LOCATING SINKS IN WIRELESS SENSOR NETWORKS

by

Louisa Harutyunyan

B.Comp.Sci., Concordia University, 2007

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Summer 2009

APPROVAL

Name: Louisa Harutyunyan

Degree: Master of Science

Title of Thesis: Locating Sinks in Wireless Sensor Networks

Examining Committee: Dr. Binay Bhattacharya
Chair

Dr. Arthur Liestman, Professor, Computing Science
Simon Fraser University
Senior Supervisor

Dr. Joseph Peters, Professor, Computing Science
Simon Fraser University
Senior Supervisor

Dr. Qianping Gu, Professor, Computing Science
Simon Fraser University
SFU Examiner

Date Approved: August 6, 2009

Declaration of
Partial Copyright Licence

Abstract

We study the problem of positioning *sinks*, or data collection stops, in wireless sensor networks. To reduce the path lengths from sensors to sinks we introduce multiple sinks. We find a group of sinks where every sensor is within distance k of at least p sinks. We model the wireless sensor network as a *unit disk graph* $G = (V, E)$ and find a distance-k total p-dominating set $S \subseteq V$ for fixed positive integers k and p. If we place sinks at the positions of the vertices of S, then every sensor is within distance k of p sinks.

To find a distance-k total p-dominating set of minimum size is NP-hard. We give $2(2k+1)^2$ and $p \cdot ln\Delta_k$ approximation algorithms, where Δ_k is the largest cardinality k-neighborhood. We propose several greedy based heuristics and conduct several experiments to compare the performance of our algorithms. We give a statistical performance analysis for our experimental results.

Keywords: sink; unit disk graph; distance-k total p-dominating set

To my family!

Acknowledgments

I would like to sincerely thank my senior supervisors Joe Peters and Art Liestman for their help, support and guidance and for allowing me to pursue my own interests. Special thanks to Dr. Thomas Loughin and Dr. Debaraj Sen for their helpful discussions on the statistical analysis of this thesis. I would like to thank Dr. Qianping Gu for being my thesis examiner and for his helpful comments and suggestions.

Most importantly, I would like to thank my family for giving me their constant love and encouragement over the years. Without them I would not be here today. Thank you for your endless support and understanding throughout this work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

A *sensor* is a small device that collects data. The data may be of a physical nature such as light intensity, temperature, sound, or proximity to objects. A sensor has memory capacity for storing the collected data and energy for communication and processing data. A sensor is equipped with a radio transceiver for communicating with other devices and a battery as its energy source. In some applications the energy of a sensor may be collected from the environment by solar cells.

The size and cost of a sensor puts constraints on the memory and energy resources as well as on the communication and processing capabilities of a sensor. The size of a sensor may vary. For example, a weather station is a large sensor since it consists of several meteorological sensors such as an anemometer for measuring wind speed, a wind direction sensor, a barometer, a rain sensor, a humidity sensor and a temperature sensor. On the other hand, sensors used in military surveillance should be as small as possible so as not to be seen. The cost of a sensor is similarly variable ranging from a few cents to hundreds of dollars, depending on the size and complexity of the sensor. The larger the size of the sensor, the greater its storage capacity, and the higher its computation speed and communication bandwidth. Therefore, energy and other resources available to a sensor vary from application to application. The computational power of a sensor or a group of sensors is affected by the sensors' resource constraints.

A group of sensors can be used over a given region to collect data and communicate it to each other. A *wireless sensor network* is a large number of sensor nodes spatially

distributed over a geographical region to cooperatively collect data for monitoring physical or environmental conditions. The number of sensors in the network is determined by the requirements of the network connectivity, coverage, and the size of the area of interest being monitored. Hence, the network size may vary from a few sensors to thousands of sensors depending on the application.

Sensors in a wireless network communicate among themselves using radio transceivers. Each sensor node has a *transmission range*, the maximum distance it can transmit data. A single radio transmission of a sensor node can be received by all of its neighbors within that range. Two sensors can communicate directly if they are within each other's transmission ranges. Sensors that are further away from each other may communicate by sending messages through intermediate sensor nodes. We will refer to the minimum number of transmissions required to send a message from a sensor node u to a sensor node v as the *distance* from u to v.

Some sensors in a wireless sensor network are designated as *sink* nodes to which other sensors send their data. A wireless sensor network may have one or more sink nodes. In general, sinks do much more computation than other sensors. They inspect and manipulate the collected data (e.g. aggregating similar data or filtering redundant information) and communicate it to a central unit for processing.

The central unit is a gateway between the sensor network and the user. A sensor does not communicate collected data directly to the user. A node's primary focus of communication is interaction with other nodes, not delivery of data to the user. Due to the large number of nodes in sensor networks, the user does not manage the flow of information. Users are not aware of every datum and computation done by each sensor. Instead they are informed only of highest level conclusions or results. Hence the goal of the network is not to provide a complete record of every sensor's reading as raw data, but rather to perform a synthesis that provides higher level information.

Communication in a wireless sensor network is affected by network properties such as the network topology and whether the network is connected. Sensors communicate directly with their neighbors or through intermediate neighboring nodes. There are several ways to transfer data from a source node to a destination node. A simple way to transfer data from a source to a destination is for the source node to send the data to all of its neighbors, which in return send the piece of data to their neighbors. This process is continued until the data reaches the destination node. This method is known as *flooding* the network. Flooding is

not an efficient way of communicating information within the sensor network. It is costly in terms of bandwidth and energy consumption at the nodes not on the shortest path from the source to the destination. Flooding causes data redundancy. It may also cause duplicate data packets to circulate around the network forever unless some precautions are taken against it.

To reduce the number of nodes used for transferring data from the source node to the destination node as well as to reduce data redundancy, routing tables can be used. Each sensor node keeps a routing table consisting of the nodes on the route between itself and the destination node. Data is sent from a source node to a destination node using the appropriate neighbor at each intermediate node. It is clearly more efficient to route data using routing tables instead of flooding the network. Often routing tables are built according to a predefined structure such as a tree or a set of connected stars. Hence, the topology of the network and the choice of routing tables affects data routing as does the *latency*, the time it takes for data to travel from one node to another.

Another important issue is to have a *connected* network. A *connected* network is defined by the transmission range, the network density, and the physical location of each sensor. A wireless sensor network is *connected* if there is a path between any two sensors either directly or through other intermediate sensors. It is crucial for some applications that the network is not partitioned into disjoint connected components. A connected network facilitates the development of guidelines regarding the design and operation of sensor networks, such as communication protocols and methods for data gathering. Over time, the network may become disconnected due to battery failure of sensor nodes, software bugs, or because of the physical environment surrounding them.

Over time, the structure of the network changes and the network degrades. One of the causes for this degradation is sensor batteries running out of power. Sensors may be deployed in an area that is not accessible or where recharging batteries may not be feasible. Failure of sensors may result in a disconnected network, which has a direct impact on the routing of data between sensor nodes and on the lifetime of the network. Extending the lifetime of a sensor network is an important issue when the goal is to prolong the functionality of such a network. There exist different definitions for the lifetime of a network. One definition is the time that the first sensor node in the network fails because it is out of energy. Another definition extends the previous definition: the lifetime of a network is the time during which the network stays connected.

In sensor networks, communication is expensive in terms of energy, limited in bandwidth, and nontrivial in terms of routing. Each data transmission by a sensor node consumes energy. In a sensor network with one sink node, all sensors transmit their data to the sink. Sensor nodes that are neighbors of the sink, directly send their data to the sink. However, sensors that are not direct neighbors of the sink send their data through other neighboring nodes. Sensor nodes that are far away from the sink, deplete energy at all of the intermediate nodes on the path to the sink when they send data to the sink. Sensor nodes close to the sink have their energy depleted due to forwarding data on behalf of other nodes. Hence, they are likely to run out of energy sooner than other nodes. Since each data transmission consumes energy, by decreasing the number of transmissions made by the sensors, we can decrease the use of the batteries and prolong network functionality. If we limit the distance each piece of data travels to get to a sink, we can have a significant savings in energy.

A network with only one sink is prone to failure. The sink gathers data from other sensors. In case of sink failure, the network will cease to function properly. To address the problem of sink failure, we introduce more sinks into the network. However, we still may have sinks clustered together in one area of the network. Hence, remote sensor nodes that send data to a sink cause the network to deplete energy at all intermediate nodes on the path to the sink. To decrease the distance from any sensor to a sink we limit the distance each piece of data travels to reach more than one sink. This suggests the problem of finding a group of sinks such that every node is within distance k of at least p sinks.

1.2 Problem Statement

In the previous section, we introduced the problem of finding a group of sinks such that every sensor node is within distance k of at least p sinks. To address this problem formally, we need some definitions. All definitions are obtained from two books on domination in graphs by Haynes, Hedetniemi and Slater [16], [17].

Consider a wireless sensor network consisting of n wireless sensor nodes distributed over a two dimensional plane. Assume that all sensors have distinct identities (denoted ID hereafter). Assume the maximum transmission ranges of all sensor nodes are identical. Two sensors can communicate with each other if they are within each other's transmission range. We represent the sensor network as a *unit disk graph*, where there is an edge between two nodes if and only if their Euclidean distance is at most one unit. This corresponds to the

sensors being within the transmission range of each other. More formally, the network is represented by graph $G = (V, E)$, where V is the set of sensor nodes and E is the edge set described above. We denote an edge between two vertices u and v as (u, v). For any $(u, v) \in E$, we say u and v are adjacent.

The minimum number of transmissions required to send a piece of data from a vertex u to a vertex v is the *distance* from u to v denoted as $d(u, v)$.

The *open neighborhood* $N(v)$ of the vertex v consists of the set of vertices adjacent to v, that is, $N(v) = \{w \in V | (v, w) \in E\}$. The *closed neighborhood* of v is the set $N[v] = N(v) \cup \{v\}$.

The *open k-neighborhood* of a vertex $v \in V$, denoted $N_k(v)$, is the set $N_k(v) = \{u | u \neq v$ and $d(u, v) \leq k\}$. The set $N_k[v] = N_k(v) \cup \{v\}$ is called the *closed k-neighborhood* of v. Every vertex $w \in N_k[v]$ is said to be *k-adjacent* to v.

A *dominating set* of a graph $G = (V, E)$ is a set $S \subseteq V$ such that for every $v \in V \backslash S$, there exists a vertex $u \in S$ such that v is adjacent to u. We say vertices of the dominating set S *dominate* the entire vertex set V, where each vertex $u \in S$ dominates its closed neighborhood. A *minimum dominating set* of graph G is a dominating set of G such that its cardinality is the smallest among all dominating sets of G. A minimum dominating set is not necessarily unique for a given graph. We show two minimum dominating sets for a particular graph in Figure 1.1. The first set is shown by the black nodes, $\{v_3, v_6\}$, and the second by the circled nodes $\{v_2, v_6\}$.

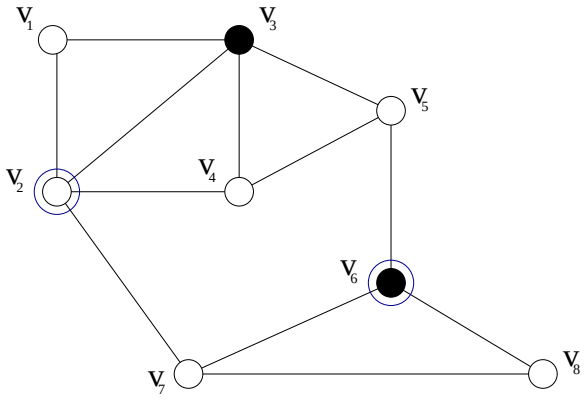


Figure 1.1: $\{v_3, v_6\}$ and $\{v_2, v_6\}$ are minimum dominating sets.

There are different variations of the dominating set, six of which we define here. For all of these variations we are interested in finding sets of minimum cardinality.

One generalization of the dominating set is the *p-dominating set*, which may be referred to as *multiple domination*. For a given positive integer p, a *p-dominating set* is a set $S \subseteq V$ such that for every vertex $v \in V \backslash S$, v has at least p adjacent vertices in S. Dominating set is special case of p-dominating set where $p = 1$. A minimum 2-dominating set is given by the black vertices in Figure 1.2.

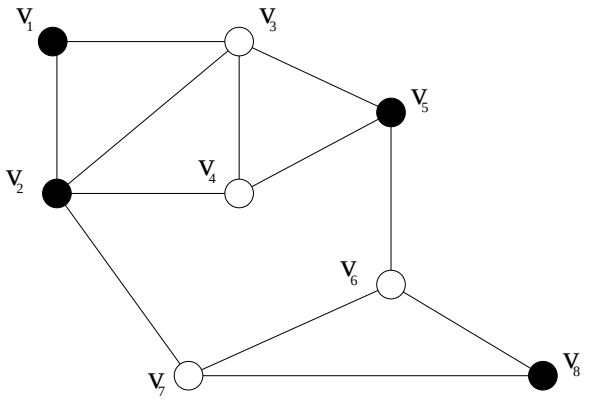


Figure 1.2: $\{v_1, v_2, v_5, v_8\}$ is a minimum 2-dominating set.

A variation of the p-dominating set is the *p-tuple dominating set*. For a given positive integer p, a *p-tuple dominating set* for graph G is a set $S \subseteq V$ such that every vertex $v \in V$ is dominated by at least p vertices in S, where each vertex dominates its closed neighborhood. A dominating set is a special case of the p-tuple dominating set when $p = 1$. The difference between a p-dominating set and a p-tuple dominating set is that with p-domination the vertices in $V \backslash S$ are the only ones that must be multiply dominated, while in p-tuple domination every vertex in V must be multiply dominated. An example of minimum p-tuple dominating set is shown in Figure 1.3.

For a fixed positive integer k, a *distance-k dominating set* is a set $S \subseteq V$ such that for every vertex $v \in V \backslash S$, $d(v, S) \leq k$, where $d(v, S)$ is the minimum distance between vertex v and a vertex $u \in S$. A special case of the distance-k dominating set is when $k = 1$, that is, when every vertex $v \in V \backslash S$ is adjacent to at least one vertex of S. In other words, dominating set is a special case of distance-k dominating set. Figure 1.4 shows an example of a distance-2 dominating set, where v_5 dominates its closed 2-neighborhood. Note that a minimum distance-k dominating set is not necessarily unique. In Figure 1.4, the sets $\{v_2\}$ and $\{v_7\}$ are also minimum distance-2 dominating sets.

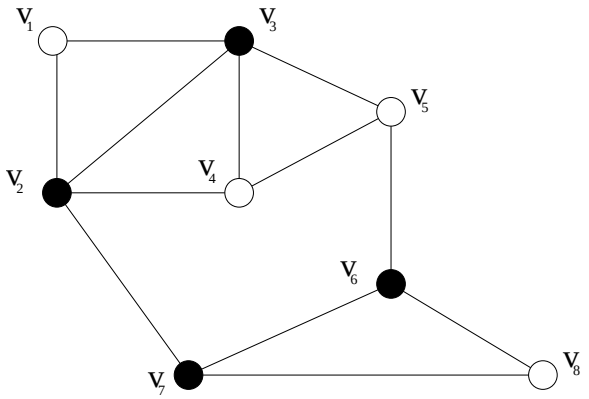


Figure 1.3: $\{v_2, v_3, v_6, v_7\}$ is a minimum 2-tuple dominating set.

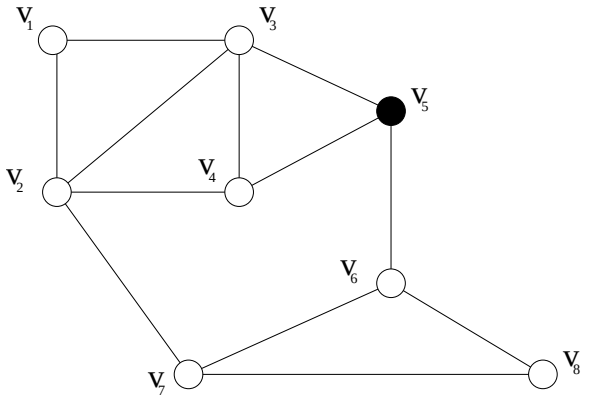


Figure 1.4: $\{v_5\}$ is a minimum distance-2 dominating set.

A *total (open) dominating set* is a set $S \subseteq V$ such that for every $v \in V$, there exists a vertex $u \in S$, $u \neq v$, such that u is adjacent to v. A total (open) dominating set is commonly referred as total dominating set. Note that for total (open) domination all vertices in V must be adjacent to at least one vertex in S. By contrast, with dominating set only vertices in $V \backslash S$ are required to be adjacent to at least one vertex in S. Figure 1.1 shows that $\{v_3, v_6\}$ is a minimum dominating set S for G. However, this is not a total (open) dominating set since the vertices in the dominating set, v_3 and v_6, are not adjacent to any vertex in S. An example of a minimum total dominating set is given in Figure 1.5. Note again, that there are two minimum total dominating sets, $\{v_3, v_5, v_6\}$ and $\{v_2, v_6, v_7\}$.

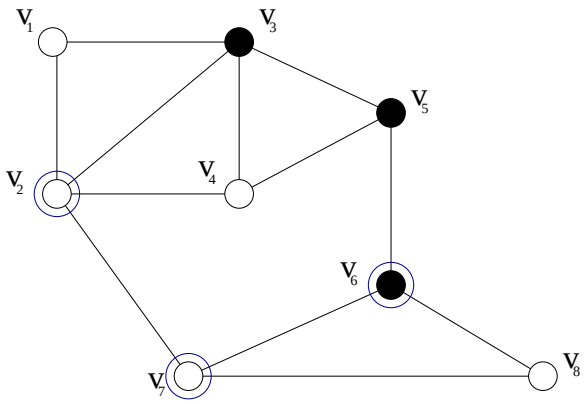


Figure 1.5: $\{v_3, v_5, v_6\}$ and $\{v_2, v_6, v_7\}$ are minimum total (open) dominating sets.

For any positive integer p, a *total (open) p-dominating set* is a set $S \subseteq V$ such that for every $v \in V$, v is adjacent to at least p vertices in S. Total (open) p-dominating set is also referred to as total p-dominating set. When $p = 1$, the total (open) 1-dominating set is simply a total (open) dominating set. Total (open) p-dominating set is also a variation of the previously defined t-tuple dominating set. The difference between the two definitions is that for the t-tuple dominating set, each vertex dominates its closed neighborhood. Hence, it is not the same as total (open) p-dominating set in which each vertex dominates its open neighborhood (i.e each vertex $v \in V$ must be adjacent to at least p vertices in S not including itself). An example of a total (open) 2-dominating set is shown in Figure 1.6, where all vertices are dominated by at least two vertices. A *connected total p-dominating set* S is a total p-dominating set S whose induced subgraph is connected. The total-2 dominating set in Figure 1.6 is connected.

The *distance-k p-dominating set* is a combination of two previously defined problems, distance-k dominating set and p-dominating set for positive integers k and p. A *distance-k p-dominating set* is a set $S \subseteq V$ such that each vertex $v \in V \backslash S$ is within distance k of p vertices of S. The distance-k p-dominating set is simply a dominating set when $k = p = 1$. An example of a minimum distance-2 2-dominating set is given in Figure 1.7.

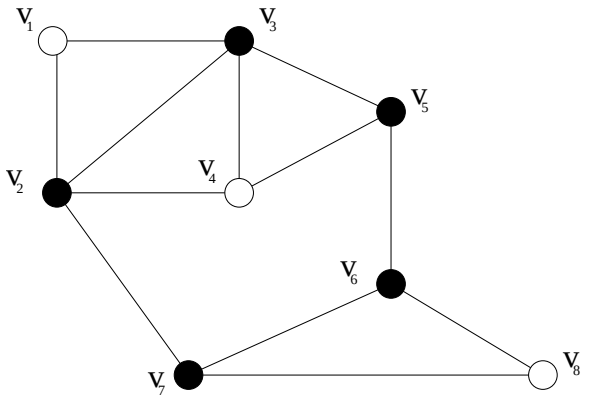


Figure 1.6: $\{v_2, v_3, v_5, v_6, v_7\}$ is a minimum total (open) 2-dominating set.

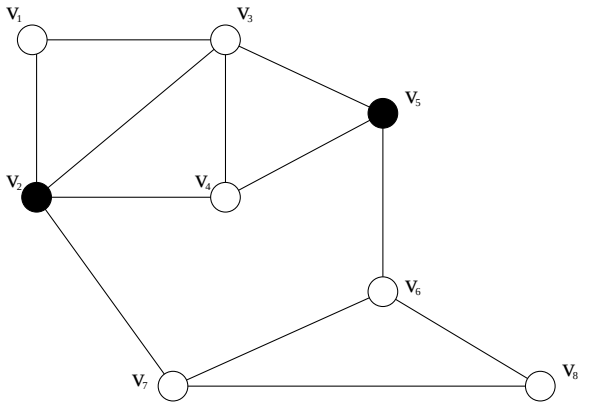


Figure 1.7: $\{v_2, v_5\}$ is a minimum distance-2 2-dominating set.

Now, we introduce a combination of distance-k and total (open) p-dominating sets called *distance-k total (open) p-dominating set*. For positive integers k and p, a *distance-k total (open) p-dominating set*, denoted as $D_{k,p}$, is a set $S \subseteq V$ such that each vertex $v \in V$ is within distance k of p vertices of S not including v itself. A *minimum* $D_{k,p}$ for the graph G is a $D_{k,p}$ of G such that its cardinality is the smallest among all $D_{k,p}$ of G. A distance-k total (open) p-dominating is also referred to as distance-k total p-dominating set. A special case of $D_{k,p}$ occurs for $k = p = 1$. $D_{1,1}$ is simply a total (open) dominating set, so total (open) dominating set is a special case of $D_{k,p}$. An example of a minimum $D_{2,2}$ is shown in Figure 1.8. Note that v_6 is part of $D_{2,2}$ since both v_2 and v_5 need to be within distance two of two vertices in $D_{2,2}$.

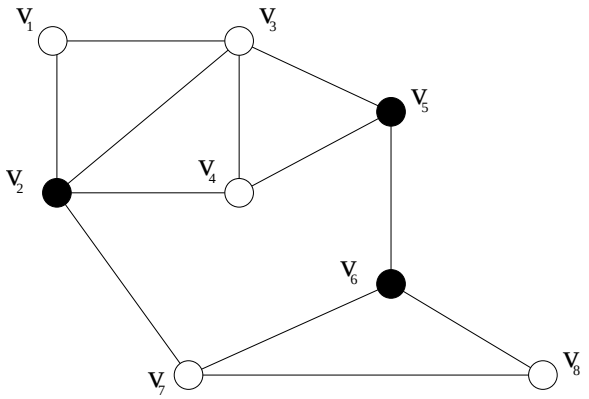


Figure 1.8: $\{v_2, v_5, v_6\}$ is a minimum distance-2 total (open) 2-dominating set.

We show the relationships among the various types of dominating sets and their special cases in Figure 1.9. The legend of Figure 1.9 is explained as follows. The two vertical bars represent equivalence between two sets. The lines with black arrow heads indicate that a set at the tail of the arrow is composed of the combination of two sets at the head of the black arrow. The lines with the white arrow head indicate that the set at the head of the arrow is a special case of the set at the tail of the arrow. In Figure 1.9 this has two meanings. For example, when $k = 1$, dominating set is a base case of distance-k domination set. Similarly, a total dominating set is a special case of total p-dominating set. However, with white arrow heads we also indicate that one set is a restricted version of another. For example, a p-tuple dominating set is a p-dominating set. However, the converse is not true. Similarly, a distance-k total p-dominating set is a distance-k p-dominating set.

In the previous section we proposed the following problem: place a group of sinks in a wireless sensor network such that every sensor node is within distance k of at least p sinks. More formally, find a distance-k total (open) p-dominating set in a unit disk graph $G = (V, E)$.

A related concept of interest is the *independent set*. An *independent set* of a graph $G = (V, E)$ is a set $S \subseteq V$ such that for any pair of vertices u, $v \in S$, $(u, v) \notin E$. We are interested in finding independent sets of maximum cardinality. A *maximal independent set* of G is an independent set $S \subseteq V$ such that S is not a subset of any other independent set of G. A *maximum independent set* of G is an independent set of G such that its cardinality is largest among all independent sets of G. A maximum independent set is not necessarily unique. An example of a maximum independent set is given in Figure 1.10.

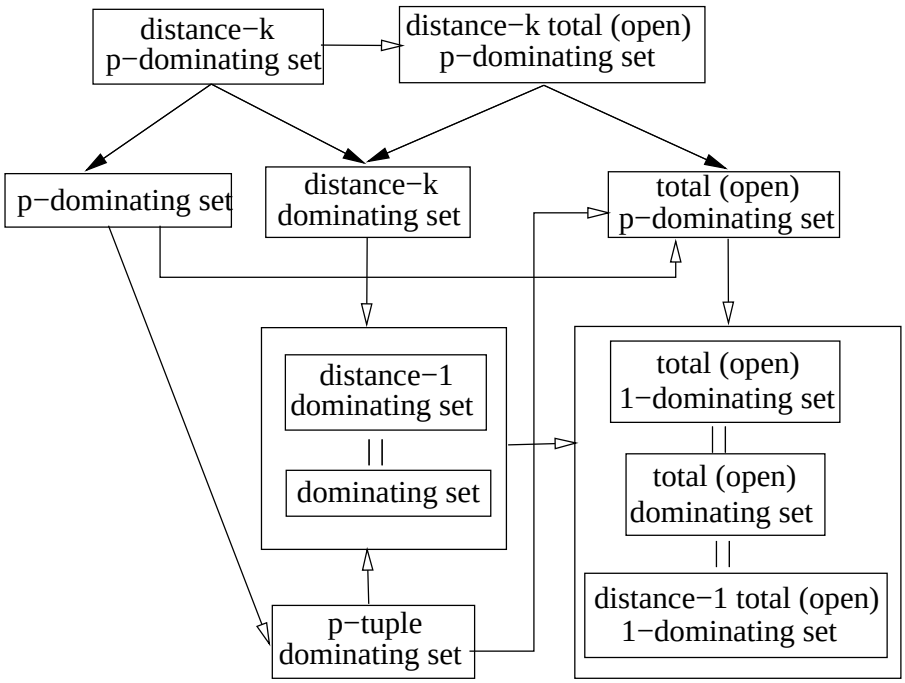


Figure 1.9: Various dominating sets relationships.

A maximal independent set of graph G is also a dominating set of G. One way to find a dominating set for G is to find an independent set of G. One would like to find a dominating set of minimum cardinality. However, when finding an independent set, we are interested in a maximum sized set of nodes. Therefore, the problem of finding a *maximal independent set* is of much more interest than that of finding a maximum independent set, since the size of a *maximal independent set* may be smaller than that of a maximum independent set.

A generalization of the independent set is the *distance-k independent set*. For a positive integer k, a *distance-k independent set* is a subset of vertices $S \subseteq V$ such that for any pair of vertices u, $v \in S$, $d(u,v) \geq k+1$. A set S is a *maximal distance-k independent set* of G, denoted MIS_k, if S is not a subset of any other distance-k independent set of G. It can be shown that a maximal distance-k independent set is a distance-k dominating set. An example of maximum distance-2 independent set is shown in Figure 1.11.

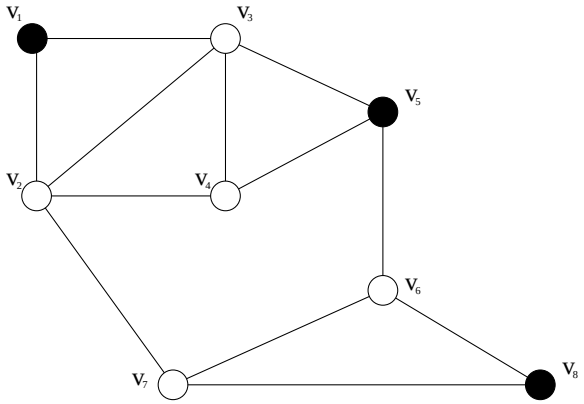


Figure 1.10: $\{v_1, v_5, v_8\}$is a maximum independent set.

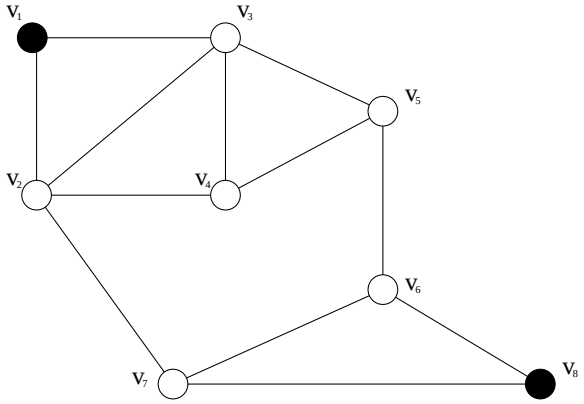


Figure 1.11: $\{v_1, v_8\}$ is a maximum distance-2 independent set.

1.3 Outline of Thesis

In Chapter 2 we give a survey of several results in the literature for the various types of dominating sets discussed in the previous section. Chapter 3 gives several approximation algorithms and heuristics to find a distance-k total p-dominating set for a given graph G. We give the performance ratios for our approximation algorithms and the complexity analysis for the heuristics. The effects of various variables on the size of the distance-k total p-dominating set are analyzed for three of the algorithms in Chapter 4 using a multiple regression model. In Chapter 5 we analyze and compare the performance of one of the heuristics and two of the approximation algorithms. Finally, in Chapter 6 we conclude this thesis and discuss possible future work.

Chapter 2

Related Work

The study of dominating sets dates back to 1862 when de Jaenisch [20] studied the problem of determining the minimum number of queens which are necessary to cover (or dominate) an $n \times n$ chess board.

The mathematical study of dominating sets began around 1960. In 1958 Berge [2] wrote a book on graph theory in which he defined the concept of the domination number of a graph. He called the domination number the *coefficient of external stability*. In 1962 Ore [36] published a book on graph theory, in which he used, for the first time, the terms *dominating set* and *domination number*. Cockayne and Hedetniemi in 1977 published a survey of known results about dominating sets in graphs [8]. During the last few decades the area has vastly grown.

The decision problem for the dominating set can be stated as follows.
DOMINATING SET
INSTANCE: A graph $G = (V, E)$ and a positive integer k
QUESTION: Does G have a dominating set of size $\leq k$?

Garey and Johnson showed that DOMINATING SET is NP-complete for arbitrary graphs [14]. However, it is solvable in polynomial time in trees [7]. Garey and Johnson also show that the connected dominating set is NP-complete [14]. Since the dominating set is NP-complete, its generalizations are also NP-hard. Clark et al. have shown that the dominating set and the connected dominating set are NP-complete in unit disk graphs [6]. Therefore, we develop heuristics to find dominating sets for a given graph G. A heuristic for

which we can give a bound for the size of the solution returned is an approximation algorithm. An approximation algorithm finds a near-optimal solution in polynomial time. An approximation scheme is an approximation algorithm that takes as input an instance of the problem and a value $\epsilon > 0$ such that for any fixed ϵ, the scheme is a $(1 + \epsilon)$-approximation algorithm. An approximation scheme is a polynomial-time approximation scheme ($PTAS$) if for any fixed $\epsilon > 0$, the scheme runs in time polynomial in the size n of its input instance.

2.1 Dominating Set Algorithms

There are various algorithms that find dominating sets for a given graph G. Some are approximation algorithms while others are heuristics. We discuss some of the algorithms that have appeared in the literature in this section.

2.1.1 Dominating Sets in General Graphs

Consider a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges.

Parekh introduced a greedy algorithm to find small dominating sets in undirected graphs of n vertices [37]. This greedy algorithm is an analog of an algorithm of Chvatal [4] for finding set covers. Since any dominating set problem can be formulated as a set covering problem, the results of the set covering problem can be specialized to the dominating set problem. The algorithm is as follows.

Let $G = (V, E)$ where V is a set of n vertices and E is a set of edges. Let D be a dominating set of graph G. A vertex u is said to be covered if $u \in D$ or if u is adjacent to a vertex $v \in D$. A vertex that is not covered is said to be uncovered. Initially define $D = \emptyset$. In each iteration of the algorithm, the uncovered vertex of least index that covers the maximum number of uncovered vertices is added to the dominating set. This process stops once all vertices are covered. The cardinality of the dominating set, d_g, returned by the algorithm is bounded by $d_g \leq n + 1 - \sqrt{2|E| + 1}$.

Based on the above greedy algorithm, Sanchis gave several heuristics to find dominating sets in general graphs [39]. We briefly discuss one of these heuristics called *GreedyVote*. GreedyVote operates similarly as Parekh's greedy algorithm. However, GreedyVote also pays attention to the specific vertices which would be covered, and to whether or not they could be covered in some alternative way. For example, if a vertex v would cover a vertex u if v were added to the dominating set, and u has low degree then v has a stronger reason for

being added to the dominating set than it would if u had high degree and may therefore be covered by many other vertices. An extension of GreedyVote is the *GreedyVoteGr* algorithm in which a local search procedure is added [39]. The search determines whether it is possible to remove any two vertices from the dominating set and replace them with either one or no vertices while still retaining a dominating set. Sanchis gave several experimental results comparing GreedyVote, GreedyVoteGr and other greedy based heuristics with the regular Greedy algorithm. The experimental results determined that GreedyVoteGr outperforms other heuristics.

2.1.2 Dominating Sets in Unit Disk Graphs

Nieberg and Hurink gave a polynomial time approximation scheme for the minimum dominating set problem in unit disk graphs with a $(1+\varepsilon)$ performance ratio [34]. Before discussing their technique, we define some terms.

Consider a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. We extend the definition of closed k-neighborhood from a vertex to a subgraph $G' = (V', E')$ of G: $N(V') : \bigcup_{w \in V'} N[w]$.

Let $\mathcal{P}(V)$ denote the set of all subsets of vertices. Define a function $D : \mathcal{P}(V) \rightarrow \mathcal{P}(V)$ which returns a dominating set of minimum cardinality for the subset given as the argument. For example, $D(V')$ dominates V' where $V' \subseteq V$. Note that the following property holds true: $V' \subset N(D(V'))$ and $D(V') \subset N(V')$. We show an example of this property in Figure 2.1.

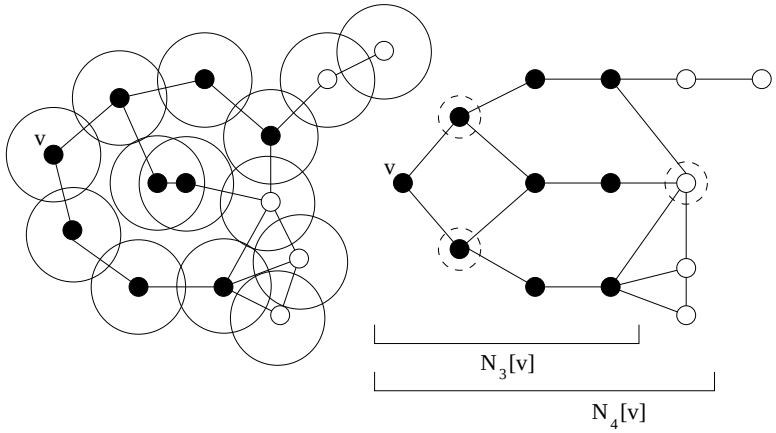


Figure 2.1: $D(V') \subset N(V')$, where V' is $N_3[v]$. $D(V')$ is in $N_4[v]$.

The technique of Nieberg and Hurink is based on finding dominating sets of subgraphs of G. Hence, they introduce the concept of a *2-separated collection* of subsets, $S = \{S_1, \cdots, S_k\}$. S is defined as a collection of subsets of vertices $S_i \subset V$ for $i = 1, \cdots, k$ such that for any two vertices $s \in S_i$, $s' \in S_j$, and $i \neq j$, $d(s, s') > 2$. This is illustrated in Figure 2.2 where all vertices in one subgraph are at least distance three from all other vertices in another subgraph. The dominating set of one subgraph does not overlap the dominating set of another subgraph since $D(V') \subset N(V')$ for any subgraph G'. It can be proved that for a 2-separated collection $S = S_1, \cdots, S_k$ in a graph $G = (V, E)$, we have $|D(V)| \geq \sum_{i=1}^{k} |D(S_i)|$. This provides a lower bound on the minimum dominating set of the graph $G = (V, E)$. A 2-separated collection can also be used to find an approximate solution. The idea is to enlarge each subgraph S_i to obtain another subgraph T_i such that the size of the minimum dominating set of T_i, denoted $|D(T_i)|$, is smaller than $(1+\epsilon)|D(S_i)|$. Therefore, we can conclude that $\sum_{i=1}^{k} D(T_i) \leq (1+\epsilon) \sum_{i=1}^{k} D(S_i) \leq (1+\epsilon)|D(V)|$. Thus, if S_i and T_i are chosen such that $\bigcup_{i=k}^{k} D(T_i)$ dominates G completely, then a dominating set of size at most $(1+\epsilon)$ times the size of the minimum dominating set can be obtained.

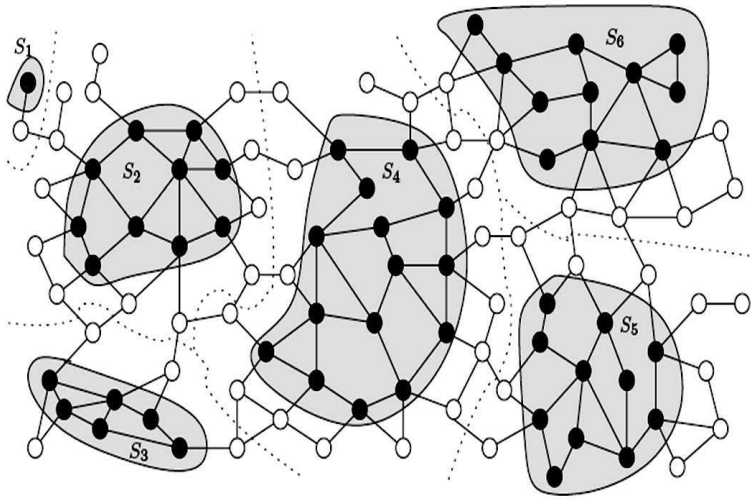


Figure 2.2: A 2-separated collection $S = \{S_1, \cdots, S_6\}$ in a graph $G = (V, E)$

Nieberg and Hurink [34] describe the algorithm to construct subgraphs S_i and T_i as follows. Choose an arbitrary vertex $v \in V$ and find a dominating set of $N_r[v]$ for $r = 1, 2, \cdots$ using an exhaustive search, until the cardinality of the dominating set does not grow too

much any more. More precisely, stop expanding the radius r of the neighborhoods if

$$|D(N_{r+2}[v])| \geq (1+\epsilon)|D(N_r[v])|$$

is violated.

The smallest value that violates the inequality above is denoted r'. Then $N_{r'}[v]$ stands for the subgraph S_i, and $N_{r'+2}[v]$ stands for T_i. In the next step of the algorithm, find another arbitrary vertex $v' \in G - N_{r'+2}[v]$ and construct another S_i and T_i. Repeat this process until no vertex in G remains to be covered. From the construction we can see that:

- the S_i subgraphs form a 2-separated collection
- for each i, $|D(T_i)| \leq (1+\epsilon)|D(S_i)|$
- the union of $D(T_i)$ for all i is a dominating set for G.

The above algorithm does not have any precondition of using a unit disc graph as the input. However, this fact is used to prove the polynomial running time of the algorithm.

Neiberg and Hurink also give a distributed implementation of their algorithm to find the minimum dominating set for a graph G. [26]

2.1.3 Connected Dominating Sets in Unit Disk Graphs

A *connected dominating set* S is a dominating set S whose induced subgraph is connected [38]. Since a dominating set must contain at least one vertex from each component of G, it follows that only connected graphs have a connected dominating set. Note that any nontrivial connected dominating set is also a total dominating set.

Cheng et al. give a PTAS to find connected dominating sets [3]. In this section we give a brief overview of a distributed algorithm given by Alzoubi et al. to construct a minimum connected dominating set in a unit disk graph [44]. Given a unit disk graph, the algorithm first builds a spanning tree using the distributed leader election algorithm [5] with $O(n)$ time complexity and $O(n\log n)$ message complexity. The construction of a connected dominating set then consists of two phases. In the first phase, a dominating set is constructed by constructing a maximal independent set. In the second phase, vertices are added to the dominating set to construct a connected dominating set. The algorithm has an approximation factor of at most 8, $O(n)$ time complexity, and $O(nlogn)$ message complexity. Funke et al. further improved this result to an approximation factor of at most 6.91 [13].

2.2 Maximal Independent Set

In Chapter 1 we noted that a maximal independent set S for a graph G is a dominating set for G. The geometric structure of a unit disk graph allows us to obtain an approximation on the size of maximal independent set. Hence, we can obtain an approximation for dominating set as well.

Marathe et al. gave a centralized algorithm to construct maximal independent sets in unit disk graphs [30]. Their algorithm is the following: select an arbitrary vertex v, add v to the maximal independent set, delete v and $N(v)$ from the graph. Marathe et al. showed that in any unit disk graph, the size of the maximum independent set in the subgraph G induced by the neighborhood of any vertex is at most 5. Since a maximal independent set is a dominating set, the size of any maximal independent set for a unit disk graph is within a factor of 5 of the size of a minimum dominating set [30].

Alzoubi et al. showed this same result [1]. They also show the following result for maximum distance-k independent sets in unit disk graphs.

Lemma 1. *For every node v, the number of dominators inside the disk centered at v with radius k units is bounded by a constant l_k.*

Proof. In a unit disk graph, any two vertices u and w that are k-adjacent to v and in an independent set are at least one unit apart. The half-unit disks centered at u and w are disjoint from each other. In addition, all such vertices are in the disk centered at v and with radius k. Then l_k is bounded by how many disjoint half-unit disks can fit in the disk centered at v with radius $k + 0.5$. See Figure 2.3. Using an area argument, we obtain $l_k \leq \frac{\pi(k+0.5)^2}{\pi(0.5)^2} = (2k+1)^2$. □

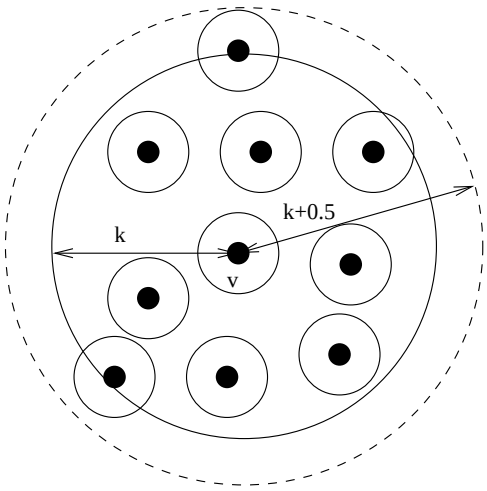


Figure 2.3: For every vertex v, the number of independent dominators within distance k is bounded by a constant l_k.

2.3 Generalized Dominating Set Algorithms

There are several approximation algorithms for finding p-dominating sets, distance-k dominating sets, total (open) dominating sets, total (open) p-dominating sets, and distance-k p-dominating sets in unit disk graphs.

2.3.1 p-Dominating Set

We may view a dominating set S as a set of sensors that either monitors or controls the vertices in $V \backslash S$. The removal, or failure, of an edge from G may result in a set which is no longer dominating. To make sure that S is still a dominating set after a failure of an edge, dominating each vertex with multiple vertices is proposed.

The idea of dominating each vertex in $V \backslash S$ multiple times originated with Fink and Jacobson [12]. A variation of p-dominating set is the p-tuple dominating set which was introduced by Harary and Haynes in [15].

Dai et al. [11] proposed three localized algorithms to construct a k-connected k-dominating set. For two positive integers m and k, an m-connected k-dominating set is a subset $S \subseteq V$ such that every vertex $u \in V \backslash S$ is adjacent to at least k vertices in S and there are at least m disjoint paths between each pair of vertices in S.

Two algorithms, k-gossip algorithm and color-based (k,k)-CDS algorithm, are probabilistic. In the k-gossip algorithm, each vertex decides to be in the dominating set with a probability based on the network size, deploying area size, transmission range, and k. In the color-based (k,k)-CDS algorithm, each vertex randomly selects one of the k colors such that the network is divided into k disjoint subsets based on the colors of the vertices. For each subset of vertices, a connected dominating set is constructed and (k,k)-CDS is the union of the k connected dominating sets. The deterministic algorithm, k-coverage, only works in very dense networks and no upper bound on the size of a resultant dominating set is analyzed.

Shang et al. gave a centralized algorithm for finding a connected p-dominating set [41]. A connected p-dominating set is a p-dominating set S such that all vertices in S are connected. The algorithm is as follows. Construct a maximal independent set I_1 and choose a set $C \subseteq V$ such that $I_1 \cup C$ is a connected dominating set [44]. Then for $2 \leq i \leq p$, the algorithm iteratively constructs a maximal independent set I_i in $V \backslash (I_1 \cup I_2 \cup \cdots \cup I_{i-1})$. Shang et al. also proved an approximation ratio of $(5 + \frac{5}{p})$ for $p \leq 5$ and an approximation ratio of 7 for $p > 5$ for their algorithm [41]. Shang et al. also gave two algorithms that construct a $(2,k)$-CDS and a (m,k)-CDS [41].

An incremental algorithm has also been given to construct a p-dominating set in unit disk graphs [10]. The algorithm iteratively constructs a monotone family of dominating sets $D_1 \subseteq D_2 \cdots \subseteq D_i \cdots \subseteq D_k$ such that each D_i is a an i-dominating set. To construct each dominating set a maximal independent set is constructed. The results show that for unit disk graphs, the size of each of the resulting i-dominating sets is at most six times the optimal [10].

Shang et al. proposed three centralized approximation algorithms to construct k-tuple dominating sets and m-connected k-tuple dominating sets for $m = 1, 2$ respectively [40].

2.3.2 Distance-k Dominating Set

A distance-k dominating set is also sometimes referred to as a k-dominating set or a k-hop dominating set. This variant was first introduced by Henning [18]. In this section we briefly discuss several heuristics given by Nguyen and Huynh to construct connected distance-k dominating sets [32].

The first algorithm, DCON, is a generalization of the greedy algorithm introduced in Section 2.1.1 for dominating sets. The DCON algorithm is as follows. Initially define the

connected distance-k dominating set $S = \emptyset$. In each iteration of the algorithm, an uncovered vertex v that covers the maximum number of uncovered vertices in its k-neighborhood, is added to S.

The second algorithm, DLCA, constructs a connected distance-k dominating set S based on the method of constructing a maximal independent set. The algorithm has two phases. During phase one, an uncovered vertex v that has the smallest ID is added to S. All $u \in N_k(v)$ are removed from the graph and this process is repeated until there are no more vertices to be covered. In phase two, a shortest path is found for every pair of vertices in S. Two vertices u, $v \in S$ are chosen and every vertex on the shortest path between u and v is added to S. This step is repeated until the induced subgraph of S is connected.

The third algorithm, DPMD, is a combination of the two previous algorithms, DCON and DLCA. DPMD consists of two phases. Phase one of DPMD is the same as phase one of DCON. Phase two of DPMD is phase two of DLCA, where vertices are added to the distance-k dominating set to make it connected.

Nguyen and Huynh presented other heuristics as well for the connected distance-k dominating set problem [32]. They also gave experimental results on the performance of their algorithms. The results show that DCON has the best performance out of all the algorithms they give.

Other results include several heuristics given by Nguyen et al. [33] for distance-k dominating set in planar geometric graphs. A geometric graph $G = (V, E)$ is a set V of vertices where each vertex is specified by its x and y coordinates and its transmission range. An edge exists between two vertices if they are within the transmission range of each other. G is planar if no edges crosses another.

2.3.3 Total (open) Dominating Set

The total (open) dominating set problem was defined by Cockayne, Dawes, and Hedetniemi [9].

Marathe et al. gave a centralized approximation algorithm for finding total dominating sets in unit disk graphs [30]. Their algorithm first constructs a maximal independent set S for graph G. Then, for each vertex $v \in S$, if v is not adjacent to another vertex $u \in S$ such that $u \neq v$ then choose a vertex $w \in V \backslash S$ such that v is adjacent to w and add it to S. This procedure leads to a total dominating set of size at most $2|S|$. The vertices that are added to S in the second phase of the algorithm are chosen in such a way that the obtained

total dominating set is also a connected dominating set. Since the size of any maximal independent set is within a factor of 5 of the size of a minimum dominating set in a unit disk graph, this algorithm for finding a total dominating set is a 10-approximation. It should be mentioned that when the x or y-coordinates of the vertices in the unit disk graph are known and sorted, the 5-approximation ratio for maximal independent set can be reduced and a 3-approximation algorithm can be attained [30]. This leads to a 6-approximation ratio for the total dominating set.

Wang et al. introduced the self-protection problem in wireless sensor networks. A sensor is p-self-protected if each sensor is covered by at least $p-1$ other sensors [45]. Wang et al. focused on 2-self-protection, where each sensor is covered by at least one other sensor. A set that provides 2-self-protection is equivalent to a total dominating set, where each sensor is adjacent to at least one sensor in the total dominating set. Wang et al. gave a centralized algorithm to construct a total dominating set by constructing a dominating set. The algorithm consists of two steps. In step one, a dominating set S is constructed. In step two, for each $v \in S$, if v is not adjacent to a $u \in S$ and $u \neq v$, then choose a vertex $w \in V \backslash S$ such that v is adjacent to w and add it to S. This leads to a total dominating set of size at most $2|S|$. Wang et al. used a $(1 + \log|V|)$ approximation algorithm given by Johnson [22] to construct a dominating set. By doubling this, one can easily obtain a $2(1 + \log|V|)$ approximation algorithm for a total dominating set.

Wang et al. also gave distributed algorithms for total dominating set [45], which yields the same approximation ratio.

2.3.4 Total (open) p-Dominating Set

The total (open) p-dominating set was defined by Kulli [27]. Wang et al. extended 2-self protection in wireless sensor networks to p-self protection. The p-self protection problem in wireless sensor networks is modelled as a unit disk graph and Wang et al. redefined it as follows. A wireless sensor network is p-self protected, if for any wireless sensor there are at least p sensors that can monitor it. This is equivalent to a total p-dominating set [46]. Wang et al. gave centralized and distributed approximation algorithms for the problem in unit disk graphs. The centralized algorithm is an extension of the algorithm given by Marathe et al. [30] for finding a total dominating set. More precisely, the algorithm constructs a maximal independent set for p rounds to obtain a total (open) p-dominating set. Both the centralized and distributed algorithms give a 10-approximation for unit disk graphs.

Wang et al. also introduced a distributed algorithm for sensor networks with heterogenous transmission ranges [46]. Their algorithm uses the partitioning method given by Li et al. [29]. Li et al. proved their partitioning method for Extended Yao graphs which is a subclass of unit disk graphs [29]. However, Li et al. did not extend the partitioning method to directed disk graphs, where there are heterogenous transmission ranges.

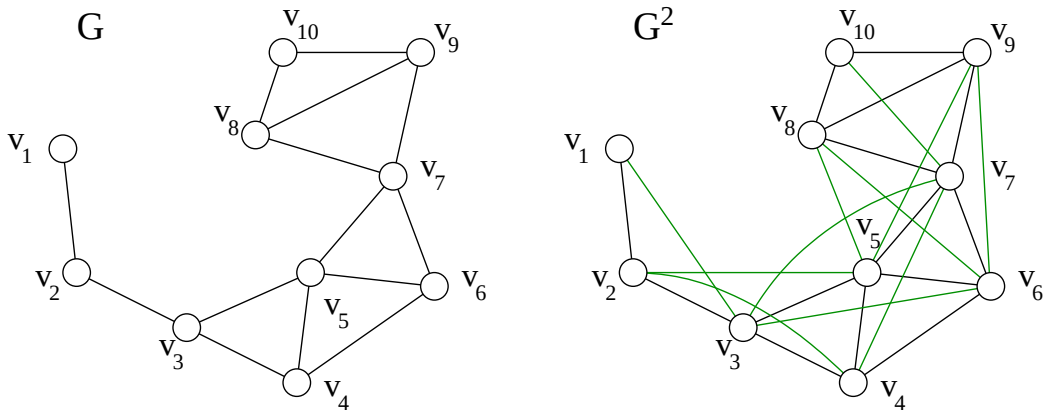
When the transmission ranges of all vertices are not equal, the wireless network can be modelled as a disk graph $G = (V, E)$. Vertices in V are located in a Euclidean plane and each $v_i \in V$ has a transmission range $r_i \in [r_{min}, r_{max}]$, where r_{min} and r_{max} are the shortest and longest transmission ranges respectively in G. A directed edge $(v_i, v_j) \in E$ if and only if the Euclidean distance between v_i and v_j is less than or equal to r_i. an edge (v_i, v_j) is bidirectional if both (v_i, v_j) and (v_j, v_i) are in E. In other words, v_i and v_j are adjacent if they are within each others transmission ranges. When all edges in G are bidirectional, G is called a bidirectional disk graph. Note that when $\frac{r_{max}}{r_{min}} = 1$, a bidirectional disk graph is a unit disk graph. Thai and Du proved that the partitioning method can be extended to disk graphs with bidirectional links [43].

2.3.5 Distance-k p-Dominating Set

Distance-k p-dominating set was first introduced by Joshi et al.. They introduced it as p-neighbor k-domination and proved it to be NP-complete on interval graphs [24]. A distance-k p-dominating set is also known as a k-hop p-dominating set. Distance-k p-dominating sets have been used for clustering techniques. Spohn et al. [42] proposed a clustering technique to address redundancy for bounded distance clusters. This is similar to computing a distance-k p-dominating set. They presented centralized and distributed solutions to minimum distance-k p-dominating set for arbitrary topologies. They gave a $(p\ln\Delta_k)$-approximation for the centralized algorithm, where Δ_k is the largest cardinality among all k-neighborhoods in the network.

Li et al. proposed two approximation algorithms for minimum connected distance-k p-dominating set in unit disk graphs [28]. The construction in the first algorithm is based on power graphs. For any positive integer k, the k-th power graph of $G = (V, E)$, $G^k(V, E^k)$, consists of the vertex set V and the edge set E^k, where there is an edge in E^k between two vertices u, $v \in V$ if and only if $d(u, v)$ in G is at most k. Figure 2.4 shows an example of a graph G and its power graph G^2 with the added edges in green.

The first algorithm given by Li et al. to construct a connected distance-k t-dominating



(a) Original graph G. (b) The 2 power graph G^2.

Figure 2.4: Power graph for a given graph G.

set S is as follows. Given a graph G, the algorithm first constructs the power graph G^k. The algorithm then is divided into two phases. In phase one, the algorithm constructs a maximal independent set I_i using vertices from $E^k \backslash (I_1 \cup I_2 \cup \cdots \cup I_{i-1})$ for $1 \leq i \leq p$. Note that all vertices in $E^k \backslash (I_1 \cup I_2 \cup \cdots \cup I_{i-1})$ are dominated by I_1, I_2, $\cdots$, I_{i-1}. $(I_1 \cup I_2 \cup \cdots \cup I_{i-1})$ is an $(i-1)$-dominating set of G^k. Let $S = I_1 \cup I_2 \cup \cdots \cup I_p$. Phase two of the algorithm adds extra vertices into S such that the subgraph induced by S in G is connected.

A maximal independent set in G^k is a maximal distance-k independent set in G. Therefore, the above algorithm constructs a connected distance-k t-dominating set. Li et al. also proved an approximation ratio of $(2k+1)^3$ for $p \leq (2k+1)^2$ and an approximation ratio of $(2k+1)((2k+1)^2+1)$ for $p > (2k+1)^2$.

The second approximation algorithm of Li et al. is a greedy based heuristic. Before we describe the algorithm, we first introduce some notation.

• Consider a graph $G = (V, E)$ and let $S \subseteq V$ be a distance-k p-dominating set for G.

• Let Δ_k denote the largest cardinality among all k-neighborhoods in G. i.e. $\Delta_k = max\{|N_r([u])| \ \ |u \in V\}$.

• For every non p-dominated vertex u, let $f(u)$ be the number of vertices in S that are k-adjacent to u. Let $D(u)$ denote the number of vertices needed to dominate u, i.e. $D(u) = p - f(u)$.

• Let $T(u) = \sum_{v \in N_k[u]} D(v)$.

The algorithm consists of two phases. In phase one, a distance-k p-dominating set S is

constructed. A vertex v with maximum T value is repeatedly selected to be in S until every vertex is p-dominated (i.e. the D values of all vertices are 0). In phase two, vertices are added into S such that the subgraph induced by S is connected. Li et al. showed that this algorithm returns a connected distance-k p-dominating set of size at most $(2k+1)\ln\Delta_k|OPT|$ for any undirected graph, where OPT is an optimal solution for minimum connected distance-k p-dominating set.

Li et al. showed through experimental results that the greedy algorithm outperforms the first algorithm. The reason for this is because the second algorithm always chooses the "best" vertex to dominate the graph, while the first algorithm may not.

Chapter 3

Algorithms for Minimum $D_{k,p}$

In this section we give three approximation algorithms to find a small distance-k total p-dominating set in unit disk graphs. Two of the algorithms are centralized and the third is distributed. To achieve a constant approximation ratio for the distributed algorithm and one of the centralized algorithms, we use a maximal distance-k independent set in the construction of a small $D_{k,p}$. The second centralized algorithm is based on a greedy heuristic. Before we discuss the three approximation algorithms, we first give the algorithm that obtains a maximal distance-k independent set.

Denote a maximal distance-k independent set by MIS_k. Recall that Alzoubi et al. showed that for unit disk graphs, for every vertex v there is at most a constant number of vertices, $l_k \leq (2k+1)^2$, in MIS_k k-adjacent to v [1]. In other words, for every vertex v the number of vertices in MIS_k within distance k of v is bounded by l_k. Thus, the size of a maximal distance-k independent set is at most $l_k \leq (2k+1)^2$ times the optimal solution.

The rest of this chapter is divided as follows. In section 3.1 we give a centralized algorithm to construct a maximal distance-k independent set. In section 3.2, a centralized algorithm for minimum distance-k total p-dominating set is given based on the construction of an MIS_k. A distributed algorithm for minimum $D_{k,p}$ is given in section 3.3. Section 3.4 introduces a centralized greedy heuristic to find a small distance-k total p-dominating set. For all of these algorithms we consider a unit disk graph $G = (V, E)$.

3.1 Centralized Algorithm for MIS_k

In this section we give the centralized algorithm for maximal distance-k independent set. The idea of the algorithm is the following. Choose an arbitrary vertex v and add v to the maximal distance-k independent set. Delete v and $N_k(v)$ from G. Repeat this process until there are no more vertices in G. The pseudo code of the algorithm is given below.

Algorithm 1 Maximal Distance-k Independent Set

Input: $G = (V, E)$, $k \geq 1$.
Output: A maximal distance-k independent set S.

1: $S = \emptyset$, $D = V$.
2: **while** $D \neq \emptyset$ **do**
3: Choose an arbitrary vertex $v \in D$.
4: Add v to S (i.e. $S = S \cup \{v\}$).
5: Delete v and $N_k(v)$ from D (i.e. $D = D \backslash N_k[v]$).
6: **end while**

Proposition 1. *Algorithm 1 produces a maximal distance-k independent set.*

Proof. Assume that the set S produced by Algorithm 1 is not a maximal distance-k independent set. This implies that there exists a vertex u that can be added to S (i.e. u is at least distance $k+1$ from all vertices $w \in S$). However, this cannot be since all vertices $v \notin S$ are within distance k of at least one vertex in S according to steps 4-6 in Algorithm 1. Since vertex $u \notin S$, u is dominated by a vertex $w \in S$. Therefore, $S \cup \{u\}$ is not a maximal distance-k independent set. □

Proposition 2. *A maximal distance-k independent set is a distance-k dominating set.*

Proof. We will give a proof by contradiction. Assume that a maximal distance-k independent set S is not a distance-k dominating set. This implies that there exists a vertex $v \in V \backslash S$ such that for every vertex $u \in S$, $d(u, v) \geq k+1$. From the definition of a maximal distance-k independent set we know that for any pair of vertices a, $b \in S$, $d(a, b) \geq k+1$. Therefore, if $v \notin S$ and for any $u \in S$, $d(u, v) \geq k+1$ then v should be added to S. This is a contradiction since S is already a maximal distance-k independent set. Thus, a maximal distance-k independent set is a distance-k dominating set. □

3.2 Centralized Algorithm for Minimum $D_{k,p}$

In this section we give a centralized algorithm for the distance-k total p-dominating set. We use Algorithm 1 to construct a small $D_{k,p}$. The method, Algorithm 2, produces a constant approximation ratio of $2(2k+1)^2$ when k is constant. Recall that two special cases of $D_{k,p}$ are the total dominating set when $k = p = 1$ and total p-dominating set when $k = 1$ and $p \geq 1$. For both total dominating set and total p-dominating set, 10-approximation algorithms have been given by Marathe et al. [30] and by Wang et al. [46] respectively. Both algorithms were discussed in Chapter 2.

Algorithm 2 Centralized method for Minimum $D_{k,p}$

Input: $G = (V, E)$, $k \geq 1$, $p \geq 1$.
Output: A distance-k total p-dominating set S.

1: $S = \emptyset$.
2: Let $i = 1$.
3: **while** $i \leq p$ **do**
4: Construct a maximal distance-k independent set I_i for G using vertices in $V \backslash (I_1 \cup I_2 \cup \cdots \cup I_{i-1})$.
5: **end while**
6: Let $S = I_1 \cup I_2 \cup \cdots \cup I_p$.
7: For each vertex $v \in S$, if the number of vertices of S k-adjacent to v is less than p, then find a vertex $x \notin S$ that is k-adjacent to v and add it to S.

Proposition 3. *For a given graph G, if a distance-k total p-dominating set exists, the set of vertices S produced by Algorithm 2 is a distance-k total p-dominating set and has size at most $2(2k+1)^2$ times the optimum solution of minimum distance-k total p-dominating set.*

Proof. Each round i of Algorithm 2 produces a new maximal distance-k independent set I_i (steps $3-5$). We showed in Proposition 2 that a maximal distance-k independent set is a distance-k dominating set. The set S produced by Algorithm 2 is a union of p distance-k dominating sets (step 6). Thus, S is a distance-k dominating set.

We now need to show that S is also a total p-dominating set. For each vertex $u \notin S$, u has at least p k-adjacent vertices in S, since in each round i an I_i is constructed where there is at least one vertex I_i that dominates u. Note that the vertices selected to be in I_i during round i are not selected during the previous $i-1$ rounds since each new I_i is constructed

from the vertex set $V \backslash (I_1 \cup I_2 \cup \cdots \cup I_{i-1})$. For each vertex $u \in S$, u is k-adjacent to at least $p-1$ vertices in S since u is dominated in each round i except the round it is selected to be in S. If u has only $p-1$ k-adjacent vertices, Algorithm 2 adds to S a vertex $x \notin S$ k-adjacent to u (step 7). Hence, all vertices are k-adjacent to p vertices in S. Therefore, Algorithm 2 produces a distance-k total p-dominating set.

Now we show that the approximation ratio for Algorithm 2 is $2(2k+1)^2$. We know that for each vertex v at most $(2k+1)^2$ vertices k-adjacent to v are chosen to be in S in each round i of I_i. Thus, for each vertex, there are at most $(2k+1)^2 \cdot p$ k-adjacent vertices in S. For the optimal solution of $D_{k,p}$, there are at least p k-adjacent vertices in S for each vertex. Thus, the number of vertices in S is at most $(2k+1)^2$ times of the optimal solution of $D_{k,p}$. In step 7 of Algorithm 2, for each vertex $w \in S$ with $p-1$ k-adjacent vertices, one additional vertex $x \notin S$ that is k-adjacent to w is added to S. Thus, the total number of vertices selected to be in S is at most $2(2k+1)^2$ times of the optimal solution of distance-k total p-dominating set. □

3.3 Distributed Algorithm for Minimum $D_{k,p}$

A centralized solution is good for sensor networks with centralized control. However, in many applications of sensor networks, there is no centralized control. Instead all sensors are self-organized. Thus, each sensor needs to make decisions based on limited information. For such self-organized sensor networks, it is preferred to design a distributed method to address distance-k total p-dominating set.

Our distributed approximation algorithm for minimum distance-k total p-dominating set given in Algorithm 3 is extended from the centralized method, Algorithm 2. We assume that each vertex v maintains information about itself and the set $N_k(v)$ of vertices that are k-adjacent to v. Let set S be a distance-k total p-dominating set returned by Algorithm 3. The following notation will be used for Algorithm 3.

- $ID(v)$: distinct ID of vertex v.
- $d(v)$: number of vertices in S that are k-adjacent to vertex v.
- $s(v)$: the status of vertex v shows the current role of vertex v (Undecided, Dominator, or Dominated).

We also use three kinds of messages to exchange information among neighboring vertices:

- Dominate(x, d, dist): vertex x uses this message to tell its k-neighborhood that it has become a dominating vertex and that it is dominated by d other vertices. The parameter *dist* is used to make sure the message from vertex x is received by all vertices in $N_k(x)$. A dominating vertex x initializes $dist = 0$. Vertices in $N_1(x)$ will increment $dist = 1$. Vertices in $N_i(x)$ will increment $dist = i$.

- ReqDom(x, y, dist): A dominating vertex x that has less than p dominating k-adjacent vertices sends this message to a dominated k-adjacent vertex y asking y to become a dominating node.

- Update(x, dist): vertex x uses this message to inform all vertices in $N_k(x)$ that its status has changed from Undecided to Dominated.

The idea of the distributed algorithm is as follows. All vertices are initially Undecided and S is initially empty. Each vertex v has information about all of its k-adjacent vertices and hence knows the number of times each of its k-adjacent vertices is dominated. If an Undecided vertex v has the smallest ID among all Undecided vertices that are in $N_k[v]$, then v becomes a vertex in S. It then sends a Dominate(v, d, dist) message to all vertices in $N(v)$. The vertices in $N(v)$ then propagate the Dominate(v, d, dist) message to all vertices in $N_k(v)$. Every vertex $u \in N_k(v)$ that receives the Dominate(v, d, dist) message update their local value of $d(v)$. When vertex v and all $u \in N_k(v)$ have p or $p-1$ dominators, vertex v decides its status (i.e. whether it should be Dominator or Dominated). All vertices in S are marked as Dominator while vertices with Undecided status are marked Dominated. A Dominator vertex v may have less than p k-adjacent vertices at the end of p rounds. In such a case, v sends a ReqDom(v, u, dist) message to a Dominated k-adjacent vertex $u \in N_k(v)$ and asks u to become a Dominator. When u receives the ReqDom(v, u) message, it changes its status to Dominator and sends an Update message to its k-neighborhood.

Algorithm 3 Distributed Algorithm for Minimum $D_{k,p}$ at vertex v

Input: $G = (V, E)$, $k \geq 1$, $p \geq 1$.
Output: A distance-k total p-dominating set S.

Initialization: $d(v) = 0$, $s(v)$ =Undecided.

while ($\exists\ u \in N_k[v]$ such that ($d(u) < p$ and $s(u)$ =Undecided) or
($d(u) < p - 1$ and $s(u)$ =Dominator)) **do**

{Case 1: Vertex v is ready to become a dominating vertex}
if $s(v)$ =Undecided **then**
  if $ID(v) < ID(u)$ for every $u \in N_k(v)$ such that $s(u)$ =Undecided **then**
    v becomes a vertex in S.
    v sends Dominate(v, d(v), dist=0) message to all vertices in $N(v)$.
  end if
end if

{Case 2: Vertex v receives a Dominate message}
if v receives Dominate(x, d, dist) message and $s(x)$ =Undecided **then**
  $d(v) = d(v) + 1$
  v updates its local copy of $s(x)$ =Dominator
  v updates its local copy of $d(x)$
  if dist $< k - 1$ **then**
    v sends Dominate(x, d, dist+1) message to $N(v)$.
  end if
  if $d = -1$ **then**
    v updates its local copy of $s(x)$ =Dominator.
  end if
end if

{Case 3: Vertex v has enough dominators}
if ($d(v) \geq p$ and $s(v)$ =Undecided) or ($d(v) \geq p - 1$ and $s(v)$ =Dominator) and each vertex in $N_k(v)$ also satisfies these conditions **then**
  if $s(v)$ =Undecided **then**
    $s(v)$ =Dominated
    v sends Update(v, dist=0) message to $N(v)$
  else if $d(v) < p$ **then**
    select a vertex $u \in N_k(v)$ such that $s(u)$ =Dominated.
    v sends a ReqDom(v, u, dist=0) message to u.
  end if
end if

Algorithm 4 Distributed Algorithm for Minimum $D_{k,p}$ at vertex v continued

```
{Case 4: Vertex v receives a ReqDom message}
if v receives a message ReqDom(u, y, dist) then
  if v = y then
    s(v) =Dominator
    v sends message Dominate(v, d(v), dist=0) to N(v).
  else if dist< k − 1 then
    v sends ReqDom(u, y, dist+1) to N(v)
  end if
end if

{Case 5: Vertex v receives an Update message}
if v receives an Update(x, dist) message then
  v updates its local copy of s(x) =Dominated.
  if dist< k − 1 then
    v sends Update(x, dist+1) to N(v)
  end if
end if
```

Proposition 4. *Algorithm 3 produces a distance-k total p-dominating set of size at most* $2(2k+1)^2$ *times the optimum solution of* $D_{k,p}$.

The proof is similar to that of the centralized algorithm for minimum $D_{k,p}$.

Proposition 5. *The message complexity of Algorithm 3 is* $O(n \cdot \Delta_k)$.

Proof. We count the messages by different types: Dominate, ReqDom, and Update. Let Δ_k denote the maximum cardinality of the k-neighborhoods in G and let S be the set containing all Dominator vertices.

A Dominate message is sent once by each vertex in S. There are at most n such messages sent. Each time a vertex $v \in S$ sends a Dominate message, the message propagates through $N_k(v)$. Hence, in total there are at most $n \cdot \Delta_k$ Dominate messages sent.

The number of ReqDom messages is also limited by $n \cdot \Delta_k$. Only vertices in S with less than p k-adjacent vertices in S use the ReqDom message. Again, there are at most n such vertices, and each time a vertex $v \in S$ sends a ReqDom message, it propagates to $N_k(v)$. Therefore, a total of at most $n \cdot \Delta_k$ messages are sent.

The number of Update(v, dist) messages that are sent is at most $n \cdot \Delta_k$ since each vertex v sends at most one such message and the Update message of each vertex v propagates to $N_k(v)$. Thus, the total number of messages used by Algorithm 3 is bounded by $O(n \cdot \Delta_k)$. □

3.4 Centralized Greedy Algorithm for Minimum $D_{k,p}$

The third algorithm we introduce is an extension of the first algorithm given by Spohn et al. for the distance-k p-dominating set [42]. This algorithm is a centralized method based on a greedy heuristic. Recall that a distance-k p-dominating set for graph $G = (V, E)$ is a set $S \subseteq V$, such that for every vertex $v \notin S$, v has p k-adjacent vertices in S. Every vertex $u \in S$, may or may not have any k-adjacent vertices in S. To obtain a distance-k total p-dominating set, every vertex $u \in S$ must have p k-adjacent vertices in S. Hence, the following proposition follows.

Proposition 6. *For fixed integers $k \geq 1$ and $p \geq 1$ the number of vertices in a minimum distance-k total p-dominating set is at most p times the number of vertices in a minimum distance-k p-dominating set.*

Proof. A distance-k p-dominating set is a set S of vertices where all vertices not in S have p k-adjacent vertices in S. One can see that a minimum $D_{k,p}$ is a distance-k p-dominating set. We now prove, by contradiction, that the number of vertices in a minimum distance-k p-dominating set is at least $\frac{1}{p}$ of the number of vertices in a minimum $D_{k,p}$.

If the distance-k p-dominating set contains fewer vertices than $\frac{1}{p}$ of the minimum $D_{k,p}$, then we add p k-adjacent vertices for each vertex in this minimum distance-k p-dominating set. The resulting set of vertices is a distance-k total p-dominating set. This contradicts that the number of vertices selected to be in $D_{k,p}$ is minimum. □

We now show that an approximation algorithm exists for minimum distance-k total p-dominating set through the minimum distance-k p-dominating set.

Proposition 7. *For fixed integers $k \geq 1$ and $p \geq 1$, a $p \cdot \ln\Delta_k$-approximation algorithm exists for minimum distance-k total p-dominating set.*

Proof. Let Δ_k be the largest cardinality k-neighborhood in G. A $ln\Delta_k$-approximation algorithm for minimum distance-k p-dominating set is given by Spohn et al. [42]. Since a minimum distance-k total p-dominating set will not be smaller than a minimum distance-k p-dominating set, then by adding p vertices for each vertex in the distance-k p-dominating set, we will have a $p \cdot ln\Delta_k$-approximation algorithm. □

3.5 Heuristics for $D_{k,p}$

In the previous sections we introduced three approximation algorithms. In this section, we give three greedy based heuristics to find small distance-k total p-dominating sets. The three algorithms are called Ran&Greedy, Greedy and Greedy2. All three algorithms are centralized methods.

3.5.1 Centralized Algorithm, Ran&Greedy, for $D_{k,p}$

The first algorithm, Ran&Greedy, is a modification of Algorithm 2 from section 3.2. We call Algorithm 2 Random. However, note that Algorithm 2 is not completely random, since phase two of the algorithm is not done in a random manner. The selection of a vertex in phase two of Algorithm 2 depends on the adjacencies in the graph.

Recall that Algorithm 2 consists of two phases. In phase one, the algorithm constructs maximal distance-k independent sets for p rounds to obtain a distance-k p-tuple dominating set S. In phase two, it adds extra vertices to S to obtain a distance-k total p-dominating set. For Ran&Greedy, we modify phase two of Algorithm 2 so that the number of extra vertices added to S is as small as possible.

Algorithm 5 Centralized Algorithm: Ran&Greedy for Minimum $D_{k,p}$

Input: $G = (V, E)$, $k \geq 1$, $p \geq 1$.
Output: A distance-k total p-dominating set S.

1: $S = \emptyset$.
2: Let $i = 1$.
3: **while** $i \leq p$ **do**
4: Construct a maximal distance-k independent set I_i for G using vertices from $V \backslash (I_1 \cup I_2 \cup \cdots \cup I_{i-1})$.
5: **end while**
6: Let $S = I_1 \cup I_2 \cup \cdots \cup I_p$.
7: For each vertex $v \in S$, if the number of vertices that are k-adjacent to v is less than p, then add to S a vertex $x \notin S$ that is k-adjacent to v such that x dominates the maximum number of vertices in S.

Since Ran&Greedy is a modification of Algorithm 2, the set of vertices S produced by Ran&Greedy is a distance-k total p-dominating set and has size at most $2(2k+1)^2$ times of

the optimum solution of minimum distance-k total p-dominating set. The proof is the same as that of proposition 3.

The complexity analysis of Algorithm 5 is as follows. Let n be the number of vertices in G, d be the size of the distance-k total p-dominating set and Δ_k be the largest cardinality k-neighborhood of G. In phase one of Algorithm 5, Algorithm 1 is used to construct a maximal distance-k independent set. Algorithm 1 takes $O(d\,\Delta_k) \leq O(n\,\Delta_k) \leq O(n^2)$. Phase one of Algorithm 5 runs p times and thus takes $O(p\,d\,\Delta_k) \leq O(p\,n\,\Delta_k)$. Phase two of Algorithm 5 takes $O(2|S|\,\Delta_k) \leq O(n^2)$. Thus, Algorithm 5 takes $O(p\,d\,\Delta_k + 2|S|\,\Delta_k) \leq O(p\,n^2)$.

3.5.2 Centralized Algorithm, Greedy, for $D_{k,p}$

The second algorithm we introduce to obtain a small distance-k total p-dominating set is based on a greedy heuristic. The algorithm is simply called Greedy. Like Ran&Greedy, Greedy consists of two phases as well. In phase one, the Greedy algorithm constructs a distance-k p-tuple dominating set S. In phase two, extra vertices are added to S to obtain a distance-k total p-dominating set. Both phases operate in a greedy manner. The difference between phase one of Ran&Greedy and Greedy is as follows. The Ran&Greedy algorithm iteratively constructs a maximal distance-k independent set, which is a distance-k dominating set. The Greedy algorithm in phase one constructs a distance-k dominating set iteratively in a greedy manner. The construction of a distance-k dominating set in a greedy manner is similar to the construction of a maximal distance-k independent set. The idea of the algorithm is the following. Choose a vertex v with maximum degree and add v to the distance-k dominating set. Delete v and $N_k(v)$ from G. Repeat this process until there are no more vertices in G. The algorithm for Greedy distance-k dominating set is called Greedy_D$_k$ and the pseudo code of the algorithm is given as Algorithm 6.

Algorithm 6 produces a maximal distance-k independent set. The proof is similar to that of Proposition 1. Hence, Algorithm 6 produces a distance-k dominating set by Proposition 2. The Greedy algorithm uses Algorithm 6 in phase one and returns a distance-k total p-dominating set. The proof is similar to the proof of Proposition 3.

Algorithm 6 Greedy_D$_k$

Input: $G = (V, E)$, $k \geq 1$.
Output: A distance-k dominating set S.

1: $S = \emptyset$, $D = V$.
2: **while** $D \neq \emptyset$ **do**
3: Choose a vertex $v \in D$ such that v has the maximum degree.
4: Add v to S (i.e. $S = S \cup \{v\}$).
5: Delete v and $N_k(v)$ from D (i.e. $D = D \backslash N_k[v]$).
6: **end while**

The complexity analysis for the Greedy algorithm is as follows. We first give the complexity of Algorithm 6. Using an adjacency list implementation for the graph, choosing a vertex of maximum degree out of n vertices takes $O(n)$ time. Thus, Algorithm 6 takes $O(d(n + \Delta_k)) \leq O(n^2)$. Phase one of Algorithm 7 is repeated p times and thus, takes $O(p\, d(n + \Delta_k)) \leq O(p\, n^2)$. Phase two of Algorithm 7 takes $O(2|S|\, \Delta_k) \leq O(n^2)$. Thus, Algorithm 7 takes $O(p\, d(n + \Delta_k) + 2|S|\, \Delta_k) \leq O(p\, n^2)$.

Algorithm 7 Greedy Algorithm for Minimum $D_{k,p}$

Input: $G = (V, E)$, $k \geq 1$, $p \geq 1$.
Output: A distance-k total p-dominating set S.

1: $S = \emptyset$.
2: Let $i = 1$.
3: **while** $i \leq p$ **do**
4: Construct a distance-k dominating set D_i for G using vertices from $V \backslash (D_1 \cup D_2 \cup \cdots \cup D_{i-1})$.
5: **end while**
6: Let $S = D_1 \cup D_2 \cup \cdots \cup D_p$.
7: For each vertex $v \in S$, if the number of vertices that are k-adjacent to v is less than p, then add to S a vertex $x \notin S$ that is k-adjacent to v and such that x dominates the maximum number of vertices in S.

3.5.3 Centralized Algorithm, Greedy2, for $D_{k,p}$

The third heuristic, Greedy2, that we introduce is an extension of a greedy algorithm given by Sanchis [39]. Greedy2 operates similarly to Greedy. However, unlike Greedy, Greedy2 pays

attention to the specific vertices which would be dominated, and to whether or not they could be dominated in some alternative way. If a vertex v would dominate a vertex u if v were added to the distance-k total p-dominating set, and u had a small degree, then v has a stronger reason for being added to the $D_{k,p}$ than it would for potentially covering a vertex w which has high degree and may therefore be dominated by many vertices. Greedy2 consists of two phases. In phase one, it constructs a distance-k p-tuple dominating set S. In phase two, extra vertices are added to S to obtain a distance-k total p-dominating set.

To implement this algorithm we define the following for each vertex $v \in V$. Let $deg(v)$ denote the number of adjacent vertices to v. The quantity $1 + deg(v)$ represents the number of vertices that can dominate v including itself. Define $T(v) = \frac{1}{1+deg(v)}$ for each vertex v. Let $weight(v)$ of each vertex v be the sum of all the quantities $T(w)$ where $w \in N(v)$, has not yet been dominated, and can be dominated by v. A vertex v with maximum value $weight(v)$ is chosen at each iteration of the algorithm.

The complexity analysis for the Greedy2 algorithm is as follows. Phase one of Algorithm 8 is repeated p times and thus, takes $O(p\,(n\,\Delta_k + d(n + \Delta_k)) \leq O(p\,n^2)$. Phase two of Algorithm 8 takes $O(2|S|\,\Delta_k) \leq O(n^2)$. Thus, Algorithm 8 takes $O(p\,(n\Delta_k + d(n + \Delta_k)) + 2|S|\,\Delta_k) \leq O(p\,n^2)$.

Algorithm 8 Greedy2 for Minimum $D_{k,p}$

Input: $G = (V, E)$, $k \geq 1$, $p \geq 1$.
Output: A distance-k total p-dominating set S.

1: $S = \emptyset$, $D = V$.
2: Let $i = 1$.
3: **while** $i \leq p$ **do**
4: **for** all vertices $v \in D$ **do**
5: $T(v) = \frac{1}{1+deg(v)}$
6: $weight(v) = T(v)$
7: **end for**
8: **for** all vertices $v \in D$ **do**
9: **for** all vertices $u \in N(v)$ **do**
10: $weight(v) = weight(v) + T(u)$
11: **end for**
12: **end for**
13: **while** $D \neq \emptyset$ **do**
14: Choose a vertex $v \in D$ such that $weight(v)$ is maximum.
15: Add v to S (i.e. $S = S \cup \{v\}$).
16: Delete v and $N_k(v)$ from D (i.e. $D = D \backslash N_k[v]$).
17: **end while**
18: **end while**
19: For each vertex $v \in S$, if the number of vertices that are k-adjacent to v is less than p, then add to S a vertex $x \notin S$ that is k-adjacent to v and such that x dominates the maximum number of vertices in S.

Chapter 4

Multiple Regression Analysis

4.1 Method

In Chapter 4 we study the performance of three algorithms that we introduced in Chapter 3. The three algorithms tested for experimental results are two approximation algorithms and one heuristic, namely Random, Ran&Greedy, and Greedy. We do not use Greedy2 in our extensive experimental results. Greedy2 is an enhanced version of the Greedy algorithm. However, brief experimental results have shown that it does not perform better than the Greedy algorithm. Hence, we exclude it from our experimental analysis.

Given a graph G, all algorithms return a distance-k total p-dominating set obtained for G. We would like to compare the sizes of the distance-k total p-dominating sets returned by each algorithm. This will let us compare the three algorithms and see which one returns the best result. To determine this result we conduct several experiments for each algorithm. To model our results statistically, we use a multiple regression model [31], [25], [21]. A multiple regression model allows us to estimate a random variable as a function of several other variables. The following terms are used in our design and analysis of experiments:

- *response variables*–variables which are estimated in the experiment. One would like either to minimize or maximize the response.
- *predictor variables*–variables used to predict the response. Predictor variables are also referred to as *predictors*. Some predictor variables will have more of an effect on the response variable than others. The predictor variables may all be separate variables,

or some may be functions of a few variables. Each predictor variable may have several alternatives.

- *levels*–the values that a predictor variable can assume. Each level of a predictor variable constitutes one alternative for that predictor.
- *replication*–repetition of all or some experiments.
- *interaction*–two predictor variables A and B are said to interact if the effect of one depends upon the level of the other.

The general form of a regression model for m predictor variables is given by

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m + \varepsilon \tag{4.1}$$

where y is the response variable, $x_1, x_2, \cdots, x_m$ are the predictor variables, ε is the model error and $\beta_0, \beta_1, \cdots, \beta_m$ are the *regression coefficients* that need to be estimated. We ordinarily view the above model in a data setting where n observations $(x_{11}, x_{21}, \cdots, x_{m1}, y_1)$, $(x_{12}, x_{22}, \cdots, x_{m2}, y_2)$, $\cdots$, $(x_{1n}, x_{2n}, \cdots, x_{mn}, y_n)$ are taken, and estimates of the regression coefficients are sought. We then write the model as

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_m x_{mi} + \varepsilon_i \tag{4.2}$$

where $i = 1, 2, \cdots, n; n \geq m + 1$.

The general multiple regression model given in (4.1) only considers main effects. If we want to consider all interactions between all predictor variables, then the model changes. A multiple regression model with three predictor variables, x_1, x_2, x_3 and their all interactions is given as follows

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_1 x_2 + \beta_5 x_1 x_3 + \beta_6 x_2 x_3 + \beta_7 x_1 x_2 x_3 + \varepsilon. \tag{4.3}$$

An important goal of the regression analysis is to estimate the regression coefficients. The regression coefficients are often called *partial regression* coefficients. For example, parameter β_1 is interpreted as the expected change in the response (positive or negative) per unit change in x_1 with the other x's held constant. Similar interpretations can be made for the other β's.

Our analysis in this chapter is divided into two parts. In the first part we consider the analysis of finding a distance-k total p-dominating set where k and p are fixed at the value of 1. This is a total dominating set. In the second part, we consider the analysis for finding a distance-k total p-dominating set where k and p are not fixed to a single value.

4.2 Response Variable

All of the algorithms presented in Chapter 3 return a set of vertices that is a distance-k total p-dominating set for positive integers k and p. We would like the size of this set to be as small as possible. We measure the performance of our algorithms by the size of the distance-k total p-dominating set returned by each algorithm, which is the response variable in our experiments. When $k = p = 1$, the set returned by each algorithm is a total dominating set.

4.3 Predictor Variables

We study five predictor variables, which are used to estimate the size of a distance-k total p-dominating set. The five predictors can be divided into three groups. The first group includes the predictors that describe the graph, the second group consists of the predictors that define the problem we consider, and the third group solely contains the algorithm which is used. The first group has two predictor variables: the number of vertices in the graph and the density of the graphs. The second group also has two predictors: the constants k, and p.

4.3.1 Number of Nodes

In our experiments, the number of nodes in a wireless sensor network reflects the scale of the network. The scale of our networks is as small as 75 nodes and as large as 375 nodes. For finding a total dominating set, that is, when $k = p = 1$, five levels of this predictor variable are used. The five levels are 75, 150, 225, 300 and 375. For finding a distance-k total p-dominating set, where k and p are not fixed at the value of 1, we use three levels for the number of nodes variable, namely, 75, 225 and 375.

4.3.2 Transmission Range

We change the maximum transmission range to vary the density of our graphs. One way to quantify the density of the graph is simply the number of edges in the graph. It is difficult to control the number of edges in the randomly generated connected unit disk graphs. Instead, we use the maximum transmission range. By varying the maximum transmission range of the sensors, we can increase or decrease the number of their neighbors. By increasing the maximum transmission range, we increase the number of vertices adjacent to each vertex, thereby increasing the number of edges in the graph. The opposite effect takes place when we decrease the maximum transmission range. The three levels of the transmission range that we use in our experiments are 15, 25, and 35 units.

4.3.3 Positive Integer k

The positive integer k indicates the maximum distance from a vertex to a member of a distance-k total p-dominating set. Since our graphs are not very large in size we do not consider large k. The levels of k we consider are $k = 1$, $k = 2$, and $k = 3$. Note that k essentially plays the same role as the graph density variable. Although k does not directly increase the number of neighbors for a given node v, it does increase the number of adjacencies associated with v. Increasing k to be larger than three will result in all algorithms essentially returning the same size of distance-k total p-dominating sets for small graphs. In such a case, it is difficult to distinguish among the performances of the algorithms from the experimental results.

4.3.4 Positive Integer p

The positive integer p indicates the number of vertices that dominate each vertex $v \in V$. For distance-k total p-dominating set if we increase p too much then almost the entire vertex set will be required to be in the $D_{k,p}$. Hence, the largest p value we use is 3. The three levels for p we use in our experiments are $p = 1$, $p = 2$, and $p = 3$.

4.3.5 Algorithms

We wish to compare the algorithms by determining whether the choice of the algorithm affects the size of the distance-k total p-dominating set. We compare three of our algorithms. The three levels of the algorithm variable are Random, Greedy, and Ran&Greedy. Unlike

the previously discussed predictor variables, the algorithm variable does not take on a quantitative value. It is a categorical variable, that is, its levels are much like categories.

It is desirable not to use the Algorithm as a predictor variable in our experiments but rather compare the sizes of $D_{k,p}$ returned by each algorithm to that of the optimum. However, since finding a distance-k total p-dominating set is NP-hard, finding the optimum solution is very difficult. Therefore, we use the Algorithm as a predictor variable.

4.4 Area

The variable Area is the square region where the coordinates of each node is plotted. The area is held constant relative to the number of nodes being plotted. For example, consider a network with 75 nodes plotted in an area A. If we increase the number of nodes, N, from 75 to 225 and keep A the same, the density of the network will increase. We do not use the Area variable to vary the density of the network since this is already done by the maximum transmission range. So not to do the same task twice, as we increase $N = 75$ to $N = 225$ we also increase the Area such that the density of the nodes for both values of N is roughly the same.

4.5 Methodology

We divide the multiple regression analysis of our experiments into two parts. In the first part we give the regression analysis for algorithms returning a total dominating set where $k = p = 1$. In the second part, we give the multiple regression analysis for finding a distance-k total p-dominating set where k and p are not fixed at one value. In this case, we would like to determine the effects k and p have on the size of the distance-k total p-dominating sets returned by the three algorithms. For both parts, we use a multiple regression model to determine the percentage of variation caused by each predictor variable and their interactions. This will give us a good indication of the relative importance of all predictor variables and their interactions. We are particularly interested in the Algorithm variable.

Each algorithm is run on 100 different graphs. That is, we have 100 replications. The 100 graphs are the same for all three algorithms. The graphs are connected unit disk graphs. The generation of these graphs is given by Jorgic et al. [23]. The idea behind the generation

of the graphs is as follows. The position of the first node is randomly generated. The remaining nodes are placed one by one. In iteration i a position of a node is generated at random. If this position is within a specified Euclidean distance of at least one of the previously generated nodes, then a node is placed at this position. Otherwise, a new random position is generated until an acceptable position is obtained.

4.6 Multiple Regression Analysis for Finding a Total Dominating Set

For finding a total dominating set, the predictor variables and all values assumed by each predictor used in the multiple regression model are shown in Table 4.1. Note again that k and p are held constant at the value of 1. So the multiple regression model is given by

$$y = \beta_0 + \beta_1 N + \beta_2 R + \beta_3 A + \beta_4 NR + \beta_5 NA + \beta_6 RA + \beta_7 NRA. \tag{4.4}$$

Predictor Variable	Level 1	Level 2	Level 3	Level 4	Level 5
N (Number of Nodes)	75	150	225	300	375
R (Maximum Transmission Range)	15	25	35	–	–
A (Algorithm)	Random	Greedy	Ran&Greedy	–	–

Table 4.1: Predictor variables and their levels used in the multiple regression model for finding a total dominating set.

The effects of all three predictors and their interactions on the size of the total dominating set are shown in Table 4.2. All numbers in Table 4.2 are percentages. The percentages of variation in Table 4.2 are the percentages of the total variation caused by the predictor variables and their interactions on the size of the total dominating set.

The sum of the variations of the primary effects and the first and the second order interactions indicates the variation explained by the regression model. The fraction of variation that is explained determines the goodness of the regression and is called the *coefficient of determination*, R^2, where $0 \leq R^2 \leq 1$. The higher the value of R^2, the better the regression. In our analysis for determining the size of the total dominating set, $R^2 = 0.9425$. Thus, the regression explains 94.25% of the variation of the size of the total dominating set. Note that

R^2 is not related to the maximum transmission range denoted as R. It is standard notation in the literature to use R^2 for the coefficient of determination.

The simulation results were analyzed using the `jmp` statistical software. The `jmp` software applies an F-test to verify the statistical significance of the values illustrated in Table 4.2. According to the F-test, all results in Table 4.2 that represent more than 0.5% of the change in the size of the total dominating set are statistically significant at a 95% confidence interval. We exclude the detailed analysis of the F-test. Instead we show the significant and the non-significant results at a 95% confidence interval of all the predictors and their interactions in Table 4.2.

Predictor Variable	Percentage of Variation
Primary Effects	84.82
N	33.19
R	49.76
A	1.87
First Order Interactions	9.36
N,R	8.58
N,A	*0.40*
R,A	*0.38*
Second Order Interactions	*0.06*
Error	5.75

Table 4.2: The effects of the predictor variables and their interactions on the size of the total dominating set.

The effects which are not significant in Table 4.2 are shown in italics. None of these effects account for more than 1%. The maximum transmission range and the number of vertices account for the most percentage of the total variation in determining the size of the total dominating set. The number of nodes accounts for 33.19% of the total variation of the size of the total dominating set. This is expected since the size of the total dominating set is directly dependant on the number of vertices in the graph. The more vertices we have in the graph, the larger we expect the size of the total dominating set to be. The maximum transmission range accounts for 49.76% of the total variation of the size of the total dominating set. The maximum transmission range, which we use to vary the density of the graph, determines the number adjacencies for each vertex. If each vertex has large degree, then the size of the total dominating set is bound to be smaller than if vertices have

small degree. The interaction between the number of nodes and the transmission range has the most effect among all interactions between the predictors. It is accountable for 8.58% of the total variation.

The algorithm does not have much of an effect on the total variation of the size of the total dominating set. It accounts for only 1.87% of the total variation. To look at this in more detail we observe the regression coefficients for the primary predictors. The regression coefficients for the three predictors is shown in Table 4.3.

Predictor Variable	Regression Coefficient
Intercept	$\beta_0 = 46.597$
N	$\beta_1 = 0.107$
R	$\beta_2 = -1.685$
A	$\beta_{31} = -6.557$ $\beta_{32} = -5.811$

Table 4.3: Regression coefficients for the predictor variables for finding a total dominating set.

The regression coefficient $\beta_0 = 46.597$ in Table 4.3 represents the average size of the total dominating set returned by the Random algorithm while N and R were varied. The regression coefficient $\beta_1 = 0.107$ represents the slope of N as R and A are held constant. The regression coefficient $\beta_2 = 1.685$ indicates the slope of R as N and A are held constant. Both indications of β_1 and β_2 are consistent with the effects shown in Table 4.2.

The regression coefficient that represents the algorithm consists of two parts. The regression coefficient β_{31} represents the difference between the average size of the total dominating set returned by the Random algorithm and the average size of the total dominating set returned by the Greedy algorithm. A negative value of β_{31} indicates that the Greedy algorithm performs better than the Random algorithm. The second regression coefficient, β_{32} represents the difference between the average size of the total dominating set returned by the Ran&Greedy algorithm and the average size of the total dominating set returned by the Random algorithm. A negative value of β_{32} indicates that the Ran&Greedy algorithm performs better than the Random algorithm. From $\beta_{31} = -6.557$ we observe that on average the size of the total dominating set returned by the Greedy algorithm is 6.557 less than that returned by the Random algorithm. From $\beta_{32} = -5.811$ we see that the size of the total dominating set returned by the Ran&Greedy algorithm on average is 5.811 less than that returned by

the Random algorithm.

To better understand why the algorithm accounts for only 1.87% of the total variation, we consider the multiple regression analysis for each phase of the presented algorithms. Recall that all three algorithms considered for experimental results have two phases. Phase one finds a dominating set S and phase two adds extra vertices to S to obtain a total dominating set.

4.6.1 Multiple Regression Analysis for Phase One

We show the results of the multiple regression analysis of phase one of the algorithms in Table 4.4. Insignificant results in Table 4.4 are shown in italics. We observe again that the maximum transmission range and the number of nodes have the most effect on the total variation of the size of the obtained dominating set at 50.916% and 30.817% respectively. Their interaction accounts for almost 9% of the total variation. Variable A on the other hand has essentially no effect on the size of the returned dominating set. It only accounts for 0.025% of the total variation. Note that all first and second interactions that involve the variable A are also insignificant.

Predictor Variable	Percentage of Variation
Primary Effects	84.76
N	33.817
R	50.916
A	*0.025*
First Order Interactions	9.05
N,R	8.991
N,A	*0.001*
R,A	*0.062*
Second Order Interactions	*0.01*
Error	6.175

Table 4.4: The effects of the predictor variables and their interactions on the size of the dominating set obtained in phase one.

We show the values of the regression coefficients of all predictors for the first phase of the algorithms in Table 4.5. Again $\beta_0 = 31.586$ is the average size of the dominating set returned in phase one by the Random algorithm. From the regression coefficients of $\beta_{31} = -0.463$

and $\beta_{32} = 0$ of the variable A, we observe that on average there is no difference in the size of the dominating set returned in phase one of all three algorithms. Recall that β_{32} is the regression coefficient which describes the difference between the Random and the Ran&Greedy algorithms in phase one. Both the Random and the Ran&Greedy algorithms have the same phase one. Thus, $\beta_{32} = 0$. From the regression coefficients we can say that the Greedy and the Ran&Greedy algorithms do better or as well as the Random algorithm. However, the difference is negligible. This suggests that the difference between the results returned by each algorithm comes mostly from phase two. Since the algorithms produce a result similar in size in phase one, we can assume a constant size of a dominating set and do only a multiple regression analysis for the results produced in phase two.

Predictor Variable	Regression Coefficient
Intercept	$\beta_0 = 31.586$
N	$\beta_1 = 0.075$
R	$\beta_2 = -1.150$
A	$\beta_{31} = -0.463$ $\beta_{32} = 0$

Table 4.5: Regression coefficients for the predictor variables in phase one for finding a dominating set.

4.6.2 Multiple Regression Analysis of Phase Two

We give the multiple regression analysis of phase two in Table 4.6. As predicted earlier, the difference between the results returned by the three algorithms results from phase two of each algorithm. Insignificant effects in Table 4.6 are in italics. The variables N and R together account for about 70.46% of the total variation. The algorithm accounts for 12.02% of the total variation. The first order interactions are important as well since they all together account for 11.45% of the total variation. It is only the second order interactions which are insignificant and account for only 0.28%, less than 1% of the total variation of the size of the total dominating set.

The regression coefficients of the three predictor variables are shown in Table 4.7. The positive value of β_1 indicates a positive slope for N as R and A are held constant. The negative value of β_2 indicates a negative slope for R as N and A are held constant. That

Predictor Variable	Percentage of Variation
Primary Effects	82.47
N	28.33
R	42.13
A	12.02
First Order Interactions	11.45
N,R	6.95
N,A	2.68
R,A	1.82
Second Order Interactions	*0.28*
Error	5.80

Table 4.6: The effects of the predictor variables and their interactions on the size of the total dominating set in phase two.

Predictor Variable	Regression Coefficient
Intercept	$\beta_0 = 15.011$
N	$\beta_1 = 0.031$
R	$\beta_2 = -0.535$
A	$\beta_{31} = -6.093$
	$\beta_{32} = -5.811$

Table 4.7: Regression coefficients for the predictor variables in phase two.

is, as R increases the number of vertices added in phase two decreases since the graphs become denser. The regression coefficient $\beta_0 = 15.011$ represents the average number of vertices added in phase two by the Random algorithm. The negative value of $\beta_{31} = -6.093$ indicates that on average the number of vertices returned by the Greedy algorithm in phase two is 6.093 less than that returned by the Random algorithm. The value of $\beta_{32} = -5.811$ indicates that on average the number of vertices added in phase two by the Ran&Greedy algorithm is 5.811 less than that added by the Random algorithm.

4.7 Multiple Regression Analysis for a Distance-k Total p-Dominating Set

In this section we give a multiple regression analysis for finding a distance-k total p-dominating set for positive integers k and p. For finding a distance-k total p-dominating set the predictors and their assumed values for the multiple regression model are shown in Table 4.8. The multiple regression model considered includes all predictors and all of their first, second, third and fourth order interactions.

Predictor Variable	Level 1	Level 2	Level 3
N (Number of Nodes)	75	225	375
R (Maximum Transmission Range)	15	25	35
k	1	2	3
p	1	2	3
A (Algorithm)	Random	Greedy	Ran&Greedy

Table 4.8: Predictor variables and their levels for $D_{k,p}$.

The effects of all predictors and their interactions on the size of the distance-k total p-dominating set is shown in Table 4.9. All numbers in Table 4.9 are percentages that indicate the percentage of the total variation caused by each predictor and their interactions on the size of the distance-k total p-dominating set.

The primary effects as well as their interactions explain the regression model with $R^2 = 0.8390$. Thus, the regression explains 83.90% of the variation of the size of the distance-k total p-dominating set.

Nonsignificant effects in Table 4.9 are shown in italics and do not account for more than 1% of the variation. The primary effects account for 65.466% of the total variation. Much like the analysis for finding a total dominating set, the number of nodes and the maximum transmission range have the largest effects in determining the size of the distance-k total p-dominating set. The variable N accounts for 17.984% of the total variation and R accounts for 24.259% of the total variation. Their interaction is significant as well at 4.886%. For finding a distance-k total p-dominating set, note that the percentage of variation for N and R has decreased from that of finding a total dominating set in the previous section. The reason for this is the addition of the k and p variables.

Predictor Variable	Percentage of Variation
Primary Effects	65.466
N	17.984
R	24.259
k	13.151
p	9.875
A	*0.196*
First Order Interactions	16.527
N,R	4.866
N,A	*0.038*
N,k	2.887
N,p	1.977
R,A	*0.057*
R,k	2.652
R,p	2.286
k,p	1.705
k,A	*0.022*
p,A	*0.015*
Second Order Interactions	1.817
Third Order Interactions	*0.089*
Fourth Order Interactions	*0.00027*
Error	16.10

Table 4.9: The effects of the predictor variables and their interactions on the size of distance-k total p-dominating set.

Integers k and p also have a large effect on the size of the distance-k total p-dominating set. Integer k accounts for 13.151% of the total variation. As explained earlier, k plays a similar role to the maximum transmission range. Hence, it is expected for k to have a large effect on the size of the distance-k total p-dominating set. As k increases, the number of adjacencies associated with each node increases, thereby decreasing the size of the sought distance-k total p-dominating set. Integer k and R both affect size of the distance-k total p-dominating set and their interaction is significant 2.652%. The interaction between integer k and the variable N is significant as well since both k and N affect the size of the distance-k total p-dominating set. This interaction is accountable for 2.887%.

Integer p is accountable for 9.875% of the total variation. As we increase p, the size of the distance-k total p-dominating set increases. However, this increase in size of the distance-k

total p-dominating set is not solely dependent on p. It also depends on the variables N, R and k. We see all three interactions between N, R, k and p individually are significant. The interaction between N and p accounts for 1.977% of the total variation. The effect of k and p is 1.705%. The effects of R and p is the largest of the three interactions is accountable for 2.886%.

The first order interactions account for more than 16% of the total variation. The second order interactions account for more than 1% of the total variation. Most of this effect comes from the interactions between R, p, k; N, p, k; and R, N, k. The third and fourth order interactions both account for less than 1% of the total variation and are insignificant.

As we saw in the analysis of the total dominating set, the algorithm does not have much of an effect on the size of the total dominating set. The same is true for distance-k total p-dominating set. It is accountable for only 0.196% of the total variation of the size of the distance-k total p-dominating set and is considered insignificant. To understand the average difference between the algorithms, we observe the regression coefficients of the multiple regression model. Regression coefficients for all five predictor variables is shown in Table 4.10.

Predictor Variable	Regression Coefficient
Intercept	$\beta_0 = 43.281$
N	$\beta_1 = 0.087$
R	$\beta_2 = -1.494$
k	$\beta_3 = -11.298$
p	$\beta_4 = 10.357$
A	$\beta_{51} = -2.752$ $\beta_{52} = -2.010$

Table 4.10: Regression coefficients for the predictor variables for finding a distance-k total p-dominating set.

The regression coefficient $\beta_0 = 43.281$ represents the average size of the distance-k total p-dominating set returned by the Random algorithm for all possible N, R, k, and p values. The positive value of $\beta_1 = 0.087$ indicates that as N increases the size of the distance-k total p-dominating set increases. The regression coefficient $\beta_2 = -1.494$ indicates that as R increases the size of the distance-k total p-dominating set decreases. The regression coefficient $\beta_3 = -11.298$ represents the slope of k. As k increases the size of the distance-k

total p-dominating set decreases. The coefficient β_4 represents the slope of p while other predictors are held constant. From $\beta_4 = 10.357$ we observe that as p increases the size of the distance-k total p-dominating set increases as well. The variable A has two regression coefficients associated with it. The regression coefficient $\beta_{51} = -2.752$ implies that on average the size of the distance-k total p-dominating set returned by the Random algorithm is 2.752 more than that returned by the Greedy algorithm. From β_{52} we observe that on average the size of the distance-k total p-dominating set returned by the Ran&Greedy algorithm is 2.01 less than that returned by the Random algorithm. From both β_{51} and β_{52} we can conclude that both the Ran&Greedy and Greedy algorithms on average outperform the Random algorithm. However, the difference shown by the multiple regression analysis is small.

To understand in detail why the algorithm is accountable for only 0.196% of the total variation of the size of the distance-k total p-dominating set, we will consider the multiple regression analysis for each phase of the algorithms separately.

Before we discuss each phase separately, we discuss the error term in the multiple regression model for finding a distance-k total p-dominating set. From Table 4.9 we observe that the error is accountable for 16.10% of the total variation. This is too large to be due to randomness in our data. The reason for a large error is because our multiple regression model does not consider several interactions. Notable such interactions are R^2, k^2, p^2 and all second, third, and fourth order interactions that include the three squared interactions. The R^2 interaction is explained as follows.

Consider the multiple regression model where R^2 is not added. To observe how the size of the distance-k total p-dominating set changes as R is changed, we keep all predictor variables except R constant. In such a case, all terms are either constant or they are in terms of R. Thus, the size of the distance-k total p-dominating set is a linear function of R. Now, introduce R^2 into the model. We keep all predictors but R constant. Thus, all terms are either constant, in terms of R or in terms of R^2. From this we can conclude that the size of the distance-k total p-dominating set is a quadratic function of R. Other interactions are explained similarly.

The reason the size of the distance-k total p-dominating set is a quadratic function of the transmission range is as follows. For a given vertex v, v dominates the vertices within its transmission range R as other predictors are held constant. In other words, v covers the disk with an area πR^2. As R increases, the area covered by v increases. Therefore, the size

of the distance-k total p-dominating set decreases quadratically. Other predictors that affect the size of the distance-k total p-dominating set quadratically have similar explanations.

None of the effects shown in Table 4.9 are changed when interactions in terms of R^2, k^2 and p^2 are added into the multiple regression model. We omit the detailed analysis of this new model. Note that N^2 or A^2 are not added into the new model because they do not have any effect on the new model. From our analysis we observed that the error term reduces considerably. After the addition of the new interactions the error is only accountable for 3.53% of the total variation. The regression in this case explains not 83.90% of the variation of the size of the distance-k total p-dominating set, but rather 96.47% of the variation.

4.7.1 Multiple Regression Analysis for Phase One

We show the results of the multiple regression analysis of phase one of the algorithms in Table 4.11. Insignificant results in Table 4.11 are shown in italics. The primary effects account for 62.294%. The variables R and N have the most effect on the size of the distance-k total p-dominating set. The variable R is accountable for 20.575% of the total variation and N accounts for 15.903% of the total variation. The integers k and p account for 11.463% and 14.336% of the total variation respectively. All second order interactions involving the variables R, N, k, and p are significant. The variable A accounts for 0.017% of the total variation and is insignificant. All interactions involving A are also insignificant.

We show the values of the regression coefficients of all predictors for phase one in Table 4.12. Again $\beta_0 = 31.474$ is the average size of the distance-k total p-dominating set returned in phase one by the Random algorithm. From the regression coefficients $\beta_{51} = -0.631$ and $\beta_{52} = 0$ of the variable A we conclude that on average there is no difference between the Ran&Greedy and Random algorithms and between the Greedy and Random algorithms. From β_{51}, the Greedy algorithm returns a distance-k total p-dominating set of size only 0.631 less than that returned by the Random algorithm. The regression coefficient β_{52} indicates that the Ran&Greedy algorithm returns the same size of a distance-k total p-dominating set as the Random algorithm. This is the case since phase one of both algorithms is the same. Since the three algorithms produce a result similar in size in phase one, we can assume a constant size of a dominating set and do only a multiple regression analysis for the results produced in phase two.

Note again that the error term accounts for 15.21% of the total variation. Again, this does not include the terms R^2, k^2, or p^2 in the multiple regression model. After the inclusion

Predictor Variable	Percentage of Variation
Primary Effects	62.294
N	15.903
R	20.575
k	11.463
p	14.336
A	*0.017*
First Order Interactions	19.456
N,R	4.247
N,A	*0.002*
N,k	2.528
N,p	3.155
R,A	*0.016*
R,k	2.530
R,p	3.679
k,p	3.299
k,A	*0.00002*
p,A	*0.00039*
Second Order Interactions	2.860
Third Order Interactions	*0.179*
Fourth Order Interactions	*0.00005*
Error	15.21

Table 4.11: The effects of the predictor variables and their interactions on the size of the distance-k total p-dominating set in phase one.

of these terms, the error reduces to 3.39%.

Predictor Variable	Regression Coefficient
Intercept	$\beta_0 = 31.474$
N	$\beta_1 = 0.074$
R	$\beta_2 = -1.234$
k	$\beta_3 = -9.604$
p	$\beta_4 = 10.646$
A	$\beta_{51} = -0.631$ $\beta_{52} = 0$

Table 4.12: Regression coefficients for the predictor variables for finding a distance-k total p-dominating set in phase one of the algorithms.

4.7.2 Multiple Regression Analysis for Phase Two

We give the multiple regression analysis of phase two in Table 4.13. Insignificant effects in Table 4.13 are shown in italics. The variable R, N and k account for the most effect on the size of the distance-k total p-dominating set. The effects for R, N and k are 25.645%, 15.718%, and 12.310% respectively. The first order interactions between these three variables are also significant. Note that integer p in phase two of the algorithms is only accountable for 1.119% of the total variation. The reason for this is that by the end of phase one almost all vertices in the distance-k total p-dominating set are already either dominated p times or $p-1$ times. All first order interactions involving the variable p are insignificant. The variable A accounts for 4.124% of the total variation. Although this effect is not high, we still can conclude that the difference between the algorithms is determined by phase two. The algorithm does not have a large effect on the size of the distance-k total p-dominating set and all first order interactions involving the variable A are insignificant.

The regression coefficients for determining the number of vertices added to the distance-k total p-dominating set in phase two are given in Table 4.14. On average, the number of vertices added to the distance-k total p-dominating set in phase two by the Random algorithm is given by $\beta_0 = 11.807$. The regression coefficients of $\beta_{51} = -2.121$ and $\beta_{52} = -2.019$ suggest a small difference between the algorithms. On average the number of vertices added to the distance-k total p-dominating set in phase two by the Greedy algorithm is 2.121 less than that of added by the Random algorithm. From β_{52} we can conclude that on average the number of vertices added to the distance-k total p-dominating set in phase two by the Ran&Greedy algorithm is 2.019 less than that of the Random algorithm.

We note that after including the terms R^2, k^2, p^2 and all other second, third, and fourth order interactions involving these terms, the error term reduces from 20.16% to 9.84%. At the beginning of phase two the vertices that act as the input graph for the three algorithms are different. Since we do not consider phase one when we analyze phase two, the analysis of phase two is a comparison of three algorithms with different graphs as input. The size of these graphs may be different as well. Thus, the large error is possibly due to the data not considered in phase one with phase two.

Predictor Variable	Percentage of Variation
Primary Effects	58.915
N	15.718
R	25.645
k	12.310
p	1.119
A	4.124
First Order Interactions	15.965
N,R	4.632
N,A	*0.894*
N,k	2.642
N,p	*0.752*
R,A	*0.787*
R,k	1.536
R,p	0.946
k,p	2.585
k,A	*0.661*
p,A	*0.532*
Second Order Interactions	4.389
Third Order Interactions	*0.563*
Fourth Order Interactions	*0.010*
Error	20.16

Table 4.13: The effects of the predictor variables and their interactions on the size of the distance-k total p-dominating set in phase two.

Predictor Variable	Regression Coefficient
Intercept	$\beta_0 = 11.907$
N	$\beta_1 = 0.013$
R	$\beta_2 = -0.260$
k	$\beta_3 = -1.694$
p	$\beta_4 = -0.286$
A	$\beta_{51} = -2.121$ $\beta_{52} = -2.019$

Table 4.14: Regression coefficients for the predictor variables for finding a distance-k total p-dominating set in phase two.

Chapter 5

Experimental results

In Chapter 4 we used a multiple regression model to determine the effect of all predictors on the size of the distance-k total p-dominating set returned by each of the three algorithms. In this chapter, we further study the performance of the three algorithms with respect to the predictor variables discussed in Chapter 4.

In order to examine the performance of our algorithms in terms of the size of the distance-k total p-dominating set, we compare the responses returned by each algorithm. Given two algorithms, this is done by taking the difference between the responses produced by the two algorithms. The three algorithms we considered in our analysis in Chapter 4 were Random, Greedy and Ran&Greedy. We consider all three differences among the three algorithms, that is, we look at the difference between Random and Greedy, the difference between Random and Ran&Greedy, and the difference between Greedy and Ran&Greedy.

Each algorithm consists of two phases. We would like not only to observe the difference in the sizes of the distance-k total p-dominating sets returned by each algorithm, but also look at the difference between the number of vertices returned separately in phase one and in phase two of each algorithm.

The difference between the responses of the Random and the Greedy algorithms allows us to investigate both phases of the algorithms. The difference between the responses of Random and Ran&Greedy shows us the difference in phase two since phase one of both algorithms produces the same result. The third difference between Greedy and Ran&Greedy shows us the difference between the two algorithms in phase one. That is, how well does phase one perform and the effect it has on phase two.

We examine each of the above three cases separately. For each case, we consider each

combination of the specific levels of all predictor variables separately.

5.1 Difference Between Random and Greedy

This section discusses the difference between the responses returned by the Random and Greedy algorithms on the same 100 randomly generated connected unit disk graphs that were used in Chapter 4. We use $diff_{Random-Greedy}$ to denote the size of the distance-k total p-dominating set returned by the Random algorithm minus the distance-k total p-dominating set returned by the Greedy algorithm for each of the 100 graphs.

The difference $diff_{Random-Greedy}$ is shown in Figure 5.1 for $k = p = 1$. Each chart plots $diff_{Random-Greedy}$ against the graph number. Each data point in a given chart represents the $diff_{Random-Greedy}$ for a single graph. There are three sets of points in each chart. Each set represents the values of $diff_{Random-Greedy}$ at a different level of the variable R for the given value of the variable N. The three charts are drawn for the three levels of the network size N.

It is apparent in Figure 5.1 that the $diff_{Random-Greedy}$ are almost all positive for all values of N and R. This means that the Greedy algorithm almost all the time returns a distance-k total p-dominating set of smaller size than the Random algorithm.

The graph size N has a large impact on $diff_{Random-Greedy}$. Figure 5.1 shows that for a fixed value of R the difference increases as N increases. We can deduce that the Random algorithm returns more vertices than the Greedy algorithm. Thus, the Greedy algorithm does much better in larger graphs.

The three sets of points in each graph in Figure 5.1 show $diff_{Random-Greedy}$ for different values of R. As R increases, we see from Figure 5.1 that $diff_{Random-Greedy}$ decreases. The larger R is, the denser the graphs are since every node is expected to have more neighbors. As the graphs get dense, they get close to becoming a complete graph. In a complete graph, we cannot distinguish the difference between the performance of the two algorithms since they both will return the same result. This can be especially seen in the first chart of Figure 5.1 where $N = 75$. For $R = 35$ the difference between the two algorithms is very small. Note that all results produced in Figure 5.1 are consistent with the multiple regression analysis discussed in Chapter 4.

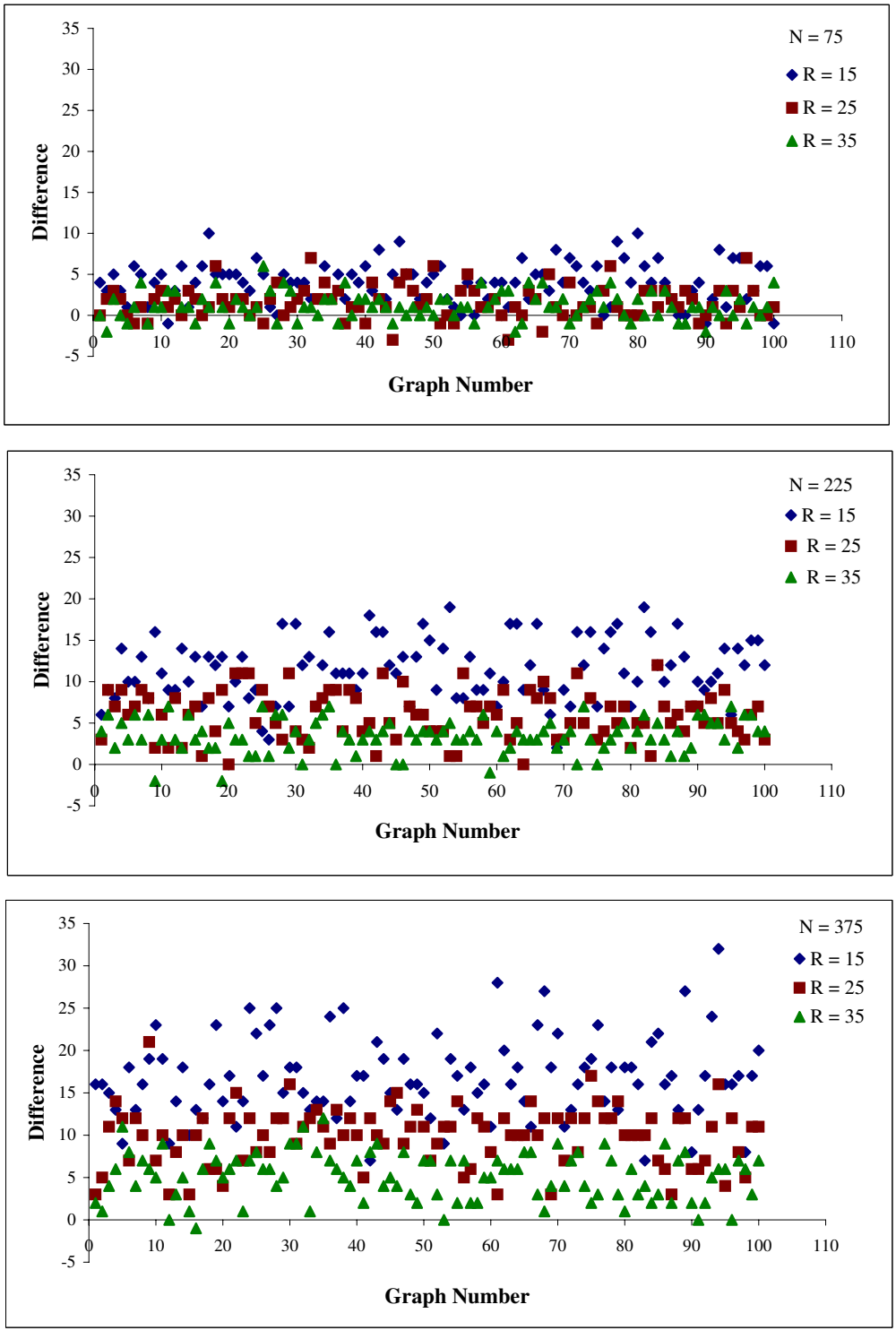


Figure 5.1: The difference $diff_{Random-Greedy}$ as N and R are varied.

In Chapter 4 we analyzed each phase of the algorithm separately using the multiple regression model. This was done to see which part of the algorithms was causing the most difference. To see the difference between the Random and Greedy algorithms in more detail, we consider the two phases of each algorithm separately.

5.1.1 Difference Between Random and Greedy in Phase One

In this section we consider the difference between the two algorithms in phase one. Figure 5.2 shows the difference between the Random and Greedy algorithms in terms of the number of vertices returned in phase one.

The trend shown in Figure 5.2 from chart to chart is similar to that of shown in Figure 5.1. The values of $diff_{Random-Greedy}$ in Figure 5.2 are smaller than those in Figure 5.1. The $diff_{Random-Greedy}$ values are almost equally dispersed above and below the x-axis. We can deduce that the two algorithms perform similarly.

The graph size N has a significant impact on $diff_{Random-Greedy}$. Figure 5.2 shows that for a fixed value of R $diff_{Random-Greedy}$ increases as N increases. This difference can be seen when R is fixed at the value of 15. For $R = 25$ and $R = 35$, the increase in the difference between the two algorithms is negligible. Thus, we can conclude for large graphs that the Greedy algorithm outperforms the Random algorithm.

It is apparent in Figure 5.2 that as R increases, $diff_{Random-Greedy}$ decreases. The difference between the algorithms on average becomes very close to 0. This can be seen for all N in Figure 5.2. Our reason for this is similar to that explained in Section 5.1. In Chapter 4 we concluded that the difference between the Random and Greedy algorithms in phase one is negligible. From the regression coefficients we saw that the Greedy algorithm on average did a little better than Random. This can clearly be seen in Figure 5.2 when N is varied for a fixed $R = 15$. However, for other values of R and N, the difference is very small on average.

5.1.2 Difference Between Random and Greedy in Phase Two

In this section we discuss the difference between the Random and Greedy algorithms in phase two. The difference between the two algorithms in phase two is given in Figure 5.3.

It is apparent in Figure 5.3 that the $diff_{Random-Greedy}$ are almost all positive for all values of N and R except for few data points when $N = 75$ in the first chart. The trend as

well as the values for $diff_{Random-Greedy}$ shown in Figure 5.3 from chart to chart is similar to that shown in Figure 5.1.

The graph size N has a large impact on $diff_{Random-Greedy}$. Figure 5.3 shows that for a fixed value of R the difference increases as N increases. This is clearly seen for all three values of N. Thus, for large graphs, the Greedy algorithm outperforms the Random algorithm in phase two.

As R increases, we see from Figure 5.3 that $diff_{Random-Greedy}$ decreases. The pattern in Figure 5.3 is similar to that in Figure 5.1. Figure 5.3 confirms the results of the multiple regression analysis in Chapter 4 for phase two of the algorithms. We can conclude that overall, the Greedy algorithm performs better than the Random algorithm in phase two of each algorithm. From Figures 5.1, 5.2 and 5.3 it is clear that it is phase two of the Random and Greedy algorithms that largely determines the final size of the distance-k total p-dominating set.

5.2 Difference Between Random and Ran&Greedy

In this section we give similar results comparing the Random algorithm with the Ran&Greedy algorithm. We use $diff_{Random-Ran\&Greedy}$ to denote the size of the distance-k total p-dominating set returned by the Random algorithm minus the distance-k total p-dominating set returned by the Ran&Greedy algorithm for each of the 100 graphs. Phase one of both algorithms is the same. Thus, the difference in the size of the distance-k total p-dominating sets returned by the algorithms is the difference in the numbers of vertices added to the distance-k total p-dominating set in phase two of both algorithms. The difference $diff_{Random-Ran\&Greedy}$ is shown in Figure 5.4 for $k = p = 1$ and the three levels of the network size N. The charts in Figure 5.4 are set up similarly to Figure 5.1. The analysis is similar to that discussed in Section 5.1.

It is apparent in Figure 5.4 that $diff_{Random-Ran\&Greedy}$ is positive for all values of N and R except for several values when $N = 75$ in the first chart. We can conclude that for the fixed values of $k = p = 1$, phase two solely determines that the Ran&Greedy algorithm outperforms the Random algorithm.

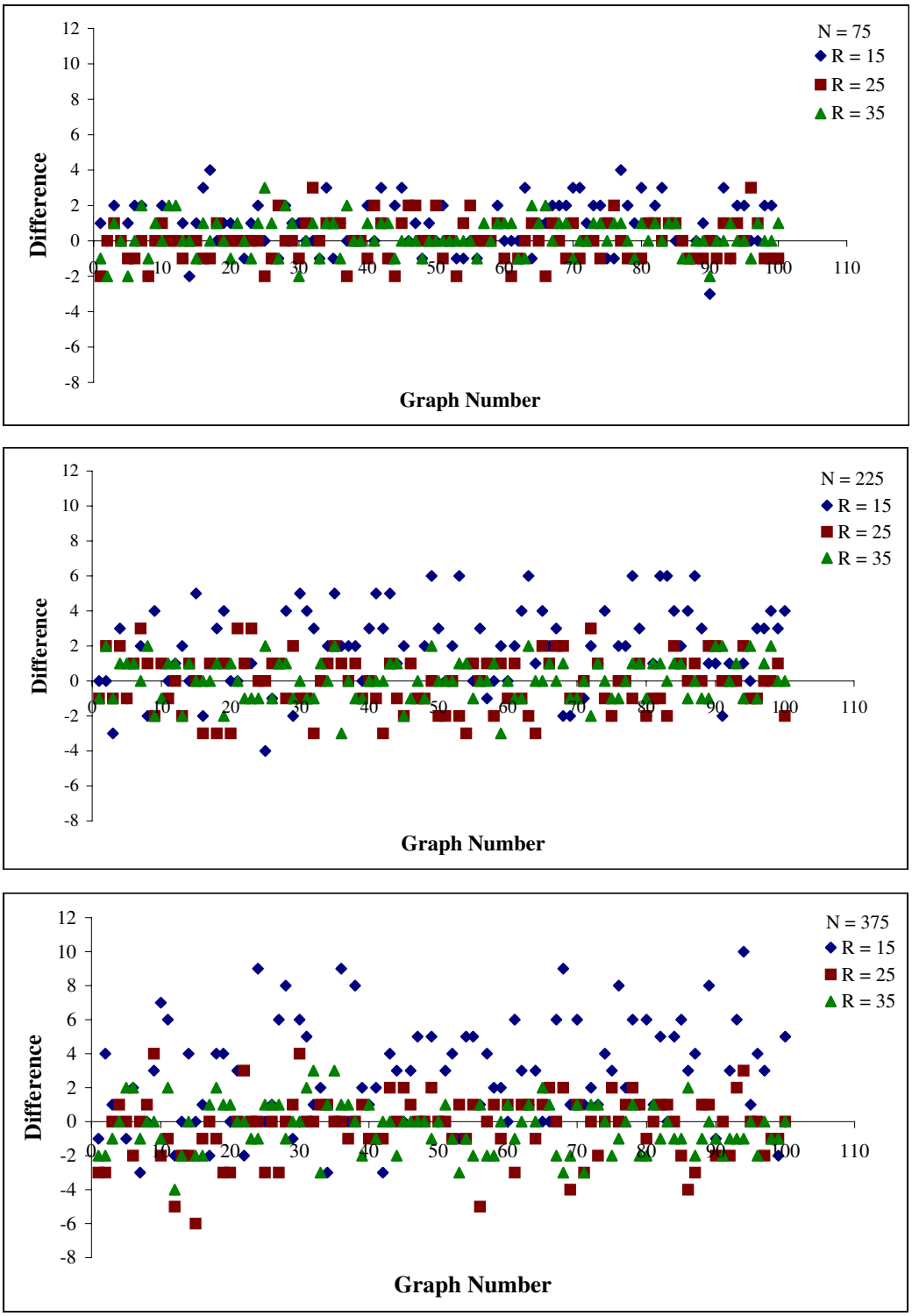


Figure 5.2: The difference $diff_{Random-Greedy}$ as N and R are varied in phase one.

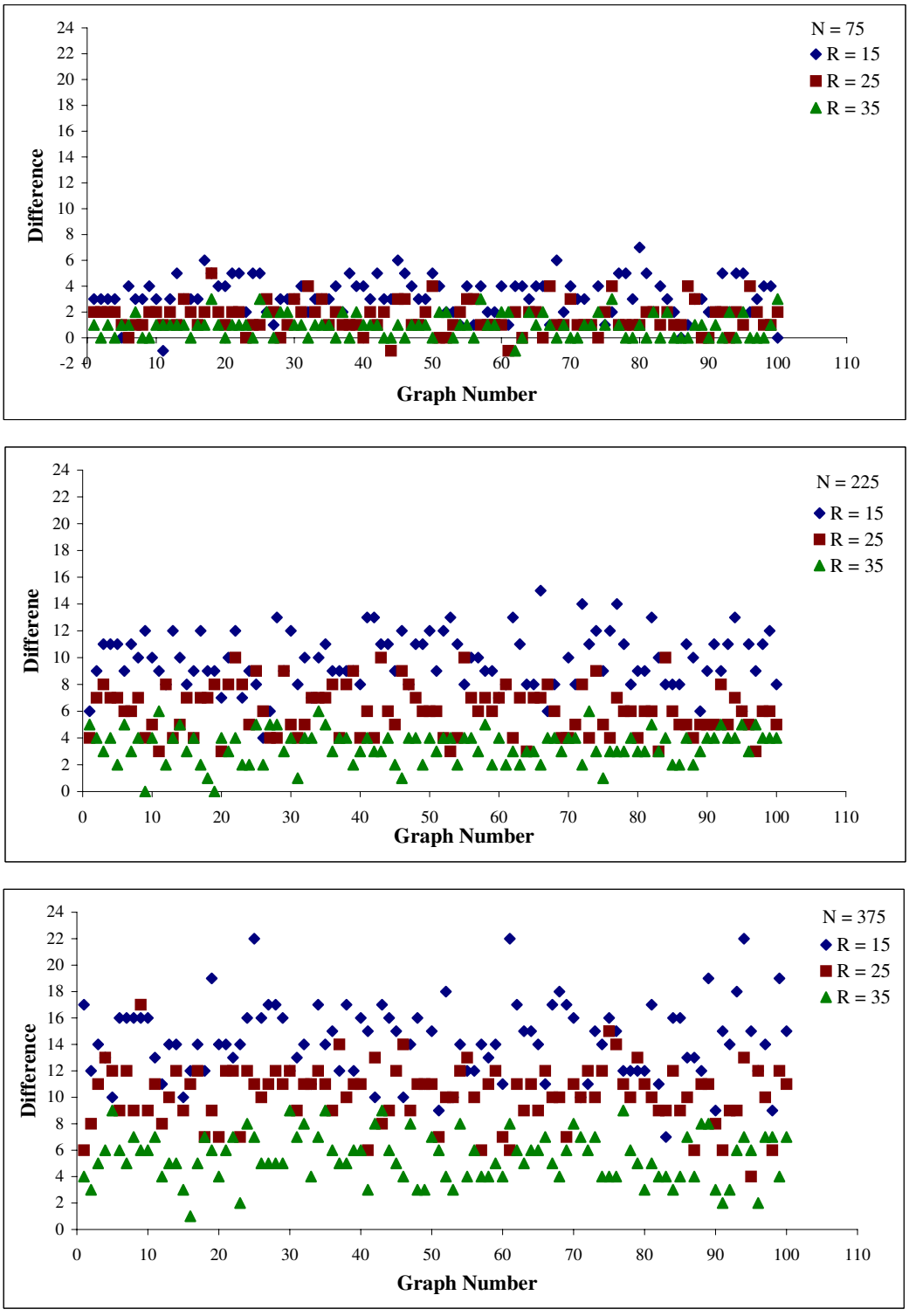


Figure 5.3: The difference $diff_{Random-Greedy}$ as N and R are varied in phase two.

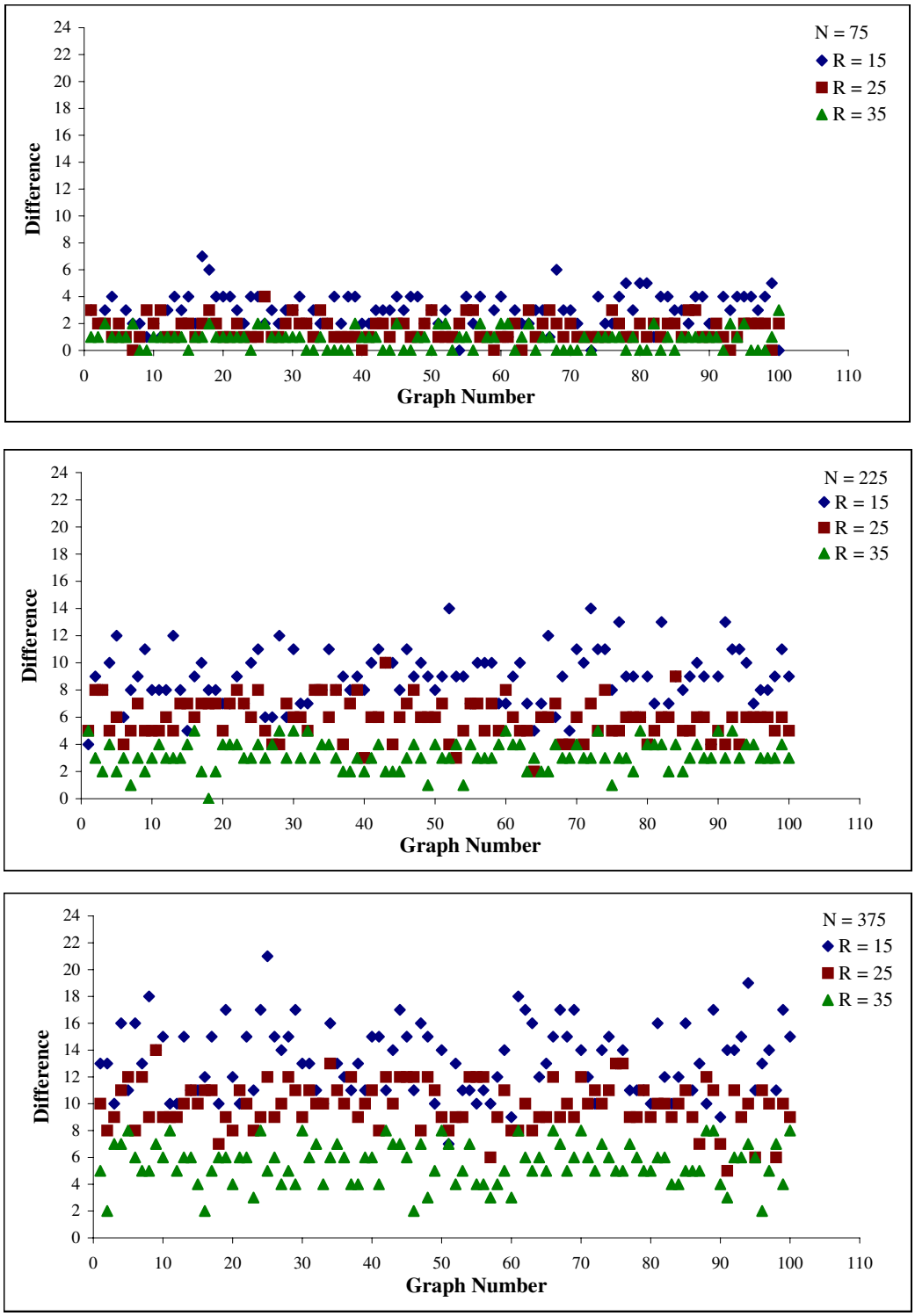


Figure 5.4: The difference $diff_{Random-Ran\&Greedy}$ as N and R are varied.

5.3 Difference Between Ran&Greedy and Greedy

In this section we compare the Ran&Greedy and the Greedy algorithms. The difference between the size of the distance-k total p-dominating set returned by the Ran&Greedy and the Greedy algorithm is denoted by $diff_{Ran\&Greedy-Greedy}$. The difference $diff_{Ran\&Greedy-Greedy}$ is shown in Figure 5.5 for three levels of the network size N. Figure 5.5 is set up in a similar fashion to Figure 5.1.

It is apparent from Figure 5.5 that the $diff_{Ran\&Greedy-Greedy}$ are equally dispersed above and below the x-axis for $R = 25$ and $R = 35$ for all three levels of N. This suggests that for dense graphs both algorithms perform similarly. When $R = 15$, for all levels of N, $diff_{Ran\&Greedy-Greedy}$ is mostly positive for all graphs. As N increases the difference $diff_{Ran\&Greedy-Greedy}$ increases. This suggests that the Greedy algorithm does better in sparser graphs than the Ran&Greedy algorithm.

To see the difference between the algorithms in more detail we consider the difference between the two algorithms in the two phases separately. Note that the difference between Ran&Greedy and Greedy in phase one has already been discussed in section 5.1.1. Thus we only observe the difference between the two algorithms in phase two in Figure 5.6.

The analysis of phase two is similar to that of phase one and phase two together. It is apparent from Figure 5.6 that the $diff_{Ran\&Greedy-Greedy}$ are equally dispersed above and below the x-axis for all values of R and N for all three levels of N. This suggests that the difference between the two algorithms in phase two is negligible. Hence, it is phase one that determines the small difference between the Ran&Greedy and Greedy algorithms. This is perhaps expected since phase two of both algorithms operate in a greedy manner.

5.4 Differences Between the Three Algorithms for all Values of k and p

An analysis similar to that in Sections 5.1, 5.2 and 5.3 for the differences between the algorithms can be obtained for all levels of k and p. We omit the detailed analysis here for all the differences between the algorithms. Instead, for given values of k, p, N and R, we consider the range of the obtained differences in the responses between the algorithms. We use $range_{A-B}$ to denote the range of the differences between two algorithms A and B. The individual differences in the responses when comparing two given algorithms on the

100 graphs falls into this range. We plot this range for all values of k, p, N and R for the differences between the Random and the Greedy algorithms, for the difference between the Random and the Ran&Greedy algorithms, and for the difference between the Ran&Greedy and the Greedy algorithms.

5.4.1 Difference Between Random and Greedy

Figure 5.7 shows the $range_{Random-Greedy}$ for the differences between the Random and Greedy algorithms. The $range_{Random-Greedy}$ is plotted against three predictor variables, namely R, k and p. Each level of R contains all levels of k and each level of k contains all levels of p. Therefore, each bar in Figure 5.7 is the $range_{Random-Greedy}$ for fixed values of k, p and R. The line through each bar represent the median of the given interval. The three charts in Figure 5.7 represent the three levels of the network size N.

We observe the change in $range_{Random-Greedy}$ as the predictor p is increased. As p increases, we observe a decrease or almost no change in the $range_{Random-Greedy}$ for small values of R and k. The opposite effect takes place when $N = 375$ for the values of $R = 35$ and $k = 2$, $k = 3$ as well as for $R = 25$ and $k = 3$. In these cases, as p increases, the $range_{Random-Greedy}$ increases. In large graphs, an increase in p results in a larger difference between the two algorithms.

As k increases, the maximum and minimum values for the $range_{Random-Greedy}$ decreases or we observe only a negligible change in the $range_{Random-Greedy}$. As k increases, the more adjacencies each node has associated with it. Therefore, each node in the distance-k total p-dominating set will potentially dominate more nodes. As a result, the size of the distance-k total p-dominating set decreases. The density of the graph is indirectly influenced by k and in such a case the Random algorithm is expected to return a good result. Thus, the obtained difference between the responses decreases and as a result, the $range_{Random-Greedy}$ decreases as well.

The decrease in the $range_{Random-Greedy}$ as k increases can be seen for $N = 225$ and $N = 375$. We can clearly see the decrease of the $range_{Random-Greedy}$ as k increases for $N = 225$ and $N = 375$ by observing the values of the medians of each bar as well. For $N = 75$, as k increases we observe the same phenomenon as for $N = 225$ and $N = 375$. However, for $N = 75$, the minimum value of the $range_{Random-Greedy}$ is rarely negative. It is only negative for large values of k and p. Similar results are observed for the values of the medians.

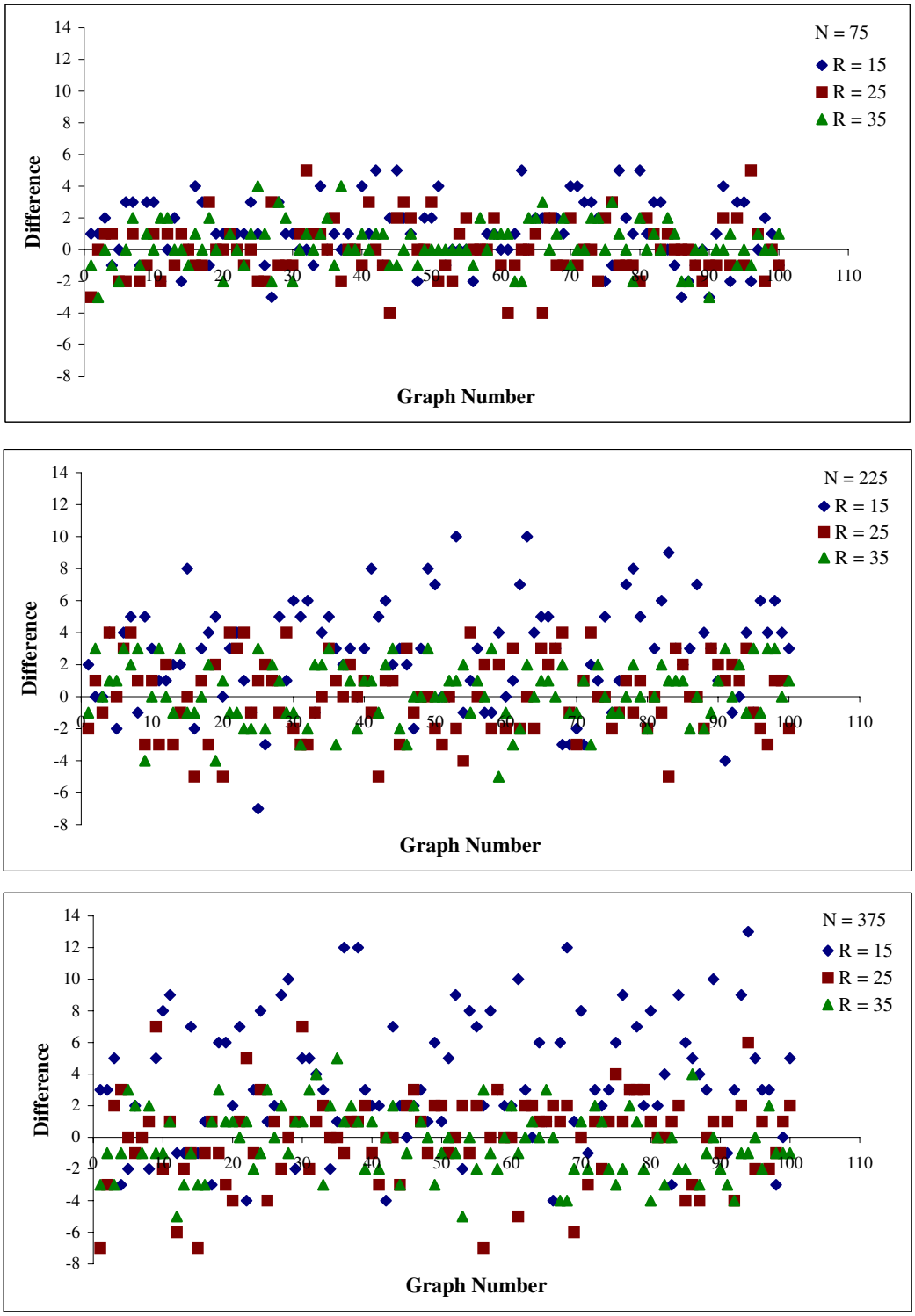


Figure 5.5: The difference $diff_{Ran\&Greedy-Greedy}$ as N and R are varied.

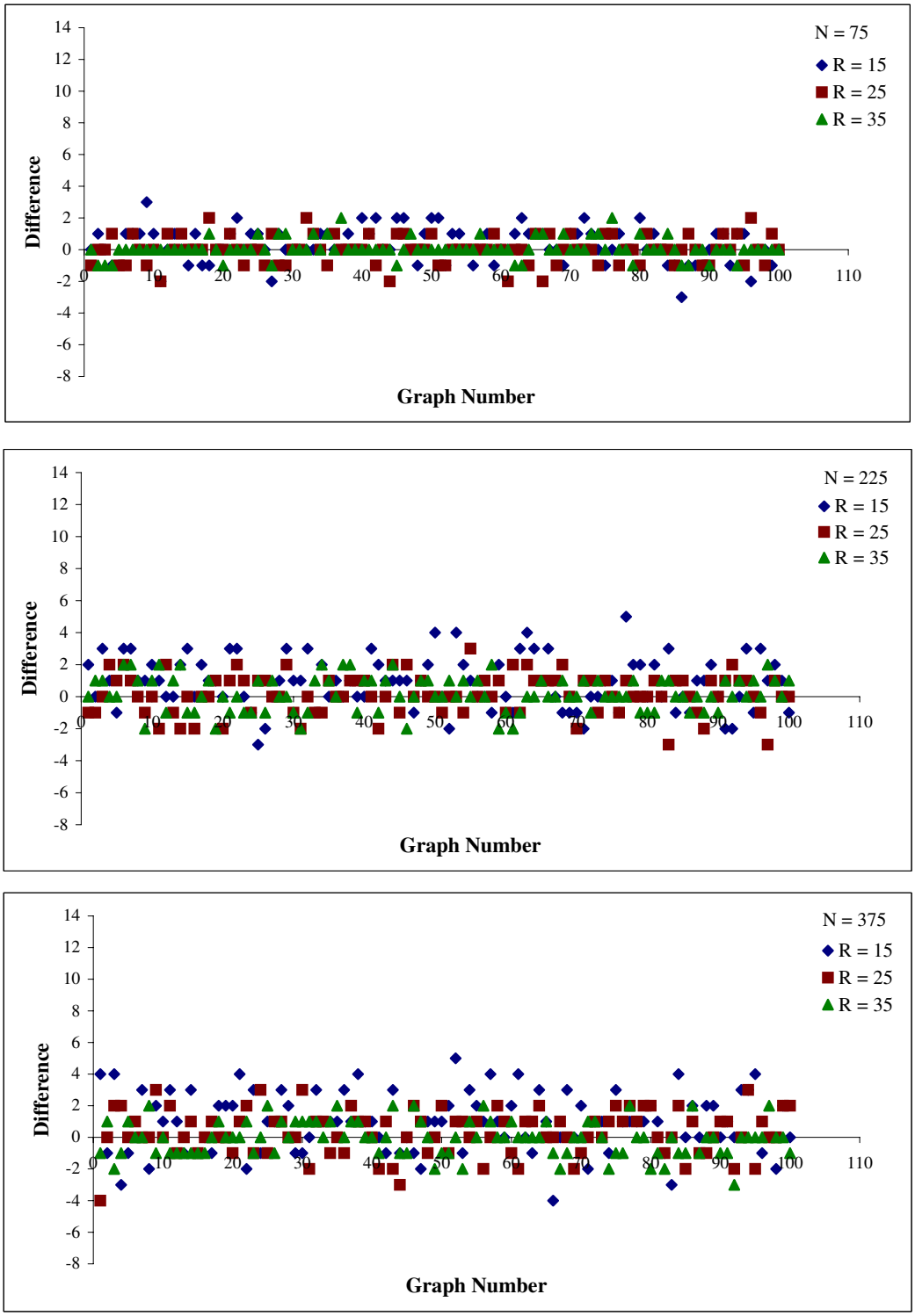


Figure 5.6: The difference $diff_{Ran\&Greedy-Greedy}$ as N and R are varied in phase two.

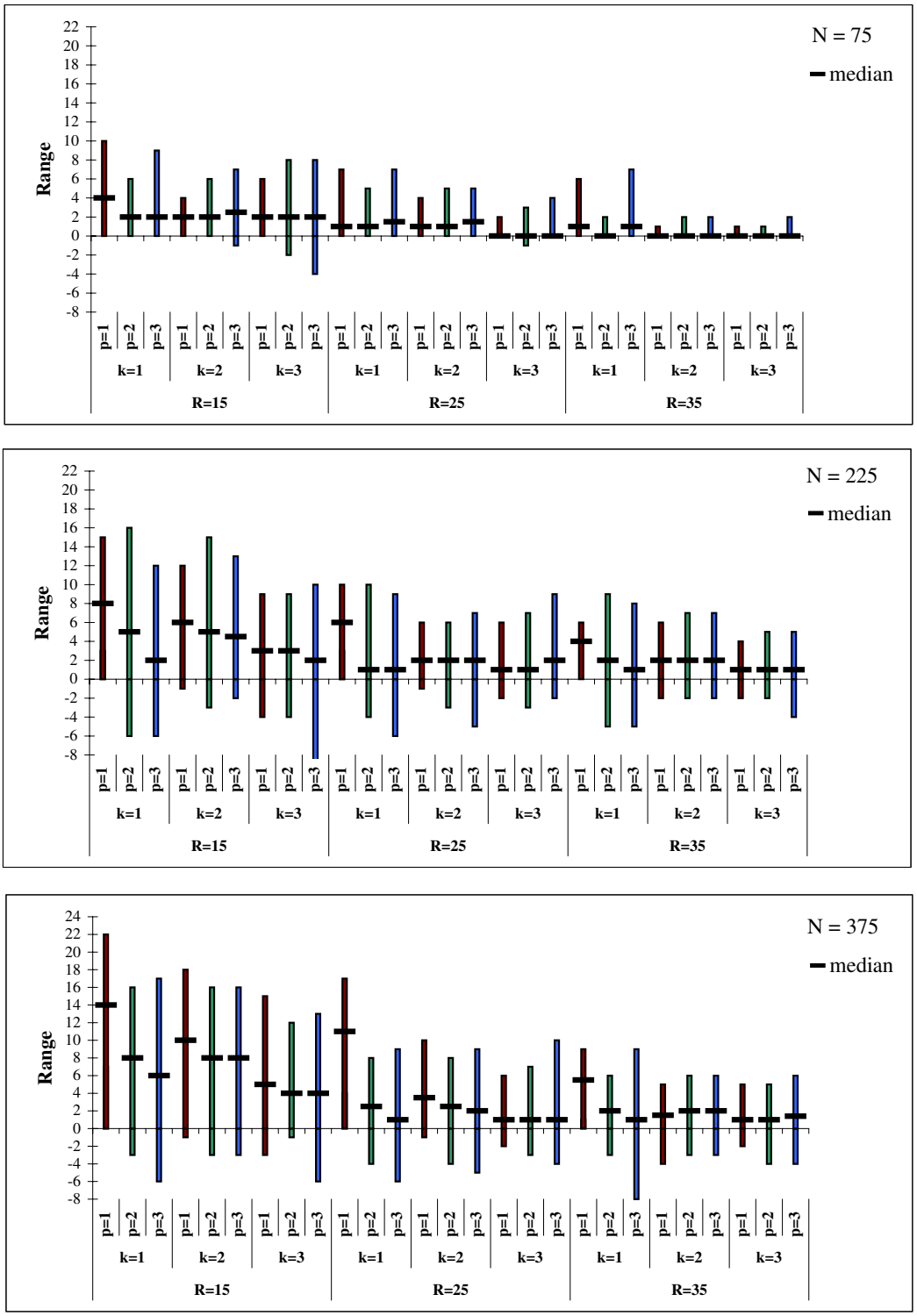


Figure 5.7: The difference $range_{Random-Greedy}$.

The $range_{Random-Greedy}$ values that do not change substantially when k are increased is explained as follows for all levels of N. Such situations are observed mainly for $k = 2$ and $k = 3$ and $p \geq 2$. For all levels of N, as R increases, the $range_{Random-Greedy}$ decreases. This is due to the graphs becoming denser. As the graphs become dense, the difference between the two algorithms decreases. This is apparent in Figure 5.7 in the values of the medians of the $range_{Random-Greedy}$.

The three charts in Figure 5.7 represent each level considered for the variable N. As N increases in Figure 5.7 the $range_{Random-Greedy}$ increases or shows negligible change for large R and k. The increase in the maximum value of $range_{Random-Greedy}$ from $N = 75$ to $N = 225$ is apparent in Figure 5.7. Note that the minimum value of $range_{Random-Greedy}$ for $N = 75$ is seldom negative. However, as N increases to 225, we see that the minimum value of $range_{Random-Greedy}$ is commonly negative for almost all levels of k and p. This is due to the increase in the value of k to $k = 2$ and $k = 3$. As mentioned earlier, the increase in k, plays the same role on the density as R. Thus, we see less difference between the two algorithms for large k. This change in the minimum value of the $range_{Random-Greedy}$ for $N = 225$ on average is the same as for $N = 375$. The values for the maximum value in $range_{Random-Greedy}$ tend to increase for sparse networks and stay the same for dense networks (i.e. when R and k are large).

5.4.2 Difference Between Random and Ran&Greedy

In this section we discuss the difference between the responses in the size of the distance-k total p-dominating sets obtained by the Random algorithm and Ran&Greedy algorithms. Figure 5.8 shows the $range_{Random-Ran\&Greedy}$ for the comparison between Random and Ran&Greedy. Phase one of both algorithms is the same. Thus, the range plotted in Figure 5.8 is the range of the differences between the Random and the Ran&Greedy algorithms in phase two. The $range_{Random-Ran\&Greedy}$ is plotted against three predictor variables, namely R, k and p. Each level of R contains all levels of k and each level of k contains all levels of p. Therefore, each bar in Figure 5.8 is the $range_{Random-Ran\&Greedy}$ for fixed values of k, p and R. The lines through the bars represent the medians for the $range_{Random-Ran\&Greedy}$. The three charts in Figure 5.8 represent the three levels of the network size N.

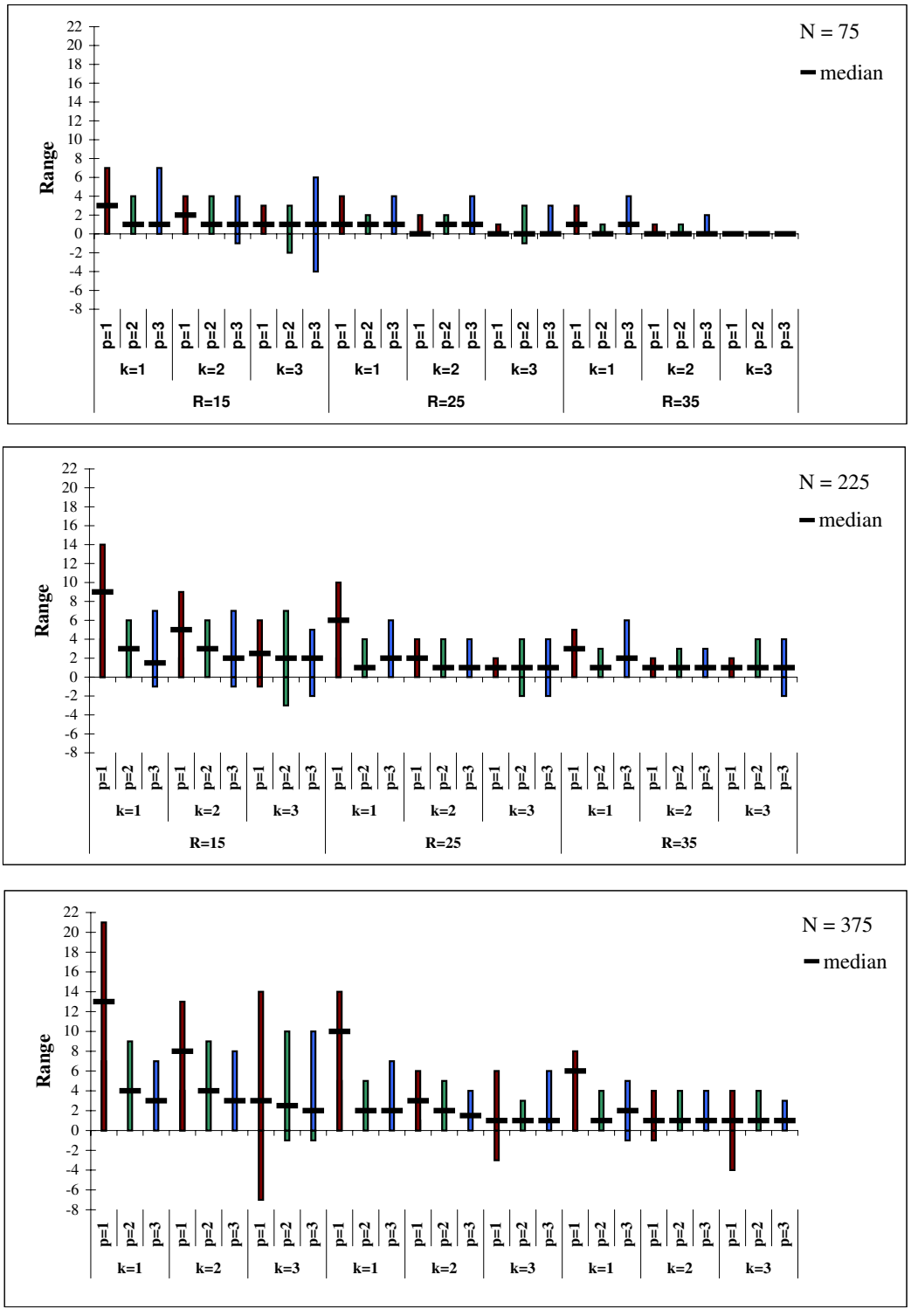


Figure 5.8: The difference $range_{Random-Ran\&Greedy}$.

The relationships between the $range_{Random-Ran\&Greedy}$ and the predictor variables are similar to those for the Random and Greedy algorithms in the previous section. Hence, the analysis is similar to that seen in Figure 5.7. There are two differences in the analysis.

In Figure 5.8 it is seldom that the minimum value of the $range_{Random-Ran\&Greedy}$ is negative for all levels of N, R, k and p. The main reason for this is that the first phase of both algorithms is the same. Thus, when we compare only phase two of the algorithms, we see positive $range_{Random-Ran\&Greedy}$. However, note that the medians in Figure 5.8 are of similar value to those in Figure 5.7.

The second difference is that the maximum value of the $range_{Random-Ran\&Greedy}$ is not as high as we observed in Figure 5.7 for the difference between the Random and Greedy algorithms. The reason for this is again because the differences between the Random and Ran&Greedy algorithms are only in phase two. Also note that when $N = 75$, for $R = 35$, $k = 3$, the median is equal to the $range_{Random-Ran\&Greedy}$ at value zero.

5.4.3 Difference Between Ran&Greedy and Greedy

The difference between the responses in the size of the distance-k total p-dominating sets obtained by the Ran&Greedy algorithm and the Greedy algorithm is shown in Figure 5.9. Similar to Figure 5.7 and Figure 5.8, the $range_{Ran\&Greedy-Greedy}$ is plotted against the three predictor variables, namely R, k and p. Each level of R contains all levels of k and each level of k contains all levels of p. Each bar in Figure 5.9 is the $range_{Ran\&Greedy-Greedy}$ for fixed values of k, p and R. The three charts in Figure 5.9 represent the three levels of the network size N.

The trends of the $range_{Ran\&Greedy-Greedy}$ and the predictor variables are similar to those between the Random and Greedy algorithms discussed previously. The $range_{Ran\&Greedy-Greedy}$ is not as large as the $range_{Random-Greedy}$ we saw in Figure 5.7. The maximum value for the $range_{Ran\&Greedy-Greedy}$ is lower than that seen in Figure 5.7 and in Figure 5.8

Recall that in Figure 5.8, the minimum values of the $range_{Random-Ran\&Greedy}$ are seldom negative. Figure 5.7 shows several intervals of the $range_{Random-Greedy}$ with negative minimum values for large N, R, and k. However, note that in Figure 5.9 all intervals have a negative minimum value except for $N = 75$, $R = 35$, $k = 3$ and $N = 225$, $k = 1$, $p = 2$. The absolute values of all min values are almost as high as the maximum values of the intervals. This indicates that the Greedy algorithm performs slightly better than the Ran&Greedy algorithm. However, their difference is negligible.

As we observe Figure 5.9 closely, we can deduce the same conclusions as above from the medians. For $N = 75$, $R = 25$ and $R = 35$, the medians are almost all at the value zero. As N increase to 225 and 335, the values of medians for $R = 25$ and $R = 35$ are mostly at the value of 1. Thus, for all levels of N and $R = 25$, $R = 35$ the difference between the two algorithms is negligible in dense graphs.

For $N = 75$ and $R = 15$, we observe the medians to be almost all at the value 1. With R fixed at 15 and N increasing the values of the medians increase to almost 3 for $N = 225$ and to almost 4 for $N = 375$. Thus, based on the medians of the intervals we can conclude that the Greedy algorithm has better performance than the Ran&Greedy algorithm for sparser graphs.

5.5 Concluding Remarks

From the experimental results we can conclude that as the density of the graphs increases (i.e. as R and k increase) the differences among the three presented algorithms becomes smaller. For sparser graphs we observe a larger difference. From Figure 5.7 and Figure 5.8 we observe that the Greedy and the Ran&Greedy algorithms outperform the Random algorithm. From Figure 5.9 we concluded that the Greedy algorithm has a slightly better performance than the Ran&Greedy algorithm. Thus, we can conclude that the best performance is returned by the Greedy algorithm, followed by the Ran&Greedy algorithm and lastly by the Random algorithm.

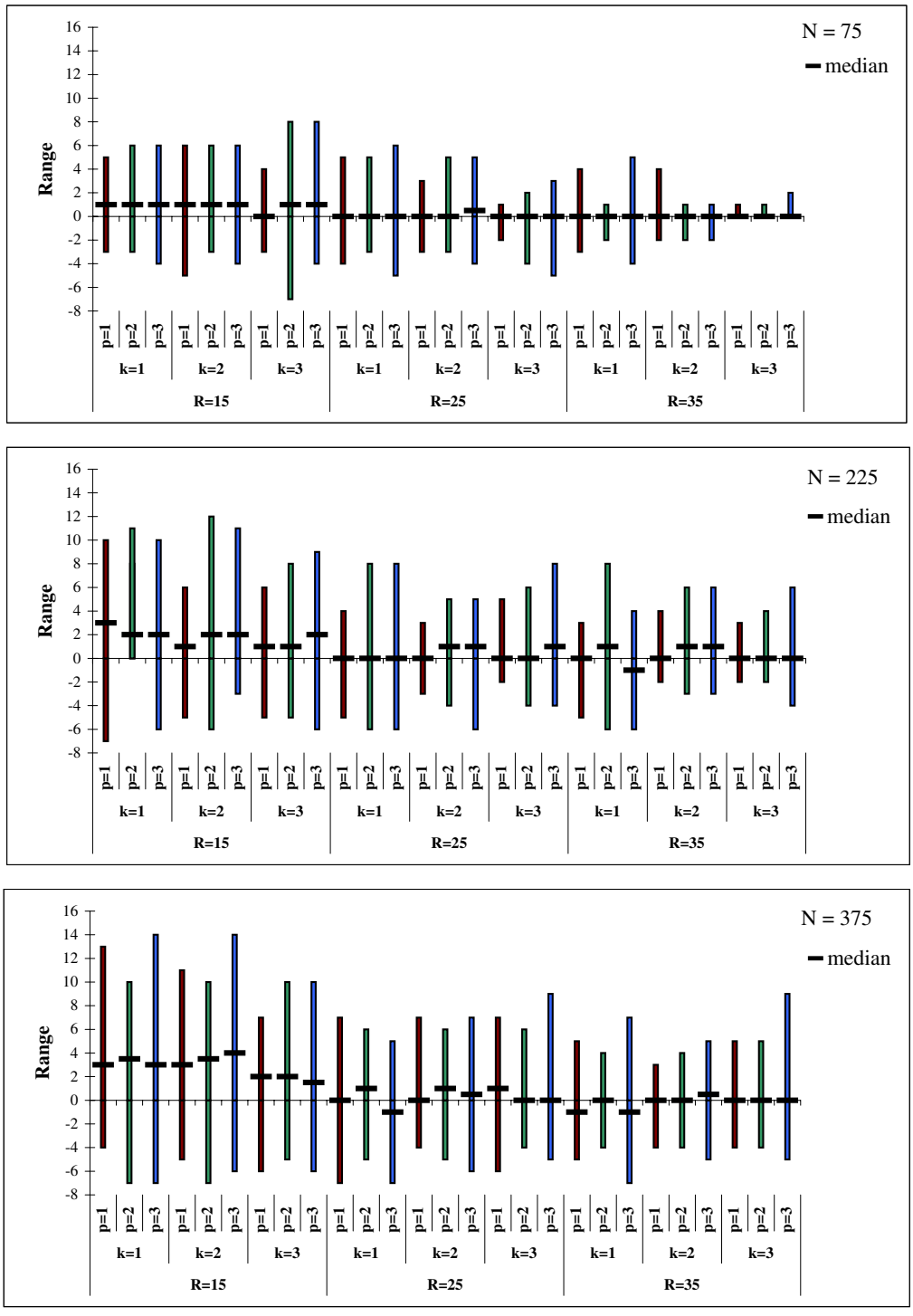


Figure 5.9: The difference $range_{Ran\&Greedy-Greedy}$.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we studied the problem of positioning multiple sinks, or data collection stops, in wireless sensor networks. Formally, we modelled this as finding a distance-k total p-dominating set in unit disk graphs where the maximum transmission range of all nodes is the same. We proposed several approximation algorithms and heuristics to find a distance-k total p-dominating set for a given graph G.

The first algorithm introduced is a centralized approximation algorithm, called Random, with a performance ratio of $2(2k+1)^2$. The algorithm consists of two phases. In phase one, the algorithm randomly finds a distance-k p-tuple dominating set S. In phase two, the algorithm adds extra vertices to S to obtain a distance-k total p-dominating set. We use the notion of maximal distance-k independent sets to obtain the approximation ratio of $2(2k+1)^2$. The second algorithm presents a distributed solution, which yields the same performance ratio. The third approximation algorithm is based on a greedy heuristic and returns an approximation ratio of $p \cdot \Delta_k$. The fourth approximation algorithm, called Ran&Greedy, is a modification of Random and yields a $2(2k+1)^2$-approximation ratio.

In addition to the four approximation algorithms, we gave two heuristics to find a distance-k total p-dominating set. The first heuristic, called Greedy, is a modification of the Ran&Greedy algorithm. The modification is in phase one, where a distance-k p-tuple dominating set is found in a greedy manner. Phase two is the same as that of the Ran&Greedy algorithm. The second heuristic presented is called Greedy2 and operates similarly to Greedy.

We ran several experiments to determine the performance of the Random, the Ran&Greedy

and the Greedy algorithms. We measured the performance of all three algorithms in terms of the size of the distance-k total p-dominating set returned by each algorithm. We used multiple regression analysis to determine the effects of each predictor variable on the size of the distance-k total p-dominating set. We further examined the important predictors and compared the performances of our algorithms. The results showed that for dense graphs the performances of all three algorithms are similar with the Greedy and the Ran&Greedy algorithms doing slightly better than the Random algorithm. For sparse graphs, the Greedy algorithm returns the best performance, the Ran&Greedy is the next best and the Random algorithm has the worst performance.

6.2 Future Work

In our experimental results, we only examined the performance of three of our algorithms. As future work we would like to determine how well our other algorithms perform in unit disk graphs.

In this thesis we assumed a static wireless sensor network. That is, the graph G and the set $D_{k,p}$ do not change over the lifetime of the network. As future work, we are interested in extending the solution of finding a distance-k total p-dominating set in static wireless sensor networks to dynamic wireless sensor networks. One approach is to find multiple disjoint $D_{k,p}$ and require that all sensors in the network be active at all times. Another approach is to allow sensors to be either active or non-active in order to prolong the network lifetime. Again we would like to find multiple disjoint $D_{k,p}$ sets, however, in this case the sensors will become active according to some schedule as well as some activation probability.

Other problems of interest as future work are to find a connected $D_{k,p}$ for a given graph G and to extend the algorithms given in this thesis as well as the experimental analysis on general graphs.

Bibliography

[1] K. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):408–421, 2003.

[2] C. Berge. *Theory of Graphs and its Applications*. Methuen, London, 1962.

[3] X. Cheng, X. Huang, D. Li, W. Wu, and D.-Z. Du. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003.

[4] V. Chvatal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[5] I. Cidon and O. Mokryn. Propagation and leader election in multihop broadcast environment. In *Distributed Computing*, volume 1499 of *Lecture Notes in Computer Science*, pages 104–118. Springer Berlin/Heidelberg, 1998.

[6] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1–3):165–177, 1990.

[7] E. Cockayne, S. Goodman, and S. Hedetniemi. A linear algorithm for the domination number of a tree. *Information Processing Letters*, 4(2):41–44, 1975.

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

[9] E. J. Cockayne, R. M. Dawes, and S. T. Hedetniemi. Total domination in graphs. *Networks*, 10:211–219, 1980.

[10] M. Couture, M. Barbeau, P. Bose, and E. Kranakis. Incremental construction of k-dominating sets in wireless sensor networks. In *Principles of Distributed Systems*, volume 4305 of *Lecture Notes in Computer Science*, pages 202–214. Springer Berlin/Heidelberg, 2006.

[11] F. Dai and J. Wu. On constructing k-connected k-dominating set in wireless networks. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, volume 1, page 81.1, 2005.

[12] J. F. Fink and M. S. Jacobson. n-domination in graphs. *Graph Theory with Applications to Algorithms and Computer Science*, pages 283–300, 1985.

[13] S. Funke, A. Kesselman, U. Meyer, and M. Segal. A simple improved distributed algorithm for minimum cds in unit disk graphs. *ACM Transaction on Sensor Networks (TOSN)*, 2(3):444–453, 2006.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[15] F. Harary and T. W. Haynes. Double domination in graphs. *Ars Combinatoria*, 55:201–213, 2000.

[16] Teresa W. Haynes, Stephen T. Hedetniemi, and Peter J. Slater. *Domination in Graphs, Advanced Topics*. Marcel Dekker, Inc., New York, 1998.

[17] Teresa W. Haynes, Stephen T. Hedetniemi, and Peter J. Slater. *Fundamental of Domination in Graphs*. Marcel Dekker, Inc., New York, 1998.

[18] M. A. Henning. Distance domination in graphs. In T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, editors, *Domination in graphs, advanced topics*, pages 321–349. Marcel Dekker, Inc., New York, 1998.

[19] R. W. Irving. On approximating the minimum independent dominating set. *Information Processing Letters*, 37(4):197–200, 1991.

[20] C. F. De Jaenisch. *Applications de l'Analyse mathematique an Jen des Echecs*. Petrograd, 1862.

[21] R. Jain. *The art of computer systems performance analysis: Technniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley and Sons, Inc., 1991.

[22] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal on Computing System Science*, 9(3):256–278, 1974.

[23] M. Jorgic, I. Stojmenovic, M. Hauspie, and D. Siplot-Ryl. Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks. In *Proceedings of the Third Annual IFIP Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 360–371, 2004.

[24] D. Joshi, S. Radhakrishnan, and N. Chandrasekharan. The k-neighbor, r-domination problems on interval graphs. *European Journal of Operational Research*, 79(2):352–368, 1994.

[25] D.G. Kleinbaum, L.L. Kupper, and K.E. Muller. *Applied Regression Analysis and Other Multivariate Methods*. PWS-KENT Publishing Company, Boston, 2 edition, 1988.

[26] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Local approximation schemes for ad hoc and sensor networks. In *Proceedings of the 3rd ACM SIGMOBILE International Workshop on Foundations of Mobile Computing (DialM-POMC 2005)*, pages 97–103, 2005.

[27] V. R. Kulli. On n-total domination number in graphs. *Graph Theory, Combinatorics, Algorithms, and Applications*, pages 319–324, 1991.

[28] D. Li, L. Liu, and H. Yang. Minimum connected r-hop k-dominating set in wireless sensor networks. *Discrete Mathematics, Algorithms and Applications*, 1(1):45–57, 2009.

[29] X.-Y. Li, W.-Z. Song, and Y. Wang. Localized topology control for heterogeneous wireless sensor networks. *ACM Transaction on Sensor Networks (TOSN)*, 2(1):129–153, 2006.

[30] M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, and D.J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995.

[31] R. Myers. *Classical and Modern Regression with Applications*. Wadsworth Publishing Company, Belmont, California, 2 edition, 1990.

[32] T. N. Nguyen and D. T. Huynh. Connected d-hop dominating sets in mobile ad hoc networks. *4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 1(4):1–8, 2006.

[33] T.N. Nguyen, D.T. Huynh, and J.A. Bolla. Minimum power minimum d-hop dominating sets in wireless sensor networks. volume 5258 of *Lecture Notes in Computer Science*, pages 176–187. Springer Berlin/Heidelberg, 2008.

[34] T. Nieberg and J. Hurink. A PTAS for the minimum dominating set problem in unit disk graphs. In *Approximation and Online Algorithms*, volume 3879 of *Lecture Notes in Computer Science*, pages 296–306. Springer Berlin/Heidelberg, 2006.

[35] T. Nieberg, J. Hurink, and W. Kern. A robust PTAS for maximum weight independent sets in unit disk graphs. In *Graph-Theoretic Concepts in Computer Science*, volume 3353 of *Lecture Notes in Computer Science*, pages 214–221. Springer Berlin/Heidelberg, 2005.

[36] O. Ore. *Theory of Graphs*, volume 38 of *American Mathematical Society Colloquium Publications*. (American Mathematical Society, Providence, RI), 1962.

[37] A. K. Parekh. Analysis of a greedy heuristic for finding small dominating sets in graphs. *Information Processing Letters*, 39(5):237–240, 1991.

[38] E. Sampathkumar and H. B. Walikar. The connected domination number of a graph. *J. Math. Phys. Sci.*, 13(6):607–613, 1979.

[39] L.A. Sanchis. Experimental analysis of heuristic algorithms for the dominating set problem. *Algorithmica*, 33(1):3–18, 2002.

[40] W. Shang, P. Wan, F. Yao, and X. Hu. Algorithms for minimum m-connected k-tuple dominating set problem. *Theoretical Computer Science*, 381(1–3):241–247, 2007.

[41] W. Shang, F. Yao, P. Wan, and X. Hu. Algorithms for minimum m-connected k-dominating set problem. In *Combinatorial Optimization and Applications*, volume 4616 of *Lecture Notes in Computer Science*, pages 182–190. Springer Berlin/Heidelberg, 2007.

[42] M.A. Spohn and J.J. Garcia-Luna-Aceves. Bounded distance multi-clusterhead formation in wireless ad hoc networks. *Ad Hoc Networks*, 5(4):504–530, 2007.

[43] M.T. Thai and D.-Z. Du. Connected dominating sets in disk graphs with bidirectional links. *IEEE Communication Letters*, 10(3):138–140, 2006.

[44] P.-J. Wan, K.M. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc network. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1597–1604, 2002.

[45] D. Wang, Q. Zhang, and J. Liu. The self-protection problem in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 3(4), 2007. Article No. 20.

[46] Y. Wang, X.-Y. Li, and Q. Zhang. Efficient algorithms for p-self-protection problem in static wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(10):1426–1438, 2008.

[47] X. Yan, Y. Sun, and Y. Wang. A heuristic algorithm for minimum connected dominating set with maximal weight in ad hoc networks. In *Grid and Cooperative Computing*, volume 3033 of *Lecture Notes in Computer Science*, pages 719–722. Springer Berlin/Heidelberg, 2004.