A New Optimality Measure for Distance Dominating Sets

by

Narges Simjour

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2006

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We study the problem of finding the smallest power of an input graph that has k disjoint dominating sets, where the ith power of an input graph G is constructed by adding edges between pairs of vertices in G at distance i or less, and a subset of vertices in a graph G is a dominating set if and only if every vertex in G is adjacent to a vertex in this subset. The problem is a different view of the d-domatic number problem in which the goal is to find the maximum number of disjoint dominating sets in the dth power of the input graph.

This problem is motivated by applications in multi-facility location and distributed networks. In the facility location framework, for instance, there are k types of services that all clients in different regions of a city should receive. A graph representing the map of regions in the city is given where the nodes of the graph represent regions and neighboring regions are connected by edges. The problem is how to establish facility servers in the city (each region can host at most one server) such that every client in the city can access a facility server in its region or in a region in the neighborhood. Since it may not be possible to find a facility location satisfying this condition, "a region in the neighborhood" required in the question is modified to "a region at the minimum possible distance d".

In this thesis, we study the connection of the above-mentioned problem with similar problems including the domatic number problem and the d-domatic number problem. We show that the problem is NP-complete for any fixed k greater than two even when the input graph is restricted to split graphs, 2-connected graphs, or planar bipartite graphs of degree four. In addition, the problem is in P for bounded tree-width graphs, when considering k as a constant, and for strongly chordal graphs, for any k. Then, we provide a slightly simpler proof for a known upper bound for the problem. We also develop an exact (exponential) algorithm for the problem, running in time $O(2.73^n)$. Moreover, we prove that the problem cannot be approximated within ratio smaller than 2 even for split graphs, 2-connected graphs, and planar bipartite graphs of degree four. We propose a greedy 3-approximation algorithm for the problem in the general case, and other approxi-

mation ratios for permutation graphs, distance-hereditary graphs, cocomparability graphs, dually chordal graphs, and chordal graphs. Finally, we list some directions for future work.

Acknowledgments

First, I would like to thank my supervisor, Naomi Nishimura, for being considerate, patient, and encouraging. I could not complete this thesis without her helpful feedbacks.

Many thanks to Jonathan Buss and Jochen Konemann for accepting to be my thesis readers and for their great suggestions.

I also would like to thank Therese Biedl for the wonderful course in graph theory she taught. The course helped me much in proving the results in this thesis.

I benefited much from my meetings with my writing tutors, Nicole Keshav and Janne Janke. I really appreciate their patience.

I want to thank the Iranian community in Waterloo who warmly welcomed me, as well as all my friends in Iran who have always been close to me. Special thanks goes to Azadeh Fakhrzadeh, Somayeh Azarnoosh, Raheleh Salari, Zahra Imanimehr, Mahdieh Soleymani, and Sara Sadeghi.

Last but definitely not least, I want to thank my parents, Afsaneh Rezaie and Reza Simjour, who have supported me through all my life in every possible way. I also would like to thank my extra parents, Farideh Kiani and Ahmad Chiniforooshan, my brothers, AmirHossein and Mohammad, and my sisters-in-law, Elham and Erfan, for their kindness and constant support. Special thanks to my husband, Ehsan Chiniforooshan, for his companionship, support, encouragement, and for his helpful discussions.

Dedication

This work is dedicated to my beloved mother and father.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The concept of dominating sets is a fundamental and practical notion in computer science. A dominating set for a graph $G = (V, E)$ is a subset S of V such that each vertex in V either is in S or has a neighbor in S. Assuming that the vertices of the input graph represent different regions of a city and the edges connect vertices representing close regions, a dominating set in the graph is a good candidate for the set of regions providing a facility. Such facility location ensures that each region can access the facility in a short time. Similar frameworks where cities can host multiple facilities, bounded by a capacity value, are also interesting.

The domatic number problem, which consists of finding the maximum number of disjoint dominating sets in an input graph, was introduced by Cockayne and Hedetniemi [15]. This number was called the domatic number of the given graph. Translated to the above-mentioned facility location problem, the domatic number problem can be viewed as a multi-facility location problem where each node of the graph has capacity one, i.e. it can provide only one facility; the goal in this problem is to find the maximum number of facilities for which a multi-facility location can be found such that in the neighborhood of every node in the graph all the facilities can be found. Feige et al. [25] claimed in their paper they can show that the general case of having arbitrary capacities for graph vertices, even different for different vertices, is reduced to this single unit capacity case, where each vertex can host at

most one facility.

Different variants of dominating sets lead to different domatic number problems. Haynes and Hedetniemi have surveyed the results on almost 150 different variants of dominating set, and some of their corresponding domatic number problems [35]. Distance dominating set, for instance, introduced by Borowiecki and Kuzak in 1976 [7], refers to a subgraph S of graph vertices such that every vertex in the graph is in S, or has a neighbor within a given distance d in S. Replacing the dominating sets with d-dominating sets (i.e. distance dominating sets with parameter d) in the definitions of the domatic number of a graph and the domatic number problem, results in the d-domatic number of a given graph and the d-domatic number problem. The d-domatic number problem has the same application as the domatic number problem, except that in the corresponding facility location problem clients are more tolerant and accept being a distance d from facility centers.

The following applications are two examples of the applications of the d-domatic number problem. The first problem is a generalized version of the communication network problem mentioned by Riege and Rothe as a justification for the domatic number problem [50]. The second one is an information replication problem that arises in networks.

1. Consider k different wireless services such as satellite TV, radio channels, and cell phones. How can we establish a number of transmission centers, e.g, TV stations and radio stations, such that every customer v in the network can access all the wireless services, i.e. for every service s, $1 \leq s \leq k$, there is a transmission center providing s within a given distance d of v? In which cases is this placement possible?

2. How can we distribute copies of data items $p_1, p_2, \ldots, p_k$ in a network of computers such that exactly one data item is stored on each computer and for every computer v in the network copies of all the data items are within a given distance d of v? We can think of d as a distance limit that specifies how far from a computer we are allowed to store data items. We can modify the question to present exactly the domatic number problem: given a graph

G and distance d, what is the maximum possible number of data items k that allows such a data distribution?

In this thesis, we study the d-domatic number problem from a slightly different point of view, in the sense that for a given graph G and an integer k, we seek the minimum d for which k disjoint d-dominating sets can be found in G; as mentioned earlier, in the d-domatic number problem the goal was to maximize the number of disjoint dominating sets k for a given distance limit d. This problem, denoted by the *minimum distance domatic number problem*, or the MDDN problem, is motivated by applications with a fixed number of services, where we want to specify the facility centers such that each client can access all services in a short time, i.e. facility centers are within an acceptable distance of each client. In such applications, it is reasonable to seek the minimum possible distance d for which a facility location, with worst case access time d, can be found. Similar problems with different optimality measures, mostly considering average access time criteria, have been studied before [5, 40, 42, 4].

Although the d-domatic number problem is not a new problem, no one has studied this problem considering d as the optimality measure. However, previous results on the domatic number and d-domatic number problems can be carried over to this version of the problem. In the rest of this section, we review the known results on the domatic number problem, the d-domatic number problem, and variants of the d-domatic number problem, along with our improvements for some cases. Then, we study the connection of these results to our work and summarize our contributions to the MDDN problem in this thesis.

We consider the results on the domatic number problem in two categories of results on arbitrary input graphs and results on special families of graphs. In each case, we survey the NP-hardness results and polynomial time algorithms, approximability results, and exponential exact algorithm developments.

The domatic number problem was one of the NP-complete problems listed by Garey and Johnson [31]. Even the restricted problem of determining whether the vertices of a given graph can be partitioned into three disjoint dominating sets,

called the three domatic number problem, has been proved to be NP-complete [31]. Riege and Rothe have shown the domatic number problem is closely connected to the boolean hierarchy levels [50] defined by Cai et al. [12, 13]; they proved that the problem of determining whether the domatic number of a graph is among k given values is complete for the $2k$th level of boolean hierarchy over NP, denoted by $\mathrm{BH}_{2k}(\mathrm{NP})$, with respect to polynomial time reductions. In particular, the $k = 1$ case is called the *exact domatic number problem* which asks whether the domatic number of a graph is equal to a given integer. Therefore, the exact domatic number problem is complete for $\mathrm{BH}_2(\mathrm{NP})$.

Feige et al. gave the first approximation result for the domatic number problem on general graphs in 2000 [25]. They developed an algorithm to partition any input graph into $(1-o(1))(\delta+1)/\ln(n)$ disjoint dominating sets, where δ is the minimum degree of vertices in the input graph. Since the number of disjoint dominating sets cannot exceed $\delta + 1$, this gives a $(1 + o(1)) \ln(n)$-approximation algorithm for the problem. Furthermore, they proved that having a $(1 - \epsilon) \ln(n)$-approximation algorithm implies that $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{O(loglogn)})$. In a non-constructive proof they showed the $(1-o(1))(\delta+1)/\ln(\Delta)$ lower bound for the domatic number of general graphs. After refinement, this proof led to an algorithm finding $\Omega(\delta/\ln(\Delta))$ disjoint dominating sets in the input graph in polynomial time. It is worth mentioning that the only previously proved lower bound on the domatic number of general graphs was $\lceil n/(n-\delta) \rceil$ [58].

As for fast exponential-time exact algorithms, the only results of which we are aware are due to Riege and Rothe [51], Fomin et al. [26], and Riege et al. [52]. Riege and Rothe obtained an $\tilde{\mathcal{O}}(2.9416^n)$ time algorithm for the three domatic number problem, where $\tilde{\mathcal{O}}$ is used to ignore polynomial factors in the running time as is usual in working with exponential-time algorithms. Then, Fomin et al. succeeded to obtain a faster algorithm solving the domatic number problem in the general case in $\tilde{\mathcal{O}}(2.8805^n)$ time. Their algorithm uses almost the same technique as the classical algorithm for the chromatic number problem, introduced by Lawler [43]. Then, we developed a $\tilde{\mathcal{O}}(2.7393^n)$ algorithm for the three domatic number problem, improving the running time of Fomin et al.'s algorithm. Recently, Riege et al.

found a new algorithm for the three domatic number problem with $\tilde{\mathcal{O}}(2.695^n)$ time complexity [52]. However, the approaches taken are different, and we present our exact algorithm in this thesis.

The difficulty of the domatic number problem in general has motivated many studies on solving the domatic number problem for special families of graphs; it has been shown that the domatic number problem can be solved in polynomial time for strongly chordal graphs [24], whereas it is NP-complete for circular-arc graphs [6], bipartite graphs, and therefore for comparability graphs, split graphs, and hence for chordal and co-chordal graphs [38]. Furthermore, in this thesis, we prove the NP-completeness of the domatic number problem for planar bipartite graphs of maximum degree four, which was not known before.

Other results on the domatic number of special families of graphs consist of exact algorithms developed for solving the three domatic number problem for graphs of maximum degree 3 and 4 [51], development of a 4-approximation algorithm for the domatic number problem on circular-arc graphs [47], and proving that the existence of an $(1-\epsilon)\ln(n)$-approximation algorithm for split or bipartite graphs implies NP $\subseteq$ DTIME($n^{O(loglogn)}$).

As explained in the introduction, a variation of the domatic number problem, called the d-domatic number problem, provides the basis for the MDDN problem's definition. As far as we know, the only paper working explicitly on the d-domatic number problem is due to Zelinka [59]; in this paper, Zelinka combined the concept of distance domination with domatic number and introduced the d-domatic number problem [59]. He gave a constructive proof stating that the d-domatic number for different graphs may be any value between $d+1$ and n. As special cases of the d-domatic number problem, he calculated the exact value of the d-domatic number for a path or a circle. Moreover, he proved that the $(k-1)$-domatic number of any graph G is at least $\min\{n, k\}$. In this thesis, we give a slightly simpler proof for this bound.

A generalization of d-domatic number, called (r,d)-configuration, was listed among the dominating set applications in a book by Haynes et al., published in

1998 [36]. A (r, d)-configuration problem asks whether in a given graph G with nodes of capacity r a multi-facility location can be found such that each node can access all facilities, where a node of capacity r can host at most r facilities; a node v has access to a facility s if and only if there is a facility center providing s within distance d of v. However, the two references [29, 30] mentioned for this concept were both considering $(r, 1)$-configurations rather than working on both parameters r and d. Later in 2002, Feige et al. claimed that the domatic number problem for the general case of having arbitrary capacities for graph vertices, even if not equal capacities for different vertices, is reduced to the single unit capacity case, where each vertex can host at most one facility [25]. In other words, computing an $(r, 1)$-configuration can be reduced to the $(1, 1)$-configuration which is the classic domatic number problem. Unfortunately, since their paper focused on a different issue, they did not give any argument to support their statement.

The *all-factor d-domatic coloring* and *matched-factor d-domatic coloring* problems, introduced by Alon et al. [1], are other variations of the d-domatic number problem. As first defined by Brigham and Dutton, a *ℓ-factorization* of a graph G is a partitioning of G into ℓ edge-disjoint subgraphs called *factors* [11]. Alon et al. generalized the factorization concept and accepted edge-overlapping subgraphs as factors. They also defined an *all-factor d-domatic coloring of a graph G with k colors* for a given t-factorization of G as partitioning $V(G)$ into k partitions such that each partition is a d-dominating set for each of the factors. In addition, they defined a *matched-factor d domatic coloring of G* for a given k-factorization of G as finding k disjoint d-dominating sets $D_1, D_2, \ldots, D_k$ in G such that D_i dominates the ith factor, $1 \leq i \leq k$. In the *all-factor coloring problem with input k and ℓ*, the goal is to find the minimum distance d for which an acceptable all-factor d-domatic coloring with k colors can be done given any ℓ-factorization of the input graph G. Alon et al. obtained two upper bounds $\lceil 3(\ell k - 1)/2 \rceil$ and $O(k \log(k\ell))$ on this minimum value, when $\ell \geq 2, k \geq \ell$ and $\ell \geq 1, k \geq 1$, respectively. In addition, they proved a lower bound $\omega(k \log(\ell))$ on this value, when $k \geq 2$ and $\ell \geq 4$ [1]. The *matched-factor coloring problem with input k* is defined similarly; it asks for the minimum distance d for which an acceptable matched-factor d-domatic

coloring with k colors can be done given any k-factorization of the input graph G. For this optimum value, Alon et al. proved a lower bound k and an upper bound $\lceil 3(k-1)/2 \rceil$, for $k \geq 2$.

Now, we explain the known results and new contributions on the MDDN problem in four categories of NP-complete and polynomial cases, upper bounds on the solution to the MDDN problem, development of exact (exponential) algorithms, and approximability aspects of the problem. We use MDDN(G, k) to denote the solution to the MDDN problem when the input graph is G and the number of required disjoint distance dominating sets is k.

We will see in the preliminaries section that the polynomial computability of the domatic number of strongly chordal graphs results in a polynomial time algorithm solving the MDDN problem for strongly chordal graphs. We will also show that MDDN(G, k) can be computed in polynomial time for bounded tree-width graphs, for any constant k. Moreover, we will show that if the domatic number problem is NP-complete for a special family of graphs, the MDDN problem is also NP-complete for it. Therefore, the previously mentioned NP-completeness results for the domatic number problem prove that the MDDN problem is NP-complete in general, and even for circular-arc graphs, bipartite graphs, split graphs, chordal graphs, and co-chordal graphs. In addition, we will give a reduction from the k-coloring problem to the MDDN problem in Chapter 3. This reduction helps us prove the NP-completeness of computing MDDN(G, k) for any fixed $k \geq 3$ even for split graphs and 2-connected graphs. Furthermore, using this reduction, we also prove that the problem of determining MDDN$(G, 3)$ is NP-complete for planar bipartite graphs of degree four.

We will see in the preliminaries section that Zelinka's bound on the $(k-1)$-domatic number results in the tight upper bound $k-1$ for MDDN(G, k). We prove Zelinka's upper bound using a slightly simpler argument.

The *all-factor d-domatic coloring* and *matched-factor d-domatic coloring* problems can be viewed as restricted forms of our problem. These problems are very similar to our problem in the sense that the goal in both these problems are finding

a minimum distance d that allows an acceptable partitioning for the vertices of a given graph into k partitions. The desired partitioning in both of these frameworks is not only an acceptable d-domatic partitioning, but also each partition must dominate $V(G)$ via a constrained set of edges. Therefore, any upper bound on these values is also an upper bound on the solution to the MDDN problem. However, all the upper bounds proved for these problems are larger than $k-1$ [1], i.e. the upper bound on the MDDN problem. Therefore, the results on these problems do not have any effect on the MDDN problem.

We will show in the preliminaries section any exact exponential algorithm for the domatic number problem or the three domatic number problem, that runs in time $\tilde{\mathcal{O}}(\alpha^n)$, gives an exact exponential algorithm of running time $\tilde{\mathcal{O}}(\alpha^n)$ for computing MDDN(G,k) in general or for $k=3$, respectively. Consequently, the exact algorithms mentioned for computing the domatic number problem give $\tilde{\mathcal{O}}(2.9416^n)$, $\tilde{\mathcal{O}}(2.8805^n)$, $\tilde{\mathcal{O}}(2.7393^n)$, and $\tilde{\mathcal{O}}(2.695^n)$ time algorithms for computing MDDN$(G,3)$, and an $\tilde{\mathcal{O}}(2.8805^n)$ time algorithm for solving the MDDN problem in general.

We propose an algorithm approximating the MDDN problem within ratio 3, and prove that this ratio cannot be improved to $(2-\epsilon)$ for any $\epsilon > 0$, unless P = NP. This inapproximability result holds also for split graphs, 2-connected graphs, and planar bipartite graphs of $\Delta \leq 4$. In addition, we obtain better approximation ratios for permutation graphs, cocomparability graphs, distance-hereditary graphs, dually chordal graphs, and chordal graphs.

In summary, in this thesis, we study the status of the MDDN problem on various graph families, develop an exact (exponential) algorithm for the problem, and investigate the approximability aspects of the MDDN problem.

The remainder of this thesis is structured as follows: we fix pertinent notation in the following chapter. We also discuss related problems and their connection to our problem. In Chapter 3, we study the complexity of the problem in the general case and for special graph classes. Then, in Chapter 4, we present a slightly simpler proof for a known upper bound for the problem. In Chapter 5, we explain a new

exact exponential-time algorithm for the problem. Chapter 6 addresses different approximability aspects of the problem. At the end, we have a conclusion and summarize possible future work.

Chapter 2

Preliminaries

In this chapter, we fix pertinent notation for the problem and list several observations and preliminary results for the problem. We first present graph theory preliminaries in Section 2.1. As will be shown in Section 2.4, the MDDN problem is an optimization problem and is NP-hard in the general case. To tackle NP-hard optimization problems, several approaches, including solving the problem on restricted domains, development of approximation algorithms, and exact (exponential) algorithms have been suggested. Motivated by these suggestions, we will look at the MDDN problem from these three points of view later in this thesis. But first, we briefly overview the basic definitions of approximation algorithms and exact (exponential) algorithms in Sections 2.2 and 2.3. Then, we present the formal definition of the main problem in Section 2.4, continued by a list of preliminary or known results for the problem.

2.1 Graph Theory

We first define the necessary notation of graph theory used in this thesis. For further information on graphs, we refer the reader to a book by Douglas B. West [56].

A graph G is a pair (V, E), where V is a set and $E \subseteq \{\{u, v\} \mid u, v \in V\}$. V and E denote the sets of *vertices* and *edges* in the graph, respectively. We may

use $V(G)$ and $E(G)$ instead of V and E. The numbers of vertices and edges in the graph G, i.e. $|V(G)|$ and $|E(G)|$, are represented by n_G and m_G. An edge $e = \{u, v\}$ is called a *self-loop* when its two *endpoints* u and v are the same vertex. A graph is *simple* if it does not have any self-loops among its edges.

We call two vertices $u, v \in V(G)$ *adjacent* if there exists an edge $e = \{u, v\}$ in E connecting u and v. For every vertex $u \in V(G)$, we use $N_G(u)$ to denote the set of vertices in G that are adjacent with u. We call this set of vertices the *neighbors of u in G*. We can also define the *neighbors of a set of vertices $U \subseteq V(G)$* as $N_G(U) = \{v \mid \exists u \in U \text{ such that } v \in N_G(u)\}$. For each vertex v, the number of adjacent vertices with v is called $deg(v)$, or the *degree* of v. In the rest of this thesis, we denote the minimum (maximum) degree of a graph G by $\delta(G)$ ($\Delta(G)$).

A *path between vertices u and v* is defined as a sequence of vertices $w_1, w_2, \ldots, w_{\ell+1}$ such that $u = w_1$, $v = w_{\ell+1}$, $\{w_i, w_{i+1}\} \in E$ for all $1 \leq i \leq \ell$, and each vertex appears at most once in this sequence. The *length* of a path P is the number of edges in P. We denote an n-vertex graph consisting of a path of length $n-1$ without any more edges by P_n. For every pair of vertices u and v in a graph G, the *distance between u and v*, denoted by $d_G(u, v)$, is the length of the shortest path from u to v in G. The *distance between a vertex w and a set of vertices $U \subseteq V$* is defined as $\min\{d_G(u, w) \mid u \in U\}$. The *diameter* of a graph G, denoted by $\text{diam}(G)$, is the maximum distance among all possible pairs of vertices in G. When we speak of *vertices within a distance r of v*, we refer to the set of vertices $\{u_1, \ldots, u_\ell\}$ such that $d_G(u_i, v)$ is at most r for all $1 \leq i \leq \ell$.

A *cycle of length ℓ* is a path of length $\ell - 1$ where the first and last vertices in the corresponding sequence are adjacent. In a cycle C, every edge connecting two non-consecutive vertices in the cycle sequence is called a *chord*. A chord in C is said to be *strong* if it connects two vertices w_i and w_j, $i \leq j$, in the cycle sequence such that either $(j-i)$ or $(n-(j-i))$ is odd. In other words, there is an odd-length path in C from w_i to w_j, or from w_j to w_i.

We may eliminate the index G whenever G is known from context.

We say a graph G is connected if for all pairs of vertices $u, v \in V(G)$ there is

a path between u and v in G. The *complement* of a graph $G = (V, E)$ is a graph whose vertex set is V and has an edge between any two vertices $u, v \in V$ if and only if $\{u, v\} \notin E$. *Removing an edge* $\{u, v\}$ from a graph G means constructing a new graph with the same vertex set connected via edge set $E - \{u, v\}$. Similarly, the graph produced by *removing a vertex* v from G is a graph with vertex set $V - v$ and edge set E, except that in this case we also remove edges of the form $\{v, u\}, u \in V$, to avoid edges with one endpoint. For a graph $G = (V, E)$, any graph generated after a sequence of vertex and edge removals is called a *subgraph of* G. A subgraph of G is *spanning* if it has the same vertex set as G. An *induced subgraph* of G is a graph resulting from removing a number of vertices from G. The *subgraph induced by a set of vertices* $U \subseteq V$ is the induced subgraph of G with vertex set U. A set of vertices U in G is an *independent set* if the subgraph induced by U has no edges. An induced subgraph H of G is called *isometric* if the distances between H vertices are the same as their distances in G. A *subdivision graph of* $G = (V, E)$ is a graph constructed by adding a new vertex w_{ij} to V for every $\{v_i, v_j\} \in E$, and replacing every edge $\{v_i, v_j\} \in E$ by two edges $\{v_i, w_{ij}\}$ and $\{w_{ij}, v_j\}$.

In a connected graph G, a vertex $v \in V(G)$ is a *cut-vertex* if removing v from G a disconnected graph. A connected subgraph B of G is called *block* if it has no cut-vertex and any other subgraph B' of G containing B, i.e. B is a subgraph of B', has a cut-vertex.

As we will look at the problem for special graph classes, here we present definitions of several special families of graphs. In the following list, G is an n-vertex graph, and each line specifies the property necessary for G to be categorized as an special graph.

- *Complete graph, denoted by* K_n*:* Each pair of vertices $u, v \in V(G)$ is adjacent.
- *Tree:* G is a connected graph without any cycles.
- *Split graph:* $V(G)$ can be divided into two partitions, i.e. sets of vertices, X and Y, such that the subgraph induced by X is a complete graph and Y is an independent set in G.

- *Bipartite graph:* $V(G)$ can be partitioned into two partitions X and Y such that both X and Y are independent sets in G.

- *k-connected graph:* For every pair of vertices $u, v \in V(G)$, there are k vertex-disjoint paths in G between u and v.

- *Block graph:* Every block in G is a complete graph.

- *Chordal graph:* Every cycle of length at least four in G has a chord.

- *Strongly chordal graph:* G is chordal, and every cycle of G on at least five vertices has a strong chord.

- *Dually chordal graph:* G has a *maximum neighborhood ordering* defined as follows. A *maximum neighbor* for a vertex v in G is a vertex u for which $N_G(w) \subseteq N_G(u)$ for any vertex $w \in N_G(v)$ other than u. A *maximum neighborhood ordering* is an ordering $(v_1, v_2, \ldots, v_n)$ for $V(G)$ such that for every $1 \leq i \leq n$, v_i has a maximum neighbor u_i in the subgraph G_i of G induced by $\{v_j \mid j \geq i\}$; that is, for every vertex $w \in N_{G_i}(v_i)$ the condition $N_{G_i}(w) \subseteq N_{G_i}(u_i)$ holds.

- *Comparability graph:* $E(G)$ can be oriented such that for every three vertices $u, v, w \in V(G)$, $\{u, v\}, \{v, w\} \in E(G)$ with $\{u, v\}$ oriented from u to v and $\{v, w\}$ oriented from v to w, $\{u, w\} \in E(G)$ and it is oriented from u to w.

- *Planar graph:* G can be drawn on the plane such that no two edges cross in a point other than their endpoints.

- *Distance-hereditary graph:* All connected induced subgraphs of G are isometric.

- *k-tree:* G is a chordal graph and $w(G) \leq (k + 1)$, where $w(G)$ is the size of the maximum complete subgraph (clique) of G.

- *Graph of tree-width k:* G is a subgraph of k-tree.

- *Bounded tree-width graphs:* Graphs whose tree-widths are bounded by a constant value.

To define the next four graph classes, we need to establish the following general term:

Definition 2.1. *The* intersection graph *of a family of non-empty sets $\mathcal{F}$ is constructed by creating a vertex for each set in $\mathcal{F}$ and putting an edge between any pair of vertices whose corresponding sets intersect.*

Various $\mathcal{F}$'s lead to different graph families:

- *Interval graph:* intersection graph of a set of intervals on the real line.
- *Circular-arc graph:* the intersection graph of a set of arcs of a circle.
- *Directed path graph:* the intersection graph of a set of directed paths of a directed tree.
- *Permutation graph:* the intersection graph of line segments in the matching diagram of a permutation; for a permutation $(\pi_1, \pi_2, \ldots, \pi_n)$ of numbers $\{1, 2, \ldots, n\}$, the *matching diagram* is drawn by putting n points in a horizontal row in correspondence with numbers $1, 2, \ldots, n$, and another n points in a horizontal row below the first row, corresponding to π_1 to π_n. Then, the points corresponding to the same numbers in the first and the second rows are connected via straight line segments. The intersection graph of these drawn line segments is a permutation graph. The matching diagram of permutation $(4, 3, 5, 1, 2)$ and the constructed permutation graph is shown in Figure 2.1.

Starting the name of any of the above-mentioned families of graphs, say family $\mathcal{F}$, with 'co' specifies another family of graphs whose complements fall in $\mathcal{F}$. For example, 'cochordal' graphs refers to the set of graphs with a chordal complement.

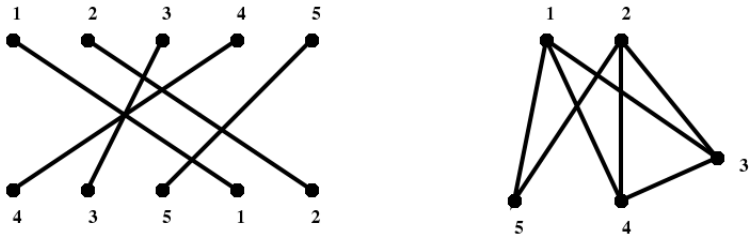


Figure 2.1: The matching diagram and the corresponding permutation graph for permutation $(4, 3, 5, 1, 2)$

2.2 Approximation Algorithms

To study the MDDN problem, it is useful to note that it is an optimization problem. An *optimization problem* is a quadruple (I, s, m, g), where I is the set of *input instances*, s is a function that returns a set of *feasible solutions* for each input instance, m is the *measure function* returning a real *measure value* for each feasible solution and input instance, and g is the *goal function* which is min or max. The goal is to find for a given input instance x a feasible solution *opt* such that $m(opt, x) = g\{m(y, x) \mid y \in s(x)\}$. We call such a solution *an optimal solution*, and its corresponding measure value the *optimum*. For further information on optimization problems the reader is referred to a book by Vazirani [55].

In difficult optimization problems, it is reasonable to seek an algorithm returning almost optimal solutions, rather than exact solutions. For a problem P, a polynomial-time algorithm A that returns a feasible solution of measure value at most $t{\cdot}\mathrm{OPT}(P)$, for every input instance $x \in I$, is called a *t-approximation algorithm for P*. We refer to the t value as the *approximation ratio* of A. In this thesis, we also use *(t, c)-approximation* to denote an algorithm always returning a value at most $t{\cdot}\mathrm{OPT}(P) + c$.

Next, we explain one of the approximation tools for graph problems. This notion will be useful in developing approximation algorithms for the MDDN problem on

special families of graphs.

2.2.1 Distance-Approximating Graphs

A well-studied problem is whether a metric space can be embedded into another metric space such that the distances in the new metric space are approximately the same as the original distances. This concept has lots of applications in computational geometry, graph algorithms, compact routing, and network design [37, 44, 20, 33, 53, 17]. A graph can be viewed as a geometric object, in a metric space, specified by vectors representing vertices, where the distance between each pair of vectors is exactly the length of a shortest path connecting their corresponding vertices. In many problems it is useful to transform the input graph to a simpler graph approximating the distances in the original graph, and solve the problem for this "distance approximating graph". We are interested in embeddings of graph metric spaces to graph metric spaces. The studies in this area have been mainly focused on mapping input graphs to sparse distance-approximating graphs [16, 49, 14], in particular spanning trees. In many cases finding distance-approximating spanning trees is not possible. For applications such as efficient routing, it is also beneficial to find a set of trees for the input graph G such that the distance between each pair of vertices in G is approximated in one of the trees [34, 22].

We define distance-approximating graphs in more detail, as we will use several known results on them to design approximation algorithms for the MDDN problem:

Definition 2.2. *A* distance-approximating graph *for a graph $G = (V, E)$ is a graph $R = (V', E')$ in which for every pair of vertices $u, v \in V$, $d_R(u, v) \leq t \cdot d_G(u, v)$ for some constant t.*

We can define a ratio for such graphs to specify how close their distances are to distances in G:

Definition 2.3. *A* distance-approximating graph of ratio (t, c) *for a graph $G = (V, E)$ is a graph $R = (V', E')$ in which for every pair of vertices $u, v \in V$, $d_R(u, v) \leq t \cdot d_G(u, v) + c$.*

Sometimes, a distance-approximating graph is required to be a spanning subgraph of the main graph:

Definition 2.4. *A* distance-approximating spanner *for a graph $G = (V, E)$ is a graph for G that is also a spanning subgraph of G.*

For simpler reference to the ratios of such spanners, the following three terms have been defined in the literature:

Definition 2.5. *A* multiplicative t-spanner *of a graph $G = (V, E)$ is a spanning subgraph $R = (V, E')$ of G in which for every pair of vertices $u, v \in V$, $d_R(u, v) \leq t \cdot d_G(u, v)$.*

Definition 2.6. *An* additive c-spanner *of a graph $G = (V, E)$ is a spanning subgraph $R = (V, E')$ of G in which for every pair of vertices $u, v \in V$, $d_R(u, v) \leq d_G(u, v) + c$.*

The additive and multiplicative spanner definitions can be written in the following generalized form:

Definition 2.7. *A* (t, c)-spanner *of a graph $G = (V, E)$ is a spanning subgraph $R = (V, E')$ of G in which for every pair of vertices $u, v \in V$, $d_R(u, v) \leq t \cdot d_G(u, v) + c$.*

As far as we know, the only special graph class of distance-approximating spanners people have worked on are tree spanners:

Definition 2.8. *A* tree (t, c)-spanner *of a graph G is a tree that is also a (t, c)-spanner for G.*

Note that not all graphs have multiplicative or additive tree (t, c)-spanners for some constants t and c. For instance, a cycle of length n does not have any tree t-spanner, when $t \leq n - 2$. The only existing spanning tree in this graph is an n-vertex path. However, the distance between the first and last vertices in the path is $n - 1$, whereas they are adjacent in G. Even restricting our attention to chordal graphs does not solve the problem; for every fixed t, there exists a planar

chordal graph that does not have any multiplicative or additive tree t-spanners [9]. Brandstädt showed that for any $t \geq 4$, checking whether a tree t-spanner exists in a given chordal graph G of $\text{diam}(G) \leq t+1$ (respectively, $t+2$) for even (odd) t's is NP-complete [9]. Moreover, for any fixed $t \geq 5$, determining whether a chordal bipartite graph has a multiplicative t-spanner is NP-complete [10]. It is also known that there are distance-hereditary graphs without any additive tree 1-spanners [41].

However, for several graph classes there are algorithms finding tree spanners. We list them for future reference:

Theorem 2.9. *[46] There is a linear-time algorithm to find multiplicative tree 3-spanners for interval graphs and permutation graphs.*

Theorem 2.10. *[48] There is an algorithm to find additive tree 2-spanners for interval graphs and distance hereditary graphs.*

Theorem 2.11. *[48] There is an algorithm to find an additive tree 4-spanner for cocomparability graphs.*

Theorem 2.12. *[8] There are linear-time algorithms to find additive tree 3-spanners for strongly chordal graphs and dually chordal graphs.*

Furthermore, although the chordal graphs do not have distance-approximating spanners, the following theorem is proved for them:

Theorem 2.13. *[8] For every chordal graph G, there is a distance-approximating graph T of ratio $(1,2)$ that is also a spanning tree for G^2. Such a tree can be found in linear time.*

We will use the previous list of algorithms and this theorem later to design approximation algorithms for the MDDN problem for permutation graphs, distance-hereditary graphs, cocomparability graphs, dually chordal graphs, and chordal graphs.

2.3 Exact (Exponential-time) Algorithms

Recently, there has been growing interest in designing relatively fast exact exponential-time algorithms for NP-hard problems. Efficient exponential-time algorithms may be useful in working with small-size inputs.

In working with exponential-time algorithms, polynomial factors are not important in running times and typically are ignored in the computations; because, increasing the base by a small amount creates a significantly higher order level: for any polynomial function p, $p(n) \cdot \alpha^n \in O((\alpha + \epsilon)^n)$. Therefore, instead of the classic O notation in which constant coefficients are ignored, the notation $\tilde{\mathcal{O}}$ is used in computations dealing with exponential algorithms:

Definition 2.14. *For two functions f and g, we say that $f(n)$ is in $\tilde{\mathcal{O}}(g(n))$ if and only if $f(n) \in O(g(n) \cdot (\log g(n))^{O(1)})$.*

Note that this definition is equivalent to the following definition for exponential $g(n)$'s:

Definition 2.15. *For functions f and $g(n) = \alpha^n$, we say that $f(n)$ is in $\tilde{\mathcal{O}}(g(n))$ if and only if there exists a polynomial p such that $f(n) \in O(p(n) \cdot g(n))$.*

As we will refer to the following technique of exact algorithms later, we briefly explain it here:

The *measure and conquer* method has been recently formalized by Fomin et al. [27]. The goal is to find an upper bound on the number of acceptable solutions of every instance $\mathcal{I}$ of the problem. In this technique, a function f, called the *problem measure*, is defined to measure the *size of each problem instance* $\mathcal{I}$, and it is assumed that the desirable upper bound on the number of acceptable solutions of $\mathcal{I}$ depends only on the problem measure, i.e. $f(\mathcal{I})$. Considering all possible cases for the acceptable solutions of $\mathcal{I}$, several recursive formulas are derived to compute the number of solutions to a problem based on the number of solutions to problems of smaller sizes. Finally, to prove a bound on the time complexity of the problem, the

number of solutions for an instance of size s is assumed to be less than a parametric function such as α^n. Then, the parameters in the formulas, i.e. α in the example, are set so that assuming the bound for the problems of smaller size, the bound can be proved for the current problem size.

For further information on this field, the reader is referred to the comprehensive surveys by Woeginger [57], Schöning [54], and Fomin et al. [28].

2.4 Formal Definitions of the Problem

Before defining our problem in this section, we bring the formal definitions of relevant terms to specify our problem in terms of these known terms, making it possible to use the literature on similar problems for our problem. Then, we specify our problem precisely. At the end, we go through preliminary and known results on the problem.

In all definitions below, and throughout the whole thesis, we only deal with simple connected undirected graphs; we will mention otherwise.

As described in the introduction, our main problem is related to the domatic number and the d-domatic number of a graph. In the following, we define these two problems using the concepts of dominating set and d-dominating set. We first describe the following useful term.

Definition 2.16. *Suppose that U and W are two sets of vertices in a graph $G = (V, E)$; that is, $U, W \subseteq V$. We say that U* covers *W if and only if $W \subseteq (N(U) \cup U)$.*

A dominating set in a graph is defined as follows.

Definition 2.17. *We call a set of vertices U in $G = (V, E)$, $U \subseteq V$, a* dominating set of *G if and only if U covers $V(G)$.*

Now, we can define the domatic number of a graph:

Definition 2.18. *The* domatic number *of a graph G, denoted by $D(G)$, is the maximum number of disjoint dominating sets in G.*

To give better insight into the above-mentioned definitions, an illustrating example is shown in Figure 2.2(a). Since one of the vertices in the example graph has degree one, the number of disjoint dominating sets in this graph is at most two. Hence, the two dominating sets, specified by labels 1 and 2 in the graph, convinces us that the maximum number of disjoint dominating sets in this graph is two. In other words, the domatic number of this graph is two.

Since every dominating set has a minimal dominating subset, we can use a different, yet equivalent, definition of the domatic number: the *domatic number* of a graph G is the maximum number of disjoint *minimal* dominating sets found in G.

Changing the above-mentioned definitions slightly, we can define the d-domatic number of a graph. Rather than dealing with dominating sets, the d-domatic number is defined based on distance dominating sets. Distance dominating set is a variant of dominating set in which the covering condition is relaxed to some extent:

Definition 2.19. *For a graph $G = (V, E)$ and an integer value d, a subset U of V* d-covers *another subset W of V if and only if for every vertex w in W there is a vertex u in U such that $d(u, w) \leq d$.*

According to this definition, in Figure 2.2(b), the two vertices labeled 4 2-cover the set of all vertices in the graph.

Definition 2.20. *A set of vertices U in $G = (V, E)$, $U \subseteq V$, is called a* distance dominating set of G with distance limit d *if and only if U d-covers V.*

Generally, we use the terminology *d-dominating set* instead of distance dominating set with distance limit d. Based on this notation, the set of vertices labeled 4 in Figure 2.2(b) form a 2-dominating set for the illustrated graph.

Below, the formal definition of the d-domatic number of a given graph is presented:

Definition 2.21. *The* d-domatic number *of a graph G, denoted by d-$D(G)$, is the maximum number of disjoint [minimal] d-dominating sets in G.*

The problem of determination of the domatic (d-domatic) number of a given graph is called the *domatic (d-domatic) number problem*. The restricted version of checking whether a graph can be partitioned into k disjoint dominating sets, i.e. $\mathrm{D}(G) \geq k$, is called the *k domatic number problem*. Although finding the domatic (d-domatic) number of a graph G is itself a hard problem, most of the time it is not enough to find this count, as we want to find a partitioning as well. The problem of partitioning graph vertices into $\mathrm{D}(G)$ (d-$\mathrm{D}(G)$) disjoint dominating (d-dominating) sets is called the *domatic (d-domatic) partition problem*. A *k-partitioning* of G denotes a partitioning of $V(G)$ into k non-empty sets.

In the sample graphs shown in Figure 2.2(a), the labels of vertices specify a 2-partitioning for the illustrated graph. As explained before, the domatic number of the graph is two, and therefore, the specified 2-partitioning solves the domatic partition problem for the graph. In Figure 2.2(b), the goal is to solve the 2-domatic partition problem. Among the ten vertices in the graph, there is only one vertex, labeled 1, that 2-covers the entire set of vertices. Equivalently, except one dominating set, all dominating sets for this graph include at least two vertices. Therefore, 2-D of this graph is at most $1 + \lfloor (10-1)/2 \rfloor = 5$. The five disjoint 2-dominating sets in the image show that this maximum is possible.

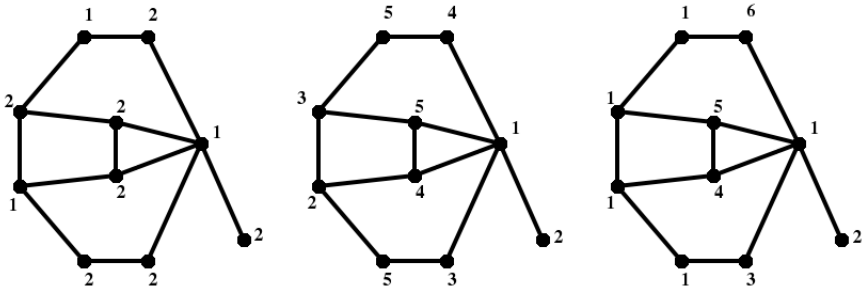
Our main problem in this thesis is computing the *minimum distance domatic number* defined in the following; we refer to this problem as the *minimum distance domatic number problem*.

Definition 2.22. *For a given graph G and an integer value k, the* minimum distance domatic number *is the minimum distance d for which d-$D(G) \geq k$.*

In is worth of mentioning that in this thesis, we will use a simplified version of this definition, which will be presented later. Throughout this thesis, we use the abbreviation MDDN rather than the minimum distance domatic number. Similarly, the MDDN problem stands for the minimum distance domatic number problem. Furthermore, we may use $\mathrm{MDDN}(G, k)$ to denote the minimum distance domatic number given G and k as inputs. We call a k-partitioning of G specifying k disjoint d-dominating sets *an optimum MDDN(G, k) partitioning*. In Figure 2.2(c), an

optimum MDDN(G, k) is found for the graph G. Note that MDDN$(G, 6) > 2$, as 2-DN$(G) = 5$. The partitioning shown in figure shows that it is possible to partition $V(G)$ into six disjoint 3-dominating sets, and as a result, MDDN$(G, 6) = 3$.

For the sake of simplicity, in the rest of this thesis we assume that every input graph G has at least k vertices. If the number of vertices in G is less than k, there cannot exist any partitioning of V into k non-empty disjoint subsets. This special case can be checked and handled separately.



(a) A domatic partition (b) A 2-domatic partition (c) An optimum MDDN$(G, 6)$ partitioning

Figure 2.2: Solving the domatic partition, the d-domatic partition, and the MDDN(G, k) problem for a sample graph G.

Using graph powers, we can simplify the previous definitions:

Definition 2.23. *The* rth power of $G = (V, E)$*, denoted by* G^r*, is a graph with the vertex set* V *in which for every pair of vertices* u, v *in* V*, there is an edge between* u *and* v *in* G^r *if and only if the distance between* u *and* v *in* G *is at most* r*.*

According to this definition, an edge between two vertices in a power of G, say G^d, is a symbol to show that the distance between these vertices in G is at most d. As a result, we can restate all previous distance-related definitions as follows:

Definition 2.24. *For a graph* $G = (V, E)$ *and an integer value* d*, a subset* U *of* V

d-covers *another subset W of V if and only if for every vertex w in W there is a vertex u in U such that $\{u, w\} \in E(G^d)$.*

Due to the definition of covering, it can also be restated in this form:

Definition 2.25. *For a graph $G = (V, E)$ and an integer value d, a subset U of V* d-covers *another subset W of V if and only if U covers W in G^d.*

Definition 2.26. *A set of vertices U in $G = (V, E)$, $U \subseteq V$, is called a* d-dominating set of G *if and only if U covers V in G^d. Or equivalently, V is a dominating set for G^d.*

Definition 2.27. *The* d-domatic number *of a graph G, denoted by d-$D(G)$, is the maximum number of disjoint [minimal] dominating sets in G^d.*

In particular, since d-$\mathrm{D}(G) \geq k$ in Definition 2.22 means that G can be partitioned into k disjoint d-dominating sets, the new definition of d-dominating set helps us restate the definition of $\mathrm{MDDN}(G, k)$ in the following intuitive form:

Definition 2.28. *For graph G and positive integer k,* $\mathrm{MDDN}(G, k)$ *is the minimum d for which G^d can be partitioned into k disjoint [minimal] dominating sets. In other words, MDDN(G, k) is the minimum d for which $D(G^d) \geq k$.*

The latter definition gives a lower bound for $\mathrm{MDDN}(G, k)$ based on the domatic number of G^r, and clarifies the strong relationship between the MDDN problem and the domatic number problem.

Definition 2.29. An optimum $\mathrm{MDDN}(G, k)$ partitioning *is a k-partitioning of G that specifies k disjoint dominating sets for $G^{\mathrm{MDDN}(G,k)}$.*

In Section 2.4.1, we will see that the MDDN problem is an NP-hard problem. To deal with the difficulty of this problem, it helps us to note that this problem fits in the framework of optimization problems mentioned in Section 2.2: in the MDDN problem, I is a graph-integer pair; for an input instance $x \in I$, $s(x)$ is the set of all k-partitionings of G, and $m(x, s(x))$ is the maximum distance between

a vertex and a partition in x, as defined in page 11; and g is the min function. We use OPT(MDDN(G, k)) to denote the optimum for this instance of the MDDN problem.

2.4.1 Preliminary and Known Results

In this section, we go through a number of observations, simple lemmas, and known facts for the MDDN problem that are needed in the next chapters. Note that we always assume that the input graph is simple and connected. Also, whenever we are working with the MDDN problem, we assume that the number of vertices in the input graph is at least k.

In a complete graph, every subset of V is a dominating set. As a result, in every k-partitioning, all vertices in the same partition form a dominating set in G^1. Hence, due to Definition 2.28, the MDDN for any K_n is 1. Note that for any graph G, $G^n = K_n$. Therefore, the MDDN of any graph is at most n. Similarly, there are n disjoint dominating sets in $G^{\text{diam}(G)} = K_n$. Therefore, the minimum d for which (G^d) has k disjoint dominating sets is at most diam(G), which leads us to the first upper bound on MDDN(G, k):

Observation 2.30. *For every graph G and any number k, MDDN$(G, k) \leq$ diam(G).*

A result by Zelinka on the d-domatic number of an arbitrary graph proves another upper bound on MDDN(G, k). He proved that $(k-1)$-D$(G) \geq \min\{n, k\}$ [59]. Equivalently, if a graph G has at least k vertices, G^{k-1} can be partitioned into k disjoint dominating sets. As we have assumed the number of vertices in the input graph of the MDDN problem is at least k, Zelinka's bound results in the following upper bound on MDDN(G, k).

Theorem 2.31. *[59] For every graph G and any number k, MDDN$(G, k) \leq k - 1$.*

Therefore, MDDN(G, k) is always bounded from above by $k - 1$ which does not depend on the size of G.

Since for some G's $\mathrm{MDDN}(G,k)$ is at least $k-1$, this bound is tight, in the sense that it cannot be improved even to $\mathrm{MDDN}(G,k) \leq k-2$; for instance, using the following observation, we can show $\mathrm{MDDN}(G,k) \geq k-1$ for P_k:

Observation 2.32. *For every graph G, $D(G) \leq \delta(G)+1$.*

Proof. Each vertex $v \in V(G)$ cannot be dominated by more than $deg_G(v)+1$. Therefore, the maximum number of disjoint dominating sets in a graph G is at most $\delta(G)+1$, resulting in $\mathrm{D}(G) \leq \delta(G)+1$. □

In P_k, $\delta(G^{k-2})+1 = (k-2)+1 = k-1$. Therefore, the maximum possible domatic number of G^{k-2}, i.e. $\delta(G^{k-2})+1$, is at most $k-1$. Hence, according to Definition 2.28 of $\mathrm{MDDN}(G,k)$, $\mathrm{MDDN}(G,k) \geq k-1$.

We can also obtain a lower bound for $\mathrm{MDDN}(G,k)$ that will help us prove an approximation ratio for our greedy approximation algorithm in Chapter 6:

Lemma 2.33. *For any graph G and positive integer k, the minimum r satisfying $k-1 \leq \delta(G^r)$ is at most MDDN(G,k).*

Proof. Based on Observation 2.32, we know that $\mathrm{D}(G) \leq \delta(G)+1$. Due to Definition 2.28, $\mathrm{MDDN}(G,k)$ is the minimum d satisfying $\mathrm{D}(G^d) \geq k$. Therefore, $k \leq \mathrm{D}(G^{\mathrm{MDDN}(G,k)}) \leq \delta(G^{\mathrm{MDDN}(G,k)})+1$. As a result, the minimum r satisfying $k \leq \delta(G^r)+1$ is at most $\mathrm{MDDN}(G,k)$. □

In the remainder of this thesis, in several cases, we will use the known algorithms on the domatic number problem for our problem. Therefore, we need the next observations and lemmas that explain the relationship between the time complexity of algorithms in the MDDN and the domatic number problems.

According to Definition 2.28, $\mathrm{MDDN}(G,k)$ is the minimum d making the condition $\mathrm{D}(G^d) \geq k$ true. Therefore $\mathrm{MDDN}(G,k) > 1$ implies that $\mathrm{D}(G) < k$; since otherwise, the minimum d satisfying $\mathrm{D}(G^d) \geq k$ would be 1, and hence, $\mathrm{MDDN}(G,k) = 1$ which is a contradiction. In addition, if $\mathrm{MDDN}(G,k) = 1$, then

there are k disjoint dominating sets in G, and hence, $\mathrm{D}(G) \geq k$. Therefore, the domatic number of a graph G can be compared to k using an algorithm that computes MDDN(G, k).

Observation 2.34. *Suppose that there exists an algorithm solving the MDDN(G, k) problem for a graph G and a fixed integer k within time $t(n)$, where n is the number of vertices in G. Then, we can decide on whether $D(G) \geq k$ in $t(n)$ time.*

As a result, if we have an algorithm solving the MDDN(G, k) problem for a graph G and any k within time $t(n)$, we can find out whether $\mathrm{D}(G) \geq k$ for any k in $t(n)$ time. Then, to calculate $\mathrm{D}(G)$ we can examine $\mathrm{D}(G) \geq k$ for $k = 1, 2, \ldots, n$, and find the maximum number k for which $\mathrm{D}(G) \geq k$. Therefore, the following observation holds:

Observation 2.35. *Suppose that we can solve the MDDN(G, k) problem for a graph G and every integer $1 \leq k \leq n$ within time $t(n)$, where n is the number of vertices in G. Then, we can compute the domatic number of G within time $n \cdot t(n)$.*

Therefore, if the domatic number problem is NP-complete for a family of graphs $\mathcal{F}$, the MDDN problem cannot be solved for all graphs in $\mathcal{F}$ and all k's in polynomial time.

Observation 2.36. *If the domatic number problem is NP-complete for a family of graphs $\mathcal{F}$, the MDDN problem is also NP-complete for $\mathcal{F}$.*

According to the NP-completeness results for the domatic number problem mentioned in Chapter 1, the MDDN problem is NP-complete for the following graph classes:

Theorem 2.37. *[6, 38] The MDDN problem is NP-complete for circular-arc graphs, bipartite graphs, comparability graphs, split graphs, and chordal and cochordal graphs.*

Although an algorithm solving the MDDN problem for general k's can be used to solve the domatic number problem, for the reverse direction we need the domatic number of the input graph and its power graphs. More formally, the following lemma holds:

Lemma 2.38. *Suppose that we can determine the domatic number of a graph G and its powers within time $t(n)$, where n is the number of nodes in G. More precisely, there is an algorithm $\mathcal{A}$ that determines whether the domatic number of G^r is exactly k, for every k and r, within time $t(n)$. Then, MDDN(G,k) can be computed in $\log(k) \cdot t(n)$ time.*

Proof. According to Definition 2.28, to compute $\mathrm{MDDN}(G,k)$ we can examine whether $\mathrm{D}(G^d) \geq k$ for $d = 1, 2, \ldots, n$ and find the minimum d satisfying this condition. If checking the condition $\mathrm{D}(G^d) \geq k$ for any d and k can be done in $t(n)$ time, the whole process takes at most $n \cdot t(n)$ time.

This time bound can be improved further: considering the $\mathrm{MDDN}(G,k) \leq k-1$ bound stated in Theorem 2.31, we can confine our search to $d = 1, 2, \ldots, k-1$, resulting in the running time $k \cdot t(n)$; moreover, we can use binary search to find the minimum power d satisfying ($\mathrm{D}(G^d) \geq k$ and $\mathrm{D}(G^{d-1}) < k$) resulting in $2\log(k-1)$ domatic number checks, which improves the running time to $O(\log(k) \cdot t(n))$. □

In this lemma, it is assumed that the power graphs are not part of the input to the algorithm $\mathcal{A}$, and this algorithm is responsible to compute its required power graphs. In the following lemma we modify this requirement, and include the computations required to construct power graphs in the final running time.

Lemma 2.39. *Suppose that we can solve the domatic number problem for any given power of a graph G within time $f(n)$, where n is the number of nodes in G. Then, MDDN(G,k) can be computed in $O(n^3 + \log(k) \cdot f(n))$ time.*

Proof. Using Floyd-Warshall algorithm, we can compute the shortest paths between all pairs of vertices in G in time $O(n^3)$ [18]. Once we compute the distances, any power of G, say G^r, can be constructed in time $O(n^2)$ by examining $d(v_i, v_j) \leq r$ for every pair of vertices $v_i, v_j \in V(G)$. Therefore, using the same binary search used in the previous lemma, the running time will be changed to

$$O(n^3) + \log(k) \cdot (f(n) + O(n^2)) \in O(n^3 + \log(k) \cdot (f(n) + n^2)).$$

Since $k \leq n$, we have:

$$O(n^3 + \log(k) \cdot f(n) + \log(n) \cdot n^2) \in O(n^3 + \log(k) \cdot f(n)).$$

Therefore, we can compute $\mathrm{MDDN}(G, k)$ in the required running time. □

Ignoring polynomial factors, we can conclude the lemma below.

Lemma 2.40. *Suppose that there is an $\tilde{\mathcal{O}}(\alpha^n)$ time algorithm for the domatic number problem on general graphs. Then, for every given graph G and every value k, the problem of computing MDDN(G, k) can be solved in time $\tilde{\mathcal{O}}(\alpha^n)$.*

Proof. Due to the previous lemma, the assumptions in this lemma enables us to compute $\mathrm{MDDN}(G, k)$ in time $O(n^3 + \log(k) \cdot \alpha^n)) \in O(n^3 + \log(n) \cdot \alpha^n) \in O(n^3 \cdot \alpha^n) = cn^3 \cdot \alpha^n$ for some constant c. Therefore, ignoring the polynomial factor cn^3, this running time is in $\tilde{\mathcal{O}}(\alpha^n)$. □

Consequently, any exact algorithm for the domatic number problem implies an exact algorithm for the MDDN problem with the same exponent.

In the next two lemmas, we study the relationship between the MDDN problem and the domatic number problem for the case $k = 3$.

Lemma 2.41. *For every graph G, the condition $D(G) \geq 3$ holds if and only if MDDN$(G, 3) = 1$.*

Proof. If $\mathrm{D}(G) \geq 3$, then due to Definition 2.28 of $\mathrm{MDDN}(G, 3)$, $\mathrm{MDDN}(G, 3) = 1$. Otherwise, according to the same definition of $\mathrm{MDDN}(G, 3)$, $\mathrm{MDDN}(G, 3) > 1$. Therefore, deciding on whether $\mathrm{D}(G) \geq 3$ determines whether $\mathrm{MDDN}(G, 3) = 1$. □

The following result is a consequence of the previous lemma and the upper bound $\mathrm{MDDN}(G, k) \leq k - 1$ mentioned in Theorem 2.31:

Lemma 2.42. *Suppose that we can decide whether $D(G) \geq 3$ in time $t(n)$ for a graph G. Then, MDDN$(G, 3)$ can also be computed in time $t(n)$.*

Combining this result with Observation 2.34 we can see that the three domatic number problem, which is the problem of determining whether $\mathrm{D}(G) \geq 3$, is equivalent to the problem of computing $\mathrm{MDDN}(G, 3)$:

Corollary 2.43. *The three domatic number problem for a family of graphs $\mathcal{F}$ is equivalent to the problem of computing MDDN$(G, 3)$ for any graph $G \in \mathcal{F}$.*

As a result, we can conclude the following statement for exact algorithms of these problems:

Corollary 2.44. *Suppose that there is an $\tilde{\mathcal{O}}(\alpha^n)$ time algorithm for the three domatic number problem on general graphs. Then, for every given graph G, the problem of computing MDDN$(G, 3)$ can be solved within time $\tilde{\mathcal{O}}(\alpha^n)$.*

Thus, any exact (exponential) algorithm solving the three domatic number problem gives an exact (exponential) algorithm with the same base. In Chapter 5, we will use this fact to develop an exact algorithm for the MDDN problem by improving the running time of Fomin et al.'s algorithm [26] for the three domatic number problem.

Our exact algorithm uses an enumeration algorithm developed by Fomin et al. to enumerate all minimal dominating sets of a graph. Since in this enumeration the problem of computing a minimal dominating set is viewed as a restricted *set cover problem*, we briefly explain this relationship: an instance of a set cover problem is a pair $(U, \mathcal{S})$, where U is a set of elements and $\mathcal{S}$ is a family of non-empty subsets of U. A *minimal set cover* is a subset $\mathcal{C}$ of $\mathcal{S}$ such that for every element $u \in U$ there is a set $S \in \mathcal{C}$ containing u. Furthermore, no subset of $\mathcal{C}$ has this property. From this perspective, a minimal dominating set of G corresponds to a minimal set cover in the set cover problem $(V(G), \cup_{v_i \in V(G)}(v_i \cup N(v_i)))$. More formally, replacing each set of the form $v_i \cup N(v_i)$ in a minimal set cover with its corresponding vertex v_i produces a minimal dominating set in G, and replacing each vertex v_i in a minimal dominating set for G with the set $v_i \cup N(v_i)$ gives a minimal set cover in the set cover problem $(V(G), \cup_{v_i \in V(G)}(v_i \cup N(v_i)))$. Note that in this framework a set of disjoint minimal dominating sets is mapped to a set of disjoint minimal set covers.

Chapter 3

NP-hardness Results and Polynomial Time Algorithms

Traditionally, one way to show that a problem is difficult is proving its NP-completeness. In the preliminaries section (Theorem 2.37), the MDDN problem was shown to be NP-complete in general. The first question that arises is that will the MDDN problem remain NP-complete for restricted inputs? For instance, will it be difficult to compute MDDN(G, k) for fixed k's or for special families of graphs? In this chapter, we first study the complexity of the MDDN problem for various k's. We show that except the $k = 1$ and $k = 2$ cases, which can be solved in linear time, the MDDN problem is NP-complete for any fixed k. Then, we concentrate on different graph classes for which the problem remains NP-complete. Finally, we explain how the problem can be solved in polynomial time for strongly chordal graphs and bounded tree-width graphs.

3.1 NP-complete Cases

In this section, we prove the NP-hardness of the problem of computing MDDN(G, k) for fixed k's greater than 3 and for various graph classes. But first, we show that for $k = 1, 2$ MDDN$(G, k) = 1$.

Obviously, the set of all vertices in a graph G is a dominating set for G:

Observation 3.1. *For every graph G, MDDN$(G, 1) = 1$.*

The next fact, which is less trivial, is that MDDN$(G, 2) = 1$ for any graph that does not have isolated vertices. This is a direct result of the following lemma, which is one of the first results proved in the literature of dominating sets. Since the lemma is used several times in this thesis, we present its proof. Recall from the preliminaries that we are only studying the MDDN problem on simple connected input graphs, which does not include any graph with isolated vertices.

Lemma 3.2. *[36, Theorem 1.2]For every graph $G = (V, E)$ without isolated vertices, the complement of any minimal dominating set S is also a dominating set.*

Proof. If $V - S$ is not a dominating set for G, then there is a vertex $v \in S$ without any neighbors in $V - S$. Therefore, every vertex in $V - S$ is covered by a vertex in S other than v. As a result, $V - S$ is also covered by $S - v$. We claim that $S - v$ is still a dominating set for G, which contradicts the minimality of S. We know that $(S - v) \cup (V - S)$ is covered by the vertices in $S - v$. It only remains to check the status of v. As G does not have an isolated vertex, v is connected to at least one other vertex $u \in S$. Thus, v is also covered by $S - v$. The resulting contradiction proves the lemma. □

Corollary 3.3. *If graph G does not have any isolated vertices, MDDN$(G, 2) = 1$.*

In the rest of this chapter, we sometimes need to extend a complexity result for the $k = 3$ case to any fixed $k > 3$.

Lemma 3.4. *The problem of determining whether MDDN$(G, 3) = 1$ can be reduced to the problem of determining whether MDDN$(G, k) = 1$ for any k greater than 3.*

Proof. We prove that MDDN$(G, 3) = 1$ if and only if MDDN$(G', k) = 1$ for the graph G' constructed by connecting the vertices of a K_{k-3} to all vertices of G. Note that each of the added $k - 3$ vertices is a dominating set for G'. In addition,

all previous vertices in G' are now connected to these new vertices. Hence, every dominating set in G is a dominating set for G'. Consequently, if $\mathrm{MDDN}(G,3)=1$, due to Lemma 2.41 $\mathrm{D}(G)\geq 3$, and therefore, $\mathrm{D}(G')\geq 3+(k-3)=k$, resulting in $\mathrm{MDDN}(G',k)=1$. To prove the reverse direction, we assume that $\mathrm{MDDN}(G',k)=1$. Then, $\mathrm{D}(G')\geq k$. Therefore, $V(G')$ can be partitioned into k disjoint dominating sets. Since these partitions are disjoint, at least three partitions among them do not have any vertex from the K_{k-3}. As a result, these three partitions also specify three disjoint dominating sets in G. Thus, $\mathrm{D}(G)\geq 3$, and $\mathrm{MDDN}(G,3)=1$. □

Corollary 3.5. *The problem of calculating MDDN$(G,3)$ can be reduced to the problem of calculating MDDN(G,k) for any k greater than 3.*

Proof. As mentioned in Chapter 2, based on a result by Zelinka [59], $\mathrm{MDDN}(G,k)\leq k-1$. Therefore, $\mathrm{MDDN}(G,3)\leq 2$.

Consequently, based on the previous lemma, to decide between the two possible cases $\mathrm{MDDN}(G,3)=1$ and $\mathrm{MDDN}(G,3)=2$ possible cases, it suffices to check whether $\mathrm{MDDN}(G',k)=1$ or $\mathrm{MDDN}(G',k)\geq 2$. □

Due to Theorem 2.37, the MDDN problem is NP-complete in general.In the following, we reduce the k-colorability problem to the MDDN problem, resulting the NP-completeness of the problem of computing $\mathrm{MDDN}(G,k)$ for any fixed $k\leq 3$. In addition, we will show that the graph constructed in the reduction has some interesting properties which helps us to prove the NP-completeness of computing $\mathrm{MDDN}(G,k)$'s for several special families of graphs, for restricted k's. Moreover, we can prove that the domatic number problem is NP-complete for planar bipartite of maximum degree four, which was not known before.

Theorem 3.6. *Suppose that G is a graph and k is a fixed integer greater than two. Then, the problem of computing MDDN(G,k) is NP-hard.*

Proof. We prove the result based on a reduction from the k-colorability problem, one of the first problems proved to be NP-complete [39]; the k-colorability problem is the problem of determining whether there exists a coloring for a given graph G

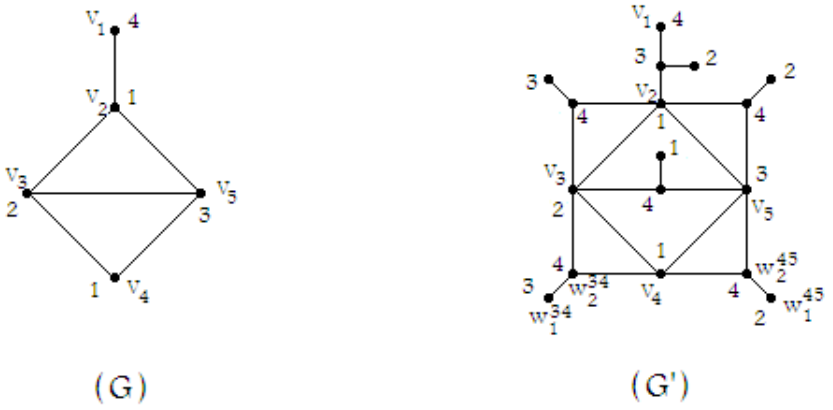


Figure 3.1: An example G' in the reduction from k-colorability to the MDDN problem

with k colors such that each set of vertices colored with the same color is an independent set. We show that k-colorability(G) reduces to the problem of determining whether MDDN$(G', k) \leq k-2$ for the graph G' constructed as follows: for every vertex $v_i \in V(G)$, we put a vertex v'_i in G'. For every edge $(v_i, v_j) \in E(G)$, we construct a path of length $k-2$ in G' using new vertices $w_1^{ij}, w_2^{ij}, \ldots, w_{k-2}^{ij}$. Then, we connect w_{k-2}^{ij} to v'_i and v'_j. The number of vertices and edges in G' are at most $O(kn^2)$ and $O(km)$. The corresponding G' for a sample G is shown in Figure 3.1, where, for example, we have added the set of new vertices $\{w_1^{34}, w_2^{34}\}$ to G' with respect to the edge $\{v_3, v_4\}$ in G.

In the case that G is k-colorable, there exists a k-coloring for $v_1, v_2, \ldots, v_n$. Assume that this coloring assigns color $C(v_i)$, $1 \leq C(v_i) \leq k$, to vertex v_i, for every $1 \leq i \leq n$. Based on this coloring, we put the vertices of G' into partitions $\{1, 2, \ldots, k\}$ such that each partition forms a $(k-2)$-dominating set for G'. First, we put every vertex v'_i, $1 \leq i \leq n$, in partition $C(v_i)$. Then, for each $\{v_i, v_j\} \in E(G)$ with colors $1 \leq C(v_i) \leq C(v_j) \leq k$, we assign vertices $w_1^{ij}, w_2^{ij}, \ldots, w_{k-2}^{ij}$ to partitions $1, 2, \ldots, C(v_i)-1, C(v_i)+1, \ldots, C(v_j)-1, C(v_j)+1, \ldots, k$, respectively. In Figure 3.1, a 4-coloring for a graph G and its resulted partitioning in G' is specified; the numbers beside vertices in G' represent the partitions they are assigned to. It is easy to verify that we have partitioned $V(G')$ into k disjoint $(k-2)$-dominating sets. The optimal solution of MDDN(G', k) is the minimum d that allows a partitioning of

$V(G')$ into k disjoint d-dominating sets. Hence, if G is k-colorable, $\text{MDDN}(G', k) \leq k-2$. Note that the number of vertices within distance $k-3$ from the w_1^{ij} vertices, for all $(v_i, v_j) \in E(G)$, is less than k. Hence, $\text{D}(G'^{k-3}) < k$. Based on Definition 2.28, $\text{MDDN}(G', k)$ is at least $k-2$. Therefore, $\text{MDDN}(G', k)$ is exactly $k-2$ whenever G is k-colorable.

In the next step, we prove that the converse is also true: if $\text{MDDN}(G', k) = k-2$, G is k-colorable. Considering the neighborhood of vertices w_1^{ij}, for every $\{v_i, v_j\} \in E(G)$, there are only k vertices within distance $k-2$ from them. Therefore, to have all the k partitions in $N_{G^{k-2}}(w_1^{ij})$, all these vertices must be in different partitions. Note that v_i' and v_j' are both in $N_{G^{k-2}}(w_1^{ij})$. As a result, v_i' and v_j' must be in different partitions. This statement holds for every $\{v_i, v_j\} \in E(G)$. Consequently, the partition numbers assigned to $v_1', v_2', \ldots, v_n'$ can be viewed as a k-coloring for G, coloring the endpoints of every edge with different colors. In other words, if such a partitioning exists, i.e. $\text{MDDN}(G', k) = k-2$, G is k-colorable. □

Note that we reduced the k-colorability problem to the problem of checking whether $\text{MDDN}(G', k) = k-2$ for a constructed G'. Hence, even this restricted case of the MDDN problem is NP-complete.

Corollary 3.7. *For a given graph G and any fixed integer $k \geq 3$, the problem of determining whether MDDN$(G, k) = k-2$ is NP-complete.*

A consequence of the above-mentioned corollary is that the decision problem of determining whether $\text{MDDN}(G, 3) = 1$ is an NP-complete problem. Using Lemma 3.4, we can conclude the following corollary. The NP-completeness of this restricted version will help us to prove an inapproximability result in Section 6.

Corollary 3.8. *For any fixed integer $k \geq 3$, it is NP-complete to decide whether MDDN$(G, k) = 1$.*

Providing a polynomial-time reduction from instances of an NP-complete problem $\mathcal{P}$ to a set of instances $\mathcal{I}$ of another problem $\mathcal{Q}$ proves that $\mathcal{Q}$ is NP-complete even if its domain is restricted to $\mathcal{I}$. Therefore, the k-colorability reduction to the

problem of determining whether $\text{MDDN}(G', k) = k - 2$ proves that the MDDN problem is NP-complete even if we restrict the input graphs to all constructible G''s. In the following corollaries we investigate the properties of G''s produced from G's in various graph classes to prove further NP-completeness results for the MDDN problem.

Corollary 3.9. *The problem of determining whether MDDN$(H, 3) = 1$ is NP-complete for planar bipartite graphs of maximum degree 4.*

Proof. It turns out that the graph G' constructed in the reduction is a bipartite graph: coloring v_i's, for every $1 \leq i \leq n$, with color 1, and $w_{k-2}^{ij}, w_{k-3}^{ij}, w_{k-4}^{ij}, \ldots, w_1^{ij}$, for every $1 \leq i, j \leq n$, with colors 2 and 1 alternatively, gives a perfect 2-coloring for G', and proves that G' is bipartite. Furthermore, the degree of every vertex v_i' in G' is equal to the degree of its corresponding vertex v_i in G, while the degrees of w_ℓ^{ij}'s are at most 3. Therefore, if $\Delta(G) \geq 3$, $\Delta(G') = \Delta(G)$.

Moreover, for the $k = 3$ case, G' is the subdivision graph of G. As a result, if G is a planar graph, G' is also planar, because inserting a point in the middle of each edge in a drawing of G gives a drawing for G'.

Since 3-coloring is NP-complete for planar graphs of maximum degree 4 [32], we can choose G from this family of graphs, and prove the NP-completeness of deciding whether $\text{MDDN}(G', 3) = 1$ for the family of constructed G''s. As explained above, G' is always bipartite, $\Delta(G') = \Delta(G) = 4$, and is planar for planar G's. Therefore, the problem of deciding whether $\text{MDDN}(G', 3) = 1$ is NP-complete, when G' is restricted to be a planar bipartite graph of maximum degree 4. □

Using this corollary, we prove the NP-completeness of the domatic number problem for planar bipartite graphs of maximum degree four, which, to the best of our knowledge, was not known before. if $\text{D}(G) \geq 3$ can be decided in polynomial time for every G in a graph class $\mathcal{F}$, then $\text{MDDN}(G, 3) = 1$ can also be checked in polynomial time. Therefore, the NP-completeness of determining whether $\text{MDDN}(G, 3) = 1$ for $G \in \mathcal{F}$ proves that specifying whether $\text{D}(G) \geq 3$ is also NP-complete when restricting the input graph to the family $\mathcal{F}$. Since the

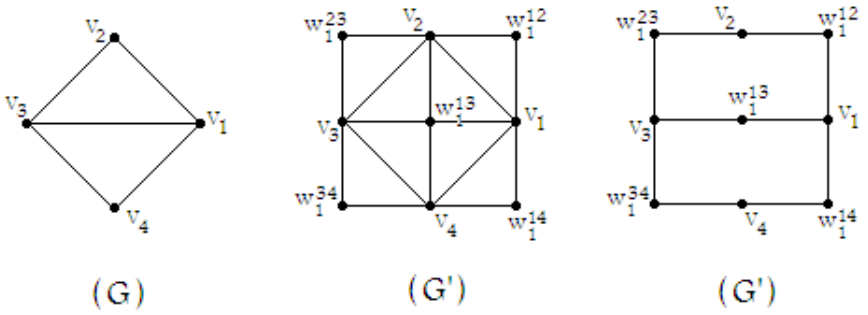


Figure 3.2: The reduction used in Corollary 3.11 and Corollary 3.12 versus the reduction in Corollary 3.9 for a sample graph G. Since these reductions only differ in the G' produced, we have not shown the coloring in G and partition numbers in G''s.

problem of checking $\mathrm{D}(G) \geq 3$ is an special case of the domatic number problem, this corollary leads us to the following result:

Corollary 3.10. *The domatic number problem is NP-complete for planar bipartite graphs of degree four.*

An interesting property in the reduction from the k-colorability problem is that the reduction does not depend on the edges between v_i''s in G'. As stated in the proof, the subgraph C of G' induced by the set $\{v_1', v_2', \ldots, v_n'\}$ of vertices is the n-vertex empty graph. Replacing C with any n-vertex graph, the proof still works, and the k-coloring problem reduces to the problem of checking whether $\mathrm{MDDN}(G', k) = k - 2$.

Exploiting this flexibility in the proof, we can prove the NP-completeness of the MDDN problem for special graph classes split graphs and 2-connected graphs.

Corollary 3.11. *For any fixed $k \geq 3$, the problem of determining whether $MDDN(G, k) = 1$ is NP-complete for split graphs.*

Proof. First, we note that in the reduction of Lemma 3.4, if G is a split graph, the new graph G' is also a split graphs; in this reduction G' is constructed by

connecting a K_{k-3} to G. Due to the definition of split graphs, $V(G)$ can be partitioned into X and Y where X is a clique and Y is an independent set. In the new graph G', Y is still an independent set, and $X + V(K_{k-3})$ produces a new clique. Thus, the new graph is a split graph. Therefore, Lemma 3.4 holds when the input graphs for both problems are confined to split graphs: the problem of determining whether $\text{MDDN}(G, 3) = 1$ for split G's can be reduced to the problem of determining $\text{MDDN}(G, k)$ for any k greater than 3 and split G's.

Consequently, it is sufficient to prove the corollary for the $k = 3$ case. As mentioned before, the reduction from the k-colorability problem works even if we replace the subgraph C of G' induced by the set $\{v'_1, v'_2, \ldots, v'_n\}$ of vertices by an arbitrary n-vertex graph. The reduction is shown in the third image in Figure 3.2 for a sample graph G and can be compared to the reduction used in Corollary 3.9 shown in the second image. Replacing C with K_n leads to a split graph G'. Thus, the 3-colorability problem reduces to determining whether $\text{MDDN}(G, 3) = 1$ for split G's, proving the NP-completeness of the corollary for $k = 3$. □

The proof for the 2-connected graphs is very similar:

Corollary 3.12. *For any fixed $k \geq 3$, the problem of determining whether MDDN$(G, k) = 1$ is NP-complete for 2-connected graphs.*

Proof. The input graph to the MDDN problem for this corollary has at least $k \geq 3$ vertices.

We first prove that the reduction in Lemma 3.4 preserves 2-connectivity, for any input graph that has at least 3 vertices. Suppose that the input graph G is 2-connected. We should prove that connecting a K_{k-3} to G keeps 2-connectivity. Any pair of vertices in G do still have their two vertex-disjoint paths connecting them. Any pair of vertices in the added K_{k-3} are adjacent, i.e. have a path of length 1 to each other, and also have a path of length 2 through an arbitrary vertex in G. Any vertex v in G and any vertex in the K_{k-3} have a path of length 1 to each other and also a path of length 2 through a neighbor of v in G. Note that such a neighbor can be found, since G is 2-connected and has at least three vertices, and

therefore, to have two vertex-disjoint paths between v and another vertex in G, v needs to have at least two neighbors.

Therefore, it is sufficient to prove the corollary for the $k = 3$ case for graphs that have at least three vertices. Now we prove that replacing the subgraph C of G' induced by $\{v'_1, v'_2, \ldots, v'_n\}$ with K_n in the reduction in Theorem 3.6 leads to a 2-connected graph G' with at least the same number of vertices of G; therefore, the NP-completeness of the 3-colorability problem for graphs of at least three vertices proves the NP-completeness of the corollary for $k = 3$.

For the case $k = 3$, G' is constructed by adding a new vertex for each edge $\{u, v\} \in E(G)$ and connecting it to both u and v. Therefore, the number of vertices in G' is at least the same as the number of vertices in G. In addition, since C is set to K_n and $n \geq 3$, every pair of vertices in the C part have a path of length one and also a path of length two via another vertex in C. For any newly added vertex w corresponding to an edge $\{u, v\} \in E(G)$, and any vertex in the C part, there are two vertex-disjoint paths of length two, one via u and the other one via v, connecting them. Finally, for every pair of newly added vertices u and v corresponding to edges $\{u_1, u_2\}$ and $\{v_1, v_2\}$ in $E(G)$, since the corresponding edges are not the same, at least one of u_1, u_2 is not equal to v_1 and v_2, say $u_1 \neq v_1, v_2$. If u_1, u_2, v_1, and v_2 are all different, the two paths connecting u and v via (u, u_1, v_1, v) and (u, u_2, v_2, v) are vertex-disjoint and we are done. Otherwise, u_2 is the same as v_1 or v_2, say v_2. Now, the two paths (u, u_1, v_1, v) and (u, u_2, v) are two vertex-disjoint oaths connecting u and v. Therefore, G' is 2-connected, and the corollary is proved. □

Other settings for the G' edges between $\{v'_1, v'_2, \ldots, v'_n\}$ in the reduction may result in recognizing more graph classes for which the problem is NP-complete. However, the NP-completeness of the MDDN problem for planar bipartite graphs of $\Delta \leq 4$ should not lead us to the conclusion that the MDDN problem is difficult even for simple graph families. In the next section, we will see the problem can be solved in polynomial time for bounded tree-width graphs and strongly chordal graphs.

3.2 Polynomial-time Solvable Cases

In the following two sections we prove that the MDDN problem is solvable in polynomial time for bounded tree-width graphs and strongly chordal graphs, including trees, interval graphs, block graphs, and directed path graphs.

3.2.1 Solving the MDDN Problem in Polynomial Time for Strongly Chordal Graphs

Strongly chordal graphs were first defined by Farber as a restricted class of chordal graphs for which a generalized dominating set problem, called "the weighted dominating set problem", can be solved polynomially. Later, more dominating-related problems, proved to be NP-complete for chordal graphs, were solved in polynomial time for strongly chordal graphs. For instance, although the domatic number problem is NP-complete even for circular-arc graphs [6], bipartite graphs, and chordal graphs [38], the domatic number of every strongly chordal graph can be computed in linear time. More precisely, the following two theorems are known:

Theorem 3.13. *[24] For every strongly chordal graph G, $D(G) = \delta(G) + 1$.*

Theorem 3.14. *[38] For every strongly chordal graph, a domatic partition can be found in linear time.*

The similarity between the MDDN problem and the domatic number problem motivates us to check whether the MDDN problem can be solved polynomially for strongly chordal graphs. In the following, we show how a polynomial-time algorithm is designed for the problem combining the following known theorem with the above-mentioned results:

Theorem 3.15. *[45] Any power of a strongly chordal graph is strongly chordal.*

Therefore, due to Theorem 3.13, we have:

Corollary 3.16. *Suppose that G is strongly chordal graph. Then, for any positive integer i, $1 \leq i \leq n$, $D(G^i) = \delta(G^i) + 1$.*

Using this corollary, we can compute the domatic number of any given power of G in time $O(n^2)$: since the minimum degree of a given graph can be specified in time $O(n^2)$, for every given power graph of G, say G^i, $\mathrm{D}(G^i) = \delta(G^i) + 1$ can be calculated in time $O(n^2)$. As a result, we can prove the following theorem for strongly chordal graphs:

Theorem 3.17. *For every strongly chordal graph G and positive integer k, MDDN(G, k) and an optimum MDDN(G, k) partitioning can be found in $O(n^3)$ time.*

Proof. We can use Lemma 2.39 for strongly chordal graphs and set the required function $f(n)$ to $O(n^2)$. Consequently, we can compute $\mathrm{MDDN}(G, k)$ for every strongly chordal G in time $O(n^3 + \log(k) \cdot n^2) \in O(n^3)$. Moreover, after finding $\mathrm{MDDN}(G, k)$, due to Theorem 3.14, a domatic partition for $G^{\mathrm{MDDN}(G,k)}$ can also be found in linear time. According to Definition 2.29, this partitioning is an optimum $\mathrm{MDDN}(G, k)$ partitioning. Therefore, we can compute an optimum $\mathrm{MDDN}(G, k)$ partitioning in time $O(n^3) + O(n) \in O(n^3)$. □

In a closer look at the properties of the domatic number of strongly chordal graphs, we can improve this running time to $O(nk^2)$:

Theorem 3.18. *For every strongly chordal graph G and positive integer k, MDDN(G, k) and an optimum MDDN(G, k) partitioning can be found in $O(nk^2)$ time.*

Proof. For every vertex v in G, we find the minimum distance ℓ such that there are at least $k - 1$ vertices within distance ℓ of v. Then, we find the maximum among all computed ℓ's. This maximum value is the minimum distance d for which all vertices in G^d have degree at least $k - 1$, and hence $\delta(G^d) \geq k - 1$. Equivalently, it is the minimum d satisfying $\mathrm{D}(G^d) = \delta(G^d) + 1 \geq k$. Due to Definition 2.28, this value is $\mathrm{MDDN}(G, k)$. Therefore, this algorithm calculates $\mathrm{MDDN}(G, k)$. As in the previous theorem, after computing $\mathrm{MDDN}(G, k)$, we can find an optimum $\mathrm{MDDN}(G, k)$ partitioning using Theorem 3.14.

In order to compute ℓ for each vertex $v \in V(G)$, we use a breadth-first search [18]. However, we keep track of the depth we are searching in. We also count the number of times we visit new vertices, including v at the first step, in the BFS and stop the BFS anytime this count reaches k. At the end, we return the current depth.

At each step of the BFS, a new edge connecting one of the visited vertices $u \in V(G)$ to another vertex $w \in V(G)$ is traversed. The vertex w is among the vertices visited before, or it is added to the visited vertices. Therefore, the number of steps in this BFS is of the order of the number of edges between the k vertices the algorithm visits: $O(k^2)$.

Consequently, performing the BFS algorithm for all vertices in $V(G)$, takes $O(nk^2)$ time. In addition, the maximization step and computing an optimum MDDN(G, k) partitioning takes $O(n)$ time. Thus, the overall running time would be $O(nk^2 + n) \in O(nk^2)$. □

Since trees, interval graphs, block graphs, and directed path graphs are all strongly chordal graphs [23], MDDN(G, k) can be computed in polynomial time for all of them.

3.2.2 Solving The MDDN Problem in Polynomial Time for Bounded Tree-Width Graphs

It was mentioned in the previous section that computing MDDN(G, k) can be done in polynomial time when the input graph is a tree. A natural question is what the complexity of our problem is on bounded tree-width graphs, which are more generalized than trees. In this section, we address this question and prove that for any constant k MDDN(G, k) can be computed in linear time for bounded tree-width graphs.

In a beautiful paper, Circle [19] proposed a general method to prove a problem is linear time solvable for bounded tree-width graphs. We show that Courcelle's method can be applied to our problem, when k is considered as a constant value, resulting in the following theorem:

Theorem 3.19. *Suppose k is a fixed integer. Then, for every graph G of constant tree-width, the problem of finding MDDN(G, k) and a corresponding partitioning can be solved in linear time.*

Proof. As defined in the preliminaries section, MDDN(G, k) is the minimum ℓ allowing a k-partitioning for G^ℓ into k disjoint dominating sets. We define DP(G, k, ℓ) to be true if G^ℓ has k disjoint dominating sets, and false otherwise. If computing DP(G, k, ℓ) can be done in polynomial time, we can compute MDDN(G, k) in linear time by examining DP$(G, k, \ell) \geq k$ for $\ell = 1, 2, \ldots, k-1$.

Based on the fact that every graph problem stated in *MSOL*(π_1, p), which is a certain variation of *monadic second order logic* defined for graphs, can be solved in linear time for graphs of bounded tree-width [19], it is sufficient to show that DP(G, k, ℓ) can be expressed in MSOL(π_1, p); MSOL(π_1, p), initially studied by Courcelle, includes the following variables, operators, and quantifiers [21]:

1. atomic variables, shown by small letters (such as x, y)
2. set variables, represented by capital letters (such as X, Y)
3. operators $\neg, \vee, \wedge, =, \neq$, and $\in$ (such as $x \in Y$, $x = v \vee x = w$)
4. existential and universal quantifiers over vertices, edges, and sets of vertices (such as $\exists\{u, v\} \in E, \forall X \subset V$)

Now, it is the time to state DP(G, k, ℓ) in MSOL(π_1, p). The representation is shown in Figure 3.3. To increase readability, we have used several auxiliary functions in this representation. The auxiliary function distance(ℓ, u, v) calculates whether there is a path of length ℓ between vertices u and v. It checks for $\ell - 1$ vertices $w_1, w_2, \ldots, w_{\ell-1}$ connecting u and v via a path. If there exists such a path, it is not necessarily the shortest path between u and v. Therefore, if distance(ℓ, u, v) is true, we can only conclude $d(u, v) \leq \ell$; otherwise, if distance(ℓ, u, v) is not true, $d(u, v) \neq \ell$, since no path of length ℓ connecting u and v exists. In particular, for the case $d(u, v) \leq \ell$, depending on the the graph, distance(ℓ, u, v) may return either

true or false. However, using this *distance* function, we construct close(ℓ, u, v) that determines $d(u, v) \leq \ell$. The key point in designing this function is that $d(u, v) \leq \ell$ if and only if there exists a path of length at most ℓ between u and v. The function intersect(U, W) returns true if and only if the sets U and W have a common member v. The function dominate(G, U, ℓ) determines whether the set of vertices $U \subseteq V$ is a ℓ-dominating set for G. This functions checks for the existence of some vertex $u \in U$ for each vertex $v \in V$, such that $d(u, v) \leq \ell$. Finally, DP(G, k, ℓ) holds if and only if graph vertices can be partitioned into k disjoint ℓ-dominating sets $V_1, V_2, \ldots, V_k$. □

In this chapter we proved that MDDN is hard even for planar bipartite graphs of maximum degree four. On the other hand, we proved that our problem is easy, i.e. polynomial-time solvable, on strongly chordal and bounded tree-width graphs. However, polynomial-time solvability is not the only difficulty measure of problems. Thus, we will proceed by studying the exact algorithms and approximation algorithms for MDDN in next chapters.

$$
\begin{aligned}
\text{distance}(\ell, u, v) &\equiv \exists w_1 \in V : \exists w_2 \in V : \ \ldots \exists w_{\ell-1} \in V : \\
&\quad \{u, w_1\} \in E \wedge \{w_1, w_2\} \in E \wedge \{w_2, w_3\} \in E \wedge \ldots \wedge \{w_{\ell-1}, v\} \in E \\
\text{close}(\ell, u, v) &\equiv (u = v) \vee \text{distance}(1, u, v) \vee \text{distance}(2, u, v) \vee \ldots \text{distance}(\ell, u, v) \\
\text{intersect}(U, W) &\equiv \exists v \in U : v \in W \\
\text{dominate}(G, U, \ell) &\equiv \forall v \in V : \exists u \in U : \ \text{close}(\ell, u, v) \\
\text{DP}(G, k, \ell) &\equiv \exists V_1 \subset V : \exists V_2 \subset V : \ \ldots \exists V_k \subset V : \\
&\quad \neg\text{intersect}(V_1, V_2) \wedge \ldots \wedge \neg\text{intersect}(V_1, V_k) \wedge \\
&\quad \neg\text{intersect}(V_2, V_3) \wedge \ldots \wedge \neg\text{intersect}(V_2, V_k) \wedge \\
&\quad \ldots\ldots \\
&\quad \neg\text{intersect}(V_{k-1}, V_k) \wedge \\
&\quad \text{dominate}(G, V_1, \ell) \wedge \text{dominate}(G, V_2, \ell) \ldots \wedge \text{dominate}(G, V_k, \ell)
\end{aligned}
$$

Figure 3.3: A monadic second order logic representation for the DP problem

Chapter 4

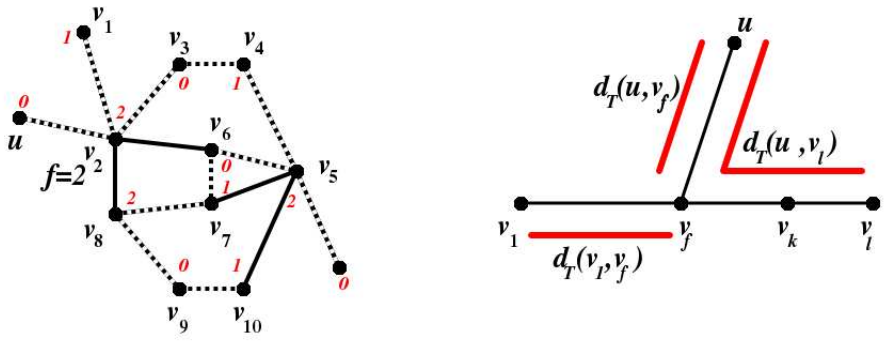
An Upper Bound on the Value of MDDN(G, k)

In this chapter, we give a slightly simpler proof for the upper bound MDDN$(G, k) \leq k - 1$ mentioned in Theorem 2.31. This bound was a direct result of an algorithm proposed by Zelinka [59] that finds k disjoint $(k-1)$-dominating sets in every input graph G. Zelinka's algorithm gives a k-partitioning for the input graph. Then, he proves that each partition is a $(k-1)$-dominating set. We propose a different k-partitioning simplifying the proof to some extent. The key idea in our proof is that the longest path in a tree can be found in polynomial time and we can use the longest path in the spanning tree of G rather than in G to give an acceptable partitioning.

Theorem 4.1. *For any graph $G = (V, E)$ and positive integer k, MDDN$(G, k) \leq k - 1$.*

Proof. Based on Observation 2.30, the theorem holds when $\text{diam}(G) \leq k - 1$. To prove the statement for the case of $\text{diam}(G) > k - 1$, it is sufficient to provide a k-partitioning for V in which each partition is a dominating set for G^{k-1}.

We propose the following partitioning for V: we construct a spanning tree T of G and consider the longest path P in T. Assuming that the vertices in P are



(a) (b)

Figure 4.1: The suggested partitioning and vertex labels in Theorem 4.1.

called $v_1, v_2, \ldots, v_\ell$, we assign each vertex $u \in V$ to partition $(d_T(u, v_1) \bmod k)$. Figure 4.1(a) illustrates this partitioning for $k = 3$ on a sample graph. In this figure, T is specified by dashed edges, P is the path connecting $v_1, v_2, \ldots, v_{10}$, and every vertex is labeled by its partition number.

Now, we claim that all the specified partitions are dominating sets for G^{k-1}. Equivalently, every vertex u is covered in G^{k-1} by each of the k specified partitions.

We prove our claim for both cases $d_T(u, v_1) \geq k - 1$ and $d_T(u, v_1) < k - 1$. In the former case, the first k vertices in the path from u to v_1, including u, are from different partitions, and cover u in G^{k-1}. For the latter case, we show that the vertices $v_1, v_2, \ldots, v_k$ are adjacent to u in G^{k-1}: Since T is connected, u must have a path to a vertex in P, say v_f. However, T is a tree and does not have any cycles. Therefore, u cannot connect to vertices in P via two paths. As a result, the only path between u and any vertex v_i, $1 \leq i \leq \ell$, consists of the unique path from u to v_f and a path from v_f to v_i. Similarly, since T does not have any cycles, v_f has a unique path to every v_i which exactly includes the set of vertices $\{v_i, v_{i+1}, \ldots, v_{f-1}, v_f\}$ for $i \leq f$, or the set of vertices $\{v_f, v_{f+1}, \ldots, v_{i-1}, v_i\}$ for $f \leq i$. The situation is shown in Figure 4.1(b). Hence, $\{v_1, \ldots, v_f\}$ are in the shortest path from u to v_1. We are considering the case $d_T(u, v_1) < k-1$, and thus,

$f \leq k-1$ and u is adjacent to $\{v_1, \ldots, v_f\}$ in G^{k-1}. In addition, as P is the longest path in T, $d_T(u, v_\ell) \leq d_T(v_1, v_\ell)$. Subtracting the common path from v_f to v_ℓ from both corresponding paths, we get $d_T(u, v_f) \leq d_T(v_1, v_f)$. Now, for every $f \leq i \leq k$, adding a common path from v_f to v_i to both these paths leads to the inequality $d_T(u, v_i) \leq d_T(v_1, v_i) = i - 1 \leq k - 1$, proving the adjacency of u and v_i in G^{k-1}. Thus, all vertices in $\{v_1, v_2, \ldots, v_k\}$ are adjacent to u in G^{k-1}, and u is covered by all partitions. □

Chapter 5

Exact Algorithms for General Graphs

In this chapter, we develop a new exact algorithm for solving MDDN$(G, 3)$ on general graphs. As mentioned in the preliminaries section (Corollary 2.43, Corollary 2.44), the problem of computing MDDN$(G, 3)$ is equivalent to the three domatic number problem. Our algorithm was originally developed to solve the three domatic number problem, and runs in time $\tilde{\mathcal{O}}(2.7393^n)$, for an n vertex graph, which is faster than the previously best known $\tilde{\mathcal{O}}(2.8805^n)$ exact algorithm for the three domatic number by Fomin et al. [26]. Recently, using a completely different approach, Riege et al. [52] have proposed another algorithm for the three domatic number problem with time complexity $\tilde{\mathcal{O}}(2.695^n)$.

Since our algorithm uses part of Fomin et al.'s algorithm, we first give an overview of their algorithm in Section 5.1. Next, in Section 5.2 we present a detailed explanation of our algorithm and its analysis.

5.1 Fomin et al.'s Algorithm

As we use part of Fomin et al.'s work [26] in our algorithm, we go through the basic idea of their algorithm.

It was stated in the preliminaries section that the domatic number of a graph G can be seen as the maximum number of disjoint *minimal* dominating sets in G. Considering this fact, Fomin et al. solve a harder problem rather than computing $\mathrm{D}(G)$: for every subset X of graph vertices $V(G)$, what is the maximum number of disjoint minimal dominating sets of G in X, i.e. $\mathrm{D}(G|X)$? Note that whenever $\mathrm{D}(G|X)$ is more than 1, there must exist a subset Y of X such that Y is a minimal dominating set for G and also $\mathrm{D}(G|(X-Y))$ is exactly $\mathrm{D}(G|X)-1$. Hence, we would have:

$$\mathrm{D}(G|X) = \max\{\mathrm{D}(G|(X-Y)) + 1 \mid Y \subseteq X \text{ and } Y \text{ is a minimal dominating set}\}.$$

The key idea of the algorithm is to enumerate all possible Y's for the given X in a relatively fast way, and find the domatic number recursively. To avoid repeating the same computation, dynamic programming is used. That is, all subsets X of $V(G)$ are listed in the order of non-decreasing $|X|$, and $\mathrm{D}(G|X)$'s are calculated and saved for them. For each $X \subseteq V(G)$, all $\mathrm{D}(G|(X-Y))$'s are saved before computing $\mathrm{D}(G|X)$. Consequently, each $\mathrm{D}(G|X)$ is determined in the same time complexity as enumerating all minimal dominating sets Y of G in X.

In the following, we give an overview of Fomin et al.'s enumeration technique. This is mainly because this part is used in our improvement for the three domatic number problem. Computations of the overall running time of their algorithm are given at the end.

To list all the minimal dominating sets of G, Fomin et al. obtain an upper bound on the number of minimal set covers of the corresponding set cover problem $(V(G), \cup_{v \in V(G)}(v \cup N(v)))$ and convert their counting method to an enumeration of all possible cases. In fact, the main contribution of their paper is bounding the number of minimal dominating sets, which also improves the time complexity of determining the domatic number of a given graph.

Applying the *measure and conquer* method [27] from exact algorithms, Fomin et al. obtain a bound on the number of minimal set covers. Fomin et al.'s problem measure computes the *size of an instance* $(U, \mathcal{S})$ *of the set cover problem* as

$f((U,\mathcal{S})) = |U| + \Sigma_{i=1}^{4}\epsilon_i n_i$, where the value of ϵ_1 to ϵ_4 will be set later; n_1, n_2, and n_3 are the numbers of sets in $\mathcal{S}$ having precisely one, two, and three elements, and n_4 is the number of sets in $\mathcal{S}$ having more than three elements. Fomin et al. assume that for each instance of the set cover problem of size s, i.e. for each $(U,\mathcal{S})$ that $f((U,\mathcal{S})) = s$, the number of minimal set covers in the problem is bounded by α^s, where α is another parameter which is set at the end of their analysis. Then, they specify the number of minimal set covers of $(U,\mathcal{S})$ in terms of the number of minimal set covers of several smaller instances. Using induction, they obtain an upper bound on the number of minimal set covers based on the bound on smaller instances.

For future reference, we list all the recursive formulas in Fomin et al.'s paper in Figure 5.1. In these inequalities, the parameters d_2, d_3, and d_4 denote $\min\{\epsilon_1, \epsilon_2 - \epsilon_1\}, \min\{d_2, \epsilon_3 - \epsilon_2\}$, and $\min\{d_3, \epsilon_4 - \epsilon_3\}$, respectively. More precisely, Fomin et al. prove the following lemma.

Lemma 5.1. *[26] Suppose* $0 \leq \epsilon_1 \leq \epsilon_2 \leq \epsilon_3 \leq \epsilon_4$, *and* d_2, d_3, d_4 *denote* $\min\{\epsilon_1, \epsilon_2 - \epsilon_1\}$, $\min\{d_2, \epsilon_3 - \epsilon_2\}$, *and* $\min\{d_3, \epsilon_4 - \epsilon_3\}$, *respectively. Then, if* U *is a set of elements,* $\mathcal{S}$ *is a family of subsets of* U, *and* $s = f((U,\mathcal{S})) = |U| + \Sigma_{i=1}^{4}\epsilon_i n_i$, *then* $(U,\mathcal{S})$ *has at most* α^s *minimal set covers, where* α *and the* ϵ_i*'s satisfy the inequalities in Figure 5.1.*

Corollary 5.2. *[26] Suppose* $G = (V,E)$ *is an* n*-vertex graph and* $X \subseteq V$. *Then, the number of minimal dominating sets of* G *that are completely in* X *is at most* $\alpha^{n+\epsilon_4|X|}$, *where* α *and* $0 \leq \epsilon_1 \leq \epsilon_2 \leq \epsilon_3 \leq \epsilon_4$ *satisfy the inequalities of Figure 5.1 and* d_2, d_3, d_4 *denote* $\min\{\epsilon_1, \epsilon_2 - \epsilon_1\}$, $\min\{d_2, \epsilon_3 - \epsilon_2\}$, *and* $\min\{d_3, \epsilon_4 - \epsilon_3\}$, *respectively.*

According to their paper this counting is easily converted to an enumeration algorithm:

Corollary 5.3. *[26] Suppose* $G = (V,E)$ *is an* n*-vertex graph and* $X \subseteq V$. *There is an algorithm enumerating all minimal dominating sets of* G *that are completely in* X *in time* $\tilde{\mathcal{O}}(\alpha^{n+\epsilon_4|X|})$, *where* α *and* $0 \leq \epsilon_1 \leq \epsilon_2 \leq \epsilon_3 \leq \epsilon_4$ *satisfy the inequalities of Figure 5.1.*

$$
\begin{aligned}
\alpha^s &\geq r\alpha^{s-r\epsilon_1-1}, \forall r \geq 2 \\
\alpha^s &\geq \alpha^{s-\epsilon_4} + \alpha^{s-5-\epsilon_4} \\
\alpha^s &\geq \alpha^{s-\epsilon_4} + \alpha^{s-4-\epsilon_4-4d_4} \\
\alpha^s &\geq \alpha^{s-1-\epsilon_1-\epsilon_2} + \alpha^{s-2-\epsilon_1-\epsilon_2-d_3} \\
\alpha^s &\geq 2\alpha^{s-2-2\epsilon_2} \\
\alpha^s &\geq 2\alpha^{s-2-2\epsilon_2-d_3} + \alpha^{s-3-2\epsilon_2-2d_3} \\
\alpha^s &\geq \alpha^{s-\epsilon_3} + \alpha^{s-3-\epsilon_3-6d_3} \\
\alpha^s &\geq \alpha^{s-\epsilon_2} + \alpha^{s-2-2\epsilon_2-2d_2} \\
\alpha^s &\geq \alpha^{s-\epsilon_2} + \alpha^{s-2-\epsilon_1-\epsilon_2-3d_2} \\
\alpha^s &\geq 3\alpha^{s-2-3\epsilon_2-2d_2} + 3\alpha^{s-3-6\epsilon_2} + \alpha^{s-4-6\epsilon_2} \\
\alpha^s &\geq \alpha^{s-\epsilon_2} + 2\alpha^{s-2-4\epsilon_2-3d_2}
\end{aligned}
$$

Figure 5.1: The final recursive formulas for the set cover problem in Fomin et al.'s paper

It only remains to set the values of ϵ_1, ϵ_2, ϵ_3, ϵ_4, and α such that all the inequalities hold. In addition, the assigned values should lead to a good upper bound on the time complexity of the whole domatic number problem. Now, we show that the suggested value assignment of Fomin et al. proves the time complexity $\tilde{\mathcal{O}}(2.8805^n)$ for their algorithm.

As mentioned in the overview of Fomin et al.'s algorithm, this algorithm finds the domatic number of a graph G by computing $D(G|X)$'s for all subsets X of $V(G)$, which can be done by enumerating all minimal dominating sets Y of G in X. Due to Corollary 5.3, this step is done in time $\tilde{\mathcal{O}}(\alpha^{n+\epsilon_4 i})$ for any X of size i. Since the number of X's, $X \subseteq V(G)$, of size i is $\binom{n}{i}$, the final time complexity is

$$\tilde{\mathcal{O}}(\sum_{i=0}^{n} \binom{n}{i} \alpha^{n+\epsilon_4 i}) \in \tilde{\mathcal{O}}(\alpha^n (1+\alpha^{\epsilon_4})^n),$$

that is converted to 2.8805^n with the following assignments:

$$\epsilon_1 = 3.3512,\ \epsilon_2 = 4.0202,\ \epsilon_3 = 4.4664,\ \epsilon_4 = 4.7164,\ \text{and}\ \alpha < 1.105579.$$

Note that these values for ϵ_i's and α satisfy the inequality set in Figure 5.1.

5.2 An Improvement for the $k = 3$ Case

In this section, we explain our algorithm which solves the three domatic number problem, and equivalently computes MDDN$(G, 3)$, in time $\tilde{\mathcal{O}}(2.7393^n)$.

The key point in our algorithm is that if we can enumerate all pairs of disjoint minimal dominating sets in a graph G, it will be straightforward to check the existence of three disjoint dominating sets in G; for every such pair (D_1,D_2), we examine whether the set of vertices $D_3 = V(G) - D_1 - D_2$ forms a dominating set for G.

If G has three disjoint dominating sets, the smallest dominating set has at most $n/3$ vertices. Thus, we can also restrict our pair enumeration to listing all pairs (D_1,D_2) that satisfy $|D_1| \leq n/3$ and $|D_1| \leq |D_2|$.

In our pair enumeration, we exploit Fomin et al.'s enumeration algorithm. However, instead of enumerating all the minimal dominating sets without any special order, we enumerate minimal dominating sets of size 1, then 2, and continue to size $n/3$. This step is done based on a small change in Fomin et al.'s algorithm that receives the size of the dominating set as an extra input. In the next step, for each dominating set D_1 we enumerate all minimal dominating sets only in $V(G) - D_1$, using Fomin et al.'s algorithm unchanged; in the analysis of this enumeration we are aware of $|D_1|$, which helps us establish a better bound on the computations. Note that since the running time function is different in this approach, we find other values for the ϵ's and α.

To give better insight, our algorithm is shown precisely in Figure 5.4, page 70. The function All-Minimal-Dominating-Sets(G, X, ℓ) is the modified enumeration function we will describe next. This function returns an enumeration of all

minimal dominating sets of graph G that only contain vertices in X and have at most ℓ vertices. As this enumeration function differs from Fomin et al.'s function only in restricting the size of returned minimal set covers, setting $\ell = n$ changes it to Fomin et al.'s algorithm.

To develop the modified enumeration algorithm, we prove the following lemma, which bounds the number of minimal dominating sets. Although the proof is very similar to Fomin et al.'s proof, the explanations are not omitted to make the result verifiable. In this proof, the size of a set cover problem is defined as $f'((U, \mathcal{S})) = |U| + \Sigma_{i=1}^{4} c_i n_i$, where n_1, n_2, and n_3 are the numbers of sets in $\mathcal{S}$ having precisely one, two, and three elements, and n_4 is the number of sets in $\mathcal{S}$ having more than three elements.

Lemma 5.4. *Suppose* $0 \leq c_1 \leq c_2 \leq c_3 \leq c_4$, $\min\{c_1, c_2 - c_1\}$ *is denoted by* d_2, $\min\{d_2, c_3 - c_2\}$ *is denoted by* d_3, *and* $\min\{d_3, c_4 - c_3\}$ *is denoted by* d_4. *Also,* U *is a set of elements,* $\mathcal{S}$ *is a family of subsets of* U, *and* s *is the size of the set cover problem* $(U, \mathcal{S})$ *defined as* $s = f'((U, \mathcal{S})) = |U| + \Sigma_{i=1}^{4} c_i n_i$ *for* n_1, n_2, n_3, n_4 *denoting the number of subsets in* $\mathcal{S}$ *of size one, two, three, and more than three, respectively. Then,* $(U, \mathcal{S})$ *has at most* $\lambda^s \beta^\ell$ *minimal set covers of size* ℓ, *where* λ, β, *and* c_i*'s satisfy the following inequalities:*

$$\lambda^s \beta^\ell \geq \max \begin{cases} r\lambda^{s-rc_1-1}\beta^{\ell-1}, \forall r \geq 2 & (1) \\ \lambda^{s-c_4}\beta^\ell + \lambda^{s-5-c_4}\beta^{\ell-1} & (2) \\ \lambda^{s-c_4}\beta^\ell + \lambda^{s-4-c_4-4d_4}\beta^{\ell-1} & (3) \\ \lambda^{s-1-c_1-c_2}\beta^{\ell-1} + \lambda^{s-2-c_1-c_2-d_3}\beta^{\ell-1} & (4A) \\ 2\lambda^{s-2-2c_2}\beta^{\ell-1} & (4B) \\ 2\lambda^{s-2-2c_2-d_3}\beta^{\ell-1} + \lambda^{s-3-2c_2-2d_3}\beta^{\ell-2} & (4C) \\ \lambda^{s-c_3}\beta^\ell + \lambda^{s-3-c_3-6d_3}\beta^{\ell-1} & (5) \\ \lambda^{s-c_2}\beta^\ell + \lambda^{s-2-2c_2-2d_2}\beta^{\ell-1} & (6A) \\ \lambda^{s-c_2}\beta^\ell + \lambda^{s-2-c_1-c_2-3d_2}\beta^{\ell-1} & (6B) \\ 3\lambda^{s-2-3c_2-2d_2}\beta^{\ell-1} + 3\lambda^{s-3-6c_2}\beta^{\ell-2} + \lambda^{s-4-6c_2}\beta^{\ell-3} & (7) \\ \lambda^{s-c_2}\beta^\ell + 2\lambda^{s-2-4c_2-3d_2}\beta^{\ell-1} & (8) \end{cases}$$

Proof. We first fix some notation that will be used in the proof. Let $\mathrm{COV}(U, \mathcal{S}, \ell)$

denote the number of minimal set covers of size at most ℓ for $(U, \mathcal{S})$, and let $\mathrm{COV}(s, \ell)$ denote the maximum possible value of $\mathrm{COV}(U, \mathcal{S}, \ell)$ over all set cover problems $(U, \mathcal{S})$ of size s. Also, we use $\mathcal{S}|U$ to denote $\{S \cap U : S \in \mathcal{S}\}$. We say an element $u \in U$ has *frequency* t if exactly t subsets in $\mathcal{S}$ contain u.

As in Fomin et al.'s proof, we use strong induction on s to prove the lemma: we prove that for any $\lambda, \beta, c_1, c_2, c_3, c_4$ satisfying the above-mentioned inequalities, and for all non-negative integers s and ℓ, $\mathrm{COV}(s, l) \leq \lambda^s \beta^\ell$.

Base Case: If $s = 0$, $|U|$ must be zero, and hence, $|\mathcal{S}| = 0$ as well. The only minimal set cover for this problem is the empty set. Therefore, $\mathrm{COV}(0, \ell) = 1$ for all $\ell \geq 0$.

Induction hypothesis: $\mathrm{COV}(k, \ell) \leq \lambda^k \beta^\ell$ for all $k < s$.

Induction step: Assuming the hypothesis, we prove that $\mathrm{COV}(U, \mathcal{S}, \ell) \leq \lambda^s \beta^\ell$ for any set cover problem $(U, \mathcal{S})$ of size s, where the size of $(U, \mathcal{S})$ is $f'((U, \mathcal{S})) = |U| + \Sigma_{i=1}^{4} c_i n_i$ for n_1, n_2, n_3, n_4 denoting the number of subsets in $\mathcal{S}$ of size one, two, three, and more than three, respectively. Suppose $\mathcal{S}$ is a family of subsets of U such that $(U, \mathcal{S})$ has size s. Also, suppose $\ell \geq 0$ is an integer.

Fomin et al. have considered nine possible cases, and proved their statement for each case. We do the same, except that we also consider the size of the set cover in the computations and we count the number of minimal set covers of size at most ℓ. We number the cases as Fomin et al. have done this. Note that the order of these cases is important. We prove that the inequality resulted from a case, say 'case i', is true for all problems $(U, \mathcal{S})$ of size s that have none of the conditions listed in 'case 0' to 'case i-1'. For instance, if the set cover problem $(U, \mathcal{S})$ has both conditions of 'case 1' and 'case 3', we only prove the inequality corresponding to 'case 1' for it. As a result, since the last case considers all remaining set cover problems not satisfying the previous cases, these cases are exhaustive.

It might be useful to note that we have only added a β parameter to Fomin et al.'s proof [26, Lemma 1], and the specified recursive formulas, as well as the exponent of λ in each case, were proved in their theorem.

Case 0: There is a $u \in U$ of frequency one.

The only set S_1 containing u must be in every minimal set cover. Thus, to enumerate all minimal set covers of size at most ℓ in $(U, \mathcal{S})$, we can enumerate all minimal dominating sets of size at most $\ell - 1$ of $(U - \{u\}, \mathcal{S} - \{S_1\})$ and add S_1 to all covers in this enumeration. Since adding S_1 to each cover in this enumeration does not change the number of set covers in the enumeration, we have

$$\mathrm{COV}(U, \mathcal{S}, \ell) \leq \mathrm{COV}(U - \{u\}, \mathcal{S} - \{S_1\}, \ell - 1).$$

In this new set cover problem, the number of elements in U and the number of subsets in $\mathcal{S}$ are decreased by one. Based on $|S_1|$, the size of this new set cover problem may be $s - 1 - c_1$, $s - 1 - c_2$, $s - 1 - c_3$, or $s - 1 - c_4$. Therefore, due to the induction hypothesis, $\mathrm{COV}(U - \{u\}, \mathcal{S} - \{S_1\}, \ell - 1)$ is less than or equal to $\lambda^{s-c_i-1}\beta^{\ell-1}$ for a $c_1 \leq c_i \leq c_4$. Setting $c_i = c_1$ gives an upper bound for this expression. Therefore,

$$\mathrm{COV}(U, \mathcal{S}, \ell) \leq \mathrm{COV}(U - \{u\}, \mathcal{S} - \{S_1\}, \ell - 1) \leq \lambda^{s-c_1-1}\beta^{\ell-1}.$$

To prove $\mathrm{COV}(U, \mathcal{S}, \ell) \leq \lambda^s\beta^\ell$, it is sufficient to have $\lambda^{s-c_1-1}\beta^{\ell-1} \leq \lambda^s\beta^\ell$, which is always true.

Case 1: There is a $u \in U$ that belongs only to size one sets.

Assume u is in $S_1, \ldots, S_r$, where $r \geq 2$. Then, every minimal set cover should contain exactly one of $S_1, \ldots, S_r$. Thus, to enumerate all minimal set covers of size at most ℓ in $(U, \mathcal{S})$ we can compute all minimal set covers of size at most $\ell - 1$ of $(U - \{u\}, \mathcal{S} - \{S_1, \ldots, S_r\})$, and compute the union of new enumerations produced by adding S_1, S_2, $\ldots$, or S_n to these set covers. Therefore,

$$\mathrm{COV}(U, \mathcal{S}, \ell) = r \cdot \mathrm{COV}(U - \{u\}, \mathcal{S} - \{S_1, \ldots, S_r\}, \ell - 1) \leq r\lambda^{s-rc_1-1}\beta^{\ell-1}.$$

The second inequality is obtained similar to the previous case, except that in this case r subsets are removed from $\mathcal{S}$ instead of one subset.

As the previous case, it is enough to have $r\lambda^{s-rc_1-1}\beta^{\ell-1} \leq \lambda^s\beta^\ell$ for every $r \geq 2$. This condition is the first inequality listed in the lemma, and therefore, it holds.

Case 2: There is an $S_1 = \{u_1, \ldots, u_r\} \in \mathcal{S}$ where $r \geq 5$.

Since the number of minimal set covers that do not have S_1 is at most $\mathrm{COV}(U, \mathcal{S} - \{S_1\}, \ell)$, and the number of minimal set covers that have S_1 is at most $\mathrm{COV}(U_1, (\mathcal{S} - \{S_1\})|U_1, \ell - 1)$, where $U_1 = U - S_1$, we have

$$\mathrm{COV}(U, \mathcal{S}, \ell) \leq \mathrm{COV}(U, \mathcal{S} - \{S_1\}, \ell) + \mathrm{COV}(U_1, (\mathcal{S} - \{S_1\})|U_1, \ell - 1).$$

The size of the first resulting set cover problem, $(U, \mathcal{S} - \{S_1\})$, is $s - c_4$, since $|S_1| \geq 5$. In the second problem, $(U_1, (\mathcal{S} - \{S_1\})|U_1)$, $|U_1| \leq |U| - 5$ and a set of size at least five is eliminated from $\mathcal{S}$. Therefore, the size of this problem is at most $s - 5 - c_4$. Using the induction hypothesis, we have:

$$\mathrm{COV}(U, \mathcal{S} - \{S_1\}, \ell) + \mathrm{COV}(U_1, (\mathcal{S} - \{S_1\})|U_1, \ell - 1) \leq \lambda^{s-c_4}\beta^{\ell} + \lambda^{s-5-c_4}\beta^{\ell-1}.$$

Based on Inequality 2, $\mathrm{COV}(U, \mathcal{S}, \ell) \leq \lambda^s \beta^{\ell}$ is obtained for this case.

Case 3: There is an $S_1 = \{u_1, u_2, u_3, u_4\} \in \mathcal{S}$.

This case is similar to Case 2, except that all subsets of size more than four are eliminated in Case 2. Also, we consider the fact that all elements of U with frequency one are also removed in Case 0. Therefore, every u_i $(1 \leq i \leq 4)$ is a member of a subset other than S_1 of size at most four. Removing u_i $(1 \leq i \leq 4)$ reduces the size of this set by one, thus reducing the size of the problem by at least $1 + \min\{c_1, c_2 - c_1, c_3 - c_2, c_4 - c_3\} = 1 + d_4$. Removing S_1 decreases the size by another c_4. Thus,

$$\begin{aligned} \mathrm{COV}(U, \mathcal{S}, \ell) &\leq \mathrm{COV}(U, \mathcal{S} - \{S_1\}, \ell) + \mathrm{COV}(U_1, (\mathcal{S} - \{S_1\})|U_1, \ell - 1) \\ &\leq \lambda^{s-c_4}\beta^{\ell} + \lambda^{s-4-c_4-4d_4}\beta^{\ell-1}, \end{aligned}$$

where $U_1 = U - S_1$. Now, Inequality 3 proves the required condition.

Case 4: There is a $u \in U$ of frequency two.

Suppose u is in S_1 and S_2, where $|S_1| \leq |S_2|$. Since the condition of Case 1 does not hold, $|S_2| \geq 2$. We consider both possible cases $|S_1| = 1$ and $|S_1| \geq 2$, where the second case is also studied for $S_1 \subseteq S_2$ and $S_1 \not\subseteq S_2$.

Subcase 4A: $|S_1| = 1$.

Thus, every minimal set cover has either S_1 or S_2, but not both. After selecting which one is put in the minimal set cover, S_1 and S_2 and the element covered by S_1 will be removed, reducing the size of the problem by at least $c_1 + c_2 + 1$. Furthermore, if we remove S_2, at least one element other than u will be eliminated. Using the same argument as in Case 3, removing an element of frequency at least two decreases the size of the problem by at least $1 + d_3$. Therefore,

$$\begin{aligned}\mathrm{COV}(U, \mathcal{S}, \ell) &\leq \mathrm{COV}(U_1, \mathcal{S} - \{S_1, S_2\}, \ell - 1) \\ &\quad + \mathrm{COV}(U_2, ((\mathcal{S} - S_1), S_2)|U_2, \ell - 1) \\ &\leq \lambda^{s-1-c_1-c_2}\beta^{\ell-1} + \lambda^{s-1-c_1-c_2-1-d_3}\beta^{\ell-1},\end{aligned}$$

where $U_1 = U - S_1$ and $U_2 = U - S_2$. Inequality 4A proves that $\mathrm{COV}(U, \mathcal{S}, \ell) \leq \lambda^s \beta^\ell$.

Subcase 4B: $|S_1| \geq 2$ and $S_1 \subseteq S_2$.

Every minimal set cover has either S_1 or S_2, but not both. Thus, after selecting either S_1 or S_2 for the minimal set cover, we remove both S_1 and S_2 from $\mathcal{S}$. Since both of them have at least two elements, removing them reduces the size of the problem by at least $(1 + c_2) + (1 + c_2)$, and we have:

$$\begin{aligned}\mathrm{COV}(U, \mathcal{S}, \ell) &\leq \mathrm{COV}(U_1, \mathcal{S} - \{S_1, S_2\}|U_1, \ell - 1) \\ &\quad + \mathrm{COV}(U_2, \mathcal{S} - \{S_1, S_2\}|U_2, \ell - 1) \\ &\leq 2\lambda^{s-2-2c_2}\beta^{\ell-1},\end{aligned}$$

where $U_1 = U - S_1$ and $U_2 = U - S_2$. Now, Inequality 4B proves the required statement.

Subcase 4C: $|S_1| \geq 2$ and $S_1 \not\subseteq S_2$.

If exactly one of S_1 and S_2 is selected for the minimal set cover, then both sets S_1, S_2 and two elements are removed and the cardinality of at least one other set is reduced. If we select both S_1 and S_2, at least three elements and two sets are removed and either the sizes of two other sets are reduced by one or the size of one

other set is reduced by two. Thus,

$$\begin{aligned}\mathrm{COV}(U,\mathcal{S},\ell) &\leq \mathrm{COV}(U_1,\mathcal{S}-\{S_1,S_2\}|U_1,\ell-1)\\ &\quad+\mathrm{COV}(U_2,(\mathcal{S}-\{S_1,S_2\})|U_2,\ell-1)\\ &\quad+\mathrm{COV}(U_3,(\mathcal{S}-\{S_1,S_2\})|U_3,\ell-2)\\ &\leq 2\lambda^{s-2-2c_2-d_3}\beta^{\ell-1}+\lambda^{s-3-2c_2-2d_3}\beta^{\ell-2},\end{aligned}$$

where $U_1 = U - S_1$, $U_2 = U - S_2$, and $U_3 = U - (S_1 \cup S_2)$. Now, it only remains to use inequality 4C.

Case 5: There is an $S_1 = \{u_1, u_2, u_3\} \in \mathcal{S}$.

This case is similar to Case 3. However, now all subsets have at most three elements and each element has frequency at least three. Therefore, removing each element reduces the size of at least two other sets that contain at most three elements, which reduces the size of the problem by at least $1 + 2 \cdot \min\{c_1, c_2 - c_1, c_3 - c_2\} = 1 + 2d_3$. Removing S_1, u_1, u_2, and u_3 reduces the problem size by at least $c_3 + 3(1 + 2d_3)$. Thus,

$$\begin{aligned}\mathrm{COV}(U,\mathcal{S},\ell) &\leq \mathrm{COV}(U,\mathcal{S}-\{S_1\},\ell)+\mathrm{COV}(U_1,(\mathcal{S}-\{S_1\})|U_1,\ell-1)\\ &\leq \lambda^{s-c_3}\beta^{\ell}+\lambda^{s-c_3-3-6d_3}\beta^{\ell-1},\end{aligned}$$

where $U_1 = U - S$. Inequality 5 proves that $\mathrm{COV}(U, \mathcal{S}, \ell) \leq \lambda^s \beta^\ell$.

Case 6: There are $S_1, S_2 \in \mathcal{S}$ such that $S_1 \subseteq S_2$.

We now know all elements have frequency at least three and all sets in $\mathcal{S}$ are of size at most two. Hence, $|S_1| \leq |S_2| \leq 2$. Also, no minimal set cover can have both S_1 and S_2. We consider two cases now:

Subcase 6A: $S_1 = \{u_1, u_2\} \in \mathcal{S}$.

Therefore, $|S_2| = 2$, and S_1 and S_2 have the same elements, each contained in at least one other subset. S_2 may be selected for the minimal set cover. If we do not choose S_2, only S_2 is removed, changing the size to $s - c_2$. If we choose S_2, S_1, S_2, u_1, and u_2 are removed, and u_1 and u_2 are removed from another set of size

one or two, containing them. Hence, the size of the problem is reduced by at least $c_2 + c_2 + 2 + 2 \cdot \min\{c_1, c_2 - c_1\} = 2 + 2c_2 + 2d_2$. Thus,

$$\begin{aligned} \mathrm{COV}(U, \mathcal{S}, \ell) &\leq \mathrm{COV}(U, \mathcal{S} - \{S_2\}, \ell) + \mathrm{COV}(U_2, (\mathcal{S} - \{S_1, S_2\})|U_2, \ell - 1) \\ &\leq \lambda^{s-c_2}\beta^{\ell} + \lambda^{s-2-2c_2-2d_2}\beta^{\ell-1}, \end{aligned}$$

where U_2 is again $U - S_2$. We can use Inequality 6A to prove the required statement now.

Subcase 6B: $S_1 = \{u_1\} \in \mathcal{S}$.

Since we have passed Case 1, u_1 is not only contained in subsets of size one. Therefore there is another subset S_3 of size two containing u_1. We call the other element of S_3, u_2. If S_3 is not selected in the minimal set cover, we remove S_3, changing the size of the problem to $s - c_2$. Otherwise, S_1, S_3, u_1, and u_2 are removed. u_1 and u_2 are in at least one and two other subsets, respectively. Therefore, the size of the problem reduces by at least $c_1 + c_2 + 2 + 3 \cdot \min\{c_1, c_2 - c_1\} = 2 + c_1 + c_2 + 3d_2$. Thus,

$$\begin{aligned} \mathrm{COV}(U, \mathcal{S}, \ell) &\leq \mathrm{COV}(U, \mathcal{S} - \{S_3\}, \ell) + \mathrm{COV}(U_3, (\mathcal{S} - \{S_1, S_3\})|U_3, \ell - 1) \\ &\leq \lambda^{s-c_2}\beta^{\ell} + \lambda^{s-2-c_1-c_2-3d_2}\beta^{\ell-1}, \end{aligned}$$

where $U_3 = U - S_3$. Inequality 6B proves the statement for this case.

Case 7: There is a $u \in U$ of frequency three.

Since the frequencies of all elements are at least three, every subset of size one in $\mathcal{S}$ is a subset of other subsets in $\mathcal{S}$. As we have passed Case 6, no subset in $\mathcal{S}$ is contained in another subset in $\mathcal{S}$. Therefore, at this step, no subset of size one exists in $\mathcal{S}$, and hence, all subsets in $\mathcal{S}$ have exactly two elements. In addition, since none of these subsets is a subset of another subset, all subsets in $\mathcal{S}$ are different. Therefore, u is contained in subsets of the form $S_1 = \{u, u_1\}$, $S_2 = \{u, u_2\}$, and $S_3 = \{u, u_3\}$.

Every minimal set cover has either one, two, or three of S_1, S_2, and S_3.

- It contains one of S_1, S_2, or S_3, say S_1, removing S_1, S_2, S_3, u, and u_1 reduces the size of the problem by at least $3c_2 + 2 + 2d_2$, since u_1 is in at least two other subsets of size two.

- It includes two of S_1, S_2, and S_3, say S_1 and S_2. Since both S_1 and S_2 are in a minimal set cover, putting any subset containing u_1 or u_2 in the minimal set cover contradicts the minimality of this set cover. There is at most one subset containing both u_1 and u_2. Therefore, we can remove at least three subsets other than S_1 and S_2 containing either u_1 or u_2. Removing these sets along with u, u_1, u_2, S_1, S_2, and S_3 reduces the size of the problem by at least $3c_2 + 3 + 3c_2 = 3 + 6c_2$.

- It contains all S_1, S_2, and S_3. Then, we can remove S_1 S_2, S_3, u, u_1, u_2, u_3, and all other subsets containing u_1, u_2 or u_3. The number of such subsets is at least three. Therefore, removing all these subsets and elements reduces the size of the problem by at least $3c_2 + 4 + 3c_2 = 4 + 6c_2$.

Thus, using the induction hypothesis, we have:

$$\begin{aligned}
\mathrm{COV}(U, \mathcal{S}, \ell) &\leq \sum_{i=1}^{3} \mathrm{COV}(U_i, (\mathcal{S} - \{S_i\}) | U_i, \ell - 1) \\
&\quad + \sum_{i=1}^{3} \mathrm{COV}(U'_i, (\mathcal{S} - (\{S_1, S_2, S_3\} - \{S_i\})) | U'_i, \ell - 2) \\
&\quad + \mathrm{COV}(U'', (\mathcal{S} - (\{S_1, S_2, S_3\}) | U'', \ell - 3) \\
&\leq 3\lambda^{s-2-3c_2-2d_2}\beta^{\ell-1} + 3\lambda^{s-3-6c_2}\beta^{\ell-2} + \lambda^{s-4-6c_2}\beta^{\ell-3},
\end{aligned}$$

where $U_i = U - S_i$, $U'_i = U - \cup_{j \neq i} S_j$, and $U'' = U - \cup_{i=1}^{3} S_i$. Now, Inequality 7 proves our desired inequality.

Case 8: None of the Cases 1-7 holds.

Suppose $S_1 = \{u, v\}$ is a set in $\mathcal{S}$ and $\mathcal{S}_u$ and $\mathcal{S}_v$ are the sets of subsets in $\mathcal{S}$ that have u and v respectively. We know the frequencies of all elements are more than three. Thus, $|\mathcal{S}_u| \geq 3$ and $|\mathcal{S}_v| \geq 3$. Also, every minimal set cover that contains S_1

cannot contain both a set from $\mathcal{S}_u$ and a set from $\mathcal{S}_v$. Thus, we have

$$\begin{aligned}\mathrm{COV}(U,\mathcal{S},\ell) \;\leq\; & \mathrm{COV}(U,\mathcal{S}-\{S_1\},\ell)+\mathrm{COV}(U_1,(\mathcal{S}-(\{S_1\}\cup\mathcal{S}_u))|U_1,\ell-1)\\ & +\mathrm{COV}(U_1,(\mathcal{S}-(\{S_1\}\cup\mathcal{S}_v))|U_1,\ell-1),\end{aligned}$$

where $U_1 = U - S_1$. Removing S_1 changes the size of the problem to $s - c_2$. In the second alternative, S_1 is selected in the minimal set cover, and we can remove v from all subsets in $\mathcal{S}_v$. Removing S_1, u, v, subsets in $\mathcal{S}_u$, and v from subsets in $\mathcal{S}_v$ reduces the size of the problem by at least $c_2 + 2 + 3c_2 + 3(c_2 - c_1) \geq 2 + 4c_2 + 3d_2$. Therefore, the above-mentioned expression is at most

$$\leq \lambda^{s-c_2}\beta^{\ell} + 2\lambda^{s-2-4c_2-3d_2}\beta^{\ell-1}.$$

We can prove $\mathrm{COV}(U,\mathcal{S},\ell) \leq \lambda^s\beta^{\ell}$ for this final case using Inequality 8. □

Using Lemma 5.4, it is easy to find an upper bound on the number of minimal dominating sets of a graph.

Corollary 5.5. *Suppose $G = (V, E)$ is an n-vertex graph and $0 \leq \ell \leq n$ is an integer. Then, the number of minimal dominating sets of size ℓ of G is at most $\lambda^{(1+c_4)n}\beta^{\ell}$, where λ, β, and $0 \leq c_1 \leq c_2 \leq c_3 \leq c_4$ satisfy the inequalities of Lemma 5.4.*

Proof. Let $U = V$ and $\mathcal{S} = \{\{v\} \cup N(v) : v \in V\}$. Moreover, let s denote the size of the set cover problem $(U, \mathcal{S})$ as defined in Lemma 5.4: $s = |U| + \Sigma_{i=1}^{4} c_i n_i$, where n_1, n_2, and n_3 stand for the number of subsets in $\mathcal{S}$ that have 1, 2, and 3 elements, and n_4 denotes the number remaining subsets in $\mathcal{S}$. Then, the number of minimal dominating sets of size ℓ of G is equal to the number of minimal set covers of size ℓ of $(U, \mathcal{S})$ which is at most $\lambda^s\beta^{\ell}$ by Lemma 5.4. Since $s \leq n + \sum_{i=1}^{4} c_4 n_i = n + c_4 n$, the number of minimal dominating sets of size ℓ of G is at most $\lambda^{(1+c_4)n}\beta^{\ell}$. □

Converting the Counting to an Enumeration Algorithm

The counting technique provided can be exploited to design an algorithm to enumerate all minimal set covers of a problem set $(U, \mathcal{S})$ (see Figure 5.2, page 69). In

the given algorithm, the problem instance $(U, \mathcal{S})$ is checked against the conditions of cases in Lemma 5.4. The algorithm acts upon the first case such that the condition is satisfied by instance $(U, \mathcal{S})$, say Case c. Based on the alternatives listed for Case c, the algorithm decides on the smaller instances it will solve to compute the enumeration. The union of minimal set covers for all possible alternatives gives the set of minimal set covers for the current problem instance. In the following, we prove that the running time of this algorithm is in $O(\text{poly}(n) \cdot (|U| + |\mathcal{S}|) \cdot \lambda^{(1+c_4)n} \beta^{\ell})$, which is converted $O(\text{poly}(n) \cdot \lambda^{(1+c_4)n} \beta^{\ell})$ if we execute it to compute the desired minimal dominating sets according to the algorithm shown in Figure 5.3, page 70.

Lemma 5.6. *Suppose U is a set of elements, $\mathcal{S}$ is a set of subsets of U, and $0 \leq \ell \leq n$ is an integer. Then,* All-Minimal-Set-Covers$(U, \mathcal{S}, \ell, \emptyset)$ *enumerates all minimal set covers of size at most ℓ of the set cover problem $(U, \mathcal{S})$ in time $O(|U|(|U| + |\mathcal{S}|)^2 \cdot \lambda^s \beta^{\ell})$, where λ, β, $0 \leq c_1 \leq c_2 \leq c_3 \leq c_4$, $d_2 = \min\{c_1, c_2 - c_1\}$, $d_3 = \min\{d_2, c_3 - c_2\}$, and $d_4 = \min\{d_3, c_4 - c_3\}$ satisfy the inequalities of Lemma 5.4, and s is the size of the problem instance $(U, \mathcal{S})$ defined as $s = f'((U, \mathcal{S})) = |U| + \Sigma_{i=1}^{4} c_i n_i$ for n_1, n_2, n_3, n_4 denoting the number of subsets in $\mathcal{S}$ of size one, two, three, and more than three, respectively.*

Proof. All-Minimal-Set-Covers considers all possible minimal set covers in problem instance $(U, \mathcal{S})$ according to the cases listed in Lemma 5.4. Therefore, the algorithm works correctly.

At each step this algorithm does the following:

1. It finds the first case for which $(U, \mathcal{S})$ satisfies the corresponding condition.
2. It calls All-Minimal-Set-Covers a constant number of times with the specified inputs.
3. It computes the union of all enumerations found.

We implement the algorithm using the following data structure for U and $\mathcal{S}$: elements in U and subsets in $\mathcal{S}$ are kept in two linked lists. Each subset in $\mathcal{S}$ has

a linked list of its elements pointing to elements of U, and each element u in U has a linked list of the subsets in $\mathcal{S}$ containing u. Having these assumptions, the conditions of cases 0 to 8 can be checked in times $O(|U|)$, $O(|U| \cdot |\mathcal{S}|)$, $O(5|\mathcal{S}|)$, $O(4|\mathcal{S}|)$, $O(|U|)$, $O(2|\mathcal{S}|)$, $O(|U| \cdot |U|)$, $O(2|U|)$, and $O(1)$. Thus, checking the conditions takes $O(|U||U| + |U||\mathcal{S}|)$ time. Computing the parameters in function calls consists of removing constant number of elements from U or from subsets in $\mathcal{S}$, thus taking at most $O(|U| + |\mathcal{S}|)$. And, computing the union of constant number of enumeration sets takes $O(1)$ time. Therefore, each step of the algorithm is performed in time $O(|U||U| + |U||\mathcal{S}|)$, and hence, the running time of the algorithm is $O(|U|(|U| + |\mathcal{S}|) \cdot \text{num})$, where num is the number of steps taken. We can view the steps of the algorithm as nodes of a tree. Since for each problem instance at each function call either $|U|$ or $|\mathcal{S}|$ is decreased, the height of this tree is at most $|U|+|\mathcal{S}|$. Since at each node a constant number of computation branches are generated, the number of nodes in this tree is of the order of the number of leaves. Note that at each leaf an acceptable minimal dominating set is produced. Therefore, the number of leaves is equal to the number of minimal dominating sets of size at most ℓ in G with all their vertices in X. Therefore, due to Lemma 5.4, the number of leaves is at most $\lambda^s \beta^\ell$. Consequently, this tree has at most $O((|U| + |\mathcal{S}|) \cdot \lambda^s \beta^\ell)$ nodes, resulting in the running time $O(|U|(|U| + |\mathcal{S}|)^2 \cdot \lambda^s \beta^\ell)$. □

We show that the running time in Lemma 5.6 can be refined to $O(n^3 \cdot \lambda^{(n+c_4|X|)} \beta^\ell)$ when we enumerate all minimal set coves corresponding to minimal dominating sets of a graph:

Lemma 5.7. *Suppose $G = (V, E)$ is an n-vertex graph, $0 \leq \ell \leq n$ is an integer, and $X \subseteq V$ is a set of vertices. Then,* All-Minimal-Dominating-Sets(G, X, ℓ) *enumerates all minimal dominating sets of size at most ℓ of G that are selected from X in time $O(n^3 \cdot \lambda^{(n+c_4|X|)} \beta^\ell)$, where λ, β, $0 \leq c_1 \leq c_2 \leq c_3 \leq c_4$, and $d_2 = \min\{c_1, c_2 - c_1\}$, $d_3 = \min\{d_2, c_3 - c_2\}$, $d_4 = \min\{d_3, c_4 - c_3\}$ satisfy the inequalities of Lemma 5.4.*

Proof. This function sets U to $V(G)$ and $\mathcal{S}$ to $\bigcup_{v \in X} \{v\} \cup N(v)$ in time $O(n^2)$. Then, it enumerates all the minimal set covers of size at most ℓ in the set cover problem

$(U, \mathcal{S})$ by executing All-Minimal-Set-Covers$(U, \mathcal{S}, \ell, \emptyset)$. Due to Lemma 5.6, this step is done in time $O(|U|(|U| + |\mathcal{S}|)^2 \cdot \lambda^s \beta^\ell)$ where $s = f'((U, \mathcal{S})) = |U| + \Sigma_{i=1}^4 c_i n_i$ for n_1, n_2, n_3, n_4 denoting the number of subsets in $\mathcal{S}$ of size one, two, three, and more than three, respectively. As described in Corollary 5.5, since $0 \leq c_1 \leq c_2 \leq c_3 \leq c_4$, $s \leq |U| + \Sigma_{i=1}^4 c_4 n_i = |U| + c_4|S| = n + c_4|X|$, and therefore, Step 7 runs in time $O(|U|(|U| + |\mathcal{S}|)^2 \cdot \lambda^{(n+c_4|X|)} \beta^\ell)$. As explained in the preliminaries section, all minimal dominating sets of graph G in $X \subseteq V(G)$ are generated by replacing all subsets of the form $\{v_i \cup N(v_i)\}$ by v_i in every minimal set cover of the set cover problem $(V(G), \bigcup_{v \in X} \{v\} \cup N(v))$. Thus, steps 8 and 9 produce all minimal dominating sets of size at most ℓ of G in X. These steps are performed in $O(n^2 \cdot num)$ where num is the number of minimal dominating sets in G. According to Corollary 5.5, we know that $num \leq \lambda^{(1+c_4)n} \beta^\ell$. Therefore, the final time complexity of the algorithm is $O(n^2 + |U|(|U| + |\mathcal{S}|)^2 \cdot \lambda^{(n+c_4|X|)} \beta^\ell + n^2 \cdot \lambda^{(1+c_4)|X|} \beta^\ell) \in O(n^2 + n(2n)^2 \cdot \lambda^{(n+c_4|X|)} \beta^\ell) \in O(n^2(n+n) \cdot \lambda^{(n+c_4|X|)} \beta^\ell) \in O(n^3 \cdot \lambda^{(n+c_4|X|)} \beta^\ell)$. □

Now, we can compute the overall running time of our main algorithm, specified in Figure 5.4, page 70, that finds three disjoint dominating sets in the input graph G, if $\mathrm{D}(G) \geq 3$:

Theorem 5.8. *The three domatic number problem can be solved in time* $\tilde{\mathcal{O}}(2.7393^n)$.

Proof. Let λ, β and c_i's be the constants satisfying inequalities in Lemma 5.4 and α and ϵ_i's be the constants of Fomin et al.'s inequalities. According to Lemma 5.7, all minimal dominating sets of size i are enumerated in time $O(n^3 \cdot \lambda^{(1+c_4)n} \beta^i)$. Then, for each dominating set D_1 of size i, all minimal dominating sets only in $V(G) - D_1$ are enumerated. Based on Corollary 5.2, this enumeration can be done in $\tilde{\mathcal{O}}(\alpha^{n+\epsilon_4|V(G)-D_1|}) \in \tilde{\mathcal{O}}(\alpha^{n+\epsilon_4(n-i)})$ time. For each pair of disjoint dominating sets (D_1, D_2) obtained, the algorithm checks whether $D_3 = V(G) - D_1 - D_2$ is a dominating set for G in time $O(n^2)$. Hence, the final time complexity is

$$O\left(\sum_{i=1}^{\lceil n/3\rceil} n^3\cdot\lambda^{(n+c_4n)}\beta^i\cdot\tilde{\mathcal{O}}(\alpha^{(n+\epsilon_4(n-i))})\cdot n^2\right) \in \tilde{\mathcal{O}}\left(\sum_{i=1}^{\lceil n/3\rceil}\lambda^{(n+c_4n)}\beta^i\cdot\alpha^{(n+\epsilon_4(n-i))}\right)$$
$$\in \tilde{\mathcal{O}}\left(\lambda^{((1+c_4)n)}\alpha^{(1+\epsilon_4)n}\sum_{i=1}^{\lceil n/3\rceil}\alpha^{(i\lg_\alpha^\beta - i\epsilon_4)}\right) \in \tilde{\mathcal{O}}\left(\lambda^{((1+c_4)n)}\alpha^{(1+\epsilon_4)n}\sum_{i=1}^{\lceil n/3\rceil}\alpha^{i(\lg_\alpha^\beta-\epsilon_4)}\right).$$

We consider the cases $\lg_\alpha^\beta < \epsilon_4$ and $\lg_\alpha^\beta \geq \epsilon_4$. For the first case, since $\alpha > 1$, $\alpha^{i(\lg_\alpha^\beta-\epsilon_4)}$ is less than 1. Therefore, $\sum_{i=1}^{\lceil n/3\rceil}\alpha^{i(\lg_\alpha^\beta-\epsilon_4)}$ is less than $n/3$, which is negligible in our computations. Consequently, the running time of the first case will be $\tilde{\mathcal{O}}\left(\lambda^{(1+c_4)}\alpha^{(1+\epsilon_4)}\right)^n$. For the second case, since $\lg_\alpha^\beta - \epsilon_4 \geq 0$, replacing i with $\lceil n/3\rceil$ will not increase the power of α. Therefore, as $\alpha > 1$, we can replace i with α without decreasing the computed running time. Consequently, the running time for the second case is at most

$$\tilde{\mathcal{O}}\left(\lambda^{((1+c_4)n)}\alpha^{(1+\epsilon_4)n}\sum_{i=1}^{\lceil n/3\rceil}\alpha^{\lceil n/3\rceil(\lg_\alpha^\beta-\epsilon_4)}\right) \in \tilde{\mathcal{O}}\left(\lambda^{((1+c_4)n)}\alpha^{(1+\epsilon_4)n}\cdot\frac{n}{3}\,\alpha^{n/3(lg_\alpha^\beta-\epsilon_4)}\right)$$
$$\in \tilde{\mathcal{O}}\left(\lambda^{(1+c_4)}\alpha^{(1+2/3\epsilon_4+1/3\lg_\alpha^\beta)}\right)^n.$$

The overall running time can be written as the sum of the above mentioned running times.

$$\tilde{\mathcal{O}}\left(\lambda^{(1+c_4)}\alpha^{(1+\epsilon_4)}\right)^n + \tilde{\mathcal{O}}\left(\lambda^{(1+c_4)}\alpha^{(1+2/3\epsilon_4+1/3\lg_\alpha^\beta)}\right)^n$$

where α, ϵ_1, ϵ_2, ϵ_3, and ϵ_4 must be set such that all inequalities in Figure 5.1 hold, and λ, c_1, c_2, c_3, c_4, and β must be set such that every inequality of Lemma 5.4 holds.

The values

$$\epsilon_1 = 3.3512,\ \epsilon_2 = 4.0202,\ \epsilon_3 = 4.4664,\ \epsilon_4 = 4.7164,\ \alpha = 1.105579\ ,$$

$$c_1 = 2.92887,\ c_2 = 3.81637,\ c_3 = 4.24856,\ c_4 = 4.46431,\ \lambda = 1.08147\ ,$$

$$\beta = 1.6344$$

satisfy the mentioned inequalities, and result in the running time of $\tilde{\mathcal{O}}(2.73929^n)$. Therefore, our proposed algorithm runs in $\tilde{\mathcal{O}}(2.73929^n)$ time. □

In this chapter, we gave an exact algorithm for computing MDDN$(G, 3)$ in time $\tilde{\mathcal{O}}(2.739^n)$. In a recent paper by Riege and Rothe, this result was improved to $\tilde{\mathcal{O}}(2.695^n)$, using a completely different approach [52]. One possible direction for future work is to combine these methods to obtain a faster algorithm. Another alternative is to work on exact algorithms for other k's. Currently, the only exact algorithm for general k's is due to Fomin et al., who solve the problem in $\tilde{\mathcal{O}}(2.88^n)$ time.

```
ALL-MINIMAL-SET-COVERS(U, S, ℓ, C)
 1  if ℓ = 0 then
 2     return C
 3  if condition of Case 0 then
 4     return ALL-MINIMAL-SET-COVERS(U − {u}, S − {S_1}, ℓ − 1, {S_1} ∪ C)
 5  if condition of Case 1 then
 6     return ⋃_{i=1}^{r} (ALL-MINIMAL-SET-COVERS(U − {u}, S − {S_1, ..., S_r}, ℓ − 1, {S_i} ∪ C))
 7  if condition of Case 2 then
 8     A_1 ← ALL-MINIMAL-SET-COVERS(U, S − {S_1}, ℓ, C)
 9     A_2 ← ALL-MINIMAL-SET-COVERS(U_1, (S − {S_1})|U_1, ℓ − 1, {S_1} ∪ C)
10     return A_1 ∪ A_2
11  if condition of Case 3 then
12     A_1 ← ALL-MINIMAL-SET-COVERS(U, S − {S_1}, ℓ, C)
13     A_2 ← ALL-MINIMAL-SET-COVERS(U_1, (S − {S_1})|U_1, ℓ − 1, {S_1} ∪ C)
14     return A_1 ∪ A_2
15  if condition of Case 4 then
16     if condition of Case 4A then
17        A_1 ← ALL-MINIMAL-SET-COVERS(U_1, S − {S_1, S_2}, ℓ − 1, {S_1} ∪ C)
18        A_2 ← ALL-MINIMAL-SET-COVERS(U_2, S − {S_1, S_2}|U_2, ℓ − 1{S_2} ∪ C)
19        return A_1 ∪ A_2
20     if condition of Case 4B then
21        A_1 ← ALL-MINIMAL-SET-COVERS(U_1, S − {S_1, S_2}|U_1, ℓ − 1, {S_1} ∪ C)
22        A_2 ← ALL-MINIMAL-SET-COVERS(U_2, S − {S_1, S_2}|U_2, ℓ − 1, {S_2} ∪ C)
23        return A_1 ∪ A_2
24     if condition of Case 4C then
25        A_1 ← ALL-MINIMAL-SET-COVERS(U_1, S − {S_1, S_2}|U_1, ℓ − 1, {S_1} ∪ C)
26        A_2 ← ALL-MINIMAL-SET-COVERS(U_2, S − {S_1, S_2}|U_2, ℓ − 1, {S_2} ∪ C)
27        A_3 ← ALL-MINIMAL-SET-COVERS(U_3, (S − S_1, S_2)|U_3, ℓ − 2, {S_1, S_2} ∪ C)
28        return A_1 ∪ A_2 ∪ A_3
```

Figure 5.2: (con't in the next page) An algorithm to enumerate minimal set covers of size at most ℓ in $(U, \mathcal{S})$. It is first called with $C = \emptyset$.

29 **if** condition of Case 5 **then**
30 $A_1 \leftarrow$ All-Minimal-Set-Covers$(U, \mathcal{S} - \{S_1\}, \ell, C)$
31 $A_2 \leftarrow$ All-Minimal-Set-Covers$(U_1, (\mathcal{S} - \{S_1\})|U_1, \ell - 1, \{S_1\} \cup C)$
32 **return** $A_1 \cup A_2$
33 **if** condition of Case 6 **then**
34 **if** condition of Case 6A **then**
35 $A_1 \leftarrow$ All-Minimal-Set-Covers$(U, \mathcal{S} - \{S_2\}, \ell, C)$
36 $A_2 \leftarrow$ All-Minimal-Set-Covers$(U_2, (\mathcal{S} - \{S_1, S_2\})|U_2, \ell - 1, \{S_2\} \cup C)$
37 **return** $A_1 \cup A_2$
38 **if** condition of Case 6B **then**
39 $A_1 \leftarrow$ All-Minimal-Set-Covers$(U, \mathcal{S} - \{S_3\}, \ell, C)$
40 $A_2 \leftarrow$ All-Minimal-Set-Covers$(U_3, (\mathcal{S} - \{S_1, S_3\})|U_3, \ell - 1, \{S_3\} \cup C)$
41 **return**
42 **if** condition of Case 7 **then**
43 $A_1 \leftarrow \bigcup_{i=1}^{3}$(All-Minimal-Set-Covers$(U_i, (\mathcal{S} - \{S_i\})|U_i, \ell - 1, \{S_i\} \cup C)$)
44 $A_2 \leftarrow$ All-Minimal-Set-Covers$(U'_1, (\mathcal{S} - \{S_2, S_3\})|U'_1, \ell - 2, \{S_2, S_3\} \cup C)$)
45 $A_3 \leftarrow$ All-Minimal-Set-Covers$(U'_2, (\mathcal{S} - \{S_1, S_3\})|U'_2, \ell - 2, \{S_1, S_3\} \cup C)$)
46 $A_4 \leftarrow$ All-Minimal-Set-Covers$(U'_3, (\mathcal{S} - \{S_1, S_2\})|U'_3, \ell - 2, \{S_1, S_2\} \cup C)$)
47 $A_5 \leftarrow$ All-Minimal-Set-Covers$(U'', (\mathcal{S} - (\{S_1, S_2, S_3\})|U'', \ell - 3, \{S_1, S_2, S_3\} \cup C)$
48 **return** $A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5$
49 **if** condition of Case 8 **then**
50 $A_1 \leftarrow$ All-Minimal-Set-Covers$(U, \mathcal{S} - \{S_1\}, \ell, C)$
51 $A_2 \leftarrow$ All-Minimal-Set-Covers$(U_1, (\mathcal{S} - (\{S_1\} \cup \mathcal{S}_u))|U_1, \ell - 1, \{S_1\} \cup C)$
52 $A_3 \leftarrow$ All-Minimal-Set-Covers$(U_1, (\mathcal{S} - (\{S_1\} \cup \mathcal{S}_v))|U_1, \ell - 1, \{S_1\} \cup C)$
53 **return** $A_1 \cup A_2 \cup A_3$

Figure 5.2: (con't)

```
ALL-MINIMAL-DOMINATING-SETS(G = (V, E), X ⊆ V, ℓ)
54  U ← V
55  S ← ∅
56  n ← |X|
57  for i ← 1 to n do
58      S_i = {v_i} ∪ N(v_i), where v_i is the ith vertex in X
59      S ← S ∪ {S_i}
60  enum ← ALL-MINIMAL-SET-COVERS(U, S, ℓ, ∅)
61  for (every set cover C ∈ enum) do
62      Replace each subset S_i in C with v_i
63  return enum
```

Figure 5.3: An algorithm enumerating minimal dominating sets of size at most ℓ in a graph G that is only selected from vertices in $X \subseteq V(G)$

```
THREE-DOMATIC-PARTITION(G = (V, E))
64  for i ← 1 to ⌈n/3⌉ do
65      enum1 ← ALL-MINIMAL-DOMINATING-SETS(G, V, i)
66      for (every D_1 ∈ enum1) do
67          enum2 ← ALL-MINIMAL-DOMINATING-SETS(G, V − D_1, n)
68          for (every D_2 ∈ enum2) do
69              D_3 ← V − D_1 − D_2
70              if (D_3 is a dominating set for G) then
71                  return {D_1, D_2, D_3}
72  return ∅
```

Figure 5.4: Our new algorithm for the three domatic number problem. This algorithm returns three disjoint dominating sets for the input graph, if $\mathrm{D}(G) \geq 3$.

Chapter 6

Approximation Algorithm

In this chapter, we study the approximability aspects of the MDDN problem.

In the following sections, we first prove an inapproximability result for the MDDN problem in the general case. In other words, we find a ratio t for which no t-approximation algorithm can be designed. Then, we show that the same result holds for split graphs, 2-connected graphs, and planar bipartite graphs of degree four. Next, we develop a greedy algorithm that returns a 3-approximation for the problem. The next section is devoted to approximation algorithms we can design using distance-approximating graphs, for special families of input graphs. In the final section, we have a discussion on the best approximation ratio and possible future work.

6.1 An Inapproximability Result

In this section, we prove it is hard to approximate the MDDN problem within a ratio better than two, unless P = NP.

There are three techniques to prove an inapproximability result: using the celebrated technique introduced by Arora et al. in 1992 [3], reducing the problem to a known inapproximable problem by a special kind of reduction called "gap-

preserving reduction" [2], or directly reducing an NP-complete problem to the problem of finding an approximate solution for the main problem.

Using the last approach, we prove the following inapproximability result for the MDDN problem.

Theorem 6.1. *For any fixed $k \geq 3$, the problem of computing MDDN(G, k) does not have an $(2-\epsilon)$-approximation algorithm for any $\epsilon > 0$, unless $P = NP$.*

Proof. We suppose that a $(2-\epsilon)$-approximation algorithm for the MDDN problem exists. If we execute this algorithm on an integer k and a graph G, it should compute a value between $\text{MDDN}(G, k)$ and $(2-\epsilon)\cdot\text{MDDN}(G, k)$. We consider the two possible cases $\text{MDDN}(G, k) = 1$ and $\text{MDDN}(G, k) > 1$. If $\text{MDDN}(G, k) = 1$, then the value returned of the algorithm cannot exceed $2-\epsilon$. Thus, the algorithm returns 1. Otherwise, since the algorithm cannot return a value less than the optimum value, it does not return 1. Consequently, based on the value returned by the algorithm we can check whether $\text{MDDN}(G, k) = 1$ in polynomial time. According to Corollary 3.8, this is a contradiction, unless P = NP. □

The proof results in a similar inapproximability result if we restrict the input graph to split graphs, 2-connected graphs, or planar bipartite graphs of degree four, and use Corollaries 3.11, 3.12, and 3.9 to prove the contradiction, respectively:

Theorem 6.2. *For any fixed $k \geq 3$, the problem of computing MDDN(G, k) does not have an $(2-\epsilon)$-approximation algorithm for any $\epsilon > 0$, even for split graphs and 2-connected graphs, unless $P = NP$.*

Theorem 6.3. *The problem of computing MDDN$(G, 3)$ does not have an $(2-\epsilon)$-approximation algorithm for any $\epsilon > 0$, even for planar bipartite graphs of degree four, unless $P = NP$.*

In summary, there is not any polynomial-time algorithm approximating the MDDN problem within ratio smaller than 2, even if the input graph is selected from split graphs, 2-connected graphs, or planar bipartite graphs of maximum degree four.

6.2 A Greedy 3-Approximation Algorithm

We develop an algorithm, specified in Figure 6.1, that gives a 3-approximation for the MDDN problem. As an example, we have illustrated how this algorithm works for the graph shown in Figure 6.2. The algorithm first finds the minimum distance MinDistance for which $\delta(G^{\text{MinDistance}}) \geq k-1$, that is 2 in the example. Then, it gives a k-partitioning in $G^{\text{MinDistance}}$ by choosing an unpartitioned vertex v having no partitioned neighbor at each step and assigning partitions 1 to k to $k-1$ of neighbors of v and v. In the figure, the selected v's in the first and second images are specified by partition number $k = 3$. After each selection, the vertices with partitioned neighbors are circled to show which vertices cannot be chosen as the next v. So, in third image, there is not any choice left for v, and all vertices are put in partition 1 according to steps 10 and 11 of the algorithm.

In the remainder of this section, we prove that this algorithm is a 3-approximation algorithm.

Theorem 6.4. *Suppose that k is an integer and G is a graph with minimum degree at least $k-1$. Then,* Greedy-Partitioning(G, k) *partitions $V(G)$ into k disjoint dominating sets in G^3.*

Proof. Greedy-Partitioning may not be applicable on some input parameters, since it may not be possible to choose $k-1$ neighbors in step 5. However, assuming that $\delta(G) \geq k-1$, this problem can not occur.

We prove that the set of vertices put in the same partition i, for any i between 1 and k, is a dominating set for G^3. In other words, every vertex in V either is in partition i, or has a neighbor in G^3 assigned to partition i. In the rest of the proof, whenever we call a vertex v a *good vertex* we mean that v is assigned to partition i. Also, when we say that a vertex v has a *good neighbor*, we mean that v has a neighbor which is a good vertex.

We check graph vertices in three distinct groups: vertices chosen as v in step 4; vertices selected as the neighbor set of a v in step 5; vertices in Unpartitioned in

Precondition: $\delta(G) \geq k-1$.

APPROXIMATE-PARTITIONING$(G=(V,E),k)$

73 MinDistance $\leftarrow \min\{d \mid \ \forall v \in V \ \deg(v) \geq k-1 \ \text{ in } G^d\}$.

74 **return** GREEDY $-$ PARTITIONING$(G^{\text{MinDistance}},k)$

GREEDY-PARTITIONING$(G=(V,E),k)$

75 Unpartitioned $\leftarrow V$

76 **while** (there exists v in Unpartitioned

77 with all neighbors in Unpartitioned) **do**

78 $v \leftarrow$ a vertex in Unpartitioned with all neighbors in Unpartitioned

79 Select $k-1$ neighbors $\{v_1, v_2, \ldots, v_{k-1}\}$ of v.

80 Unpartitioned $\leftarrow$ Unpartitioned $-\{v_1, \ldots, v_{k-1}\} - \{v\}$

81 **for** $i \leftarrow 1$ *to* $k-1$ **do**

82 Partition$(v_i) \leftarrow i$

83 Partition$(v) \leftarrow k$

84 **for** (every vertex $u \in$ Unpartitioned) **do**

85 Partition$(u) \leftarrow 1$

86 **return** Partition

Figure 6.1: A 3-approximation algorithm for the MDDN problem

step 10 of the algorithm, i.e. the vertices that are not partitioned in the while loop. According to steps 7, 8 and 9, every vertex in the first group is good or has a good neighbor. As for the second group, note that every such vertex is a neighbor of a first group vertex. Consequently, every vertex in the second group is at most two nodes away from a good vertex. Therefore, it is good or has a good neighbor in G^3. Now, it remains to check the third group. Based on the while condition, whenever the algorithm reaches step 10, every vertex in Unpartitioned is connected to at least one partitioned vertex. In addition, the only partitioned vertices at step 10

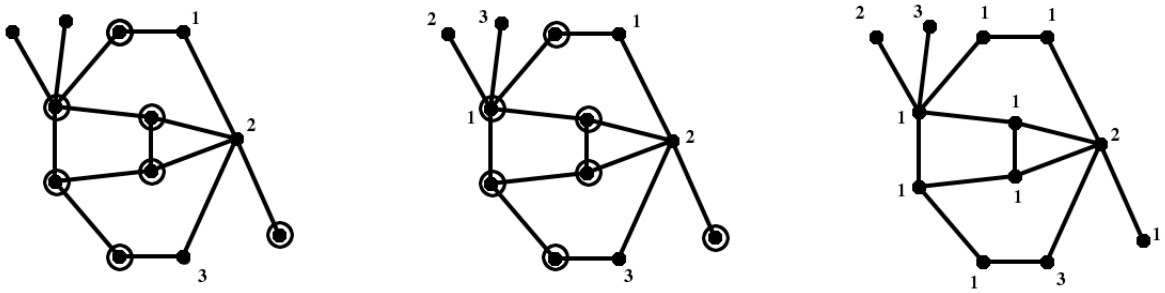


Figure 6.2: The status of vertex partitions in each iteration of the Approximate-Partitioning algorithm. k is 3 in this example. The numbers represent partitions and the vertices with a partitioned neighbor are circled.

are group 1 and 2 vertices. As mentioned before, each of these partitioned vertices is at most two nodes away from a good node. Therefore, every vertex of the third group is at most three nodes away from a good vertex and hence, either is a good vertex itself, or has a good neighbor in G^3.

□

Theorem 6.5. *For every graph $G = (V, E)$ and positive integer k,* Approximate-Partitioning(G, k) *returns a k-partitioning for G corresponding to a 3-approximation for MDDN(G, k).*

Proof. At the first step of Approximate-Partitioning, MinDistance is set to the minimum distance d satisfying $\delta(G^d) \geq k - 1$. Therefore, when, at step 2, Greedy-Partitioning is called with $(G^{\text{MinDistance}}, k)$, the precondition of Theorem 6.4 holds for the input parameters. Therefore, we can say that Greedy-Partitioning$(G^{\text{MinDistance}}, k)$ partitions V into k disjoint dominating sets of $G^{\text{MinDistance}^3} = G^{3\cdot\text{MinDistance}}$. So, we have obtained the upper bound 3·MinDistance on MDDN(G, k). Besides, according to Lemma 2.33, the minimum d satisfying $k - 1 \leq \delta(G^d)$ is at most MDDN(G, k). Since MinDistance is equal to $\min\{d \mid k - 1 \leq \delta(G^d)\}$, MinDistance $\leq$ MDDN(G, k).

Combining the two results obtained above, we have MinDistance $\leq$ MDDN$(G,k) \leq$ $3 \cdot$ MinDistance. Hence, MinDistance is a 3-approximation for MDDN(G,k), and Approximate-Partitioning gives a corresponding k-partitioning for this 3-approximating value. □

6.3 Approximation Algorithms and Special Families of Graphs

As described in Section 3, the MDDN problem can be solved on strongly chordal graphs, including trees, interval graphs, block graphs, and directed path graphs, in polynomial time. This motivates us to replace the input graph G with a graph R in one of these special families with almost the same optimum value, i.e. MDDN(R,k) is approximately equal to MDDN(G,k), and solve the problem for R instead of G.

The question is in which properties R should be similar to G to be a good representative of G in the MDDN problem.

Suppose that for all pairs of vertices (u,v), $u,v \in V(G)$, $d_R(u,v)$ is approximately $d_G(u,v)$. Then, a dominating set in G^ℓ is probably a dominating set in $R^{\ell'}$, for an ℓ' approximately equal to ℓ, and vice verse. Therefore, selecting R to be a spanning subgraph of G that approximates the distances in G might be useful in designing approximation algorithms.

In the following, we list the approximation algorithms we can obtain based on known results on distance-approximating spanners. In the discussion section (6.4), we explain another choice of R and specify a proposition sufficient to prove a 2-approximation algorithm for the MDDN problem in general.

The following theorem clarifies the connection between finding (t,c)-spanners for special families of graphs and designing (t,c)-approximation algorithms for the MDDN problem. We use a lemma and its corollary as the building blocks of this theorem:

Lemma 6.6. *Suppose that G is a graph and R is a distance-approximating graph of ratio (t, c) for G. Then, MDDN$(R, k) \leq t \cdot$MDDN$(G, k) + \ c$.*

Proof. We claim that every dominating set in a power of G, say G^i, is also a dominating set for $R^{t \cdot i + c}$. Assume that a set of vertices $U \subseteq V(G)$ covers the vertices of G^i. It means that every vertex $v \in V(G)$ is within distance i of a vertex $u \in U$ in G. Due to the definition of a distance-approximating graph of ratio (t, c), $d_R(v, u) \leq t \cdot i + c$. Hence, U is a dominating set for $R^{t \cdot i + c}$, and we are done.

As a result, every set of disjoint dominating sets in the ith power of G is a set of disjoint dominating sets in $R^{t \cdot i + c}$. Therefore, $\mathrm{D}(G^i) \leq \mathrm{D}(R^{t \cdot i + c})$ for any i.

According to Definition 2.28, $\mathrm{MDDN}(G, k)$ is the minimum r for which $\mathrm{D}(G^r) \geq k$. Since $k \leq \mathrm{D}(G^r) \leq \mathrm{D}(R^{t \cdot r + c})$, the minimum distance d making $\mathrm{D}(R^d) \geq k$ is at most $t \cdot r + c$. This statement is exactly what we wanted to prove. □

Corollary 6.7. *If R is a subgraph of a graph G, then MDDN$(G, k) \leq$ MDDN(R, k).*

Proof. Since R is a subgraph of G, for every pair of vertices $u, v \in V(G)$, $d_G(u, v) \leq d_R(u, v)$. Equivalently, G is a distance-approximating graph of ratio $(1, 0)$ for R, and Lemma 6.6 proves the corollary. □

Theorem 6.8. *Suppose that $\mathcal{F}$ is a family of graphs for which there is a polynomial-time algorithm A to find a (t, c)-spanner R. Moreover, the domatic number of R and its powers can be determined in polynomial time. Then, restricting the input graph to be in $\mathcal{F}$, we can give a (t, c)-approximation algorithm for the MDDN problem.*

Proof. We consider a graph G in $\mathcal{F}$. Running A on G, we obtain a (t, c)-spanner R for G.

Due to the previous lemma and corollary, as R is a distance-approximating graph of ratio (t, c) for G and also a subgraph of G, $\mathrm{MDDN}(G, k) \leq \mathrm{MDDN}(R, k) \leq t \cdot \mathrm{MDDN}(G, k) + c$. Consequently, an algorithm returning $\mathrm{MDDN}(R, k)$ is a (t, c)-approximation for the $\mathrm{MDDN}(G, k)$ problem.

According to Lemma 2.39, since the domatic number of R and its powers can be determined in polynomial time, MDDN(R, k) can be computed in polynomial time. The polynomial-time algorithm that computes MDDN(R, k) is our desired approximation algorithm.

□

Choosing R from trees, we can use the known results on tree (t, c)-spanners mentioned in the preliminaries section (Theorems 2.9, 2.10, 2.11, and 2.12) to design approximation algorithms for the MDDN problem:

Corollary 6.9. *We can obtain the following approximation algorithms for the MDDN problem:*

1. $(3, 0)$*-approximation, as well as* $(1, 4)$*-approximation, for permutation graphs*
2. $(1, 2)$*-approximation for distance-hereditary graphs*
3. $(1, 4)$*-approximation for cocomparability graphs*
4. $(1, 3)$*-approximation for dually chordal graphs*

As mentioned in the preliminaries (Theorem 2.13), although it is not possible to find tree (t, c)-spanners for all chordal graphs, Brandstadt et al. succeeded to obtain a distance-approximating graph R of ratio $(1, 2)$ for every chordal graph G, where R is a spanning tree for G^2 rather than G. Their theorem is useful in designing approximation algorithms for the MDDN problem on chordal graphs:

Lemma 6.10. *For any graph* G *and any subgraph* R *of* G^2, *MDDN*$(G, k) \leq 2 \cdot$*MDDN*(R, k).

Proof. In an arbitrary power of G, say G^i, the vertices of distance at most i in G are connected by edges. As a result, a path of length ℓ is reduced to a path of length $\lceil \ell / i \rceil$. That is, for every pair of vertices in $V(G)$, $d_G(u, v) \leq i \cdot d_{G^i}(u, v)$. Now, limiting our focus to the $i = 2$ case, we have $d_G(u, v) \leq 2 \cdot d_{G^2}(u, v)$ for any $u, v \in V(G)$. Since R is a subgraph of G^2, $d_{G^2}(u, v) \leq d_R(u, v)$ also holds,

Graph Class	Approximation Ratio
permutation graphs	$(1, 4)$
distance-hereditary graphs	$(1, 2)$
cocomparability graphs	$(1, 4)$
dually chordal graphs	$(1, 3)$
chordal graphs	$(2, 4)$
general graphs	$(3, 0)$

Table 6.1: A list of approximation algorithms for the MDDN problem for special classes of graphs.

leading to $d_G(u, v) \leq 2 \cdot d_R(u, v)$ for every $u, v \in V(G)$. Therefore, G is a distance-approximating graph of ratio $(2, 0)$ for R, and we can use Lemma 6.6 to conclude the result. □

Theorem 6.11. *There exists a* $(2, 4)$*-approximation algorithm for computing MDDN*(G, k) *for chordal graphs.*

Proof. We first compute the spanning tree T of G^2 mentioned in Theorem 2.13. Due to Lemma 6.6, $\text{MDDN}(T, k) \leq \text{MDDN}(G, k) + 2$. However, unlike in the previous theorem, $\text{MDDN}(T, k)$ is not necessarily greater than or equal to $\text{MDDN}(G, k)$, since the distances in T might be less than the distances in G. Whereas according to Theorem 2.13, T is a spanning tree of G^2, and according to Lemma 6.10, $\text{MDDN}(G, k) \leq 2 \cdot \text{MDDN}(T, k)$. Combining it with the previous inequality, we have:

$$\text{MDDN}(G, k) \leq 2 \cdot \text{MDDN}(T, k) \leq 2 \cdot \text{MDDN}(G, k) + 4$$

Consequently, an algorithm returning $2 \cdot \text{MDDN}(T, k)$ is a $(2, 4)$-approximation algorithm for chordal graphs. □

A summary of the approximation results on special families of graphs is shown in Table 6.1.

6.4 Discussion

We have developed a 3-approximation algorithm and proved that the approximation ratio cannot be less than 2. Hence, the best approximation ratio for the MDDN problem, denoted by ρ, is a value between 2 and 3. The question is how we can close the gap between the current upper and lower bounds for ρ.

It is worth mentioning that there is not any gap for the case $k = 3$, since there is a 2-approximation algorithm for this case: due to Theorem 2.31, $\text{MDDN}(G, 3) \leq 2$. Hence, an algorithm always returning 2 is a 2-approximation for the problem.

It might be the case that the best approximation ratio is 2 for all k's; but how can we prove that? One choice is to exploit an approach similar to that in Section 6.2. Our 3-approximation algorithm is based on Theorem 6.4. This theorem states that for every graph G, $\text{D}(G^3) \geq \delta(G)$ and provides an algorithm that finds a $\delta(G)$-partitioning for G^3 in polynomial time. A similar result on G^2, i.e. $\text{D}(G^2) \geq \delta(G)$, will give a 2-approximation algorithm. This condition is not necessary, but only sufficient, to obtain a 2-approximation algorithm. However, although we have not proved this result, we have not yet been able to find a graph G for which $\text{D}(G^2) \geq \delta(G)$, either.

One approach to obtain a lower bound for $\text{D}(G^2)$ is to design an algorithm specifying a spanning subgraph $R \subseteq G$ of a graph family $\mathcal{F}$ of known $\text{D}(R^2)$. Since $\text{D}(G^2) \geq \text{D}(R^2)$, if such a subgraph ensures $\text{D}(R^2) \geq \delta(G)$, the desired $\text{D}(G^2) \geq \delta(G)$ will hold, and ρ will be proved to be 2. As mentioned in Corollary 3.16, for any power of a strongly chordal graph S, $\text{D}(S^i) = \delta(S^i)+1$. Therefore, the strongly chordal family of graphs is a good candidate for $\mathcal{F}$. As a result, if the answer to the following question is yes, we can achieve a 2-approximation algorithm for the MDDN problem:

Open Question 1. *Does every graph G have a strongly chordal spanning subgraph R such that $\delta(R^2) \geq \delta(G)$?*

The response to this question is no if we restrict R to the family of trees; as a counterexample, we consider graph G produced by removing a perfect matching

from a complete n-vertex graph. In this example, $\delta(G) = \Delta(G) = n - 2$, and we claim that no tree subgraph $T \subseteq G$ has $\delta(T^2) \geq \delta(G) = n - 2$. Every tree T must have exactly $n-1$ edges and at least one leaf. Suppose that ℓ is a leaf of T. To have a minimum degree of $n - 2$ in T^2, ℓ must be connected to a vertex u with at least $n - 3$ neighbors other than ℓ. Since $\Delta(G) = n - 2$, u is connected to *exactly* $n - 2$ vertices in T. Thus, there is only one vertex v that is not among the neighbors of u in T. Up to now, we have identified $n - 2$ edges in T, and hence, we are only allowed to add one more edge to T. Therefore, $deg_T(v) \leq 1$, and v is also a leaf in T. Using the same argument as described for ℓ, we conclude that v must have a neighbor of degree $n - 2$. The only $(n - 2)$-degree vertex in T is u which is not connected to v. Hence, we proved that G does not have any tree subgraph T with $\delta(T^2) \geq \delta(G)$.

We have not been able to answer this question for strongly chordal R's. So, a possible direction for future work is to investigate this question.

Chapter 7

Conclusion

In this thesis, we studied the d-domatic problem, which is the problem of finding the maximum number of disjoint dominating sets for the dth power of the input graph, from a different angle: we considered the problem of computing the minimum d for a given number of disjoint dominating sets, k, and an input graph G such that $V(G)$ can be partitioned into k disjoint dominating sets for G^d. This problem is indeed a multi-facility location problem with the worst case access time criterion.

In previous chapters, we studied the status of this problem, referred to as the MDDN problem, on various graph families, developed an exact (exponential) algorithm for the problem, and investigated the approximability aspects of the problem.

We showed that the MDDN problem is in P for strongly chordal graphs, based on a result on the domatic number problem and a result on power graphs of strongly chordal graphs. Furthermore, we proved that computing MDDN(G, k) can be done in polynomial-time for bounded tree-width graphs, when k is a constant. In addition, using a reduction from the domatic number problem to the MDDN problem, we showed that the problem is be NP-complete for circular-arc graphs, bipartite graphs, and split graphs. We gave a reduction from the k-coloring problem to the MDDN problem, proving its NP-completeness on 2-connected graphs and split graphs for any fixed $k \geq 3$, and on planar bipartite graphs of degree four for $k = 3$. Furthermore, we could prove the NP-completeness of the domatic number problem

for planar bipartite graphs of maximum degree four, which was not known before.

As we explained in Section 2, Zelinka gives an upper bound for the problem [59]; we proved Zelinka's upper bound using a slightly simpler argument. It is worth mentioning that this is the best bound possible for the general case, since it cannot be improved for paths.

We developed a new exponential-time exact algorithm for computing MDDN$(G, 3)$ running in $\tilde{\mathcal{O}}(2.7393^n)$ time, improving the previous $\tilde{\mathcal{O}}(2.8805^n)$ best running time. Unfortunately, before our result was published, Riege et al. developed a faster algorithm independently, using a completely different approach [52]. Their algorithm's running time is $\tilde{\mathcal{O}}(2.695^n)$.

As for approximation algorithms, we proved that the problem cannot be approximated with ratio less than 2. The result holds for split graphs, 2-connected graphs, and planar bipartite graphs of maximum degree 4. In addition, we designed a greedy 3-approximation algorithm for the problem. Using the known results on distance-approximating spanners, we also designed some approximation algorithms for permutation graphs, cocomparability graphs, distance-hereditary graphs, and dually chordal graphs. Finally, we proposed an approximation algorithm for the problem on chordal graphs.

7.1 Future Work

Much is still left to be studied:

As mentioned in the introduction, the MDDN problem is a restricted version of multi-facility location with worst-case access time criterion. In similar problems with average access time criterion, other generalizations have been considered, some of which seem reasonable for the MDDN problem, too. For example, the problem would be more applicable if the input graph is directed or weighted, or we are not allowed to choose the dominating sets from all vertices in the graph, but only from a limited subset of vertices. Some of the algorithms in this thesis work with the second and third conditions, too.

Even for the current version of the problem there are some directions possible for future work:

In Chapter 3, we could prove that MDDN(G, k) can be computed in polynomial time for graphs of bounded tree-width, when k is considered as a constant. One question is whether the problem is still in P when k is part of the input.

Another possible question that can be studied is whether the ideas of Riege and Rothe and of our new exact algorithm can be combined to develop a faster exact algorithm to compute MDDN$(G, 3)$. As mentioned before, the running time of our proposed exact algorithm was improved by Riege and Rothe recently. Since the approaches taken are different, it might be possible to find a good combination of these two approaches.

In addition, the only exact algorithm for k's other than 3 is due to Fomin et al. that runs in time $\tilde{\mathcal{O}}(2.8805^n)$. So, there is not much work done in this part, yet.

At the end of Chapter 6, we had a discussion on the best approximation ratio for the MDDN problem, and presented an open question. The affirmative answer to this question is perhaps the key point in designing a 2-approximating algorithm for the MDDN problem for general graphs.

Bibliography

[1] N. Alon, G. Fertin, A. L. Liestman, T. C. Shermer, and L. Stacho. Factor d-domatic colorings of graphs. *Discrete Mathematics*, 262(1):17–25, 2003.

[2] S. Arora and C. Lund. Hardness of approximations. *Approximation algorithms for NP-hard problems*, pages 399–446, 1997.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of ACM*, 45(3):501–555, 1998.

[4] V. Arya, N. Garg, R. Khandekar, K. Munagala, and V. Pandit. Local search heuristic for k-median and facility location problems. In *STOC '01: Proceedings of the 33rd Annual Symposium on Theory of Computing*, pages 21–29. ACM Press, 2001.

[5] I. D. Baev and R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. In *SODA '01: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 661–670, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[6] M. A. Bonucelli. Dominating sets and domatic number of circular arc graphs. *Discrete Applied Mathematics*, 12:203–213, 1985.

[7] M. Borowiecki and M. Kuzak. On the k-stable and k-dominating sets of graphs. In M. Borowiecki, Z. Skupień, and L. Szamkolowicz, editors, *Graphs, Hyper-*

graphs and Block Systems: Proceedings of the Symposium on Combinatorial Analysis, pages 134–143, Univ. Zielona Góra, 1976.

[8] A. Brandstädt, V. Chepoi, and F. F. Dragan. Distance approximating trees for chordal and dually chordal graphs. *Journal of Algorithms*, 30(1):166–184, 1999.

[9] A. Brandstädt, F. F. Dragan, H. Le, and V. B. Le. Tree spanners on chordal graphs: complexity and algorithms. *Theoretical Computer Science*, 310(1-3):329–354, 2004.

[10] A. Brandstädt, F. F. Dragan, H. Le, V. B. Le, and R. Uehara. Tree spanners for bipartite graphs and probe interval graphs. In *WG '03: Proceedings of the 29th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 106–118. Springer-Verlag, 2003.

[11] R. Brigham and R. D. Dutton. Factor domination in graphs. *Discrete Mathematics*, 86:127–136, 1990.

[12] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The Boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, 1988.

[13] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The Boolean hierarchy II: Applications. *SIAM Journal on Computing*, 18(1):95–111, 1989.

[14] V. Chepoi, F. F. Dragan, and C. Yan. Additive spanners for k-chordal graphs. In *CIAC '03: Proceedings of the 5th Conference on Algorithms and Complexity*, pages 96–107. Springer-Verlag, 2003.

[15] E. J. Cockayne and S. T. Hedetniemi. Towards a theory of domination in graphs. *Networks*, 7:247–261, 1977.

[16] E. Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM Journal on Computing*, 28(1):210–236, 1999.

[17] D. Coppersmith and M. Elkin. Sparse source-wise and pair-wise distance preservers. In *SODA '05: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 660–669, Philadelphia, PA, USA, 2005.

[18] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

[19] Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[20] L. J. Cowen. Compact routing with minimum stretch. In *SODA '99: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 255–260, Philadelphia, PA, USA, 1999.

[21] R. G. Downey and M. R. Fellows. *Parameterized Complexity*, chapter Automata and Bounded Treewidth, page 530. Springer-Verlag, 1999.

[22] F. F. Dragan, C. Yan, and I. Lomonosov. Collective tree spanners of graphs. *SIAM Journal on Discrete Mathematics*, 20(1):240–260, 2006.

[23] M. Farber. *Applications of Linear Programming Duality to Problems Involving Independence and Domination*. Ph.D. Thesis, Rutgers University, 1982.

[24] M. Farber. Domination, independent domination, and duality in strongly chordal graphs. *Discrete Applied Mathematics*, 7:115–130, 1984.

[25] U. Feige, M. M. Halldórsson, G. Kortsarz, and A. Srinivasan. Approximating the domatic number. *SIAM Journal on Computing*, 32(1):172–195, 2002.

[26] F. Fomin, F. Grandoni, A. Pyatkin, and A. Stepanov. Bounding the number of minimal dominating sets: a measure and conquer approach. In *ISAAC '05: Proceedings of the 16th International Symposium on Algorithms and Computation*, volume 3827, pages 573–582. Springer-Verlag, 2005.

[27] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: Domination - a case study. In *ICALP '05: Proceedings of 32nd International Colloquium on Automata, Languages and Programming*, pages 191–203, 2005.

[28] F. V. Fomin, F. Grandoni, and D. Kratsch. Some new techniques in design and analysis of exact (exponential) algorithms. *Bulletin of the EATCS*, 87:47–77, 2005.

[29] S. Fujita, T. Kameda, and M. Yamashita. A resource assignment problem on graphs. In *ISAAC '95: Proceedings of the 6th International Symposium on Algorithms and Computation*, pages 418–427, London, UK, 1995. Springer-Verlag.

[30] S. Fujita, M. Yamashita, and T. Kameda. A study on r-configurations–a resource assignment problem on graphs. *SIAM Journal on Discrete Mathematics*, 13(2):227–254, 2000.

[31] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, New York, USA, 1979.

[32] M. R. Garey, D. S. Johnson, and L. J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[33] C. Gavoille. Routing in distributed networks: overview and open problems. *ACM SIGACT News*, 32(1):3652, 2001.

[34] A. Gupta and A. Kumar. Traveling with a Pez dispenser (or, routing issues in MPLS). In *FOCS '01: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 148–157, Washington, DC, USA, 2001.

[35] T. W. Haynes and S. T. Hedetniemi. *Domination in Graphs: Advanced Topics*. Marcel Dekker, New York, USA, 1998.

[36] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, editors. *Fundamentals of Domination in Graphs*. Marcel Dekker, New York, USA, 1998.

[37] P. Indyk. Algorithmic applications of low-distortion geometric embeddings. In *FOCS '01: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 10–33, Washington, DC, USA, 2001.

[38] H. Kaplan and R. Shamir. The domatic number problem on some perfect graph families. *Information Processing Letters*, 49:51–56, 1994.

[39] R. M. Karp. *Complexity of Computer Computations*, chapter Reducibility Among Combinatorial Problems, pages 85–103. Plenum Press, New York, USA, 1972.

[40] M. Korupolu, G. Plaxton, and R. Rajaraman. Placement algorithms for hierarchical cooperative caching. *Journal of Algorithms*, 38(1):260–302, 2001.

[41] D. Kratsch, H. Müllers H. Le, E. Prisner, and D. Wagner. Additive tree spanners. *SIAM Journal on Discrete Mathematics*, 17(2):332–340, 2003.

[42] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis. Joint object placement and node dimensioning for internet content distribution. *Information Processing Letters*, 89(6):273–279, 2004.

[43] E. L. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5:66–67, 1976.

[44] N. Linial. Finite metric spaces – combinatorics, geometry and algorithms. In *Proceedings of the International Congress of Mathematicians III*, pages 573–586, 2002.

[45] A. Lubiw. Γ-*free Matrices*. Master's Thesis, Department of Combinatorics and Optimization, University of Waterloo, 1982.

[46] M. S. Madanlal, G. Venkatesan, and C. P. Rangan. Tree 3-spanners on interval, permutation and regular bipartite graphs. *Information Processing Letters*, 59(2):97–102, 1996.

[47] M. V. Marathe, H. B. Hunt, and S. S. Ravi. Efficient approximation algorithms for domatic partition and on-line coloring of circular arc graphs. *Discrete Applied Mathematics*, 64(2):135–149, 1996.

[48] E. Prisner. Distance approximating spanning trees. In *STACS '97: Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science*, pages 499–510, London, UK, 1997. Springer-Verlag.

[49] Y. Rabinovich and R. Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete and Computational Geometry*, 19(1):79–94, 1998.

[50] T. Riege and J. Rothe. Complexity of the exact domatic number problem and of the exact conveyor flow shop problem. *ACM Computer Science Research Repository*, cs.CC/0212016, 2002.

[51] T. Riege and J. Rothe. An exact 2.9416^n algorithm for the three domatic number problem. In *MFCS '05: Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science*, volume 3618, pages 733–744. Springer-Verlag, 2005.

[52] T. Riege, J. Rothe, H. Spakowski, and M. Yamamoto. An improved exact algorithm for the domatic number problem. In *ICTTA '06: Proceedings of the 2nd IEEE International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1021–1022. IEEE Computer Society Press, 2006.

[53] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Buy-at-bulk network design: approximating the single-sink edge installation problem. In *SODA '97: Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 619–628, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics.

[54] U. Schöning. Algorithmics in exponential time. In *STACS '05: Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science*, volume 3404, pages 36–43. Springer-Verlag, 2005.

[55] U. U. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[56] D. B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, N.J., second edition, 2001.

[57] G. J. Woeginger. Exact algorithms for NP-hard problems: a survey. In *Proceedings of the 5th International Workshop on Combinatorial Optimization*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–207, New York, USA, 2003. Springer-Verlag.

[58] B. Zelinka. Domatic number and degree of vertices of a graph. *Mathematica Slovaca*, 33:145–147, 1983.

[59] B. Zelinka. On k-domatic numbers of graphs. *Czechoslovak Mathematical Journal*, 33:309–313, 1983.