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
Supervisors:	Prof. Debarghya Ghoshdastidar *Technical University Munich*
	Prof. Paloma Thomé de Lima *IT University Copenhagen*
Submission Date:	January 9, 2023

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

København S
January 9, 2023

Lukas Retschmeier

Acknowledgments

Where do I even begin? I have many people to thank for their tremendous support in this thesis, but first I want to start with the one person who truly made it all possible: my supervisor, *Paloma*!

Not only did you always have time for my questions and discussions, but you also showed a level of enthusiasm and passion for this problem I could not have wished more.

And yes, thank you for achieving your main mission that was to do everything to "find you a PhD programm".

Furthermore, I would like to express my sincere gratitude to my professor at TUM, *Prof. Ghoshdastidar*, for agreeing to formally supervise my thesis. Your immediate willingness to take on this role and the uncomplicated communication was greatly appreciated!

Furthermore, I would like to thank the whole algorithms group at ITU for letting me be a part of the group despite my status as a *mere* Master student.

From the moment I joined the group, you all made me feel welcome and integrated, and I am grateful for the opportunity to do my thesis in such a friendly and helpful environment

Thank you to Riko for inviting me to this year's Retriet, the running sessios with Martin and Christian

Another special thanks to Jan Arne at UiB for introducing me into the fascinating world of parameterized Complexity and engaging me to further dig into this area!

My employer Atos for providing me with the necessary flexibility of working hours

Lastly, I would like to thank openAI's *chatGPT Dec 15* for writing this acknowledgments section and *Dall-E 2* for the immense amount of creativity that went into the chapter's illustrations! What a time to be alive!

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 [**Goddard2014**], **NP**-completeness was shown for general graphs [**Henning2019**], *split graphs* [**Henning2019**], *planar graphs* [**Henning2019**], *chordal bipartite graphs* [**Henning2019**], *circle graphs* [**Kloks2021**] and *subcubic line graphs of bipartite graphs* [**Galby2020**]. On the other side, there exist polynomial-time algorithms for *AT-free graphs* [**Kloks2021**], *graphs of bounded mim-width* [**Galby2020**], *graphs of bounded clique-width* [**Courcelle1990**], and *interval graphs* [**Henning2019**].

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 [**Alber2004**] and [**Garnero2018**] 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

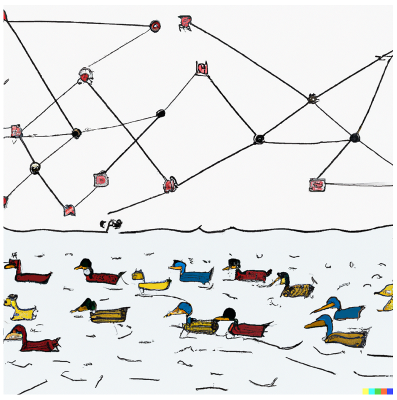
Abstract

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 [**Diekert2005**]. For definitions in the area of *parameterized complexity*, the book written by Cygan et al. [**Cygan2015**] gives an excellent introduction. For standard mathematical notation, the reader is referred to any introductory textbook into discrete mathematics (e.g. [**Rosen2012**]).

1.1 Graph Theory

If not explicitly stated otherwise, the following definitions are taken from the book *Graph Theory* written by Reinhard Diestel [**diestel10**].

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 [**Balakrishnan2012**])**.** *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 [Arora2006]

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 [Arora2006]

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 among each other. This spans a whole "web of reductions" [**Arora2006**] and gives strong evidence that none of these problems can be solved efficiently. The first results in this new field had been published independently by Cook [**Cook1971**] and Levin [**Levin1973**] after Karp [**Karp1972**] had introduced this notion of problem reductions. The Cook-Levin-Theorem [**Cook1971**] proved that the BOOLEAN SATISFIABILITY (SAT) problem is **NP**-COMPLETE any problem in NP can be reduced to SAT.

as hard as SAT, because a fast algorithm for P would immediately give a fast algorithm for SAT as well. 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 [**Arora2006**].

Definition 1.2.1 (Reductions, **NP**-hardness and **NP**-COMPLETENESS [**Arora2006**]). *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 $\mathbf{P} = \mathbf{NP}$ or not is still one of the biggest open questions in mathematics bountied with one million dollars by the *Clay Mathematical Institute* [**Fortnow2021**]. 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 [**Downey1999a**], 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 [**Cygan2015**].

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 **??**. 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*

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

If we bound 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 **??** 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 $\mathfrak{M}$ on

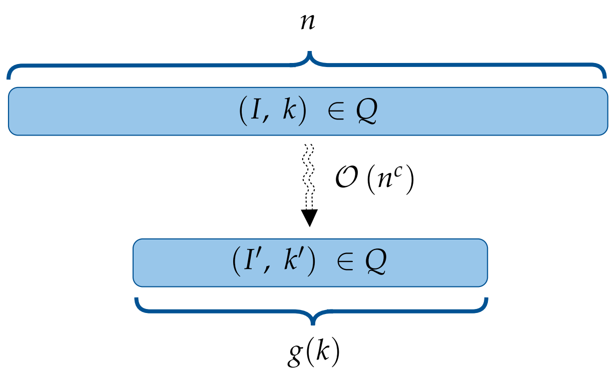


Figure 1.2: *Kernelization: Reducing an instance (I, k) of size n to a smaller instance (I', k') in polynomial time. The size of the kernel is a function $g(k)$ only dependent on k.*

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 parameterized 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. The answer is no and parameterized complexity has another tool in its toolbox that can be used to show that a problem is unlikely to be in **FPT**. The idea is to transfer the concepts of **NP**-hardness from **??** and reductions from the classical setting to the parameterized world. This raises the need for a new type of reduction that ensures that a reduced instance (I', k') is not only created in fpt time, but the new parameter k' depends only on the size of the parameter in the original instance.

There exists a whole hierarchy of classes $\mathbf{FPT} \subseteq \mathbf{W_1} \subseteq \mathbf{W_2} \subseteq \ldots \subseteq \mathbf{W_t} \subseteq \ldots$, which is known as the **W**-hierarchy. It is strongly believed that $\mathbf{FPT} \subsetneq \mathbf{W_t}$ and therefore, we do not expect the existence of an algorithm solving any $\mathbf{W_t}$-hard problem in fpt time.

Definition 1.2.5 (Parameterized Reduction)**.** *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.*

As shown in **????** **[Cygan2015]** this definition ensures that reductions are transitive and closed under fpt reductions.

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.*

If there exists a parameterized reduction transforming a $\mathbf{W_t}$-hard problem A to another problem B, then B is $\mathbf{W_t}$-hard as well. We can define the classes $\mathbf{W_1}$ and $\mathbf{W_2}$, by giving two problems that are complete for these classes.

The Classes W_1 [Downey1995] and W_2 [Cygan2015]

Independent Set is $\mathbf{W_1}$-complete.

Dominating Set is $\mathbf{W_2}$-complete.

A problem P is in the class $\mathbf{W_1}$ (resp. $\mathbf{W_2}$) if there is a parameterized reduction from P to Independent Set (Dominating Set).

We have omitted a more precise definition via the Weighted Boolean Satisfiabiliy problem as it is not important for our work. We refer the interested reader to **[Cygan2015, Fomin2019]** for more details.

CHAPTER 2

ON PARAMETERIZED SEMITOTAL DOMINATION

TODO: Make a nice chatGPT poem.

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 $n \times n$-chessboard [**Jaenisch1862**]. Because of the immense amount of publications related to domination that followed, Haynes, Hedetniemi, and Slater started a comprehensive survey of the literature in 1998 [**Haynes1998**, **Haynes1998b**]. 20 years later, by a series of three more books, Haynes, Henning and Hedetniemi complemented the survey with the latest developments [**Haynes2020**, **Haynes2021**, **Haynes2022**].

We are now introducing the problems of Dominating Set, Semitotal Dominating Set and Total Dominating Set and dedicate the rest of the chapter to giving a current status about the complexity status of various graph classes.

2.1 Domination Problems

We are now going to define the three most important domination problems for this work: Dominating Set, Semitotal Dominating Set and Total Dominating Set. Recall that a dominating set of a graph $G = (V, E)$ is a subset $D \subseteq V$, such that every vertex $V \setminus D$ is adjacent to some vertex in D. We say that a vertex d is a *dominating vertex* or *dominator* if $d \in D$ and that d *dominates* all of its neighbors.

The Dominating Set problem asks for a subset D of size at most k of vertices whose set of neighbors are adjacent o all the other vertices. In other words: every vertex $v \notin D$ needs to have at least one neighbor in D.

DOMINATING SET [Cygan2015]

Input	Graph $G = (V, E)$, $k \in \mathbb{N}$
Question	Is there a set $X \subseteq V$ of size at most k such that $N[X] = V$?

If we also demand the dominating vertices to be dominated, need the notion of Total Domination. The Total Dominating Set problem adds one additional constraint: Every vertex $v \in D$ in the dominating set must also be dominated by some vertex $v' \in D$ which we call the *witness* of v.

TOTAL DOMINATING SET [Cygan2015]

Input	Graph $G = (V, E)$, $k \in \mathbb{N}$
Question	Does there exists a set $X \subseteq V$ with $|X| \leq k$ vertices such that for every $u \in V(G)$ there exists $v \in X$ with $\{u, v\} \in E$?

Finally, Semitotal Domination was introduced by Goddard, Henning and McPillan **[Goddard2014]** as a relaxation of Total Domination. Assume that we have an arbitrary dominating set D for some (connected) graph $G = (V, E)$ that has size at most two. It is easy to observe that for each $v \in D$ there must be at least one other dominating vertex $v' \in D$ that is at most three steps away because otherwise, this would not be a dominating set. On the other side, by definition of a total dominating set D, every dominating vertex $d \in D$ has another vertex in $d' \in N(d) \cap D$ that is also dominating. It is natural to ask what happens if we restrict this distance property to be at most two, which leads us straight to the idea of Semitotal Domination. For a semitotal dominating set D, we say that v witnesses v' if $v, v' \in D$ and $d(v, v') \leq 2$.

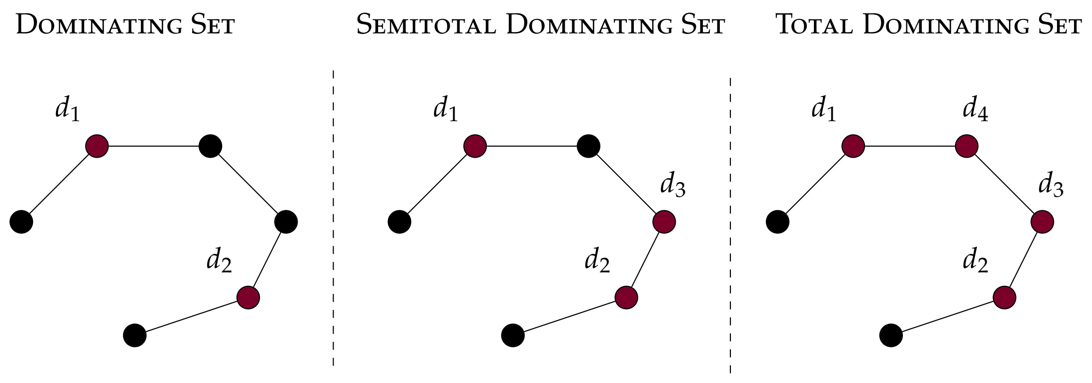


Figure 2.2: *An example for a minimum dominating set, semitotal dominating set and a total dominating set, where $\gamma(G) < \gamma_{2t}(G) < \gamma_t(G)$ are strict. In the first case, only two vertices suffice to dominate all others. In the second one, we need a witness between d_1 and d_2 that is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

SEMITOTAL DOMINATING SET [Goddard2014]

Input	Graph $G = (V, E)$, $k \in \mathbb{N}$
Question	Is there a subset $X \subseteq V$ with $\lvert X\rvert \leq k$ such that $N[X] = V$ and for all $d_1 \in X$ there exists another $d_2 \in X$ such that $d(d_1, d_2) \leq 2$?

?? demonstrates that the minimum ds, sds and tds can strictly differ in the size on a fixed graph $G = (V, E)$. a minimum ds (left) does not need any witnesses at all and we can choose d_1 and d_2 to dominate the graph. As $d(d_1, d_2) = 3$, we have to introduce additional dominating vertices for a minimum sds (middle) and tds (right). Their only purpose is, to bridge the gap between d_1 and d_2.

Definition 2.1.1 (Domination Numbers)**.** *The domination number in a graph G is the minimum cardinality of a dominating set (ds) 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 $\gamma_t(G)$.*

We say that a ds D is minimal if no proper subset $S' \subset S$ is a ds and that D is a minimum if it is the smallest ds.

Because any tds is also an sds and every sds is also a ds, $\gamma t2$ is squeezed between γ and γ_t. The following fact was first observed by Goddard and Henning [**Goddard2014**]:

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

It turns out that for some graphs, all of these inequalities are strict. See **??** for an example, where $\gamma < \gamma_{t2} < \gamma_t$ and therefore.

2.2 Complexity Status of Semitotal Dominating Set

This chapter complements the complexity status of SEMITOTAL DOMINATING SET surveyed by Galby et. al [**Galby2020**] with the latest developments and adds the corresponding parameterized complexities. It is interesting to see, how most of the results are mirrored across DOMINATING SET, SEMITOTAL DOMINATING SET and TOTAL DOMINATING SET. If not mentioned otherwise, our problem is always parameterized by solution size.

In the meanwhile, new polynomial time algorithms have been shown for *dually chordal*, *strongly chordal* [**Tripathi2021**], *AT-free* [**Kloks2021**] and *block* [**Henning2022**] graphs. Furthermore, a linear time algorithm has been found for interval graphs [**Pradhan2021**] beating the previous $\mathcal{O}(n^2)$ algorithm given by Henning and Pandey [**Henning2019**].

On the negative side, SEMITOTAL DOMINATING SET on *circle graphs* graphs and *undirected path graphs* is **NP**-COMPLETE. Our results are highlighted.

All complexity results are shown in **??**.

Graph Class	**Dominating Set**		**Semitotal Dominating Set**		**Total Dominating Set**	
	classical	Parameterized	classical	Parameterized	classical	Parameterized
bipartite	NPc [**Bertossi1984**]	W_2	NPc [**Henning2019**]	W_2 (this)	NPc [**Pfaff1983**]	
line graph of bipartite	NPc [**Korobitsin1992**]		NPc [**Galby2020**]		NPc [**McRae1995**]	
circle	NPc [**Keil1993**]	W_1 [**Bousquet2012**]	NPc [**Kloks2021**]	**Open**	NPc [**McRae1995**]	W_1 [**Bousquet2012**]
chordal	NPc [**Booth1982**]	W_2 [**Raman2008**]	NPc [**Henning2019**]	W_2 (this)	NPc [**Laskar1983**]	
s-chordal	NPc [**Liu2011**]	W_2 [**Liu2011**]	XXX	XX	NPc [**Liu2011**]	W_1 [**Liu2011**]
split	NPc [**Bertossi1984**]	W_2 XXXX false! [**Raman2008**]	NPc [**Henning2019**]	W_2 Lemma XX	NPc [**Laskar1983**]	W_1 [**Chang1998**]
3-claw-free	NPc [**Cygan2011**]	FPT [**Cygan2011**]	Prob. Unk	Prob. Unk	NPc [**McRae1995**]	Unknown
t-claw-free, $t > 3$	NPc [**Cygan2011**]	W_2 [**Cygan2011**]	Prob. Unknown	Unknown	NPc [**McRae1995**]	Prob. Unknown
chordal bipartite	NPc [**Mueller1987**]	Probably Open	NPc [**Henning2019**]	Open?	P [**Damaschke1990**]	
dually chordal	P [**Brandstaedt1998**]		Unkown	Unkown	?	
strongly chordal	P [**Farber1984**]		P [**Tripathi2021**]		NPc [**Farber1984**]	
AT-free	P [**Kratsch2000**]		P [**Kloks2021**]		P [**Kratsch2000**]	
tolerance	P Source!		Unknown	Unkown	Unknown	Unknown
block	P [**Farber1984**]		P [**Henning2022**]		P [**Chang1989**]	
planar	NPc (Sources!)	FPT [**Alber2004**]	NPc	FPT **(this)**	NPc	FPT [**Garnero2018**]
undirected path	NPc [**Booth1982**]	FPT [**Figueiredo2022**]	NPc [**Henning2022**]	Unkown	NPc [**Lan2014**]	Unkown (?)
interval	P [**Chang1998a**]		P [**Pradhan2021**]		P [**Bertossi1986**]	
bounded clique-width	P [**Courcelle1990**]		P [**Courcelle1990**]		P [**Courcelle1990**]	
bounded mim-width	P [**Belmonte2011**, **BuiXuan2013**]		P [**Galby2020**]		P [**Belmonte2011**, **BuiXuan2013**]	

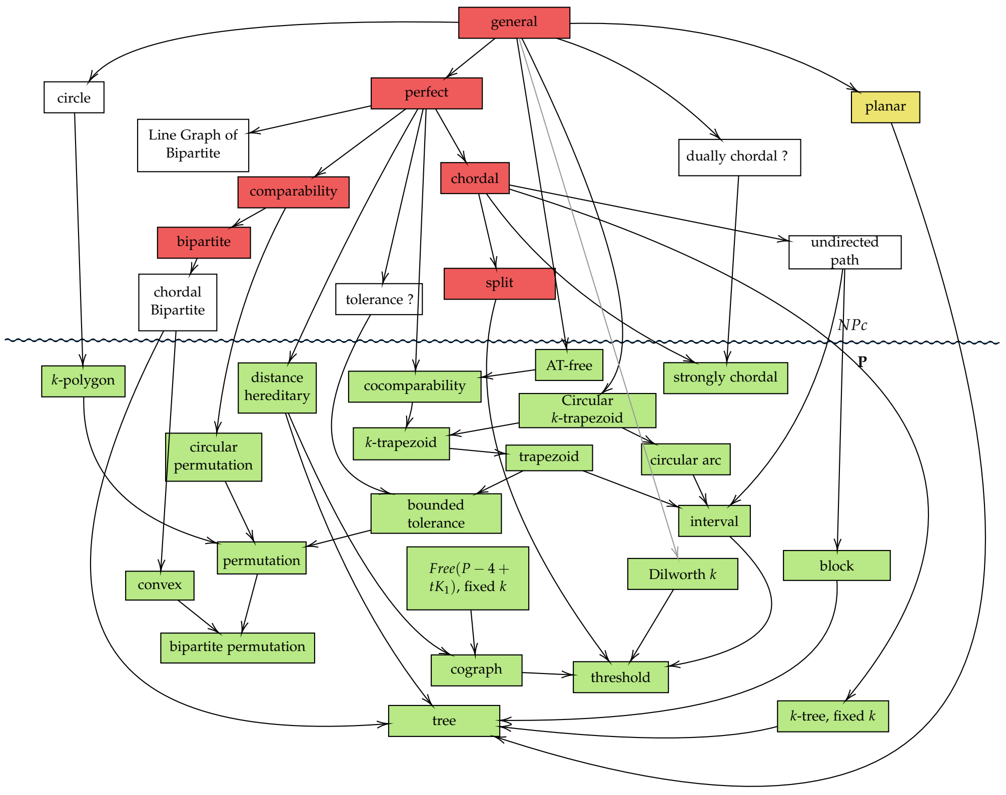


Figure 2.3: *Computational Complexity of* Semitotal Dominating Set. *Graph classes that admit a polynomial time algorithm are marked in green, those with an fpt algorithm in yellow and those that are not fpt in red. If the parameterized complexity is unknown, no color is used. This updates the figure given in [**Galby2020**].*

LIST OF FIGURES

LIST OF TABLES