a graph. *Discrete Mathematics* **309**(10), 3447 – 3451 (2009)

24. Fernau, H.: ROMAN DOMINATION: a parameterized perspective. *International Journal of Computer Mathematics* **85**, 25–38 (2008)
25. Gainer-Dewar, A., Vera-Licona, P.: The minimal hitting set generation problem: Algorithms and computation. *SIAM Journal of Discrete Mathematics* **31**(1), 63–100 (2017)
26. Golovach, P.A., Heggernes, P., Kanté, M.M., Kratsch, D., Villanger, Y.: Enumerating minimal dominating sets in chordal bipartite graphs. *Discrete Applied Mathematics* **199**, 30–36 (2016)
27. Golovach, P.A., Heggernes, P., Kratsch, D.: Enumerating minimal connected dominating sets in graphs of bounded chordality. *Theoretical Computer Science* **630**, 63–75 (2016)
28. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: *Fundamentals of Domination in Graphs*, Monographs and Textbooks in Pure and Applied Mathematics, vol. 208. Marcel Dekker (1998)
29. Haynes, T.W., Hedetniemi, S., Henning, M.A. (eds.): *Topics in Domination in Graphs*, Developments in Mathematics, vol. 64. Springer (2020)
30. Hedetniemi, S.T., Rubalcaba, R.R., Slater, P.J., Walsh, M.: Few compare to the great Roman empire. *Congressus Numerantium* **217**, 129–136 (2013)
31. Kanté, M.M., Limouzy, V., Mary, A., Nourine, L.: On the enumeration of minimal dominating sets and related notions. *SIAM Journal of Discrete Mathematics* **28**(4), 1916–1929 (2014)
32. Kanté, M.M., Limouzy, V., Mary, A., Nourine, L., Uno, T.: Polynomial delay algorithm for listing minimal edge dominating sets in graphs. In: Dehne, F., Sack, J., Stege, U. (eds.) *Workshop on Algorithms and Data Structures, WADS*. LNCS, vol. 9214, pp. 446–457. Springer (2015)
33. Kanté, M.M., Limouzy, V., Mary, A., Nourine, L., Uno, T.: A polynomial delay algorithm for enumerating minimal dominating sets in chordal graphs. In: Mayr, E.W. (ed.) *International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2015*. LNCS, vol. 9224, pp. 138–153. Springer (2016)
34. Kraner Šumenjak, T., Pavlič, P., Tepeh, A.: On the Roman domination in the lexicographic product of graphs. *Discrete Applied Mathematics* **160**(13–14), 2030–2036 (2012)
35. Liedloff, M.: *Algorithmes exacts et exponentiels pour les problèmes NP-difficiles: domination, variantes et généralisations*. PhD thesis, Université Paul Verlaine - Metz, France (2007)
36. Liedloff, M., Kloks, T., Liu, J., Peng, S.L.: Efficient algorithms for Roman domination on some classes of graphs. *Discrete Applied Mathematics* **156**(18), 3400–3415 (2008)
37. Liu, C.H., Chang, G.J.: Roman domination on 2-connected graphs. *SIAM Journal of Discrete Mathematics* **26**(1), 193–205 (2012)
38. Liu, C.H., Chang, G.J.: Upper bounds on Roman domination numbers of graphs. *Discrete Mathematics* **312**(7), 1386–1391 (2012)
39. Liu, C.H., Chang, G.J.: Roman domination on strongly chordal graphs. *Journal of Combinatorial Optimization* **26**(3), 608–619 (2013)
40. Mary, A.: *Énumération des dominants minimaux d’un graphe*. Ph.D. thesis, LIMOS, Université Blaise Pascal, Clermont-Ferrand, France (Nov 2013)
41. Mashburn, J.L., Haynes, T.W., Hedetniemi, S.M., Hedetniemi, S.T., Slater, P.J.: Differentials in graphs. *Utilitas Mathematica* **69**, 43–54 (2006)
42. Mobaraky, B.P., Sheikholeslami, S.M.: Bounds on Roman domination numbers of graphs. *Matematitchki Vesnik* **60**, 247–253 (2008)

43. Pagourtzis, A., Penna, P., Schlude, K., Steinhöfel, K., Taylor, D.S., Widmayer, P.: Server placements, Roman domination and other dominating set variants. In: Baeza-Yates, R.A., Montanari, U., Santoro, N. (eds.) Foundations of Information Technology in the Era of Networking and Mobile Computing, IFIP 17<sup>th</sup> World Computer Congress — TC1 Stream / 2<sup>nd</sup> IFIP International Conference on Theoretical Computer Science IFIP TCS. pp. 280–291. Kluwer (2002), also available as Technical Report 365, ETH Zürich, Institute of Theoretical Computer Science, 10/2001.
44. Peng, S.L., Tsai, Y.H.: Roman domination on graphs of bounded treewidth. In: The 24th Workshop on Combinatorial Mathematics and Computation Theory. pp. 128–131 (2007)
45. ReVelle, C.S., Rosing, K.E.: Defendens imperium Romanum: A classical problem in military strategy. *American Mathematical Monthly* **107**, 585–594 (2000), <http://www.jhu.edu/~jhumag/0497web/locate3.html>
46. van Rooij, J.M.M.: Exact Exponential-Time Algorithms for Domination Problems in Graphs. Ph.D. thesis, Universiteit Utrecht, The Netherlands (2011)
47. Shang, W., Wang, X., Hu, X.: Roman domination and its variants in unit disk graphs. *Discrete Mathematics, Algorithms and Applications* **2**(1), 99–106 (2010)
48. Shi, Z., Koh, K.M.: Counting the number of minimum Roman dominating functions of a graph. Tech. rep., ArXiv / CoRR, abs/1403.1019 (2014)
49. Stewart, I.: Defend the Roman Empire. *Scientific American* pp. 136,137,139 (Dec 1999)
50. Xing, H.M., Chen, X., Chen, X.G.: A note on Roman domination in graphs. *Discrete Mathematics* **306**(24), 3338–3340 (2006)
51. Xueliang, F., Yuansheng, Y., Baoqi, J.: Roman domination in regular graphs. *Discrete Mathematics* **309**(6), 1528–1537 (2009)
52. Yero, I.G., Rodríguez-Velázquez, J.A.: Roman domination in Cartesian product graphs and strong product graphs. *Applicable Analysis and Discrete Mathematics* **7**, 262–274 (2013)