# On Two Combinatorial Optimization Problems in Graphs: Grid Domination and Robustness

by

Elaheh Fata

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

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

In this thesis, we study two problems in combinatorial optimization, the dominating set problem and the robustness problem. In the first half of the thesis, we focus on the dominating set problem in grid graphs and present a distributed algorithm for finding near optimal dominating sets on grids. The dominating set problem is a well-studied mathematical problem in which the goal is to find a minimum size subset of vertices of a graph such that all vertices that are not in that set have a neighbor inside that set. We first provide a simpler proof for an existing centralized algorithm that constructs dominating sets on grids so that the size of the provided dominating set is upper-bounded by $\left\lceil \frac{(m+2)(n+2)}{5} \right\rceil$ for $m \times n$ grids and its difference from the optimal domination number of the grid is upper-bounded by five. We then design a distributed grid domination algorithm to locate mobile agents on a grid such that they constitute a dominating set for it. The basis for this algorithm is the centralized grid domination algorithm. We also generalize the centralized and distributed algorithms for the $k$-distance dominating set problem, where all grid vertices are within distance $k$ of the vertices in the dominating set.

In the second half of the thesis, we study the computational complexity of checking a graph property known as robustness. This property plays a key role in diffusion of information in networks. A graph $G = (V, E)$ is $r$-robust if for all pairs of nonempty and disjoint subsets of its vertices $A, B \subset V$, at least one of the subsets has a vertex that has at least $r$ neighbors outside its containing set. In the robustness problem, the goal is to find the largest value of $r$ such that a graph $G$ is $r$-robust. We show that this problem is **coNP**-complete. En route to showing this, we define some new problems, including the decision version of the robustness problem and its relaxed version in which $B = V \backslash A$. We show these two problems are **coNP**-hard by showing that their complement problems are **NP**-hard.

**Acknowledgements**

I would like to thank my supervisors, Shreyas Sundaram and Stephen Smith, for their guidance, encouragement and helping me to gain belief in myself. I would also like to acknowledge my coauthors, Soroush Hosseini Alamdari and Haotian Zhang.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

In this thesis, we study two problems in combinatorial optimization. Both these problems are defined on graphs, which are powerful models widely used in mathematics, engineering and many other fields of science to represent objects, settings, etc. We start this thesis by going through some basic concepts in graph theory that are later needed to introduce the two studied problems. We then briefly discuss these two problems and continue with the preliminary materials needed throughout this document. Previous work, a summary of our contributions and the overall organization of this thesis are later discussed in this chapter.

## 1.1 Background on Graph Theory

Before discussing the problems studied in this thesis, we provide a background on the material in graph theory that will be needed in defining our problems. Those familiar with basic graph terminology can skip directly to the problem definitions in Section 1.2. Also, more details on these topics can be found in theoretical computer science and combinatorial optimization references, such as [7] and [12].

A graph $G = (V, E)$ is defined as a set of vertices $V$ connected by a set of edges $E \subseteq V \times V$. The vertices that are connected by an edge are called the *endpoints*, *ends* or *end vertices* of that edge and we say that these vertices are *incident* with that edge, and vice versa. Note that there might exist some vertices in a graph that are not incident to any edge.

Graphs can be categorized into two main types: *directed* and *undirected graphs*. A directed graph (usually called a *digraph* for brevity), is a graph in which edges are ordered pair of vertices, meaning that existence of an edge from a vertex $u$ to another vertex $v$ does not necessitate the existence of the edge from $v$ to $u$. These type of graphs can be imagined as streets in cities, where both two-way and one-way streets might be found. The edge from vertex $u$ to vertex $v$ in directed graphs is denoted by $(u, v)$ and the edge from $v$ to $u$ is shown by $(v, u)$.

On the other hand, there are undirected graphs in which edges are not ordered pair of vertices. In other words, in undirected graphs we care about two vertices being connected to each other; whereas, in directed graphs in addition to that the direction of the edge matters. In undirected graphs, an edge connecting vertices $u$ and $v$ is denoted by $uv$ or $\{u, v\}$ to point out edges $uv$ and $vu$ (similarly, $\{u, v\}$ and $\{v, u\}$) are not different.

Notice that all the graphs discussed in this thesis are undirected. Moreover, throughout this document we only consider *simple graphs*, meaning that any pair of distinct vertices are connected by at most one edge, and there is no edge that starts from and ends to the same vertex. In the following, we list some graph theoretic concepts and definitions that are later referred to. All these concepts are defined on a simple undirected graph $G$ with vertex set $V$ and edge set $E$.

**Neighbors** A vertex $u \in V$ is called a *neighbor* of vertex $v \in V$ if $uv \in E$. The set of all neighbors of vertex $v$ is denoted by $N(v)$. Moreover, for a set of vertices $U \subseteq V$, we define $N(U)$ as $\bigcup_{u \in U} N(u)$, that is, the set of all vertices that have a neighbor in $U$.

**Paths and Cycles** A sequence of vertices $v_0 v_1 \dots v_k$ form a *path* with *length* $k$ in graph $G = (V, E)$ if $v_0, \dots, v_k \in V$ and for any $0 \leq i \leq k-1$ edge $v_i v_{i+1} \in E$. The vertex at which the path starts is called its *start point* and the vertex at which it ends is called its *end point*. If no vertex is visited more than once on a path, then that path is called a *simple path*. Moreover, a *cycle* is a path whose start and end points are the same. A *simple cycle* can be defined similarly.

**Connectivity** An undirected graph $G$ is *connected* if and only if for every pair of distinct vertices $u$ and $v$, there is a path from $u$ to $v$. Otherwise, we say that the graph is *disconnected*. Moreover, we say that *connectivity* of graph $G$ is $c$ if and only if for any two distinct vertices of $G$ there are $c$ pairwise disjoint paths connecting them. If the connectivity of the graph is $c$, then for any nonnegative integer $d \leq c$, we say that the graph is *$d$-connected*. By Menger's theorem, the connectivity of graph $G$ is equal to the minimum number of vertices whose removal results in the graph being disconnected.

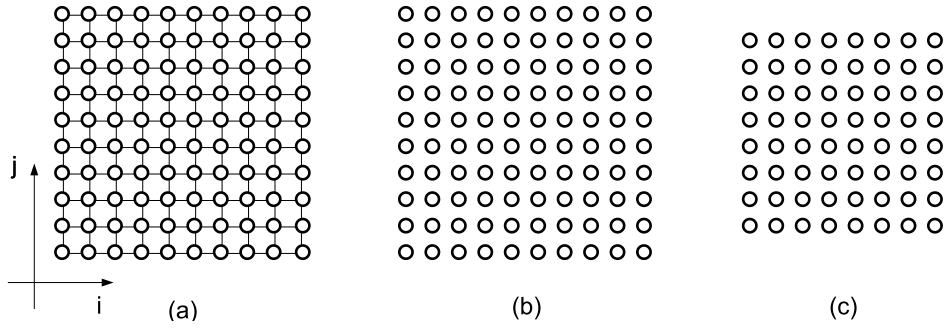



Figure 1.1: Figure (a) shows a $10 \times 10$ grid. In Figure (b), the edges of the grid are removed. For simplicity of presentation we do not demonstrate edges of grids in this document. Figure (c) shows an $8 \times 8$ grid that is the sub-grid of the grid in Figure (a).

**Distance** The *distance* between two vertices in a graph is defined as the number of edges in the shortest path joining these two vertices. Observe that if there exists no such path between two vertices, then the distance between them is taken to be infinite.

**Cliques** A subset $C$ of vertices of a graph form a *clique* if and only if any two vertices in $C$ are joined by an edge.

**Grid Graphs** An $m \times n$ *grid graph* $G = (V, E)$ is defined as a graph with vertex set $V = \{v_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n\}$ and edge set $E = \{v_{i,j}v_{i,j'} | \ |j - j'| = 1\} \cup \{v_{i,j}v_{i',j} | \ |i - i'| = 1\}$; see Figure 1.1. For ease of exposition, we will fix an orientation and labelling of the vertices, so that vertex $v_{1,1}$ is the lower-left vertex and vertex $v_{m,n}$ is the upper-right vertex of the grid. For brevity, we refer to grid graphs as *grids*. Moreover, for an $m \times n$ grid $G$, we define the *boundary* of $G$, denoted by $B(G)$, as the set of vertices with less than 4 neighbours. An $m \times n$ grid $G = (V, E)$ is called a *sub-grid* of an $m' \times n'$ grid $G' = (V', E')$ if $G$ is induced by vertices $v'_{i,j} \in V'$, where $2 \leq i \leq m' - 1$ and $2 \leq j \leq n' - 1$. If $G$ is a sub-grid of $G'$, then $G'$ is called the *super-grid* of $G$ (see Figure 1.1).

**Cuts** For a graph $G = (V, E)$, a *cut* $C = (S, V \backslash S)$ is defined as a partition of nodes of $G$ into two nonempty subsets $S \subset V$ and $V \backslash S$. The *cut-set* of a cut $C = (S, V \backslash S)$, denoted by $\delta(S)$ or $\delta(V \backslash S)$, is defined as the subset of the edges of $G$ with one endpoint in $S$ and the other in $V \backslash S$. Mathematically, we have $\delta(S) = \{uv \in E \mid |S \cap \{u, v\}| = 1\}$. Note that set $S$ can include only one vertex of the graph, say $v$. In this case, for simplicity we use $\delta(v)$ to denote the cut-set, that is, $\delta(\{v\})$.

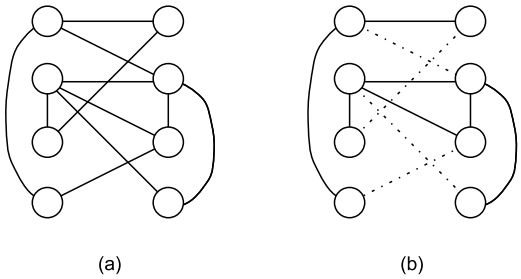



Figure 1.2: The dashed edges in Figure (b) are the subset of the edges of the graph depicted in Figure (a) that make a matching.

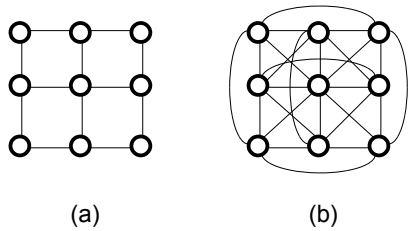



Figure 1.3: Figures (a) and (b) show a $3 \times 3$ grid graph and its second power, respectively. Vertices within distance two are connected to each other in (b).

**Matching** A subset of edges of a graph forms a *matching* if and only if no two edges in that subset share an endpoint; see Figure 1.2.

**Graph Power** For a graph $G = (V, E)$, its *k-th power*, denoted by $G^k = (V', E')$, is a graph with the same vertex set as $G$, i.e., $V = V'$, in which two distinct vertices share an edge if and only if their distance in $G$ is at most $k$ (see Figure 1.3).

## 1.2 Studied Problems

In this thesis, we focus on two problems: (i) dominating set problem, and (ii) robustness problem. These two problems are defined on graphs and they are both motivated by problems in multi-agent systems. For both of these problems it can be assumed that a set of agents located on the graph aim to fulfill some goals related to the objective of the problems. This section contains two parts each dedicated to one of the main problems

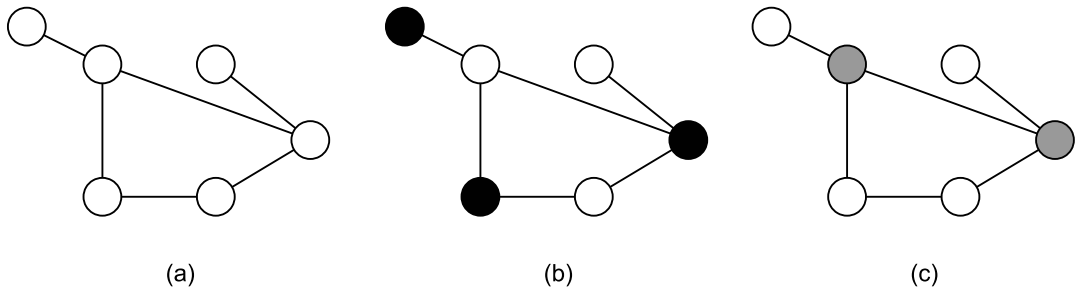



Figure 1.4: The black and grey vertices in Figures (b) and (c), respectively, make two dominating sets with different sizes for the graph depicted in Figure (a). It can be shown that the domination number of the graph is 2 and the grey vertices in Figure (c) correspond to it.

considered in this thesis. A short introduction of each of these problems is provided here and they will be studied in more details in the following chapters.

### 1.2.1 Dominating Set Problem

For a graph $G = (V, E)$, we say that a subset of the vertices of the graph $S \subseteq V$ constitutes a *dominating set* for $G$, if and only if each vertex[1] $v$ of the graph is either in $S$, or shares an edge with a vertex that is in $S$. The vertices in set $S$ are called *dominating vertices* and the vertices who share edges with dominating vertices are called *dominated vertices*. In particular, if a vertex $v \notin S$ is connected to a vertex $u \in S$, then we say that $v$ is *dominated* by $u$. Figure 1.4 demonstrates two different dominating sets for a graph, where one has size two and the other has size three.

In the *dominating set problem*, the goal is to find a subset of the vertices of a given graph $G$ that is a dominating set for $G$ and has the minimum size among all other dominating sets of the graph. A dominating set with minimum cardinality is called an *optimal dominating set* of graph $G$; its cardinality is called the *domination number* of $G$ and is denoted by $\gamma(G)$. Note that although the domination number of a graph, $\gamma(G)$, is unique, there may be different optimal dominating sets.

Many different versions of the dominating set problem have been studied in the literature. Among these are the weighted dominating set problem (where vertices of the graph

[1]We use the terms *vertex* and *node* interchangeably in this document.

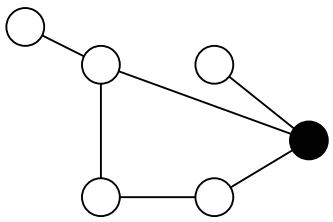


Figure 1.5: The black vertex is a 2-distance dominating set for the graph depicted in Figure 1.4(a).

have weights and the goal is to find a dominating set with minimum weight), connected dominating set problem (in which each vertex in the dominating set should be connected to other vertices in that set), independent dominating set problem (where no two vertices in the dominating set should share an edge), etc. Another variation of the dominating set problem is when a feasible solution is a subset of vertices of a given graph such that any vertex that is not in that set has distance at most $k$ from some vertex in the set. This problem is called the *$k$-distance dominating set problem*, see Figure 1.5. Although all these different versions of the dominating set problem have been intensely studied in the literature, in this thesis we only focus on the original version of this problem as well as the $k$-distance dominating set problem. Moreover, the concentration of this document is on studying the dominating set problem on a special class of graphs, called grid graphs, that is discussed further in the next section.

### 1.2.2 Robustness Problem

Before discussing the robustness problem, we need to introduce some definitions as follows.

**Definition 1.1 ($r$-Reachable)** *Consider a graph $G = (V, E)$ and let $S$ be a subset of its vertices, i.e., $S \subseteq V$. For some $r \in \mathbb{Z}_{\geq 1}$[2], the set $S$ is* $r$-reachable *if there exists at least one vertex $v \in S$ such that $v$ is connected to at least $r$ vertices of $G$ that are not in $S$.*

**Definition 1.2 ($r$-Robust)** *A graph $G$ is* $r$-robust *if for any two disjoint and nonempty subsets of its vertices, at least one of them is $r$-reachable.*

[2]Note that $\mathbb{Z}_{\geq i}$ denotes the set of all integers greater than or equal to $i \in \mathbb{Z}$.

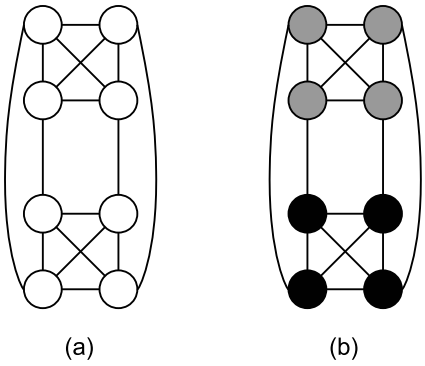



Figure 1.6: Figure (a) shows a graph with connectivity 4. In Figure (b) two subsets of vertices of the graph, one with black vertices and the other with grey vertices, are demonstrated. It can be seen that each vertex is only connected to one vertex outside its set and hence the graph is only 1-robust.

In other words, a graph $G = (V, E)$ is $r$-reachable if for any two nonempty sets $S_1, S_2 \in V$ such that $S_1 \cap S_2 = \emptyset$, at least one of these sets has at least one vertex that is connected to at least $r$ vertices outside its containing set. Note that if a graph is $r$-robust, then for any two disjoint and nonempty subsets of its vertices, at least one of them has a node that shares at least $r$ edges with nodes outside its set and hence gets sufficiently influenced by the vertices (or agents located on vertices) outside its set. The concept of robustness plays a role in the study of diffusion of information in networks [15, 38]. With the above two definitions, we are now ready to discuss our problem. The goal in the *robustness problem* is to find the largest value of $r$ such that the given graph is $r$-robust.

Note that although robustness and connectivity seem closely related to each other, the concept of robustness is much stronger than connectivity. If the connectivity of a graph is $r$, then for any two disjoint and nonempty subsets of its vertices, the nodes in each of these sets are connected to at least $r$ distinct vertices outside their sets, in total. In contrast, a graph is $r$-robust if there is vertex in one of such sets that is connected to at least $r$ vertices by itself. Figure 1.6 shows this difference more clearly. In this document, we characterize the complexity of determining the largest $r$ for which a given graph is $r$-robust.

## 1.3 Notations and Preliminaries

This section discusses the material that will be needed in the following chapters, including complexity classes, reductions and approximation algorithms.

### 1.3.1 Complexity Classes and Reducibility

Before introducing the complexity classes that are used later in this document, note that more detailed discussion on complexity classes can be found in [3]. Let us discuss what we mean by *decision problems*. Loosely speaking, a decision problem is a question with a 'Yes' or 'No' answer about the characteristics of a certain input. An example of this type of problem is the question whether a given a graph $G$ has a clique with $k$ vertices.[3]

Many mathematical problems are of another type called *optimization problems*, in which the goal is to find the best feasible solution. For most optimization problems it is possible to define an associated decision problem, called the *decision version* of that optimization problem. For example, the clique decision problem discussed above is the decision version of the clique problem, in which the goal is to find the maximum size clique in a graph. Note that usually if one can solve the decision version of a problem, then she can also solve the original problem by using binary search or other techniques.

A decision problem is *nondeterministically polynomial* or in **NP**, if and only if an input instance $x$ can be verified that it is a 'Yes' instance of the problem in polynomial time. To verify a problem is in **NP**, for any 'Yes' instance $x$ a *certificate* with polynomial size is needed, which would be the solution of the problem to instance $x$. For instance, consider the clique decision problem. The certificate for this problem is a set of $k$ vertices of $G$ that form a clique. Note that this is checkable in polynomial time by first checking the number of given vertices and also verifying that these vertices are all connected to each other.

The group of problems that are solvable in polynomial time, such as finding the shortest path between two vertices in a graph, are said to be in class **P**. For problems in this class, since the problems can be solved efficiently (i.e., in polynomial time) both 'Yes' and 'No' instances are verifiable in polynomial time. Hence, class **P** is a subset of class **NP**, i.e., $\mathbf{P} \subseteq \mathbf{NP}$. However, it is yet an open problem whether or not $\mathbf{P} = \mathbf{NP}$.

In addition to these two complexity classes, we will later refer to a complexity class called **coNP**. A problem is in class **coNP** if and only if an input instance $y$ can be verified that is a 'No' instance of the problem in polynomial time. Similar to class **NP**, to verify a

[3] This problem is known as the "clique decision problem" in the literature.

problem is in **coNP**, a polynomial size certificate should be provided for any 'No' instance $y$. An example of a problem in **coNP** is the problem of finding whether or not a given number $n$ is prime. If $n$ is not prime, then one can verify it by giving a factor of $n$ not equal to 1 or $n$.

For a given decision problem, its *complement* is obtained by reversing the 'Yes' and 'No' answers to the input instances. It can be shown that a problem is in class **coNP** if and only if its complement is in **NP** [3]. However, note that class **coNP** is not the complement of the class **NP**, as the intersection of these two is nonempty and is equal to class **P**. This is due to the fact that a problem is in **P** if it is solvable in polynomial time; therefore, both its 'Yes' and 'No' instances are verifiable in polynomial time.

Another tool that is used in this document is the notion of a *reduction*. Consider two problems $L_1$ and $L_2$ that are in **NP**. Also, let $U_1$ and $U_2$ denote the set of all possible inputs to $L_1$ and $L_2$, respectively. There is a *polynomial time reduction* from problem $L_1$ to $L_2$ (i.e., problem $L_1$ is *polynomially reducible* to $L_2$), if one can find a polynomial time algorithm that can transform any input of $L_1$, i.e., $u_1 \in U_1$, to another input of $L_2$, i.e., $u_2 \in U_2$, such that $u_1$ is a 'Yes' instance of $L_1$ if and only if $u_2$ is a 'Yes' instance of $L_2$. In other words, if $L_1$ is reducible to $L_2$, then the inputs of $L_1$ can be converted to inputs of $L_2$ in polynomial time so that if some algorithm can solve $L_2$ efficiently, then it can efficiently solve $L_1$ as well.

A problem $B$ is ***NP**-hard* if any problem $A$ in **NP** can be reduced to it in polynomial time, i.e., it is at least as hard as any problem in **NP**. Moreover, problem $B$ is said to be ***NP**-complete* if it is **NP**-hard and it is also in **NP**. Hence, **NP**-complete problems are the hardest problems in class **NP**. Note that in order to show that problem $B$ is **NP**-hard it is not necessary to reduce all problems in **NP** to $B$. Since **NP**-complete problems are the most difficult problems in **NP**, to show that $B$ is **NP**-hard it suffices to find an **NP**-complete problem $A$ that is reducible to $B$ in polynomial time. Similarly, a problem $B'$ is ***coNP**-hard* if any problem in **coNP** is polynomially reducible to it. Moreover, problem $B'$ is ***coNP**-complete* if it is **coNP**-hard and it is a member of class **coNP**. It can be shown that a problem is **NP**-hard if and only if its complement problem is **coNP**-hard. Therefore, another way of showing that a problem in **coNP** is **coNP**-complete is to show that its complement problem, which is in **NP**, is **NP**-hard.

### 1.3.2 Approximation Algorithms

Although there is no approximation algorithm discussed in this thesis, the reader should be acquainted to this concept for some sections of this work. Some very good references

on this topics are [35] and [36] that discuss different aspects of this field.

Consider an optimization problem $\Gamma$. This problem is defined by a set of *constraints* and an *objective function*. A solution to $\Gamma$ is called a *feasible solution* if it does not violate any of its constraints. Also, it is of interest to find a feasible solution to the problem that maximizes or minimizes (whichever the problem specifies) the objective function, in case such solution exists. To clarify the point, consider the problem of finding a clique that has the maximum number of vertices in a graph $G$. A solution to this problem should be a subset of vertices of $G$ and it is a feasible solution if these vertices constitute a clique. Moreover, the objective function is the number of vertices in the clique provided as a solution. The optimal solution is a clique with maximum number of vertices among the other cliques in the graph. Problem $\Gamma$ is called a *maximization problem* if its goal is to maximize the objective function; otherwise, if the goal is to minimize an objective function, it is called a *minimization problem*. For simplicity, in this section we only consider minimization problems and maximization problems can be analyzed similarly.

For a minimization problem $\Gamma$, we say a polynomial time algorithm ALG is an $\alpha$-*approximation algorithm* if the solution it finds for any input instance $\phi$, i.e., ALG($\phi$), is (i) a feasible solution to $\Gamma$, and (ii) the value of the objective function of $\Gamma$ for $\phi$ is at most $\alpha$ times its optimum value for instance $\phi$. The optimum value of the objective function for an instance $\phi$ is usually denoted by $\text{OPT}_\phi$ or OPT for brevity. If $\Gamma$ is a maximization problem, then an $\alpha$-approximation algorithm would provide feasible solutions such that the value of the objective function of $\Gamma$ for them is at least $\frac{\text{OPT}}{\alpha}$. Moreover, for both maximization and minimization problems, $\alpha$ is referred to as *approximation factor*, *approximation ratio* or *guarantee of approximation*.

Note that problems in class **P** are efficiently solvable and hence there is no point in approximating them. However, for problems that are **NP**-complete or **coNP**-complete, it is usually of interest to find approximation algorithms with polynomial runtime in the size of the input. However, in certain instances it can be shown via a reduction that for some $\alpha$, if an algorithm can approximate a problem $\Gamma$ within a factor of $\alpha$ in polynomial time, then it can also solve an **NP**-complete problem (or similarly, a **coNP**-complete problem) in polynomial time. In this case, it can be concluded that there is no polynomial time approximation algorithm for $\Gamma$ with approximation guarantee less than or equal to $\alpha$, unless $\mathbf{P} = \mathbf{NP}$ (or similarly, $\mathbf{P} = \mathbf{coNP}$).

## 1.4 Previous Work

In this section, a summary of the relevant previous work of each of the two main problems discussed in this thesis, i.e., the dominating set and robustness problems, are provided.

### 1.4.1 Dominating Set Problem

Significant attention has been devoted in recent years to the study of large-scale sensor and robotic networks due to their promise in fields such as environmental monitoring [24], inventory warehousing [18], and reconnaissance [5]. One of the key objectives in such networks is to ensure *coverage* of a given area, where every point in the space is within the sensing radius of one or more of the agents (i.e., sensors or robots). Various algorithms have been proposed to achieve coverage based on differing assumptions on the mobility and sensing capabilities of the agents [9].

In certain scenarios, the environment may impose restrictions on the feasible locations and motion of the agents [22]. In such cases, it is natural to model the environment as a graph, where each node represents a feasible location for an agent, and edges between nodes indicate available paths for the agents to follow. The coverage capabilities of any given agent are then related to the shortest-path distance metric on the graph: an agent located on a node can cover all nodes within a certain distance of that node. The goal of selecting certain nodes in a graph so that all other nodes are within a specified distance of the selected nodes is classical in graph theory, and is known as the dominating set problem [16]. Versions of this problem appear in settings such as multi-agent security and pursuit [1], routing in communication networks [37], and sensor placement in power networks [14].

Finding the *domination number* (i.e., the size of a smallest dominating set) of arbitrary graphs is **NP**-hard [16]. In fact, Raz and Safra showed that achieving an approximation ratio better than $c \log n$ for the dominating set problem in general graphs is **NP**-hard, where $c > 0$ is some constant and $n$ is the number of vertices of the graph [32]. However, there are several algorithms known for the dominating set problem for which the ratio between the size of the resulting dominating set and graph domination number can closely reach the $c \log n$ bound. The simplest of these algorithms is a *greedy algorithm* that at each step adds one vertex to the dominating set. The vertices that are already in the dominating set are marked as 'black', the vertices that share edges with black vertices are marked as 'gray' and other vertices are 'white'. At each step, a white vertex that shares the maximum number of edges with other white vertices are added to the dominating set and the color

labels of all vertices are updated according to the aforementioned rules. Another widely used approximation algorithm for the problem uses a *linear programming relaxation*. Both greedy and linear programming approaches for the dominating set problem are known to have $(\ln n + 1)$-approximation ratios [21, 27], which are in $O(\log n)$.

Even though in general graphs one cannot obtain an approximation ratio in $o(\log n)$, in special types of graphs better approximation ratios are obtainable. One of the most important classes of graphs are *planar graphs*. A *planar graph* is a graph that can be drawn in a plane so that none of its edges intersect except at their ends [7]. The dominating set problem is still **NP**-hard for planar graphs; however, the domination number of this type of graphs can be approximated within a factor of $(1 + \epsilon)$ for an arbitrarily small $\epsilon > 0$ [4].

Grid graphs are a special class of graphs that have attracted attention due to their ability to model and discretize rectangular environments [25, 26]. Grids can be used in simplifying the underlying environment by representing a certain area of the environment with only one node in the grid [6]. Moreover, due to the special structure of grid graphs that do not leave any area of environment unrepresented while transferring the problem environment into a tractable domain, they can be used as environment models on which area coverage can be performed. Hence grids are used very commonly in the network coverage and delectability literature [10, 26]. For these reasons, we focus on studying the dominating set problem when the underlying graph is a grid.

As discussed above, it is **NP**-hard to find the domination number of general or even planar graphs. It can be easily observed that grid graphs lie in the class of planar graphs and hence their domination number can be obtained within a small ratio. However, due to the special structure of grids, their domination number can in fact be determined optimally. For $m \times n$ grid graphs, an upper bound of $\left\lfloor \frac{(m+2)(n+2)}{5} \right\rfloor - 4$ on the size of the optimal dominating set was shown in [11] for $8 \leq m \leq n$ using a constructive method. Various attempts have been made in recent years to find a tight lower bound on the size of the optimal dominating set. In [2], the authors used brute-force computational techniques to find optimal dominating sets in grids of size up to $n = m = 29$. The paper [17] showed finally that the lower bound on the domination number is equal to the upper bound for $16 \leq m \leq n$, thus characterizing the domination number in grids.

### 1.4.2 Robustness Problem

The robustness problem was introduced in [23] in the context of diffusion of information in networks. In this context, there is usually a graph or a complex network with some

properties/information on some nodes (or agents) of the graph that spreads throughout the network. Hence, it is of interest to find the topological properties that allow 'good' information to diffuse throughout the network, while restricting the propagation of 'incorrect' information. An instance of such scenario is a network of sensors in a city that aims to report the true value of some measurements (i.e., good information), while several sensors within it report false measurements, e.g., due to faults or attacks. Hence, one might want to know what topological properties limit the spread of misinformation in the network.

Classically, the *connectivity* of a network plays the key role in the study of information dissemination algorithms (see [20, 28] for more information). However, these algorithms typically assume that the agents (or nodes of the network) have complete information about the structure of the graph, which is not a realistic assumption in large scale networks [34]. A more pragmatic assumption is that each agent has only local information about the topology of the network. In [23], it was shown that if a network is sufficiently *robust*, then there are algorithms that limit the spread of misinformation without requiring detailed knowledge of the global network. In [38], it was shown that certain random graph models inherently possess a certain degree of robustness.

Recall that a graph $G = (V, E)$ is $r$-robust if and only if for any two disjoint and nonempty subsets of its vertices, at least one of them has a vertex with at least $r$ neighbors outside its set. The *robustness problem* can be defined as an optimization problem whose goal is to find the value of $r$ for which a given graph is $r$-robust. As mentioned above, this problem is important in studying propagation of information in networks.

Another problem that is closely related to the robustness problem is called the *"matching-cut problem"* (this connection is fully discussed in Chapter 3). In the matching-cut problem, the goal is to find whether the input graph has a cut that is also a matching, i.e., any vertex of the graph is connected to at most one edge in the cut-set. Although the robustness problem was crystalized in the context of complex networks and diffusion of information, the matching-cut problem was first introduced in the graph drawing literature [13, 30]. It can be seen that if a graph has a matching-cut, then it is not 2-robust. It is shown in [29] that the matching-cut problem is **NP**-complete. Also, Bonsma in [8] studied the complexity of the problem in some classes of graphs, and showed that the problem is **NP**-complete even in planar graphs. Despite the known hardness of the matching-cut problem for different types of graphs, to the best of our knowledge, this is the first result on the complexity of the robustness problem.

## 1.5 Our Contribution

A summary of the contributions to the two problems studied in this document is provided in this section. Chapters 2 and 3 will be devoted to discuss these results in more details.

### 1.5.1 Dominating Set Problem

In this thesis, we make three contributions to the study of dominating sets on grids and their application to multi-agent coverage. First, we provide a new and simpler proof for an existing constructive method to obtain *near-optimal* dominating sets (i.e., that require no more than 5 vertices over the optimal number) in grids by Chang [11].

Second, based on the aforementioned centralized algorithm, we design a distributed algorithm that places a set of agents on the vertices of an $m \times n$ grid such that they construct a dominating set for the grid, with a number of agents that is within a constant additive error from the optimal. The agents require only limited memory, sensing and communication abilities, and thus the solution is applicable to multi-robot coverage applications where the environment can be discretized as a grid. Third, we generalize the construction introduced by Chang to the $k$-distance dominating set problem, where a given vertex can cover all other vertices within a distance $k$ from it. We show that our distributed algorithm can also be generalized to work in the $k$-distance domination scenario.

### 1.5.2 Robustness Problem

In this document, we fill a hole in the literature by showing that the robustness problem is **coNP**-complete. En route to proving this result, we introduce some new problems and study their complexities as well. We first introduce the decision version of the robustness problem, called the *$r$-robustness problem*. Later, we relax the constraints in the $r$-robustness problem and define the *relaxed-$r$-robustness problem*, in which a graph $G = (V, E)$ is relaxed-$r$-robust if and only if for any nonempty subset of its vertices $S \subsetneq V$ there is vertex in $S$ or $V \backslash S$ that has at least $r$ neighbors outside its containing set. To show that the relaxed-$r$-robustness in **coNP**-complete, we use that fact that the problem is in **coNP** and show that its complement problem, called the *relaxed-$\rho$-degree cut problem* which is a generalization of a known problem called the "matching-cut" problem, is **NP**-hard. We later use our construction for proving the complexity of the relaxed-$\rho$-degree cut problem to demonstrate that its non-relaxed version, called the *$\rho$-degree cut problem*, is also **NP**-hard. Later, utilizing the fact that the $\rho$-degree cut problem is the complement

of the $r$-robustness problem, we show that the latter problem is **coNP**-complete, which results in the **coNP**-completeness of the robustness problem. Moreover, we show that it is not possible to provide a polynomial time approximation algorithm with guarantee less than 2 for the problem of finding the smallest $\rho$ such that a given graph has a $\rho$-degree cut, unless $\mathbf{P} = \mathbf{NP}$.

## 1.6 Thesis Organization

In Chapter 2, we discuss the constructive centralized grid domination algorithm introduced by Chang and introduce a new and simpler proof for that construction. These materials are later used to design a distributed algorithm for the dominating set problem. We then generalize the results on both the centralized and distributed algorithms to the $k$-distance dominating set problem.

In Chapter 3, we discuss the decision version of the robustness problem and introduce its relaxed form. We then show that these problems are in **coNP** and their complements are **NP**-hard and hence conclude that they are **coNP**-complete. Later, using these results we infer that the robustness problem is also **coNP**-complete. We then briefly discuss the hardness of approximation of a problem closely related to the robustness problem.

Finally, Chapter 4 concludes the thesis and discusses the corresponding open problems.

# Chapter 2

# Dominating Set Problem on Grids

Recall that in a graph $G = (V, E)$, for a set of vertices $U \subseteq V$, we say the vertices in $N(U)$, i.e., the set of vertices that have neighbors in $U$, are *dominated* by the vertices in $U$. For graph $G$, a set of vertices $S \subseteq V$ is a *dominating set* if each vertex $v \in V$ is either in $S$ or is dominated by $S$.

A dominating set with minimum cardinality is called an *optimal dominating set* of a graph $G$; its cardinality is called the *domination number* of $G$ and is denoted by $\gamma(G)$. Note that although the domination number of a graph, $\gamma(G)$, is unique, there may be different optimal dominating sets.

As discussed in the previous chapter, we study the dominating set problem on a special class of graphs called *grid graphs*. Recall that an $m \times n$ grid graph $G = (V, E)$ is defined as a graph with vertex set $V = \{v_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n\}$ and edge set $E = \{v_{i,j}v_{i,j'} | \, |j - j'| = 1\} \cup \{v_{i,j}v_{i',j} | \, |i - i'| = 1\}$. Also remember that for ease of exposition, we fix an orientation and labelling of the vertices, so that vertex $v_{1,1}$ is the lower-left vertex and vertex $v_{m,n}$ is the upper-right vertex of the grid. We denote the domination number of an $m \times n$ grid $G$ by $\gamma_{m,n} = \gamma(G)$.

Moreover, as introduced in Chapter 1, for an $m \times n$ grid $G = (V, E)$, we define the *boundary* of $G$, denoted by $B(G)$, as the set of vertices with less than 4 neighbors. Also, an $m \times n$ grid $G = (V, E)$ is called a *sub-grid* of an $m' \times n'$ grid $G' = (V', E')$ if $G$ is induced by vertices $v'_{i,j} \in V'$, where $2 \leq i \leq m' - 1$ and $2 \leq j \leq n' - 1$. If $G$ is a sub-grid of $G'$, then $G'$ is called the *super-grid* of $G$ (see Figure 2.1(a)).

The domination number of grid graphs is computable. The following theorem gives the domination number of grids with size greater than 16..

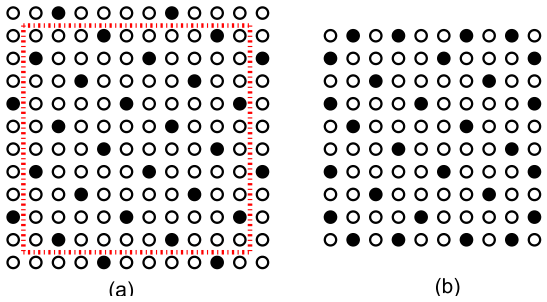



Figure 2.1: In Figure (a), a $12 \times 12$ grid $G'$ is demonstrated and its $10 \times 10$ sub-grid $G$ is highlighted by a red dashed square. Grid $G'$ is diagonalized by a set $U'$ of 28 vertices. In Figure (b), vertices in $U' \backslash V$ are projected onto their neighbors in $G$.

**Theorem 2.1 (Gonçalves *etal*, [17])** *For an $m \times n$ grid with $16 \leq m \leq n$, $\gamma_{m,n} = \left\lfloor \frac{(m+2)(n+2)}{5} \right\rfloor - 4$.*

In this chapter, we provide a distributed algorithm that locates a set of agents on the vertices of a grid such that they constitute a dominating set with near optimal size for it. Our distributed grid domination algorithm is based on a procedure developed by Chang [11]. To obtain the tools needed in our distributed algorithm, we discuss an overview of the construction introduced by Chang in Section 2.1. These tools are used in Sections 2.2 and 2.3 in the distributed domination algorithm and in developing the $k$-distance dominating set results.

## 2.1 Overview of Centralized Grid Domination Algorithm

In [2], Alanko *et al.* provided examples of optimal dominating sets for $n \times n$ grids with $1 \leq n \leq 29$, obtained via a brute-force computational method. A visual inspection of these examples shows that as the size of the grid increases, the patterns of dominating vertices become more regular in the interior of grids, with irregularities at the boundaries. Figure 2.2 demonstrates some examples of patterns that arise in the dominated grids in [2].

Among the patterns used to dominate grids, the one illustrated in Figure 2.2(b) is the most efficient, since there is no vertex that is dominated by more than one dominating

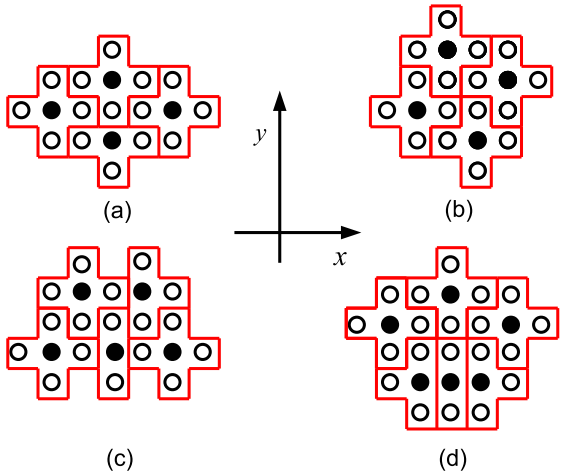



Figure 2.2: Examples of dominating vertex patterns that appear in the instances of optimal dominating sets of Alanko *et al.* [2]. The black vertices are the dominating vertices. The red line segments form regions so that in each region there exist one black vertex and at most four white vertices dominated by that black vertex.

vertex in this pattern. Hence, this pattern would be useful in obtaining dominating sets with near optimal size. We refer to the structure in Figure 2.2(b) as a *diagonal pattern*. In [11], Chang used these patterns to provide a constructive upper-bound on the domination number of grids. In this section, we provide an overview of the construction introduced by Chang, as it will be used in the subsequent sections. First, we define the diagonal patterns formally as follows. Note that the $x$ and $y$ axes are as shown in Figure 2.2.

**Definition 2.1** *(Diagonal Pattern) A set of vertices $U \subset V$ constitutes a* diagonal pattern *on grid $G = (V, E)$ if there exists a fixed $r \in \{0, 1, 2, 3, 4\}$ such that for any vertex $v_{x,y} \in U$ we have $y - 2x \equiv r \pmod 5$.*

**Definition 2.2** *(Diagonalization) A set of vertices $U \subset V$* diagonalizes *grid $G = (V, E)$ if it constitutes a diagonal pattern and there exists no vertex $v \in V \backslash U$ that can be added to $U$ so that $U$ remains a diagonal pattern.*

An example of a diagonalization is shown in Figure 2.1(a).[1] The construction that Chang introduced consists of the following two main steps:

[1]One can also define a diagonal pattern as a set of vertices whose $(x, y)$ coordinates satisfy $x - 2y \equiv r \pmod 5$, for some fixed $r$. This corresponds to swapping the $x$ and $y$ axes. For the proofs we only analyze the case mentioned in Definition 2.1; the other case can be treated similarly.

1. Diagonalization: At this step, a set of vertices $U$ that diagonalizes the grid is provided.

2. Projection: Using a process called *projection*, the vertices that were not dominated by vertices in $U$ are characterized and new vertices are added to $U$ to dominate those vertices as well.

We now discuss these two steps in more details. Chang showed that if a grid $G = (V, E)$ is diagonalized by a set of vertices $U \subset V$, then for any vertex $v \in (V \backslash U)$ that is not located on the grid's boundary, there exists exactly one vertex in $U$ that shares an edge with $v$. In other words, every node that is not located on the grid's boundary, i.e., $B(G)$, is dominated by exactly one vertex in $U$. Moreover, he proved that if a set of vertices $U \subset V$ diagonalizes an $m \times n$ grid $G = (V, E)$, then $U$ contains at most $\lceil \frac{mn}{5} \rceil$ vertices. To construct a dominating set for $G$, it only remains to add some vertices to $U$ so that the resulting set dominates the vertices on the boundary of the grid as well. The vertices located on $B(G)$ with no neighbor in $U$ are called *orphans* and are defined formally as follows.

**Definition 2.3** *(Orphans) Let $U \subset V$ be a set of vertices that diagonalizes grid $G = (V, E)$. A vertex $v \in V$ that has no neighbor in $U$ is called an* orphan *(see Figure 2.1(a)).*

To dominate orphans, Chang used the super-grid of $G$, denoted by $G' = (V', E')$. Since the vertices on the boundary of $G$ lie inside grid $G'$, a set of vertices $U' \subset V'$ that diagonalizes $G'$ dominates all vertices of $G$. Moreover, it can be easily seen that the set of vertices $U = U' \cap V$ is also a diagonalization for grid $G$.

Recall that diagonalization results in every vertex being dominated by at most one vertex in the diagonal pattern. Therefore, if a set of vertices $U' \subset V'$ diagonalizes the super-grid $G' = (V', E')$, then there are vertices in $B(G)$ that are dominated by vertices in $U' \backslash V$. Hence, the *orphan* of a vertex $v \in U' \backslash V$ is a vertex $u \in B(G)$ such that $u \in N(v)$, and is denoted by $u = \text{orphan}(v)$.

**Corollary 2.1** *For an $m \times n$ grid $G$, the number of orphans is $O(n + m)$.*

Since by diagonalizing $G'$ the orphans in $G$, i.e, vertices in $N(U' \backslash U) \cap V$, are dominated by the dominating vertices on the boundary of $G'$, a procedure called *projection* is introduced that projects the dominating vertices in $B(G')$ inside sub-grid $G$. Hence, projection results in having all vertices in $G$ being dominated. This procedure is defined formally as follows.

**Definition 2.4** *(Projection) Consider a grid $G = (V, E)$ and its super-grid $G' = (V', E')$. For a set $U' \subseteq V'$, its* projection *is defined as the set $U'' = \big(N(U' \backslash V) \cup U'\big) \cap V$. Similarly, we say a vertex $v \in U' \backslash V$ is* projected *if it is mapped to its neighbor in $V$.*

Figure 2.1(b) shows an example of a projection. For grid $G = (V, E)$, its super-grid $G' = (V', E')$, and set $U' \subset V'$ that diagonalizes $G'$, by performing projection, the size of the obtained dominating set of $G$ is between $|U'| - 4$ and $|U'|$. This is due to the fact that a vertex $v \in U'$ located at any corner of $G'$ has no neighbor in $V$ and hence, after projection it is not mapped into $V$. Since $G'$ has four corners, for the result of projection of $U'$, i.e., $U''$, we have $|U'| - 4 \leq |U''| \leq |U'|$. Hence, the number of dominating vertices used in diagonalizing the super-grid of $G$, $|U'|$, is an upper-bound on the number of dominating vertices used to fully dominate $G$ by diagonalization and projection. Since the size of super-grid of an $m \times n$ grid $G$ is $(m+2) \times (n+2)$, $|U'| \leq \left\lceil \frac{(m+2)(n+2)}{5} \right\rceil$. Hence, $\left\lceil \frac{(m+2)(n+2)}{5} \right\rceil$ is an upper-bound on the number of dominating vertices used to dominate grid $G$ by this construction. The following theorem reflects this upper-bound.

**Theorem 2.2 (Chang [11])** *For any $m \times n$ grid $G = (V, E)$ with $m, n \in \mathbb{N}$, a dominating set $S \subset V$ can be constructed in polynomial-time, such that $|S| \leq \left\lceil \frac{(m+2)(n+2)}{5} \right\rceil$. Moreover, for grids with $16 \leq m \leq n$ we have $|S| - \gamma_{m,n} \leq 5$.*

For an $m \times n$ grid $G$ with $16 \leq m \leq n$, the difference between the cardinality of the provided dominating set $S$ from the domination number, $\gamma_{m,n}$, is upper-bounded by virtue of Theorem 2.1. An example of constructing dominating sets for grids using diagonalization and projection is shown in Figure 2.1.

In the following we provide a new and simpler proof for Theorem 2.2 by showing that if a subset of vertices of an $m \times n$ grid diagonalizes it, then the size of that set is at most $\left\lceil \frac{mn}{5} \right\rceil$.

**Lemma 2.1** *If a set $U \subset V$ diagonalizes an $m \times n$ grid $G = (V, E)$, where $m, n \in \mathbb{Z}_{\geq 1}$, then $|U| \leq \left\lceil \frac{mn}{5} \right\rceil$.*

**Proof.** We first show that the difference between the size of any two sets that diagonalize an $a \times b$ grid, where $0 < a, b < 5$, is at most one. We then use this to show that the size of a set that diagonalizes an $m \times n$ grid, where $m, n \geq 1$, is at most $\left\lceil \frac{mn}{5} \right\rceil$.

Since set $U$ diagonalizes $G$, it constitutes a diagonal pattern on $G$ such that no more vertices can be added to it while maintaining a diagonal pattern. Therefore, among each

five consecutive vertices in any row or column there is exactly one vertex from $U$. Hence, the number of vertices of $U$ in a row/column of $t$ vertices is at most $\lceil \frac{t}{5} \rceil$.

Thus, there are at most five vertices from $U$ in any $5 \times 5$ grid. Hence, in any $(5q) \times (5p)$ grid with $p, q \in \mathbb{Z}_{\geq 1}$, there are at most $5pq$ vertices from $U$. For an $m \times n$ grid $G = (V, E)$ with $m = 5q + a$, $n = 5p + b$ and $0 \leq a, b < 5$, we partition $V$ into the following four sets:

$$V_1 = \{v_{i,j} \mid 1 \leq i \leq 5q, 1 \leq j \leq 5p\},$$

$$V_2 = \{v_{i,j} \mid 5q + 1 \leq i \leq m, 1 \leq j \leq 5p\},$$

$$V_3 = \{v_{i,j} \mid 1 \leq i \leq 5q, 5p + 1 \leq j \leq n\},$$

and

$$V_4 = \{v_{i,j} \mid 5q + 1 \leq i \leq m, 5p + 1 \leq j \leq n\}.$$

As stated, $|V_1 \cap U| \leq 5pq$. Grid $V_2$ has $a$ columns each having $5p$ vertices, hence $|V_2 \cap U| \leq pa$. Similarly, we have $|V_3 \cap U| \leq qb$. In summary, we so far have $|(V_1 \cup V_2 \cup V_3) \cap U| \leq 5pq + qb + pa$. Note that $5pq + qb + pa = \frac{mn}{5} - \frac{ab}{5}$.

It remains to upper-bound $|V_4 \cap U|$. Note that $V_4 \cap U$ also diagonalizes the grid induced by the set of vertices $V_4$, which we denote by $G_4 = (V_4, E_4)$. Also, recall that for any vertex $v_{x,y}$ in a set $U$ that diagonalizes a grid $G$, we have $y - 2x \equiv r \pmod 5$ for a fixed $r \in \{0, 1, 2, 3, 4\}$. Hence, there are five different subsets of vertices of $V_4$ that diagonalize grid $G_4$, denoted by $U_0, U_1, U_2, U_3$ and $U_4$. Moreover, for a vertex $v_{x,y} \in V_4$, we define its *$r$-value* as the value of $y - 2x$ modulo 5. Therefore, all vertices in $U_i$, where $i \in \{0, 1, 2, 3, 4\}$, have the same $r$-values. Without loss of generality, assume that among these five sets, $U_0$ and $U_1$ have the maximum and minimum sizes, respectively.

Any vertex that is not on the boundary of a grid has exactly four neighbors, one at each side, that is, left, above, right and bottom. Consider vertex $v_{x,y}$ that is not on the grid boundary, and its four neighbors $v_{x-1,y}, v_{x,y+1}, v_{x+1,y}$ and $v_{x,y-1}$. It can be seen that these five vertices have different $r$-values. Hence, each of them belong to a distinct set of $U_0, \ldots, U_4$. Also, if vertices $u, v$ are on the same side (say left) of vertices $u', v' \in U_i$ for some $i \in \{0, 1, 2, 3, 4\}$, then the $r$-values of $u$ and $v$ are the same. Thus, the set of vertices that are on the same side (i.e., left, above, right, or below) of the vertices in $U_0$ have the same $r$-values.

Let $B(G_4)$ denote the boundary of $G_4$. If there exists no vertex in $B(G_4)$ that is in $U_0$, i.e., $U_0 \cap B(G_4) = \emptyset$, then we argue that $|U_0| = |U_1|$. The reason is that since no vertex of $U_0$ is located on the boundary, every vertex in $U_0$ has exactly four neighbors, each in a different set $U_1, \ldots, U_4$. Hence, for every vertex in $U_0$ (i.e., a set that diagonalizes $G_4$ and

has the maximum size), there is a vertex in $U_1$ (i.e., a set with that diagonalizes $G_4$ and has the minimum size). Therefore, it can be concluded that $|U_0| = |U_1|$.

Note that since $0 \leq a, b < 5$, at most one vertex in each row or column of $G_4$ can be in $U_0$. Also, for any vertex $u \in U_0$ that lies on the boundary of $G_4$, one of the two following cases happens: (i) vertex $u$ is not on any corners of the grid and hence misses at most one of its neighbors, or (ii) it is located on a corner of the grid and misses two of its neighbors. If case (i) holds for all the vertices of $U_0$ that are on the boundary, then vertices in $U_0$ miss at most four of their neighbors and each of the missing neighbors is on a different side of the vertices in $U_0$. If case (ii) happens for a vertex $u \in U_0$ that lies on the boundary, then $u$ misses two of its neighbors. However, since it is located on both a row and a column, it does not allow another vertex in $U_0$ to be placed on that row or column. Therefore, similar to the previous case, vertices in $U_0$ have at most four missing neighbors and these missing neighbors are on different sides of the vertices in $U_0$. Hence they have different $r$-values. Thus, each of them belongs to a different set $U_1, \ldots, U_4$. Therefore, the size of $U_1$ is at most one less than the size of $U_0$.

Now, note that sets $U_0, \ldots, U_4$ partition the set $V_4$; therefore, $U_0 \cup U_1 \cup U_2 \cup U_3 \cup U_4 = V_4$. Moreover, we showed that the difference between the size of $U_0$ and any other set among $U_1 \ldots U_4$ is at most one. Hence, we have $5|U_0| - 4 \leq |V_4|$, resulting in $|U_0| \leq \frac{|V_4|+4}{5}$. Recall that $G_4$ is an $a \times b$ grid and thus $V_4 = ab$. Therefore it can be concluded that $|U_0| \leq \frac{ab+4}{5}$ and hence for any set $U'$ that diagonalizes grid $G_4$ we have $|U'| \leq \frac{ab+4}{5}$. In particular, $|U \cap V_4| \leq \frac{ab+4}{5}$; therefore, $|(V_1 \cup V_2 \cup V_3 \cup V_4) \cap U| \leq \frac{mn}{5} - \frac{ab}{5} + \frac{ab+4}{5}$. In other words, $|U| \leq \lceil \frac{mn}{5} \rceil$. □

Consider the greedy algorithm discussed in Section 1.4.1. Since at each step of this algorithm, an undominated vertex that has the maximum number of undominated neighbors is added to the dominating set, it might result in the same solution as projection and diagonalization. In the following lemma, we show that using a simple greedy algorithm does not necessarily result in diagonalizing the grid or using at most $\left\lceil \frac{(m+2)(n+2)}{5} \right\rceil$ dominating vertices to dominate the grid.

**Lemma 2.2** *In the worst case, the size of the dominating set obtained by a greedy algorithm on an $m \times n$ grid $G$ is lower-bounded by $\lceil \frac{m}{3} \rceil \lceil \frac{n}{3} \rceil + 2 \lfloor \frac{m}{3} \rfloor \lfloor \frac{n}{3} \rfloor$.*

**Proof.** As discussed in Section 1.4.1, after the first vertex $v$ is added to the dominating set $S$, the greedy algorithm chooses a vertex that does not share any neighbors with $v$, if one exists. Although this is also a property of the diagonal patterns, we argue that they

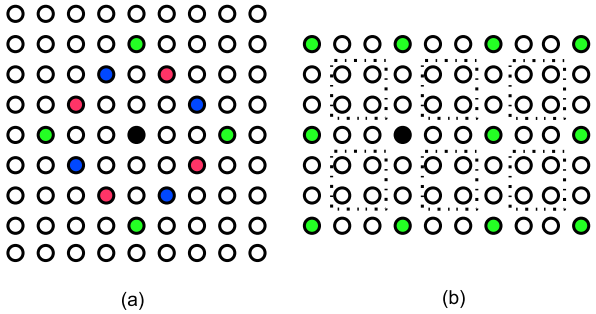



Figure 2.3: In Figure (a), after adding the black vertex to the dominating set, the next vertex that a greedy algorithm adds to that set without dominating any vertex by two dominating vertices can be any of the blue, red or green vertices. In Figure (b), a dominating set is built for another grid, by starting from the black vertex and keep adding the green vertices shown in Figure (a) to the set. Each dotted rectangle contains four vertices that are not dominating by the obtained dominating set.

will not have the same result. The set of all the vertices around $v$ that can be added to $S$ using diagonal patterns has size at most four (see Figure 2.2(b)). However, there are 12 vertices around $v$ that do not share any neighbors with $v$ and hence are candidates to be added to $S$ in a greedy algorithm (Figure 2.3(a)). At each step of a greedy algorithm, one of these 12 vertices is chosen arbitrarily. Choosing only all red vertices or all blue vertices would start developing a diagonal pattern. Other combinations of candidate vertices would fail to diagonalize the grid and in the end, some vertices of the graph would be dominated by more than one dominating vertex. Hence, the size of the constructed dominating set would be greater than $\left\lceil \frac{(m+2)(n+2)}{5} \right\rceil$.

In particular, the algorithm might add all the green vertices to $S$ and repeat the same pattern in the grid (Figure 2.3(b)). However, using this pattern, between any four green vertices there remains a set of four vertices that are not dominated by any vertex in $S$. These vertices are highlighted by dotted rectangles in Figure 2.3(b). To dominate each of these sets of vertices, at least two extra dominating vertices should be added to $S$. Therefore, the number of obtained dominating vertices would be at least $\left\lceil \frac{m}{3} \right\rceil \left\lceil \frac{n}{3} \right\rceil + 2 \left\lfloor \frac{m}{3} \right\rfloor \left\lfloor \frac{n}{3} \right\rfloor$, which is much greater than the size of the dominating set obtained by diagonalization and projection, i.e., $\left\lceil \frac{(m+2)(n+2)}{5} \right\rceil$. □

## 2.2 Distributed Grid Domination

In the preceding section, a centralized algorithm was discussed that produced a dominating set $S$ for a given $m \times n$ grid $G$ such that $|S| \leq \left\lceil \frac{(m+2)(n+2)}{5} \right\rceil$. In this section, we show how to achieve the same upper-bound in a distributed way.

### 2.2.1 Model and Notation

Here we assume that the environment is an $m \times n$ grid $G = (V, E)$ with $m, n \in \mathbb{N}$. The goal is to dominate the grid environment in a distributed fashion using several robots (or agents) without any knowledge of environment size. Initially, there exist $k$ agents in the environment, where $k$ can be smaller or greater than the number of agents needed to dominate the grid. The following assumptions are made for the grid and agents.

**Grid Assumptions:** Agents can be located only on the vertices of the grid and are able to move between the grid vertices only on the edges of the grid. At each moment, a vertex can contain more than one agent. We refer to the vertices using the standard Cartesian coordinates defined in Section 1.1.

**Agent Assumptions:** The agents, denoted by $a_1, \ldots, a_k$, are initially located at arbitrary vertices on the grid. The agents have three modes: (a) *sleep*, (b) *active*, and (c) *settled*. The mode of an agent $a$ and the vertex it is located at are denoted by mode($a$) and $v(a)$, respectively. Only agents in the active and settled modes are able to communicate. At the beginning of the procedure, all the agents are in the sleep mode and throughout the procedure it is assumed that they activate asynchronously. The activation sequence of agents is arbitrary (e.g., it can be scheduled in advance or it can be random). During each *epoch*, that is, a time interval with a specified length, one newly activated agent moves along the edges of the grid to find a suitable vertex and settles on it. The active agent can communicate with the settled agents to perform the distributed dominating set algorithm. Once an agent activates and performs its part in the algorithm, it goes to settled mode. Ultimately, all the settled agents go back to sleep mode and will not activate again.

Here, each agent is equipped with suitable angle-of-arrival (bearing) and range sensors. Using these sensors, agent $a$ computes the coordinates of other agents in its own coordinate frame $\Sigma_a$ with its origin at $v(a)$ and an arbitrary orientation, fixed relative to agent $a$. Each agent also has a compass to determine its heading direction. Additionally, agents are equipped with short-ranged proximity sensors to sense the environment boundary. Agents are able to sense the boundary only if they are on a vertex $v$ whose neighbor is a boundary

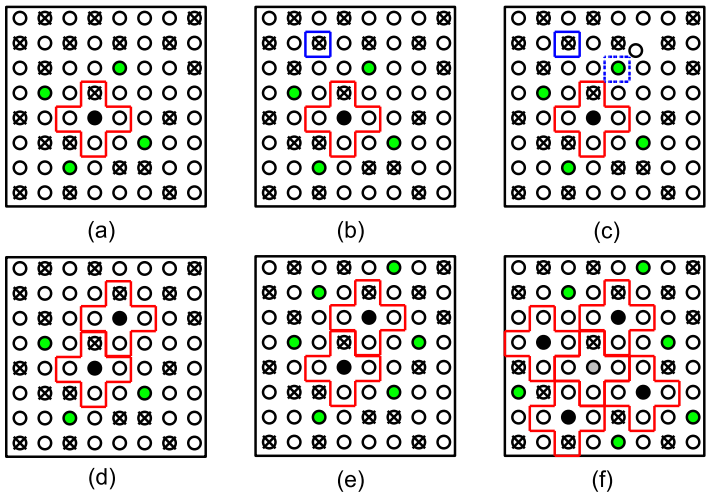



Figure 2.4: Non-activated agents are marked by black crosses and the already settled agents are shown by black circles. Agents in cluster $C$ have red crosses as their modules. Figure (a) shows the first active agent, as in Step 6 of Algorithm 1. In Figure (b), an active agent is highlighted by a blue square. Step 13 of the algorithm is depicted in Figure (c), where a dashed blue square shows the closest valid slot to the active agent. In Figure (d), the active agent moves to the valid slot and joins $C$, as in Step 15. In Figure (e), the list of valid slots is updated as in Step 24 of the algorithm. In Figure (f), the grey circle shows an agent that goes from settled to sleep mode, as in Step 26 of Algorithm 1.

vertex of the grid, i.e., $N(v) \cap B(G) \neq \emptyset$. The compass helps agents to distinguish which of the four boundary edges they are approaching.

### 2.2.2 Overview of Algorithm

The main idea in this algorithm is to implement the diagonal pattern defined in Section 2.1 on grid $G = (V, E)$, using communications among active and settled agents. A special unit called a *module* is defined for the active and settled agents. A module is a cross-like shape consisting of the agent at its center with the associated dominated vertices in the arms of the cross (see Figure 2.2(b)). For each module $m$, the vertex that contains the agent, i.e., the center vertex, is referred to as the *module center*, denoted by $c(m)$. As an agent moves on the grid to contribute to the diagonal pattern, its module moves with it as well. Modules $m_1$ and $m_2$ with module centers $c(m_1) = v_{i,j}$ and $c(m_2) = v_{i',j'}$ can connect to each other if $v_{i',j'} \in \{v_{i+1,j+2}, v_{i+2,j-1}, v_{i-1,j-2}, v_{i-2,j+1}\}$ (see Figure 2.4(f)). This condition is called the *module connection condition*. The set of centers of the connected modules is called a *cluster*. We will later show that the module connection condition ensures that the

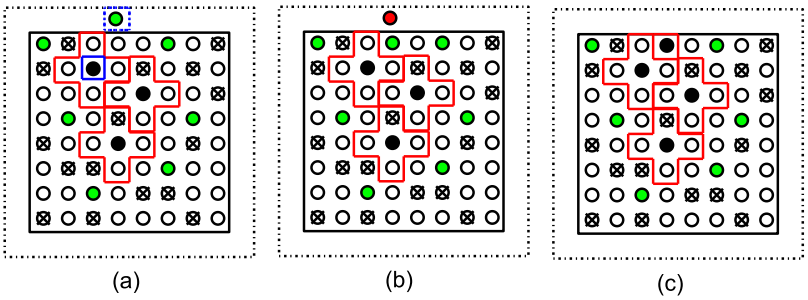



Figure 2.5: Non-activated agents are marked by black crosses and the already settled agents are shown by black circles, with red crosses as their modules. In Figure (a), a settled agent, highlighted by a solid blue square, realizes one of its slots, shown by a dashed blue square, is outside the grid boundary. In Figure (b), the settled agent replaces the slot outside the grid boundary with its orphan and name the resulting set as its valid slots. In Figure (c), the active agent locates at the orphan.

module centers are a diagonalization of the vertices covered by the modules in the cluster.

**Valid Slots:** Let $G' = (V', E')$ be the super-grid of $G$. A vertex $v_{a,b} \in V'$ is called a *slot* if $v_{a,b}$ is not already a center for a module in the cluster and there exists a module $m$ in the cluster with center $v_{i,j}$ such that $v_{a,b} \in \{v_{i+1,j+2}, v_{i+2,j-1}, v_{i-1,j-2}, v_{i-2,j+1}\}$. For a settled agent $a$ located at $v(a)$, denote the set of all its slots by slots$(a)$. Recall that the *orphan* of a vertex $v \in V' \backslash V$, i.e., orphan$(v)$, is a vertex $u \in B(G)$ such that $u \in N(v)$. The set of all *valid slots* for settled agent $a$, denoted by vslots$(a)$, is defined as (slots$(a) \cap V) \cup$ orphan(slots$(a) \backslash V$). Newly activated agents can settle only on the valid slots of the settled agents.

**Updating Valid Slots:** When an active agent settles, it creates the list of its valid slots as follows. If a settled agent $a$ cannot sense the boundary (i.e., it has no neighbor on the boundary), slots$(a) \backslash V = \emptyset$ and hence vslots$(a) =$ slots$(a)$. Conversely, a settled agent can also determine which of its slots lie outside the grid boundary (Figure 2.5(a)). Each newly settled agent marks the vertices on the grid boundary that are neighbors of slots$(a) \backslash V$ as *orphans* and so vslots$(a) =$ (slots$(a) \cap V) \cup$ orphan(slots$(a) \backslash V$) (Figure 2.5(b)). By the definition of valid slots, no valid slot exists in an orphan's neighborhood. Therefore, each orphan needs one agent to be located on itself or one of its neighbors to be dominated. For simplicity we always put an agent on the orphan itself.

When an agent activates, it transmits a signal to find the settled agents on the grid and waits for some specified time for a response from them. Since there is no settled agent in the environment when the first agent activates, it receives no signal and concludes it is the first one activated. Thus, the agent stays at its initial location and goes to the settled

mode. Subsequently, each active agent translates to the closest settled agent.[2]

### 2.2.3 Distributed Grid Domination Algorithm

During the distributed grid domination algorithm, active agents can either contribute to grid diagonalization by locating on non-orphan valid slots or can settle on orphans. In each epoch, the set of the non-orphan vertices containing the previously settled agents is called the *cluster* and is denoted by $C$, while the set of occupied orphans is denoted by $P$. At the beginning of the algorithm $C = P = \emptyset$. It should be mentioned that $C$ and $P$ are not saved by any agent, and are used only to aid in the presentation of the algorithm. Moreover, we denote the set of all settled agents at each moment by $A_s$, where at the beginning of the algorithm $A_s = \emptyset$. Also if agent $a$ is already settled and is now in sleep mode done$(a) = 1$, otherwise done$(a) = 0$. The distributed procedure for dominating grids is reflected in Algorithm 1.

**Remark 2.1** *(Comments on Algorithm 1)*
*1) Since agents can move only on the grid edges, the distance between two vertices can be computed simply by adding their $x$-coordinate and $y$-coordinate differences, i.e., $\Delta x$ and $\Delta y$. There exist many shortest paths between any two vertices of a grid and agent $a$ arbitrarily chooses one of them to traverse; for instance it can first traverse the $x$-coordinate and then the $y$-coordinate.*
*2) Note that the algorithm can be applied even if the agents have different coordinate frames. In Step 11 of the algorithm, agent $a$ locates $s$ in $\Sigma_a$ (i.e., coordinate frame of $a$), while* vslots$(s)$ *is computed by $s$ in $\Sigma_s$ in Step 12. In Step 13, agent $a$ converts the coordinates of* vslots$(s)$ *from $\Sigma_s$ to $\Sigma_a$ for traversing, using relative sensing techniques [31].*
*3) When an agent settles, all settled agents wait for a specified amount of time for the next agent to activate. If no agent activates, Algorithm 1 halts and the previously settled agents construct a subset of a dominating set of the grid. This happens when the initial number of agents is not sufficient to dominate the grid.*
*4) If the agents are equipped with GPS, then they can agree on a fixed diagonalization (i.e., agree on a value of $r$), and move to the vertices $U$ in that diagonalization. At this point, only orphan vertices exist. The remaining agents can move along the boundary to find and cover all orphans and consequently dominate the grid. Hence, in this thesis we study the case that agents are not armed with GPS.*

[2]Note that for completeness of the algorithm, it is not necessary for the active agents to go to the closest settled agents. An active agent can go toward any arbitrary settled agent to occupy its valid slot.

**Algorithm 1:** DISTRIBUTED GRID DOMINATION

**Input**: An $m \times n$ Grid and a set of agents $A$

1 **while** $\exists$ *agent* $a \in A$ *with* mode$(a)$ = *sleep and* done$(a) = 0$ **do**
2  mode$(a)$ := active, $a$ sends out signal to $A_s$ (Figure 2.4(b)).
3  **if** $A_s \neq \emptyset$ **then**
4  At least one agent in $A_s$ sends a signal out to $a$.
5  **if** $a$ *receives no signal* **then**
6  mode$(a)$ := settled (Figure 2.4(a)).
7  $A_s := \{a\}$.
8  $C := \{v(a)\}$.
9  Skip to Line 22.
10  **if** vslots$(A_s) \neq \emptyset$ **then**
11  Agent $a$ computes the closest settled agent $s \in A_s$ and notifies $A_s$.
12  Agent $s$ sends the coordinates of vslots$(s)$ to $a$.
13  Agent $a$ moves toward the closest $v \in$ vslots$(s)$.
14  **if** $v(a) = v$ **then**
15  mode$(a)$ := settled (Figure 2.4(d)).
16  $A_s := A_s \cup \{a\}$.
17  **if** $v(a)$ *and* $v(s)$ *satisfy the module connection condition* **then**
18  $C := C \cup \{v(a)\}$.
19  **else**
20  $P := P \cup \{v(a)\}$ (Figure 2.5(c)).
21  mode$(a)$ := sleep.
22  **for** $i = 1 \rightarrow |A_s|$ **do**
23  **if** $v(A_s(i)) \in C$ *and* mode$(A_s(i)) \neq$ sleep **then**
24  Update vslots$(A_s(i))$ (Figures 2.4(e)).
25  **if** vslots$(A_s(i)) = \emptyset$ **then**
26  mode$(A_s(i))$ := sleep (Figure 2.4(f)).
27  done$(A_s(i)) := 1$.
28  **else**
29  Break.
30 The remaining non-activated agents leave the grid.

### 2.2.4 Analysis of Distributed Grid Domination Algorithm

We now prove that the set of vertices determined by Algorithm 1, i.e., $C \cup P$, creates a dominating set for the grid. Recall that at each epoch, $C$ is the set of non-orphan vertices containing the previously settled agents and $P$ is the set of occupied orphans.

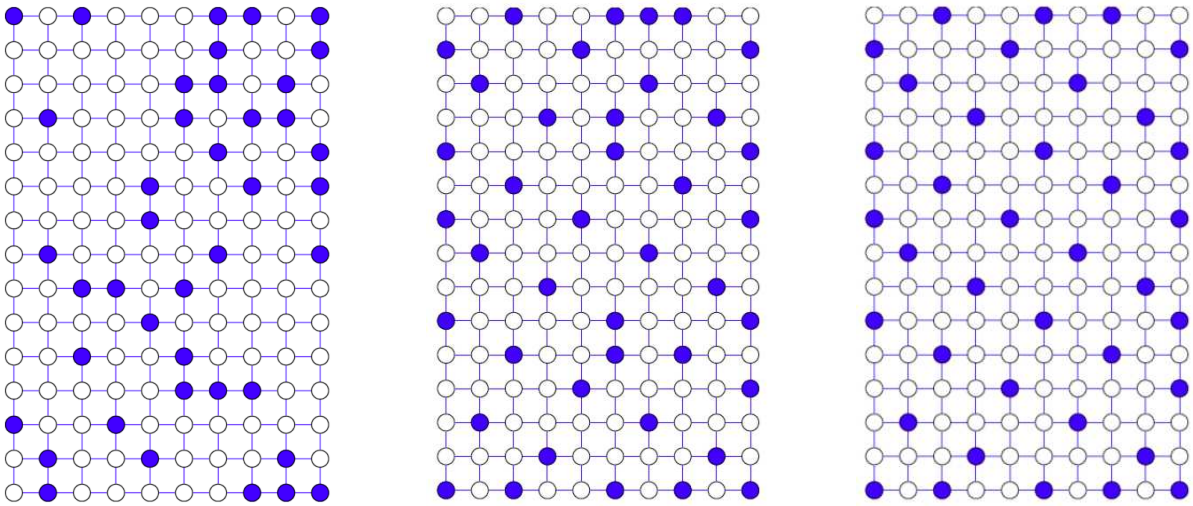
**Lemma 2.3** *During the operation of Algorithm 1, the module connection condition forces the vertices in $C$ to create a diagonal pattern.*

**Proof.** This will be proved using induction on the size of $C$ during the operation of the algorithm. According to the module connection condition, the module of agent $a$ located at vertex $v(a) = v_{i',j'} \notin C$ can connect to the module of vertex $v_{i,j} \in C$ if $v_{i',j'} \in \{v_{i+1,j+2}, v_{i+2,j-1}, v_{i-1,j-2}, v_{i-2,j+1}\}$. The base of induction is $|C| = 0$, when the first agent is about to be added to $C$. In this case, the first agent settles at its current location $v(a) = v_{i,j}$ and establishes the value $r \equiv j - 2i \pmod 5$.

For $|C| > 1$, $C$ already has a diagonal pattern and an active agent $a$ at $v(a) = v_{i',j'}$ aims to join it by connecting to a module centered at $v_{i,j}$. Since $v_{i,j}$ is already in $C$, $j - 2i \equiv r \pmod 5$. It can be seen that for a vertex $v_{i',j'}$ that satisfies the module connection condition with respect to $v_{i,j}$ we have $j' - 2i' \equiv r \pmod 5$. Therefore, the resulting set has a diagonal pattern. □

**Theorem 2.3** *The number of agents used to dominate an $m \times n$ grid $G = (V, E)$ by Algorithm 1 is upper-bounded by $\left\lceil \frac{(m+2)(n+2)}{5} \right\rceil$. For grids with $16 \leq m \leq n$, the number of agents used is upper-bounded by $\gamma_{m,n} + 5$.*

**Proof.** We first prove Algorithm 1 is correct and then show the upper-bound holds. Let $G' = (V', E')$ be the super-grid of $G$ and $C$ denote the non-orphan vertices occupied by previously settled agents when the algorithm finishes. By Lemma 2.3, $C$ constitutes a diagonal pattern and by the condition in Step 10 of the algorithm no other agent can be added to $C$; therefore, $C$ diagonalizes $G$. Moreover, orphans are neighbors of the vertices in $V' \backslash V$ that are initially detected as slots by the settled agents and hence diagonalize $G'$ by Lemma 2.3. Thus, locating one agent on each orphan is equivalent to the projection process. Hence, if a sufficient number of agents exist in the grid, Algorithm 1 provides a dominating set for $G$ (from Theorem 2.2). Consequently, the algorithm is *complete*, meaning it always finds a solution, if one exists.



(a) (b) (c)

Figure 2.6: A $10 \times 15$ grid is depicted with agents shown in blue. In Figure (a) the initial configuration of the agents is shown and Figure (b) demonstrates the agents configuration when Algorithm 1 is finished. In Figure (c), all non-settled and non-asleep agents leave the grid.

Furthermore, since Algorithm 1 performs diagonalization and projection on $G$, from Theorem 2.2 it immediately follows that the number of agents used in the algorithm, $n_a$, is upper-bounded by $\left\lceil \frac{(m+2)(n+2)}{5} \right\rceil$. Also by Theorem 2.1, for $16 \leq m \leq n$ we have $n_a - \gamma_{m,n} \leq 5$. □

Note that while the agents do not form a dominating set for $G$, an active agent finds a valid slot in at most $n+m$ steps. A step is a specified time duration within which an agent performs its basic operation, such as traversing an edge or transmitting signals. Since the number of agents needed to dominate an $m \times n$ grid is less than $mn$, Algorithm 1 takes at most $mn(m+n)$ steps to construct a dominating set for $G$.

### 2.2.5 Simulations

To augment and examine the results discussed in this section, we simulated Algorithm 1 on various grids and different initial configurations of agents on grid vertices. Figure 2.6(a) demonstrates a $10 \times 15$ grid graph with 41 agents located randomly on it. The first agent that activates is located on vertex $(5, 9)$ and hence stays on that vertex. Figure 2.6(b)

shows the location of the agents when Algorithm 1 is complete. It can be seen that every vertex is dominated. However, there are some agents located at vertices (such as $(6, 5)$, $(6, 12)$ and $(7, 15)$) that are never activated in the algorithm. These are the additional agents that are not required to dominate the grid and they are removed in Figure 2.6(c).

## 2.3 $k$-Distance Domination on Grids

In this section, we generalize the centralized algorithm for grid domination discussed in Section 2.1 to the $k$-distance dominating set problem, where a vertex dominates all the vertices within distance $k$ from it. Before defining the problem formally, let $d(u, v)$ denote the shortest path distance between vertices $v, u \in V$ in $G = (V, E)$. Moreover, vertex $u \in V$ is defined as a *$k$-neighbor* of vertex $v \in V$, if $0 < d(u, v) \leq k$. The set of all $k$-neighbors of $v$ is denoted by $N^k(v)$. Moreover, for a set of vertices $W \subset V$ and a vertex $v \in V \backslash W$, we have $u = \text{friend}^k(v, W)$ if (i) $u \in W$, (ii) $u \in N^k(v)$, and (iii) $d(v, u) \leq d(v, w), \forall w \in W$. In other words, $u = \text{friend}^k(v, W)$ if $u$ is the closest vertex in $W$ to $v$ that is within distance $k$ from $v$.

**Definition 2.5** *($k$-Distance Dominating Set Problem) Given a graph $G = (V, E)$, a* $k$-distance dominating set *is a set of vertices $S \subseteq V$ such that for any vertex $v \in V \backslash S$, there exists a vertex $u \in S$ where $u \in N^k(v)$. In the* $k$-distance dominating set problem*, the goal is to find a $k$-distance dominating set that has minimum cardinality. The cardinality of a smallest $k$-distance dominating set for $G$ is called the* $k$-distance domination number *of $G$ and is denoted by $\gamma^k(G)$ [19].*

We say that vertex $u \in S$ *$k$-distance dominates* $v \in V \backslash S$ if $d(u, v) \leq k$. The regular dominating set problem is a special case of the $k$-distance dominating set problem, where $k = 1$. Therefore, $k$-distance domination is also **NP**-hard on general graphs. However, to the best of our knowledge the $k$-distance domination number of grids is not known and the complexity of the problem is open. In Section 2.3.1, we generalize the approaches in Sections 2.1 and 2.2 to provide a $k$-distance dominating set for an $m \times n$ grid graph $G$.

### 2.3.1 Centralized $k$-Distance Domination on Grids

Before discussing the $k$-distance domination algorithms on grids we introduce the following definitions.

**Definition 2.6** *(k-Sub-Grids and k-Super-Grids) An $m \times n$ grid $G = (V, E)$ is called a* $k$-sub-grid *of an $m' \times n'$ grid $G' = (V', E')$ if $G$ is induced by vertices $v'_{i,j} \in V'$, where $k+1 \leq i \leq m'-k$ and $k+1 \leq j \leq n'-k$. If $G$ is a $k$-sub-grid of $G'$, then $G'$ is called the* $k$-super-grid *of $G$.*

**Definition 2.7 ($k$-Boundary)** *For an $m \times n$ grid $G = (V, E)$, the $k$-boundary, denoted by $B^k(G)$, is defined as the set of vertices of $G$ that are not in the $k$-sub-grid of $G$, denoted by $G' = (V', E')$, i.e., $B^k(G) = V \backslash V'$.*

**Lemma 2.4** *For an $m \times n$ grid $G = (V, E)$, we have $|N^k(v)| \leq 2k^2 + 2k + 1$.*

**Proof.** Since $G$ is a grid, the $k$-neighbors of $v$ form a diamond around it with a diameter of $2k+1$ (see the red regions in Figure 2.7). Thus $|N^k(v)|$ is upper-bounded by the area of this region, which is $\left\lceil \frac{(2k+1)^2}{2} \right\rceil = 2k^2 + 2k + 1$. □

In what follows we define $N^k_{\max} = 2k^2 + 2k + 1$.

**Definition 2.8** *(k-Diagonal Pattern) A set of vertices $U \subset V$ constitutes a* $k$-diagonal pattern *on grid $G = (V, E)$ if there exists a fixed $0 \leq r < N^k_{\max}, r \in \mathbb{Z}_{\geq 1}$ such that for any vertex $v_{x,y} \in U$ we have $ky - (k+1)x \equiv r \pmod{N^k_{\max}}$ (see Figure 2.7).*

**Definition 2.9** *(k-Diagonalization) A set of vertices $U \subset V$* $k$-diagonalizes *grid $G = (V, E)$ if it constitutes a $k$-diagonal pattern and there exists no vertex $v \in V \backslash U$ that can be added to $U$ so that $U$ remains a $k$-diagonal pattern.*

Moreover, for a grid $G = (V, E)$ and its $k$-super-grid $G' = (V', E')$, the *$k$-projection* is defined as a special mapping from a subset of vertices in $V' \backslash V$ to their $k$-neighbors in $V$. It is defined formally as follows.

**Definition 2.10** *(k-Projection) Consider a grid $G = (V, E)$ and its $k$-super-grid $G' = (V', E')$. The* $k$-projection *for a set $U' \subseteq V'$ is defined as the set $U'' = \bigcup_{v \in U' \backslash V} \{\text{friend}^k(v, V)\} \cup (U' \cap V)$ (see Figure 2.8).*

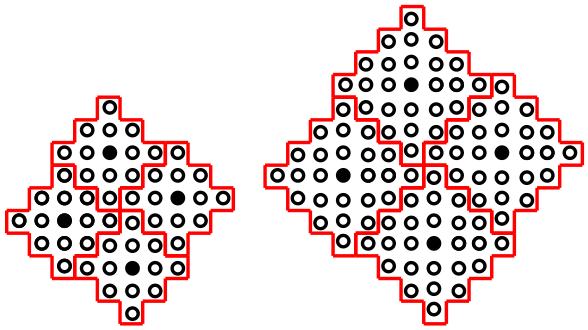



Figure 2.7: A 2-diagonal pattern and a 3-diagonal pattern are depicted. Observe that the structure is similar to the regular diagonal pattern.

**Lemma 2.5** *Let $U$ be a set of vertices that $k$-diagonalizes a grid $G = (V, E)$. For any two vertices $v_{x,y}, v_{x',y'} \in U$ we have $d(v_{x,y}, v_{x',y'}) \geq 2k + 1$.*

**Proof.** Since $v_{x,y}, v_{x',y'} \in U$, we have $y = \frac{1}{k}((k+1)x + r + qN^k_{\max})$ and $y' = \frac{1}{k}((k+1)x' + r + q'N^k_{\max})$, where $r, q, q' \in \mathbb{Z}$ and $0 \leq r < N^k_{\max}$. We define $\Delta_q = q' - q$, $\Delta_x = x' - x$ and $\Delta_y = y' - y = \frac{k+1}{k}\Delta_x + \frac{N^k_{\max}}{k}\Delta_q$. The shortest distance between $v_{x,y}, v_{x',y'}$ is equal to $|\Delta_x| + |\Delta_y|$. From $\Delta_y = \frac{k+1}{k}\Delta_x + \frac{N^k_{\max}}{k}\Delta_q$ it can be observed that as $\Delta_x$ grows, $\Delta_y$ grows faster compared to $\Delta_x$. Hence $|\Delta_x| + |\Delta_y|$ is minimum when $\Delta_y = 0$ and $|\Delta_x| = |\frac{N^k_{\max}}{k+1}\Delta_q|$. Note that the minimum (non-zero) distance occurs for $\Delta_q = 1$ and also is an integer, hence it is lower-bounded by $\left\lceil \frac{2k^2+2k+1}{k+1} \right\rceil = 2k + 1$. □

**Lemma 2.6** *Consider a grid $G = (V, E)$ and its $k$-super-grid $G' = (V', E')$. If $U' \subset V'$ $k$-diagonalizes $G'$, then each vertex in $V$ is $k$-distance dominated by exactly one vertex from $U'$.*

**Proof.** For each vertex $v_{x,y} \in V$, let $r_{v_{x,y}} \equiv ky - (k+1)x \pmod{N^k_{\max}}$. Consider any vertex $v \in V$ and its $k$-neighborhood $N^k(v)$. The distance between any two vertices in $J = \{v\} \cup N^k(v)$ is at most $2k$. Also, there are exactly $N^k_{\max}$ vertices in this set. Thus, for any two distinct vertices $u, w \in J$ we have $r_u \neq r_w$ by Lemma 2.5. Hence each vertex $u \in N^k(v)$ has a distinct value of $r_u$. Consequently, for the value of $r$ that corresponds to

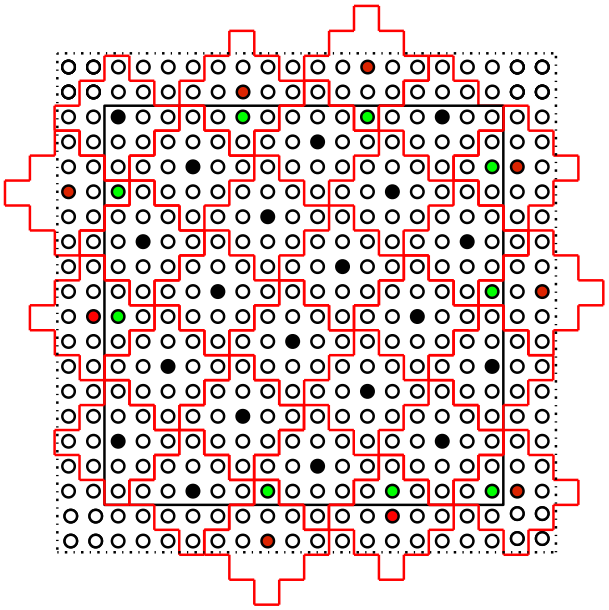


Figure 2.8: A $16 \times 16$ grid $G$ and its 2-super-grid $G'$ are shown by solid and dashed squares, respectively. Both grids are 2-diagonalized. The black circles are the vertices that 2-diagonalize $G$. The union of the red and black circles 2-diagonalizes $G'$. The green circles are the 2-projections of the red circles onto $G$. Before projection the currently green vertices are called orphans.

the diagonalization $U'$, there is exactly one vertex in the $k$-neighborhood of $v$ such that $r_v = r$ and thus $v$ is $k$-distance dominated by exactly one vertex from $U'$. □

**Lemma 2.7** *If a set of vertices $U \subset V$ $k$-diagonalizes an $m \times n$ grid $G = (V, E)$, then $U$ contains at most $\left\lceil \frac{mn}{N^k_{\max}} + k \right\rceil$ vertices.*

**Proof.** Similar to the proof of Lemma 2.1, we partition the set of vertices of grid $G$ and argue about the number of vertices of $U$ that lies in each of those partitions. Since $U$ $k$-diagonalizes $G$, it constitutes a $k$-diagonal pattern on $G$ such that no more vertices can be added to it while maintaining a $k$-diagonal pattern. Therefore, among each $N^k_{\max}$ consecutive vertices in any row or column there is exactly one vertex from $U$. Hence, the number of vertices of $U$ in a row/column of $t$ vertices is at most $\left\lceil \frac{t}{N^k_{\max}} \right\rceil$.

Thus, there are at most $N^k_{\max}$ vertices from $U$ in any $N^k_{\max} \times N^k_{\max}$ grid. Hence, in any $(qN^k_{\max}) \times (pN^k_{\max})$ grid with $p, q \in \mathbb{Z}_{\geq 1}$, there are at most $pqN^k_{\max}$ vertices from $U$. For

an $m \times n$ grid $G = (V, E)$ with $m = qN_{\max}^k + a$, $n = pN_{\max}^k + b$ and $0 \leq a, b < N_{\max}^k$, we partition $V$ into the following four sets:

$$V_1 = \{v_{i,j} |\ 1 \leq i \leq qN_{\max}^k, 1 \leq j \leq pN_{\max}^k\},$$

$$V_2 = \{v_{i,j} |\ qN_{\max}^k + 1 \leq i \leq m, 1 \leq j \leq pN_{\max}^k\},$$

$$V_3 = \{v_{i,j} |\ 1 \leq i \leq qN_{\max}^k, pN_{\max}^k + 1 \leq j \leq n\},$$

and

$$V_4 = \{v_{i,j} |\ qN_{\max}^k + 1 \leq i \leq m, pN_{\max}^k + 1 \leq j \leq n\}.$$

As stated, $|V_1 \cap U| \leq pqN_{\max}^k$. Grid $V_2$ has $a$ columns each having $pN_{\max}^k$ vertices, hence $|V_2 \cap U| \leq pa$. Similarly, we have $|V_3 \cap U| \leq qb$. In summary, we so far have $|(V_1 \cup V_2 \cup V_3) \cap U| \leq pqN_{\max}^k + qb + pa$. Note that $pqN_{\max}^k + qb + pa = \frac{mn}{N_{\max}^k} - \frac{ab}{N_{\max}^k}$.

It remains to upper-bound $|V_4 \cap U|$. By an argument similar to the proof of Lemma 2.1 we show that $|V_4 \cap U| \leq \frac{ab+(N_{\max}^k-1)k}{N_{\max}^k}$. Let $G_4 = (V_4, E_4)$ denote the grid induced by the set of vertices $V_4$ and observe that $|V_4 \cap U|$ $k$-diagonalizes this grid. There are $N_{\max}^k$ different subsets of vertices of $G_4$ that $k$-diagonalize it, each with a different $r$-value. We denote these sets by $U_0, U_1, \ldots, U_{N_{\max}^k-1}$. Without loss of generality assume that $U_0$ and $U_1$ have the maximum and minimum sizes among these sets. By showing that $|U_0| \leq \frac{ab+(N_{\max}^k-1)k}{N_{\max}^k}$ we can conclude that $|V_4 \cap U| \leq \frac{ab+(N_{\max}^k-1)k}{N_{\max}^k}$.

Consider the case that no vertex in $U_0$ is in $B^k(G_4)$, that is, the $k$-boundary of $G_4$. By an argument similar to the one provided in the proof of Lemma 2.1, it can be seen that in this case $|U_0| = |U_1|$. Moreover, since the dimensions of $G_4$, i.e., $a$ and $b$, are less than $N_{\max}^k$, at most one vertex of any row or column of $G_4$ is in $U_0$. We now argue that $|U_0| - |U_1| \leq k$.

Let the vertices in $U_1$ be written of the form $v_{x+i,y+j}$ for some $x, y \in \mathbb{Z}_1$ such that the value of $y - 2x$ modulo $N_{\max}^k$ corresponds to the $r$-value associated with set $U_0$ and $i$ and $j$ are integers such that $-k \leq i, j \leq k$, $-k \leq i + j \leq k$ and $i + j \neq 0$. Assuming $i, j < 0$ (the other three cases can be analyzed similarly), the only vertices in $U_0$ that are missing neighbors from $U_1$ are those that are either in the $|j|$ bottommost rows of the $k$-boundary of $G_4$ or in the $|i|$ leftmost columns of the $k$-boundary. Recall that $|i+j| \leq k$; hence, there are at most $k$ such rows and columns in total. Also, each of these rows/columns contains at most one vertex of $U_0$. Hence, $|U_0| - |U_1| \leq k$.

Since sets $U_0, \ldots, U_{N_{\max}^k-1}$ partition the set of vertices of $G_4$, $|U_0| + \ldots + |U_{N_{\max}^k-1}| = |V_4| = ab$. Moreover, we showed that for any $i \in \{1, \ldots, N_{\max}^k - 1\}$, $|U_i| \geq |U_0| - k$;

therefore, we have $N_{\max}^k|U_0| - (N_{\max}^k - 1)k \leq ab$. Thus, $|V_4 \cap U| \leq |U_0| \leq \frac{ab+(N_{\max}^k-1)k}{N_{\max}^k}$. Combining this with $|(V_1 \cup V_2 \cup V_3) \cap U| \leq \frac{mn}{N_{\max}^k} - \frac{ab}{N_{\max}^k}$ and the fact that $k < N_{\max}^k$, we have $|(V_1 \cup V_2 \cup V_3 \cup V_4) \cap U| \leq \frac{mn}{N_{\max}^k} + \frac{(N_{\max}^k-1)k}{N_{\max}^k} = \left\lceil \frac{mn}{N_{\max}^k} + k \right\rceil$. □

**Theorem 2.4** *For an $m \times n$ grid $G = (V, E)$, a $k$-distance dominating set $S \subset V$ can be constructed using $k$-diagonalization and $k$-projection in polynomial-time such that $|S| \leq \left\lceil \frac{(m+2k)(n+2k)}{N_{\max}^k} + k \right\rceil$.*

**Proof.** The proof follows from Lemmas 2.4, 2.6 and 2.7 and by replacing the diagonalization and projection operations with the $k$-diagonalization and $k$-projection operations in the proof of Theorem 2.2 [11]. □

**Lemma 2.8** *If $S \subset V$ is a $k$-distance dominating set for an $m \times n$ grid $G = (V, E)$, then $|S| \geq \left\lceil \frac{mn}{N_{\max}^k} \right\rceil$.*

**Proof.** According to Lemma 2.4, a vertex $v \in V$ $k$-distance dominates at most $N_{\max}^k$ vertices. Hence, at least $\left\lceil \frac{mn}{N_{\max}^k} \right\rceil$ dominating vertices are needed to $k$-distance dominate an $m \times n$ grid. Note that we use $|S| \geq \left\lceil \frac{mn}{N_{\max}^k} \right\rceil$ instead of $|S| \geq \left\lfloor \frac{mn}{N_{\max}^k} \right\rfloor$ since dominating vertices in the $k$-neighborhood of vertices on the grid boundary do not have all their $k$-neighbors in $V$. □

**Corollary 2.2** *Let $S$ be a $k$-distance dominating set for an $m \times n$ grid $G = (V, E)$ obtained by $k$-diagonalization and $k$-projection and let $L$ denote the lower-bound for the size of set $S$ from Lemma 2.8. For any constant $k \in \mathbb{Z}_{\geq 1}$, the approximation ratio $\frac{|S|}{L}$ satisfies $\lim_{n,m\to\infty} \frac{|S|}{L} = 1$.*

**Proof.** From Theorem 2.4 and Lemma 2.8, we have

$$\frac{|S|}{L} \leq \frac{\lceil (m+2k)(n+2k)/N_{\max}^k + k \rceil}{\lceil mn/N_{\max}^k \rceil}.$$

Therefore,

$$\frac{(m+2k)(n+2k)/N_{\max}^k + k}{mn/N_{\max}^k + 1} \leq \frac{|S|}{L},$$

and

$$\frac{|S|}{L} \leq \frac{(m+2k)(n+2k)/N_{\max}^k + k + 1}{mn/N_{\max}^k}.$$

Hence, we have

$$\frac{(m+2k)(n+2k) + kN_{\max}^k}{mn + N_{\max}^k} \leq \frac{|S|}{L},$$

and

$$\frac{|S|}{L} \leq \frac{(m+2k)(n+2k) + (k+1)N_{\max}^k}{mn}.$$

For constant $k$ we have

$$\lim_{n,m\to\infty} \frac{(m+2k)(n+2k) + kN_{\max}^k}{mn + N_{\max}^k} = \lim_{n,m\to\infty} \frac{(m+2k)(n+2k) + (k+1)N_{\max}^k}{mn} = 1.$$

Therefore by the Squeeze Theorem $\lim_{n,m\to\infty} \frac{|S|}{L} = 1$. □

Recall from Section 1.1 that for a graph $G$, its *$k$-th power*, denoted by $G^k = (V', E')$, is a graph with the same vertex set as $G$, i.e., $V = V'$, in which two distinct vertices are connected by an edge if and only if their distance in $G$ is at most $k$. Hence, in $G^k$ each vertex is connected to the vertices it $k$-distance dominates in $G$. We finish this section with the following remark that relates the $k$-distance dominating set problem in grids to the regular dominating set problem in their $k$-th power graphs.

**Remark 2.2 ($k$-th Power of Grids)** *It might seem that a reasonable approach for $k$-distance domination on a grid $G = (V, E)$ is to simply take the $k$-th power of the grid to obtain $G^k$, and then perform regular domination algorithms on $G^k$. Note that by the definition of $G^k$, a regular dominating set in $G^k$ is equivalent to a $k$-distance dominating set in $G$ and hence $\gamma(G^k) = \gamma^k(G)$. Unfortunately, $G^k$ is no longer a grid (e.g., there are diagonal edges connecting $v_{x,y}$ to $v_{x+1,y+1}$ for $k \geq 2$). In fact, it is not even a planar graph for $m \times n$ grids with $m, n \geq 2$. Therefore, as discussed in Section 1.4.1, choosing dominating vertices greedily in $G^k$ might obtain a dominating set with size as large as $O(\log(|V|))\gamma^k(G)$.*

### 2.3.2 Distributed $k$-Distance Domination on Grids

Using the algorithm explained in Section 2.3.1, a distributed $k$-distance domination algorithm can be designed for grids. In practice, the $k$-distance dominating set problem corresponds to settings where agents are equipped with longer range sensory equipment and can sense vertices up to distance $k$ from them. Therefore, the goal is to arrange the agents on the grid vertices in a distributed way such that for each vertex there exists at least one agent with distance at most $k$ from it.

This algorithm is similar to Algorithm 1 in Section 2.2.3, except for two modifications. The first modification is that in the distributed $k$-distance domination, module $m_2$ and module $m_1$ with module centers $c(m_1) = v_{i,j}$ and $c(m_2) = v_{i',j'}$ can now connect to each other if $v_{i',j'} \in \{v_{i+k,j+k+1}, v_{i+k+1,j-k}, v_{i-k,j-k-1}, v_{i-k-1,j+k}\}$ (see Figure 2.7). These constitute the *slots*. The second modification is the definition of *orphans*. If $U'$ is a set of vertices that $k$-diagonalizes the $k$-super-grid of $G$, then vertex $v \in V$ is an orphan if it satisfies the two following conditions: (i) $v$ has no $k$-neighbor in $U' \cap V$, and (ii) $v$ is in the $k$-neighborhood of a vertex $u \in U' \backslash V$ with the same $x$ or $y$ coordinates. Hence, *valid slots* are defined for each settled agent as the union of its slots located inside the grid and the orphans of its slots located outside the grid (see Figure 2.8).

# Chapter 3

# Robustness Problem

As discussed in Chapter 1, the notion of robustness was first introduced in [23] as a topological property in graphs. Before discussing this concept, some preliminary definitions should be provided. Consider a simple undirected graph $G = (V, E)$. We say that the *connectivity* of $G$ is $c$ if and only if for any two vertices $u, v \in V$ there are at least $c$ pairwise disjoint paths that connect them. Also, from Definition 1.1 recall that a nonempty subset of vertices $A \subset V$ is said to be *$r$-reachable* if there exists a vertex $u \in A$ such that $u$ has at least $r$ neighbors that are not in $A$, i.e., $|N(u)\backslash A| \geq r$, where $r \in \mathbb{Z}_{\geq 0}$. Moreover, as discussed in Definition 1.2, graph $G$ is *$r$-robust* if for any two nonempty and disjoint subsets of its vertices $A, B \subset V$, at least one of them is $r$-reachable, where again $r \in \mathbb{Z}_{\geq 0}$. In other words, graph $G$ is $r$-robust if for any $\emptyset \neq A, B \subset V$ such that $A \cap B = \emptyset$, there exists a vertex $u \in A \cup B$ so that $u$ has at least $r$ neighbors outside its containing set.

A natural question is given a graph $G$ and a number $r$, how hard is it to determine whether $G$ is $r$-robust? One can formalize this decision problem into an optimization problem and ask to find the largest value of $r$ for which a given graph $G$ is $r$-robust. In the next section, we formally define these problems and discuss their hardness. Before going through those results, let us show the relation between the notion of robustness and connectivity in the following lemma.

**Lemma 3.1** *If a graph $G = (V, E)$ is $r$-robust, then the connectivity of $G$ is at least $r$ [23].*

**Proof.** Assume, by the way of contradiction, that the connectivity of $G$ is less than $r$. Hence, there exists a subset of vertices $S \subset V$ with cardinality less than $r$ whose removal

results in several disjoint connected components. Take any two of these components and let $A$ and $B$ denote their set of vertices. Now, consider the same subset of vertices in $G$ and call them sets $A'$ and $B'$. It can be observed that neither of these sets are $r$-reachable; otherwise, there is vertex $u$ in $A'$ or $B'$ with at least $r$ neighbors outside its set. Since the size of the removed subset of vertices is less than $r$, then $u$ has neighbors in other disjoint connected components caused by the removal of $S$ which contradicts the disjointness of those components. □

In the rest of this chapter, we study the complexity of the robustness problem and show that it lies in the complexity class **coNP**-complete. Furthermore, we discuss a short result on the hardness of approximating a problem closely related to the robustness problem, which is defined in the following section.

## 3.1 Complexity of Determining Degree of Robustness in General Graphs

The following simple result establishes the complexity of checking whether a given graph is 1-robust.

**Lemma 3.2** *A graph $G$ is 1-robust if and only if it is 1-connected.*

**Proof.** By Lemma 3.1, we know that 1-robustness implies 1-connectedness; hence, we just need to prove the opposite statement. If the graph is 1-connected, then every subset of its nodes has at least one neighbor outside that set; otherwise, the graph will be disconnected. Thus, for every pair of disjoint and nonempty sets, each of them contains at least one node with at least one neighbor outside its set and so the graph is 1-robust. □

Since there exist polynomial time algorithms to determine graph connectivity [12], 1-robustness can be checked in polynomial time. However, in the rest of this section we will show that finding whether general graphs are $r$-robust for $r \geq 2$ is **coNP**-hard. This is done by showing that a problem closely related to the robustness problem is **NP**-hard in general graphs. Before discussing the reduction, recall from Chapter 1 that for a graph $G = (V, E)$, a *cut* $C = (S, V \backslash S)$ is defined as a partition of nodes of $G$ into two nonempty subsets $S \subset V$ and $V \backslash S$. The *cut-set* of a cut $C = (S, V \backslash S)$ is defined as the subset of the edges of $G$ with one endpoint in $S$ and the other in $V \backslash S$. We now can define the robustness and $r$-robustness problems formally.

**Definition 3.1 (Robustness and $r$-Robustness Problems)** *Given a connected graph $G$, the* robustness problem *determines the largest value of $r$ such that $G$ is $r$-robust. The* $r$-robustness problem *is the decision version of the robustness problem that determines whether graph $G$ is $r$-robust for a given $r \in \mathbb{Z}_{\geq 1}$.*

If a graph is not $r$-robust, then there exist two nonempty and disjoint subsets of nodes $A$ and $B$ such that all nodes in these sets have at most $r-1$ neighbors outside their containing sets. Note that nodes in set $X = V \backslash (A \cup B)$ can have any number of neighbors outside $X$. In other words, if one claims that a graph is $r$-robust, then there is no apparent way to prove this without checking all pairs of disjoint and nonempty subsets of vertices of that graph and showing that at least one set in each pair is $r$-reachable. This is intractable as the number of such subsets is exponential in the size of the input graph. On the other hand, to prove that a graph is not $r$-robust, one only needs to provide one pair of disjoint and nonempty subsets of vertices such that neither of them are $r$-reachable. Therefore, in the $r$-robustness problem, the 'No' instances (input graphs that are not $r$-robust) have certificates that can be checked in polynomial time. Thus, the $r$-robustness problem is in the complexity class **coNP** [3].

The complexity of the robustness problem is equivalent to the complexity of the $r$-robustness problem within a factor of $O(\log n)$ (as a binary search can be used for finding the largest value of $r$ for which the graph is $r$-robust). To prove the **coNP**-completeness of the $r$-robustness problem, we instead show that the complement of the $r$-robustness problem, which we call the "$\rho$-degree cut problem", is **NP**-hard. Note that the complement of a decision problem is obtained by reversing the 'Yes' and 'No' answers of all input instances. In other words, if problems $P_1$ and $P_2$ are complements, then the output of $P_1$ to an input instance is 'Yes' if and only if the output of $P_2$ to that instance is 'No'. Therefore, the complement of a problem in **NP** is in **coNP**, and vice versa. Hence, the $\rho$-degree cut problem, formally defined below, is in **NP**.

**Definition 3.2 ($\rho$-Degree Cut Problem)** *Given a connected graph $G = (V, E)$, the* $\rho$-degree cut problem *determines whether there exist two nonempty and disjoint subsets of vertices $A, B \subset V$ such that each vertex in $A$ (resp. $B$) has at most $\rho$ neighbors outside $A$ (resp. $B$), where $\rho \in \mathbb{Z}_{\geq 0}$.*

It can be shown that if a problem is **NP**-hard, then its complement is **coNP**-hard (see [3] for a detailed discussion on complexity classes **NP** and **coNP**). Hence, instead of directly showing that the $r$-robustness problem is **coNP**-complete we show that its

complement, i.e. the $\rho$-degree cut problem, is **NP**-hard. Moreover, recall that the $r$-robustness problem is in **coNP**; therefore, knowing that the $\rho$-degree cut problem is **NP**-hard it can be concluded that the $r$-robustness problem is **coNP**-complete. To prove the **NP**-hardness of the $\rho$-degree cut problem we first study the hardness of a relaxed version of this problem called the *relaxed-$\rho$-degree cut problem*, defined as follows.

**Definition 3.3 (Relaxed-$\rho$-Degree Cut Problem)** *The* relaxed-$\rho$-degree cut problem *determines whether there exists a cut $C = (A, V \backslash A)$ in a given graph $G$ such that each node of the graph is incident to at most $\rho$ edges in the cut-set, where $\rho \in \mathbb{Z}_{\geq 0}$.*

Note that the relaxed-1-degree cut problem is equivalent to a known problem called the "matching-cut problem" in which the goal is to find whether there exists a cut in the graph which is also a matching, i.e., the edges in the cut-set form a matching [29]. By the definition of matching, a set of edges form a matching if and only if they do not share an endpoint [12]. It can be seen that this is the same as the definition of the relaxed-1-degree cut problem and hence the two problems are equivalent. Moreover, in [29] it is shown that the matching-cut problem is **NP**-complete via a reduction from NAE3SAT [33]. By the equivalence of the relaxed-1-degree cut problem and the matching-cut problem, this shows the **NP**-hardness of the relaxed-1-degree cut problem as well. However, in order to prove the **NP**-hardness of the 1-degree cut problem (and subsequently the $\rho$-degree cut problem), we will need some alterations to the proof in [29]. We will thus start by modifying the reduction in [29] from NAE3SAT to the matching-cut problem (or the relaxed-1-degree cut problem) and then extend that to prove the **NP**-hardness of the 1-degree cut problem and eventually the $\rho$-degree cut problem. We start by defining NAE3SAT formally [33].

**Definition 3.4 (NAE3SAT)** *For a set of clauses each containing three literals from a set of boolean variables in* Conjunctive Normal Form *(CNF),* NAE3SAT *determines whether there exists a truth assignment of the variables so that each clause contains at least one true and one false literal.*

It was shown by Shaefer in [33] that NAE3SAT is **NP**-complete. Here, we discuss a reduction from NAE3SAT to the relaxed-1-degree cut problem by constructing a graph $G(\phi)$ for any given CNF formula $\phi$ such that $G(\phi)$ has a relaxed-1-degree cut (i.e., $G(\phi)$ has a cut where each vertex of the graph is incident to at most one edge in the cut-set) if and only if $\phi$ can be satisfied within the NAE3SAT constraints. Let formula $\phi$ consist of $m$ clauses $C_1, \ldots, C_m$, where each clause contains three literals from the set of variables

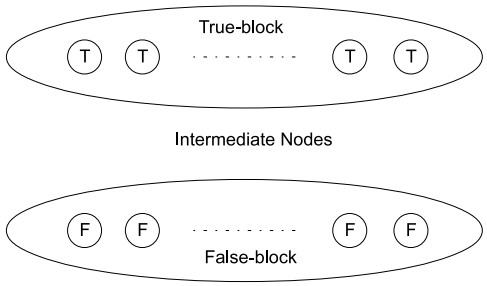



Figure 3.1: The True-block and False-block in the construction of $G(\phi)$. Each block is a complete subgraph with $4m + t$ vertices.

$X = \{x_1, \ldots, x_t\}$. The construction of graph $G(\phi)$ from a given CNF formula $\phi$ is as follows.

We first build two *blocks*, where each block is a complete graph of $4m + t$ vertices. The upper and lower blocks are labeled the *True-block* and *False-block*, respectively, as illustrated in Figure 3.1. We complete the construction of $G(\phi)$ by adding subgraphs representing the variables and clauses of $\phi$ to these blocks in such a way that if $G(\phi)$ has a relaxed-1-degree cut, then there exists a cut that separates the two blocks by cutting through these subgraphs and partitioning the vertices into two sets that are 1-reachable but not 2-reachable.

The subgraphs to be added to the blocks are of two types: (i) variable-gadgets, and (ii) clause-gadgets. A variable-gadget is incorporated for each variable $x_i \in X$. This gadget contains two vertices representing $x_i$ and $\bar{x}_i$ (the binary complement of $x_i$), each connected to the True and False-blocks as illustrated in Figure 3.2(a). Moreover, for each clause in $\phi$, a clause-gadget is constructed by connecting three nodes (each representing a literal of the clause) in addition to some extra nodes to the True and False-blocks as depicted in Figure 3.2(b). Finally, there are edges, called the 'intermediate edges', connecting each literal-node in each clause-gadget to its corresponding variable-node in the variable-gadgets. An example of $G(\phi)$ for $\bar{x}_1 \vee x_2 \vee x_3$ is demonstrated in Figure 3.3.

If graph $G(\phi)$ has a relaxed-1-degree cut, then there exists a cut $C = (A, V \backslash A)$ that partitions the nodes of $G(\phi)$ into two sets $A$ and $B = V \backslash A$ such that no node of the graph has more than one neighbor outside its set. In the following lemmas, we show that this cut $C$ satisfies some useful properties (all of these lemmas assume that graph $G(\phi)$ has a relaxed-1-degree cut and pertain to the cut $C = (A, B)$ just described).

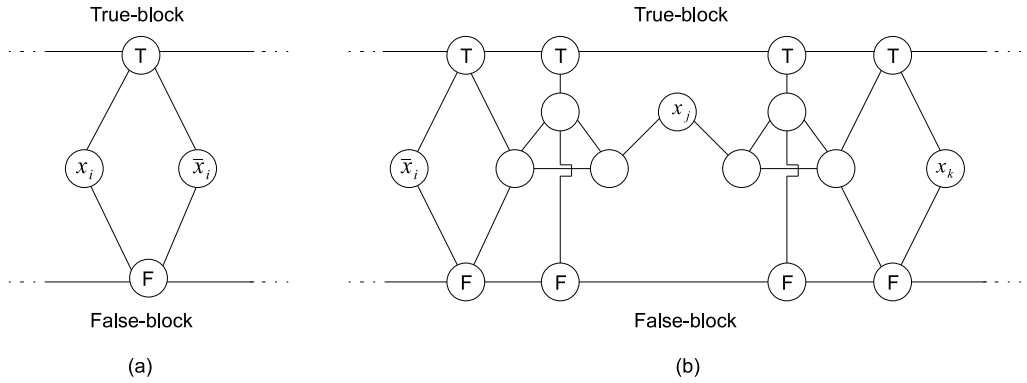



Figure 3.2: Figure (a) demonstrates the variable-gadget for variable $x_i$, and (b) shows the clause-gadget for clause $\bar{x}_i \vee x_j \vee x_k$.

**Lemma 3.3** *Let $T$ (resp. $F$) be the set of all the vertices in the True-block (resp. False-block) of graph $G(\phi)$. Then, $T \subseteq A$ or $T \subseteq B$ (resp. $F \subseteq A$ or $F \subseteq B$).*

**Proof.** Since each block is a complete graph with more than three nodes, cut $C = (A, B)$ cannot separate the nodes in the same block; otherwise, there exists a node in the block that has at least two neighbors outside its own set. □

By the above lemma, cut $C$ cannot go through the True and False-blocks of $G(\phi)$. We assign true values to the vertices in the variable and clause-gadgets that are left connected to the True-block by $C$ and false values to the others, i.e., vertices that are left connected to the False-block by $C$.

**Lemma 3.4** *Cut $C = (A, B)$ has the following two properties:*

1. *For each variable-gadget, cut $C$ leaves the variable-node and its negation node in opposite sets, i.e., they have opposite truth assignments.*
2. *For each clause-gadget, cut $C$ leaves at least one literal-node in set $A$ and one literal-node in $B$, i.e., at least one literal-node has true assignment and one has false assignment.*

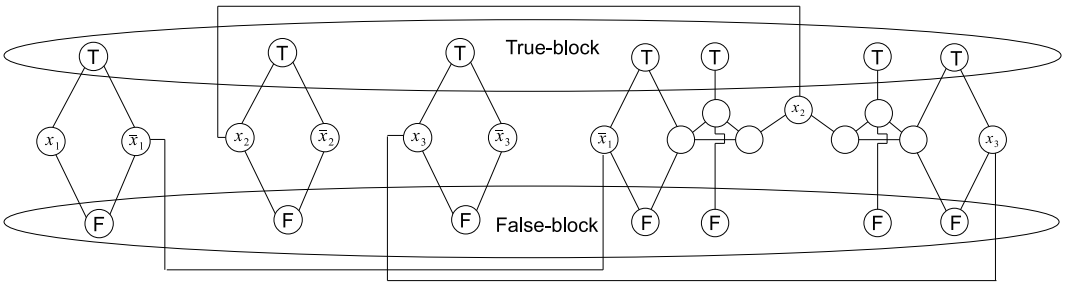



Figure 3.3: The construction of clause $\bar{x}_1 \vee x_2 \vee x_3$ with each literal-node connected to its corresponding variable-node in the variable-gadgets.

**Proof.** By Lemma 3.3, there are only two cases to consider: (i) all nodes in both the True and False-blocks are in $A$ (or in $B$), and (ii) all nodes in the True-block are in $A$ and all nodes in the False-block are in $B$ (or vice versa).

In case (i), if there exists a node from a variable-gadget in set $B$, then that node immediately has at least two neighbors in $A$, contradicting the definition of cut $C$ (see Figure 3.2(a)). Similarly, it can be argued that in this case no node of any clause-gadget can be in set $B$ (see Figure 3.4). The vertices labeled 1, 2, 3, 7, 8 and 9 cannot be in $B$ since they would have at least two neighbors in $A$ otherwise. Since vertices 2, 3, 7 and 8 are in $A$, vertices 4 and 6 cannot be in $B$ either. Thus vertex 5 should also be in $A$; otherwise, it violates the definition of cut $C$.

Hence, the only possibility is that all nodes in variable-gadgets and clause-gadgets are in $A$. This makes $B$ empty and violates the definition of cut $C$. Thus, case (i) cannot hold and it only remains to study case (ii).

In case (ii), if both nodes of a variable-gadget are in the same set (say $A$), then a node from the False-block in set $B$ has two neighbors in $A$ (as seen in Figure 3.2(a)). This contradicts the definition of cut $C$. The only possible cuts through variable-gadgets for this case are shown in Figure 3.5, which leave any variable-node and its negation node on opposite sides of $C$ and thus concludes the first property in the lemma.

Moreover, in case (ii) suppose all three literal-nodes of a clause-gadget are in set $A$ (the case that all three literal-nodes of a clause-gadget are in $B$ can be handled via identical arguments). Since the vertex labeled 5 (in Figure 3.4) is in set $A$, then to respect the properties of cut $C$, at least one of its neighbors should also lie in $A$. Without loss of generality

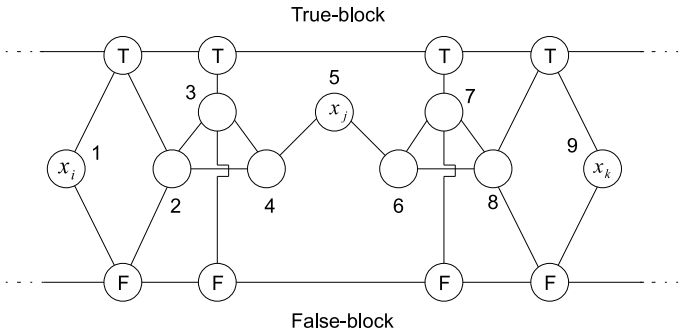



Figure 3.4: All the nine vertices in a clause-gadget are labeled.

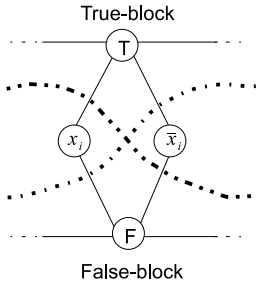



Figure 3.5: The only two possible cuts through a variable-gadget resulting in all nodes having at most one neighbor on opposite sides of the cut.

due to symmetry, assume that vertex 4 is in $A$. This implies that vertex 2 is also in $A$; otherwise, cut $C$ results in a 2-reachable set. Finally, since both vertices 1 and 2 are in $A$, the vertex in the False-block connected to both of them has two neighbors in $A$. This contradicts the definition of $C$; thus $C$ cannot leave all literal-nodes in a clause-gadget in the same set. The only possible cuts through clause-gadgets for this case are the ones illustrated in Figure 3.6. It can be seen that none of these six cuts leaves all three literal-nodes of a clause-gadget on one side of the cut. Hence the second property in the lemma also holds. □

**Lemma 3.5** *All literal-nodes have the same truth values as their corresponding variable-nodes.*

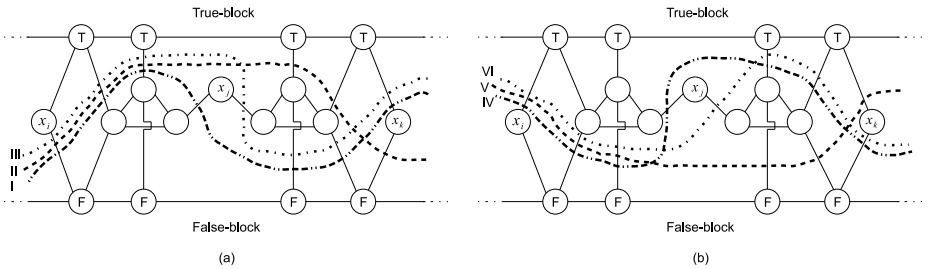



Figure 3.6: The six allowed cuts through the clause-gadget shown in Figure 3.2(b) that result in two 1-reachable but not 2-reachable sets.

**Proof.** First, note that a literal-node has the same truth value as its corresponding variable-node if and only if they lie on the same side of cut $C = (A, B)$. In the proof of Lemma 3.4, it was shown that the only possible case for the True and False-blocks is the case that all nodes in the True-block are in $A$ and all nodes in the False-block are in $B$ (or vice versa).

In this case, assume that there exists a variable-node $x$ in set $A$ such that its corresponding literal-node lies in set $B$ (the case that the variable-node is in $B$ and the corresponding literal-node is in $A$ can be analyzed similarly). Then, node $x$ has the following two neighbors in set $B$: its corresponding literal-node and the node in the False-block that it is connected to (see Figure 3.5). Hence, the edges connecting these two nodes to node $x$ are excised by cut $C$. This contradicts the fact that $A$ is not 2-reachable and therefore is not possible. □

Using the properties stated in the above lemmas we obtain the following result.

**Lemma 3.6** *The relaxed-1-degree cut problem is **NP**-hard.*

**Proof.** We prove this claim holds by a reduction from NAE3SAT. To do that we show graph $G(\phi)$ has a relaxed-1-degree cut if and only if $\phi$ has a solution within the NAE3SAT constraints. Suppose that $G(\phi)$ has a relaxed-1-degree cut. Then, a cut $C = (A, B)$, where $B = V \backslash A$, can be found such that all vertices of the graph are incident with at most one edge in the cut-set.

By the first part of Lemma 3.4, cut $C$ has to go through the edges of each variable-gadget as depicted in Figure 3.5 and leaves each variable-node and its negation node on opposite sides of the cut and so determines their truth assignments. Also, by the second

| $x_i$ | $x_j$ | $x_k$ | Cut |
|---|---|---|---|
| T | T | T | no cut |
| T | T | F | I |
| T | F | T | II |
| T | F | F | III |
| F | F | T | IV |
| F | T | F | V |
| F | T | T | VI |
| F | F | F | no cut |

Table 3.1: The truth assignments corresponding to different cuts in the clause-gadget demonstrated in Figure 3.6.

part of Lemma 3.4, the clause-gadgets are cut by $C$ according to one of the six cases illustrated in Figure 3.6, which results in having at least one true and one false literal-node in each clause-gadget. Furthermore, note that by Lemma 3.5, the intermediate edges are never cut by $C$. Hence, all the literal-nodes corresponding to the same variable-node are left in the same set as that variable-node and the negated literal-nodes are in the other set. Consequently, if $G(\phi)$ has a relaxed-1-degree cut, then $\phi$ is satisfiable within the NAE3SAT constraints.

On the other hand, if $\phi$ has a solution under the NAE3SAT constraints, then a cut $C = (A, V \backslash A)$ can be found in $G(\phi)$ such that (i) each variable-gadget is cut so that the variable-node and its negation node are connected to the blocks labeled with their truth values, and (ii) each clause-gadget is cut according to its truth assignment as illustrated in Table 3.1. It can be easily observed that using this cut, no node of graph $G(\phi)$ is incident with more than one edge of the cut-set and hence $G(\phi)$ has a relaxed-1-degree cut. □

Recall that the relaxed-1-degree cut problem is equivalent to the matching-cut problem that was shown to be **NP**-complete in [29]. The difference between the proof we provided here and the proof in [29] is in the construction of the clause-gadgets; our construction will allow us to show the **NP**-hardness of the (more general) 1-degree cut problem. These changes are made so that if $G(\phi)$ has a 1-degree cut then $X = \emptyset$; which results in having a relaxed-1-degree cut that was earlier shown to be **NP**-hard.

We now construct a graph based on $G(\phi)$ to show that the 1-degree cut problem is also **NP**-hard. Graph $H(\phi)$ is built by taking three copies of $G(\phi)$ and adding edges to form one complete subgraph on all nodes in the three True-blocks and another complete subgraph on all nodes in the three False-blocks. We refer to each of these copies of $G(\phi)$ used in building

$H(\phi)$ as a 'box'. Figure 3.7 illustrates $H(\phi)$ using the graph $G(\phi)$ for $\phi = \bar{x}_1 \vee x_2 \vee x_3$ from Figure 3.3. Using this construction we can now prove the **NP**-hardness of the 1-degree cut problem.

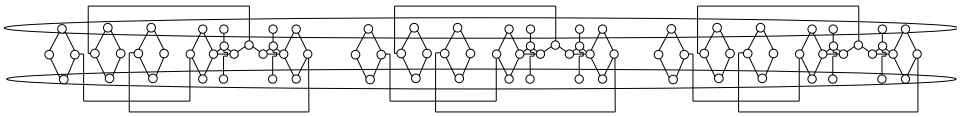


Figure 3.7: The construction of $H(\phi)$ from the graph $G(\phi)$ depicted in Figure 3.3.

**Theorem 3.1** *The 1-degree cut problem is **NP**-hard.*

**Proof.** We show that the 1-degree cut problem is **NP**-hard by showing that $H(\phi)$ has a 1-degree cut if and only if $G(\phi)$ has a relaxed-1-degree cut for any instance $\phi$ of NAE3SAT. It can be easily seen that if $G(\phi)$ has a relaxed-1-degree cut then $H(\phi)$ also has a relaxed-1-degree cut (e.g., simply replicate the cut in $G(\phi)$ for each box in $H(\phi)$) and thus a 1-degree cut. It only remains to show if $H(\phi)$ has a 1-degree cut then $G(\phi)$ has a relaxed-1-degree cut. Assume that sets $A, B$ and $X$ partition the nodes of $H(\phi)$ such that (i) $A$ and $B$ are nonempty, and (ii) each node in $A$ and $B$ has at most one neighbor outside its own set (i.e., $A$, $B$ and $X$ specify a 1-degree cut). Then, we show that there exist two nonempty sets $A'$ and $B'$ that partition the nodes of $G(\phi)$ such that no node in $A'$ has more than one neighbor in $B'$, and vice versa.

First, for any clique in $H(\phi)$ with at least three nodes, the fact that $A$ and $B$ are not 2-reachable implies the following two properties:

1. Set $X$ can contain zero, one, or all nodes of the clique, and
2. If a node of the clique is in $A$ (resp. $B$), then no node of that clique is in $B$ (resp. $A$).

Let $T$ and $F$ denote the set of all nodes in the True and False-blocks of graph $H(\phi)$. Several different scenarios can take place for sets $T$ and $F$ with respect to sets $A, B$ and $X$. First, consider the case that both $T$ and $F$ are subsets of $A$ (the case that both $T$ and $F$ are subsets of $B$ can be analyzed similarly). Since each box of $H(\phi)$ is isomorphic to $G(\phi)$, by Lemma 3.3 this scenario is not possible. Now, by property (2) stated above and

without loss of generality due to symmetry, assume that $T \subseteq (A \cup X)$ and $F \subseteq (B \cup X)$. Suppose that $T \cap X \neq \emptyset$; therefore, by property (1) either $T \subseteq X$ or $|T \cap X| \leq 1$. In the former case, the True-blocks of all boxes in $H(\phi)$ are subsets of $X$. In the latter case, there exist at least two boxes in $H(\phi)$ whose True-blocks are subsets of $A$. The same argument holds for the False-block of $H(\phi)$ and hence if $F \nsubseteq X$, then there exist at least two boxes in $H(\phi)$ whose False-blocks are subsets of $B$. Consequently, there exists a box in $H(\phi)$, denoted by $G'(\phi)$, such that its True-block is either a subset of $A$ or $X$ and its False-block is either a subset of $B$ or $X$.

Furthermore, if $T \subseteq X$ and $F \subseteq X$, then due to nonemptiness of set $A$ (resp. $B$), there should exist at least one node in the set of all the other nodes in the variable and clause-gadgets of $H(\phi)$ that is in $A$ (resp. $B$). However, the variable-nodes in variable gadgets and vertices labeled 1, 2, 3, 7, 8 and 9 in clause-gadgets (as in Figure 3.4) cannot lie in sets $A$ or $B$; otherwise, they have at least two neighbors in the union of $T$ and $F$ which are subsets of $X$ and hence imply that set $A$ or $B$ is 2-reachable. Now that vertices 2, 3, 7 and 8 are in $X$, in order to avoid set $A$ or $B$ being 2-reachable, vertices 4 and 6 should lie in $X$ as well. This results in vertex 5 being in $X$, leaving $A$ and $B$ empty.

Consequently, it is not possible for the True and False-blocks of $H(\phi)$, i.e., sets $T$ and $F$, to both be subsets of set $X$. This implies that the True and False-blocks of $G'(\phi)$ cannot both be subsets of $X$ either. In summary, the only possible cases for the True and False-blocks of $G'(\phi)$ are as follows: the True-block of $G'(\phi)$ is a subset of $X$ and the False-block is a subset of $A$ or $B$ (or vice versa), or the True-block of $G'(\phi)$ is a subset of $A$ and the False-block is subset of $B$ (or vice versa). These cases are further studied in the following. Before discussing these cases, recall that $G'(\phi)$ and $G(\phi)$ are isomorphic and sets $A$ and $B$ can be used interchangeably in the arguments by symmetry.

First, assume that all nodes in the True-block of $G'(\phi)$ are in $X$ and all nodes in the False-block of $G'(\phi)$ are in $B$. The argument in this case is similar to the argument provided for the case that both the True and False-blocks of $H(\phi)$ were in $X$. Again, nodes in the variable-gadgets and vertices labeled 1, 2, 3, 7, 8 and 9 in the clause-gadgets cannot lie in $A$. This results in vertices labeled 4 and 6 not being in $A$. Consequently, vertices labeled 5 cannot be in $A$ either and hence set $A$ becomes empty which is not admissible.

Therefore, we now assume that all nodes in the True-block of $G'(\phi)$ are in $A$. Hence, all nodes in the False-block are either in $X$ or $B$. If the nodes in the False-block are in $X$, then the same argument as above shows that set $B$ should be empty which violates its properties and hence is not allowed. If all the nodes in the False-block are in $B$, then we show that no node of $G'(\phi)$ can be in set $X$.

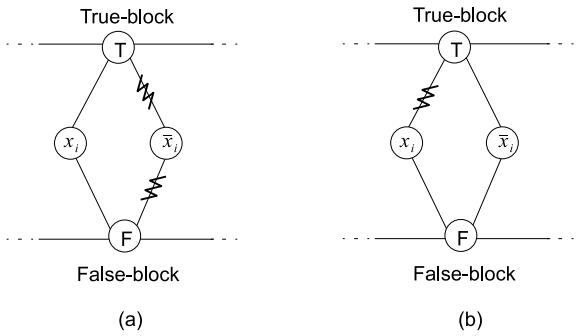



Figure 3.8: Here, the True-block is a subset of $A$, the False-block is a subset of $B$ and the variable-node corresponding to $\bar{x}_i$ is in $X$. Figure (a) shows the edges connected to node $\bar{x}_i$ that are cut. In Figure (b), without loss of generality, it is assumed that node $x_i$ is either in $B$ or $X$ and its incident edges with the other endpoints in $A$ are marked. It can be seen that the node in the True-block has two adjacent nodes outside $A$.

Suppose that there exists a node in a variable-gadget in $G'(\phi)$ that lies in $X$. Then the two nodes in the True and False-blocks connected to that node have a neighbor outside their containing sets, i.e., a neighbor in $X$ (Figure 3.8(a)). Moreover, the other variable-node in that variable-gadget is in $A, B$ or $X$. In all these cases this node is in a different set from at least one of its two neighbors in the True and False-blocks (Figure 3.8(b)). Hence, there exists at least one node in $A$ or $B$ that has two neighbors outside $A$ or $B$, respectively. Therefore, no node in the variable-gadgets can be in set $X$.

Furthermore, observe that since the True-block is a subset of $A$ and the False-block is a subset of $B$, then each variable-node has at least one neighbor outside its containing set. Since it was assumed that each node in $A$ and $B$ has at most one neighbor outside its set, it follows that all other neighbors of a variable-node should lie in the same set as that node. Therefore, in $G'(\phi)$, the endpoints of all intermediate edges, i.e., the edges connecting literal-nodes in the clause-gadgets to the corresponding variable-nodes, lie in the same sets. This, in combination with the fact that none of the variable-nodes in $G'(\phi)$ are in $X$ shows that no literal-node in any clause-gadget of $G'(\phi)$ is in $X$. It only remains to show that the non-literal-nodes in the clause-gadgets of $G'(\phi)$, i.e., nodes labeled 2, 3, 4, 6, 7 and 8 as in Figure 3.4, are not in $X$ either. By the same argument as for variable-nodes, nodes labeled 2 and 8 cannot lie in $X$. Also, note that since the True and False-blocks of $G'(\phi)$ are subsets of $A$ and $B$, respectively, and node 2 (resp. node 8) is either in $A$ or $B$, then one of the edges connecting node 2 (resp. node 8) to the True and False-blocks is

excised. Therefore, all its other neighbors, i.e., nodes 3 and 4 (resp. nodes 6 and 7), should lie in the same set as node 2 (resp. node 8). As a result, nodes 3, 4, 6 and 7 cannot be in $X$. Consequently, no node of $G'(\phi)$ lies in $X$.

We have thus shown that if $H(\phi)$ has a 1-degree cut then $G'(\phi)$ has a relaxed-1-degree cut. Since $G(\phi)$ and $G'(\phi)$ are isomorphic, graph $G(\phi)$ also has a relaxed-1-degree cut. Also, we define sets $A'$ and $B'$ as the subsets of nodes in $G(\phi)$ that correspond to the intersections of $A$ and $B$ with the set of nodes of $G'(\phi)$, respectively. Therefore, sets $A'$ and $B'$ are nonempty and partition the nodes of $G(\phi)$. Moreover, each node of $G(\phi)$ has at most one neighbor outside its containing set. Consequently, $H(\phi)$ has a 1-degree cut if and only if $G(\phi)$ has a relaxed-1-degree cut for any NAE3SAT instance $\phi$. □

The **NP**-hardness of the 1-degree cut problem results in the **coNP**-hardness of its complement problem, i.e., the 2-robustness problem. Since the 2-robustness problem is in **coNP**, we can conclude that it is **coNP**-complete. We now seek a stronger result, that is, showing that for any $r \in \mathbb{Z}_{\geq 2}$ the $r$-robustness problem is **coNP**-complete. This requires the **NP**-hardness of the $\rho$-degree cut problem for any $\rho \in \mathbb{Z}_{\geq 1}$ and to prove this, we first show that for any $\rho \in \mathbb{Z}_{\geq 1}$, the relaxed-$\rho$-degree cut problem is **NP**-hard. This is done in the following lemma by making the necessary modifications to the graph $G(\phi)$ built for the NAE3SAT instance $\phi$, as introduced earlier in this section (e.g., Figure 3.3).

**Lemma 3.7** *For any $\rho \in \mathbb{Z}_{\geq 1}$, the relaxed-$\rho$-degree cut problem is **NP**-hard.*

**Proof.** Recall that in order to show that the relaxed-1-degree cut problem is **NP**-hard, for any NAE3SAT instance $\phi$ we constructed a graph $G(\phi) = (V, E)$ such that $G(\phi)$ had a relaxed-1-degree cut if and only if $\phi$ was satisfiable within the NAE3SAT constraints. Here, for any $\rho \in \mathbb{Z}_{\geq 1}$, we make two modifications to $G(\phi)$ to construct graph $G_\rho(\phi) = (V_\rho, E_\rho)$ so that $G_\rho(\phi)$ has a relaxed-$\rho$-degree cut if and only if the original graph $G(\phi)$ has a relaxed-1-degree cut. We can then conclude that the relaxed-$\rho$-degree cut problem is **NP**-hard.

The two modifications to graph $G(\phi)$ are: (i) a modification to the nodes in the True and False-blocks of $G(\phi)$, and (ii) a modification to the other nodes of $G(\phi)$, i.e. nodes in the variable and clause-gadgets. In modification (i), for any node $v$ in the True-block (resp. False-block) of $G(\phi)$, we add $\rho - 1$ nodes in the False-block (resp. True-block) and connect them to $v$. Thus, this step adds a total of $2(\rho - 1)(4m + t)$ nodes to the graph. In modification (ii), for any node $u$ in the variable and clause-gadgets of $G(\phi)$ that is not in the True or False-blocks, we add $\rho - 1$ nodes in each of the True and False-blocks and connect $u$ to them. This step adds a total of $2(\rho - 1)(9m + 2t)$ nodes to the graph. The

new graph is called $G_\rho(\phi)$. Note that the nodes added to the True and False-blocks of $G(\phi)$ to construct $G_\rho(\phi)$ are connected to all other vertices in those blocks and hence the True and False-blocks of $G_\rho(\phi)$ are complete subgraphs. Figure 3.9 demonstrates $G_2(\phi)$ for graph $G(\phi)$ shown in Figure 3.3.

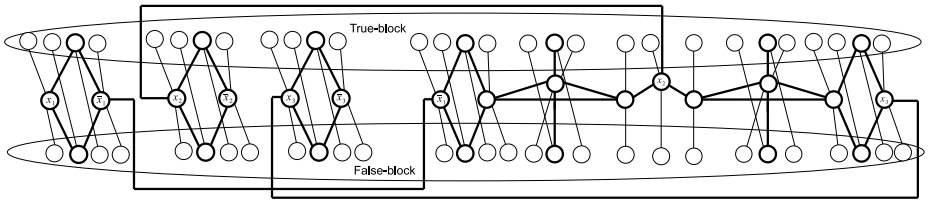



Figure 3.9: Graph $G_2(\phi)$ constructed from graph $G(\phi)$ demonstrated in Figure 3.3. The highlighted nodes and edges are the ones that exist in $G(\phi)$ as well.

If graph $G_\rho(\phi)$ has a relaxed-$\rho$-degree cut, then there exists a cut $C_\rho = (A, V_\rho \backslash A)$ that partitions the nodes of $G_\rho(\phi)$ into two nonempty sets $A$ and $B = V_\rho \backslash A$ such that no node of the graph has more than $\rho$ neighbors outside its containing set. It is easy to observe that if $G(\phi)$ has a relaxed-1-degree-cut $C = (A', B')$, then $G_\rho(\phi)$ has a relaxed-$\rho$-degree cut $C_\rho = (A, B)$, by assigning the vertices corresponding to those in $A'$ and $B'$ in $G(\phi)$ to sets $A$ and $B$ in $G_\rho(\phi)$, respectively. Furthermore, the newly added vertices in the True and False-blocks of $G_\rho(\phi)$ should be assigned to the sets the already allocated vertices in those blocks are assigned to (recall that by Lemma 3.3, in the relaxed-1-degree cut problem all the vertices in each of the True and False-blocks belong to the same subset of vertices). So, it remains to prove the converse statement holds.

Note that the True and False-blocks of $G_\rho(\phi)$ cannot both lie in set $A$ (or similarly $B$). It can be easily checked that except for vertices labeled 4, 5 and 6 as in Figure 3.4, if any other vertex of $G_\rho(\phi)$ lies in $B$, then it has at least $2\rho$ neighbors outside its containing set in the True and False-blocks and hence violates the definition of cut $C_\rho$. This results in the fact that now vertices labeled 4 and 6 cannot lie in $B$, since they have $2\rho$ neighbors in set $A$. Consequently, vertices labeled 5 should also lie in $A$ which makes $B$ empty and hence is not allowed.

Without loss of generality, we now assume that the True-block of $G_\rho(\phi)$ is a subset of $A$ and its False-block is a subset of $B$. In the following, we prove that in this case, cut $C_\rho$ has three properties.

First, similar to the relaxed-1-degree cut problem, both a variable-node and its negation node in a variable-gadget cannot lie in set $A$ (resp. $B$); otherwise, the node in the False-

block (resp. True-block) connected to both these nodes has $\rho + 1$ neighbors outside its set.

Second, since a variable-node and its negation node in a variable-gadget lie in different sets, each of these nodes are incident with $\rho$ edges in the cut-set. Therefore, no other edge connected to these nodes can be excised by cut $C_\rho$. In particular, the intermediate edges connecting literal-nodes in clause-gadgets to their corresponding variable-nodes should be left uncut.

Third, we argue that the three literal-nodes in a clause-gadget cannot be in the same subset of nodes, say $A$. Suppose that the literal-nodes corresponding to nodes labeled 1 and 9 in Figure 3.4 are in set $A$. Due to the same argument as variable-gadgets, the nodes that share a neighbor in the True and False-blocks with these nodes, i.e., the nodes labeled 2 and 8 in Figure 3.4, should lie in $B$. This results in the fact that the other two nodes connecting to each of these nodes, i.e., nodes labeled 3, 4, 6 and 7 in Figure 3.4, should also be in $B$. Given this condition, now if the literal-nodes corresponding to vertex labeled 5 in Figure 3.4 lies in $A$, then it has at least $\rho + 1$ neighbors outside its containing set, which is not admissible.

The above three properties force $C_\rho$ to partition the vertices of $G_\rho(\phi)$ into two sets $A$ and $B$ such that sets $A' = V \cap A$ and $B' = V \cap B$ are nonempty and partition vertices of $G(\phi)$ into one of the forms shown in Figures 3.5 and 3.6, without cutting through the True or False-blocks of $G(\phi)$. Therefore, if graph $G_\rho(\phi)$ has a relaxed-$\rho$-degree cut then $G(\phi)$ has a relaxed-1-degree cut and hence the relaxed-$\rho$-degree cut problem is **NP**-hard. □

Using the above lemma, we can now prove the following stronger result.

**Theorem 3.2** *For any $\rho \in \mathbb{Z}_{\geq 1}$, the $\rho$-degree cut problem is **NP**-hard.*

**Proof.** As we did in the **NP**-hardness proof of the 1-degree cut problem, for the $\rho$-degree cut problem we build graph $H_\rho(\phi)$ by taking $2\rho + 1$ copies of $G_\rho(\phi)$ and adding edges to construct complete subgraphs on all the nodes in the True and False-blocks of these $2\rho + 1$ copies. Again, each copy of $G_\rho(\phi)$ in $H_\rho(\phi)$ is called a box.

We prove that the $\rho$-degree cut problem is **NP**-hard by showing that $H_\rho(\phi)$ has a $\rho$-degree cut if and only if $G_\rho(\phi)$ has a relaxed-$\rho$-degree cut. It is not hard to see that if $G_\rho(\phi)$ has a relaxed-$\rho$-degree cut, then $H_\rho(\phi)$ has a $\rho$-degree cut and hence it remains to show the converse statement also holds. As in the proof of Theorem 3.1, here we assume that sets $A$, $B$ and $X$ partition the vertices of $H_\rho(\phi)$ such that $A$ and $B$ are nonempty. The difference

between this proof and the proof of Theorem 3.1 is that here each node in $A$ and $B$ has at most $\rho$ neighbors outside its set. We then show that there exist nonempty sets $A'$ and $B'$ partitioning the set of vertices of $G_\rho(\phi)$ such that no vertex has more than $\rho$ neighbors outside its containing set. It can be easily observed that the following generalized version of the two properties in the proof of Theorem 3.1 holds for cliques with at least $2\rho + 1$ nodes:

1. Set $X$ can contain either up to $\rho$ nodes or all nodes of the clique, and
2. If a node of the clique is in $A$ (resp. $B$), then no node of that clique is in $B$ (resp. $A$).

We denote the True and False-blocks of $H_\rho(\phi)$ by $T$ and $F$, respectively. As in the proof of Theorem 3.1, it can be argued that the only possible cases for $T$ and $F$ are $T \subseteq (A \cup X)$ and $F \subseteq (B \cup X)$, or $T \subseteq (B \cup X)$ and $F \subseteq (A \cup X)$. The analysis of the latter case is removed due to symmetry. Also, by property (1), either $T \subseteq X$ or $|T \cap X| \leq \rho$. Hence, either the True-blocks of all boxes in $H_\rho(\phi)$ are subsets of $X$, or there exist $\rho + 1$ boxes in $H_\rho(\phi)$ whose True-blocks are subsets of $A$. The same argument holds for the False-blocks of the boxes in $H_\rho(\phi)$. Consequently, there is a box in $H_\rho(\phi)$ whose True and False-blocks are subsets of $A$ or $X$, and $B$ or $X$, respectively. This box is denoted by $G'_\rho(\phi)$.

The case where both the True and False-blocks of $H_\rho(\phi)$ are subsets of $X$ is not possible, since it leaves sets $A$ and $B$ empty (by a similar argument as in the proof of Lemma 3.7). Therefore, the only possible cases for the True and False-blocks of $G'_\rho(\phi)$ are as follows: the True-block of $G'_\rho(\phi)$ is a subset of $X$ and its False-block is a subset of $B$, or the True-block of $G'_\rho(\phi)$ is a subset of $A$ and its False-block is subset of $B$. Note that we used the fact that sets $A$ and $B$ can be used interchangeably to reduce the number of possible cases.

In the first case, if the True-block of $G'_\rho(\phi)$ is a subset of $X$ and its False-block is a subset of $B$, then by the same argument as for the case that both the True and False-blocks of $H_\rho(\phi)$ were subsets of $X$ it can be shown that set $A$ becomes empty. Therefore, this case is not allowed.

In the second case, we have that the True-block of $G'_\rho(\phi)$ is a subset of $A$ and its False-block is a subset of $B$. We show that no node in $G'_\rho(\phi)$ can lie in set $X$. Suppose that a node in a variable-gadget of $G'_\rho(\phi)$ lies in $X$. Regardless of the set the other node in that variable-gadget is in, either the node in the True-block or the one in the False-block to which both these variable-nodes are connected has $\rho + 1$ neighbors outside its set, which is not allowed (see Figure 3.10 for $\rho = 2$).

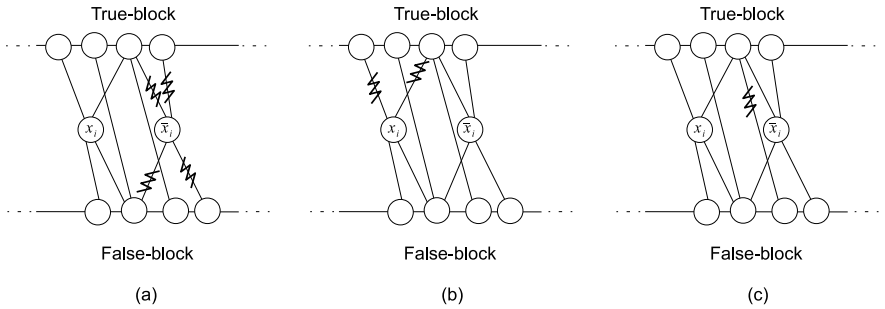



Figure 3.10: The True and False-blocks are subsets of $A$ and $B$, respectively, and the variable-node corresponding to $\bar{x}_i$ is in $X$. Figure (a) shows the edges incident to node $\bar{x}_i$ that are excised. In Figure (b), without loss of generality, it is assumed that node $x_i$ is either in set $B$ or $X$ and the edges that connect the vertices in $A$ to it are depicted. Figure (c) demonstrates another edge that connects a vertex in $A$ that is in the True-block to a vertex in $B$ that is in the False-block. It can be seen that the node in the True-block connected to both the nodes in the variable-gadget has three neighbors outside its set.

Moreover, since the True and False-blocks of $G'_\rho(\phi)$ are subsets of $A$ and $B$, respectively, each variable-node has $\rho$ neighbors outside its set and hence cannot afford to be connected to another vertex outside its set. Therefore, the edges connecting literal-nodes to their corresponding variable-nodes, i.e., the intermediate edges, cannot be excised. This in combination with the fact that no variable-node is in $X$ gives that no literal-node in any clause-gadget is in $X$. Also, by the same argument provided for variable-gadgets, it can be shown that the nodes labeled 2 and 8 in Figure 3.4 cannot lie in set $X$. Moreover, the nodes labeled 2, 3 and 4 in Figure 3.4 (resp. nodes labeled 6, 7 and 8) should lie in the same set; otherwise, node 2 (resp. node 8) has more than $\rho$ neighbors outside its set. Thus, no node in any clause-gadget can be in $X$. As a result, no node in $G'_\rho(\phi)$ is in $X$.

Thus, if $H_\rho(\phi)$ has a $\rho$-degree cut, then $G'_\rho(\phi)$ and hence $G_\rho(\phi)$ has a relaxed-$\rho$-degree cut, due to the fact that $G'_\rho(\phi)$ and $G_\rho(\phi)$ are isomorphic. Furthermore, sets $A'$ and $B'$ are defined as the subsets of nodes in $G_\rho(\phi)$ corresponding to the intersections of $A$ and $B$ with the set of nodes of $G'_\rho(\phi)$, respectively. Note that $A'$ and $B'$ are nonempty and partition the set of nodes of $G_\rho(\phi)$ and any node in $G_\rho(\phi)$ has at most $\rho$ neighbors outside its sets. Therefore, the converse statement is proved which concludes the **NP**-hardness of the $\rho$-degree cut problem. □

Knowing that the $\rho$-degree cut is **NP**-hard for any $\rho \in \mathbb{Z}_{\geq 1}$, it can be concluded that its complement problem, i.e., the $r$-robustness problem for any $r \in \mathbb{Z}_{\geq 2}$ is **coNP**-hard.

Combining this with the fact that the the $r$-robustness problem is in **coNP** gives that for any $r \in \mathbb{Z}_{\geq 2}$ the $r$-robustness problem is **coNP**-complete. Recall that earlier in this section it was argued that the complexity of the robustness problem is within a logarithmic factor of the complexity of the $r$-robustness problem. Hence, the following corollary can be concluded.

**Corollary 3.1** *The robustness problem is **coNP**-complete.*

## 3.2 Hardness of Approximation

Having established the complexity of the robustness problem, we now turn to the problem of *approximating* the degree of robustness of graphs. To show that a graph is not $r$-robust, one needs to find an $(r-1)$-degree cut in that graph. For any given graph $\mathcal{G}$, let $\text{OPT}(\mathcal{G})$ be the smallest nonnegative integer $\rho$ such that $\mathcal{G}$ has a $\rho$-degree cut. Suppose that we have a (polynomial-time) approximation algorithm ALG whose input is graph $\mathcal{G}$ and whose output $\text{ALG}(\mathcal{G})$ guarantees that graph $\mathcal{G}$ has an $\text{ALG}(\mathcal{G})$-degree cut. Define the *approximation ratio* $\alpha$ of the algorithm to be such that $\text{ALG}(\mathcal{G}) \leq \alpha \text{OPT}(\mathcal{G})$ for all graphs $\mathcal{G}$. Observe that since the $\rho$-degree cut problem is **NP**-hard, one cannot hope to reach an approximation ratio of $\alpha = 1$, unless $\mathbf{P} = \mathbf{NP}$. In the following lemma, we show that it is unlikely to find an approximation algorithm for this problem with approximation ratio less than 2.

**Lemma 3.8** *The problem of identifying the smallest $\rho$ such that a given graph has a $\rho$-degree cut is not approximable within any factor less than 2, unless $\boldsymbol{P} = \boldsymbol{NP}$.*

**Proof.** Assume that some polynomial-time algorithm ALG provides an approximation ratio $1 < \alpha < 2$ for all graphs $\mathcal{G}$. Consider the set of connected graphs that have a 1-degree cut; on any graph $\mathcal{G}$ from this set, the output of the algorithm must satisfy $\text{ALG}(\mathcal{G}) \leq \alpha \text{OPT}(\mathcal{G}) < 2$, i.e., $\text{ALG}(\mathcal{G}) = 1$. Thus, given any connected graph $\mathcal{G}$, the algorithm would output $\text{ALG}(\mathcal{G}) = 1$ if and only if the graph has a 1-degree cut, contradicting the fact that this is an **NP**-hard problem (as shown in Theorem 3.1). Therefore, an approximation ratio less than 2 is not obtainable for the $\rho$-degree cut problem unless $\mathbf{P} = \mathbf{NP}$. □

Note that the inapproximability result in Lemma 3.8 is a consequence of the two following facts: (i) parameter $\rho$ only takes integer values, and (ii) the problem can be solved

in polynomial time if the smallest value of $\rho$ for which the input graph has a $\rho$-degree cut is 0 and **NP**-hard if it is 1. Hence, this argument carries over for any other problem in which the feasible solutions are integers and there is some $a \in \mathbb{Z}$ such that the problem is efficiently solvable if the value of the solution is less than or equal to $a$, but **NP**-hard (or **coNP**-hard) for solutions with values greater than $a$.

# Chapter 4

# Conclusion

In this chapter, we review and conclude the studies discussed in this thesis. In Section 4.1, the main results obtained in this document are reviewed. Section 4.2 discusses the future research directions for the two problems studied in this thesis, i.e., the dominating set and the robustness problems.

## 4.1 Review

Here, we go through the main results obtained for each of the two problems discussed in this thesis.

### 4.1.1 Dominating Set Problem

In this thesis, we studied the dominating set and $k$-distance dominating set problems on $m \times n$ grids. A set $S \subseteq V$ constitutes a $k$-distance dominating set (which is equivalent to a dominating set if $k = 1$) for graph $G = (V, E)$ if for every vertex $v \in V \backslash S$ there is a vertex in $S$ whose distance from $v$ is at most $k$. We discussed a construction from [11] to obtain dominating sets for grids with near optimal size and generalized it to work in the $k$-distance domination scenario.

Moreover, we used these constructions in distributed algorithms and showed that the resulting dominating sets are upper-bounded by $\left\lceil \frac{(m+2k)(n+2k)}{2k^2+2k+1} + k \right\rceil$. The difference between the acquired upper-bound and the domination number of grid is at most five, for grids

with $16 \leq m \leq n$ and $k = 1$. However, via a more detailed case-based analysis in the grid corners, our distributed procedure can be used to obtain optimal dominating sets for $16 \leq m \leq n$.

### 4.1.2 Robustness Problem

Another problem discussed in this thesis was the robustness problem. Graph $G = (V, E)$ is $r$-robust if for any two nonempty and disjoint subsets of its vertices $A$ and $B$, at least one of them has a vertex that has at least $r$ neighbors outside its containing set. In the robustness problem, the goal is to find the largest $r$ for a graph $G$ such that $G$ is $r$-robust.

In Chapter 3, we showed that the robustness problem is **coNP**-complete. To do that, we introduced the $r$-robustness problem and the relaxed-$r$-robustness problem which are the decision version of the robustness problem and its relaxed version in which $B = V \backslash A$, respectively. We showed that both these problems are **coNP**-hard by proving that their complement problems, which we called the $\rho$-degree cut problem and the relaxed-$\rho$-degree cut problem, are **NP**-hard via a reduction from NAE3SAT. To be more specific, the $\rho$-degree cut problem determines whether there are two nonempty and disjoint subsets of vertices of a graph, $A$ and $B$, such that no vertex in $A$ or $B$ is connected to more than $\rho$ vertices outside its own set. On the other hand, the relaxed-$\rho$-degree cut problem determines whether there is a cut in the graph such that no vertex is incident with more than $\rho$ edges in the cut-set. In fact, the relaxed-$\rho$-degree cut problem is a generalization of a known **NP**-hard problem, called the "matching-cut problem", in which $\rho = 1$.

Furthermore, a short result on the hardness of approximation of the problem of determining the smallest $\rho$ for which the graph has a $\rho$-degree cut was discussed in Section 3.1. We showed that for this problem an approximation algorithm with guarantee less than 2 is not obtainable, unless $\mathbf{P} = \mathbf{NP}$.

## 4.2 Future Directions

In this section, we propose some future directions for the dominating set problem on grids and the robustness problem.

### 4.2.1 Dominating Set Problem

There are many open problems that can be considered for grid domination. The complexity of $k$-distance dominating grids and the $k$-domination number of grids are still unknown. Moreover, in Section 2.3, we provided an upper-bound on the number of vertices to $k$-distance dominate an $m \times n$ grid obtained by $k$-diagonalization and $k$-projection, i.e., $\left\lceil \frac{(m+2k)(n+2k)}{N^k_{\max}} + k \right\rceil$, where $N^k_{\max} = 2k^2 + 2k + 1$. However, a tighter upper-bound on the number of $k$-dominating vertices used by $k$-diagonalization and $k$-projection was obtained by simulation, which is $\left\lceil \frac{(m+2k)(n+2k)}{N^k_{\max}} + \frac{k-1}{2} \right\rceil$. This upper-bound is still open to be proved.

It is also of interest to find centralized and distributed algorithms for dominating subgraphs of grids, that is, grids with some of their vertices or edges missing. Generalizing these algorithms to the cases where the underlying graphs are cubic or hyper-cubic grids is another direction of this research.

### 4.2.2 Robustness Problem

Although we showed that there exists no approximation algorithm with guarantee less than 2 for determining the smallest $\rho$ for which a graph $G$ has a $\rho$-degree cut, we do not know whether this bound is tight. Moreover, it is of interest to find approximation algorithms for this problem.

# References


[1] W. Abbas and M. B. Egerstedt. Securing multiagent systems against a sequence of intruder attacks. In *American Control Conference*, Montreal, Canada, June 2012.

[2] S. Alanko, S. Crevals, A. Isopoussu, and V. Pettersson. Computing the domination number of grid graphs. *The Electronic Journal of Combinatorics*, 18(P141):1, 2011.

[3] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[4] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.

[5] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, 18(6):911–922, 2002.

[6] J. Blum, M. Ding, A. Thaeler, and X. Cheng. Connected dominating set in sensor networks and manets. *Handbook of Combinatorial Optimization*, pages 329–369, 2005.

[7] J.J.A. Bondy and U.S.R. Murty. *Graph Theory*. Graduate Texts in Mathematics Series. Springer London, 2008.

[8] P. Bonsma. The complexity of the matching-cut problem for planar graphs and other graph classes. *Journal of Graph Theory*, 62(2):109–126, 2009.

[9] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009.

[10] M. Cardei and J. Wu. Coverage in wireless sensor networks. *Handbook of Sensor Networks*, pages 422–433, 2004.

[11] T. Y. Chang. *Domination numbers of grid graphs*. PhD thesis, University of South Florida, 1992.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[13] G. Di Battista, M. Patrignani, and F. Vargiu. A split&push approach to 3d orthogonal drawing. In *Graph Drawing*, pages 87–101. Springer, 1998.

[14] M. Dorfling and M. A. Henning. A note on power domination in grid graphs. *Discrete Applied Mathematics*, 154(6):1023 – 1027, 2006.

[15] D. Easley and J. Kleinberg. *Networks, crowds, and markets*, volume 8. Cambridge Univ Press, 2010.

[16] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences. W. H. Freeman, 1979.

[17] D. Gonçalves, A. Pinlou, M. Rao, and S. Thomassé. The domination number of grids. *SIAM Journal on Discrete Mathematics*, 25(3):1443–1453, 2011.

[18] E. Guizzo. Three engineers, hundreds of robots, one warehouse. *IEEE Spectrum*, 45(7):26–34, July 2008.

[19] M. A. Henning. Distance domination in graphs. In *Domination in Graphs: Advanced Topics*, pages 321–349. Marcel Dekker, New York, 1998.

[20] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger. *Dissemination of Information in Communication Networks*. Springer-Verlag, 2005.

[21] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3):256–278, 1974.

[22] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. Available at `http://planning.cs.uiuc.edu`.

[23] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram. Resilient asymptotic consensus in robust networks. *Selected Areas in Communications, IEEE Journal on*, 31(4):766–781, 2013.

[24] N. E. Leonard, D. Paley, F. Lekien, R. Sepulchre, D. M. Fratantoni, and R. Davis. Collective motion, sensor networks and ocean sampling. *Proceedings of the IEEE*, 95(1):48–74, 2007.

[25] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *International conference on Mobile computing and networking*, Boston, MA, 2000.

[26] B. Liu and D. Towsley. On the coverage and detectability of large-scale wireless sensor networks. In *WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[27] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 13(4):383–390, 1975.

[28] N. A. Lynch. *Distributed Algorithms*. Elsevier (imprint: Morgan Kaufmann), 1996.

[29] M. Patrignani and M. Pizzonia. The complexity of the matching-cut problem. In *Graph-Theoretic Concepts in Computer Science*, pages 284–295. Springer, 2001.

[30] M. Patrignani and F. Vargiu. 3dcube: A tool for three dimensional graph drawing. In *Graph Drawing*, pages 284–290. Springer, 1997.

[31] G. Piovan, I. Shames, B. Fidan, F. Bullo, and B. D. O. Anderson. On frame and orientation localization for relative sensing networks. In *IEEE Conf. on Decision and Control*, pages 2326–2331, Cancún, México, December 2008.

[32] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 1997.

[33] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, 1978.

[34] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterative strategies in the presence of malicious agents. *IEEE Transactions on Automatic Control*, 56(7):1495–1508, 2011.

[35] V.V. Vazirani. *Approximation Algorithms*. Springer, 2004.

[36] D.P. Williamson and D.B. Shmoys. *The Design of Approximation Algorithms*. The Design of Approximation Algorithms. Cambridge University Press, 2011.

[37] J. Wu, M. Gao, and I. Stojmenovic. On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. In *International Conference on Parallel Processing*, pages 346 –354, 2001.

[38] H. Zhang and S. Sundaram. Robustness of complex networks with implications for consensus and contagion. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 3426–3432. IEEE, 2012.