Exact Treewidth and Exponential-time Algorithms for Defense-like Domination Problems

Mats Veldhuizen

February 23, 2020

1 Defense-like Domination Problems

1.1 Introduction

In this thesis proposal we will set out to construct exact exponential-time and treewidth algorithms for a subset of defense-like domination problems. Defense-like domination problems are variants of the Dominating Set problem in which we try to defend a graph from a sequence of attacks. These problems derives from the idea of defending the Roman empire from attacks.

Our definition of defense-like domination problems is a generalization of the Weak Roman Domination problem, introduced by Henning et al. [9]. This problem originates from the Roman Domination problem, introduced by Cockayne et al. [6]. Roman Domination is a variant of Dominating Set. For Dominating Set the currently best known exact exponential-time algorithms are the results from Iwata [10]. These results are an $\mathcal{O}^*(1.4864^n)$ time algorithm using polynomial space and an $\mathcal{O}^*(1.4689^n)$ time algorithm using exponential space. There also exists a treewidth algorithm for Dominating Set running in $\mathcal{O}^*(3^k)$ time (for graphs of treewidth k), by van Rooij et al. [14]. In the Roman Domination variant the rules are inspired by a decree from emperor Constantine of the roman empire. These rules stated that a vertex can protect itself when it has 1 legion present and it can protect all its neighbours when it has 2 legions present. This is because Constantine required any location to have 2 legions present in order to allow one of these legions to move to a nearby location. For Roman Domination there also exist exact exponential-time algorithms. The best known exact exponential-time algorithms are an $\mathcal{O}^*(1.5673^n)$ time algorithm using polynomial space and an $\mathcal{O}^*(1.5014^n)$ time algorithm using exponential space, by Shi et al. [13]. These results are actually just applying the results from Nederlof et al. [12] to this problem. The same treewidth algorithm for Dominating Set can also be applied to Roman Domination with a minor adjustment, yielding an $\mathcal{O}^*(4^k)$ time algorithm. Weak Roman Domination relaxes the decree from Constantine in the following way. In the Weak Roman Domination problem we also allow legions to move to neighbouring locations when this does not result in an undefended vertex, where an undefended vertex is defined as a vertex for which the closed neighbourhood does not contain any vertex with 1 or more legions. For Weak Roman Domination there already exists an $\mathcal{O}^*(2^n)$ time algorithm using exponential space and an $\mathcal{O}^*(2.2279^n)$ time algorithm using polynomial space by Chapelle et al. [5], however to our knowledge no treewidth algorithm exists. Another problem that relates closely to Weak Roman Domination is Secure Domination, introduced by Cockayne et al. [7]. Secure Domination is essentially Weak Roman Domination, with the exception that we cannot place 2 legions on one vertex, but at most 1. For this problem there exists a branching algorithm by Burger et al. [4] for which the run time is $\mathcal{O}^*2^{n-s-r}$, where s and r are the number of support vertices and redundant vertices of the graph respectively, as well as a binary programming algorithm by Burger [3]. This last result has recently been improved by Burdett et al. [1].

We will generalize the definition of Weak Roman Domination in every reasonable way. The resulting definition will be our definition of defense-like domination problems. This general definition will encapsulate several known variants. First we introduce a number of parameters, which will lead us to our parameterized problems (Section 1.3.1). To our knowledge the only instances of our parameterized problems that have been studied are Weak Roman Domination and Secure Domination, however a variant in which the number of turns is infinite has been studied more widely as Eternal Domination [8]. However to our knowledge this problem is not in NP, since checking a solution means checking an infinite number of possible moves. Therefore this problem is also not included in our parameterized variants. In addition to the parameterized variants we will also look at variants with different defense strategies in Section 1.3.2. All of these strategies have been defined in earlier works, but to our knowledge no attempts at either exact exponential-time or treewidth algorithms have been made for these variants.

In the rest of this section we will first give the general definition defense-like domination problems (Section 1.2). We will follow this by more concrete definitions of the possible variants in Section 1.3.

1.2 Definitions

In this section we will give the most general definition of defense-like domination problems. In Section 1.3.1 we will use this definition to define a number of more concrete parameterized variants, which will be the variants we will study in this thesis. In order to give this general definition we will first need the following definitions.

For a graph $G = (V, E)$, a guard assignment or guard function is a function $f : V \to \{0, 1, ..., c\}$ where, for every $v \in V$, the value of $f(v)$ is defined as the number of guards on vertex v. The guards serve the same purpose as the legions in Weak Roman Domination, but with a much less Roman term. The capacity c is the maximum number of guards that can be assigned to a single vertex. An attack sequence is defined as $A = \{A_0, A_1, ..., A_k\}$, where $A_i \subset V$ for all $0 \leq i \leq k$. Here A_i represents the set of vertices that are under attack at turn i. We will define the maximum attack size of sequence A by $a = \max_{0 \leq i \leq k}\{|A_i|\}$.

A step is defined by an edge $e = (u, w) \in E$ and represents a single guard g moving from u to w. We say that guard g has moved one step along edge e. This results in a new guard function $f_{u \to w}$ defined by

$$f_{u\to w}(v) = \begin{cases} f(v) + 1 & \text{if } v = w \\ f(v) - 1 & \text{if } v = u \\ f(v) & \text{otherwise} \end{cases}$$

During an attack we can move a subset of our guards of size at most t, at most s steps each in order to defend our graph G. A collection of moves that satisfies these 2 restrictions is called a valid collection of moves.

We say that f can defend A_0, if there exists a valid collection of moves that generates a new assignment f_0, from f, such that $\forall u \in A_0 : f_0(u) \geq 1$. Inductively we say that f can defend A_{i+1} if f can defend A_i and there exists a valid collection of moves generating f_{i+1}, from f_i, such that $\forall u \in A_{i+1} : f_{i+1}(u) \geq 1$. We call f a defense-like dominating function (dld-function for short), if f can defend every attack in the sequence A, for every possible sequence A.

The cost of a dld-function is defined as $cost(f) = \sum_{v \in V} f(v)$. The cost represents the number of guards needed for the assignment. Defense-like domination problems are then defined as follows. We have as input a graph $G = (V, E)$ and as output we will need to find a dld-function f of minimum cost.

1.3 Variants

In this thesis proposal we will look at a number of variations of defense-like domination problems. In particular we will study a couple of parameterized variants (Section 1.3.1), as well as some variants with different defense strategies (Section 1.3.2).

1.3.1 Parameters

In Section 1.2 we gave a general definition for defense-like domination problems. In doing so we defined a number of parameters. These are the following.

- The number of turns, k.
- The maximum attack size, a.
- The number of guards that are allowed to move each turn, t.

- The maximum number of guards per vertex, c.
- The maximum number of steps per guard, s.

Using these parameters we can define the variants we will study. We will always assume the capacity c to be infinite and the rest of these parameters to be 1 unless stated otherwise. e.g. k-TURN DEFENSIVE DOMINATION would be a variant in which c is infinite, so $f : V \to \mathbb{N}$, k is the parameter that can vary and a, t and s are all equal to 1. This results in the following list of parameterized problems.

- k-TURN DEFENSIVE DOMINATION
- a-ATTACK DEFENSIVE DOMINATION
- t-MOVE DEFENSIVE DOMINATION
- c-CAPACITATED DEFENSIVE DOMINATION
- s-STEP DEFENSIVE DOMINATION

Using these definitions we see that we can refer to WEAK ROMAN DOMINATION as simply DEFENSIVE DOMINATION. This is because even though DEFENSIVE DOMINATION would allow any number of guards on a vertex, having more than 2 guards there does not do anything more. Following the same logic DOMINATING SET will be 0-TURN DEFENSIVE DOMINATION and SECURE DOMINATION is 1-CAPACITATED DEFENSIVE DOMINATION. ROMAN DOMINATION is not part of this definition. Furthermore ETERNAL DOMINATION could be referred to as ∞-TURN DEFENSIVE DOMINATION. However we will not be working on this last problem in this thesis.

1.3.2 Strategies

In our general definition of defense-like domination problems (Section 1.2), we talk about moving guards to defend vertices from attack sequences. However, which guards to move and which vertices will be attacked, after we have chosen our guard assignment, is not yet discussed. In this section we will present 3 different problem variants, that handle this. We say that we have an attacking strategist. He or she constructs the attack sequence. We also have a defending strategist. He or she decides at every attack which guards to move in order to defend this attack. We will always assume the defending strategists to be perfect, i.e. he or she will always make the best choice based on the information available. According to [11] there are 2 problem variants regarding the way the attacking strategist chooses and reveals the attack sequence for defense-like domination problems.

1. **Offline:** In the offline version the entire attack sequence is chosen and revealed beforehand. The defending strategist can use its knowledge of the entire sequence to find a sequence of moves to defend against this.
2. **Online:** In the online version the attack sequence is chosen but not revealed beforehand and the defending strategist has to move the guards based only on its knowledge of the current attack.

There is also a third option in which the attacking strategist does not choose the attack sequence beforehand but chooses its next attack based on the result of the previous attacks. This does not change our problem at all since our assignment of guards has to be able to defend every possible attack sequence as also stated in [11]. This gives us two different strategic variants for defense-like domination problems.

So far we have been assuming that we have a perfect defending strategist that can make the best decisions in the defense against attacks. If we drop this requirement we get the third and last strategic variant, described in [2].

3. **Foolproof:** In the foolproof variant we require the assignment of guards to be such that at every attack any successful defensive move will result in a successful defense of the entire sequence. In other words it doesn't matter which defensive choices we make at every attack, because each defensive choice is just as effective.

Unless explicitly stated the reader may always assume we are referring to the online variant of a problem.

2 Goal

2.1 Research Questions

The goal of this thesis is to find new exact exponential-time algorithms and/or treewidth algorithms for a subset of the problems defined in Section 1.3. We will also explore lower bounds for these problems, if time permits. Initial research into these variants has led us to believe that the most interesting questions to research are the following.

1. What is the fastest treewidth algorithm we can construct for Defensive Domination (Weak Roman Domination)?
2. What is a non-trivial (faster than trivial enumeration) exact exponential-time algorithm for k-Turn Defensive Domination (either capacitated or not capacitated)?
3. Can we extend the algorithm from Question 1 to a version for 1-Capacitated Defensive Domination (Secure Domination)?
4. What is the fastest treewidth algorithm we can construct for k-Move Defensive Domination for each $k \in \mathbb{N}$?
5. What is a non-trivial exact exponential-time algorithm for k-Move Defensive Domination? We expect to be able to modify the algorithm from Chapelle et al. [5] for this problem.
6. Can we extend our algorithms to work for the foolproof variant?
7. What is a non-trivial exact exponential-time algorithm for k-Step Defensive Domination?

2.2 Methodology

To answer the research questions from Section 2.1 we will employ the following methodology. For both treewidth questions (Questions 1 and 4), we will look for a treewidth algorithm similar to the classical result for Dominating Set. This is achieved by finding the correct state set and finding the right decisions for every node variant in a nice tree decomposition. To improve on these results we will apply the techniques from van Rooij [14]. Question 3 will probably directly follow from the result from Question 1, so this result will be constructed accordingly.

For Question 2 we will look for any algorithm we can find that runs in $\mathcal{O}^*(c^n)$ where c is a constant and n is the number of nodes in the input graph. We will search for the lowest possible c by using bounds and dynamic programming techniques.

We expect to be able to answer Question 5 using the same method as the exact exponential algorithm from Chapelle et al. [5], since increasing the number of moves seems to reduce the complexity of the problem. We will also look for any improvement we can make, due to this reduction in complexity. We do not expect to find much here so we will not spend too much time on this.

We believe that most of the algorithms we construct will also work for the foolproof variant, possibly with slight modification, since this variation seems to reduce complexity as well. We hope to find improvements on these results in the foolproof variant, because of this same reason.

The last question we will treat the same as Question 2. We expect this problem to be related to the Distance-r Dominating Set problem, so we will incorporate results for this problem in our attempts. We expect to find a result for this problem by combining the algorithm from Chapelle et al. [5], with Distance-r Dominating Set.

2.3 Preliminary Results

Initial research has led us to the following preliminary results for our research questions (for now without proof). Thus far we have found some results for Questions 1 and 2. The results we found for 2 seem to work for every possible instance of Defensive Domination for which we can check a solution in polynomial time.

1. There exists an $\mathcal{O}^*(9^t)$ time algorithm for Defensive Domination for graphs of treewidth t.
2. There exists an $\mathcal{O}^*(4^n)$ time algorithm for k-Turn Defensive Domination for graphs with n vertices, for each fixed k.
3. There exists an $\mathcal{O}^*(8^t)$ time algorithm for 1-Capacitated Defensive Domination for graphs of treewidth t.

2.4 Planning

We will answer the questions from Section 2.1 in the order they are presented there. Since we already have some results for the first 2 questions, the first 3-5 weeks will be spend working out these results. We expect the third question to follow quite directly from the first so we will include this result in the same time. Following this we will spend the next 5-7 weeks on the fourth (and fifth) question. Lastly we will look at the last two research questions in 4-6 weeks each.

References

[1] R. Burdett and M. Haythorpe. An improved binary programming formulation for the secure domination problem. 2019, arXiv:1911.02198.

[2] A. P. Burger, E. J. Cockayne, W. R. Gründlingh, C. M. Mynhardt, J. H. van Vuuren, and W. Winterbach. Infinite order domination in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 50:179–194, 2004.

[3] A. P. Burger, A. P. De Villiers, and J. H. van Vuuren. A binary programming approach towards achieving effective graph protection. In *ORSSA*, pages 19–30, 2013.

[4] A. P. Burger, A. P. De Villiers, and J. H. Van Vuuren. Two algorithms for secure graph domination. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 85:321–339, 2013.

[5] M. Chapelle, M. Cochefert, J. F. Couturier, D. Kratsch, R. Letourneur, M. Liedloff, and A. Perez. Exact algorithms for weak Roman domination. *Discrete Applied Mathematics*, 248:79–92, 2018.

[6] E. J. Cockayne, P. A. Dreyer, S. M. Hedetniemi, and S. T. Hedetniemi. Roman domination in graphs. *Discrete Mathematics*, 278(1-3):11–22, 2004.

[7] E. J. Cockayne, P. J. P. Grobler, W. R. Grundlingh, J. Munganga, and J. H. van Vuuren. Protection of a Graph. *Utilitas Mathematica*, 67:19–32, 2005.

[8] W. Goddard, S. M. Hedetniemi, and S. T. Hedetniemi. Eternal Security in Graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 52:1–12, 2005.

[9] M. A. Henning and S. T. Hedetniemi. Defending the Roman Empire - A new strategy. *Discrete Mathematics*, 266(1-3):239–251, 2003.

[10] Y. Iwata. A Faster Algorithm for Dominating Set Analyzed by the Potential Method. In D. Marx and P. Rossmanith, editors, *Parameterized and Exact Computation*, pages 41–54, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[11] W. F. Klostermeyer and C. M. Mynhardt. Protecting a graph with mobile guards. *Applicable Analysis and Discrete Mathematics*, 10(1):1–29, 2016.

[12] J. Nederlof and J. M. M. van Rooij. Inclusion/Exclusion Branching for Partial Dominating Set and Set Splitting. In V. Raman and S. Saurabh, editors, *Parameterized and Exact Computation*, pages 204–215, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[13] Z. Shi and K. M. Koh. Counting the Number of Minimum Roman Dominating Functions of a Graph. 2014, 1403.1019.

[14] J. M. M. van Rooij, H. L. Bodlaender, and P. Rossmanith. Dynamic Programming on Tree Decompositions Using Generalised Fast Subset Convolution. In A. Fiat and P. Sanders, editors, *Algorithms - ESA 2009*, pages 566–577, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.