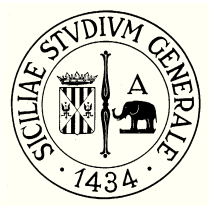


Università degli Studi di Catania

Dipartimento di Matematica e Informatica

Dottorato di Ricerca in Informatica

Vincenzo Currò

The Roman Domination Problem on Grid Graphs

A dissertation submitted to the Department of Mathematics and Computer Science and the committee on graduate studies of University of Catania, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in computer science.

Advisor:

Chiar.mo Prof. Vincenzo Cutello

Coordinator:

Chiar.mo Prof. Vincenzo Cutello

XXV Ciclo

Abstract

Domination is a rapidly developing area of research in graph theory. This dissertation focuses on the Roman Domination Problem; it was introduced quite recently and has some interesting applications in real world problems such military strategies and wireless networking.

Given a graph, a Roman Dominating Function is a function that labels the vertices of the graph with an integer between 0, 1, 2, satisfying the condition that every vertex labeled by 0 is adjacent to at least one vertex labeled by 2. The weight of a Roman Dominating Function is the sum of all the labels, and the minimum weight is called the Roman Domination Number. The Roman Domination Problem is to find such number and function.

In this dissertation we study the Roman Domination Problem when restricted to the class of grid graphs, i.e. graphs that, when drawn on an Euclidean Plane, form a specific regular tiling. A review of well–known results is given, and new results are presented. We aimed to find an algorithm that can find an exact solution for all the grid graphs, and, to do so, we present some important results: we prove a better lower-bound and present an upper-bound on the Roman Domination Number which improves the previous one and, we conjecture, is the Roman Domination Number for many, if not all, grid graphs.

Sommario

La Dominazione è un'area della ricerca nella teoria dei grafi in rapida evoluzione. Questa dissertazione si propone di studiare il Problema della Dominazione Romana: introdotto di recente, questo problema ha alcune interessanti applicazioni nel mondo reale, quali le strategie militari e lo studio delle reti senza fili.

Dato un grafo non orientato, una Funzione di Dominazione Romana è una funzione che etichetta i vertici con un numero intero compreso tra 0, 1, 2, e che inoltre soddisfa la seguente condizione: ogni vertice etichettato 0 deve essere adiacente ad almeno un vertice etichettato 2. Il peso di una Funzione di Dominazione Romana è la somma di tutte le etichette, e il peso minimo è detto Numero di Dominazione Romana. Il Problema della Dominazione Romana è proprio quello di trovare tali numero e funzione. Dopo aver elencato alcuni risultati salienti, studieremo il Problema della Dominazione Romana restringendo il campo ai grafi a griglia, cioè grafi che, quando rappresentati su di un Piano Euclideo, formano una specifica copertura regolare.

Il nostro obbiettivo è quello di trovare un algoritmo che possa trovare una soluzione esatta per tutti i grafi a griglia, e per fare questo presentiamo alcuni risultati notevoli: proveremo un limite inferiore migliore al Numero di Dominazione Romana e presenteremo un limite superiore che migliora il limite precedente e che noi congetturiamo sia il Numero di Dominazione Romana per molti, se non tutti, i grafi a griglia.

Acknowledgements

So here we are, after a long series of satisfactions and sacrifices, to the last page of this dissertation, which coincides with the end of my graduate studies; during these years I had the opportunity to expand my knowledge both scientific and personal. So I want to remember and thank all those people who were close to me, supporting me during all this time.

Much appreciation and gratitude go to my advisor, Prof. Vincenzo Cutello, who patiently followed me during these years, and to the teachers that I got to know and esteem during these years.

I would like to thank my family, who allowed me to get where I am: my parents, who died a short time ago; they are not here physically to see me reach this milestone, but I am sure they are watching me from up there; my brother Francesco, who always encouraged me to go ahead; and my aunt Francesca, who is for me a second mother.

Thank you Mario, colleague and traveling companion, with whom I shared not only the office; thank you Alessandro, Lorenzo, Jole, Gianni and all the other colleagues who were always present and with whom I shared many good times.

I sincerely thank all the "old" friends: Carmelo, Antonio, Walter, Nino, Angelo, Carmelo, Salvo, Eugenio, Alessandra, Lisa and their "new" families, who have always supported me and always will. We are getting older.

Furthermore I thank the friends I shared many wonderful adventures with; they always will have a special place in my heart: Antonio, Dora, Carmelo, Valentina, Dino, Lucrezia, Elisa, Carmela, Valeria, Daniele, Katia and Andrea.

To my parents,

in loving memory.

Contents

List of Figures

List of Tables

Introduction

"Optimization is the art of compromise."

- William A. Dembski -

Combinatorial Optimization is a field of applied mathematics and theoretical computer science that consists in the search of an optimal solution in a finite set of objects, so in principle to solve them it is possible to use any kind of search algorithm or *heuristic*. In computer science, a heuristic is a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution. Combinatorial Optimization is a well studied field because it has important applications in many other fields, like software engineering, machine learning, artificial intelligence, etc...

However for many problems related to Combinatorial Optimization it is not feasible to do an exhaustive research, because generic search algorithms are not guaranteed to find an optimal solution, nor are they guaranteed to run quickly, i.e. in polynomial time. Some famous combinatorial problem are the *Minimum Spanning Tree Problem*, the *Traveling Salesman Problem*, the *Knapsack Problem*, the *Eight Queens Puzzle*, the *Domination Problem*.

Unless $\mathcal{P} = \mathcal{NP}$, we can divide the Combinatorial Optimization Problems in: problems that have a *polynomial-time algorithms* that find the exact

solution, like the Minimum Spanning Tree; and problems that do not have such algorithms, like the Knapsack Problem or the Traveling Salesman Problem. For the last kind of problems we can however have:

- polynomial-time exactly solvable special cases of the problem;
- algorithms that perform well on *random* instances;
- *approximation algorithms* that run in polynomial time and find a solution that is close to optimal.

Since the XX century the *Domination Problem* is one of the most studied Combinatorial Optimization Problem; its origins can be related to the Eight Queens Puzzle and the *Domination Queens Problem*, two problems related to the game of chess. In Graph theory, a *dominating set* for a graph is a subset of vertices such that any other vertex of the graph is adjacent to at least one vertex in the subset and the *domination number* is the number of vertices in a smallest dominating set. The Domination Problem is to find such number, and because it is a $\mathcal{NP}$-hard problem, for a proof see [47], it is believed there is no efficient algorithm to find a smallest dominating set for a given graph.

Among the many variants of the Domination Problem, the *Roman Domination Problem*, introduced quite recently, has some interesting applications in real world problems such military strategies and wireless networking.

Given a graph, a *Roman Dominating Function* is a function that labels the vertices of the graph with an integer between $0, 1, 2$, satisfying the condition that every vertex labeled by 0 is adjacent to at least one vertex labeled by 2. The weight of a Roman Dominating Function is the sum of all the labels, and the minimum weight is called the *Roman Domination Number*. The Roman Domination Problem is to find such number and function.

In the first chapter of this dissertation we introduce the Roman Domination Problem, give a historical motivation for its name, and enumerate some of

its theoretical properties; we also give a demonstration of its complexity, by reduction from the well known *3–SAT Problem*, and a non-deterministic polynomial time algorithm for solving the problem.

In the second chapter we amply study the Roman Domination Problem when restricted to the class of *grid graphs*. The grid graphs are graphs that, when drawn on an Euclidean Space, form a regular tiling. We will always use the *square grid graphs*, whose vertices corresponds to the points of the Euclidean Plane with integer coordinates and two vertices are connected by an edge if and only if the corresponding points are at distance 1; also we will always refer to a finite section of the infinite graph, like *the grid graph of m rows and n columns*.

For grid graphs of m rows and n columns, with $1 \leq m \leq 8$, there exist polynomial-time algorithms that find an exact solution of the Roman Domination Problem. For generic grid graphs, i.e. grid graphs of m rows and n columns, with $m, n \geq 5$, also exist the *lower-* and *upper-bounds* for the Roman Domination Number. We aimed to find an algorithm that can find an exact solution for generic grid graphs.

In the third and the fourth chapters we present some important results, in fact in the third chapter we prove a better lower-bound for generic grid graphs, while in the fourth chapter we define and then discuss some general schemas to produce good Roman Dominating Functions over generic grid graphs that lead to an upper-bound on the Roman Domination Number which improves the previous upper-bound and, we conjecture, is the Roman Domination Number for many, if not all, generic grid graphs.

Finally in the fifth chapter we present some implementations in high-level languages that helped us with our studies: first of all we created a dataset of various classes of graphs, not only grid graphs, then implemented some algorithms that can be used as heuristic for finding an approximate solution

to the Roman Domination Problem. The most performant heuristic uses the concept of *GainFactor*, i.e. a dynamic value assigned to every vertex. We also developed some genetic algorithms, using a virus variant of the *Uniform Crossover*.

Chapter 1

The Roman Domination Problem

1 Introduction

As we said in the introductory chapter, in this dissertation we will talk about the *Roman Domination Problem* (RDP), a combinatorial optimization problem that was introduced fairly recently by Stewart in [53]. The Roman Domination Problem is a variation of the *Domination Problem* (DP), a problem that has been well studied, see [30] and [31].

Given a graph, i.e. an ordered pair $G = (V, E)$ where V is a set, elements of which are called vertices or nodes, and E is a set of unordered pairs of distinct vertices called edges or lines, the Domination Problem is to find a sub-set of vertices $S \subseteq V$, the *dominating set*, such that every other vertex is linked by an edge to at least a member of the sub-set, i.e. every vertex $v \in V$ either is in S or is adjacent to a vertex in S. A *minimal dominating set* in a graph G is a dominating set that contains no dominating set as a proper subset. The Domination Number $\gamma(G)$ of a graph is the number of vertices in a smallest dominating set.

Let f be a function $f\colon V \to \{0, 1\}$, the function f is called a *dominating*

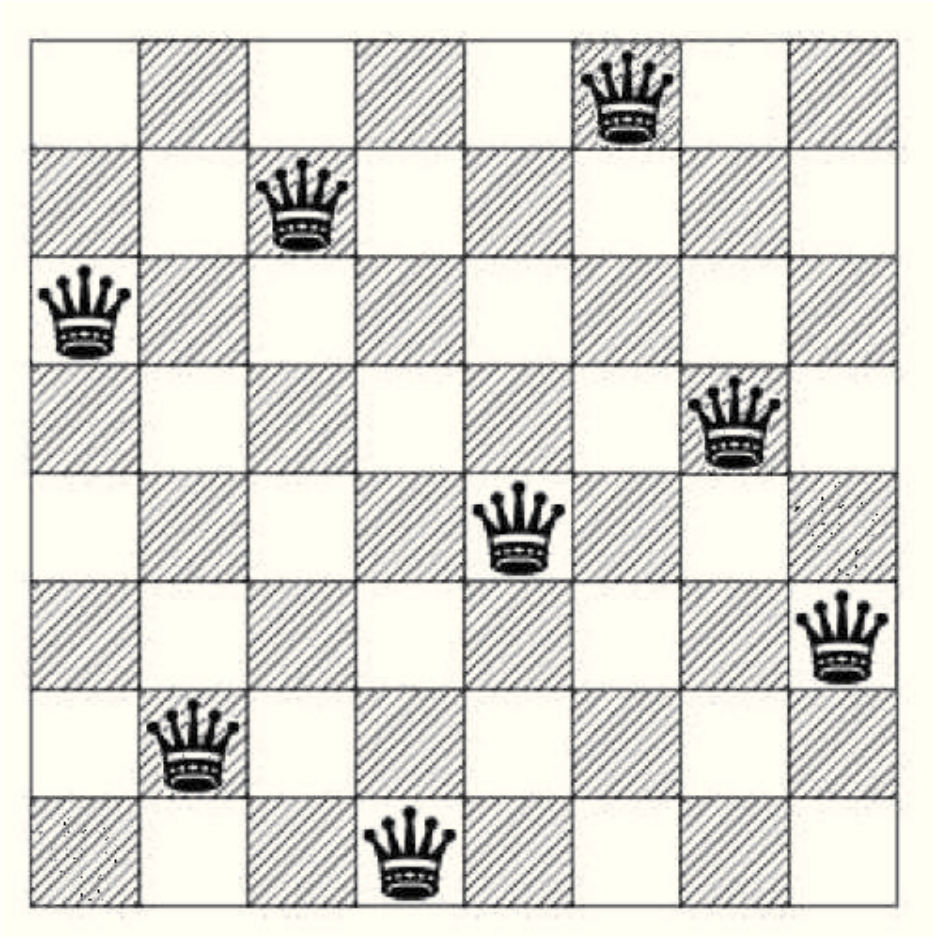


Figure 1.1: The Eight Queens Puzzle, a possible solution.

function if:

$$\forall\, v \in V\colon f(v) = 0 \quad \exists\, u \in V\colon f(u) = 1 \wedge e(u,v) \in E\ .$$

The function f is *minimal* if there does not exist a dominating function $g \neq f$ such that $g(v) \leq f(v)$ for each $v \in V$. The Domination Number can be defined as:

$$\gamma(G) = \min_f \sum_{v \in V} f(v)$$

In [30] are listed over 1.200 papers related to Domination in graphs, and several hundred other papers on the topic have been written since its publication.

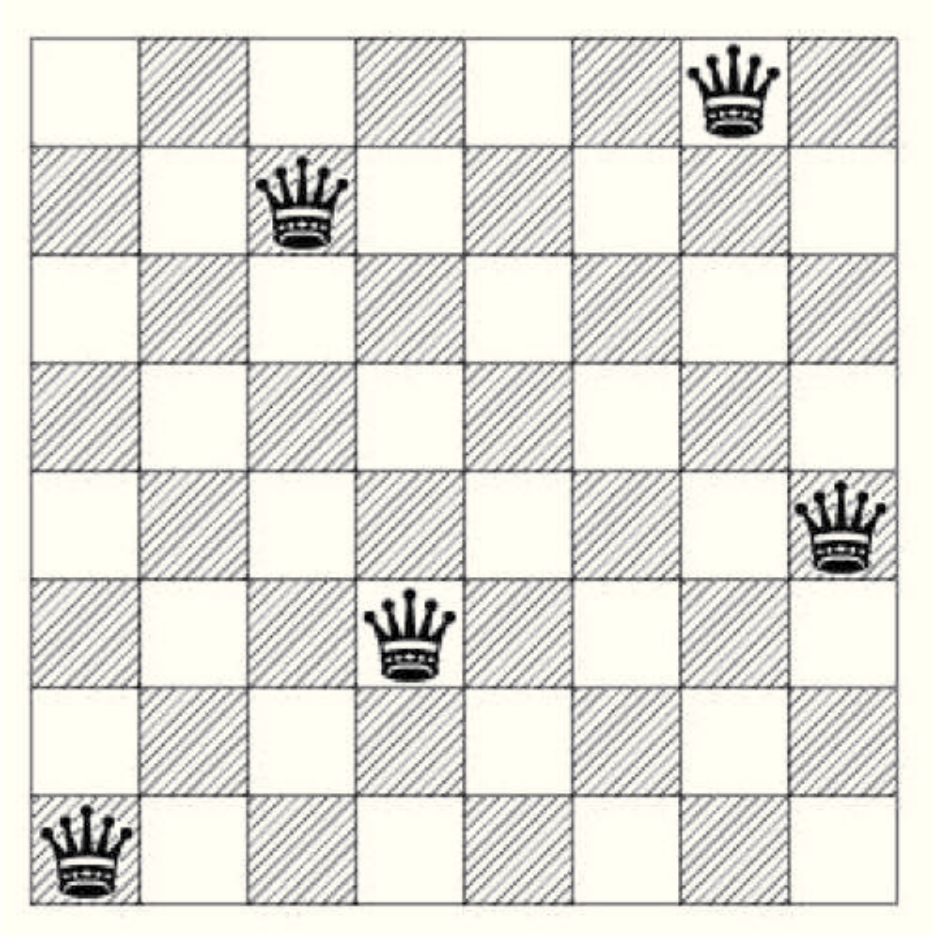


Figure 1.2: A possible solution of The Queens Domination Problem, or how to place a minimum number of queens on a chessboard so that each square is controlled by at least one queen.

Historically, the first domination-type problems came from the game of chess. In the XIX century, chess players were interested in many problems concerning the game as: the *Eight Queens Puzzle* (see Figure 1.1), its generalized n-Queens version, the *Queens Domination Problem* (see Figure 1.2), *Queens and Knights*, etc...

For example, the Eight Queens Puzzle is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens can attack each other. Recall that a queen can move any number of squares horizontally,

vertically, or diagonally on the chessboard, thus a solution requires that no two queens share the same row, column, or diagonal.

The Queens Domination Problem, instead, is to place on the chessboard the minimum number of queens such that every square on the chess board either contains a queen or is attacked by a queen. In 1862, the chess master C.F. de Jaenisch wrote a treatise [37] on the applications of mathematical analysis to chess in which he also considered the number of queens necessary to attack every square on a $n \times n$ chess board.

Apart from chess, Domination in graphs has applications in several other fields. Domination arises in facility location problems, where the number of facilities (e.g. police stations, hospitals, fire stations) is fixed and one attempts to minimize the distance that a person needs to travel to get to the closest facility. A similar problem occurs when the maximum distance to a facility is fixed and one attempts to minimize the number of facilities necessary so that everyone is served. Concepts from Domination also appear in problems involving finding sets of representatives, in monitoring communication or electrical networks, and in land surveying (e.g., minimizing the number of places a surveyor must stand in order to take height measurements for an entire region). Another interesting field is of application is social networking [54]: if we consider the vertices as people and two vertices are linked if there exists a friendship between the two people, then a dominating set is the set of people who knows everyone.

At present, Domination is considered to be one of the fundamental concepts in graph theory and its various applications to ad hoc networks, biological networks, distributed computing, social networks and web graphs partly explain the increased interest. Such applications usually aim to select a subset of vertices that will provide some definite service such that every vertex in the network is *close* to some vertex in the subset.

It is easy to see that, for a given graph $G(V, E)$ with $|V| = n$, the domination number can have a value in the following range: $1 \leq \gamma(G) \leq n$. In particular, $\gamma(G) = 1$ if and only if $\Delta(G) = n$, and the equality for the upper bound is true if and only if $\Delta(G) = 0$, where $\Delta(G)$ is the maximum degree of a graph. The question whether there exists a polynomial time algorithm for determining $\gamma(G)$ naturally arises. However, for arbitrary graphs, there is no algorithm that has better time complexity than exponential–time algorithms.

In fact, in [23] and [47] it is shown that Domination Problem is $\mathcal{NP}$-complete for arbitrary graphs. Thus, it is of importance to determine bounds for $\gamma(G)$ and various similar parameters.

In general, many domination parameters can be defined by combining Domination with another graph theoretical property.

In [29] this concept is formalized by imposing an additional constraint on the dominating set: for a given property P, the conditional domination number $\gamma(G : P)$ is the smallest cardinality of a dominating set $D \subseteq V$ such that the induced subgraph $G[D]$ satisfies property P. Thus, by considering different properties P, many new variants of Domination can be defined.

Also in [30] are listed more than 75 of such varieties. One of such variation is the Roman Domination Problem.

Like many Domination Problems on graphs, the Roman Domination Problem has several applications, for example either in military strategy [4], in which the authors point out the usefulness of the graph theory in studying optimization problems in the real world, and propose a comparative analysis between the Roman and the British Empire, in using a defense strategy devised by Constantine the Great (see 1.1), or for wireless server placements over networks, see [45] and [50].

Mathematically, given a graph, the Roman Domination Problem is to find a vertex labeling, such that every vertex must be labeled with values in

the set $\{0, 1, 2\}$, and every vertex labeled by 0 must have at least a neighbor labeled by 2.

The idea is that the labels 1 and 2 represent either one or two Roman armies stationed at a given location, say vertex v. A nearby location (an adjacent vertex u) is considered to be *unsecured* if no armies are stationed there, but become *securable* by sending an army from v, given the constraint that at least one army has to be left behind, i.e. $f(v) = 2$.

The Roman Domination Number of a graph ($\gamma_R(G)$) is the smallest value of the sums of labels, which satisfy the condition above.

1.1 Historical Basis for The Roman Domination Problem

The Roman Domination Problem can be defined and explained using an interesting historical anecdote about the Roman Empire's military strategies between the III and the IV century of the Common Era (CE), see [4], [49] and [53].

In the III century the Roman Empire dominated the majority of Europe, North Africa and Near East. To defend its borders and the empire itself, it was used a *forward defense strategy*, whereas fifty legions secured even the furthermost areas of the empire.

In the IV century, when the empire lost much of its power, and the number of its legions dropped drastically, Constantine the Great devised the *defense in depth strategy*, where local troops were used to disrupt invasions and rebellions, and mobile groups of six legions, called Field Armies, were used to stop them definetively. With this new strategy, a region is considered *secure* if it has one or more Field Armies stationed in it. On the other hand, a region may be *securable* if a Field Army may be able to deploy in a single

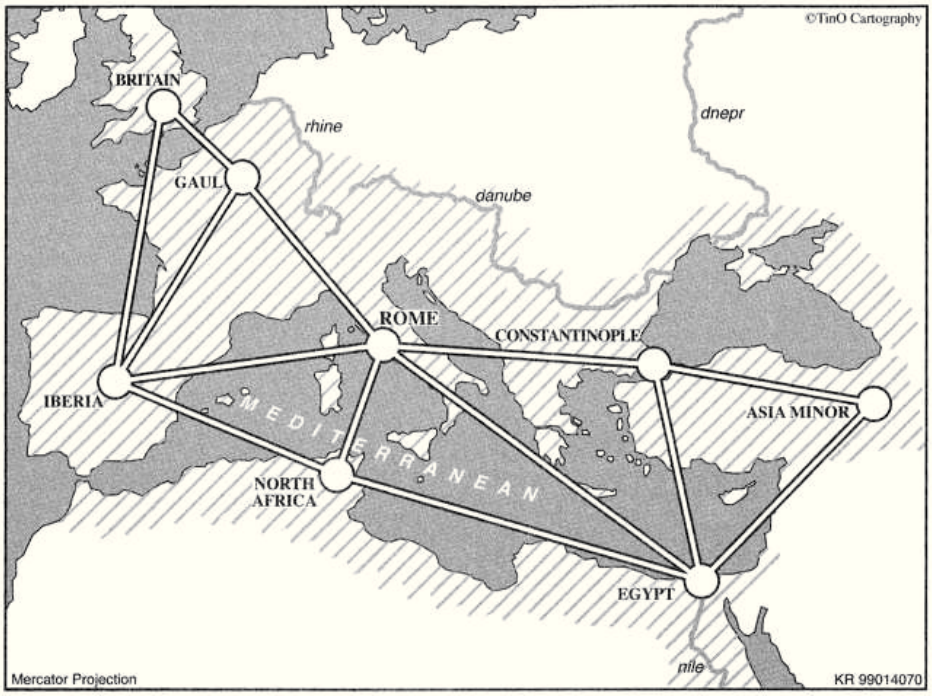


Figure 1.3: The Roman Empire at Constantinian Era.

step from an adjacent region and only if it moves from a region where there is at least an other Field Army.

At that time, the empire was connected as in Figure 1.3, where each region is represented as a circle and movements along a line between regions represent a step. While in the previous century, the there were eight Field Armies, one for each region of the empire, the challenge for Constantine was to allocate just four Field Armies to defend all the empire.

He chose to place 2 Field Armies in Rome and 2 Field Armies in Constantinople, at that time the two capitals of the empire. With this deployment all the regions were secure or securable, with the exception of Britain, in fact it is no coincidence that Britain was the first region of the Empire to be lost. In a modern analysis, there are other solutions that will secure all the empire. One solution is to deploy 1 Field Army in Britain, 2 Field Armies in Rome and 1 Field Army in Asia Minor; another solution is to deploy 2 Field Armies

in Gaul and 2 Field Armies in Egypt. These solutions, however, have the disadvantage to leave completely defenseless at least one of the capitals of the Roman Empire, a unthinkable solution for Constantine.

The same strategy was used at least two other times: the first time by the British Empire in the XIX century to defend its vast regions; in this case the regions are the oceans and seas of the world: they had four *Battle Fleets* and six sea regions to defend. The second time by the USA in the post Cold War Era: they had a similar situation, with four *Unit of Forces* and five regions, see [4].

2 Some Definitions over Graphs

We assume that the readers are familiar with many terms throughout this dissertation, but for sake of clarity we present here some definitions, and others will be introduced when necessary.

A *graph* $G(V, E)$ consists of a set of *vertices* V and a set E of unordered pairs of vertices called *edges*. For a graph $G(V, E)$, $n = |V|$ is the *order* of G. In a *directed graph* the set of edges is replaced with a ordered pairs of vertices called *arcs*. If (u, v) is an edge in a graph G, we say that u and v are *adjacent* in G, whereas if $[u, v]$ is an arc in a directed graph D, then we will say that v is adjacent to u. We will assume that all graphs are simple, i.e. no edge can appear in the edge set more than once and there is no edge between the same vertex.

In a graph $G(V, E)$, the *degree* of a vertex u, denoted with $\delta(u)$, is the number of vertices adjacent to u. The minimum and maximum degree of a graph $G(V, E)$ are denoted respectively with:

$$\delta(G) = \min\{\delta(u) \colon u \in V\}$$

and

$$\Delta(G) = \max\{\delta(u) \colon u \in V\} \ .$$

A vertex is *isolated* if its degree is zero, and a graph is *isolate-free* if it has no isolated vertices.

In a graph, a *path* of length k from vertex v_0 to vertex v_k is a collection of edges, denoted with $P = v_0 v_1 \dots v_{k-1} v_k$. A *cycle* is a closed path where $v_0 = v_k$. A graph is *connected* if there is a path from every vertex to every other vertex. In a connected graph, the *distance* between two vertices u and v, denoted as $d(u, v)$, is the shortest path joining u and v.

A graph $H(W, F)$ is a *subgraph* of the graph $G(V, E)$ if $W \subseteq V$ and $F \subseteq E$. A connected graph is a *tree* if it does not contain a cycle as a subgraph.

For any vertex $u \in V$, the *open neighborhood* of u is the set

$$N(u) = \{v \in V \colon (u, v) \in E\}$$

and the *closed neighborhood* is the set

$$N[u] = N(u) \cup \{u\} \ ,$$

thus for a graph $G(V, E)$, a set of vertices S is a dominating set of G if $N[S] = V$.

There are some classes of graphs that we will encounter in the next chapters:

- P_n denotes the graph consisting of a path of n vertices.
- C_n denotes the graph consisting of a cycle of n vertices.
- K_n denotes the complete graph of n vertices, where every vertex is adjacent to every other vertex.
- $\overline{K_n}$ denotes the graph of n isolated vertices.

- A graph is *bipartite* if the vertex set V can be partitioned into two disjoint subsets A and B such that vertices in A are only adjacent to vertices in B and vice versa.
- A *unit disk graph* is the intersection graph of a family of unit disks in the Euclidean plane, that is, we form a vertex for each disk, and connect two vertices by an edge whenever the corresponding disks have non-empty intersection.
- For graphs G and H, the *cartesian product* of G and H, denoted with $G \times H$ is the graph with vertex set $\{(u, v) \colon u \in V(G), v \in V(H)\}$. Two vertices (u_1, v_1) and (u_2, v_2) in $G \times H$ are adjacent if and only if one of the following conditions is true:
  - $u_1 = u_2$ and v_1 is adjacent to v_2 in H;
  - $v_1 = v_2$ and u_1 is adjacent to u_2 in G.
- The *join* of two graphs G and H, denoted with $G + H$, is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H) \cup \{(u, v) \colon u \in V(G), v \in V(H)\}$.

3 State of the Art

3.1 The First Studies

In [49], the authors connected the Roman Domination Problem to the class of Location Problems, i.e. the $0, 1$ Optimization Problems that seek *yes* or *no* location decisions at discrete points. So, using the integer programming, they define two problems:

- the *Set Covering Deployment Problem* (SCDP), a variation of the *Location Set Covering Problem*: given a graph $G(V, E)$, for each vertex

v_i are defined two $\{0,1\}$ variables x_i and y_i, representing the first and the second armies on the vertex, we have to find and place the least number of armies so that every vertex have at least one army within a minimum distance.

$$\text{MINIMIZE} \quad \sum_{i \in V} (x_i + y_i)$$

$$\text{with} \quad y_i \leq x_i \quad \forall i$$

$$\text{and} \quad x_i + \sum_{(v_i, v_j) \in E} y_j \geq 1 \quad \forall i \; .$$

Where x_i is 1 if the vertex i contains one or two armies and 0 otherwise, y_i is 1 if the vertex i contains two armies and 0 otherwise. The constraints guarantee that every vertex either has a army on it or has a neighbor with two armies on it.

- the *Maximal Covering Deployment Problem* (MCDP), the complementary of the problem above, is a variation of the *Maximal Covering Location Problem*. The objective in this problem is to maximize the number of regions secured by a fixed number of armies, p. Given a graph $G(V, E)$, for each vertex v_i are defined three $\{0,1\}$ variables x_i, y_i and U_i; x_i and y_i representing the first and the second armies on the vertex, while $U_i = 1$ if v_i has an army or it exists a neighbor with two armies on it, $U_i = 0$ otherwise.

$$\text{MAXIMIZE} \quad \sum_{i \in V} U_i$$

$$\text{with} \quad y_i \leq x_i \quad \forall i$$

$$\text{and} \quad U_i \leq x_i + \sum_{(v_i, v_j) \in E} y_j \quad \forall i$$

$$\text{and} \quad \sum_{i=1}^{n} (x_i + y_i) = p \; .$$

The techniques used by the authors to solve these problems are the *branch and bound* algorithms, i.e. a systematic enumeration of all candidate solutions, where large subsets of fruitless candidates are discarded *en masse*, by using upper and lower estimated bounds of the quantity being optimized.

3.2 The Modern Approach

In [11], the authors studied the Roman Domination Problem as a variation of the Domination Problem on graphs using a purely graph theoretical approach, and formalized for the first time the Roman Domination Problem as we know it.

We can imagine the regions of the Roman Empire as the vertices of a graph, whereas the links are the edges. Each vertex has a label with three possible values: $0, 1, 2$. If a vertex has label 0, then it has to be adjacent to at least one vertex with label 2.

Formally, given a graph $G = (V, E)$, a *Roman Dominating Function* (RDF) is a labeling function $f\colon V \to \{0, 1, 2\}$, satisfying the following:

$$\forall\, u \in V: \quad f(u) = 0 \qquad \Rightarrow \qquad \exists\, v \in V: \quad (u, v) \in E \ \wedge\ f(v) = 2\,.$$

The function f induces a partition of $V = (V_0, V_1, V_2)$, where

$$V_i = \{v \in V : f(v) = i\}\ .$$

If we define $n_i = |V_i|$ then we will have $n_0 + n_1 + n_2 = n = |V|\,.$

The weight of f is

$$f(V) = \sum_{u \in V} f(u) = 2 \cdot n_2 + 1 \cdot n_1 + 0 \cdot n_0 = 2 \cdot n_2 + n_1\ .$$

For each graph G, we define the *Roman Domination Number* $\gamma_R(G)$ as the minimum value of a Roman Dominating Function:

$$\gamma_R(G) = \min_{f \in \mathcal{F}}\ f(V) = \min_{f \in \mathcal{F}}\ \sum_{u \in V} f(u)$$

where $\mathcal{F}$ is the set of all Roman Dominating Functions for G.

In [11], among many others, were proven the following facts about the Roman Dominating Function and $\gamma_R(G)$ and $\gamma(G)$, i.e. the Domination Number of G:

1. For any graph $G(V, E)$:

$$\gamma(G), \gamma_R(G) \leq n \ .$$

2. For any graph $G(V, E)$:

$$\gamma(G) \leq \gamma_R(G) \leq 2 \cdot \gamma(G) \ .$$

3. A graph $G(V, E)$ is said to be *Roman* if

$$\gamma_R(G) = 2 \cdot \gamma(G) \ .$$

   This imply that there exists a minimal Roman Dominating Function such that:

$$n_1 = |V_1| = 0 \ .$$

4. For any graph $G(V, E)$:

$$\gamma(G) = \gamma_R(G) = 1 \iff G = K_n \ ,$$

   i.e. G is a complete graph.

5. For any graph $G(V, E)$:

$$\gamma(G) = \gamma_R(G) = n \iff G = \overline{K_n} \ ,$$

   i.e. G is an edgeless graph.

6. For any graph $G(V, E)$, such that $G[V_1]$ is the subgraph induced by V_1:

$$\Delta(G[V_1]) = 1 \ .$$

7. For any graph $G(V, E)$, $u \in V_1$, $v \in V_2$:

$$(u, v) \notin E \ .$$

8. For any graph $G(V, E)$ with no isolated vertices, such that n_1 is minimum:

$$n_0 \geq 3 \cdot n/7 \ .$$

9. For any graph $G(V, E) \neq \overline{K_n}$:

$$\gamma_R(G) \geq \left\lceil \frac{2 \cdot n}{\Delta(G) + 1} \right\rceil$$

where $\Delta(G)$ is the *maximum degree* of G. The proof is in Chapter 3, see also [12].

10. Given a graph $G(V, E)$, by using a probabilistic method due to [3], we have the following upper bound on γ_R:

$$\gamma_R(G) \leq n \cdot \frac{2 + \ln\left(\frac{(1 + \delta(G))}{2}\right)}{1 + \delta(G)}$$

where $\delta(G)$ is the *minimum degree* of G.[1]

In [48], using again a probabilistic method, it is proved an improved upper-bound for the Roman Domination Number:

$$\gamma_R(G) \leq 2\ n \cdot \left(1 - \frac{2^{1/\delta(G)} \cdot \delta(G)}{(1 + \delta(G))^{1+1/\delta(G)}}\right)$$

and, using the inequality $1 - p \leq e^{-p}$:

$$\gamma_R(G) \leq n \cdot \frac{2\ln(1 + \delta(G)) - \ln 4 + 2}{1 + \delta(G)}$$

Moreover, the author proved that this upper-bound is asymptotically best possible, i.e.:

[1] Note that this upper-bound contains a misprint in the original paper. The exact value is proved below.

Theorem 1. *When n is large, there exists a graph $G(V, E)$ such that*

$$\gamma_R(G) \geq n \cdot \frac{2\ln(1+\delta(G)) - \ln 4 + 2}{1+\delta(G)} \cdot (1 + o(1))$$

In [8] the authors proved two *upper-bounds* for any connected graph G:

1. $\gamma_R \leq 4n/5$. Actually, to be true n has to be greater or equal to 3, in fact if $n = 2$, $\gamma_R = 2 > 4 \cdot n/5 = 8/5$.
2. if $\delta(G) \geq 2 \ \wedge \ n \geq 9$ then $\gamma_R \leq 8n/11$.

Still for any connected graph G with $n \geq 3$, in [21] it was proven that:

1. $|V_0| \geq n/5 + 1$;
2. $|V_1| \leq 4n/5 - 2$;
3. $|V_2| \leq 2n/5$.

In [40] the authors showed some algorithms to compute γ_R in linear time for some classes of graphs, such as *interval graphs* and *co-graphs*. Additionally they gave a polynomial algorithm for *AT-free* graph.

In [19] the author gives a polynomial-time algorithm for computing a minimal Roman Dominating Function for *trees*; this implies that we can compute a minimal Roman Dominating Function for acyclic graphs, because an acyclic graph G can be viewed as a *forest of trees* $\cup T_i$, where $T_1, \ldots, T_n$ are the single trees, and so the Roman Domination Number of G is the sum of the single Roman Domination Numbers of T_i, i.e. $\gamma_R(G) = \sum \gamma_R(T_i)$.

In [45] the Roman Domination is applied to solve a network location problem, while in [50] is applied to *unit disk graph*, a mathematical model for wireless networking, and it is found an approximation algorithm in linear time.

4 Complexity

We will show now that the decision problem corresponding to computing $\gamma_R(G)$ is $\mathcal{NP}$-complete, see [19].

The (decisional) Roman Domination Problem can be stated as follows:

Problem 2 (dRDP)**.** *Given a graph $G = (V, E)$ and an integer $\varrho \leq |V|$, is there a Roman Dominating Function $f = (V0, V1, V2)$ such that*

$$\gamma_R(G) \leq \varrho \ ?$$

To show that this problem is $\mathcal{NP}$-complete, we construct a polynomial transformation from the *3-Satisfiability Problem* (3–SAT).

Problem 3 (3–SAT)**.** *Given a collection $C = \{C_1, C_2, \ldots, C_m\}$ of clauses on a finite set $X = \{X_1, X_2, \ldots, X_n\}$ of variables, is there a truth assignment for X such that all the clauses in C are satisfied ?*

A literal is either a variable $X_j \in X$ or its complement $\overline{X_j}$. A clause C_i, formed by three *literals*, is satisfied by a truth assignment, a function $A\colon X \to \{TRUE, FALSE\}$, if one of the three literals is set to TRUE.

Let I be an instance of 3–SAT, as defined above. We produce a graph $G(I)$ and a positive integer ϱ such that I as a satisfying truth assignment if and only if $G(I)$ has a Roman Dominating Function $f = (V0, V1, V2)$ with $f(V) \leq \varrho = 2n$.

Corresponding to each clause C_i, we construct a vertex labeled c_i. Corresponding to each variable $X_i \in X$ we construct a copy of the complete bipartite graph $K_{2,3}$, with partite sets $R_i = \{x_i, \overline{x_i}\}$ and $S_i = \{u_i, v_i, w_i\}$. To each $K_{2,3}$ we add the edge $(x_i, \overline{x_i})$, and we add three edges from each clause vertex c_i to the three vertices corresponding to the three literals in clause C_i. An example of the construction of $G(I)$ is given in Figure 1.4. We claim that

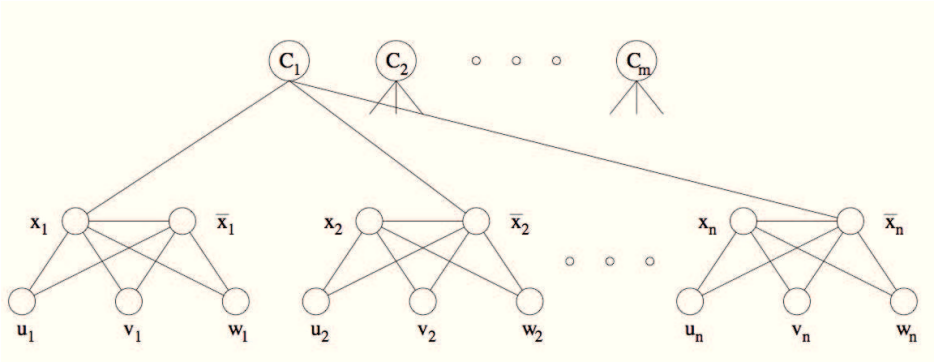


Figure 1.4: Constructing a reduction from 3–SAT, $C_1 = X_1\overline{X_2X_n}$.

the given instance of 3–SAT has a satisfying truth assignment if and only if the graph $G(I)$ has a Roman Dominating Function f with $f(V) \leq 2n$.

Let the given instance of 3–SAT have a satisfying truth assignment A. For each variable X_i, if $A(X_i) = TRUE$, then let $f(x_i) = 2$. Otherwise, let $f(\overline{x_i}) = 2$. For all other vertices w in $G(I)$, let $f(w) = 0$. It is easy to see that the function f so defined is a Roman Dominating Function of weight $\varrho \leq 2n$, since either x_i or $\overline{x_i}$ is in V_2, and either vertex defends $R_i \cup S_i$. Also, every vertex c_i has at least one neighbor in V_2 because A assigns at least one literal to TRUE in every clause C_i.

Conversely, assume that $G(I)$ has a Roman Dominating Function f of weight $f(V) = \varrho \leq 2n$. It is easy to see that for each induced subgraph on $R_i \cup S_i$, we must have either $f(x_i) = 2$ or $f(\overline{x_i}) = 2$. Therefore, f has weight at least $2n$. It follows that the set of vertices v for which $f(v) = 2$ defends the set of clause vertices. Therefore, the given instance of 3–SAT has a satisfying truth assignment A, where $A(X_i) = TRUE$ if and only if $f(x_i) = 2$.

Using similar constructions, it can be proved that the (decisional) Roman Domination Problem is $\mathcal{NP}$-complete even when restricted to bipartite or planar graphs.

Because both the Domination Problem, as showed in [30], and Roman

Domination Problem, as showed above, are $\mathcal{NP}$-complete, it is interesting to study the properties of γ and γ_R for specific classes of graphs. (See [1], [9], [26], [11], [**?**], [32], [52], [46]).

4.1 A Non-Deterministic Algorithm

Algorithm 1 A non-deterministic Algorithm for the RDP.

ND-RDF $(G = (V, E))$
$\gamma = 0$
for $v \in V$ **do**
  if Guess-label$(v, 2) == TRUE$ **then**
    label$(v) = 2$ {we guess v must be labeled 2}
    $\gamma+ = 2$
  else
    label$(v) = -1$ {-1 stands for unlabeled}
  end if
end for
for $v \in V$ **do**
  if label $(v) == -1$ **then**
    if $(\exists u \in V : (u, v) \in E \wedge$ label $(v) = 2)$ **then**
      label$(v) = 0$
    else
      label$(v) = 1$
      $\gamma+ = 1$
    end if
  end if
end for
return γ, G

We can utilize the non-deterministic polynomial time Algorithm 1 for computing γ_R. Such algorithm can, obviously, be used to prove that the Roman Domination Problem is in $\mathcal{NP}$.

From the definition of the Roman Domination Problem, it is clear that once we decide which vertices have to be labeled with 2, then we can label all other vertices: with 0 the unlabeled vertices connected to at least one vertex labeled by 2, and with 1 all the remaining vertices. This will give us Roman Dominating Functions that will have a special property:

Definition 4 (weak Property of non–Redundancy)**.** *Given a Roman Dominating Function f over a graph $G(V, E)$, we said that the function f is weakly non–redundant if:*

- *Every vertex v labeled by* 1 *has no vertices labeled by* 2 *in its adjacency list, see the Propositions 6 and 7 above.*

There exists a strong version of the property, that we will use in the next chapters:

Definition 5 (strong Property of non–Redundancy)**.** *Given a Roman Dominating Function f over a graph $G(V, E)$, we said that the function f is strongly non–redundant if:*

- *Every vertex v labeled by* 1 *has no vertices labeled by* 2 *in its adjacency list, see the Propositions 6 and 7 above.*
- *Every vertex v labeled by* 0 *is covered by only one vertex labeled by* 2*.*

The idea of finding the best vertices to be labeled with 2 can be used in developing approximation algorithms. In such cases, one needs to find effective heuristics to do so. However, for a special class of graphs, described in the next chapter, we can decide in a precise manner which vertices are to be labeled with 2, obtaining a polynomial time algorithm which gives excellent solutions, that we believe are the optimal ones.

Chapter 2

Grid Graphs

1 Introduction

The following contents and that of Chapter 3 and Chapter 4 are object of the paper *The Roman Domination Problem on Grid Graphs*, presented at the *Middle-European Conference on Applied Theoretical Computer Science* (MATCOS-13), [14].

As we already said in the introductory chapter, the grid graphs are a class of graphs that, when drawn on an Euclidean Space, form a regular tiling. We will always use the *square grid graphs*, whose vertices corresponds to the points of the Euclidean Plane with integer coordinates and two vertices are connected by an edge if and only if the corresponding points are at distance 1. We will always refer to a finite section of the infinite graph, like *the grid graph of m rows and n columns*, as x-coordinates being in the range $0, \ldots, n-1$, y-coordinates being in the range $0, \ldots, m-1$.

In alternative, we can define a grid graph as a *Cartesian product* of two path graphs with $n-1$ and $m-1$ edges, see Figure 2.1.

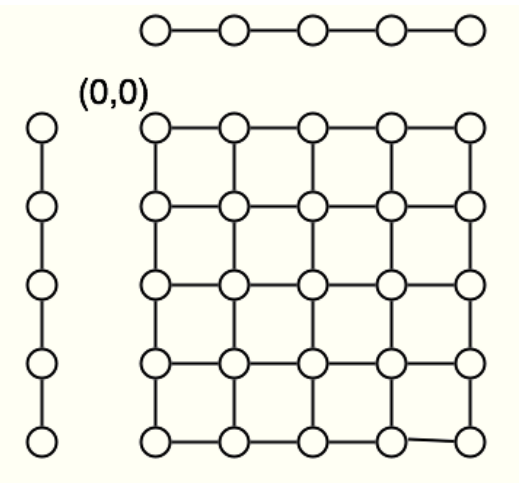


Figure 2.1: The Grid Graph $G_{5,5}$ as cartesian product of two path graphs P_5.

We will denote a grid graph with m rows and n columns by $G_{m,n}$. Also we will call a *tile* of dimension n, and we will denote it by T_n, any grid graph of type $G_{n,n}$.

Each vertex of the grid graph is characterized by its *coordinates* (i, j) and is connected to (at most) 4 vertices, namely, the vertices of coordinates $(i, j-1)$, $(i, j+1)$, $(i-1, j)$, $(i+1, j)$, if they exist. Obviously vertices on the grid graph's corners are connected to only 2 other vertices, while vertices on the borders (but not on the corners) are connected to 3 vertices.

For sake of clarity, we specify that the coordinates of the top-left corner of the grid are $(0, 0)$ and the coordinates of the bottom-right corner of the grid graph $G_{m,n}$ are $(m-1, n-1)$, see again Figure 2.1.

1.1 Roman Dominating Function for some type of Grid Graphs

In [11] was given a characterization of the Roman Domination Problem for grid graphs $G_{1,n}$, i.e. the path graphs P_n, see Figure 2.2, and grid graphs

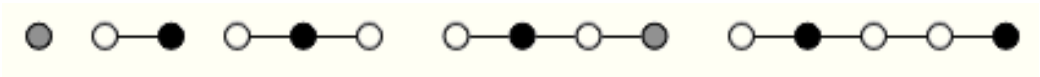


Figure 2.2: The coloring of the path graphs P_n, for $1 \leq n \leq 5$. In black the vertices labeled by 2, while in gray the vertices labeled by 1 and in white the vertices labeled by 0.

Figure 2.3: The coloring of the grid graphs $G_{2,n}$, for $1 \leq n \leq 5$.

$G_{2,n}$, see Figure 2.3, leading to the following equations:

$$\gamma_R(G_{1,n}) = \left\lceil \frac{2n}{3} \right\rceil ;$$

$$\gamma_R(G_{2,n}) = n + 1 \ .$$

They also left an open problem, i.e. to give a characterization of the RDP for grid graphs $G_{m,n}$, for $m, n \geq 3$.

This open problem was addressed by Dreyer in [19], Cockayne *et al.* in [12] and, quite recently, Pavlic and Zerovnik in [46].

In particular, in [19] the author proposes a covering method for grid graphs of type $G_{3,n}$, see Figure 2.4, and one for grid graphs of type $G_{4,n}$, see Figure 2.5. The first method leads to:

$$\gamma_R(G_{3,n}) = \begin{cases} 6k+1 & n = 4 \cdot k \\ 6k+2 & n = 4 \cdot k + 1 \\ 6k+4 & n = 4 \cdot k + 2 \\ 6k+6 & n = 4 \cdot k + 3 \end{cases} . \tag{2.1}$$

For this type of grid graphs, indeed, we independently derived a more simple equation:

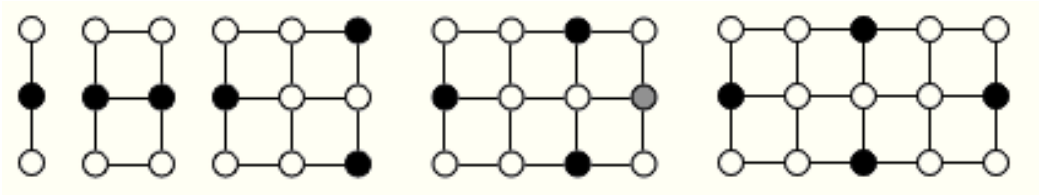


Figure 2.4: The coloring of the grid graphs $G_{3,n}$, for $1 \leq n \leq 5$.

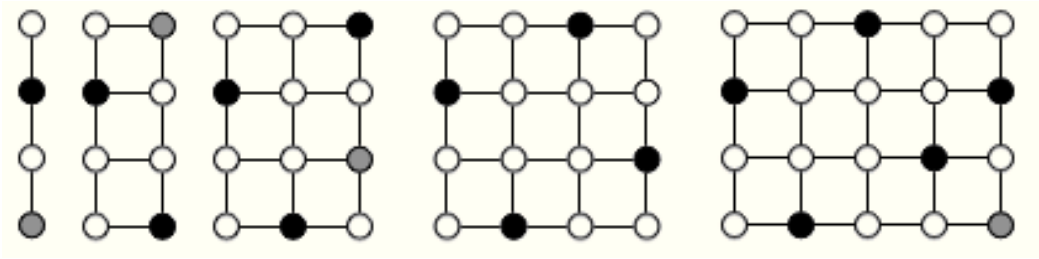


Figure 2.5: The coloring of the grid graphs $G_{4,n}$, for $1 \leq n \leq 5$.

$$\gamma_R(G_{3,n}) = \begin{cases} \left\lfloor \frac{3n}{2} \right\rfloor + 1 & n = 4 \cdot k + 1 \\ \left\lceil \frac{3n}{2} \right\rceil + 1 & \text{otherwise} \end{cases} . \tag{2.2}$$

The second method leads to:

$$\gamma_R(G_{4,n}) = \begin{cases} 2n+1 & n = 1, 2, 3, 5, 6 \\ 2n & \text{otherwise} \end{cases} . \tag{2.3}$$

In [46] the authors showed covering methods for grid graphs of type $G_{k,n}$ with $5 \leq k \leq 8$, see Figures 2.6, 2.7, 2.8, 2.9, which lead to the following equations:

$$\gamma_R(G_{5,n}) = \begin{cases} 8 & n = 3 \\ \left\lfloor \frac{12n}{5} \right\rfloor + 2 & \text{otherwise} \end{cases} ; \tag{2.4}$$

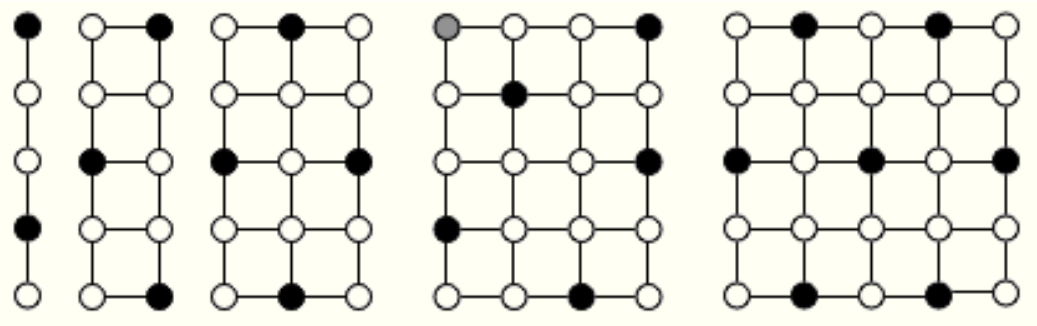


Figure 2.6: The coloring of the grid graphs $G_{5,n}$, for $1 \leq n \leq 5$.

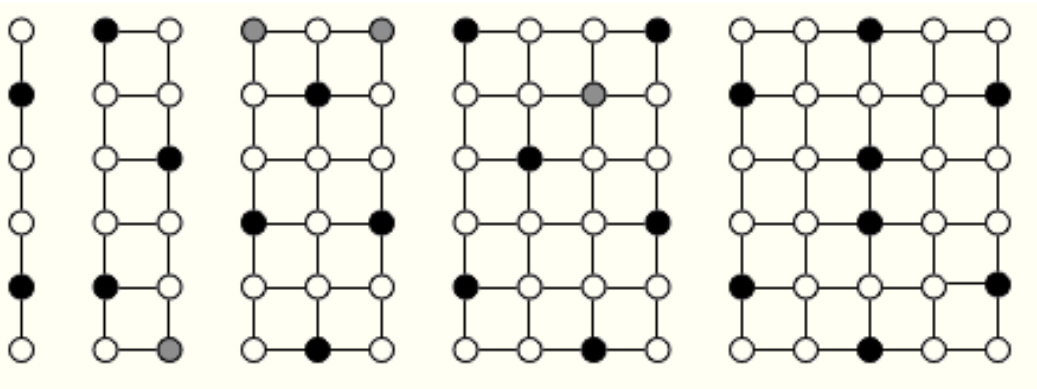


Figure 2.7: The coloring of the grid graphs $G_{6,n}$, for $1 \leq n \leq 5$.

$$\gamma_R(G_{6,n}) = \begin{cases} \left\lfloor \frac{14n}{5} \right\rfloor + 2 & n < 5 \quad | \quad n \in \{5k, 5k+3, 5k+4 \colon k \in \mathbb{N}\} \\ \left\lfloor \frac{14n}{5} \right\rfloor + 3 & \text{otherwise} \end{cases} ; \tag{2.5}$$

$$\gamma_R(G_{7,n}) = \begin{cases} \left\lfloor \frac{16n}{5} \right\rfloor + 2 & n \in \{1, 2, 4, 7, 5k \colon k \in \mathbb{N}\} \\ \left\lfloor \frac{16n}{5} \right\rfloor + 3 & \text{otherwise} \end{cases} ; \tag{2.6}$$

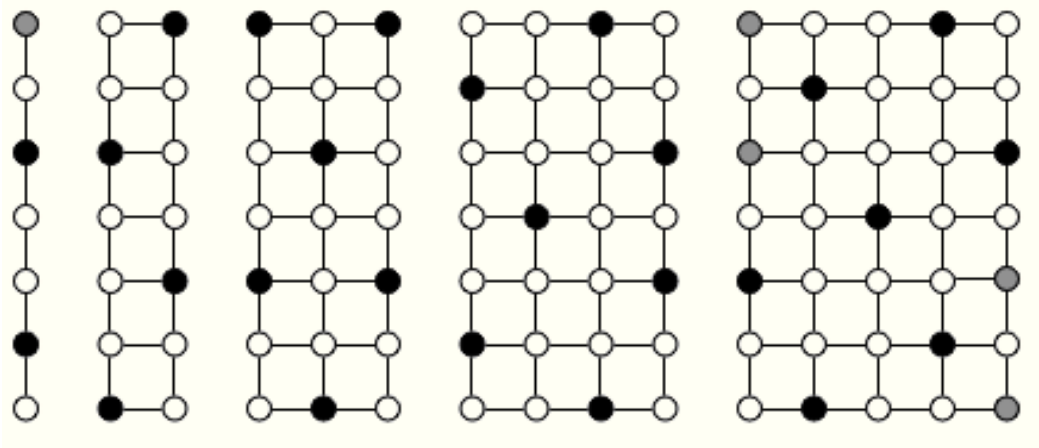


Figure 2.8: The coloring of the grid graphs $G_{7,n}$, for $1 \leq n \leq 5$.

$$\gamma_R(G_{8,n}) = \begin{cases} 9 & 2 \\ 16 & 4 \\ \left\lfloor \frac{18n}{5} \right\rfloor + 4 & n \in \{5k+3 \colon k \in \mathbb{N}\} \\ \left\lfloor \frac{18n}{5} \right\rfloor + 3 & \text{otherwise} \end{cases}. \tag{2.7}$$

In [12] the authors proposed an upper-bound on the grid graphs $G_{m,n}$, with $m, n \geq 5$, see Theorem 6 in the section below.

However, the problem of finding a unique formula and covering method for a generic grid graph was still open.

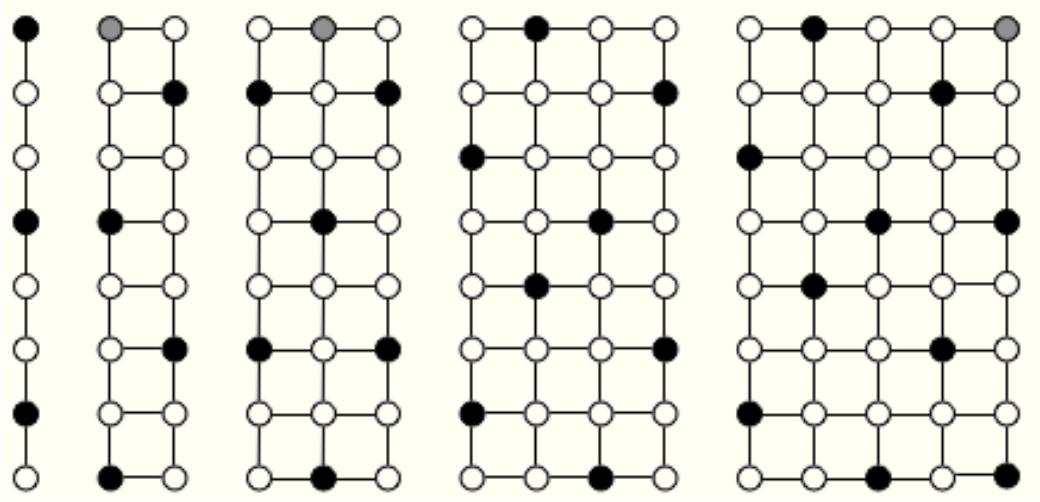


Figure 2.9: The coloring of the grid graphs $G_{8,n}$, for $1 \leq n \leq 5$.

2 Covering Schemas for generic Grid Graphs

In what follows, we will describe a covering method that lead to five different schemas for finding Roman Dominating Functions for grid graphs that are good, if not optimal. The method works as follows:

1. Choose the vertices of the grid graph where to place the label 2.
2. Place the label 0 on all the vertices connected to at least a vertex labeled by 2.
3. Place the label 1 on the remaining vertices.

Basically we use the Algorithm 1, only this time we know which vertices have to be labeled by 2. The underlying assumption is clear: once you decide where to place the labels 2, the labels of the remaining vertices are automatically determined (as in Step 2 and Step 3 above) to guarantee a Roman Dominating Function enjoying the weak Property of non–Redundancy, see Property 4.

2.1 The New Method

After some try-outs, we noticed that if we labeled a inner vertex by 2, we could use a regular pattern to place the others labels 2 that will guarantee a Roman Dominating Function enjoying the strong Property of non–Redundancy, see Property 5. Such pattern is similar to a move typical of the game of chess: the *Knight Movement*; a knight can move in two way:

- Clockwise, i.e. it makes a movement of a square and then a movement of two squares.
- Counter-clockwise, i.e. it makes a movement of two squares and then a movement of a square.

In Figure 2.10 we show how the pattern works:

1. Initially we label with 2 a vertex, in black in the figure, and with 0 its adjacent vertices.
2. In the second step, using the clockwise Knight Movement, we select the four red vertices, label them with 2 and then their adjacent vertices with 0.
3. In the third step, again using the clockwise Knight Movement and as starting point the red vertices, we select the eight blue vertices, label them with 2 and their adjacent vertices with 0.
4. And so on, until there are no more vertices reachable by a clockwise knight movement.

2.2 The Five Schemas

We noticed that, using this method, we will obtain a very regular Roman Dominating Function, and based on which vertices is chosen as first, this

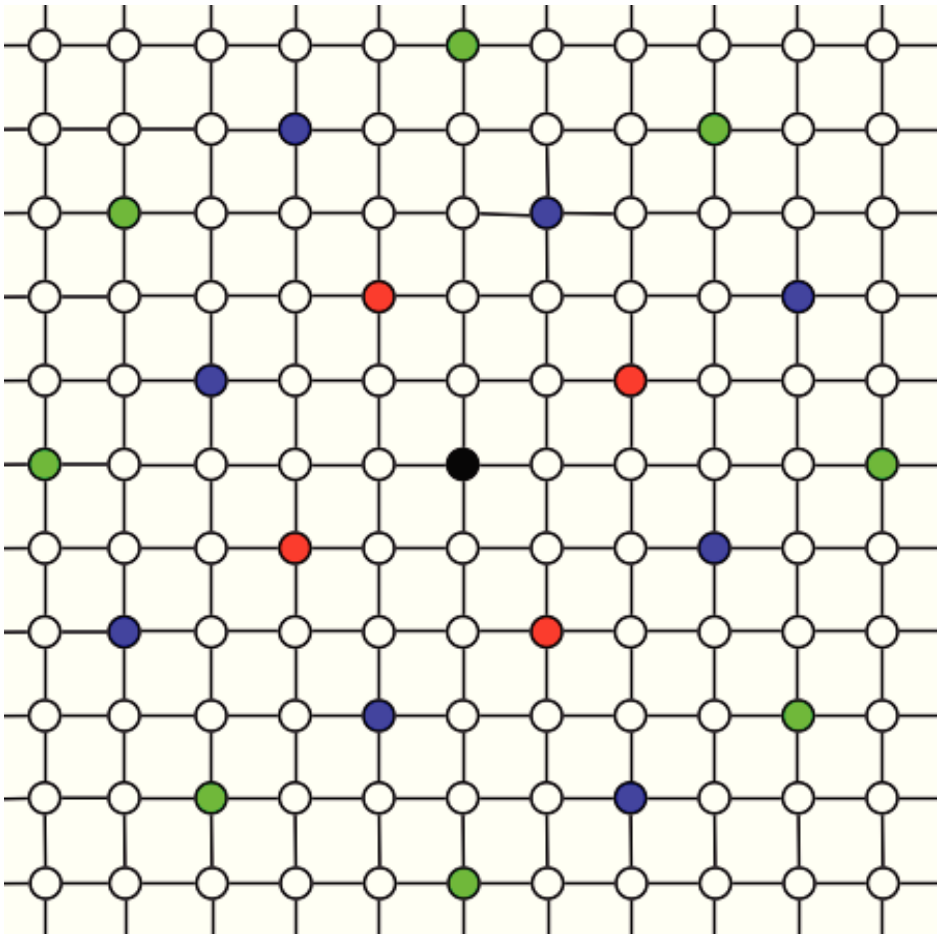
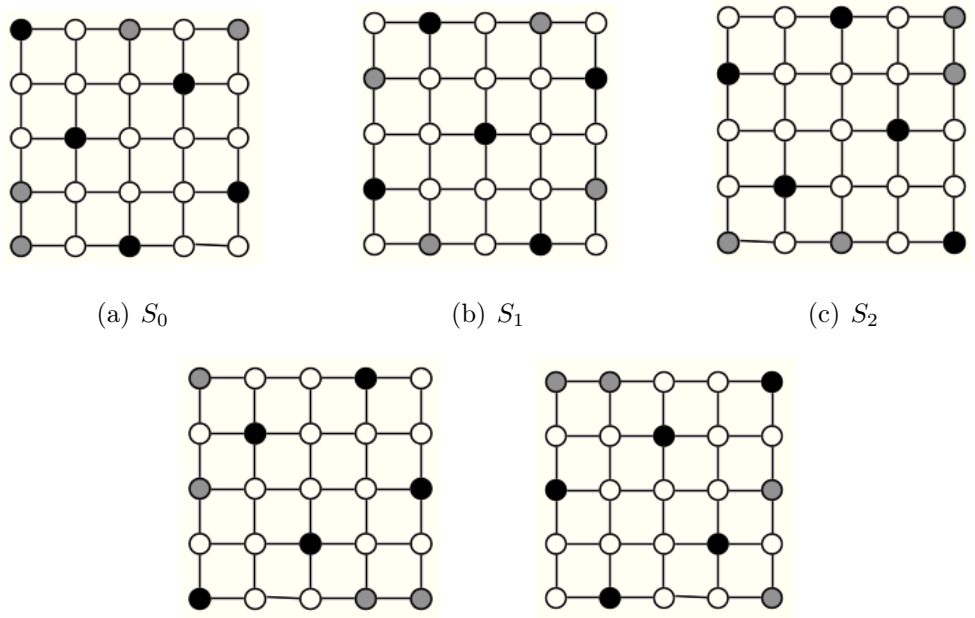


Figure 2.10: The Knight Movement: in black the initial position, in red the first four positions that are compatible with a optimal covering, and so on, we have the blue positions and after the green ones.

method will lead to five initial points, i.e. we obtain five schemas, which we will denote as S_i for $i = 0, \ldots, 4$. The initial vertex of the schema S_i is the vertex $(0, i)$.

To describe the five schemas, we will make use of the tile T_5, see Figures 2.11(a)-(e), because we are interested only in grid graphs with $m, n \geq 5$.

When applied to T_5, S_0 places the label 2 on the vertices of coordinates $(0, 0)$, $(1, 3)$, $(2, 1)$, $(3, 4)$, $(4, 2)$. As we said earlier, such pattern is then repeated on grid graphs with a larger number of rows and/or columns. To



(a) S_0 (b) S_1 (c) S_2

(d) S_3 (e) S_4

Figure 2.11: The 5 Covering Schemas. In black there are the vertices labeled by 2, in grey by 1 and in white by 0. The schemas only give the positions of the vertices labeled by 2 (vertices are then labeled by 1 whenever this is necessary).

obtain the schema S_i from S_0, we shift in every row the labels 2 by i, for $i = 1, 2, 3, 4$.

Table 2.1 tells us where the labels 2 are placed according to the five different schemas.

The first schema, S_0, (see Figure 2.11(a) for a visual description of it on a T_5), was introduced in [12], where for grid graphs of type $G_{m,n}$, with $m, n \geq 5$, they proved the following result:

Theorem 6. *Given a grid graph $G_{m,n}$, with $m, n \geq 5$, we have*

$$\gamma_R(G_{m,n}) \leq 2 \cdot \left(\left\lceil \frac{m \cdot n}{5} \right\rceil + \left\lceil \frac{m}{5} \right\rceil + \left\lceil \frac{n}{5} \right\rceil \right) .$$

Proof. Clearly, following the Schema S_0, $|V_2| \leq \left\lceil \frac{m \cdot n}{5} \right\rceil$. Each vertex of

	S_0	S_1	S_2	S_3	S_4
$i \mod 5$	$j \mod 5$	$j \mod 5$	$j \mod 5$	$j \mod 5$	$j \mod 5$
0	0	1	2	3	4
1	3	4	0	1	2
2	1	2	3	4	0
3	4	0	1	2	3
4	2	3	4	0	1

Table 2.1: Rules for placing labels 2 on the grid graph.

$V \setminus V_2$ which is not on the borders is adjacent to a vertex in V_2 and thus labeled by 0, see Thoerem 7. On the borders, some vertices are in $N[V_2]$, labeling them with 0, while some others have to be placed in V_1, labeling them with 1. Note that on the borders at most one every five vertices has to be labeled by 1, so $|V_1| \leq 2 \cdot \left\lceil \frac{m}{5} \right\rceil + 2 \cdot \left\lceil \frac{n}{5} \right\rceil$.

Thus we can say that:

$$\gamma_R(G_{m,n}) = 2 \cdot |V_2| + |V_1| \leq 2 \cdot \left\lceil \frac{m \cdot n}{5} \right\rceil + 2 \cdot \left\lceil \frac{m}{5} \right\rceil + 2 \cdot \left\lceil \frac{n}{5} \right\rceil$$

□

In general, every five vertices, one vertex is labeled by 2 and four vertices by 0, with the exception of the borders, where every five vertices, at most one vertex is labeled by 1. Such coverings guarantees that none of the internal vertices of the grid will be labeled by 1 (see Theorem 7).

Theorem 7. *Let $G_{m,n}$ be a grid graph. If a Roman Dominating Function f of $G_{m,n}$ is obtained by using one of the schemas $S_0, \cdots, S_4$, then it enjoys the strong Property of non–Redundancy, i.e. every internal vertex of G using f is either labeled by 2 or it is connected to one and only one vertex labeled*

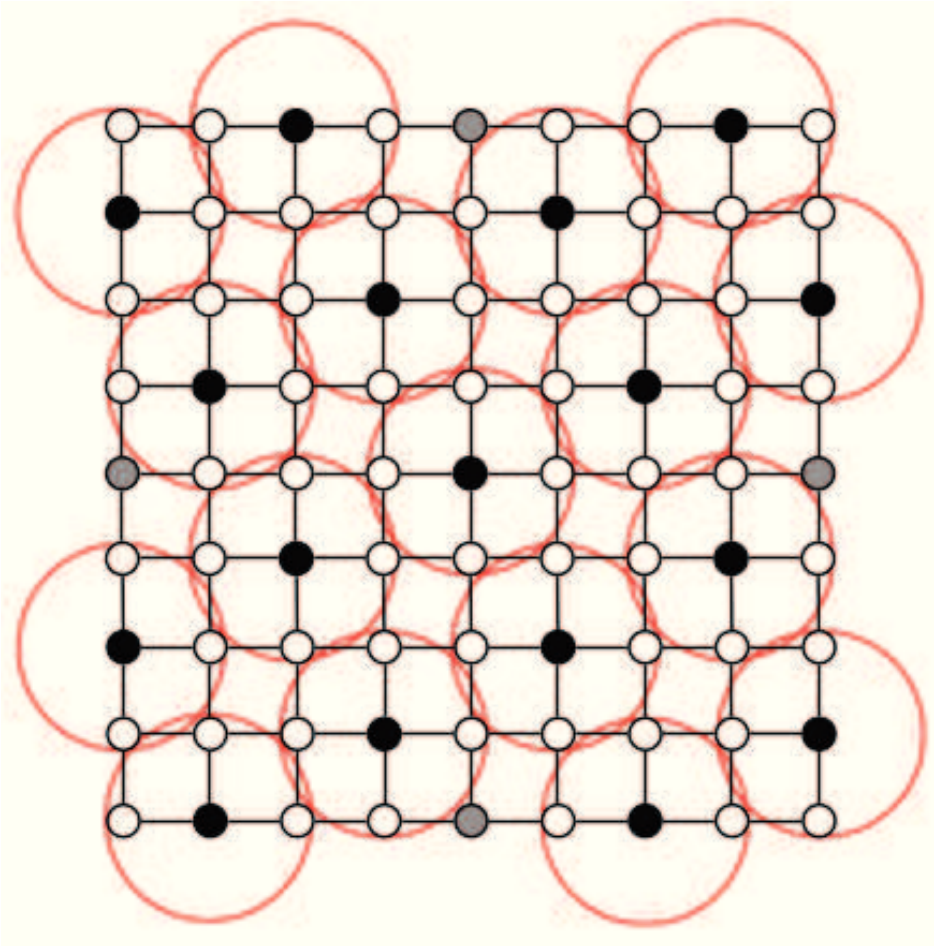


Figure 2.12: Covering of the Tile T_9 by the schema S_2. The circles define the influence of the vertices labeled by 2.

by 2 *and thus labeled by* 0 *and every vertex labeled by* 1 *has no vertices labeled by* 2 *in its adjacency list.*

Proof. The theorem clearly holds by construction. □

We studied the other four covering schemas, shown in Figure 2.11(b)-(e), because we noticed that they generally give better results than the schema S_0, see Table 2.2.

For example, if we consider the schema S_2 and the tile T_9, we would have the covering shown in Figure 2.12. In this case, only one vertices for each

border has to be labeled by 1.

In what follows, given a grid graph $G_{m,n}$, we will denote by $\gamma_i(G_{m,n})$ the Roman Domination value obtained by schema S_i.

3 Remarks

At this point, we had a lower-bound, see the subsection 3.2 of Chapter 1, and an upper-bound, see Theorem 6; our aim was to find the exact Roman Domination Number, and in doing so we improved those bounds.

In Chapter 3 we will prove a better lower-bound for grid graphs; while in Chapter 4 we will prove a sharper upper-bound.

Grid Graph	γ_0	γ_1	γ_2	γ_3	γ_4
$G_{5,5}$	**14**	**14**	**14**	**14**	**14**
$G_{5,6}$	17	**16**	**16**	17	**16**
$G_{5,7}$	19	19	**18**	19	19
$G_{5,8}$	22	**21**	**21**	**21**	**21**
$G_{5,9}$	24	24	**23**	24	**23**
$G_{6,5}$	17	**16**	**16**	**16**	17
$G_{6,6}$	20	**19**	**19**	**19**	**19**
$G_{6,7}$	**22**	**22**	**22**	**22**	**22**
$G_{6,8}$	25	**24**	25	25	25
$G_{6,9}$	28	**27**	**27**	28	28
$G_{7,5}$	19	19	19	**18**	19
$G_{7,6}$	**22**	**22**	**22**	**22**	**22**
$G_{7,7}$	**25**	**25**	**25**	**25**	26
$G_{7,8}$	29	**28**	**28**	**28**	29
$G_{7,9}$	32	32	**31**	**31**	32
$G_{8,5}$	**21**	**21**	**21**	**21**	22
$G_{8,6}$	25	25	**24**	25	25
$G_{8,7}$	**28**	29	**28**	**28**	29
$G_{8,8}$	**32**	**32**	**32**	**32**	**32**
$G_{8,9}$	**35**	36	**35**	36	36
$G_{9,5}$	24	24	**23**	**23**	24
$G_{9,6}$	28	28	**27**	28	**27**
$G_{9,7}$	**31**	32	**31**	32	32
$G_{9,8}$	36	**35**	**35**	36	36
$G_{9,9}$	40	40	**38**	40	40

Table 2.2: A Comparison between the five Schemas; the minimal values found by the schemas is emphasized in bold characters.

Chapter 3

Improving the bounds: The Lower-Bound

1 Introduction

In the subsection 3.2 of Chapter 1, we remarked that for any graph $G \neq \overline{K_n}$ with $|V| = n$, we have the following lower-bound:

$$\gamma_R(G) \geq \frac{2 \cdot n}{\Delta(G) + 1} \ .$$

We chose to use the *ceiling function* in the disequation because the Roman Domination Number is always an integer:

$$\gamma_R(G) \geq \left\lceil \frac{2 \cdot n}{\Delta(G) + 1} \right\rceil \ .$$

We give here a short proof of this.

Proof. Let $G(V, E) \neq \overline{K_n}$ be a graph, with maximum degree $\Delta(G)$, and $f = (V_0, V_1, V_2)$ be a minimal Roman Dominating Function of G, with weight

$\gamma_R(G)$. Since each vertex $v \in V_0$ is adjacent to at least a vertex in V_2, we can say that:

$$|V_0| \leq \Delta(G) \cdot |V_2| \,. \tag{3.1}$$

Thus:

$$(\Delta(G)+1) \cdot \gamma_R(G) = (\Delta(G)+1) \cdot |V_1| + (\Delta(G)+1) \cdot 2\,|V_2|$$

$$(\Delta(G)+1) \cdot \gamma_R(G) = (\Delta(G)+1) \cdot |V_1| + 2\Delta(G) \cdot |V_2| + 2\,|V_2| \geq$$

$$\geq (\Delta(G)+1) \cdot |V_1| + 2\,|V_0| + 2\,|V_2| \qquad \text{by Equation 3.1.}$$

And, because $G \neq \overline{K_n}$ implies $\Delta(G) \geq 1$, we will have:

$$(\Delta(G)+1) \cdot \gamma_R(G) \geq (1+1)|V_1| + 2\,|V_0| + 2\,|V_2| = 2\,n \,.$$

Finally:

$$\gamma_R(G) \geq \frac{2 \cdot n}{\Delta(G)+1}$$

and, as $\gamma_R(G)$ is always an integer:

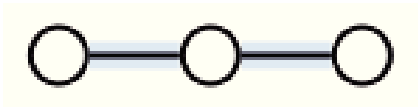
$$\gamma_R(G) \geq \left\lceil \frac{2 \cdot n}{\Delta(G)+1} \right\rceil \,.$$

□

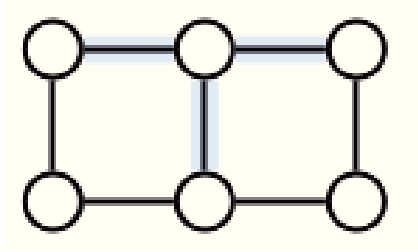
If we compute this lower bound for a grid graph $G_{m,n}$, with $m, n \geq 3$ we will have $|V| = m \cdot n$ and $\Delta(G_{m,n}) = 4$, i.e.:

$$\gamma_R(G_{m,n}) \geq \left\lceil \frac{2 \cdot (m \cdot n)}{4+1} \right\rceil = \left\lceil \frac{2 \cdot (m \cdot n)}{5} \right\rceil \,.$$

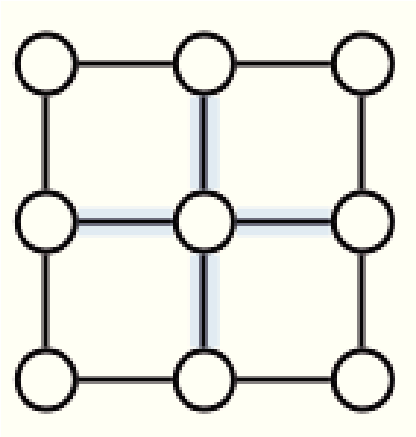
We chose $m, n \geq 3$ because every grid graph $G_{m,n}$ with $m, n \geq 3$ will have $\Delta(G) = 4$, instead a grid graph $G_{2,n}$, with $n \geq 3$, will have $\Delta(G) = 3$ and a grid graph $G_{1,n}$, better known as a path graph, with $n \geq 3$, will have $\Delta(G) = 2$, see Figures 3.1.



(a) Grid graph $G_{1,n}$ with $n \geq 3$



(b) Grid graph $G_{2,n}$ with $n \geq 3$



(c) Grid graph $G_{m,n}$ with $m, n \geq 3$

Figure 3.1: The Maximum Degree for Grid Graphs.

2 Improving the Bound

In order to improve the lower-bound, we introduce now the idea of *cost* (χ) of a vertex, i.e. for each vertex in a given graph we associate a new value that, given a Roman Dominating Function f, represents how much we spend to protect that vertex.

We suppose that the Roman Dominating Function f is weakly non–redundant, i.e. every vertex labeled by 1 has no neighbor labeled by 2.

If a vertex v is labeled by 2, then its value is shared equally among the vertex itself and its neighbors, i.e. among its closed neighborhood $N[v]$. So, the *shared value* (σ) of v among the vertices of its closed neighborhood is

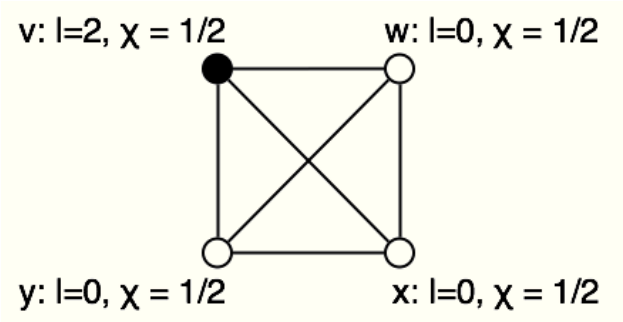


Figure 3.2: The Graph K_4: the vertex v is labeled by 2 and the others by 0. The cost of all the vertices is 1/2.

defined as:

$$\sigma(v, w) = \frac{2}{|N[v]|} \qquad \forall w \in N[v] \ .$$

If a vertex v is labeled by 1, then its value is not shared with any other vertex, i.e.

$$\sigma(v, w) = \begin{cases} 1 & w = v \\ 0 & \forall w \in N(v) \end{cases} .$$

If a vertex v is labeled by 0, then there is no value to be shared among its neighborhood and thus:

$$\sigma(v, w) = 0 \qquad \forall w \in N[v] \ .$$

So the cost of a vertex v is the sum of all the shares from all the vertices in its closed neighborhood, i.e.:

$$\chi(v) = \sum_{w \in N[v]} \sigma(w, v) \ .$$

In particular, if a vertex v is labeled by 1, then its cost is 1, i.e.:

$$\chi(v) = 1 \ .$$

For example, if we consider a K_n, then in a Roman Dominating Function weakly non–redundant, a vertex is labeled by 2 and all the others by 0, and each vertex has a cost of $2/n$, see for example Figure 3.2.

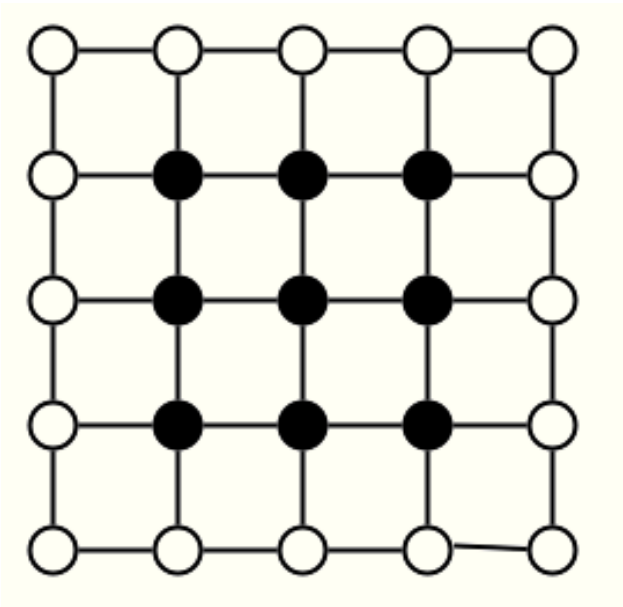


Figure 3.3: The vertices of a grid graph are divided in two sets: in black the internal vertices (Set A) and in white the external ones (Set B).

Clearly, we will have that the sum of all the costs is equivalent to the sum of the value of the chosen Roman Dominating Function, i.e.:

$$\sum_{v \in V} \chi(v) = \sum_{v \in V} f_{RD}(v) \ .$$

In the specific case of grid graphs, we divide the vertices in two sets: A is the set of the internal vertices and B is the set of vertices on the borders, see Figure 3.3.

The minimum cost of the vertices in A is $2/5$. Indeed, in the best case scenario, we have a vertex v labeled by 2 and its four neighbors labeled by 0 and covered only by v.

For the vertices in B, the problem of finding the minimum cost is a bit trickier. In theory, every vertex v on the borders can have a cost of $2/5$, and this is the case when v is covered only by an internal vertex labeled by 2, so

we find the initial lower-bound:

$$\gamma_R(G) \geq \frac{2}{5} \cdot |A| + \frac{2}{5} \cdot |B| = \frac{2}{5} \cdot |V| = \frac{2 \cdot (m \cdot n)}{5} \ .$$

But not every vertex in B has a cost of $2/5$, in fact a vertex in B labeled by 0 can have a cost of:

1. $2/4 + 2/5 + 2/4 = 13/10$, if it has the three adjacent vertices labeled by 2.
2. $2/4 + 2/3 = 7/6$, if it has an adjacent vertex on the border labeled by 2 and an adjacent vertex on the corner labeled by 2.
3. $2/4 + 2/4 = 1$, if it has the two adjacent vertices on the border labeled by 2.
4. $2/5 + 2/4 = 9/10$, if it has the internal adjacent vertex labeled by 2 and an adjacent vertex on the border labeled by 2.
5. $2/3$, if it is covered only by a corner vertex labeled by 2.
6. $2/4$, if it covered only by a border vertex labeled by 2.
7. $2/5$, if it is covered only by an internal vertex labeled by 2.

A vertex in B labeled by 2 can have a cost of:

1. $2/3 + 2/4 + 2/4$, if it is a vertex on the corner, and has two adjacent vertices labeled by 2.
2. $2/4 + 2/4 + 2/4$, if it has the two adjacent vertices on the border labeled by 2 and the internal one labeled by 0.
3. $2/3 + 2/4$, if it is a vertex on the corner, and has an adjacent vertex labeled by 2 and one labeled by 0.
4. $2/4 + 2/4$, if it has an adjacent vertices on the border labeled by 2 and the others adjacent vertices labeled by 0.
5. $2/3$, if it is a vertex on the corner, and has two adjacent vertices labeled by 0.

6. 2/4, if it has the three adjacent vertices labeled by 0.

Finally, a vertex in B labeled by 1 has a cost of 1.

2.1 A Preparatory Lemma

We now define B^* as the set of vertices, whose cost is less than 2/4:

$$B^* = \{v \in B \colon \chi(v) < 2/4\} \ .$$

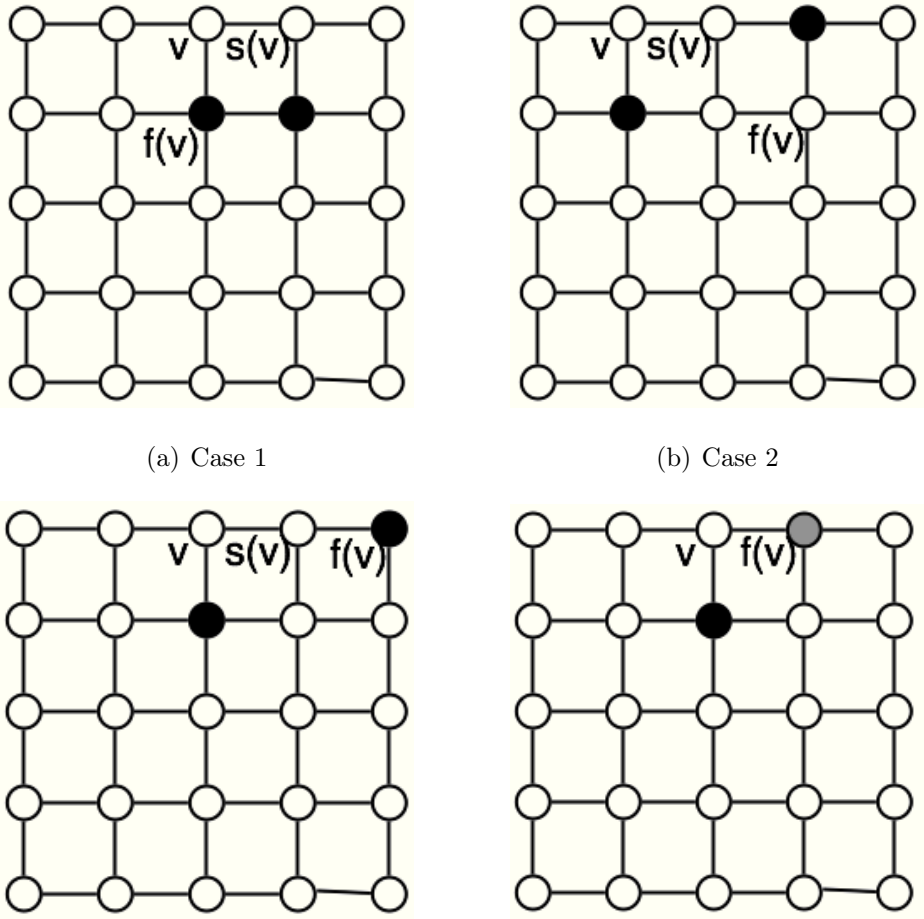
As we said above, the only case for a vertex to be in B^* is when its cost is 2/5 and it is covered only by an internal vertex, see case seven above.

Lemma 8. *Let $G_{m,n}$ be a grid graph, covered by a Roman Dominating Function weakly non–redundant, then there exists a injective function $f \colon B^* \to A \cup B$, such that $\forall v \in B^*$:*

$$\chi(v) + \chi(f(v)) \geq \begin{cases} 2/5 + 2/4 & \text{if} \quad f(v) \in A \\ 2/4 + 2/4 & \text{if} \quad f(v) \in B \end{cases} \ . \tag{3.2}$$

Proof. To prove the lemma, we will move clock-wise along the borders of $G_{m,n}$, starting from the vertex at position $(0, 1)$ and ending at position $(0, 0)$. Given a vertex $v \in B^*$, $s(v)$ and $i(v)$ will be respectively the successor and internal adjacent vertex of v, in such clock-wise order. There are four possible cases:

1. $s(v) \in B^*$, then this implies two adjacent vertices in A labeled by 2, and their cost is at least $2 \cdot 2/5 = 4/5$, so we define $f(v) = i(v)$, see Figure 3.4(a).
2. $s(v) \notin B^*$, with a cost of 2/4, then this implies that $s(s(v))$ is labeled by 2 and $i(s(s(v)))$ has a cost at least2/4, so we define $f(v) = i(s(s(v)))$, as in Figure 3.4(b).



(a) Case 1

(b) Case 2

(c) Case 3

(d) Case 4

Figure 3.4: The four Cases of the Lemma 3.2.

3. $s(v) \notin B^*$, with a cost of 2/3, so we define $f(v) = s(s(v))$, as in Figure 3.4(c).
4. $s(v) \notin B^*$, with a cost of 1, so we define $f(v) = s(v)$, as in Figure 3.4(d).

Clearly, the function f is injective by construction. □

2.2 The New Lower-Bound

We now define a partition of B^* and the images on V of the injective function f of such partition:

- B_1^* as the set of vertices such that

$$v \in B_1^* \quad \text{iff} \quad v \in B^* \wedge f(v) \in A \; ;$$

- B_2^* as the set of vertices such that

$$v \in B_2^* \quad \text{iff} \quad v \in B^* \wedge f(v) \in B \; ;$$

- K' as the set of vertices such that

$$v \in K' \quad \text{iff} \quad \exists\, u \in B^* \colon f(u) = v \wedge v \in A \; ;$$

- K'' as the set of vertices such that

$$v \in K'' \quad \text{iff} \quad \exists\, u \in B^* \colon f(u) = v \wedge v \in B \; .$$

Once defined the sets above, we can demonstrate our theorem:

Theorem 9. *Given a Roman Dominating Function weakly non–redundant, and the function f defined above, the Roman Domination Number of a grid graph $(G_{m,n})$, with $m, n \geq 3$, has a lower-bound of:*

$$\gamma_R(G) \geq \frac{|A| \cdot 2}{5} + \frac{|B| \cdot 2}{4} \; .$$

Proof.

$$\sum_{v \in V} \chi(v) = \sum_{v \in A} \chi(v) + \sum_{v \in B} \chi(v) =$$

$$= \sum_{v \in A - K'} \chi(v) + \sum_{v \in B - (B^* \cup K'')} \chi(v) + \sum_{v \in K'} \chi(v) + \sum_{v \in B^*} \chi(v) + \sum_{v \in K''} \chi(v) \geq$$

As $|B^*| = |B_1^*| + |B_2^*|$

$$= \sum_{v \in A - K'} \chi(v) + \sum_{v \in B - (B^* \cup K'')} \chi(v) + \sum_{v \in K'} \chi(v) +$$

$$+\sum_{v\in B_1^*}\chi(v)+\sum_{v\in B_2^*}\chi(v)+\sum_{v\in K''}\chi(v)=$$

$$=\sum_{v\in A-K'}\chi(v)+\sum_{v\in B-(B^*\cup K'')}\chi(v)+\left(\sum_{v\in B_1^*}\chi(v)+\sum_{v\in K'}\chi(v)\right)+$$

$$+\left(\sum_{v\in B_2^*}\chi(v)+\sum_{v\in K''}\chi(v)\right)=$$

and $|K'|=|B_1^*|, |K''|=|B_2^*|$

$$=\sum_{v\in A-K'}\chi(v)+\sum_{v\in B-(B^*\cup K'')}\chi(v)+\left(\sum_{v\in B_1^*}\chi(v)+\sum_{v\in B_1^*}\chi(f(v))\right)+$$

$$+\left(\sum_{v\in B_2^*}\chi(v)+\sum_{v\in B_2^*}\chi(f(v))\right)\geq$$

for Equation (3.2)

$$\geq\sum_{v\in A-K'}\chi(v)+\sum_{v\in B-(B^*\cup K'')}\chi(v)+\left(\sum_{v\in B_1^*}\left(\frac{2}{5}+\frac{2}{4}\right)\right)+\left(\sum_{v\in B_2^*}\left(\frac{2}{4}+\frac{2}{4}\right)\right)\geq$$

$$\geq\frac{|A-K'|\cdot 2}{5}+\frac{|B-(B^*\cup K'')|\cdot 2}{4}+\sum_{v\in B_1^*}\frac{2}{5}+\sum_{v\in B_1^*}\frac{2}{4}+\sum_{v\in B_2^*}\frac{2}{4}+\sum_{v\in B_2^*}\frac{2}{4}\geq$$

$$\geq\frac{|A-K'|\cdot 2}{5}+\frac{|B-(B^*\cup K'')|\cdot 2}{4}+\sum_{v\in K'}\frac{2}{5}+\sum_{v\in B_1^*}\frac{2}{4}+\sum_{v\in B_2^*}\frac{2}{4}+\sum_{v\in K''}\frac{2}{4}\geq$$

$$\geq\frac{(|A-K'|+|K'|)\cdot 2}{5}+\frac{(|B-(B^*\cup K'')|+|B_1^*|+|B_2^*|+|K''|)\cdot 2}{4}=$$

$$=\frac{|A|\cdot 2}{5}+\frac{|B|\cdot 2}{4}\,.$$

□

3 Conclusions

If we consider a grid graph $G_{m,n}$ with $m, n \geq 3$, the cardinality of the set of internal vertices of G is

$$|A| = (m-2) \cdot (n-2)$$

and the cardinality of the external vertices is

$$|B| = m \cdot n - ((m-2) \cdot (n-2)) = 2m + 2n - 4 \,,$$

then the lower-bound for a grid graph is:

$$\gamma_R(G_{m,n}) \geq \frac{|A| \cdot 2}{5} + \frac{|B| \cdot 2}{4} = \frac{(m-2) \cdot (n-2) \cdot 2}{5} + \frac{(2m+2n-4) \cdot 2}{4} =$$
$$= \frac{((2 \cdot m \cdot n + 8 - 4m - 4n) + (5m + 5n - 10))}{5} =$$
$$= \frac{(2 \cdot m \cdot n + m + n - 2)}{5}$$

As before, we can use the ceiling function, so we will have:

$$\gamma_R(G_{m,n}) \geq \left\lceil \frac{(2 \cdot m \cdot n + m + n - 2)}{5} \right\rceil$$

This is a sharper lower-bound, in fact if:

$$\frac{m+n-2}{5} > 1 \quad \rightarrow \quad m+n-2 > 5 \quad \rightarrow \quad m+n > 7$$

we will have:

$$\left\lceil \frac{(2 \cdot m \cdot n + m + n - 2)}{5} \right\rceil \geq \left\lceil \frac{2 \cdot (m \cdot n)}{5} + 1 \right\rceil > \left\lceil \frac{2 \cdot (m \cdot n)}{5} \right\rceil \,.$$

Chapter 4

Improving the bounds: The Upper-Bound

1 Introduction

In this chapter we will prove a strict upper-bound that we believe can be the Roman Domination Number for most, if not all, grid graphs.

As grid graphs have a regular structure, given any grid graph $G_{m,n}$ and its transpose $G_{n,m}$, it follows

$$\gamma_R(G_{m,n}) = \gamma_R(G_{n,m}) \ .$$

Remember that, given a grid graph $G_{m,n}$, we denoted by $\gamma_i(G_{m,n})$ the Roman Domination value obtained by schema S_i, see Subsection 2.2 of Chapter 2. In what follows, we will use only the schema S_2 as covering method, because it always gives the best results, as we can see in Table 2.2, except for two classes of grid graphs: $G_{1+5h,3+5k}$ and $G_{2+5h,5k}$, for integers $h, k \geq 1$. In such cases, though, the schema S_2 gives the best results on the transpose grid graphs, see again Table 2.2.

Thus we define $\gamma_2^*(G_{m,n})$ as

$$\gamma_2^*(G_{m,n}) = \min\{\gamma_2(G_{m,n}), \gamma_2(G_{n,m})\}.$$

We will now state our main theorem, which, as said before, improves the upper-bound given in Theorem 6.

Theorem 10 (Grid Theorem)**.** *Let $G_{m,n}$ be a grid graph, with $m, n \geq 5$. Then, by using Schema S_2:*

$$\gamma_R(G_{m,n}) \leq \gamma_2^*(G_{m,n}) = \begin{cases} \left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor - 1 & \text{if} \quad m, n \mod 5 = 4 \\ \\ \left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor & \text{otherwise} \end{cases} . \tag{4.1}$$

Initially, we will prove that the Grid Theorem is true in some special cases, see Lemma 11 and Lemma 13, then we will prove the general case.

2 Preparatory Lemmas

We will prove now the first lemma.

Lemma 11. *The Grid Theorem is true for the grid graphs $G_{m,n}$ with $5 \leq m, n \leq 9$.*

Proof. The Lemma follows from Table 4.1. □

In Table 4.1 we can see a comparison, for the grid graphs $G_{m,n}$ with $5 \leq m, n \leq 9$, between the exact value of $\gamma_R(G)$, $\gamma_2^*(G)$, the Grid Theorem and Theorem 6. The exact values of $\gamma_R(G)$ are given by the Equations on Page 27.

Grid Graph	γ_R	γ_2^*	Grid Theorem	Theorem 6
5x5	**14**	**14**	**14**	14
5x6	**16**	**16**	**16**	18
5x7	**18**	**18**	**18**	20
5x8	**21**	**21**	**21**	22
5x9	**23**	**23**	**23**	24
6x5	**16**	**16**	**16**	18
6x6	**19**	**19**	**19**	24
6x7	**22**	**22**	**22**	26
6x8	**24**	**24**	**24**	28
6x9	**27**	**27**	**27**	30
7x5	**18**	**18**	**18**	20
7x6	**22**	**22**	**22**	26
7x7	**24**	25	25	28
7x8	**28**	**28**	**28**	32
7x9	**31**	**31**	**31**	34
8x5	**21**	**21**	**21**	22
8x6	**24**	**24**	**24**	28
8x7	**28**	**28**	**28**	32
8x8	**32**	**32**	**32**	34
8x9	**35**	**35**	**35**	38
9x5	**23**	**23**	**23**	24
9x6	**27**	**27**	**27**	30
9x7	**31**	**31**	**31**	34
9x8	**35**	**35**	**35**	38
9x9	n.d.	**38**	**38**	42

Table 4.1: A comparison between the exact value of γ_R, $\gamma_2^*(G)$, the Grid Theorem and Theorem 6.

We can add that, with the exception of $G_{7,7}$, of which we will discuss in the conclusions, and $G_{9,9}$, the Grid Theorem gives the exact value of γ_R, as we can see in Table 4.1. We do not know the exact value of $\gamma_R(G_{9,9})$, but as we know $\gamma_R(G_{8,9})$, we can say:

$$35 = \gamma_R(G_{8,9}) \leq \gamma_R(G_{9,9}) \leq \gamma_2^*(G_{9,9}) = 38$$

$$35 \leq \gamma_R(G_{9,9}) \leq 38 \, .$$

The next lemma, about a special case of graphs, a Cartesian Product of a path e a cycle, will be useful for the demonstration of the Lemma 13 and for the Grid Theorem.

Lemma 12. *Let $G = C_5 \otimes P_n$ with $n \geq 5$ be a* Cartesian product *of a cycle graph with* 4 *edges and a path graph with $n-1$ edges. Then, G can be covered with any schema S_i and we will have:*

$$\gamma_i(C_5 \otimes P_n) = \gamma_i(P_n \otimes C_5) = \gamma_R(C_5 \otimes P_n) = 2 \cdot n + 2 \, .$$

Proof. The Cartesian product of a cycle graph with 4 edges and a path graph with $n-1$ edges is a cylinder with a base formed by 5 vertices and height of n vertices.

Whatever schema you choose, it is invariant respect a rotation along the height. Using any schema S_i, we have to place a label 2 in each column; only 2 vertices, one for each base, remain uncovered and has to be labeled by 1. This is the optimal solution for this kind of graphs, i.e.:

$$\gamma_i(C_5 \otimes P_n) = \gamma_i(P_n \otimes C_5) = \gamma_R(C_5 \otimes P_n) = 2 \cdot n + 2 \, .$$

□

A mathematical proof is found in [46].

The next lemma is an intermediate step, in which we prove the Grid Theorem for grid graphs of fixed m.

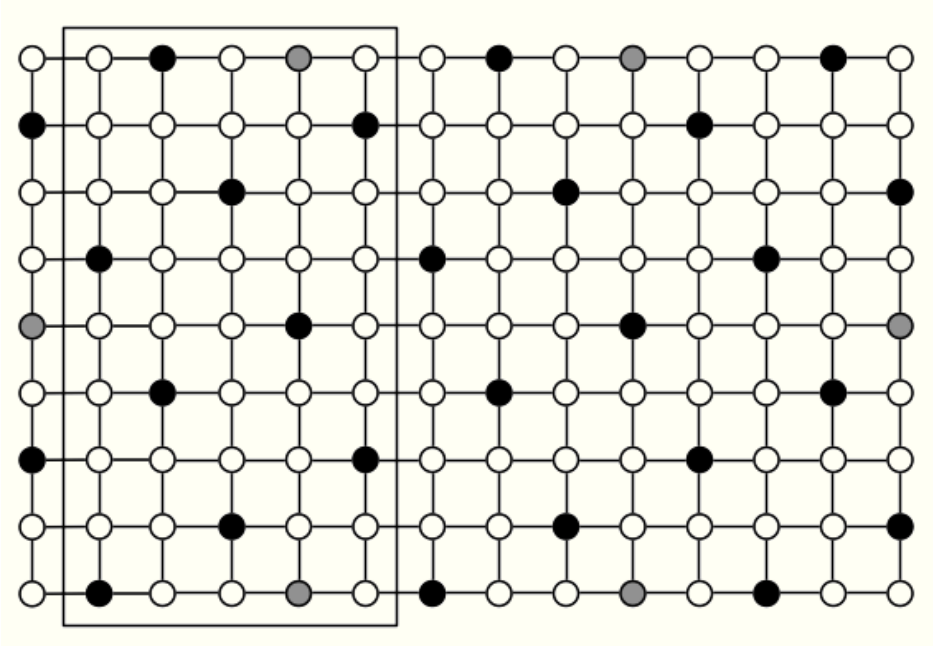


Figure 4.1: Grid Graph $G_{9,9}$ plus a insertion of Grid Graph $G_{9,5}$.

Lemma 13. *The Grid Theorem is true for grid graphs $G_{m,n}$, with $5 \leq m \leq 9$ and $n \geq 10$.*

Proof. Let $k' = n \mod 5$, and let $b > 0$ be the integer such that $n = 5 \cdot (b+1) + k'$. Then $n = 5 \cdot b + k$ with $5 \leq k \leq 9$, where $k = k' + 5$.

Grid graph $G_{m,n}$ can be seen as an insertion of grid graph $G_{m,5 \cdot b}$ in grid graph $G_{m,k}$, as we can see in Figure 4.1. Moreover, the grid graph $G_{m,5 \cdot b}$ can be seen as a composition of b sub-grid graphs of type $G_{m,5}$. Each of these b sub-grid graphs can be seen as an *open* graph $C_5 \otimes P_m$, and by Lemma 12, it can be covered using m labels 2 (one per row) and 2 labels 1 (one for the upper row and one for the lower row), see Figure 4.1.

Thus

$$\gamma_R(G_{m,n}) \leq \gamma_2^*(G_{m,n}) = (2 \cdot m + 2) \cdot b + \gamma_2^*(G_{m,k}) \tag{4.2}$$

Because $5 \leq m, k \leq 9$, by Lemma 11 it follows:

$$\gamma_2^*(G_{m,k}) = \begin{cases} \left\lfloor \frac{2 \cdot (m \cdot k + m + k)}{5} \right\rfloor - 1 & \text{if } m, k \mod 5 = 4 \\ \left\lfloor \frac{2 \cdot (m \cdot k + m + k)}{5} \right\rfloor & \text{otherwise} \end{cases}.$$

We discuss the first case, the second is analogous.

Let $m = 9$ and $k = 9$, then using Equation 4.2 we obtain

$$\gamma_R(G_{m,n}) \leq \gamma_2^*(G_{m,n}) = (2 \cdot m + 2) \cdot b + \left\lfloor \frac{2 \cdot (m \cdot k + m + k)}{5} \right\rfloor - 1 =$$

$$= \left\lfloor \frac{5 \cdot (2 \cdot m + 2) \cdot b + 2 \cdot (m \cdot k + m + k)}{5} \right\rfloor - 1 =$$

$$= \left\lfloor \frac{2 \cdot (5 \cdot m \cdot b + 5 \cdot b + m \cdot k + m + k)}{5} \right\rfloor - 1 =$$

$$= \left\lfloor \frac{2 \cdot (m \cdot (5 \cdot b + k) + m + (5 \cdot b + k))}{5} \right\rfloor - 1 =$$

$$= \left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor - 1$$

□

3 Equality for Some type of Grid Graphs

In the following Lemma, we show that for grid graphs $G_{m,n}$ with $5 \leq m \leq 8$ and $n \geq 5$, with the exception of grid graph $G_{7,7}$, our upper-bound is the Roman Domination Number.

Lemma 14. *For grid graphs $G_{m,n}$ with $5 \leq m \leq 8$ and $n \geq 5$, with the exception of grid graph $G_{7,7}$, the Grid Theorem gives the value of $\gamma_R(G_{m,n})$.*

Proof. The equations for $\gamma_R(G_{m,n})$ are that of Page 27.

We have four cases:

- $m = 5$

$$\left\lfloor \frac{2 \cdot (5 \cdot n + 5 + n)}{5} \right\rfloor = \left\lfloor \frac{12 \cdot n + 10}{5} \right\rfloor = \left\lfloor \frac{12 \cdot n}{5} \right\rfloor + 2 = \gamma_R(G_{5,n})$$

- $m = 6$

$$\left\lfloor \frac{2 \cdot (6 \cdot n + 6 + n)}{5} \right\rfloor = \left\lfloor \frac{14 \cdot n + 12}{5} \right\rfloor =$$

$$= \begin{cases} 14k + 2 & n = 5k \\ 14k + \lfloor 26/5 \rfloor = 14k + 5 & n = 5k + 1 \\ 14k + \lfloor 40/5 \rfloor = 14k + 8 & n = 5k + 2 \\ 14k + \lfloor 54/5 \rfloor = 14k + 10 & n = 5k + 3 \\ 14k + \lfloor 68/5 \rfloor = 14k + 13 & n = 5k + 4 \end{cases} =$$

$$= \begin{cases} 14k + 2 & n = 5k \\ 14k + \lfloor 14/5 \rfloor + 3 = \lfloor 14n/5 \rfloor + 3 & n = 5k + 1 \\ 14k + \lfloor 28/5 \rfloor + 3 = \lfloor 14n/5 \rfloor + 3 & n = 5k + 2 \\ 14k + \lfloor 42/5 \rfloor + 2 = \lfloor 14n/5 \rfloor + 2 & n = 5k + 3 \\ 14k + \lfloor 56/5 \rfloor + 2 = \lfloor 14n/5 \rfloor + 2 & n = 5k + 4 \end{cases} =$$

$$= \gamma_r(G_{6,n})$$

- $m = 7 \quad (n \neq 7)$

$$\left\lfloor \frac{2 \cdot (7 \cdot n + 7 + n)}{5} \right\rfloor = \left\lfloor \frac{16 \cdot n + 14}{5} \right\rfloor =$$

$$= \begin{cases} 16k + 2 & n = 5k \\ 16k + \lfloor 30/5 \rfloor = 16k + 6 & n = 5k + 1 \\ 16k + \lfloor 46/5 \rfloor = 16k + 9 & n = 5k + 2 \\ 16k + \lfloor 62/5 \rfloor = 16k + 12 & n = 5k + 3 \\ 16k + \lfloor 78/5 \rfloor = 16k + 15 & n = 5k + 4 \end{cases} =$$

$$= \begin{cases} 16k+2 & n = 5k \\ 16k + \lfloor 16/5 \rfloor + 3 = \lfloor 16n/5 \rfloor + 3 & n = 5k+1 \\ 16k + \lfloor 32/5 \rfloor + 3 = \lfloor 16n/5 \rfloor + 3 & n = 5k+2 \\ 16k + \lfloor 48/5 \rfloor + 3 = \lfloor 16n/5 \rfloor + 3 & n = 5k+3 \\ 16k + \lfloor 64/5 \rfloor + 3 = \lfloor 16n/5 \rfloor + 3 & n = 5k+4 \end{cases} =$$

$$= \gamma_r(G_{7,n})$$

- $m = 8$

$$\left\lfloor \frac{2 \cdot (8 \cdot n + 8 + n)}{5} \right\rfloor = \left\lfloor \frac{18 \cdot n + 16}{5} \right\rfloor =$$

$$= \begin{cases} 18k+3 & n = 5k \\ 18k + \lfloor 34/5 \rfloor = 18k+6 & n = 5k+1 \\ 18k + \lfloor 52/5 \rfloor = 18k+10 & n = 5k+2 \\ 18k + \lfloor 70/5 \rfloor = 18k+14 & n = 5k+3 \\ 18k + \lfloor 88/5 \rfloor = 18k+17 & n = 5k+4 \end{cases} =$$

$$= \begin{cases} 18k+3 & n = 5k \\ 18k + \lfloor 18/5 \rfloor + 3 = \lfloor 18n/5 \rfloor + 3 & n = 5k+1 \\ 18k + \lfloor 36/5 \rfloor + 3 = \lfloor 18n/5 \rfloor + 3 & n = 5k+2 \\ 18k + \lfloor 54/5 \rfloor + 4 = \lfloor 18n/5 \rfloor + 4 & n = 5k+3 \\ 18k + \lfloor 72/5 \rfloor + 3 = \lfloor 18n/5 \rfloor + 3 & n = 5k+4 \end{cases} =$$

$$= \gamma_r(G_{8,n})$$

□

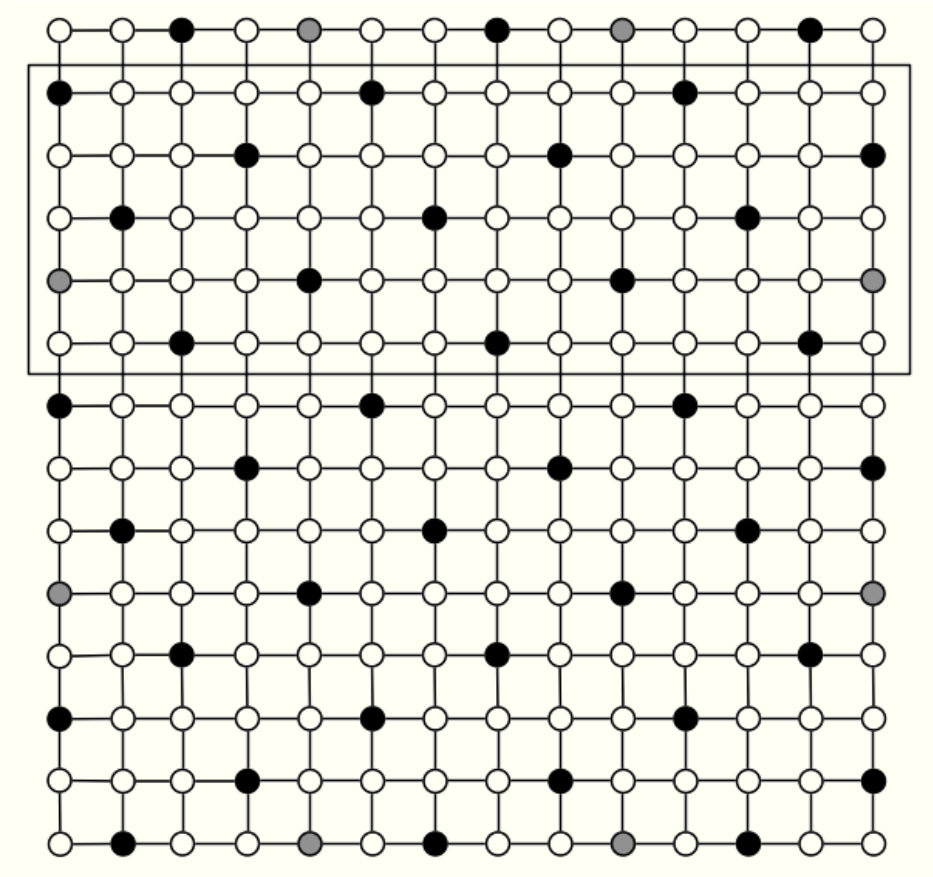


Figure 4.2: Grid Graph $G_{9,14}$ plus a insertion of Grid Graph $G_{5,14}$.

4 The Proof of the Grid Theorem

Let us now prove the general case of the Grid Theorem.

Theorem (Grid Theorem)**.** *Let $G_{m,n}$ be a grid graph, with $m, n \geq 5$. Then, by using Schema S_2:*

$$\gamma_R(G_{m,n}) \leq \gamma_2^*(G_{m,n}) = \begin{cases} \left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor - 1 & \textit{if} \quad m, n \mod 5 = 4 \\ \left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor & \textit{otherwise.} \end{cases} \tag{4.3}$$

Proof. Let $G_{m,n}$ be a grid graph with $m, n \geq 5$.

If $5 \leq m, n \leq 9$, the proof follows from Lemma 11.

If $5 \leq m \leq 9$ and $n \geq 10$, or if $5 \leq n \leq 9$ and $m \geq 10$ the proof follows from Lemma 13.

Else, let $G_{m,n}$ be a grid graph with $m, n \geq 10$.

Let $h' = m \mod 5$, and let $a > 0$ be the integer such that $m = 5 \cdot (a + 1) + h'$. Then $m = 5 \cdot a + h$, where $h = h' + 5$.

Grid graph $G_{m,n}$ can be seen as an insertion of grid graph $G_{5 \cdot a,n}$ in grid graph $G_{h,n}$, as we can see in Figure 4.2. Moreover, the grid graph $G_{5 \cdot a,n}$ can be seen as a composition of a sub-grid graphs of type $G_{5,n}$. Each of these b sub-grid graphs can be seen as an *open* graph $C_5 \otimes P_n$, and by Lemma 12, it can be covered using n labels 2 (one per column) and 2 labels 1 (one for the leftmost column and one for the rightmost column), see Figure 4.2.

Thus

$$\gamma_R(G_{m,n}) \leq \gamma_2^*(G_{m,n}) = (2 \cdot n + 2) \cdot a + \gamma_2^*(G_{h,n}) \ . \tag{4.4}$$

Because $5 \leq h \leq 9$, by Lemma 13 it follows:

$$\gamma_2^*(G_{h,n}) = \begin{cases} \left\lfloor \frac{2 \cdot (h \cdot n + h + n)}{5} \right\rfloor - 1 & \text{if} \quad h, n \mod 5 = 4 \\ \\ \left\lfloor \frac{2 \cdot (h \cdot n + h + n)}{5} \right\rfloor & \text{otherwise} \end{cases} .$$

Again, we discuss the first case, the second is analogous.

Let $h = 9$ and $n \mod 5 = 4$, then, using Equation 4.4 we obtain

$$\gamma_R(G_{m,n}) \leq \gamma_2^*(G_{m,n}) = (2 \cdot n + 2) \cdot a + \left\lfloor \frac{2 \cdot (h \cdot n + h + n)}{5} \right\rfloor - 1 =$$

$$= \left\lfloor \frac{5 \cdot a \cdot (2 \cdot n + 2) + 2 \cdot (h \cdot n + h + n)}{5} \right\rfloor - 1 =$$

$$= \left\lfloor \frac{2 \cdot (5 \cdot a \cdot n + 5 \cdot a + h \cdot n + h + n)}{5} \right\rfloor - 1 =$$

$$= \left\lfloor \frac{2 \cdot ((5 \cdot a + h) \cdot n + (5 \cdot a + h) + n)}{5} \right\rfloor - 1 =$$

$$= \left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor - 1$$

□

5 Conclusions and Open Problems

As we can clearly see, the Grid Theorem gives a sharper upper-bound than the Theorem 6 over the Grid Graphs. Indeed, we have:

- $m, n \mod 5 = 4$:

$$\left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor - 1 \leq 2 \cdot \left(\left\lceil \frac{m \cdot n}{5} \right\rceil + \left\lceil \frac{m}{5} \right\rceil + \left\lceil \frac{n}{5} \right\rceil \right)$$

- otherwise:

$$\left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor \leq 2 \cdot \left(\left\lceil \frac{m \cdot n}{5} \right\rceil + \left\lceil \frac{m}{5} \right\rceil + \left\lceil \frac{n}{5} \right\rceil \right)$$

The equality holds only when $m, n \mod 5 = 0$. In all the other cases, the difference between the old upper-bound and the new one is at most 5, as in case of the grid graph $G_{6,6}$; although the improvement is very minimal, is nonetheless an important improvement.

We show now that the Grid Theorem also gives sharp upper-bounds to the classes of graphs $G_{1,n}$, $G_{2,n}$, $G_{3,n}$ and $G_{4,n}$. We have:

- $G_{1,n}$, i.e. $m = 1$:

$$\left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor = \left\lfloor \frac{2 \cdot (1 \cdot n + 1 + n)}{5} \right\rfloor = \left\lfloor \frac{2 \cdot (2 \cdot n + 1)}{5} \right\rfloor =$$

$$= \left\lfloor \frac{4 \cdot n + 2}{5} \right\rfloor = \left\lfloor \frac{4 \cdot n}{5} + \frac{2}{5} \right\rfloor = \left\lfloor \frac{12 \cdot n}{15} + \frac{6}{15} \right\rfloor =$$

  - if $n < 5$: by enumeration;
  - if $n \geq 5$:

$$= \left\lfloor \frac{10 \cdot n}{15} + \frac{2 \cdot n}{15} + \frac{6}{15} \right\rfloor \geq \left\lfloor \frac{10 \cdot n}{15} + \frac{10}{15} + \frac{6}{15} \right\rfloor \geq$$

$$\geq \left\lfloor \frac{10 \cdot n}{15} + \frac{16}{15} \right\rfloor \geq \left\lceil \frac{10 \cdot n}{15} \right\rceil \geq \left\lceil \frac{2 \cdot n}{3} \right\rceil \geq \gamma_R(G_{1,n})$$

- $G_{2,n}$, i.e. $m = 2$:

$$\left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor = \left\lfloor \frac{2 \cdot (2 \cdot n + 2 + n)}{5} \right\rfloor = \left\lfloor \frac{2 \cdot (3 \cdot n + 2)}{5} \right\rfloor =$$

$$= \left\lfloor \frac{6 \cdot n + 4}{5} \right\rfloor = \left\lfloor \frac{6 \cdot n}{5} + \frac{4}{5} \right\rfloor \geq \left\lfloor \frac{5 \cdot n}{5} + \frac{1}{5} + \frac{4}{5} \right\rfloor \geq n + 1 \geq \gamma_R(G_{2,n})$$

- $G_{3,n}$, i.e. $m = 3$:

$$\left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor = \left\lfloor \frac{2 \cdot (3 \cdot n + 3 + n)}{5} \right\rfloor = \left\lfloor \frac{2 \cdot (4 \cdot n + 3)}{5} \right\rfloor =$$

$$= \left\lfloor \frac{8 \cdot n + 6}{5} \right\rfloor = \left\lfloor \frac{8 \cdot n}{5} + \frac{6}{5} \right\rfloor = \left\lfloor \frac{16 \cdot n}{10} + \frac{12}{10} \right\rfloor$$

  - if $n = 1$:

$$\geq \left\lfloor \frac{15 \cdot n}{10} + \frac{10}{10} \right\rfloor \geq \left\lfloor \frac{3 \cdot n}{2} + 1 \right\rfloor \geq \gamma_R(G_{3,1})$$

  - if $n \neq 1$:

$$\geq \left\lfloor \frac{15 \cdot n}{10} + \frac{n}{10} + \frac{12}{10} \right\rfloor \geq \left\lceil \frac{15 \cdot n}{10} + \frac{10}{10} \right\rceil \geq$$

$$\geq \left\lceil \frac{3 \cdot n}{2} + 1 \right\rceil \geq \gamma_R(G_{3,n})$$

- $G_{4,n}$, i.e. $m = 4$:

$$\left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor = \left\lfloor \frac{2 \cdot (4 \cdot n + 4 + n)}{5} \right\rfloor = \left\lfloor \frac{2 \cdot (5 \cdot n + 4)}{5} \right\rfloor =$$

$$= \left\lfloor 2 \cdot n + \frac{8}{5} \right\rfloor = 2 \cdot n + 1 \geq \gamma_R(G_{4,n})$$

We have indeed an equation that is a sharp upper-bound to all type of grid graphs.

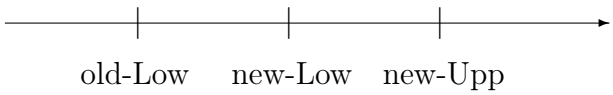


Figure 4.3: A comparison between the old lower-bound, the new lower-bound and the new upper-bound.

A Comparison between The New Bounds

Summing up, if we make a comparison between the new lower- and the new upper-bound, we have:

$$\left\lceil \frac{(2 \cdot m \cdot n + m + n - 2)}{5} \right\rceil \leq \gamma_R(G_{m,n}) \leq \left\lfloor \frac{2 \cdot (m \cdot n + m + n)}{5} \right\rfloor .$$

The gap is circa $(m + n)/5 = \Theta(m + n)$, which is sub-linear, given the number of vertices, $m \cdot n$, of a grid graph. Also, we halved the gap between the old lower- and upper-bound, see Figure 4.3:

$$(m + n)/5 \simeq \frac{\left(2 \cdot \left(\left\lfloor \frac{m \cdot n}{5} \right\rfloor + \left\lfloor \frac{m}{5} \right\rfloor + \left\lfloor \frac{n}{5} \right\rfloor\right) - \left\lceil \frac{(2 \cdot m \cdot n)}{5} \right\rceil\right)}{2} .$$

5.1 Open Problems

In this chapters, we have presented a method for producing good coverings for the Roman Domination Problem over grid graphs. In particular, we proved that these coverings have a sharper upper-bound on the Roman Domination Number over grid graphs, with respect to the previous known upper-bound, see Theorem 6 and [12].

We already proved that for grid graphs $G_{m,n}$ with $5 \leq m \leq 8$ and $n \geq 5$, except the grid graph $G_{7,7}$, and grid graphs $G_{m,n}$ with $m = 9$ and $5 \leq n \leq 8$, our upper-bound is the Roman Domination Number.

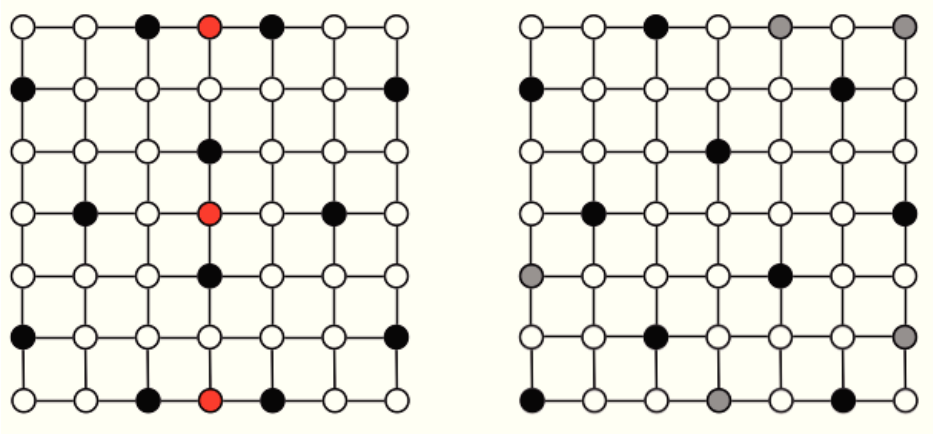


Figure 4.4: Grid Graph $G_{7,7}$. On the left we have the optimal covering, $\gamma_R = 24$, with in red the vertices covered by two black ones. On the right we have the covering by schema S_2, $\gamma_2 = 25$.

The grid graph $G_{7,7}$ is an exception, see Figure 4.4, because its optimal covering is a variation of our schema S_2, where some internal vertices were covered by two vertices labeled by 2. We conjecture that this is the only case where our method fails. In fact, attempts to replicate the schema used in the optimal covering of grid graph $G_{7,7}$ on other grid graphs failed; we suppose because the number of internal vertices covered by two vertices labeled by 2 increased, leading to a non–optimal covering; while with our method the internal vertex are always covered by only one vertex labeled by 2, and the number of vertices labeled by 1 is fixed.

So, as the covering method we proposed enjoys the strong Property of non–Redundancy, see Property 5, and given the regular structure of the grid graphs, we conjecture that, given a grid graph $G_{m,n}$ with $m, n \geq 5$, except the grid graph $G_{7,7}$, our upper-bound is the exact value of the Roman Domination Number of that grid graph, i.e.

$$\text{if} \quad m, n \geq 5 \text{ and } m, n \neq 7 \quad \text{then} \quad \gamma_R(G_{m,n}) = \gamma_2^*(G_{m,n}) \ .$$

Chapter 5

The Implementations

"Good programmers know what to write.
Great ones know what to rewrite (and reuse)."
- Eric Steven Raymond -

1 Introduction

As we said, the Roman Domination Problem is feasible only for specific class of graph, see for example [40]: *interval graphs*, *co-graphs*, *trees*.

As for generic graphs the Roman Domination Problem is $\mathcal{NP}-$hard, in this chapter we discuss some implementations: we tried to find heuristic algorithms that could compute in polynomial time a Roman Dominating Function that was the best possible. To do so, first of all, we had to generate a congruent dataset on which we could test our algorithms, because there is no such thing freely available.

We implemented our graphs' generators using *Objective–C* and the *XCode* integrated enviroment. After, we implemented some heuristics and genetic Algorithms, again using Objective–C and XCode.

The following contents are object of the article *Heuristics with Dynamic Parameters for Roman Domination Problem*, to be submitted, [13], and are explained in detail in the dissertation *Algoritmi Euristici per il Problema della Dominazione Romana* by Dott. Mario Nolassi, see [44].

2 Graphs Dataset Generation

We generated a graphs' dataset large enough to be representative of the class of graphs that we wanted to study and on which we could try the heuristics and the genetic algorithms for finding good roman covering.

We generated:

- Gilbert random graphs;
- Barabási-Albert random graphs;
- bipartite graphs;
- planar graphs;
- grid graphs.

The theory of random graphs is a miscellaneous in-between graph theory and probability theory, see [6]. To generate a random graph we need some random process that we can use as *seed*.

The first random model we implemented is the Gilbert's random model, see [24]. This model is the most simple and is denoted by $G(n, p)$ where n is the number of vertices, while $0 \leq p \leq 1$ is the edge's probability of existence. Initially we generate a set of n isolated vertices and then we randomly add edges between them.

However, sometimes a graph, even if random, need to have some properties and so the generation procedure can have some parameters. Different

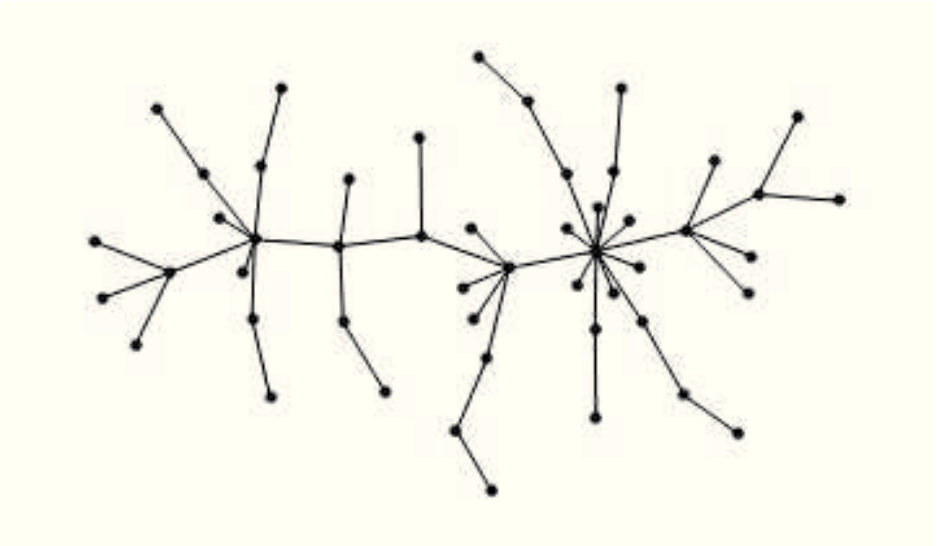


Figure 5.1: A Barabási-Albert Random Graph.

parameters can generate different kind of random graphs and will produce a different probability function.

An example of this is the second random model we implemented: the Barabási-Albert's random model, see [5]. In this model is introduced a new idea: the scale-free networks. With this property, the vertices' degree follow a power law, i.e. they have a scale-free degree distribution, see Figure 5.1. This degree distribution follow a power law of the form

$$P(K) \sim K^{-3}$$

A characteristic of this model is that there exist some vertices called *hub*, that have a degree much larger than the mean.

Scale-free networks are widely observed in natural and human-made systems, including the Internet, the world wide web, citation networks, and some social networks. It incorporates two important general concepts: growth and preferential attachment.Growth means that the number of vertices in the network increases over time. Preferential attachment means that the more connected a vertex is, the more likely it is to receive new links. Vertices with

higher degree have stronger ability to grab links added to the network.

Intuitively, see [55] and [42], the preferential attachment can be understood if we think in terms of social networks connecting people. Here a link from A to B means that person A *knows* or *is acquainted with* person B. Heavily linked vertices represent well-known people with lots of relations. When a newcomer enters the community, he is more likely to become acquainted with one of those more visible people rather than with a relative unknown.

Using the Gilbert model we generated also bipartite graphs and planar graphs. For the planar graph, initially, we associated a pair of geometric coordinates to every vertices, and we randomly linked the vertices giving an higher probability to nearer vertices. After that, we verify the planarity using a planarity test, [36].

Also we generated the grid graphs, to support the theoretic research.

For all the graphs we generated for our dataset, n varies between 50 and 10.000, and when arise necessity to define a probability of connectivity p, it varies between 0.01 and 0.9.

We decided to use the adjacency list for representing the graphs and memorize its using a formatted text file: the GDL (Graph Description Language) format, implementing the interface for importing and exporting the graphs. Such format can be output on screen using *aiSee*, a software whose design has been optimized to handle automatically large graphs generated by other applications and moreover can minimize the intersection of the edges when drawing the graph, see [17] and [38].

3 Heuristic Algorithms

In this section we will discuss the heuristic algorithms we used to obtain a good approximation of the Roman Domination Number for generic grid graphs.

3.1 The Simple Heuristic

The first heuristic we implemented is the most simple and straightforward; using a greedy algorithm we:

- sort in decreasing order the vertices using their degree;
- label the first vertex by 2 and its open neighborhood by 0;
- if a vertex has no open neighborhood or its neighborhood has been already labeled, label it with 1;
- continue until there are no more vertices unlabeled.

With this strategy, we label with 2 the most connected vertices. Also, if we use a post-processing routine, we can have better results: we reduce the vertices labeled by 1, i.e. if a vertex v labeled by 0 has two or more vertices u_i in its neighborhood labeled by 1, then we can label v with 2 and the vertices u_i with 0.

This heuristic has complexity $O(|V|\cdot|E|)$ time, as that is the time occurring to the processing routine and to the post-processing routine.

3.2 The GainFactor Heuristic

This heuristic use a new parameter: a value, called *GainFactor*, associated to each vertex; specifically, the GainFactor of a vertex v means the gain that

we could have if we label v with 2, and labeling with 0 its neighborhood vertices, previously labeled by 1; it is a dynamic parameter, because its value can change during the computation.

We define:

$$GainFactor(v,t) = \text{label}(v) - 2 + \text{adj}_1(v)$$

where t is the time of computation, label(v) is the current labeling of v and $\text{adj}_1(v)$ is the count of all the vertices labeled by 1 in the open neighborhood of v. If the sum is negative, we pone the GainFactor equals to 0. We can prove that, during the computation, the GainFactor of each vertex can only decrease, because $\text{adj}_1(v)$ can only decrease and label(v) can only increase when v is selected for processing and this implies that $\text{adj}_1(v)$ will decrease.

When we select a vertex v for processing, we modify its label from 0 or 1 to 2, and so we update the vertices labeled by 1 in the open neighborhood of v, labeling them with 0; this implies that we have to update the GainFactor value of v, of its open neighborhood and of its open neighborhood of second level $N_2(v)$ because processing v implies a possible update of all the label of $N[v]$ and this implies the update of the closed neighborhood of second level.

We define the open neighborhood of second level of a vertex v as the union of set of vertices in the adjacency list of the vertices in the adjacency list of v, i.e. $N_2(v) = \{x : u \in N(v) \wedge e \in E \wedge e = (u, x)\}$ and the closed neighborhood of second level as $N_2[v] = N_2(v) \cup N[v]$.

Experimentally, even for graphs with connectivity 0.1, the closed neighborhood of second level may be equal to the entire set of vertices, so instead of computing the two sets $N[v]$ and $N_2[v]$ it is better to update the GainFactor of all the vertices; if we keep a list of all the vertices with positive GainFactor, we update only the vertices in this list, and as the GainFactor can not increase, this implies that at every step the list will became shorter.

At time $t = 0$ each vertex is labeled with 1 in the initialization routine. This guarantees us that at each step, we are dealing with a Roman Dominating Function.

Then we compute the GainFactor for all the vertices, and maintain a list A of the positive ones.

At each iteration, we select from the list A, the vertex v with greater GainFactor, label it with 2, and label with 0 all the vertices in its open neighborhood previously labeled by 1; then we update the GainFactor of the vertices in the list A and eliminate from the list A the vertex v and the vertices with negative GainFactor. This until the list A is not empty.

Finally, we check if a vertex v, labeled with 0 or 1, has in its neighborhood one or more vertices u_i labeled by 1; if this is true, we label v with 2 and u_i with 0.

The complexity of this heuristic is $O(V) \cdot O(V + E)$ time, because the most expensive operation is to update the GainFactor ($O(V + E)$ time) of all the vertices ($|V|$).

3.3 Variations of the GainFactor Heuristic

After, we implemented some variations on the GainFactor Heuristic:

- *Random Selection.* When we had to select the vertex with greater GainFactor in the GainFactor Heuristic, if there were two or more such vertices, we simply selected the first. We tried other methods of selection, but the results were more or less the same, so we chose to select the first vertex to avoid more computations and discard these variations.
- *Varying Initial Point.* In this variation, we run the GainFactor Heuristic multiple times; if we have n vertices, we run the GainFactor Heuristic

n times, each time forcing the vertex v_i to be labeled with 2. This variation gives better solutions, but has a drawback: often the iterations select the vertices in the same order, thus wasting computational time.

- *Sequence Control.* In this variation, we tried to solve the drawback of the previous; for each iteration we maintain an auxiliary list, containing the sequence of vertices' selection. If two sequence are equals, then discard the last selection and select another vertex. Theoretically, this variation has to give the better solutions, but is quite time-consuming, and it is infeasible for graphs with $|V| > 500$.

3.4 Conclusions

We tested the heuristics over our dataset. As we can see in Table 5.1, the Simple Heuristic give solutions that are worst than the GainFactor Heuristic, and the gap increase proportionally as the number of vertices.

In Table 5.2 we can see that all the solutions are below the known upper-bounds, see Section 3.2 in Chapter 1.

For generic graphs, the stricter upper-bound is:

$$\gamma_R(G) \leq \left\lfloor n \cdot \frac{2 \cdot \ln(1 + \delta(G)) - \ln 4 + 2}{1 + \delta(G)} \right\rfloor \tag{5.1}$$

Finally in Table 5.3 we can see how good do the various heuristics on some grid graphs; the Simple Heuristic is always the worst, while the GainFactor Heuristic with Varying Initial Point, although is the better heuristic, is always over the new Upper-Bound proven before.

Graph	$\gamma_{SH}(G)$	$\gamma_{GFH}(G)$
Random, 150 vertices, $p = 0.1$	41	29
Random, 150 vertices, $p = 0.2$	23	17
Random, 150 vertices, $p = 0.3$	18	13
Bipartite, 100 + 100 vertices, $p = 0.1$	69	52
Bipartite, 100 + 100 vertices, $p = 0.2$	51	33
Bipartite, 100 + 100 vertices, $p = 0.3$	34	22
Random, 2000 vertices, $\Delta(G) = 6$	1109	785
Random, 2000 vertices, $\Delta(G) = 11$	829	532
Random, 2000 vertices, $\Delta(G) = 21$	570	335
Random, 5000 vertices, $\Delta(G) = 6$	2761	1954
Random, 10000 vertices, $\Delta(G) = 6$	5540	3913
Random, 10000 vertices, $\Delta(G) = 11$	4156	2622
Random, 10000 vertices, $\Delta(G) = 21$	2948	1690

Table 5.1: Simple Heuristic vs GainFactor Heuristic.

Graph	$\gamma_{GFH}(G)$	Eq. (5.1)
Random, 100 vertices, $p = 0.1$, $\delta(G) = 4$	27	76
Random, 200 vertices, $p = 0.2$, $\delta(G) = 26$	18	53
Bipartite, 100 + 100 vertices, $p = 0.1$, $\delta(G) = 3$	52	169
Bipartite, 200 + 200 vertices, $p = 0.05$, $\delta(G) = 3$	109	338
Random, 450 vertices, $p = 0.05$, $\delta(G) = 12$	70	198
Random, 700 vertices, $p = 0.1$, $\delta(G) = 51$	44	114
Bipartite, 350 + 350 vertices, $p = 0.3$, $\delta(G) = 76$	32	45
Random, 850 vertices, $p = 0.3$, $\delta(G) = 218$	18	44
Random, 1000 vertices, $p = 0.1$, $\delta(G) = 70$	48	128
Random, 2000 vertices, $\Delta(G) = 21$, $\delta(G) = 4$	335	1533
Random, 5000 vertices, $\Delta(G) = 21$, $\delta(G) = 4$	847	3832
Random, 10000 vertices, $\Delta(G) = 21$, $\delta(G) = 5$	1690	6995

Table 5.2: GainFactor Heuristic vs Theoretical Upper-Bound.

Grid Graph	$\gamma_{SH}(G)$	$\gamma_{GFH}(G)$	$\gamma_{GFH+}(G)$	$\gamma_2^*(G)$
$G_{10,10}$	84	54	52	48
$G_{10,20}$	174	105	103	92
$G_{15,15}$	198	127	115	102
$G_{20,20}$	364	205	202	176
$G_{30,20}$	554	312	303	260

Table 5.3: A Comparison over Grid Graphs. Simple Heuristic vs GainFactor Heuristic vs GainFactor Heuristic with Varying Initial Point vs the New Upper-bound.

4 Genetic Algorithms

A *Genetic Algorithm* is a search heuristic that use the process of natural selection as a template, and is used to generate approximate solutions to optimization and search problems. They are inspired by natural evolution, such as selection, crossover, mutation and inheritance.

Genetic algorithms are used in every field of human studies, like bioinformatics, engineering, computational science, chemistry, economics, physics, ...

In a genetic algorithm there is a set of candidate solutions (called *population of individuals*) that are evolved toward better solutions. Each individual has a *genetic representation*, i.e. a set of properties that can be altered; usually are used binary strings (or arrays of bits) of $0, 1$ values.

Because usually all the individuals' representations have the same length, it is simple to operate on pair of individuals, for example using a crossover operation. For our problem, the Roman Domination Problem, an element of the population represent a candidate solution, and a position of the array represent a vertex of the graph.

4.1 Fitness Function

Initially the population is randomly generated and at each iteration, a new generation is created. A *fitness function* is used to evaluate the goodness of each individual, and the most fit individuals are randomly chosen from the current generation and are recombined and possibly mutated to form a new generation. The genetic algorithms terminate usually if it has been reached a prefixed level of fitness or it has passed a prefixed number of generation.

For the Roman Domination Problem, the fitness function is computed

using the Roman Dominating Function:

$$f_f = f_{RD} = 2 * |V_2| + 1 * |V_1| + 0 * |V_0| = 2 * |V_2| + |V_1|$$

where, as we already know, if $G(V, E)$ is a graph and f_{RD} a partitioning of the vertices, V_i is the set of vertices labeled by i, where $i = 0, 1, 2$; thus the fitness function is the sum of the labels of every vertex.

To do so, however, we need a *transformation function* between the $0, 1$ values of the arrays of bits and the $0, 1, 2$ values of the Roman Dominating Function. Given an enumeration of the vertices, the transformation function works this way:

1. Label with 2 every vertex corresponding to a position in the array with 1;
2. Label with 0 every vertex adjacent to a vertex labeled by 2;
3. Label with 1 every other vertex.

We need also an inverse of the transformation function:

1. Set 1 on every position corresponding to a vertex labeled by 2;
2. Set 0 on the other vertices.

This work because knowing the vertices labeled by 2 gives a unique Roman Dominating Function weakly non–redundant, i.e. every vertex v labeled by 1 has no vertices labeled by 2 in its adjacency list.

Given an array of bits, representing an individual of the population, to compute its fitness value we use the transformation function, and then compute the sum of all the labels.

4.2 Genetic Operators

During each generation, a sub-set of the population is selected to produce a new generation. The modality of selection can vary from a purely ramdom selection to a purely deterministic selection, based only on the fitness of the individuals. Usually the fitness of an individual increase the probability of such individual to be chosen, i.e. fitter individuals are more likely to be selected.

Once we have selected the individuals, we can use two kind of genetic operators: the crossover and the mutation.

For the crossover operator we need at least two parents. The crossover can be with k points or uniform:

- k-point crossover: we select randomly k point in the array of bits, and then change, alternatively, some portion of the array;
- Uniform crossover: this method enables the individuals to contribute at the bit level, i.e. the offspring has approximately half of the genes from first parent and the other half from second parent.

With the mutation operator, we alter one or more bits of an individual, and by doing so, we introduce some diversity in the population from a generation to the next. The probability of mutation is usually very low, because if it is set too high, then the genetic search will turn in a random search. Usually is used the single point mutation, where a random bit is chosen and with probability $p = 1/l$ is inverted, where l is the length of the array.

There can be many variation to genetic algorithms, but the most widely used is the *elitist selection*, in which some of the better individuals of a generation are allowed to continue to the next generation without alterations.

4.3 Methodology

We tried many variations of genetic algorithms, and also studied a new type of crossover with a *mutant virus*, that acts as a non–purely random mutation operator, see [15]:

1. 1-Point Crossover with no-Mutation;
2. 1-Point Crossover with Mutation;
3. 5-Point Crossover with no-Mutation;
4. 5-Point Crossover with Mutation;
5. Uniform Crossover with no-Mutation;
6. Uniform Crossover with Mutation;
7. Uniform Crossover with Virus.

Every algorithm needs some parameters, some are common to all, like the population's dimension, the number of generations and the initialization of the individuals, i.e. how many positions will be set to 1. For the Uniform Crossover with Virus we need to set a supplementary array of $2 + \lceil log_2|V| \rceil$ bits for each individual.

We decided to set the following parameters: the number of individual is equal to the number of vertices, the number of generations is 100 and the individuals will have 10% of 1. We use elitism to preserve only the best individual of a generation.

Uniform Crossover with Virus

A more detailed explanation is needed for the Uniform Crossover with Virus. The parents are chosen among the population, based on the fitness, the reproduction uses the uniform crossover, however the virus reproduce in a way of its own.

As we said, for the virus variant the array length is $|V|$, plus 2 control bits (the activation state) and $\lceil \log_2 |V| \rceil$ bits that represent the virus. The virus identify exactly a bit in the array of the individual; if $|V|$ is not a power of 2 then the virus may represent a bit outside of the array. In this case, the virus has no effect.

The virus is active only if the two control bits are set to 1. We can divide the population in three sub-set:

- healthy individuals, if both control bits are set to 0;
- healthy carriers, if the control bits have different values;
- sick individuals, if both control bits are set to 1.

When the virus is active, i.e. the individual is sick, the virus identify a position in the array, that are set to 1. If two individuals, both healthy carriers, reproduce, then the offsprings will be sick with probability 25%.

During the initialization, the virus is randomly generated with uniform distribution, while the control bits are set to 1 with probability $p_1 = 10\%$. The virus reproduction is slightly different from the uniform crossover; in fact in the uniform crossover, if in a given position, the parents have the same value, the the two children will have the same value, while if the values differs, one child will have the position set to 1 and the other to 0. For the virus, if the parents have the same value, the children will have that value, but if the values differ, then the children will have a random value with probability $p_2 = 50\%$.

The control bits will reproduce too; if c_1, c_2 are a child's control bits, and c_{11}, c_{12} of one parent and c_{21}, c_{22} of the other, we will have that c_1 will be randomly chosen between c_{11}, c_{12}, while c_2 between c_{21}, c_{22}; lastly, if both c_1 and c_2 are set to 0, then c_1 will be set to 1 with probability $p_3 = 16.6\%$.

Graph (%)	$\|V\|$	p	$\gamma_{UC-V}(G)$	$\gamma_{GFH}(G)$
Random	50	1	33	32
Random	50	2	36	35
Planar	50		8	8
Bipartite	$50 + 50$	1	68	61
Bipartite	$50 + 50$	2	66	60
Planar	100		12	11
Random	150	1	113	95
Random	150	2	93	85
Planar	150		16	11
Bipartite	$100 + 100$	1	145	121
Bipartite	$100 + 100$	2	147	121
Planar	200		21	18
Random	250	1	185	138
Random	250	2	139	105
Planar	250		26	21
Bipartite	$200 + 200$	1	310	242
Bipartite	$200 + 200$	2	260	182

Table 5.4: A Comparison between Genetic Algorithm with Virus and GainFactor Heuristic.

Grid Graph	$\gamma_{UC-V}(G)$	$\gamma_{GFH}(G)$	$\gamma_2^*(G)$
$G_{10,10}$	60	54	48
$G_{10,15}$	91	80	70
$G_{10,20}$	129	105	92
$G_{20,20}$	271	205	176
$G_{30,20}$	422	312	260

Table 5.5: A Comparison over Grid Graphs between Genetic Algorithm with Virus, GainFactor Heuristic and the New Upper-bound.

4.4 Conclusions

The tests used the dataset we generated. For each graph, we execute 3 run of every genetic algorithm, and took the best results. In the Table 5.4 there is a comparison between the performance of the genetic algorithm with virus and the GainFactor Heuristic for some random, bipartite and planar graphs; as we can see, although the genetic algorithm with virus has a good performance, the GainFactor Heuristic has a better one. The same thing is true for grid graphs, see Table 5.5.

Bibliography

[1] S. Alanko, S. Crevals, A. Isopoussu, P. Ostergard, V. Pettersson, *Computing the domination number of grid graphs*, The Electronic Journal of Combinatorics, 18 (2011).

[2] R. Albert, A-L. Barabási, *Statistical mechanics of complex networks*, Rev. Mod. Phys., volume 74, pp. 47 – 97, 2002.

[3] N. Alon, J.H. Spencer, *The Probabilistic Method*, Wiley, New York, 1992.

[4] J. Arquilla, H. Fredricksen, *"Graphing" an optimal grand strategy*, Military Operations Research, Fall 1995.

[5] A-L. Barabási, R. Albert, *Emergence of scaling in random networks*, Science, volume 286, pp. 509–512, 1999.

[6] B. Bollobás, *Random Graphs*, Cambridge University Press, 2001.

[7] J. A. Bondy, U. S. R. Murty, *Graph Theory with Applications*, North Holland, 1976.

[8] E.W. Chambers, W. Kinnersley, N. Prince, D.B. West, *Extremal problems for Roman domination*, SIAM Journal on Discrete Mathematics, 23 (3) (2009), pp. 1575-1586.

[9] T. Y. Chang, E. O. Hare, *Domination numbers of complete grid graphs, I*, Ars Combinatoria, 38 (1994), pp. 97–11.

[10] J. Chen, L. A. Kanj, G. Xia, *Improved Parameterized Upper Bounds for Vertex Cover*, volume 4162, pp. 238–249. Springer Berlin Heidelberg, 2006.

[11] E.J. Cockayne, P.A. Dreyer, Jr., S.M. Hedetniemi, S.T. Hedetniemi, *Roman domination in graphs*, Discrete Mathematics, 278 (2004), pp. 11–22.

[12] E.J. Cockayne, P.J.P. Grobler, W.R. Gründlingh, J. Munganga, J.H. van Vuuren, *Protection of a Graph*, Utilitas Mathematica, 67 (2005), pp. 19–32.

[13] V. Currò, V. Cutello, S.M. Nolassi, *Heuristics for Roman Domination Problem*, preprint.

[14] V. Currò, V. Cutello, S.M. Nolassi, *The Roman Domination Problem on Grid Graphs*, In Middle-European Conference on Applied Theoretical Computer Science (MATCOS- 13), 2013.

[15] V. Cutello, F. Pappalardo, *The Hitting Set Problem and Evolutionary Algorithmic Techniques with ad-hoc Viruses (HEAT-V)*, GECCO 2003, pp. 1570–1571

[16] S. Davis, R. Impagliazzo, *Models of greedy algorithms for graph problems*, Algorithmica, 54(3), pp. 269 – 317, 2009.

[17] G. Di Battista, P. Eades, R. Tamassia, I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall PTR, 1998.

[18] R. G. Downey, M. R. Fellows, C. McCartin, F. Rosamond, *Parameterized approximation of dominating set problems*, Information Processing Letters, 109(1), pp. 68 – 70, 2008.

[19] P.A. Dreyer, Jr., *Applications and variations of domination in graphs*, Ph.D. Thesis, Rutgers University, October 2000.

[20] M.H. El-Zahar, S.M. Khamis, Kh.M. Nazzal, *On the domination number of the cartesian product of the cycle of length n and any graph*, Discrete Applied Mathematics, 155(4), pp. 515 – 522, 2007.

[21] O. Favaron, H. Karami, R. Khoeilar, S.M. Sheikholeslami, *On the Roman domination number of a graph*, Discrete Mathematics 309 (10) (2009).

[22] X. Fu, Y. Yang, B. Jiang, *Roman domination in regular graphs*, Discrete Mathematics, 309(6), pp. 1528 – 1537, 2009.

[23] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the theory of NP-completeness*, W.H. Freeman, San Francisco, 1979.

[24] E. N. Gilbert, *Random Graphs*, volume 30, pp. 1141–1144. JSTOR, 1959.

[25] A. Gibbons, *Algorithmic graph theory*, chapter 5. Cambridge University Press, 1985.

[26] D. Goncalves, A. Pinlou, M. Rao, S. Thomasse, *The domination number of grids*, SIAM Journal on Discrete Mathematics, 25 (3) (2011), pp. 1443–1453.

[27] M. Göös, J. Suomela, *No sublogarithmic-time approximation scheme for bipartite vertex cover*, In Proceedings of the 26th International Conference on Distributed Computing, pp. 181–194, 2012.

[28] S. Gravier, M. Mollard, *On domination numbers of cartesian product of paths*, Discrete Applied Mathematics, 80, pp. 247 – 250, 1997.

[29] F. Harary, T.W. Haynes, *Conditional graph theory IV: Dominating sets*, Utilitas Math. 48 (1995), pp. 179–192.

[30] T. W. Haynes, S. T. Hedetniemi, P. J. Slater, *Fundamentals of domination in graphs*, Marcel Dekker, New York, 1998.

[31] T. W. Haynes, S. T. Hedetniemi P. J. Slater, *Domination in graphs: Advanced topics*, Marcel Dekker, New York, 1998.

[32] M. A. Henning, *A characterization of Roman trees*, Discussiones Mathematicae Graph Theory, 22 (2) (2002), pp. 325 – 334.

[33] M. A. Henning, *Defending the roman empire from multiple attacks*, Discrete Mathematics, 271(1 - 3), pp.101 – 115, 2003.

[34] M. A. Henning, S.T. Hedetniemi, *Defending the roman empire - a new strategy*, Discrete Mathematics, 266, pp. 239 – 251, 2003.

[35] J. E. Hopcroft, R. M. Karp, *An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs*, SIAM Journal on Computing, volume 2(4), pp. 225–231, 1973.

[36] J. Hopcroft, R. Tarjan, *Efficient planarity testing*, Journal of the ACM, volume 21(4), pp. 549–568, 1974.

[37] C.F. de Jaenisch. *Trait des Applications de l'Analyse Mathematique au Jeu des Echecs.* Petrograd, 1862.

[38] M. Kaufmann, D. Wagner, *Drawing graphs: methods and models*, Springer-Verlag, 2001.

[39] D. König, *Gráfok és alkalmazásuk a determinánsok és a halmazok elméletére*, Matematikai és Természettudományi Értesíto?, pp. 104–119, 1916.

[40] M. Liedloff, T. Kloks, J. Liu, S.-L. Peng, *Efficient algorithms for Roman domination on some classes of graphs*, Discrete Applied Mathematics, 156 (18) (2008), pp. 3400-3415.

[41] C-H. Liu, G. J. Chang, *Upper bounds on roman domination numbers of graphs*, Discrete Mathematics, 2012.

[42] M. E. J. Newman, D. J. Watts, S. H. Strogatz, *Random graph models of social networks*, Proceedings of the National Academy of Sciences of USA, volume 99, pp. 2566–2572, 2002.

[43] T. Nieberg, *Independent and Dominating Sets in Wireless Communication Graphs*, University of Twente, 2006.

[44] S.M. Nolassi, *Algoritmi Euristici per il Problema della Dominazione Romana*, Ph.D. Thesis, Università degli Studi di Catania, Novembre 2013.

[45] A. Pagourtzis, P. Penna, K. Schlude, K. Steinhofel, D.S. Taylor, P. Widmayer, *Server Placements, Roman Domination and other Dominating Set Variants*, Proceedings of the IFIP 17th World Computer Congress - TC1 Stream / 2nd IFIP International Conference on Theoretical Computer Science: Foundations of Information Technology in the Era of Networking and Mobile Computing, 2002, pp. 280–291.

[46] P. Pavlic, J. Zerovnik, *Roman domination number of the Cartesian products of paths and cycles*, Institute of Mathematics, Physics and Mechanics, Preprint series 49 (2011).

[47] S. Pemmaraju, S. Skiena, *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Cambridge University Press, 2003.

[48] A. Poghosyan, *The Probabilistic Method for Upper Bounds in Domination Theory*, PhD Thesis, University of the West of England, January 2010.

[49] C. S. ReVelle, K. E. Rosing, *Defendens Imperium Romanum: A classical problem in military strategy*, American Mathematical Monthly, 107 (2000), pp. 585–594.

[50] W. Shang, X. Hu, *The roman domination problem in unit disk graphs*, Computational Science, volume 4489. Springer Berlin Heidelberg, 2007.

[51] M. Soltys, A. Fernandez, *A linear-time algorithm for computing minimum vertex covers from maximum matchings*, In press, 2012.

[52] X. Song, W. Shang, *Roman domination in a tree*, Ars Combinatoria, 98 (2011), pp. 73–82.

[53] I. Stewart, *Defend the Roman Empire!*, Scientific American, 281(6) (1999), pp. 136–139.

[54] F. Wang, E. Camacho, K.Xu, *Positive influence dominating set in online social networks.*, Combinatorial Optimization and Applications. Springer Berlin Heidelberg, 2009.

[55] S. Wasserman, K. Faust, *Social Network Analysis*, Cambridge University Press, 1994.