Discrete Applied Mathematics 248 (2018) 79–92

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Exact algorithms for weak Roman domination[☆,☆☆]

Mathieu Chapelle[c], Manfred Cochefert[b], Jean-François Couturier[a], Dieter Kratsch[b], Romain Letourneur[c], Mathieu Liedloff[c,*], Anthony Perez[c]

[a] CReSTIC, IFTS, Pôle de haute technologie, 08000 Charleville-Mézières, France
[b] Laboratoire d'Informatique Théorique et Appliquée, Université de Lorraine, 57045 Metz Cedex 01, France
[c] Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, FR-45067 Orléans, France

A R T I C L E I N F O

Article history:
Received 31 May 2016
Received in revised form 12 September 2017
Accepted 23 September 2017
Available online 6 November 2017

Keywords:
Exact algorithm
Graph algorithm
Roman domination
Interval graphs

A B S T R A C T

We consider the Weak Roman Domination problem. Given an undirected graph $G = (V, E)$, the aim is to find a *weak Roman domination* function (wrd-function for short) of minimum cost, *i.e.* a function $f : V \to \{0, 1, 2\}$ such that every vertex $v \in V$ is *defended* (*i.e.* there exists a neighbor u of v, possibly $u = v$, such that $f(u) \geqslant 1$) and for every vertex $v \in V$ with $f(v) = 0$ there exists a neighbor u of v such that $f(u) \geqslant 1$ and the function $f_{u\to v}$ defined by $f_{u\to v}(v) = 1$, $f_{u\to v}(u) = f(u) - 1$ and $f_{u\to v}(x) = f(x)$ otherwise does not contain any undefended vertex. The *cost* of a wrd-function f is defined by $cost(f) = \sum_{v\in V} f(v)$. The trivial enumeration algorithm runs in time $\mathcal{O}^*(3^n)$ and polynomial space and is the best one known for the problem so far. We are breaking the trivial enumeration barrier by providing two faster algorithms: we first prove that the problem can be solved in $\mathcal{O}^*(2^n)$ time needing *exponential space*, and then describe an $\mathcal{O}^*(2.2279^n)$ algorithm using *polynomial space*. Our results rely on structural properties of a wrd-function, as well as on the best polynomial space algorithm for the Red-Blue Dominating Set problem. Moreover we show that the problem can be solved in linear-time on interval graphs.

1. Introduction

In this paper we investigate a domination-like problem from the exact exponential algorithms viewpoint. In the classical Dominating Set problem, one is given an undirected graph $G = (V, E)$, and asked to find a dominating set S, *i.e.* every vertex $v \in V$ either belongs to S or has a neighbor in S, of minimum size. The Dominating Set problem ranges among one of the most famous *NP*-complete covering problems [9], and has received a lot of attention during the last decades. In particular, the trivial enumeration algorithm of runtime $\mathcal{O}^*(2^n)$[1] has been improved by a sequence of papers [8,16,27]. The currently best known algorithms for the problem run in time $\mathcal{O}^*(1.4864^n)$ using polynomial space, and in time $\mathcal{O}^*(1.4689^n)$ needing exponential space [16].

Many variants of the Dominating Set problem have been introduced and studied extensively both from structural and algorithmic viewpoints. The number of papers on domination in graphs and its variants is in the thousands, and several

☆ A preliminary version of this paper appears in Chapelle et al. (2013) [2].
☆☆ This work was supported by the French AGAPE project (ANR-09-BLAN-0159).
* Corresponding author.
E-mail addresses: Mathieu.Ch@gmail.com (M. Chapelle), manfred.cochefert@gmail.com (M. Cochefert), jean-francois.couturier@univ-reims.fr (J.-F. Couturier), dieter.kratsch@univ-lorraine.fr (D. Kratsch), romain.letourneur@univ-orleans.fr (R. Letourneur), mathieu.liedloff@univ-orleans.fr (M. Liedloff), anthony.perez@univ-orleans.fr (A. Perez).
1 The notation $\mathcal{O}^*(f(n))$ suppresses polynomial factors.

https://doi.org/10.1016/j.dam.2017.09.015
0166-218X/© 2017 Elsevier B.V. All rights reserved.

well-known surveys and books are dedicated to the topic (see, *e.g.* , [13]). One of those variants called Roman Domination was introduced in [6], motivated by the articles by Stewart [25] and by ReVelle and K.E. Rosing [23]. In general, the aim is to protect a set of locations (vertices of a graph) by using a smallest possible amount of legions (to be placed on those vertices).

Since then, numerous articles have been published around this problem, which has been studied from combinatorics bounds on the minimum cost or related parameters, characterizations of graphs achieving such bounds and algorithms to compute function of minimum costs (see, *e.g.* , [1,3,5,7,19,20,28]). In particular, this *NP*-complete problem has been tackled using exact exponential algorithms. The first non-trivial one achieved had running time $\mathcal{O}^*(1.6183^n)$ and used polynomial space [17]. This result has recently been improved to $\mathcal{O}^*(1.5673^n)$ [26], which can be lowered to $\mathcal{O}^*(1.5014^n)$ at the cost of exponential space [26]. Moreover, the Roman Domination problem can be related to several other variants of *defense-like* domination, such as *secure domination* (see, *e.g.* , [4,5,12]), or *eternal domination* (see, *e.g.* , [10,11]).

We focus our attention on yet another variant of the Roman Domination problem. In 2003, Henning et al. [14] considered the following idea: location t can also be protected if one of its neighbors possesses one legion that can be moved to t in such a way that the whole collection of locations (set of vertices) remains protected. This variation adds some kind of dynamics to the problem and gives rise to the Weak Roman Domination problem. Formally, it can be defined as follows:

Weak Roman Domination:
Input: An undirected graph $G = (V, E)$.
Output: A weak Roman domination function f of G of minimum cost.

A weak Roman domination (wrd-function) is a function $f : V \to \{0, 1, 2\}$ such that every vertex $v \in V$ is *defended* (*i.e.* there exists a neighbor u of v, possibly $u = v$, such that $f(u) \geqslant 1$) and for every vertex $v \in V$ with $f(v) = 0$ there exists a neighbor u of v such that $f(u) \geqslant 1$ and the function $f_{u\to v}$ defined by $f_{u\to v}(v) = 1, f_{u\to v}(u) = f(u) - 1$ and $f_{u\to v}(x) = f(x)$ otherwise does not contain any undefended vertex. I.e., one can move one legion from u to v without creating any undefended vertex. The *cost* of a wrd-function f is defined by $cost(f) = \sum_{v\in V} f(v)$.

While several structural results on Weak Roman Domination are known, see, *e.g.* , [4,5,14,22], few algorithmic results are known. Note that a preliminary version of this paper appears in Chapelle et al. (2013) [2]. The problem has been shown to be NP-hard, even when restricted to bipartite or chordal graphs [14] and a linear-time algorithm is known for block graphs [21].

Our contribution. In this paper, we give the first algorithms tackling this problem faster than by the $\mathcal{O}^*(3^n)$ brute-force algorithm obtained by enumerating all legion functions. Both our algorithms rely on structural properties for weak Roman domination functions, described in Section 3. In Section 4, we first give an $\mathcal{O}^*(2^n)$ time and exponential space algorithm. We then show how the exponential space can be avoided by using an exponential algorithm for the Red-Blue Dominating Set problem [26], which leads to an $\mathcal{O}^*(2.2279^n)$ algorithm. In Section 5, we show that the problem can be solved in linear-time on interval graphs.

2. Preliminaries and notations

We consider simple undirected graphs $G = (V, E)$ and assume that $n = |V|$. Given a vertex $v \in V$, we denote by $N(v)$ its *open neighborhood*, by $N[v]$ its *closed neighborhood* (*i.e.* $N[v] = N(v) \cup \{v\}$). For $X \subseteq V$, let $N[X] = \cup_{v\in X} N[v]$ and $N(X) = N[X] \setminus X$. Similarly, given $S \subseteq V$, we use $N_S(v)$ to denote the set $N(v) \cap S$. A subset of vertices $S \subseteq V$ is a *dominating set* of G if for every vertex $v \in V$ either $v \in S$ or $N_S(v) \neq \emptyset$. Furthermore, $Y \subseteq V$ dominates $X \subseteq V$ in $G = (V, E)$ if $X \subseteq N[Y]$. A subset of vertices $S' \subseteq V$ is an *independent set* in G if there is no edge in G between any pair of vertices in S'. Finally, a graph $G = (V, E)$ is *bipartite* whenever its vertex set can be partitioned into two independent sets V_1 and V_2.

2.1. Legion and wrd-functions

A function $f : V \to \{0, 1, 2\}$ is called a *legion function*. With respect to f, a vertex $v \in V$ is said to be *secured* if $f(v) \geqslant 1$, and *unsecured* otherwise. Similarly, a vertex $v \in V$ is said to be *defended* if there exists $u \in N[v]$ such that $f(u) \geqslant 1$. Otherwise, v is said to be *undefended*. The function f is a *weak Roman domination function* (wrd-function for short) if there is no undefended vertex with respect to f, and for every vertex $v \in V$ with $f(v) = 0$ there exists a secured vertex $u \in N(v)$ such that the function $f_{u\to v} : V \to \{0, 1, 2\}$ defined by:

$$f_{u\to v}(x) = \begin{cases} 1 & \text{if } x = v \\ f(u) - 1 & \text{if } x = u \\ f(x) & \text{if } x \notin \{u, v\} \end{cases}$$

has no undefended vertex (see Fig. 1(a)). In other words, $f_{u\to v}$ denotes the legion function obtained by *moving* one legion from u to v.

Given a legion function f, we let V_f^1, V_f^2 denote the sets $\{v \in V : f(v) = 1\}$ and $\{v \in V : f(v) = 2\}$, respectively, and define its *underlying set* as $V_f = V_f^1 \cup V_f^2$. The *cost* of f is then defined by $cost(f) = \sum_{v\in V} f(v) = |V_f^1| + 2|V_f^2|$. Notice that when f is a wrd-function, the set V_f is a (not necessarily minimal) dominating set of G. A pair (V_1, V_2) of subsets of vertices is

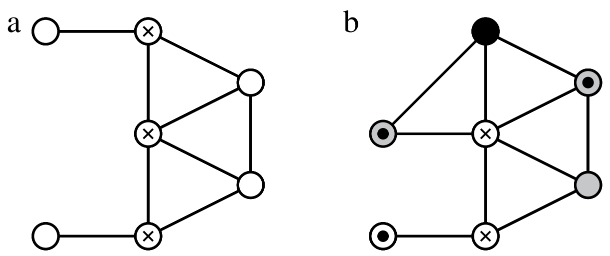


Fig. 1. (a) A graph $G = (V, E)$, and a wrd-function where each legion is represented by a cross. Any vertex is safely defended. (b) The black vertex is safely defended (one can safely move a legion on it without creating any undefended vertex), the gray vertices are non-safely defended (any move creates an undefended vertex) and the disked vertices are weakly defended.

a (minimum) weak Roman dominating set if there exists a wrd-function f (of minimum cost, respectively) such that $V_f^1 = V_1$ and $V_f^2 = V_2$. We let $cost(V_1, V_2)$ as the cost of any wrd-function f respecting (V_1, V_2), *i.e.* $cost(V_1, V_2) = |V_1| + 2|V_2|$.

2.2. Safely-defended vertices

We now distinguish two types of *defended* vertices. Let $v \in V$ be any vertex and f be a legion function. We say that v is *safely defended by* f if one of the following holds:

- v is secured (i.e. $f(v) \geqslant 1$).
- there exists a neighbor u of v such that $f(u) = 2$.
- there exists a neighbor u of v such that $f(u) = 1$ and the function $f_{u\to v}$ creates no new undefended vertices compared to ones undefended by f.

Otherwise, we say that v is *non-safely defended*. Notice that a legion function f is a wrd-function if and only if every vertex $v \in V$ is safely-defended by f.

Observe that for any non-safely defended vertex v, we have $f(v) = 0, f(u) = 1$ for every secured neighbor u of v and the legion function $f_{u\to v}$ previously defined contains (among possibly others) an undefended vertex $w \in N(u)$ for any such neighbor u. In the following, we will refer to w as *weakly defended by* u, *weakly defended due to* v, or simply *weakly defended* when the context is clear. Observe that a weakly defended vertex has exactly one secured neighbor. These notions are illustrated in Fig. 1(b).

3. Structure of a weak Roman domination function

In this section, we prove several key structural properties of a wrd-function that will be used in our algorithms.

Given a graph $G = (V, E)$ and a subset of vertices $V' \subseteq V$, we define the legion function $\chi^{V'}$ as the indicator function of the subset V':

$$\chi^{V'}(x) = \begin{cases} 1 & \text{if } x \in V' \\ 0 & \text{otherwise.} \end{cases}$$

We start with the following Lemma and show that if f is a minimum cost wrd-function, the set V_f^2 is a minimal dominating set of the vertices non-safely defended by χ^{V_f}. At the end of this section, we prove that in fact, V_f^2 is a minimum dominating set.

Lemma 1. *Let $G = (V, E)$ be a graph, f be a wrd-function of G of minimum cost, and V_f its underlying set. Then V_f^2 is a* minimal dominating set *of the vertices non-safely defended by χ^{V_f}.*

Proof. Let $u \in V \setminus V_f$ be a vertex non-safely defended by χ^{V_f}. Recall that u is non-safely defended by χ^{V_f} if for every $u' \in N_{V_f}(u)$ the legion function $\chi^{V_f}_{u'\to u}$ contains an undefended vertex. Hence, for every vertex $u' \in N_{V_f}(u)$, there exists a vertex u'' weakly defended due to u. In particular, this means that $u'u'' \in E$ and $uu'' \notin E$. We prove Lemma 1 through the following claims.

Claim 1. *V_f^2 is a dominating set of the vertices non-safely defended by χ^{V_f}.*

Proof. Assume for a contradiction that there exists a vertex $u \in V \setminus V_f$ non-safely defended by χ^{V_f} such that $N_{V_f^2}(u) = \emptyset$. Let u'' be any vertex weakly defended due to u, and let u' be the common neighbor of u and u'' in V_f. Recall that $N(u'') \cap V_f = \{u'\}$, since otherwise u'' would be defended by $\chi^{V_f}_{u' \to u}$. Moreover, we know by assumption that $f(u') = 1$. Hence, the vertex u'' is undefended by $\chi^{V_f}_{u' \to u}$, which contradicts the fact that f is a wrd-function. $\diamond$

Claim 2. *V_f^2 is a minimal dominating set of the vertices non-safely defended by χ^{V_f}.*

Proof. Assume for a contradiction that there exists $u \in V_f^2$ such that $V_f^2 \setminus \{u\}$ is a dominating set of the vertices non-safely defended by χ^{V_f}. We claim that the legion function f_u defined as:

$$f_u(x) = \begin{cases} 1 \text{ if } x = u \\ f(x) \text{ otherwise} \end{cases}$$

is a wrd-function. To see this, observe that since $V_f^2 \setminus \{u\}$ is a dominating set of the vertices non-safely defended by χ^{V_f}, any vertex of $N_{V \setminus V_f}(u)$ is safely defended by f_u. It follows that f_u is a wrd-function with $cost(f_u) < cost(f)$, a contradiction. $\diamond$

This completes the proof of Lemma 1. $\square$

Now we show that, given a dominating set V' of a graph $G = (V, E)$, a wrd-function can be obtained by computing a dominating set of the set $\overline{D}$ of all vertices non-safely defended by $\chi^{V'}$.

Lemma 2. *Let $V' \subseteq V$ be a dominating set of a graph $G = (V, E)$, and let S be a dominating set of all vertices $\overline{D}$ non-safely defended by $\chi^{V'}$. Then the function $f : V \to \{0, 1, 2\}$ defined by*

$$f(x) = \begin{cases} 2 \ \textit{if } x \in V' \cap S \\ 1 \ \textit{if } x \in (V' \cup S) \setminus (V' \cap S) \\ 0 \ \textit{otherwise} \end{cases}$$

is a wrd-function.

Proof. Let S be a dominating set of $\overline{D}$ in G. Observe first that since $V_f = V' \cup S$, and since V' is a dominating set, then so is V_f. We now show that the set $\overline{D}'$ of vertices non-safely defended by f is empty. Observe that since $V' \subseteq V_f$, we have $\overline{D}' \subseteq \overline{D} \setminus S$. Assume for a contradiction that $\overline{D}' \neq \emptyset$, and let $x \in \overline{D}'$. We distinguish two cases:

(i) If $N(x) \cap V' \cap S \neq \emptyset$ then x has a neighbor of f-value 2, and thus x is safely-defended, contradicting the choice of x.
(ii) Otherwise, by definition of V' and S, x has a neighbor y in S which does not belong to V'. We claim that the legion function $f_{y \to x}$ cannot contain any undefended vertex. Indeed, since y does not belong to the original dominating set V', all vertices are defended by V' in $f_{y \to x}$ (recall that any vertex v of V' satisfies $f(v) \geqslant 1$).

These two cases imply that $\overline{D}'$ is empty, and thus f is a wrd-function. $\square$

To conclude this section, the next Lemma and its forthcoming Corollary 1 show that given an underlying set V' of a function f, a minimum dominating set of the vertices non-safely defended by $\chi^{V'}$ is a good choice to decide the vertices of V_f^2 so that f is a wrd-function of minimum cost among all wrd-functions having the same underlying set V'. Corollary 1 concludes that, for any wrd-function f of minimum cost, the set V_f^2 is a dominating set of minimum size.

Lemma 3. *Let $V' \subseteq V$ be a dominating set of a graph $G = (V, E)$. Let S be a minimum dominating set of all vertices $\overline{D}$ non-safely defended by $\chi^{V'}$. Then the function $f : V \to \{0, 1, 2\}$ defined by*

$$f(x) = \begin{cases} 2 \ \textit{if } x \in V' \cap S \\ 1 \ \textit{if } x \in (V' \cup S) \setminus (V' \cap S) \\ 0 \ \textit{otherwise} \end{cases}$$

is a wrd-function of minimum cost among the ones having underlying set $V_f = V' \cup S$.

Proof. Let V', S and f as described in the Lemma and let $V_f = V' \cup S$ be the underlying set of f. Suppose that there exists a wrd-function f' with the same underlying set V_f, such that $cost(f') < cost(f)$ and assume that f' is of minimum possible cost. Observe that V_f is a dominating set (as $V' \subseteq V_f$) and since V_f is the underlying set of f', $f'(v) \geq 1$ for all $v \in V_f$. Since $cost(f') < cost(f)$ and the underlying sets of f and f' are the same, it follows that $|V_{f'}^2| < |V_f^2|$. Consider now the set $\widehat{D}$ of non-safely defended vertices by χ^{V_f}. Since $V' \subseteq V_f$, we have $\widehat{D} \subseteq \overline{D}$. Moreover, no vertex of $\widehat{D}$ has a neighbor in $S \setminus V'$, since otherwise such a vertex would be safely defended since it has also a neighbor in V' (recall that V' is a dominating set of the graph G). As a consequence, we can assume that $f'(v) = 1$ for all $v \in S \setminus V'$ since, otherwise, f' would not be of minimum cost. Thus $V_{f'}^2 \subseteq V'$ and, by definition of f, $V_f^2 \subseteq V'$. Since f' is a wrd-function, by Lemma 1, $V_{f'}^2$ is a dominating set of $\widehat{D}$ and the set $S' = (S \setminus V') \cup V_{f'}^2$ is a dominating set of $\overline{D}$ of size smaller than S, which is a contradiction. Thus, if S is a minimum dominating set of $\overline{D}$, then f is a wrd-function of minimum cost among the wrd-functions with underlying set V_f. $\square$

Algorithm 1: The preprocessing step algorithm.

```
for $k = 0$ to $n$ do
    $\mathsf{DS}[\emptyset, k] = \emptyset$;
foreach $X \subseteq V$ s.t. $|X| \geq 1$ do
    $\mathsf{DS}[X, 0] = \{\infty\}$;
    // The set $\{\infty\}$ is a sentinel used to denote the non existence of a set $Y_k$ which dominates a
       nonempty set $X$; its cardinality is set to $\infty$.
foreach $X \subseteq V$ by increasing order of cardinality do
    for $k = 1$ to $n$ do
        $\mathsf{DS}[X, k] = \left\{ \begin{array}{l} \text{a set of minimum cardinality chosen amongst} \\ \mathsf{DS}[X, k-1] \text{ and } \{v_k\} \cup \mathsf{DS}[X \setminus N[v_k], k-1]. \end{array} \right\}$
```

Corollary 1. *Let $G = (V, E)$ be a graph, f be a wrd-function of G of minimum cost, and V_f its underlying set. Then V_f^2 is a* minimum dominating set *of the vertices non-safely defended by χ^{V_f}.*

Proof. Observe that the underlying set V_f of f is a dominating set of the graph G. By Lemma 1, the set V_f^2 is a minimal dominating set of the vertices non-safely defended by χ^{V_f}. The proof of Lemma 3 shows that if f is a wrd-function of minimum cost then V_f^2 is of minimum size. □

4. Exact algorithms for weak Roman domination

We now describe our exact algorithms solving the Weak Roman Domination problem. Observe that this problem can trivially be solved in $\mathcal{O}^*(3^n)$ time by enumerating all three-partitions of the set of vertices, which constitutes the best known bound for the problem so far. We first present an $\mathcal{O}^*(2^n)$ time and space algorithm, then an $\mathcal{O}^*(2.2279^n)$ time algorithm that only uses polynomial space.

4.1. Using exponential space

We first show that a wrd-function of minimum cost can be computed in $\mathcal{O}^*(2^n)$ time and space. Thanks to Corollary 1, a wrd-function f of minimum cost can be obtained by first guessing its underlying set V_f and then computing a minimum dominating set $V_f^2 \subseteq V_f$ of the vertices non-safely defended by χ^{V_f}. Finding such a set V_f^2 is done by a preprocessing step which involves a dynamic programming inspired by the one given in [18]. This preprocessing step results in an exponential space complexity, which will be reduced to polynomial space in Section 4.2. However, instead of guaranteeing that indeed $V_f^2 \subseteq V_f$, the preprocessing step computes a minimum dominating set V_f^2 of the vertices non-safely defended by χ^{V_f} without constraint, *i.e.* $V_f^2 \subseteq V$ is allowed. We show in Lemma 4 the correctness of this approach. Let us first describe the preprocessing step; its correctness is shown after the description of the main algorithm.

Let $G = (V, E)$ be a graph of the Weak Roman Domination problem, and let $V = \{v_1, v_2, \dots, v_n\}$. For each subset $X \subseteq V$ we start by computing a minimum dominating set Y of X in G, i.e. a subset $Y \subseteq V$ such that $X \subseteq N[Y]$. This is done by dynamic programming: for each subset X and each integer k $(1 \leqslant k \leqslant n)$, $\mathsf{DS}[X, k]$ denotes a minimum dominating set Y_k of X such that $Y_k \subseteq \{v_1, v_2, \dots, v_k\}$, if one exists. Algorithm 1 computes a corresponding table DS by dynamic programming.

4.1.1. Main algorithm

The main steps of our exact algorithm are depicted in Algorithm 2 . For each subset $V' \subseteq V$, we first verify whether $\chi^{V'}$ is (already) a wrd-function, *i.e.*, whether the set $\overline{D}$ of vertices non-safely defended by $\chi^{V'}$ is empty. Otherwise, we need to compute the set V_f^2. The preprocessing step then ensures that $S = \mathsf{DS}[\overline{D}, n]$ is a minimum dominating set of $\overline{D}$. If S is a subset of V', then a wrd-function f can be computed by Lemma 2; otherwise Lemma 4 ensures that there exists some other underlying set V'', being better than V'.

Lemma 4. *Let $V_1' \subseteq V$ be a dominating set of a graph $G = (V, E)$ and let S_1 be a minimum dominating set of the set $\overline{D_1}$ of all vertices non-safely defended by $\chi^{V_1'}$. Suppose that $S_1 \not\subseteq V_1'$. Then there exists a superset $V_2' \supset V_1'$ such that for any minimum dominating set S_2 of the set $\overline{D_2}$ of all vertices non-safely defended by $\chi^{V_2'}$, it holds that cost$(f_2) \leq$ cost(f_1), where f_i $(i \in \{1, 2\})$ is the legion function defined as:*

$$f_i(x) = \begin{cases} 2 & \text{if } x \in V_i' \cap S_i \\ 1 & \text{if } x \in (V_i' \cup S_i) \setminus (V_i' \cap S_i) \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 2: An $\mathcal{O}^*(2^n)$ exponential space algorithm for Weak Roman Domination.

```
foreach dominating set $V' \subseteq V$ do
    foreach $v \in V$ do
        Let $f(v) = 1$ if $v \in V'$, and $f(v) = 0$ otherwise;
    Compute the set $\overline{D}$ of vertices non-safely defended by $\chi^{V'}$;
    if $\overline{D} \neq \emptyset$ then
        $S = \mathsf{DS}[\overline{D}, n]$;
        if $S \subseteq V'$ then
            foreach $v \in S$ do
                Let $f(v) = f(v) + 1$;
return the computed wrd-function $f$ of minimum cost;
```

Proof. Assume that there exist three sets V'_1, S_1 and $\overline{D_1}$ as stated in the lemma and assume that $S_1 \not\subseteq V'_1$. Let $V'_2 = V'_1 \cup S_1$. Since $S_1 \not\subseteq V'_1$, it follows that $V'_2 \supset V'_1$. Let $\overline{D_2}$ be the set of vertices non-safely defended by $\chi^{V'_2}$. Observe that $\overline{D_2} \subseteq \overline{D_1}$, since $V'_1 \subset V'_2$. By Lemma 2, we know that the legion function f_1 is in fact a wrd-function. Hence, by Lemma 1, we also have that $V'_1 \cap S_1$ is a dominating set of $\overline{D_1}$, and thus of $\overline{D_2}$.

Denote by S_2 a minimum dominating set of $\overline{D_2}$. Then $|S_2| \leqslant |V'_1 \cap S_1|$. We now consider the legion function f_2 as defined in the lemma. By Lemma 2, we know that f_2 is a wrd-function. Finally, since $|V'_2| = |V'_1| + |S_1 \setminus V'_1|$ and $|S_2| \leq |V'_1 \cap S_1|$, we conclude the proof by the relation $cost(f_1) = |V'_1| + |S_1| = |V'_1| + |S_1 \setminus V'_1| + |S_1 \cap V'_1| \geq |V'_2| + |S_2| = cost(f_2)$. □

4.1.2. *Correctness*

The correctness of the preprocessing step is based on arguments of [18]. If the set X is empty then the initialization $\mathsf{DS}[\emptyset, k] = \emptyset$, for any $0 \leq k \leq n$, is clearly correct. If the set X is non empty but no vertex can be used to dominate X (i.e. $k = 0$), then $\mathsf{DS}[X, 0]$ is set to $\{\infty\}$ as a sentinel, meaning that there is no set Y (with $Y = \emptyset$) that can dominate X. The cardinality of $\{\infty\}$ is set to ∞. Finally the computation of $\mathsf{DS}[X, k]$ is done via an induction formula: either $v_k \notin \mathsf{DS}[X, k]$ or $v_k \in \mathsf{DS}[X, k]$ and in that latter case, $N(v_k)$ is dominated by v_k. As the sets X are considered by increasing order as well as the values of k, we note that the values $\mathsf{DS}[X, k-1]$ and $\mathsf{DS}[X \setminus N[v_k], k-1]$ have already been computed when the computation of $\mathsf{DS}[X, k]$ is done.

Now we show the correctness of Algorithm 2 . It enumerates all possible sets V' as being possible candidates for the underlying set V_f. In particular, we discard any subset V' that does not induce a dominating set. By Lemma 1, it is sufficient to compute a dominating set $S \subseteq V'$ of the set of vertices $\overline{D}$ being non-safely defended by $\chi^{V'}$. Lemma 4 shows that if S is not included in V', then there exists a proper superset of V' which gives a wrd-function of cost being no more than the one obtained from V' and S (Lemma 2). Let $V'_0 = V'$ and $S_0 = S$. As the graph is finite and the superset given by Lemma 4 is proper, there exists a finite $\ell \leq n$ and a sequence $V'_0 \subset V'_1 \subset \dots \subset V'_\ell \subseteq V$ such that $S_i \not\subseteq V'_i$, for all $0 \leq i < \ell$, and $S_\ell \subseteq V'_\ell$. Since the algorithm enumerates all supersets of V', it follows that the set V'_ℓ will be considered at some iteration of the for-loop. This shows the correctness of Algorithm 2 .

4.1.3. *Complexity*

The preprocessing step needs to consider each subset X of V and each value of k, $1 \leq k \leq n$. For each such couple (X, k), it retrieves the values of $\mathsf{DS}[X, k-1]$ and $\mathsf{DS}[X \setminus N[v_k], k-1]$ previously computed, and stores the new value in DS. Thus the preprocessing step requires $\mathcal{O}^*(2^n)$ time and space. The main part of the algorithm considers each (dominating set) $V' \subseteq V$, and computes in polynomial-time the set $\overline{D}$ of vertices non-safely defended by $\chi^{V'}$. A dominating set S of $\overline{D}$ is then retrieved in the already computed table DS in polynomial-time.

Theorem 1. Weak Roman Domination *can be solved in* $\mathcal{O}^*(2^n)$ *time and space.*

4.2. *Using polynomial space*

In order to obtain an exact exponential algorithm using only polynomial space, we need to avoid any exponential space consuming *preprocessing step* such as the one in the previous section. For this purpose, we use instead an exact exponential algorithm for Red-Blue Dominating Set using polynomial space to decide which vertices will be valued 2 to dominate the non-safely defended vertices.

Red-Blue Dominating Set:
Input: A bipartite graph $G = (R \cup B, E)$.
Output: A subset $S \subseteq R$ of minimum size dominating B.

Algorithm 3: An $\mathcal{O}^*(2.2279^n)$ poly-space algorithm for the Weak Roman Domination problem.

```
foreach dominating set $V' \subseteq V$ do
    foreach $v \in V$ do
        Let $f(v) = 1$ if $v \in V'$, and $f(v) = 0$ otherwise;
    Compute the set $\overline{D}$ of vertices non-safely defended by $\chi^{V'}$;
    if $\overline{D} \neq \emptyset$ then
        Compute the set $\overline{C} \subseteq V'$ of secured vertices which have at least one neighbor in $\overline{D}$;
        /* Cleaning step */
        foreach $v \in \overline{C}$ with at least two weakly non-safely defended neighbors in $\overline{D}$ do
            Set $f(v) = 2$;
            Remove $N_{\overline{D}}(v)$ from $\overline{D}$;
            Remove $v$ from $\overline{C}$;
        Let $I = (\overline{C} \cup \overline{D}, E)$ be an instance of Red-Blue Dominating Set;
        if $I$ admits a minimum red-blue dominating set $S \subseteq \overline{C}$ then
            Set $f(v) = 2$ for every $v \in S$;
        else
            The current function $f$ cannot yield a wrd-function;
return the computed wrd-function $f$ of minimum cost;
```

Theorem 2 ([26]). *The* Red-Blue Dominating Set *problem can be solved in $\mathcal{O}^*(1.2279^{|R|+|B|})$ time and polynomial space.*

4.2.1. Algorithm

We consider the algorithm depicted in Algorithm 3 , which might be seen as some modification of the previous Algorithm 2 .

Observe that before computing a minimum red-blue dominating set on the bipartite graph $(\overline{C} \cup \overline{D}, E)$, we may modify the sets $\overline{C}$ and $\overline{D}$ as follows: for every vertex $v \in \overline{C}$, if v has at least two weakly non-safely defended neighbors, then we set $f(v) = 2$, and remove v from $\overline{C}$ and $N_{\overline{D}}(v)$ from $\overline{D}$.

Proposition 1. *The cleaning step on $\overline{C}$ and $\overline{D}$ does not modify a solution for* Red-Blue Dominating Set *on instance $I = (\overline{C} \cup \overline{D}, E)$.*

Proof. Let $v \in \overline{C}$ be a secured vertex with at least two weakly non-safely defended neighbors, say w_1 and w_2. Since w_1 and w_2 are weakly defended, their only secured neighbor is $v \in V'$; since they are non-safely defended, they need to be dominated by V_f^2 in order for f to be a wrd-function (Lemma 1). Thus we must set $f(v) = 2$. It follows that any minimum red-blue dominating set on instance $I = (\overline{C} \cup \overline{D}, E)$ must put $v \in \overline{C}$ into the red-blue dominating set in order to dominate all weakly non-safely defended neighbors of v in $\overline{D}$.

Now, observe that since all the neighbors of v are safely defended (because dominated by V_f^2), they can safely be removed from $\overline{D}$. Since v has no non-safely defended neighbor left, it can be removed from $\overline{C}$. □

4.2.2. Correctness

The correctness of the algorithm follows from Lemma 1, Corollary 1 and the proof of correctness of Algorithm 2 . The main difference lies in the computation of the dominating set of the vertices non-safely defended by $\chi^{V'}$. Indeed, in that case, we use Theorem 2 to find the vertices of V' that must have value 2 in order to dominate the vertices non-safely defended by $\chi^{V'}$. The correctness of this step follows from Lemma 1 and Proposition 1.

4.2.3. Complexity

Let us now give the time and space complexities of Algorithm 3 . It is easy to see that for every subset $V' \subseteq V$, the initialization of $f(x)$ for every $x \in V$ as well as the computation of the set $\overline{D}$ can be done in polynomial time and space, and that the cleaning step is also polynomial.

Regarding the legion function f being constructed, for any $V' \subseteq V$, our algorithm computes and reduces the set $\overline{D}$ of vertices non-safely defended by $\chi^{V'}$, and the set $\overline{C}$ of secured vertices which have at least one neighbor in $\overline{D}$. Those two sets are considered as an instance of Red-Blue Dominating Set to be solved in $\mathcal{O}^*(1.2279^{|\overline{D}|+|\overline{C}|})$ time and polynomial space using an algorithm from van Rooij [26]. To conclude our analysis, we need the following result.

Proposition 2. *For any $V' \subseteq V$, $|\overline{D}| + |\overline{C}| \leq |V| - |V'|$.*

Proof. For every vertex $v \in V'$, one of the following statements holds:

(i) v has no neighbor in $\overline{D}$, that is no neighbor non-safely defended by $\chi^{V'}$;
(ii) there exists at least one vertex $w \in V \setminus V'$ which is weakly defended by v.

First notice that (i) and (ii) are the only two possible cases. Indeed, if there exists $v \in V'$ such that $N_{\overline{D}}(v) \neq \emptyset$ but no vertex in $V \setminus V'$ is weakly defended by v, then the vertices in $N_{\overline{D}}(v)$ are safely defended, which is a contradiction.

If the first statement holds, then v is not included in $\overline{C}$. If the second statement holds, then either w is safely defended, or w is non-safely defended. If w is safely defended (that is no other neighbor of v is weakly defended by v), then w is not included in $\overline{D}$. If w is non-safely defended, then v has at least two weakly non-safely defended neighbors. Indeed, since w is weakly defended by v (as the second statement holds), v is the only neighbor of w in V'. Hence, there exists a nonempty set $\overline{D}_{v,w} = N_{\overline{D}}(v) \setminus N(w)$ such that w is weakly defended due to each vertex in $\overline{D}_{v,w}$. Now, since w is non-safely defended by v, there must exist a vertex $w' \in \overline{D}_{v,w}$ which is also weakly defended due to w. Then the cleaning step on $\overline{D}$ and $\overline{C}$ applies, which implies that v is removed from $\overline{C}$ and all neighbors of v (including w) are removed from $\overline{D}$. Altogether, for every vertex $v \in V'$, at least one vertex from V is not included in $\overline{C} \cup \overline{D}$, and hence at least $|V'|$ vertices from V are not included in $\overline{D} \cup \overline{C}$. ◇

The overall algorithm iteratively runs all the previously described computations for every subset $V' \subseteq V$, and stores the minimum wrd-function considered so far using polynomial space. We claim that its worst-case time complexity corresponds to the following:

$$\mathcal{O}^*\Big(\sum_{i=1}^{n} \binom{n}{i} \cdot T(n-i)\Big) = \mathcal{O}^*\Big(\sum_{i=1}^{n} \binom{n}{i} \cdot 1.2279^{n-i}\Big) = \mathcal{O}^*\big(2.2279^n\big),$$

where $T(p)$ stands for the time complexity needed to compute a minimum red-blue dominating set in a graph with p vertices (here we use the one of [26]) and since by the well-known *binomial Theorem* $\sum_{i=0}^{n} \binom{n}{i} \lambda^i = \sum_{i=0}^{n} \binom{n}{i} \lambda^{n-i} = (1+\lambda)^n$. Indeed, for any subset $V' \subseteq V$ containing i vertices, we apply Theorem 2 on the bipartite graph induced by $\overline{C}$ and $\overline{D}$, which contain less than $|V| - |V'| = n - i$ vertices (Proposition 2).

Theorem 3. WEAK ROMAN DOMINATION *can be solved in* $\mathcal{O}^*(2.2279^n)$ *time and polynomial space.*

5. A linear-time algorithm on interval graphs

A graph $G = (V, E)$ is an interval graph if G is the intersection graph of a family of n intervals on the real line, where $n = |V|$. In this section, we give a linear-time algorithm to compute a weak Roman dominating function of minimum cost of a given interval graph. Our algorithm is based on a non-trivial greedy strategy which is described in Section 5.1. To achieve a linear running-time, we use a preprocessing being given in Section 5.2.

5.1. Description of the algorithm

Let $G = (V, E)$ be an interval graph of order n. In the next we assume that $G = (V, E)$ is provided together with its normalized model $\mathcal{I}$, *i.e.* all endpoints have distinct values from $\{1, 2, \ldots, 2n\}$. We refer the interested reader to the book by Spinrad for a discussion about (linear-time) recognition algorithms of such graphs [24]. We also refer to vertices and their corresponding intervals interchangeably. For a vertex $v \in V$, we denote by $l(v)$ (respectively $r(v)$) the left endpoint (respectively the right endpoint) of the interval corresponding to v in the model.

For a value $d \in \{1, \ldots, 2n\}$, let $\mathcal{D}_{>d} = \{v \in V : l(v) > d\}$ and let $G_{\leq d}$ be the graph induced by the set of vertices $\{v \in V : l(v) \leq d\}$. Observe that $G_{\leq d}$ is the graph $G[V \setminus \mathcal{D}_{>d}]$.

Given two sets V_1 and V_2, we let $r(V_1, V_2) = \max\{r(v) : v \in V_1 \cup V_2\}$ for $V_1 \cup V_2 \neq \emptyset$. If $V_1 \cup V_2 = \emptyset$ then let $r(V_1, V_2) = 0$.

For $d, d' \in \{1, \ldots, 2n\}$, $mark \in \{\star, -\}$ and $V_1, V_2 \subseteq V$, we define a tuple $\big(d, d', mark, V_1, V_2\big)$ as a *good partial* weak Roman dominating set if the following properties are fulfilled:

[P1] (V_1, V_2) is a minimum weak Roman dominating set of $G_{\leq d}$ such that $r(V_1, V_2)$ is maximum;
[P2] for any $x \in \mathcal{D}_{>d}$, (V_1, V_2) is not a weak Roman dominating set of $G_{\leq d} \cup \{x\}$;
[P3] $r(V_1, V_2) = d'$;
[P4] if *mark* is $\star$ then we require that each vertex of $G_{\leq d} \setminus (V_1 \cup V_2)$ has at least one neighbor in $(V_1 \cup V_2) \setminus \{\kappa\}$, where κ is the vertex in V_1 with $r(\kappa) = d'$.

Given a tuple $t = \big(d, d', mark, V_1, V_2\big)$, naturally we let $cost(t) = cost(V_1, V_2)$. We define $\mathcal{T}$ as the set of tuples which are good partial weak Roman dominating sets, *i.e.* the tuples satisfying the previous four properties [P1]–[P4].

Remark 1 (*On Property [P4]*)**.** Note that if κ is the vertex with $r(\kappa) = d'$ but $\kappa \in V_2$, then *mark* is necessary set to "−". Suppose that a tuple $t = \big(d, d', mark, V_1, V_2\big)$ is a good partial weak Roman dominating set where $mark = \star$ and each vertex of $G_{\leq d} \setminus (V_1 \cup V_2)$ has at least one neighbor in $(V_1 \cup V_2) \setminus \{\kappa\}$, where κ is the vertex with $r(\kappa) = d'$. Assume that $\kappa \in V_2$. Then one can observe that $t' = \big(d, d', mark, V_1 \cup \{\kappa\}, V_2 \setminus \{\kappa\}\big)$ is a good partial weak Roman dominating set satisfying the four properties [P1]–[P4] and $cost(t') < cost(t)$, which contradicts property [P1] of tuple t.

Algorithm 4: WeakRomanDom$(G = (V, E), \mathcal{I})$

Input: An interval graph $G = (V, E)$ and its interval model $\mathcal{I}$.
Output: A WRD set (V_1, V_2) of minimum cost of G.

```
1  Let $(d, d', mark, V_1, V_2) \leftarrow (0, 0, -, \emptyset, \emptyset)$
2  while $\mathcal{D}_{>d}$ is non empty do
3      Let $x \in \mathcal{D}_{>d}$ s.t. $r(x)$ is minimum
4      Let $y \in N[x]$ s.t. $r(y)$ is maximum
5      if $r(y) = d'$ then
6          $(d, d', mark, V_1, V_2) \leftarrow (r(y), r(y), -, V_1 \setminus \{y\}, V_2 \cup \{y\})$
7      else
8          if $mark = \star$ then
9              Let $x'$ s.t. $x' \neq y$, $l(x') > d'$ and $r(x')$ is minimum
10             if there is no such $x'$ or $r(x') \geq r(y)$ then $\overline{d} \leftarrow r(y)$
11             else $\overline{d} \leftarrow r(x')$
12         else if $r(x) < d'$ then $\overline{d} \leftarrow d'$
13         else $\overline{d} \leftarrow r(x)$
14         if there is no $z$ s.t. $z \neq y$ and $d' < l(z) \leq \overline{d}$ then $\overline{mark} \leftarrow \star$
15         else $\overline{mark} \leftarrow -$
16         $(d, d', mark, V_1, V_2) \leftarrow (\overline{d}, r(y), \overline{mark}, V_1 \cup \{y\}, V_2)$
17 return $(V_1, V_2)$ as an optimal solution
```

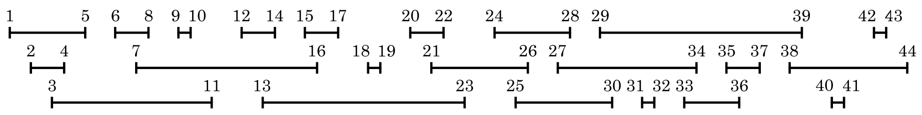


Fig. 2. A step-by-step execution, over the collection of intervals depicted on the figure, produces the following successive tuples: $(0, 0, -, \emptyset, \emptyset)$; $(4, 11, -, \{(3, 11)\}, \emptyset)$; $(11, 16, \star, \{(3, 11), (7, 16)\}, \emptyset)$; $(19, 23, -, \{(3, 11), (7, 16), (13, 23)\}, \emptyset)$; $(23, 26, \star, \{(3, 11), (7, 16), (13, 23), (21, 26)\}, \emptyset)$; $(32, 34, -, \{(3, 11), (7, 16), (13, 23), (21, 26), (27, 34)\}, \emptyset)$; $(36, 39, -, \{(3, 11), (7, 16), (13, 23), (21, 26), (27, 34), (29, 39)\}, \emptyset)$; $(41, 44, -, \{(3, 11), (7, 16), (13, 23), (21, 26), (27, 34), (29, 39), (38, 44)\}, \emptyset)$; $(44, 44, -, \{(3, 11), (7, 16), (13, 23), (21, 26), (27, 34), (29, 39)\}, \{(38, 44)\})$. The algorithm returns $(V_1 = \{(3, 11), (7, 16), (13, 23), (21, 26), (27, 34), (29, 39)\}, V_2 = \{(38, 44)\})$ as an optimal solution.

Given two tuples of $\mathcal{T}$, $t = (d, d', mark\,, V_1, V_2)$ and $\widetilde{t} = (\widetilde{d}, \widetilde{d'}, \widetilde{mark}\,, \widetilde{V_1}, \widetilde{V_2})$, we write $t \prec \widetilde{t}$ whenever $d < \widetilde{d}$. We are now ready to present our greedy algorithm to compute a weak Roman dominating set of an interval graph. It iteratively computes tuples $t_0, t_1, \ldots, t_k$ such that $t_0 \prec t_1 \prec \cdots \prec t_k$. (Observe that this sequence is finite as the value of the first coordinate of each tuple is increasing but smaller than $2n$.) The algorithm starts with the dummy tuple $t_0 = (0, 0, -, \emptyset, \emptyset)$ and iteratively extends the current tuple (which corresponds to a good partial solution) until a weak Roman dominating set (of cost k) of the whole interval graph have been computed. See Algorithm 4 (named **WeakRomanDom** hereafter) and Fig. 2 which depicts an example of its execution.

To show the correctness of our algorithm we prove the following loop invariant.

Lemma 5. *At each iteration of the while-loop in algorithm* ***WeakRomanDom****, the tuple* $(d, d', mark, V_1, V_2)$ *belongs to* $\mathcal{T}$.

Proof. The proof is done by induction on the tuples produced by the algorithm. Let $t_0 = (0, 0, -, \emptyset, \emptyset)$ be the first tuple sets at the initialization step at line 1 of the algorithm. Clearly t_0 fulfills [P1]–[P4] since $V_1 = V_2 = \emptyset$, and thus t_0 belongs to $\mathcal{T}$.

Assume now by induction that $t = (d, d', mark\,, V_1, V_2)$ is a tuple of $\mathcal{T}$ which has been computed at some iteration of the while-loop. Consider $\widetilde{t} = (\widetilde{d}, \widetilde{d'}, \widetilde{mark}\,, \widetilde{V_1}, \widetilde{V_2})$ the new tuple built by algorithm **WeakRomanDom** during the next while-loop iteration. Let x and y be two vertices as described on lines 3 and 4 of algorithm **WeakRomanDom**, *i.e.* let $x \in \mathcal{D}_{>d}$ such that $r(x)$ is minimum and let $y \in N[x]$ such that $r(y)$ is maximum. Since $x \in \mathcal{D}_{>d}$, we have $r(x) > d$. Moreover, by the construction of $\widetilde{d}$ (which pick one of the values as defined on lines 6, 10, 11, 12 or 13), we have $d < \widetilde{d}$. Thus $t \prec \widetilde{t}$ holds. Since t satisfies [P1] and [P2], for any tuple t' such that $t \prec t'$ we necessarily have $cost(t') > cost(t)$. In particular, by the construction of $\widetilde{t}$, we have $cost(\widetilde{t}) = cost(t) + 1$. This is easy to see since in each iteration of the while-loop, either a vertex is moved from V_1 to V_2, or a vertex is added to V_1. Thus, $(\widetilde{V_1}, \widetilde{V_2})$ is a minimum weak Roman dominating set of $G_{<\ell}$ for a certain ℓ. In the next we show that $\ell = \widetilde{d}$. By the construction of $\widetilde{d'}$, we have $\widetilde{d'} = r(y)$ (note that $\widetilde{d'}$ is set either on line 6 or on line 16 of the algorithm). Since $\{y\} = (\widetilde{V_1} \setminus V_1) \cup (\widetilde{V_2} \setminus V_2)$ (*i.e.* y is the new vertex added to V_1 by line 16 or the vertex moved to V_2 by line 6), we have $\widetilde{d'} \geq d' = r(V_1, V_2)$, and so property [P3] is satisfied by $\widetilde{t}$. In the remaining of the proof, we distinguish some cases to prove that $\widetilde{t}$ also satisfies the other properties.

Case 1: "*y is added to* V_2". Suppose first that $r(y) = d'$ and consider the tuple $\left(\widetilde{d}, \widetilde{d'}, \widetilde{mark}, \widetilde{V_1}, \widetilde{V_2}\right) = \left(r(y), r(y), -, V_1 \setminus \{y\}, V_2 \cup \{y\}\right)$, as set by the algorithm on line 6 . We show that this tuple satisfies properties [P1] and [P2]. Recall that by induction hypothesis $t \in \mathcal{T}$ and thus fulfills [P1]–[P4]. Since $x \in \mathcal{D}_{>d}$ such that $r(x)$ is minimum (line 3) and (V_1, V_2) fulfills [P2], it follows that x is the undefended vertex (by (V_1, V_2)) with the smallest right endpoint. Vertex y is greedily chosen among $N[x]$ with the largest right endpoint (line 4) and moved from V_1 to V_2, as it was already in V_1 (line 6). Since y is added to V_2 and (V_1, V_2) is a weak Roman dominating set of $G_{\leq d}$ (by [P1]) it follows that $(\widetilde{V_1}, \widetilde{V_2}) = (V_1 \setminus \{y\}, V_2 \cup \{y\})$ is a weak Roman dominating set of $G_{\leq r(y)}$ where $r(\widetilde{V_1}, \widetilde{V_2}) = d' = r(y)$. By the greedy choice (*i.e.* $y \in N([x])$ such that $r(y)$ is maximum), it is clear that there is no minimum weak Roman dominating set $(\widetilde{V'_1}, \widetilde{V'_2})$ of $G_{\leq r(y)}$ with $r(\widetilde{V'_1}, \widetilde{V'_2}) > r(\widetilde{V_1}, \widetilde{V_2})$. (In other words, any weak Roman dominating set $(\widetilde{V'_1}, \widetilde{V'_2})$ of $G_{\leq r(y)}$ with $r(\widetilde{V'_1}, \widetilde{V'_2}) > r(\widetilde{V_1}, \widetilde{V_2})$ and $cost(\widetilde{V'_1}, \widetilde{V'_2}) = cost(\widetilde{V_1}, \widetilde{V_2})$ either contradicts that (V_1, V_2) fulfills [P1] or leaves x undefended). We have now almost shown that $(\widetilde{V_1}, \widetilde{V_2})$ satisfies [P1]; it remains to show that $(\widetilde{V_1}, \widetilde{V_2})$ is of minimum cost. Observe that (V_1, V_2) is a minimum weak Roman dominating set of $G_{\leq d}$ such that $r(V_1, V_2)$ is maximum (by [P1]) and x is not defended by (V_1, V_2) (by [P2]). Thus, any minimum weak Roman dominating $(\widetilde{V'_1}, \widetilde{V'_2})$ for which x is defended, has $cost(\widetilde{V'_1}, \widetilde{V'_2}) > cost(V_1, V_2)$. Since y has been moved from V_1 to V_2 in $(\widetilde{V_1}, \widetilde{V_2})$, we have $cost(\widetilde{V_1}, \widetilde{V_2}) = cost(V_1, V_2) + 1$. This proves property [P1]. Obviously [P2] is satisfied since no vertex of $\mathcal{D}_{>r(y)}$ has a neighbor in $(\widetilde{V_1}, \widetilde{V_2})$ (recall that $r(\widetilde{V_1}, \widetilde{V_2}) = r(y)$). As $\widetilde{mark} = -$, there is nothing to prove for [P4].

Case 2: "*y is added to* V_1". Assume now that $r(y) \neq d'$ and consider the tuple $\left(\widetilde{d}, \widetilde{d'}, \widetilde{mark}, \widetilde{V_1}, \widetilde{V_2}\right) = \left(\overline{d}, r(y), \overline{mark}, V_1 \cup \{y\}, V_2\right)$, where $\overline{d}$ and $\overline{mark}$ are defined as in the algorithm (lines 8 to 16 of the algorithm). Denote by $\widetilde{x}$ and $\widetilde{y}$ the two vertices considered in the previous iteration of the while-loop. In the remaining of the proof, we denote by f a weak Roman dominating function corresponding to (V_1, V_2); recall that by induction hypothesis $t = (d, d', mark, V_1, V_2) \in \mathcal{T}$ and thus t fulfills [P1]–[P4]. We now distinguish three cases corresponding to the conditional statement on lines 8 , 12 and 13 of the algorithm, which define the value of $\overline{d}$:

- "if $mark = \star$" (line 8). Observe that whenever the algorithm adds a vertex to V_2 it sets $mark$ to $-$; since $mark = \star$ it follows that $\widetilde{y}$ has been added to V_1 in the previous while-loop iteration and $r(\widetilde{y}) = d'$. The reader may find it helpful to refer to Fig. 3.
  By [P4], each vertex of $G_{\leq d} \setminus (V_1 \cup V_2)$ has at least one neighbor in $(V_1 \cup V_2) \setminus \{\kappa\}$, where κ is the vertex in V_1 with $r(\kappa) = d'$, *i.e.* $\kappa = \widetilde{y}$. Observe now that each vertex $z \notin (V_1 \cup V_2 \cup \{y\})$ with $d \leq l(z) \leq d'$ is both adjacent to $\widetilde{y}$ and to y. Moreover since $mark = \star$, by [P4], each neighbor w of $\widetilde{y}$ with $l(w) \leq d$ is also adjacent to another vertex of $V_1 \cup V_2$, or belongs to $V_1 \cup V_2$. Thus z is defended and $f_{\widetilde{y} \to z}$ does not contain an undefended vertex in $G_{\leq \overline{d}}$. Also, any vertex v with $d' < l(v) \leq r(x')$ is defended by y and $f_{y \to v}$ does not contain an undefended vertex in $G_{\leq \overline{d}}$. As a consequence, $(\widetilde{V_1}, \widetilde{V_2}) = (V_1 \cup \{y\}, V_2)$ is a weak Roman dominating set of $G_{\leq \overline{d}}$, where $\overline{d}$ is set to $\min(r(y), r(x'))$ by the algorithm on lines 10 and 11 (depending also on the existence of x' on line 9). By the greedy choice of y (being the neighbor of x with largest right endpoint), and for the same reasons than in the previous case, there is no weak Roman dominating set $\widetilde{V'_1}, \widetilde{V'_2}$ of $G_{\leq \overline{d}}$ with $r(\widetilde{V'_1}, \widetilde{V'_2}) > r(\widetilde{V_1}, \widetilde{V_2})$. Also, any minimum weak Roman dominating set $(\widetilde{V'_1}, \widetilde{V'_2})$, for which the vertices of $G_{\leq \overline{d}} \setminus G_{\leq d}$ are defended, has a cost larger than $cost(V_1, V_2)$ (by [P1] and [P2], this cost cannot be equal to $cost(V_1, V_2)$). Since y is added to V_1, we have $cost(\widetilde{V_1}, \widetilde{V_2}) = cost(V_1, V_2) + 1$. So we proved that $(\widetilde{V_1}, \widetilde{V_2})$ is a minimum weak Roman dominating set of $G_{\leq \overline{d}}$, where $r(\widetilde{V_1}, \widetilde{V_2})$ is maximum, and thus [P1] is thus fulfilled. Property [P2] is also fulfilled as any vertex v with $l(v) > \overline{d}$ either has no neighbor in $\widetilde{V_1} \cup \widetilde{V_2}$ if $\overline{d} = r(y)$; or the function $f_{y \to v}$ would leave x' as undefended if $\overline{d} = r(x')$.
- "if $r(x) < d'$" (line 12). We can assume now that $mark = -$. We observe that, by the choice of x and y (lines 3 and 4), any vertex w such that $d < l(w) \leq d'$ is both adjacent to $\widetilde{y}$ and to y. Let $\overline{d} = d'$ (as on line 12). The pair $(\widetilde{V_1}, \widetilde{V_2}) = (V_1 \cup \{y\}, V_2)$ is a weak Roman dominating set of $G_{\leq \overline{d}}$ as any vertex z with $d < l(z) \leq d'$ is defended and $f_{y \to z}$ does not contain an undefended vertex in $G_{\leq \overline{d}}$. By the greedy choice of y and by the same argument than before on the cost of the solution it follows that $(\widetilde{V_1}, \widetilde{V_2})$ fulfilled [P1]. Property [P2] is also fulfilled as for any vertex v with $l(v) > d'$ the function $f_{y \to x}$ would leave v undefended (and by induction $f_{\widetilde{y} \to x}$ would leave a vertex undefended since $mark = -$).
- "otherwise" (line 13). In this latter case, we let $\overline{d} = r(x)$ (as on line 13) and we show that $(\widetilde{V_1}, \widetilde{V_2}) = (V_1 \cup \{y\}, V_2)$ is a minimum weak Roman dominating set of $G_{\leq \overline{d}}$. Indeed, one can observe that the set of vertices z with $d < l(z) \leq r(x)$ form a clique (which also contains y). This is due to the choice of x and y on lines 3 and 4 : Any vertex z with $d < l(z) \leq r(x)$ satisfies $l(z) \leq r(x) \leq r(z) \leq r(y)$. So $f_{y \to z}$ does not contain undefended vertices in $G_{\leq \overline{d}}$. Thus $(\widetilde{V_1}, \widetilde{V_2}) = (V_1 \cup \{y\}, V_2)$ is a weak Roman dominating set of $G_{\leq \overline{d}}$. Again, since x is the undefended (by (V_1, V_2)) vertex in $G_{\leq d}$ with smallest right endpoint, and by the greedy choice of y (which is chosen among $N[x]$ so that $r(y)$ is maximum), there is no weak Roman dominating set $\widetilde{V'_1}, \widetilde{V'_2}$ of $G_{\leq \overline{d}}$ with $r(\widetilde{V'_1}, \widetilde{V'_2}) > r(\widetilde{V_1}, \widetilde{V_2})$. As before, any minimum weak Roman dominating set $(\widetilde{V'_1}, \widetilde{V'_2})$, for which x is defended, has a cost larger than $cost(V_1, V_2)$ (by [P1] and [P2]). Adding y to V_1 gives $cost(\widetilde{V_1}, \widetilde{V_2}) = cost(V_1, V_2) + 1$. Thus [P1] is fulfilled. It is also clear that [P2] is fulfilled as for any vertex v with

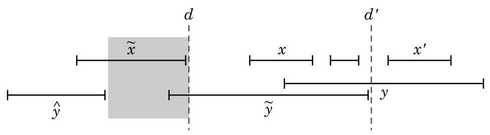


Fig. 3. A set of intervals corresponding to a tuple $\left(d, d', \star, V_1, V_2\right)$ where $\{\widehat{y}, \widetilde{y}\} \subseteq V_1$. Vertices $\widetilde{x}$ and $\widetilde{y}$ are the two vertices considered in the previous iteration of the while-loop, *i.e.* $\widetilde{y}$ is the last vertex added to V_1. Vertex $\widehat{y}$ is the vertex added to V_1 right before $\widetilde{y}$. Let x, y and x' be the vertices chosen by algorithm **WeakRomanDom** during the next while-loop iteration. By [P4], there is no vertex (except eventually $\widetilde{y}$) whose left endpoint is in the shaded area.

$l(v) > r(x)$ the function $f_{y\to x}$ would leave v undefended (and by induction $f_{\widetilde{y}\to x}$ would leave a vertex undefended in $G_{\leq d}$).

Finally observe that in all these cases, if there is no z such that $z \neq y$ and $\widetilde{d'} < l(z) \leq \overline{d}$ (which we also consider to be true when $d' = \overline{d}$), then property [P4] is fulfilled and we let $\widetilde{mark} = \star$, otherwise $\widetilde{mark} = -$. This conclude the proof. □

Lemma 6. *After a finite number of iterations, algorithm* ***WeakRomanDom*** *ends with a tuple* $\left(d, d', mark, V_1, V_2\right)$*, where* (V_1, V_2) *is a minimum weak Roman dominating set of the input graph* G*.*

Proof. The algorithm halts as soon as a tuple $t = \left(d, d', mark, V_1, V_2\right)$ has been computed such that $\mathcal{D}_{>d}$ is empty. By Lemma 5, $t \in \mathcal{T}$ and (V_1, V_2) is a minimum weak Roman dominating set of the input graph G.

Since the value of the first coordinate of each tuple successively computed by the algorithm is increasing, but smaller than $2n$, one can observe that this tuple t is obtained after a finite number of iterations. □

Theorem 4. *Algorithm* ***WeakRomanDom*** *computes a WRD set* (V_1, V_2) *of minimum cost of a given interval graph and can be implemented to run in linear-time.*

Proof. Lemmata 5 and 6 show the correctness of the algorithm. The running-time analysis is provided in the next section. □

5.2. Running-time analysis

Next Lemma 7 shows that a linear number of good partial weak Roman dominating sets are computed during an execution of algorithm **WeakRomanDom**. Clearly, all operations can be implemented in polynomial-time and thus the algorithm works in polynomial-time. In the remaining of the section, we show that the algorithm can be implemented so that the running-time is linear. This speed-up is achieved by using a preprocessing. Using a preprocessing is not a new idea as this was already used, *e.g.* in [15] to solve the minimum dominating set problem on circular-arc graphs in linear-time. In the next of the section we further extend the preprocessing given in [19] (for the Roman domination problem on interval graphs), to show that **WeakRomanDom** can run in linear-time.

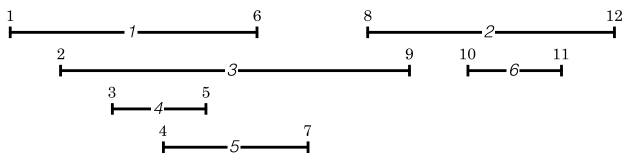
Lemma 7. *The number of iterations of the while-loop in algorithm* ***WeakRomanDom*** *is at most* n*, where* n *is the number of intervals in the input model* $\mathcal{I}$*.*

Proof. Observe that during an iteration of the while-loop, a tuple $t = \left(d, d', mark, V_1, V_2\right)$ is replaced by a new computed tuple $\widetilde{t} = \left(\widetilde{d}, \widetilde{d'}, \widetilde{mark}, \widetilde{V_1}, \widetilde{V_2}\right)$ such that $\widetilde{d} > d$, *i.e.* it respects the relation $t \prec \widetilde{t}$. Moreover, these values of d and $\widetilde{d'}$ correspond to the right endpoints of some intervals. Since there are at most n different values from $\{1, 2 \ldots, 2n\}$ corresponding to a right endpoint, this show the lemma. □

To speed up the algorithm and achieve a linear running-time, we have to compute the following operations in constant time during each while-loop iteration:

- for a given $d \in \{0, 1, \ldots, 2n\}$, decide whether $\mathcal{D}_{>d}$ is empty;
- for a given $d \in \{0, 1, \ldots, 2n\}$, find a vertex $x \in \mathcal{D}_{>d}$ such that $r(x)$ is minimum;
- for a given vertex x, find a vertex $y \in N[x]$ such that $r(y)$ is maximum;
- for a given $d' \in \{0, 1, \ldots, 2n\}$ and a given vertex y, find a vertex x' such that $x' \neq y$, $l(x') > d'$ and $r(x')$ is minimum;
- for some given $d', \overline{d} \in \{0, 1, \ldots, 2n\}$ and a given vertex y, determine whether there is a vertex z such that $z \neq y$ and $d' < l(z) \leq \overline{d}$.

Suppose that the intervals of $\mathcal{I}$ are arbitrarily numbered from 1 to n. We first apply a linear-time preprocessing over the interval model $\mathcal{I}$ so that the previous operations can be supported in constant time. This preprocessing, named **PreProcessing**, is described by Algorithm 6 which first make use of a subroutine **PreProcessing-Sort** (see Algorithm 5) to sort the intervals according to their left and right endpoints. Given an interval graph (and its interval model), **PreProcessing-Sort** constructs four arrays T'_L, T'_R, T_L and T_R:



$l:$	1	8	2	3	4	6

$r:$	6	12	9	5	7	11

$x:$	1	2	3	4	5	
MaxR :	3	2	2	3	3	2

$d:$	0	1	2	3	4	5	6	7	8	9	10	11	12
MinR :	4	4	4	5	6	6	6	6	6	6	NIL	NIL	NIL
	1	5	5	6	2	2	2	2	NIL	NIL	NIL	NIL	NIL

$d':$	0	1	2	3	4	5	6	7	8	9	10	11	12
MinL :	1	3	4	5	2	2	2	2	6	6	NIL	NIL	NIL
	3	4	5	2	6	6	6	6	NIL	NIL	NIL	NIL	NIL

Fig. 4. An interval collection given by a normalized model. The corresponding left and right endpoints of the intervals are given by l and r. The arrays MaxR, MinR and MinL are the ones being computed during the preprocessing by algorithm **PreProcessing**. For the interval graph corresponding to the given collection, algorithm **WeakRomanDom** iteratively computes the tuples $t_0 = (0, 0, -, \emptyset, \emptyset)$, $t_1 = (5, 9, -, \{3\}, \emptyset)$ and $t_2 = (11, 12, -, \{3, 6\}, \emptyset)$, leading to the optimal solution $(V_1, V_2) = (\{3, 6\}, \emptyset)$.

Algorithm 5: PreProcessing-Sort$(G = (V, E), \mathcal{I})$

Input: An interval graph and its interval model $\mathcal{I}$.
Output: Arrays T'_L, T'_R, T_L, T_R needed for the preprocessing

```
1  Let $T'_L, T'_R$ be two arrays of size $2n$ initialized with NIL
2  Let $T_L, T_R$ be two arrays of size $n$
3  foreach $x \in V$ do
4      $T'_L[l(x)] \leftarrow x$
5      $T'_R[r(x)] \leftarrow x$
6  $index_L, index_R \leftarrow 1$
7  for $k = 1$ to $2n$ do
8      if $T'_L[k] \neq$ NIL then
9          $T_L[index_L] \leftarrow T'_L[k]$
10         $index_L \leftarrow index_L + 1$
11     if $T'_R[k] \neq$ NIL then
12         $T_R[index_R] \leftarrow T'_R[k]$
13         $index_R \leftarrow index_R + 1$
14 return $(T'_L, T'_R, T_L, T_R)$
```

- for all $i \in \{1, 2, \ldots, 2n\}$, $T'_L[i]$ (resp. $T'_R[i]$) contains the (number of the) interval x such that $l(x) = i$ (resp. $r(x) = i$) if such an interval x exists, and is equal to NIL otherwise;
- tables T_L and T_R are of size n, and contains the n intervals sorted according to respectively their left endpoint and their right endpoints.

Algorithm **PreProcessing** (see also Fig. 4 of a detailed example) outputs three arrays MaxR, MinR and MinL such that :

- for all $x \in \{1, 2, \ldots, n\}$, MaxR$[x]$ is equal to the (number of the) interval y such that $y \in N[x]$ and $r(y)$ is maximum;
- for all $d \in \{0, 1, \ldots, 2n\}$, MinR$[1][d]$ and MinR$[2][d]$ are the (numbers of the) two intervals x and x' such that $x, x' \in \mathcal{D}_{>d}$, $x \neq x'$ and $r(x)$, $r(x')$ are minimum (these values are set to NIL if they are undefined);
- for all $d' \in \{0, 1, \ldots, 2n\}$, MinL$[1][d']$ and MinL$[2][d']$ are the (numbers of the) two intervals z and z' such that $z, z' \in \mathcal{D}_{>d'}$, $z \neq z'$ and $l(z)$, $l(z')$ are minimum (these values are set to NIL if they are undefined).

It is easy to see that Algorithm **PreProcessing** runs in linear-time. Its correctness is straightforward and mostly technical (see [19] for a preprocessing with the same kind of flavor). It remains to explain how arrays MaxR, MinR and MinL may be used in algorithm **WeakRomanDom**. First, the condition of the while-loop (*i.e.* checking whether $\mathcal{D}_{>d}$ is non-empty), can be done by inspecting the value of MinR$[1][d]$, which should be different from NIL. Then, Lines 3 and 4 can be replaced by

Algorithm 6: PreProcessing($G = (V, E), \mathcal{I}$)

```
**Input**: An interval graph and its interval model $\mathcal{I}$.
**Output**: Arrays MaxR, MinR and MinL
1  $(T'_L, T'_R, T_L, T_R) \leftarrow$ **PreProcessing-Sort**$(G = (V, E), \mathcal{I})$
2  Let MaxR be an array of size $n$
3  $k, index \leftarrow n$
4  while $k \geq 1$ do
5      if $l(T_R[index]) \leq r(T_R[k])$ then
6          MaxR$[T_R[k]] \leftarrow T_R[index]$
7          $k \leftarrow k - 1$
8      else $index \leftarrow index - 1$
9  Let MinR be an array of size $2 \times (2n + 1)$ initialized with NIL
10 $index \leftarrow 0$
11 for $k = 1$ to $2n$ do
12     if $T'_R[k] \neq$ NIL then
13         while $index < l(T'_R[k])$ do
14             MinR$[1][index] \leftarrow T'_R[k]$
15             $index \leftarrow index + 1$
16 $index \leftarrow 0$
17 for $k = r($MinR$[1][0]) + 1$ to $2n$ do
18     if $T'_R[k] \neq$ NIL then
19         $prev \leftarrow$ MinR$[1][index]$
20         while $l(T'_R[k]) > index$ and MinR$[1][index] = prev$ do
21             MinR$[2][index] \leftarrow T'_R[k]$
22             $index \leftarrow index + 1$
23 Let MinL be an array of size $2 \times (2n + 1)$ initialized with NIL
24 $index \leftarrow 1$
25 for $k = 0$ to $2n$ do
26     while $index \leq n$ and $l(T_L[index]) \leq k$ do $index \leftarrow index + 1$
27     if $index \leq n$ then MinL$[1][k] \leftarrow T_L[index]$
28     if $index \leq n - 1$ then MinL$[2][k] \leftarrow T_L[index + 1]$
29 return (MaxR, MinR, MinL)
```

$x \leftarrow \texttt{MinR}[1][d]$ and $y \leftarrow \texttt{MaxR}[x]$. Line 9 can be replaced by

$$x' \leftarrow \begin{cases} \texttt{MinR}[1][d'] & \text{if } \texttt{MinR}[1][d'] \neq y\,; \\ \texttt{MinR}[2][d'] & \text{otherwise.} \end{cases}$$

Finally, testing the condition of line 14 (*i.e.* there is no z s.t. $z \neq y$ and $d' < l(z) \leq \overline{d}$) can be done by evaluating the formula $\big(\texttt{MinL}[1][d'] = \text{NIL}$ or $(\texttt{MinL}[1][d'] = y$ and $\texttt{MinL}[2][d'] = \text{NIL})$ or $(\texttt{MinL}[1][d'] = y$ and $l(\texttt{MinL}[2][d']) > \overline{d})$ or $(\texttt{MinL}[1][d'] \neq y$ and $l(\texttt{MinL}[1][d']) > \overline{d})\big)$.

References

[1] E. Chambers, B. Kinnersley, N. Prince, D. West, Extremal problems for roman domination, SIAM J. Discrete Math. 23 (3) (2009) 1575–1586.

[2] M. Chapelle, M. Cochefert, J. Couturier, D. Kratsch, M. Liedloff, A. Perez, Exact algorithms for weak roman domination, in: T. Lecroq, L. Mouchard (Eds.), Combinatorial Algorithms - 24th International Workshop, IWOCA, 2013, Rouen, France, July 10-12, 2013, Revised Selected Papers, in: Lecture Notes in Computer Science, vol. 8288, Springer, 2013, pp. 81–93. http://dx.doi.org/10.1007/978-3-642-45278-9_8.

[3] M. Chellali, N. Rad, L. Volkmann, Some results on roman domination edge critical graphs, AKCE Int. J. Graphs Comb. 9 (2) (2012) 195–203.

[4] E. Cockayne, O. Favaron, C. Mynhardt, Secure domination, weak roman domination and forbidden subgraphs, Bull. Inst. Combin. Appl. 39 (2003) 87–100.

[5] E. Cockayne, P. Grobler, W. Gründlingh, J. Munganga, J. van Vuuren, Protection of a graph, Util. Math. 67 (2005) 19–32.

[6] E. Cockayne, P.D. Jr, S. Hedetniemi, S. Hedetniemi, Roman domination in graphs, Discrete Math. 278 (1–3) (2004) 11–22.

[7] O. Favaron, K. Karami, R. Khoeilar, S. Sheikholeslami, On the roman domination number of a graph, Discrete Math. 309 (2009) 3447–3451.

[8] F. Fomin, F. Grandoni, D. Kratsch, A measure & conquer approach for the analysis of exact algorithms, J. ACM 56 (5) (2009).

[9] M. Garey, D. Johnson, Computers and Intractability, Freeman, 1979.

[10] W. Goddard, S. Hedetniemi, S. Hedetniemi, Eternal security in graphs, J. Combin. Math. Combin. Comput. 52 (2005) 160–180.

[11] J. Goldwasser, W. Klostermeyer, Tight bounds for eternal dominating sets in graphs, Discrete Math. 308 (2008) 2589–2593.
[12] P. Grobler, C. Mynhardt, Secure domination critical graphs, Discrete Math. 309 (2009) 5820–5827.
[13] T. Haynes, S. Hedetniemi, P. Slater, Domination in Graphs: Advanced Topics, in: Pure and Applied Mathematics, vol. 209, Marcel Dekker Inc, 1998.
[14] M.A. Henning, S.T. Hedetniemi, Defending the roman empire–a new strategy, Discrete Math. 266 (1–3) (2003) 239–251. http://dx.doi.org/10.1016/S0012-365X(02)00811-7.
[15] W. Hsu, K. Tsai, Linear time algorithms on circular-arc graphs, Inform. Process. Lett. 40 (3) (1991) 123–129. http://dx.doi.org/10.1016/0020-0190(91)90165-E.
[16] Y. Iwata, A faster algorithm for dominating set analyzed by the potential method, in: IPEC'11, Proceedings of the 11th International Symposium on Parameterized and Exact Computation, in: LNCS, vol. 7112, 2011, pp. 41–54.
[17] M. Liedloff, Algorithmes exacts et exponentiels pour les problèmes NP-difficiles: domination, variantes et généralisations, in: Laboratoire D'Informatique Théorique et Appliquée, Université Paul Verlaine, Metz, 2007 (Ph.d. thesis).
[18] M. Liedloff, Finding a dominating set on bipartite graphs, Inform. Process. Lett. 107 (5) (2008) 154–157.
[19] M. Liedloff, T. Kloks, J. Liu, S. Peng, Efficient algorithms for roman domination on some classes of graphs, Discrete Appl. Math. 156 (18) (2008) 3400–3415. http://dx.doi.org/10.1016/j.dam.2008.01.011.
[20] C.-H. Liu, G. Chang, Roman domination on 2-connected graphs, SIAM J. Discrete Math. 26 (1) (2012) 193–205.
[21] C.-S. Liu, S.-L. Peng, C.Y. Tang, Weak roman domination on block graphs, in: Proceedings of the 27th Workshop on Combinatorial Mathematics and Computation Theory, Providence University, Taichung, Taiwan, April 30–May 1, 2010, 2010, pp. 86–89.
[22] T.N.M.M. Mai, P.R.L. Pushpam, Weak roman domination in graphs, Discuss. Math. Graph Theory 31 (1) (2011) 161–170. http://dx.doi.org/10.7151/dmgt.1532.
[23] C. ReVelle, K. Rosing, Defendens imperium Romanum: a classical problem in military strategy, Math. Assoc. Amer. 107 (7) (2000) 585–594.
[24] J. Spinrad, Efficient Graph Representations, in: Fields Institute Monographs, American Mathematical Soc, 2003.
[25] I. Stewart, Defend the Roman empire!, Sci. Am. 281 (6) (1999) 136–139.
[26] J. van Rooij, Exact Exponential-Time Algorithms for Domination Problems in Graphs, Utrecht University, Netherlands, 2011 (Ph.d. thesis).
[27] J. van Rooij, H. Bodlaender, Exact algorithms for dominating set, Discrete Appl. Math. 159 (17) (2011) 2147–2164.
[28] H.-M. Xing, X. Chen, X.-G. Chen, A note on roman domination in graphs, Discrete Math. 306 (2006) 3338–3340.