DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY MUNICH

Master Thesis

On the parameterized Complexity of Semitotal Domination on Graph Classes

Lukas Retschmeier

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY MUNICH

Master Thesis

On the parameterized Complexity of Semitotal Domination on Graph Classes

Über die Parametrisierte Komplexität des Problems der halbtotalen stabilen Menge auf Graphklassen

Author:	Lukas Retschmeier
Supervisor:	Prof. Debarghya Ghoshdastidar
Advisor:	Prof. Paloma Thomé de Lima
Submission Date:	December 19, 2022

I confirm that this master thesis is my own work and I have documented all sources and material used.

København S
December 19, 2022

Lukas Retschmeier

Acknowledgments

CONTENTS

ABSTRACT

Abstract

For a graph $G = (V, E)$, a set D is called a *semitotal dominating set*, if D is a dominating set and every vertex $v \in D$ is within distance two to another witness $v' \in D$. The MINIMUM SEMITOTAL DOMINATING SET problem is to find a semitotal dominating set of minimum cardinality. The semitotal domination number $\gamma_{t2}(G)$ is the minimum cardinality of a semitotal dominating set and is squeezed between the domination number $\gamma(G)$ and the total domination number $\gamma_t(G)$. Given a graph $G = (V, E)$ and a positive integer k, the SEMITOTAL DOMINATION DECISION problem asks if G has a semitotal dominating set of size at most k.

After the problem was introduced by Goddard, Henning and McPillan in [13], NP-completeness was shown for general graphs [19], *split graphs* [19], *planar graphs* [19], *chordal bipartite graphs* [19], *circle graphs* [22] and *subcubic line graphs of bipartite graphs* [11]. On the other side, there exist polynomial-time algorithms for *AT-free graphs* [22], *graphs of bounded mim-width* [11], *graphs of bounded clique-width* [5], and *interval graphs* [19].

In this thesis, we start the systematic look through the lens of *parameterized complexity* by showing that SEMITOTAL DOMINATING SET is $\omega[2]$-hard for bipartite graphs and split graphs. By applying the techniques proposed in [1] and [12] for DOMINATING SET and TOTAL DOMINATING SET, we are going to construct a $359k$ kernel for SEMITOTAL DOMINATING SET in planar graphs. This result further complements known linear kernels for other domination problems like PLANAR CONNECTED DOMINATING SET, PLANAR RED-BLUE DOMINATING SET, PLANAR EFFICIENT DOMINATING SET, PLANAR EDGE DOMINATING SET, INDEPENDENT DOMINATING SET and PLANAR DIRECTED DOMINATING SET.

Keywords: Domination; Semitotal Domination; parameterized Complexity; Planar Graphs; Linear Kernel

ZUSAMMENFASSUNG

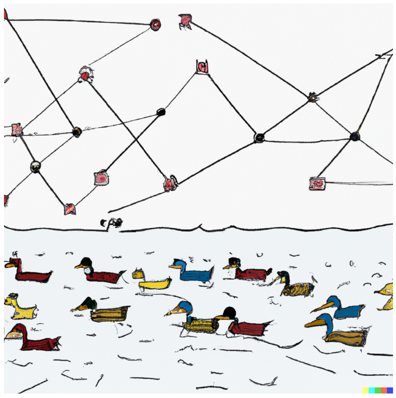
Abstract

Hier kommt noch ein weiterer Abstract rein.

Schlagworte: Stabile Menge; Halbtotale Stabile Menge; Parametrisierte Komplexität; Plättbare Graphen; Linearer Problemkern

CHAPTER 1

TERMINOLOGY AND PRELIMINARIES



"All we have to decide is what to do with the time that is given to us."

J. R. R. Tolkien, *Gandalf* in *Lord of the Rings*

In this chapter, we will introduce the core definitions used throughout this thesis. Most of the definitions of graph theory are taken from [7]. For definitions in the area of *parameterized complexity*, the book written by Cygan et al. [6] gives an excellent introduction. For standard mathematical notation, the reader is referred to any introductory textbook into discrete mathematics (e.g. [25]).

1.1 Graph Theory

If not explicitly stated otherwise, the following definitions are taken from the book *Graph Theory* written by Reinhard Diestel [8].

1.1.1 Basic Terminology

Definition 1.1.1 (Graph). *A simple graph is a pair $G = (V, E)$ of two sets where V denotes the vertices and $E \subseteq V \times V$ the edges of the graph. A vertex $v \in V$ is incident with an edge $e \in E$ if $v \in e$. Two vertices x, y are adjacent, or neighbours, if $\{x, y\} \in E$. By this definition, graph loops and multiple edges are excluded.*

A multigraph is a pair (V, E) of disjoint sets together with a map $E \to V \cup [V]^2$ assigning to every edge either one or two vertices, its ends. Multigraphs can have loops and multiple edges.
We usually denote the vertex set by $V(G)$ and its edge set by $E(G)$.

Unless stated otherwise, we usually consider only *simple graphs*, but the notion of *multigraphs* gets important when we later talk about the *underlying multigraph* of a *D-region decomposition*.

Definition 1.1.2 (Subgraph and Induced Subgraph)**.** *Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. If $V' \subseteq V$ and $E' \subseteq E$ then G' is a subgraph of G. If G is a subgraph of G' and G' contains all the edges to G with both endpoints in $V(G')$, then G' is an induced subgraph of G and we write $G' = G[V(G')]$.*

Definition 1.1.3 (Degrees)**.** *Let $G = (V, E)$ be a graph. The degree $d_G(v)$ (shortly $d(v)$ if G is clear from the context) of a vertex $v \in V$ is the number of neighbors of v. We call a vertex of degree 0 as isolated and one of degree 1 as a pendant. If all the vertices of G have the same degree k, then g is k-regular.*

Definition 1.1.4 (Closed and Open Neighborhoods [3])**.** *Let $G = (V, E)$ be a (non-empty) graph. The set of all neighbors of v is the open neighborhood of v and denoted by $N(v)$; the set $N[v] = N(v) \cup \{v\}$ is the closed neighborhood f v in G. When G needs to be made explicit, those open and closed neighborhoods are denoted by $N_G(v)$ and $N_G[v]$.*

Definition 1.1.5 (isomorphic Graphs)**.** *Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. We call G and G' isomorphic, if there exists a bijection $\phi : V \to V'$ with $\{x, y\} \in E \Leftrightarrow \phi(x)\phi(y) \in E'$ for all $x, y \in V$. Such a map ϕ is called isomorphism.*
If a graph G is isomorphic to another graph h, we denote $G \simeq H$.

Definition 1.1.6 (Paths and Cycles)**.** *A path is a non-empty graph $P = (V, E)$ of the form $V = \bigcup_{i \in [k]} \{x_i\}$ and $E = \bigcup_{i \in [k-1]} \{x_i x_{i+1}\}$ where the x_i are distinct. The vertices x_0 and x_k are linked by P and are called the ends of P. The length of a path is its number of edges and the path on n vertices is denoted by P_n. We refer to a path P by a natural sequence of its vertices: $P = x_0 x_1 ... x_k$. Such a path P is a path between x_0 and x_k, or a x_0, x_k-path. If $P = x_0 ... x_k$ is a path and $k \geq 2$, the graph with vertex set $V(P)$ and edge set $E(P) \cup \{x_k x_0\}$ is a cycle. The cycle on n vertices is denoted as C_n. The distance $d_G(v, w)$ from a vertex v to a vertex w in a graph g is the length of the shortest path between v and w. If v and w are not linked by any path in G, we set $d_G(v, w) = \infty$. Again, if G is clear from the context, we omit the subscripted G and just write $d(v, w)$ instead.*

1.1.2 Graph Classes

A *graph class* is a set of graphs $\mathfrak{G}$ that is closed under isomorphism that is if $G \in \mathfrak{G}$ and a $H \simeq G$ then $H \in \mathfrak{G}$ as well.

Definition 1.1.7 (Graph Parameters)**.** *Let $G = (V, E)$ be a graph. An independent set of G is a set of pairwise non-adjacent vertices. A clique of G is a set of pairwise adjacent vertices. A vertex cover of G is a subset of vertices containing at least one endpoint of every edge. A dominating set is a subset D of vertices such that all vertices not contained in are adjacent to some vertex in D.*

Graph Class 1 (r-partite)**.** *Let $r \geq 2$ be an integer. A Graph $G = (V, E)$ is called r-partite if V admits a partition into r classes such that every edge has its ends in different classes: Vertices in the same partition class must not be adjacent. A 2-partite graph is called bipartite.*

An r-partite graph in which every two vertices from different partition classes are adjacent is called complete. For the complete bipartite graph on bipartitions $X \uplus Y$ of size m and n, we shortly write $K_{m,n}$.

Graph Class 2 (Complete)**.** *If all vertices of a graph $G = (V, E)$ are pairwise adjacent, we say that G is complete. A complete graph on n vertices is a K_n. A K_3 is called a triangle.*

Graph Class 3 (Chordal)**.** *For a graph $G = (V, E)$, an edge that joins two vertices of a cycle, but is not itself an edge of the cycle is a chord of that cycle.*

Furthermore, we say G is chordal (or triangulated) if each of its cycles of length at least four has a chord. In other words, it contains no induced cycle other than triangles.

Graph Class 4 (Split)**.** *A split graph is a graph $G = (V, E)$ whose vertices can be partitioned into a clique and an independent set.*

Graph Class 5 (Planar)**.** *A plane graph is a pair (V, E) of finite sets with the following properties:*

- *$V \subseteq \mathbb{R}^2$ (Vertices),*
- *Every edge is an arc between two vertices,*
- *different edges have different sets of endpoints, and*
- *The interior of an edge contains no vertex and no point of any other edge*

An embedding in the plane, or planar embedding, of an (abstract) graph G is an isomorphism between G and a plane graph H. A plane graph can be seen as a concrete ***embedding*** *of the planar graph into the "plane" $\mathbb{R}^2$.*

1.2 Computational Complexity Theory

Computational complexity investigates the question of how many computational resources are required to solve a specific problem. We are about to introduce two of the most important classes of problems in classical complexity theory:

The Class P [2]

If we denote **DTIME** as the set of decision problems that are solvable in $\mathcal{O}(n^k)$ time by a deterministic Turing Machine, we can define the class **P** as:

$$\mathrm{P} := \bigcup_{k \in \mathbb{N}} (\mathrm{DTIME}(n^k))$$

The Class NP [2]

A language $L \subseteq \{0,1\}^*$ is in **P** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time Turing Machine M such that for every $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} s.t.\ M(x,u) = 1$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, then we call u a *certificate* for x.

P denotes the class of all problems that are *efficiently solvable* whereas **NP** contains all problems whose solution can efficiently be verified. Note that **P** $\subseteq$ **NP**, but the opposite is unknown.

1.2.1 NP-Completeness

A major discovery in the early 1970s was the fact that some problems in **NP** are *at least as hard as* as any other problem in **NP** by reducing them to each other spanning a whole "web of reductions" [2]. The first results in this area had been published independently by Cook [4] and Levin [23] after Karp [21] had introduced this idea of problem reductions. The Cook-Levin-Theorem [4] states that the BOOLEAN SATISFIABILITY PROBLEM is **NP**-COMPLETE, which implies that one single algorithm for any of these problems would be enough to efficiently solve all of them. For a comprehensive introduction to classical complexity theory, the reader is referred to [2].

Definition 1.2.1 (Reductions, **NP**-hardness and **NP**-COMPLETENESS [2])**.** *We say that a language $A \subseteq \{0,1\}^*$ is polynomial-time Karp reducible to a language $B \subseteq \{0,1\}^*$ (denote $A \leq_p B$) if there is a poly-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $x \in A$ if and only if $f(x) \in B$.*
We say that a problem B is **NP**-HARD *if $A \leq_p B$ for every $A \in$* **NP** *and B is* **NP**-COMPLETE *if additionally $B \in NP$ holds.*

There are thousands of **NP**-COMPLETE problems we do not expect to be solvable in polynomial time. The famous question of whether **P** $=$ **NP** or not is still one of the biggest open questions in mathematics bountied with one million dollars by the *Clay Mathematical Institute* [10]. Most of the domination problems like Dominating Set, Semitotal Dominating Set, Total Dominating Set are **NP**-Complete.

Coping with NP-Completeness Even though we do not expect **NP**-Complete problems to have a polynomial-time algorithm, there are some strategies to cope with them. We can either give up the exactness of a solution to possibly find fast *approximation algorithms* or abandon the search for a polynomial-time algorithm in favor of finding good *Exact Exponential (EEA) Algorithms* instead.

A third technique is using additional structural parameters of a specific problem instance and therefore **restricting the input to special cases**. This idea lead to the development of *Parameterized complexity*.

1.2.2 Definitions in Parameterized Complexity

Introduced by Downey and Fellows [9], parameterized complexity extends the classical theory with a framework that allows a more finely-grained analysis of computationally hard problems. The idea is to measure a problem in terms of input size and an additional (structural) parameter k.

We like to find an algorithm that is only exponential in a function $f(k)$, but polynomial in the instance size. k denotes how difficult the problem is:

If k is small then the problem can still be considered tractable although the underlying **NP**-hard problem counts as intractable in general. Therefore k can be seen as a measure of the difficulty of a given instance. If not marked otherwise, all definitions are taken from [6].

Definition 1.2.2 (Parameterized Problem)**.** *A parameterized problem is a $L \subseteq \Sigma^* \times \mathbb{N}$ (Σ finite fixed alphabet) for an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, where k is called the parameter.*

The size of an instance of an instance (x, k) of a parameterized problem is $|(x, k)| = |x| + k$ where the parameter k is encoded in unary by convention.

1.2.3 Fixed-Parameter Tractability

We say that a problem is *fixed-parameter tractable (fpt)* if problem instances of size n can be solved in $f(k)n^{\mathcal{O}(1)}$ time for some function f independent of n. Like the class **P** can be seen as a notion of *tractability* in classical complexity theory, there is an equivalent in parameterized complexity, which we denote as Fixed-Parameter Tractable (**FPT**) and which we can define the following way:

The Class FPT

A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed-parameter tractable* if there exists an algorithm A (called a *fixed-parameter algorithm*), a computable function $f : \mathbb{N} \to \mathbb{N}$ and a constant c such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathcal{A}$ correctly decides whether $(x, k) \in L$ in time bounded by $f(k) \cdot |(x, k)|^c$. The complexity class containing all fixed-parameter tractable problems is called **FPT**.

1.2.4 Kernelization

A kernelization algorithm is a natural and intuitive way to approach problems and can be seen as a preprocessing procedure that simplifies parts of an instance already before the actual solving algorithm is run. A visualization of this idea can be seen in Figure 1.2. One can introduce *reduction rules* that iteratively reduce the instance until we are left with a small kernel.

Definition 1.2.3 (Kernelization and Reduction Rules). *A kernelization algorithm or kernel is an algorithm $\mathfrak{A}$ for a parameterized problem Q that given an instance (I, k) of Q runs in polynomial time and returns an equivalent instance (I', k') of Q. Moreover, we require that $size_{\mathfrak{A}}(k) \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.*

A reduction rule is a function $\phi : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ that maps an instance (x, k) to an equivalent instance (x', k') such that ϕ is computable in time polynomial in $|x|$ and k.

A reduction rule is sound (or safe) if $(I, k) \in Q \Leftrightarrow (I', k') \in Q$.

We can give a precise definition of the size of the kernel, after a preprocessing algorithm $\mathfrak{A}$ has been executed. $size_{\mathfrak{A}}$ denotes the largest size of any instance I after $\mathfrak{A}$ has been applied. We consider the size to be infinite if it cannot be bounded by a function in k.

Definition 1.2.4 (Output size of a Preprocessing Algorithm). *The output size of a preprocessing algorithms $\mathfrak{A}$ is defined as*

$$\mathrm{size}_{\mathfrak{A}}(k) = \sup\{\left|I'\right| + k' : (I', k') = \mathfrak{A}(I, k), I \in \Sigma^*\}$$

If we bound $\mathrm{size}_{\mathfrak{A}}$ by a polynomial in k, we say that the problem admits a **polynomial kernel**. Analogous, if the size after the reduction is only linear k, we refer to it as a **linear** kernel.

The following Lemma 1.2.1 shows the relation between the complexity class **FPT** and a kernelization algorithm. If we find a kernelization algorithm $\mathfrak{A}$ for a (decidable) problem P, we immediately obtain an fpt algorithm by first running the $\mathfrak{A}$ on an instance I of P in polynomial time. Assuming that P can be solved by an algorithm $\mathfrak{M}$ running in time $g(n)$ we can use the fact that the kernel is bounded by a function $f(k)$ and apply

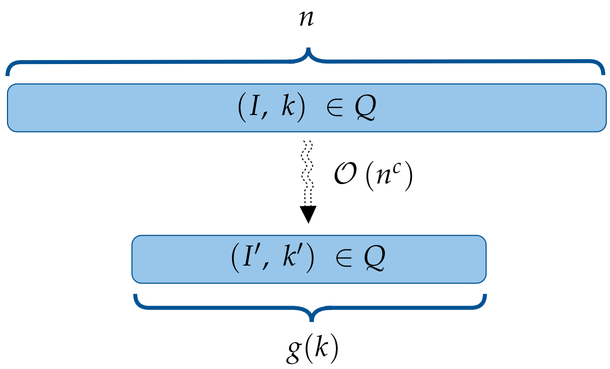


Figure 1.2: *Kernelization: Reducing an instance (I, k) of size n to a smaller instance (I', k') in polynomial time. The size of the kernel is a function $g(k)$ only dependent on k.*

$\mathfrak{M}$ on the kernel resulting in a total running time of the order $\mathcal{O}(g(f(k)) \cdot \text{poly}(n))$ which is fpt. Surprisingly, also the converse is true:

Lemma 1.2.1. *If a parametrized problem Q is* **FPT** *if and only if it admits a kernelization algorithm.*

In **??** we will use this and by explicitly constructing a kernel for Planar Semitotal Dominating Set, we show membership of the problem in **FPT**.

1.2.5 Reductions and Parameterized Intractability

It is natural to ask whether all hard problems are also fixed-parameter tractable It turns out that there is an equivalent Like the class **NP** gives strong evidence must be adjusted

The class **FPT** can be seen as a parameterized equivalent to the class ¶in the classical setting.

As we can see the class **FPT** as some parameterized equivalent for the ¶in the classical settings, we might ask wether there is also a notion of *parameterized intractability*. Are there any problems that are not fixed-p

It turns out that there is whole hierarchy, the so called W-hierarchy.

Before defining these classes, we need a

We can transfer the definitions given in section 1.2.1 to the parametrized setting as well. We will establish a similar notion for *hardness*

Before defining these classes, we need the notion of a *parameterized reduction* that transfers fixed-parameter tractability. These reduction preserve *hardness* in the parameterized setting.

Parameterized Reductions

Definition 1.2.5 (Parameterized Reduction [6, Def 13.1]). *Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ two parameterized problems. A parameter preserving reduction from A to B is an algorithm that, given an* *instance (x,k) of A, outputs an instance (x',k') of B such that:*

- *(x,k) is a yes instance of A **iff** (x',k') is a yes instance of B,*
- *$k' \leq g(k)$ for some computable function g, and*
- *runs in fpt-time $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f.*

The following two Lemmas 1.2.2 and 1.2.3 [6] are crucial for proving parameterized intractability and transfer properties, which we had in the classical setting as well.

Lemma 1.2.2 (Closed under fpt-reductions)**.** *If there is a parameterized reduction from A to B and $B \in$ **FPT**, then $A \in$ **FPT**, too.*

Lemma 1.2.3 (Transitivity)**.** *If there are parameterized reductions from A to B and from B to C, then there is a parameterized reduction from A to C.*

Parameterized Complexity class? ParaNP? Whoe w hierarhcy.

We say a problem is *hard* We are now ready to define two of the most important classes for fpt.-intractability.

complete, if We will omit a comprehensive introduction of the W-hierarchy as it is not required for proofing the

The Classes W[1] and W[2]

CLIQUE is fpt-complete for $W[1]$.
DOMINATING SET is fpt-complete for $W[2]$.

It is strongly believed that **FPT** $\subsetneq$ and therefore, we do not expect the existence of an algorithm solving any $W[i]$-hard problem in fpt time.

CHAPTER 2

ON PARAMETERIZED SEMITOTAL DOMINATION

In computer science, parameterized complexity is a framework for studying the complexity of computational problems, in which the complexity of a problem is not just a function of the size of the input, but also a function of one or more additional parameters that describe the problem instance.

chatGPT, 2022

In connection with various chessboard problems, the concept of domination can be traced back to the mid-1800s. For example, de Jaenosch attempted in 1862 to solve the minimum number of queens required to fully cover an n x n-chessboard [20]. Because of the immense amount of publications related to domination, Haynes, Hedetniemi, and Slater started a comprehensive survey of the literature [14, 15]. 20 years later, by a series of three more books, Haynes, Henning and Hedetniemi updated the survey with the latest developments [16, 17, 18].

After introducing the problem, we will dedicate the rest of this chapter to giving a current status about the complexity status of Dominating Set, Semitotal Dominating Set and Total Dominating Set on various graph classes.

2.1 The Domination Problem

Semitotal domination was introduced by Goddard, Henning and McPillan [13] as a relaxed form of *total domination*.

DOMINATING SET DECISION [6, p. 586]

Input:	Graph $G = (V, E)$ and an integer k
Question:	Is there a set $X \subseteq V$ of size at most k such that $N[X] = V$?

Goddard, Henning and McPiallan a

SEMITOTAL DOMINATING SET DECISION [13]

Input:	Graph $G = (V, E)$ and an integer k
Question:	Is there a subset $X \subseteq V$ of size at most k such that $N[X] = V$ and for all $d_1 \in X$ there exists another $d_2 \in X$ such that $d(d_1, d_2) \leq 2$?

TOTAL DOMINATING SET DECISION [6, p. 596]

Input:	Graph $G = (V, E)$ and an integer k
Question:	Does there exists a set $X \subseteq V$ of at most k vertices of G such that for every $u \in V(G)$ there exists $v \in X$ with $\{u, v\} \in E$

Definition 2.1.1 (Domination Parameters)**.** *The domination number in a graph G is the minimum cardinality of a dominating set of G, denoted as $\gamma(G)$. The total domination number is the minimum cardinality of a total dominating set (tds) of G, denoted by $\gamma_t(G)$. The semitotal domination number is the minimum cardinality of a semitotal dominating set (sds) of G, denoted by $\gamma_t(G)$*

Since every total dominating set is also a semitotal dominating set and every semitotal dominating set is also a dominating set , we have the following fact first observed by Goddard and Henning [13].

Fact 2.1.1. *For every graph G with no isolated vertex, $\gamma(G) \leq \gamma_{t2}(G) \leq \gamma_t(G)$*

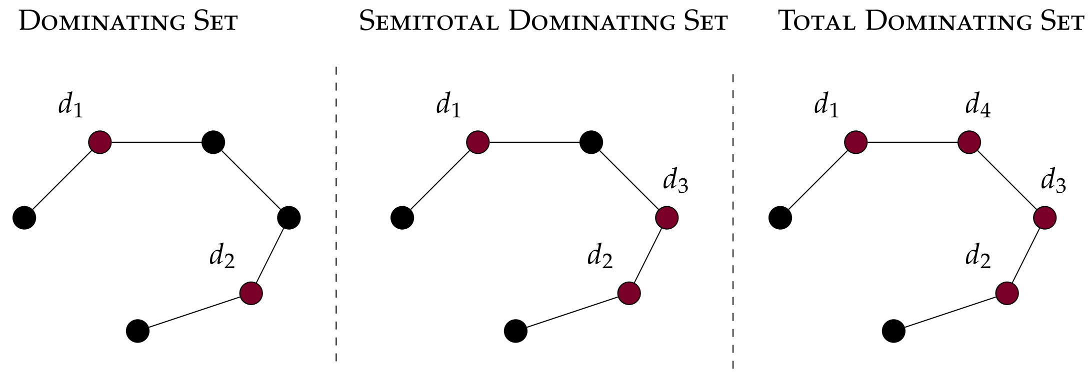


Figure 2.2: *An example for a dominating set, semitotal dominating set and a total dominating set, where $\gamma(G) < \gamma_{2t}(G) < \gamma_t(G)$ are strict. In the first case, only two vertices suffice to dominate all others. In the second one, we need a witness between d_1 and d_2 that is at most distance two. In the last case, d_1 and d_2 both need a neighbor in the total dominating set.*

We can see that the semitotal domination number γ_{t2} is squeezed between the *domination* number and the *total domination* number. It turns out that for some graphs, all of these inequalities can be strict. See Figure 2.2 for an example, where $\gamma(G) < \gamma_{t2} < \gamma_t(G)$.

2.1.1 Preliminaries

* Witness * u pendant ofrom a vertex c if $N(u) = \{w\}$ * domination

Let D be a dominating set of G and $w \in V(G) \setminus D$. For any neighbor $v \in D \cap N(w)$, we say that d_1 *dominates* w For two dominating vertices $d_1, d_2 inD$. If

Definition, dominating number

2.2 Complexity Status of Semitotal Dominating Set

2.3 *w[i]*-Intractibility

Now some w[i] hard classes.

2.3.1 Warm-Up: *W[2]*-hard on General Graphs

As any bipartite graphswith bipartition can be split further into r-partite graphsthis results also implies the $w[1]$-hardness of r-partite graphs

2.3.2 *W[2]*-hard on Bipartite Graphs

We are showing that SEMITOTAL DOMINATING SET is $\omega[2]$-hard on bipartite graphs by a parameterized reduction from DOMINATING SET on bipartite graphs which is known to be $\omega 2$-hard ([24, Theorem 1]).

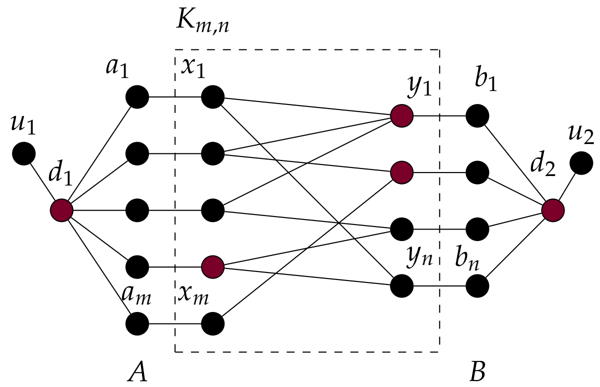


Figure 2.3: *Constructing a bipartite G' from the bipartite graph $K_{m,n}$ by duplicating all vertices and adding exactly two forced witnesses.*

Theorem 1. SEMITOTAL DOMINATING SET *is $\omega[2]$ hard for bipartite Graphs*

Proof. Given a bipartite Graph $G = (\{X \cup Y\}, E)$, we construct a bipartite Graph $G' = (\{X' \cup Y'\}, E')$ in the following way:

1. For each vertex $x_i \in X$, we add a new vertex $a_i \in A$ and an edge $\{x_i, a_i\}$ in between.
2. For each vertex $y_j \in Y$, we add a new vertex $b_j \in B$ and an edge $\{y_j, b_j\}$ in between.
3. We add fource vertices with with edges $\{u_1, d_1\}$ and $\{u_2, d_2\}$, and connect them with all $\{d_1, a_i\}$ and $\{d_2, b_j\}$ ($i \in [m]$ and $j \in [n]$) respectively.

Observation: The constructed G' is bipartite because A and B form an independent set on G′ that can be cross-wise attached to one of the previous vertex sets. Setting $X' = X \cup \{u_2, d_1\} \cup B$ and $Y' = X \cup \{u_1, d_2\} \cup A$ form the partitions of the new bipartite G′.

Corollary 2.3.1. *G has a dominating set of size k iff G has a semitotal dominating set of size* $k' = k + 2$

⇒ Assume a ds in G of size k. We know that $D' = D \cup \{d_1, d_2\}$ is an sds in G' of size $k' = k + 2$, because d_1 dominates u_1 and all $a_i \in A$; d_2 dominates u_2 and all

$b_i \in B$. The rest is dominated by the same vertices as they were in G, but now all of them have either d_1 or d_2 as a witness. More formally, we have by construction of G' that $\forall v \in (D \cap X) : d(v, d_1) = 2$ and $\forall v \in (D \cap Y) : d(v, d_2) = 2$.

$\Leftarrow$ On the contrary, assume any sds D' in G' with size k'. Without loss of generality, we can assume that $u_1, u_2 \notin D'$, because choosing d_1 and d_2 instead is always as least as good and does not violate any witnesses. Therefore, the construction forces d_1, $d_2 \in D'$.

All $a_i \in A$ can only be useful to dominate their partnering x_i ($b_i \in B$ for y_i), because $d_1, d_2 \in D$ is the only second neighbor they have. If $a_i, b_i \in D'$ we replace it with x_i and y_i preserving the size D. As d_1 and d_2 suffice to provide a witness for every vertex in the graph and do not lose any other witnesses, this operation is sound.

In the end, $D = D' \setminus \{d_1, d_2\}$ gives us a ds in G with size $k = k' - 2$

As G′ can be constructed in linear time and the parameter k is only blown up by a constant, this reduction is an FPT reduction. Because Dominating Set is already $w[2]$-hard on bipartite graphs ([)], we imply that Semitotal Dominating Set is $w[2]$-hard as well. □

2.3.3 *W[2]*-hard on Split Graphs

TODO Getting started with that.

2.3.4 *W[2]*-hard on Chordal Graphs

Although the previous result implies $w[2]$-hardness for chordal graphs, we found another reduction from Dominating Set on chordal graphs.

We will introduce the notion of an elimination ordering.

Definition 2.3.1 ([Rose1960]). *In a graph $G = (V, E)$ with n vertices, a vertex is called **simplicial** if and only if the subgraph of G induced by the vertex set $\{v\} \cup N(v)$ is a complete graph.*

*G is said to have a **perfect eliminiation ordering** if and only if there is an ordering $(v_1, ...v_n)$ of the vertices, such that each v_i is simplicial in the subgraph induced by the vertices $v_1, ..., v_i\{\}$*

The following lemma shows that

Lemma 2.3.1 ([Rose1960]). *A graph $G = (V, E)$ is chordal if and only if G has a perfect elimination ordering.*

Theorem 2. Semitotal Dominating Set *restricted to chordal graphs is $\omega[2]$-hard.*

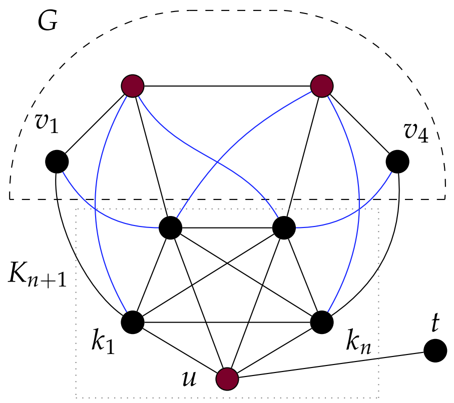


Figure 2.4: *Constructing a chordal G' from the chordal graph P_4 by adding a K_5, connecting its vertices pairwise to G. Adding the (blue) auxiliary vertices are necessary to preserve chordality.*

Proof. We will give a reduction from Dominating Set on chordal graphs. Given $G = (V, E)$ with vertex set $V = \{v_1, ...v_n\}$, we construct a chordal graph G' as described below:

1. Add one complete graph K_{n+1} consisting of the vertices $\{k_1, ..., k_n, u\}$ and an edge $\{v_i, k_i\}$ to each vertex $v_i \in V$ of G. One vertex of the complete subgraph is not connected to any $v \in V$. Denote it as u.
2. Add one additional vertex t and connect it with u vie the edge $\{u, t\}$.
3. For all vertices $v_i \in V$ in G, add a new edge $\{n, k_i\}$ for all neighbors $n \in N(v_i)$.

An example reduction on the graph P_4 is shown in section 2.3.4.

Corollary 2.3.2. *$N(v_i) \in G$ forms a clique iff $N(v_i)$ forms a clique in G'*

Proof. Asuming that $N(v_i)$ forms a clique in G, we show that it also forms a clique in G′ by induction over the number of neighbors $z = abs(N(v_i))$ in G.

- $z = 0$: Holds trivially as we do not have a neighbor in G and in G′ the connected k_i forms a P_1, hence a clique.
- $z = z + 1$:

  By IH, we already know that all neighbors $n_1, ..., n_z$ form a clique together with their vertices in k_i. As $k_{z+1}, v_{z+1} \in N(v_i)$ now also in G′, we show that $N(v_i)$ still forms clique in G′.

  Let k_i be the vertex that was connected with n_i during step 1. All we have to show is that v_{z+1} and k_{z+1} extend our previous clique, hence are fully connected with $N(v_i)$.

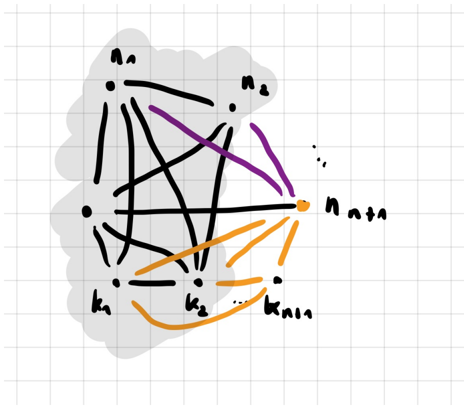


Figure 2.5: Induction Step

v_{z+1} connects to $N(v_i)$ in G by assumption. By our construction, there exists an edge to $k_1, ..., k_z$, because we add an edge (n_{z+1}, k_i) if there is an edge from (n_{z+1}, n_i). (See fig 2.5)

k_{z+1} form a complete subgraph with the other k_i and is connected to all n_i by construction because the edge (n_{z+1}, n_i) exists.

Therefore, $N(v_i)$ will also form a clique in G'.

On the other side, if $N(v_i)$ forms a clique in G′, the vertices of $N(v_i)$ in G form an induced subgraph of G′, hence preserving the clique.

■

Corollary 2.3.3. *G is Chordal iff G′ is chordal.*

Proof. ⇒: Assume G chordal. Then exists a total elimination order $o = (v_1, ..., v_n)$ in G where removing v_j sequentially returns cliques in $N(v_i)$. Define $o' = (v_1, ..., v_n, k_1, ..., k_n, u, t)$. Applying corollary 2.3.2 states that $(v_1, ...v_n)$ always gives cliques in G and according to corollary 2.3.2 also in G′. As the rest is directly part of a clique in G′ by definition with an additional vertex of degree 1, o′ is a total elimination order for G', hence G′ chordal. ⇐: Holds as o′ is always a total elimination order in G′ and removing the complete subgraph K_{n+1} and u gives a total elimination order in G. ■

Corollary 2.3.4. *G has a Dominating Set of size k iff G' has a sds of size $k+1$*

Proof. Assume a ds D of size k in G. $D \cup \{u\}$ is an sds in G' of size $k+1$, because u dominates t and for each $v \in DS : d(v, u) \leq 2$.

Contrary, assume an sds SD in G'. To dominate t, $u \in SD$ must hold, hence already dominating the complete subgraph K_{n+1}. If a vertex $k_i \in SD$, we exchange it with v_i not losing the domination property. Taking $D = SD - \{u\}$ gives our desired ds of size k. ■

As this reduction runs in FPT time and the parameter is only bounded by a function of k, this is an FPT reduction. As Dominating Set on Chordal Graphs is $w[2] - hard$, so is Semitotal Dominating Set on Chordal Graphs.

□

BIBLIOGRAPHY

[1] J. Alber, M. R. Fellows, and R. Niedermeier. "Polynomial-time data reduction for dominating set." In: (May 2004), pp. 363–384. DOI: 10.1145/990308.990309.

[2] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2006. ISBN: 978-0-521-42426-4.

[3] R. Balakrishnan and K. Ranganathan. *A textbook of graph theory*. English. 2nd ed. Universitext. New York, NY: Springer, 2012. ISBN: 978-1-4614-4528-9; 978-1-4614-4529-6. DOI: 10.1007/978-1-4614-4529-6.

[4] S. A. Cook. "The complexity of theorem-proving procedures." In: *Proceedings of the third annual ACM symposium on Theory of computing*. STOC '71. Shaker Heights, Ohio, USA: Association for Computing Machinery, May 1971, pp. 151–158. ISBN: 9781450374644. DOI: 10.1145/800157.805047.

[5] B. Courcelle. "The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs." In: *Inf. Comput.* 85.1 (Mar. 1990), pp. 12–75. ISSN: 0890-5401. DOI: 10.1016/0890-5401(90)90043-H.

[6] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Berlin, Heidelberg: Springer, 2015. ISBN: 978-3-319-21275-3.

[7] V. Diekert and B. Durand, eds. *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*. Vol. 3404. Lecture Notes in Computer Science. Springer, 2005. ISBN: 3-540-24998-2. DOI: 10.1007/b106485.

[8] R. Diestel. *Graph Theory*. Fourth. Vol. 173. Graduate Texts in Mathematics. Heidelberg; New York: Springer, 2010. ISBN: 9783642142789 3642142788 9783642142796 3642142796.

[9] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Ed. by D. Gries and F. B. Schneider. Monographs in Computer Science. New York, NY: Springer New York, 1999. ISBN: 9781461267980 9781461205159. DOI: 10.1007/978-1-4612-0515-9.

[10] L. Fortnow. "Fifty Years of P vs. NP and the Possibility of the Impossible." In: *Commun. ACM* 65.1 (Dec. 2021), pp. 76–85. ISSN: 0001-0782. DOI: 10.1145/3460351.

[11] E. Galby, A. Munaro, and B. Ries. "Semitotal Domination: New Hardness Results and a Polynomial-Time Algorithm for Graphs of Bounded Mim-Width." In: *Theor. Comput. Sci.* 814.C (Apr. 2020), pp. 28–48. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2020.01.007.

[12] V. Garnero and I. Sau. "A Linear Kernel for Planar Total Dominating Set." In: *Discrete Mathematics & Theoretical Computer Science* Vol. 20 no. 1 (May 2018). Sometimes we explicitly refer to the arXiv version: https://doi.org/10.48550/arXiv.1211.0978. DOI: 10.23638/DMTCS-20-1-14. eprint: 1211.0978.

[13] W. Goddard, M. A. Henning, and C. A. McPillan. "Semitotal domination in graphs." In: *A Canadian journal of applied mathematics, computer science and statistics* 94 (June 2014).

[14] T. W. Haynes, S. Hedetniemi, and P. Slater. *Domination in Graphs: Volume 2: Advanced Topics*. New York: Routledge, Oct. 1998. ISBN: 9781315141428. DOI: 10.1201/9781315141428.

[15] T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs*. Boca Raton: CRC Press, Jan. 1998. ISBN: 9780429157769. DOI: 10.1201/9781482246582.

[16] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. *Domination in Graphs: Core Concepts*. 1st ed. Not yet released by the writing of this theiss. Springer Nature, 2022.

[17] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. *Structures of Domination in Graphs*. Google-Books-ID: YbosEAAAQBAJ. Springer Nature, May 2021. ISBN: 9783030588922.

[18] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. *Topics in Domination in Graphs*. 1st ed. Springer Nature, 2020. ISBN: 978-3-030-51116-6.

[19] M. A. Henning and A. Pandey. "Algorithmic aspects of semitotal domination in graphs." In: *Theoretical Computer Science* 766 (2019), pp. 46–57. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2018.09.019.

[20] C. F. d. Jaenisch. *Traité des applications de l'analyse mathématique au jeu des échecs, précédé d'une introduction à l'usage des lecteurs soit étrangers aux échecs, soit peu versés dans l'analyse.* 3 vol. Saint-Pétersbourg: Dufour et cie; [etc., etc.], 1862, 3 vol.

[21] R. M. Karp. "Reducibility among Combinatorial Problems." In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Ed. by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9.

[22] T. Kloks and A. Pandey. “Semitotal Domination on AT-Free Graphs and Circle Graphs.” In: *Algorithms and Discrete Applied Mathematics: 7th International Conference, CALDAM 2021, Rupnagar, India, February 11–13, 2021, Proceedings*. Rupnagar, India: Springer-Verlag, 2021, pp. 55–65. ISBN: 978-3-030-67898-2. DOI: 10.1007/978-3-030-67899-9_5.

[23] L. Levin. “Universal sequential search problems.” In: *Problemy PeredachiInformatskii* (1973).

[24] V. Raman and S. Saurabh. “Short Cycles Make W-hard Problems Hard: FPT Algorithms for W-hard Problems in Graphs with no Short Cycles.” In: *Algorithmica* 52.2 (2008), pp. 203–225. ISSN: 1432-0541. DOI: 10.1007/s00453-007-9148-9.

[25] K. Rosen and K. Krithivasan. *Discrete Mathematics and Its Applications: With Combinatorics and Graph Theory*. McGraw-Hill Companies, 2012. ISBN: 9780070681880.

LIST OF FIGURES

LIST OF TABLES