

2019

Optimization Approaches for Open-Locating Dominating Sets

Daniel Blair Sweigart

College of William and Mary - Arts & Sciences, blairsweigart@gmail.com

Follow this and additional works at: <https://scholarworks.wm.edu/etd>



Part of the [Applied Mathematics Commons](#)

Recommended Citation

Sweigart, Daniel Blair, "Optimization Approaches for Open-Locating Dominating Sets" (2019).
Dissertations, Theses, and Masters Projects. Paper 1582642584.
<https://doi.org/10.21220/rvmt-3s44>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

Optimization Approaches for Open-Locating Dominating Sets

Daniel Blair Sweigart

Fairfax, VA

Master of Science, College of William & Mary, 2014
Bachelor of Science, United States Coast Guard Academy, 2004

A Dissertation presented to the Graduate Faculty
of The College of William & Mary in Candidacy for the Degree of
Doctor of Philosophy

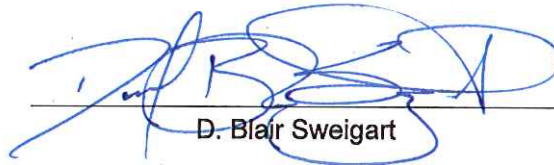
Department of Applied Science

College of William & Mary
August, 2019

APPROVAL PAGE

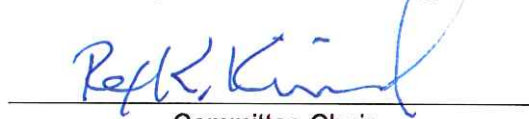
This Dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy



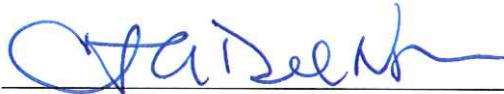
D. Blair Sweigart

Reviewed by the Committee, August 2019



Committee Chair

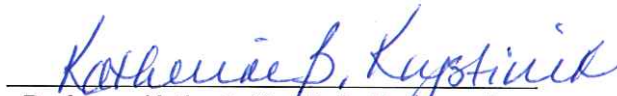
Professor Rex K. Kincaid, Mathematics
College of William & Mary



Professor Christopher A. Del Negro, Applied Science
College of William & Mary



Assistant Professor Daniel S.M. Runfola, Applied Science
College of William & Mary



Professor Kathy B. Krystinik, Mathematics
United States Coast Guard Academy

ABSTRACT

An Open Locating-Dominating Set (OLD set) is a subset of vertices in a graph such that every vertex in the graph has a neighbor in the OLD set and every vertex has a unique set of neighbors in the OLD set. This can also represent where “sensors,” capable of detecting an event occurrence at an adjacent vertex, could be placed such that one could always identify the location of an event by the specific vertices that indicated an event occurred in their neighborhood. By the open neighborhood construct, which differentiates OLD sets from identifying codes, a vertex is not able to report if it is the location of the event. This construct provides a robustness over identifying codes and opens new applications such as disease carrier and dark actor identification in networks. This work explores various aspects of OLD sets, beginning with an Integer Linear Program for quickly identifying the optimal OLD set on a graph. As many graphs do not admit OLD sets, or there may be times when the total size of the set is limited by an external factor, a concept called maximum covering OLD sets is developed and explored. The coverage radius of the sensors is then expanded in a presentation of Mixed-Weight OLD sets where sensors can cover more than just adjacent vertices. Finally, an application is presented to optimally monitor criminal and terrorist networks using OLD sets and related concepts to identify the optimal set of surveillance targets.

TABLE OF CONTENTS

| | |
|---|------|
| Acknowledgments | v |
| Dedications | vi |
| List of Tables | vii |
| List of Figures | viii |
| Chapter 1. Introduction | 1 |
| 1.1 Background Concepts and Definitions | 2 |
| 1.2 Location and Domination in Graphs | 5 |
| 1.2.1 Domination | 5 |
| 1.2.2 Location | 6 |
| 1.3 Location-Domination Concepts | 7 |
| 1.3.1 Identifying Codes | 8 |
| 1.3.2 Locating-Dominating Sets | 9 |
| 1.3.3 Strongly Identifying Codes | 10 |
| 1.4 Open Locating-Dominating Sets | 10 |
| 1.4.1 Relationship between Location-Domination Concepts | 12 |
| 1.4.2 OLD set Literature Review | 13 |
| 1.4.3 Thesis Outline | 15 |

| | | |
|------------|--|----|
| Chapter 2. | OLD set Formulation and Complexity | 17 |
| 2.1 | OLD set Complexity: Coverage Radius 1 | 18 |
| 2.2 | OLD set ILP Formulation | 24 |
| 2.2.1 | Pre-constructed ILP Formulation | 24 |
| 2.2.2 | Dynamic ILP Formulation | 26 |
| 2.3 | OLD set Results and Examples | 28 |
| 2.3.1 | OLD sets on Randomly Generated Graphs | 28 |
| 2.3.2 | Comparison and Observations | 31 |
| 2.3.3 | Computational Size Limitations of the ILP | 31 |
| 2.3.4 | Admissibility and Inadmissibility | 35 |
| Chapter 3. | Maximum Covering OLD set | 39 |
| 3.1 | IP Formulation | 40 |
| 3.2 | Results and Examples | 44 |
| 3.3 | Linearization | 48 |
| Chapter 4. | Mixed Weight OLD sets: Coverage Radius > 1 | 51 |
| 4.1 | Homogeneous OLD sets | 52 |
| 4.1.1 | R-Complete Graphs | 53 |
| 4.2 | Heterogeneous OLD sets | 56 |
| 4.2.1 | MWOLD set ILP Formulation | 56 |
| 4.2.2 | MWOLD Correctness | 59 |
| 4.2.3 | MWOLD set Examples | 59 |
| 4.3 | Complexity | 63 |
| 4.3.1 | Run Time Analysis | 63 |
| 4.4 | Results and Examples | 65 |

| | |
|---|-----|
| Chapter 5. Terrorist Network Analysis | 67 |
| 5.1 Terrorist Networks | 70 |
| 5.1.1 Paris Attacks | 71 |
| 5.1.2 9-11 Attacks | 73 |
| 5.2 Surveillance: Identifying Codes | 73 |
| 5.2.1 Identifying Code: Paris Network | 75 |
| 5.2.2 Identifying Code: 9-11 Network | 76 |
| 5.3 Hiding Activity: OLD sets | 78 |
| 5.3.1 OLD and MWOLD sets: Paris Network | 80 |
| 5.3.2 OLD and MWOLD sets: 9-11 Attack Network | 81 |
| 5.4 Robustness: Strongly Identifying Codes | 84 |
| 5.4.1 Strongly Identifying Code: Paris Network | 86 |
| 5.4.2 Strongly Identifying Code: 9-11 Network | 86 |
| 5.5 Confidential Informants: Locating-Dominating Sets | 88 |
| 5.5.1 Locating Dominating IP: Correctness | 89 |
| 5.5.2 Locating-Dominating Set: Paris Network | 90 |
| 5.5.3 Locating Dominating Set: 9-11 Network | 91 |
| 5.6 Combined Decision Support Tool for Terrorist Network Monitoring | 91 |
| 5.6.1 Combined IP: Correctness | 93 |
| 5.6.2 Combined Monitoring Plan: Paris Network | 94 |
| 5.6.3 Combined Monitoring Plan: 9-11 Network | 95 |
| 5.7 Conclusions | 97 |
| Appendix A. AMPL Code: Models and Example Data Files | 99 |
| Appendix B. Pseudo Code for Matrix Generation | 117 |

ACKNOWLEDGMENTS

I would like to thank the numerous people who helped me in this effort: First, my wife, Ashley, for, well, everything. Encouragement, support, occasional prodding, patience, and love. I could not have even dreamed of this without her support.

My mother, who encouraged me to think and challenge the world around me; and my father, who encouraged pursuit of my dreams.

I owe a special thanks to Dr. Jackie Earle for her friendship and guidance throughout the last five years.

Thanks to my many Coast Guard colleagues and friends who encouraged this effort including Bill Csisar, Jim Herlong, Tim Williams, Dr. Yukari Hughes, Julia Harder and Beth Ledbetter; John O'Connor, Ed.D., has been an outstanding mentor and role model both militarily and academically; and Greg Hersh, who kept me sane enough these last two years to cross the finish line.

Thank you to my committee members: Dan Runfolo, who provided outstanding insights and questions that will spur future research; Kathy Krystinik, who helped me find a passion in operations research nearly twenty years ago; and Christopher Del Negro, whose guidance helped convince me this path was even feasible.

A deep and heartfelt thanks to the wonderful Applied Science team, Lianne Ashburne and Lydia Whitaker, for their unparalleled patience, direction, friendly guidance, and wonderful chats.

And finally, to my advisor, Rex Kincaid, who first introduced me to this topic many years ago and encouraged my exploration thereof. His gracious support and guidance have meant the world to me, and I cannot thank him enough.

This dissertation is dedicated to three dear friends who passed on before I fully understood the gifts they had imparted:

To Phil Ross, an early computer systems integrator who first introduced me to computers and a love of learning, before I could spell “computers” (or “learning”)

To Ann Blocksom, my first CompSci teacher, who helped me see the vast possibilities of the field, and maybe a bit of potential in myself

And to my Uncle, Dick Bowie, whose love I still carry with me, whose genius numerical mind still inspires me, and whose slide rule still sits on my desk.

LIST OF TABLES

| | | |
|-----|--|----|
| 1.1 | Enumeration of Results for Location-Domination Concepts (figure 1.1) | 7 |
| 1.2 | Enumeration of OLD set Results for 5 Vertex Graph (figure 1.2) | 12 |
| 2.1 | ILP Results on Randomly Generated Graphs | 32 |
| 3.1 | Enumeration of Maximum Covering OLD set on a 5 Vertex Graph | 45 |
| 3.2 | Run Times for Linearized Model vs. Quadratic Model | 50 |
| 4.1 | Coverage Matrices for 3 Vertex Path, P3 | 59 |
| 4.2 | Shortest Path Distance Matrix for the 10 Vertex Tree | 62 |
| 4.3 | Incoming Ball $B^+(v)$ for 10 Vertex Tree | 63 |
| 4.4 | Run Time Analysis for Homogeneous and Mixed Weight OLD set | 64 |
| 5.1 | Terrorist Network Monitoring - Formulation Types | 96 |
| 5.2 | Terrorist Network Monitoring - Comparison | 97 |

LIST OF FIGURES

| | | |
|-----|--|----|
| 1.1 | Comparison of Locating-Dominating Set Types | 8 |
| 1.2 | OLD set on 5 Vertex Graph | 12 |
| 2.1 | Construction of the G_{x_i} and G_{c_j} Graphs | 21 |
| 2.2 | Plot of Pre-Constructed vs. Dynamic Run Times | 32 |
| 2.3 | OLD set on 100 Vertex Geometric Graph (Max λ_2) | 33 |
| 2.4 | OLD set on 100 Vertex Erdős-Rényi Graph (Min λ_2) | 34 |
| 2.5 | Best Feasible OLD set on a 1,000 Vertex Geometric Graph | 36 |
| 2.6 | Star Graph | 37 |
| 2.7 | Hub-and-Spoke Pattern | 38 |
| 3.1 | Maximum Covering OLD set, P = 2 | 44 |
| 3.2 | Maximal Covering Set on a 3 Vertex Path, P_3 | 45 |
| 3.3 | Maximum Covering OLD set on a 15 Vertex Graph, P=6 | 46 |
| 3.4 | Maximum Covering Set on a 100 Vertex Scale-Free Graph | 47 |
| 4.1 | R-Complete 5 Vertex Graph, $r \leq 2$ | 53 |
| 4.2 | OLD set on a 10 Vertex Complete Graph | 55 |
| 4.3 | MWOLD results on a 3 Vertex Path, P_3 | 60 |
| 4.4 | MWOLD set Results on a 10 Vertex Tree | 62 |
| 4.5 | MWOLD set Results on a 100 Vertex Scale-Free Graph | 65 |
| 5.1 | ISIL Europe Network - Paris Attacks | 72 |

| | | |
|------|---|----|
| 5.2 | Al Qaida Network - 9-11 Attacks | 72 |
| 5.3 | Minimum Identifying Code for Paris Attack Network | 75 |
| 5.4 | Minimum Identifying Code for 9-11 Attack Network | 77 |
| 5.5 | OLD set for Paris Attack Network | 79 |
| 5.6 | MWOLD set for Paris Attack Network | 80 |
| 5.7 | Max-Covering OLD set for 9-11 Attack Network | 81 |
| 5.8 | MWOLD set for 9-11 Attack Network | 82 |
| 5.9 | Weighted MWOLD set for 9-11 Attack Network | 83 |
| 5.10 | Strongly Identifying Code for Paris Attack Network | 87 |
| 5.11 | Strongly Identifying Code for 9-11 Attack Network | 87 |
| 5.12 | Locating-Dominating Set for Paris Attack Network | 90 |
| 5.13 | Locating-Dominating Set for 9-11 Attack Network | 91 |
| 5.14 | Combined Monitoring Solution for Paris Attack Network | 95 |
| 5.15 | Combined Monitoring Solution for 9-11 Attack Network | 96 |

Chapter 1

Introduction

Open locating-dominating sets (OLD sets) fall into a family of problems within network location theory that deals with event detection and location identification. The fundamental question addressed is that of sensor placement. Assuming each sensor can detect events at some specified radius, the goal is to ensure every network location is covered by at least one sensor and every location is covered by a unique set of sensors. Based on this unique set of sensors, the location of the event will be immediately known. Other closely related concepts are identifying codes[22], locating-dominating sets[34], metric bases[16], and strongly identifying codes[19]. Each of these topics incorporate the fundamental concepts of location and domination, and aim to identify optimal locations in a graph for 2-state or 3-state sensors that fulfill these two requirements. Optimal is usually taken to mean placement of a minimum number of sensors that satisfy the requirements. The sensors have certain detection limitations, usually in terms of coverage radius[4]. The specific detection properties are the primary difference between the concepts within this family of problems, e.g. the difference between Identifying Codes and Open locating-dominating Sets is the ability of the sensor to “self-detect.”[33].

1.1 Background Concepts and Definitions

- Graph Notation: Using standard graph theory notation, a graph $G = (V, E)$ consists of a vertex set $V = \{v_1, v_2, \dots, v_n\}$ and an edge set $E = \{e_1, e_2, \dots, e_m\}$. For ease of notation, vertices (v_i, v_j, \dots) may also be referenced by their subscript (i, j, \dots) . For example, $\forall v_i, v_j \in V$ is listed as $\forall i, j \in V$.
- Shortest Path Distance: The shortest path distance between two vertices, v_i and v_j is the minimum number of edges between them, and is denoted

$d(v_i, v_j)$. This may also be denoted $d(i, j)$.

- Coverage Radius r : The distance at which a sensor can detect an event, where distance is in terms of shortest path distance. A coverage radius of 1 indicates a sensor can detect an event at, or “cover,” adjacent vertices. The coverage radius of a vertex v_i , at which a sensor is located, will be denoted r_{v_i} or r_i .
- Coverage: A vertex v_i is said to be covered when there is at least one sensor that can detect an event at v_i . That is, there exists at least one sensor, located at v_j such that $d(v_i, v_j) \leq r_{v_j}$.
- Homogeneous System: A homogeneous system is one in which every sensor coverage radius is equal. $r_{v_i} = r \quad \forall i \in V$.
- Heterogeneous System: A heterogeneous, or Mixed Weight, system allows different sensors to have different coverage radii. A maximum allowable coverage radius, R , will be defined where $r_{v_i} \leq R \quad \forall i \in V$.
- Ball B_{v_i} : A ball of radius r , centered on vertex v_i is the set of vertices that are at most distance r from v_i [22]. For heterogeneous systems, one must also examine incoming and outgoing balls.
- Incoming Ball $B_{v_i}^+$: The incoming ball of a vertex v_i is the set of vertices which can cover v_i , i.e. every vertex v_j where $d(v_i, v_j) \leq r_{v_j}$.
- Outgoing Ball $B_{v_i}^-$: The outgoing ball of a vertex v_i is the set of vertices which can be covered by v_i , i.e. every vertex v_k where $d(v_i, v_k) \leq r_{v_i}$.

- Code: A code is the subset of vertices at which sensors are placed. An identifying code (IC), locating-dominating set (or locating-dominating code) (LDS or LDC), and an open locating-dominating set (OLD set) are all “codes”.
- Codeword: A codeword is a vertex in a code. Imagine a small graph $V = \{1, \dots, 5\}$ with an identifying code, $C = \{3, 4, 5\}$. $\{3\}$, $\{4\}$, and $\{5\}$ are codewords.
- Neighborhood $N(v)$: The neighborhood of a vertex v is the set of vertices that can cover or be covered by v . In a homogeneous system, $N(v) = B_v = B_v^+ = B_v^-$. In a heterogeneous or Mixed Weight system, $N(v) = B_v^+ \cup B_v^-$. Neighborhoods can be either open or closed and are depicted as $N(v)$ and $N[v]$ respectively. In a closed neighborhood, a vertex is included in its own neighborhood. In an open neighborhood, it is not: $N[v] = N(v) \cup v$.
- Separation: Two vertices are said to be separated if they have distinct intersections with the code. i.e. For a given vertex pair, there is some vertex in the code that is a neighbor of one of members of the vertex pair, but not the other. Separated vertices are also called *distinguished*.
- Twin Vertices: Twin vertices are two vertices that have the same neighborhood. Because they have the same neighborhood, they can not be separated. Twins are dependent on the neighborhood construct (open or closed), and two vertices that may be twins in a closed construct might be separable in an open construct.
- Self-Report: A sensor is said to self-report if it can provide a positive response if an event occurs at the vertex where it is located. Equivalently, it

can detect at coverage radius 0 and its diagonal entry in an adjacency matrix is 1. This is a characteristic of a closed neighborhood.

- Self-Identify: A sensor is said to be self-identifying if it can unilaterally report the location of an event that occurs at the vertex where it is located. i.e. if an event occurs at its vertex, it can say “the event happened here, you can stop looking.” This is a characteristic of locating-dominating sets, and is the defining characteristic of a 3-state sensor.
- Linear Program (LP): A problem formulation for constrained optimization problems involving decision variables, a linear objective function, and linear constraints. The goal is to minimize or maximize the objective function.[40].
- Integer Program (IP): Similar to an LP, but where some or all of the variables are constrained to be integers. IPs include Integer Linear Programs, where the objective function and constraints are linear. These are usually much harder to solve than LPs[40].

1.2 Location and Domination in Graphs

1.2.1 Domination

Domination in graphs is well presented by Haynes et. al in [17]. A set $S \subset V$ of vertices in a graph $G=(V,E)$ is called a dominating set if every vertex $v \in V$ is either an element of S or is adjacent¹ to an element of S . Indeed, [17], presents five equivalent definitions:

¹[17] uses the standard coverage radius of 1, which implies adjacency. This need not be the case, and the term “adjacent to” in this section may be substituted by “covered by”. Similarly, where 1 appears in the inequalities, this could also be r_v

- for every vertex $v \in V - S$, there exists a vertex $u \in S$ that is adjacent to v ;
- for every vertex $v \in V - S$, $d(v, S) \leq 1$;
- $N[S] = V$;
- for every vertex $v \in V - S$, $|N(v) \cap S| \geq 1$;
- $V - S$ is enclaveless, where for $T \subset V$, a vertex $v \in T$ is called an enclave of T if $N[v] \subset T$.

To this, the following may be added: A set \mathcal{D} is a dominating set if

$$B_{v_i}^+ \cap \mathcal{D} \neq \emptyset \quad \forall i \in V$$

It is natural to consider a minimum dominating set problem, where one seeks the smallest cardinality of the set $S \subset V$ such that S is a dominating set of V . The decision problem for finding a domination number is formally stated: Given a graph $G = (V, E)$ and a positive integer k , does G have a dominating set of size $\leq k$? The problem was shown to be NP-Complete by David Johnson in [14], by reduction from the 3SAT² problem and is an extension of the set cover problem. The four graphs shown in figure 1.1 are all dominating sets. The graph on the top left is the minimum cardinality dominating set.

1.2.2 Location

Location problems examine the ability to pinpoint the origin of an event. Both 2-state and 3-state sensors are considered: 2-state sensors are able to provide a binary response only: They indicate yes if an event occurs within their coverage

²The 3SAT problem is discussed in detail in section 2.1

radius, and *no* if no such event is detected. 3-state sensors can provide the same two responses as 2-state sensors, but can also sense and indicate when the event occurs at their location (i.e. can self-identify)[33, 36]. 2-state sensors are used in identifying code problems, while 3-state sensors are used in locating-dominating problems. As 2-state sensors can only provide a binary response as to whether they detect an event or not, the location is achieved by having a unique subset of sensors that can detect an event at a given vertex. While 3-state sensors can self-identify, all vertices without sensors must be covered by a similar unique set of sensors to satisfy the locating property (see section 1.3.2 Locating-Dominating sets). A set \mathcal{L} is a locating set³ if: $N(v_i) \cap \mathcal{L} \neq N(v_j) \cap \mathcal{L} \quad \forall i, j \in V : i \neq j$. The sets indicated in the four graphs shown in figure 1.1 are all locating sets, but of different types.

1.3 Location-Domination Concepts

| Vertex | Locating-Dominating Set {1,3} | Identifying Code {1,4,5} | OLD set {1,2,3} | Strongly Identifying Code {2,3,4,5} |
|--------|---|------------------------------------|---------------------------|---|
| 1 | self identifying | {1,5} | {2,3} | {2,3,5} |
| 2 | {1,3} | {1} | {1,3} | {2,3} |
| 3 | self identifying | {1,4} | {1,2} | {2,3,4} |
| 4 | {3} | {4,5} | {3} | {3,4,5} |
| 5 | {1} | {1,4,5} | {1} | {4,5} |

Table 1.1: Enumeration of results for each location-domination structure in figure 1.1. Codewords shown in bold for each instance.

³A locating-dominating set operates on the same construct but with the following change: $N(v_i) \cap \mathcal{L} \neq N(v_j) \cap \mathcal{L} \quad \forall i, j \in V \setminus S : i \neq j$. Those vertices in S can self report and need not be specifically separated.

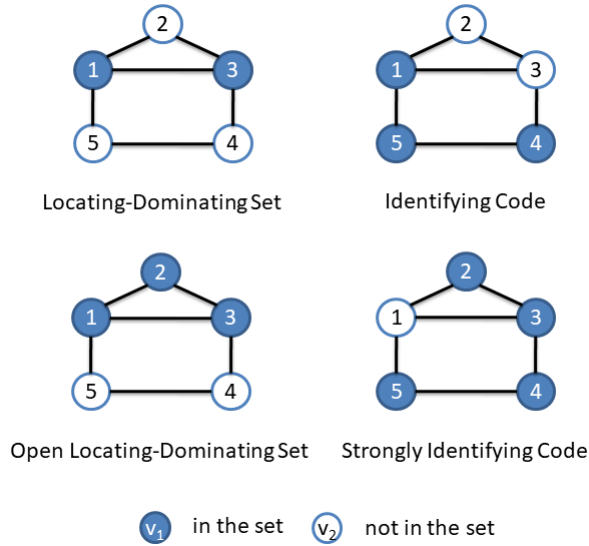


Figure 1.1: Comparison of Locating-Dominating Set Types

1.3.1 Identifying Codes

Identifying codes were introduced in 1998 by Karpovsky, Chakrabarty, and Levitin[22], with the original motivation of processor fault detection. An identifying code, \mathcal{I} is a subset of vertices in a graph such that every vertex in the graph has a neighbor in \mathcal{I} and no two vertices have the same set of neighbors in \mathcal{I} . Equivalently, each vertex is uniquely identified by its set of neighbors in \mathcal{I} . For a given code to be an identifying code, it must be both a locating set and a dominating set, as presented in section 1.1. The top right graph in figure 1.1 shows an identifying code. The domination may be verified by observing that each vertex has an adjacent vertex in the identifying code. One may verify the locating constraint by comparing each vertex's neighbors (closed) in the set, as enumerated in table 1.1. The formal definition of identifying codes is almost exactly that of OLD sets, which is presented in section 1.4, the difference being that OLD sets operate on the open neighborhood of the vertices, while identifying codes operate on the closed neigh-

borhoods. Beginning with [22], most work on identifying codes focused on lower cardinality bounds in arbitrary graphs, infinite graphs of specific structures, and specific finite graph structures such as trees and bipartite graphs[6, 12, 26].

1.3.2 Locating-Dominating Sets

Locating-dominating sets, LDS, were introduced in 1987 by Peter Slater in [34]. In a locating-dominating set construct, if an event occurs at a vertex with a sensor, that vertex can self-identify, independently reporting its own location as the event location. Otherwise, all vertices in the set can only report yes/no if an event occurs within their neighborhood.⁴ A LDS is denoted \mathcal{L} . This represents a change in the locating requirements, which is formally stated:

$$N[v_i] \cap \mathcal{L} \neq N[v_j] \cap \mathcal{L} \quad \forall i, j \in V \setminus \mathcal{L} | i \neq j$$

The top left graph in figure 1.1 shows a locating-dominating set. Again, the domination is easy to verify. The location property is verified by comparing the in-set neighbors of vertices 2, 4, and 5 ($\{1,3\}, \{3\}$, and $\{1\}$ respectively), Vertices 1 and 3 are in the set and can self-identify. Locating-dominating sets represent a slightly smarter version of identifying codes, but are otherwise fundamentally similar. Importantly, all graphs admit an LDS (trivially $\mathcal{L} = V$), while this is not the case for other locating-dominating concepts.

⁴Locating-dominating sets employ 3-state sensors.

1.3.3 Strongly Identifying Codes

A further connection between OLD sets and identifying codes is found in [19], where “strongly identifying codes” are introduced. A strongly identifying code must meet both the locating and dominating constraints on the open and closed neighborhoods simultaneously. In this way it is a fault-tolerant version of an identifying code. The bottom right graph in figure 1.1 shows a strongly identifying code. The domination is easy to verify. Verifying the locating property, however, is somewhat involved because the graph must be checked for open-neighborhood location and closed-neighborhood location. The rightmost column in table 1.1 lists the unique intersections and shows vertices that change between open and closed constructs in italics. By examination, one can see each of these are unique regardless of whether the italicized vertex is included or not. Because the strongly identifying code incorporates the OLD set in addition to the identifying code, most of the formulations presented in this paper can be easily adapted for use in strongly identifying codes.

1.4 Open Locating-Dominating Sets

OLD sets, are a fault tolerant⁵ version of identifying codes and locating-dominating sets[38]. In an OLD set, if an event occurs at a vertex that is a member of the OLD set, (i.e. a vertex with a sensor) that vertex (sensor) is unable to report anything. One may assume the sensing mechanism has been rendered inoperable by the event or a saboteur, though both conceptualizations overlook more passive potential instances, such as a disease carrier who might be asymptomatic and un-

⁵OLD sets are arguably “fault dependent” as they will not function if a sensor self reports

detectable. Other conceptual applications include a time-release contamination in a water supply system that cannot be detected until it has spread, or a dark actor in an adversary network who takes pains to hide his/her activity[38]. The OLD set operates on the open neighborhood of every vertex in the graph, where $N(v_i) = B_r(v_i) \setminus v_i$.

OLD sets were first introduced by Slater in [33], with the motivation of an intrusion detection sensor network in a group of buildings under the assumption that the intruder would render the sensor at the target building inoperable. Generally, this prevents any report from a sensor at the event location. If sensors were able to detect intrusions at adjacent buildings, then the OLD set represents the locations at which the sensors should be placed such that one can always immediately determine the location of the intrusion by the unique subset of sensors in the set that indicate an event in their neighborhood, allowing for the sabotage. As with locating-dominating sets and identifying codes, an OLD set is a subset of vertices that provides that the incoming ball of every vertex in the graph has a unique intersection with the OLD set. Fundamentally, an OLD set must meet a locating criteria and a dominating criteria, as described in section 1.2, but operating on the open neighborhoods. An OLD set is denoted \mathcal{D} . Formally:

Consider a graph G , with vertex set V , an edge set E , and the open neighborhoods $N(v), \forall v \in V$. A set $\mathcal{D} \subseteq V$ is an open locating-dominating set if:

$$\forall v \in V, N(v) \cap \mathcal{D} \neq \emptyset \quad (1.1)$$

$$\forall v_i, v_j \in V, \text{ where } v_i \neq v_j, N(v_i) \cap \mathcal{D} \neq N(v_j) \cap \mathcal{D} \quad (1.2)$$

Equation 1.1 is the dominating criteria, and equation 1.2 is the locating crite-

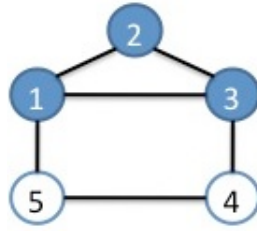


Figure 1.2: OLD set on 5 Vertex Graph

| Vertex | $v_i \cap \mathcal{D}$ |
|--------|------------------------|
| 1 | 2, 3 |
| 2 | 1, 3 |
| 3 | 1, 2 |
| 4 | 3 |
| 5 | 1 |

Table 1.2: Enumeration of OLD set Results for 5 Vertex Graph (figure 1.2)

ria. Typically, in the literature, a detection radius of 1 is used, which provides that the neighborhood consists of adjacent vertices, though other radii are considered below.

Figure 1.2 [33] depicts an OLD set for a simple graph. A cursory examination shows that the dominating constraint is satisfied. For the locating constraint, table 1.2 iterates the unique subset of sensors in each vertex’s neighborhood.

1.4.1 Relationship between Open Locating-Dominating Sets, Locating-Dominating Sets, and Identifying Codes

In investigating the proofs upon which similar work has been conducted, specifically proofs showing that identifying codes and locating-dominating sets are NP-Complete, a distinction arises which is irrelevant practically, but important philosophically. Both locating-dominating sets (or locating-dominating codes, as presented frequently in the early texts) and identifying codes rely on a dominating

constraint ($|N[v] \cap \mathcal{D}| \geq 1 \forall v \in V$) and a locating constraint, but differ in that a sensor in a locating-dominating set can self-identify as the source of the event, whereas in an identifying code it can only report that it detects an event. Considering an OLD set, where a vertex cannot self-report at all, the distinction between a locating-dominating set and an identifying code is lost. Specifically, an identifying code on the open neighborhood and a locating-dominating set on an open neighborhood are identical.

1.4.2 OLD set Literature Review

While the literature on OLD sets is not extensive, Lobstein maintains a bibliography of over 300 entries of works relating to watching systems, identifying, locating-dominating, and discriminating codes[27]. Relevant to the concepts of identifying OLD sets for applications, there are four main themes in the literature: complexity, admissibility, cardinality and density bounds, and solution approaches.

The problem of finding a dominating set is an extension of a set-cover problem [17, 14, 42]. The complexity of finding an identifying code was shown to be NP-Complete by Charon et al. in [5] using a reduction to the 3SAT problem⁶. The complexity of finding an OLD set with $r=1$ is established by Slater in [33], again using a reduction to the 3SAT problem. In [15], Givens offers a proof of NP-completeness for Mixed Weight problems, $r \geq 1$.

All graphs admit locating-dominating sets [33, 7], which can be verified by the trivial case $\mathcal{L} = V$. However, not all graphs admit identifying codes [33, 7], as seen in any graph with twin vertices or hub-and-spoke patterns. Similarly, not all graphs admit open locating-dominating sets [33, 7, 38]. This is explored later in

⁶The 3SAT problem is discussed in section 2.1

Chapter 2, and motivates Chapter 3.

In the inaugural paper on OLD sets, [33], Slater claims and proves the following relationship between the cardinality of a graph and its OLD set, \mathcal{D} , if it admits one: For a graph G with a minimum OLD set cardinality $|\mathcal{D}| = k$, $|V(G)| \leq 2^k - 1$. In terms of the OLD set, $|\mathcal{D}| \geq \log(|V(G)| + 1)$. The proof follows from the requirement that for each vertex v there must be a distinct subset $N(v) \cap S$, which amounts to requiring n distinct subsets. In other words, the objective is to seek the minimum size of $S \subset V(G)$ such that there are k distinct subsets within S , where a set of size k is known to have 2^k subsets. In [7], Chellali et al. offer insight to the upper bound by showing that for a graph with $|V| = n$ and $|\mathcal{D}| = k \geq 2$, there exists a graph of order n , such that $k + 1 \leq n \leq 2^k - 1$. They go on to show that there exists a small, specialized class of graphs of order $n \leq 6$, where $OLD(G) = n$. For general graphs, however, we have that $\log(|V(G)| + 1) \leq |\mathcal{D}| \leq |V(G)| - 1$. Chellali et al. [7] also explore the relation between the size and degree of a graph and its OLD set: For a graph G of order n and maximum degree δ , if G has an OLD set, then $|\mathcal{D}| \geq \frac{2n}{1+\delta}$. There has also been a flurry of work examining minimum OLD set density, denoted $OLD\%(G)$, on infinite graphs of various constructions [33, 24, 9]. In [33], Slater provides a general density bound: if a countably infinite graph G is regular of degree r , then $OLD\%(G) \geq \frac{2}{(1+r)}$.

Researchers have begun to examine approaches for solving identifying codes, locating-dominating sets, and open locating-dominating sets. This work falls into three categories. Many papers examine specific algorithms for special graph structures such as trees, de Bruijn graphs, circular and circulant graphs, and bounded degree graphs [33, 32, 30, 20, 29]. In [36], Suomela examines approximability for location-domination problems, providing that a minimum cardinality identifying or locating-dominating code can be approximated within a logarithmic

factor, but sublogarithmic factors are intractable. In [20], Horan, Adachi, and Bak examine the use of quantum D-wave computing to identify solutions to certain locating-dominating and identifying code problems, specifically using a reduction to a satisfiability problem of a de Bruijn graph. While the structure of an arbitrary identifying or locating-dominating set is still too complex to be easily approximated using this technique, quantum approaches will be exciting to watch in the future. In [42, 43], Xu and Xiao examine programming formulations to solve identifying code problems. These results, along with Suomela [36] touch on the concept of a pre-computed distinguishing matrix, much like the independently derived β matrix presented in Chapter 2. Xu and Xiao go on to explore solving the identifying code problem via a genetic algorithm in [43], but with approximate results for graph sizes that are solved exactly far more quickly via methods presented in Chapter 2. There has also been recent work in examining more complex instances of locating-dominating problems such as multi-stage optimization problems, as explored in [35] by Sonyç. These seem to add an artificial level of complexity to justify the multi-stage construct. Examples include examining an attacker-defender construct for identifying codes and incorporating Stackleberg concepts from game-theory.

1.4.3 Thesis Outline

To be useful in applications, methods are needed to quickly identify OLD sets in real-world, finite graphs. Off the shelf branch and bound codes for binary ILPs offer a method by which OLD sets can be identified, given any underlying graph as an input. Chapter 2 addresses traditional OLD sets and presents an integer linear program (ILP) formulation for identification of OLD sets, as well as examples

and results on graphs from the literature and random graphs of various constructions. This chapter further explores the complexity of this problem class, offering a more concise proof of NP-Completeness for identifying an OLD set. Within this chapter, it is noted that many real world graphs do not admit OLD sets because of a specific, and common, construction involving a hub-and-spoke pattern. This construct is akin to a “strong support vertex,” as covered in [32, 18]. To address this problem, Chapter 3 presents a new concept called a “Maximum Covering OLD set,” which employs a maximum set covering concept to the OLD set problem. This chapter also presents an Integer Program to identify Maximum Covering OLD sets, explores the complexity of these sets, and presents results on various graphs. Chapter 4 expands upon the traditional OLD sets discussed in literature, where $r = 1$, and explores cases when $r \geq 1$. This chapter covers formulations and results for two general cases. The first case is homogeneous, where all strengths are the same: $1 \leq r_i = \hat{r} \forall i \in V$. The second case is for heterogeneous radii, also called Mixed Weight open locating-dominating sets (MWOLD sets), where $r_i \geq 1 \forall i \in V$, but each vertex may have a sensor of any allowable strength. These sets were recently introduced by Givens in [15]. Finally, Chapter 5 explores the use of OLD sets, Identifying Codes, Strongly Identifying Codes, and Locating-Dominating sets to monitor criminal and terrorist networks to learn of pending attacks and identify those planning to carry them out. Chapter 5 presents a combined integer program that simultaneously solves for identifying codes and locating-dominating sets as a comprehensive decision support tool for government authorities to optimally monitor such networks given a variety of tools that can be used in conjunction with each other.

Chapter 2

OLD set Formulation and Complexity

While much of the OLD set literature to date explores the theoretical side of this concept, applications require methods to quickly identify an OLD set on a given graph. Integer programming offers a potential method by which to identify these sets. Such methods have been briefly, and conceptually explored by Xu and Xiao in [42, 43] and Suomela in [36], but remained focused on theoretical formulations, rather than exploring results on specific graphs. Finding identifying codes and OLD sets of minimum cardinality were shown to be NP-Hard by Charon in [5] and Slater in [33], respectively, though a more straight-forward proof for OLD set complexity is offered in section 2.1. This complexity suggests larger problem instances will experience run-time issues when using optimization solvers, which is consistent with results presented later in this chapter. For this chapter, the sensor coverage radius is assumed to be equal to one, and the neighborhood of a vertex is represented by its open adjacency matrix, \mathbf{A} , with elements $\alpha_{i,j}$, where $\alpha_{i,j} = 1$ if $d(i, j) = 1$ and is 0 otherwise. Graphs are assumed to be undirected.

2.1 OLD set Complexity: Coverage Radius 1

Finding an OLD set of a given size, k , for a coverage radius $r = 1$ was shown to be NP Complete in [33] using a reduction from the satisfiability problem (3SAT). The 3SAT problem consists of a set of variables, or literals, $X = \{x_i, i = 1, 2, \dots, n\}$, which can be TRUE or FALSE, and a set of clauses $\Theta = \{c_j, j = 1, 2, \dots, m\}$. As each variable may be negated, let $u_i \in U = \{x_1, x_2, \dots, x_n, \bar{x}_1, \bar{x}_2, \dots, \bar{x}_n\}$. Each clause in the 3SAT problem contains exactly 3 disjunctively combined variables, any of which may be negated, for example $c_1 = \{x_1 \vee \bar{x}_2 \vee x_3\}$. The clause evaluates to TRUE if any of the three conditions evaluate as TRUE. For the example clause c_1 , x_1 must be TRUE or x_2 must be FALSE or x_3 must be TRUE. The clause

will evaluate as FALSE only if x_1 is FALSE and x_2 is TRUE and x_3 is FALSE. Each clause is conjunctively combined into an expression $\Upsilon = c_1 \wedge c_2 \wedge \dots \wedge c_m$. For Υ to evaluate to TRUE, each clause must evaluate to TRUE. To satisfy this problem, there must exist some assignment of TRUE/FALSE to each variable such that Υ evaluates to TRUE. 3SAT is known to be NP Complete, as presented in Garey and Johnson's seminal text on Computational Complexity[14]. The construction presented by Slater [33] used graphs that are polynomial in the size of the 3SAT, $21N + 7M$, and needed an OLD set of size $10N + 3M$. Offered below is an alternate construction using an $8N + 3M$ construction from the 3SAT problem, needing an OLD set of size $5N + 2M$. This proof is a modification of the identifying code construction and proof offered by Charon in [5].

In the spirit of many of the papers in the identifying code and locating-dominating set literature, let $OS(v) = N(v) \cap \mathcal{D}$, i.e. the set of neighbors of vertex v in the OLD set.

Before proceeding with the proof, consider the phrase "an OLD set of size *at most* k ."

Lemma 2.1 *There exists an Open Locating-Dominating Set of size "at most k " if and only if there exists an OLD set of exactly size k .*

Proof: If there exists an OLD set of size k , there exists one of size at most k . This is equivalent to saying if there exists an OLD set of size k , there exists a set of size $k' \leq k$, which trivially holds at equality. Conversely, if there exists an OLD set of size at most k , there exists one of size k . This rests on the fact that one can always add vertices to an OLD set, and it will remain an OLD set.

Consider a feasible OLD set \mathcal{D} , on any graph, $G = (V, E)$, where $|\mathcal{D}| < |V|$,

$|V| \geq 3$.¹ Consider any two vertices, v_1, v_2 . Since \mathcal{D} is a feasible OLD set, each vertex is covered, and the two vertices are separated. A vertex v_3 is then added as a codeword to the OLD set: $\mathcal{D}' = \mathcal{D} \cup v_3$. The action falls into one of three exhaustive categories: v_3 is not in the neighborhood of either vertex; v_3 is in the neighborhoods of both vertices; or v_3 is in the neighborhood of one vertex, but not the other. In the first case, no change occurs in the domination or separation of either vertex: $N(v_i) \cap \mathcal{D} = N(v_i) \cap \mathcal{D}'$, $i = \{1, 2\}$. In the second case $v_3 \in N(v_i), i = \{1, 2\}$: both vertices meet the dominating constraint under \mathcal{D}' as they did under \mathcal{D} . Since they were separated under \mathcal{D} , they remain separated under \mathcal{D}' because there is already some other codeword $v_s \in \mathcal{D} : v_s \in OS(v_1), v_s \notin OS(v_2)$. Adding v_3 to \mathcal{D} does not alter this separation; \mathcal{D}' is feasible. In the third case, let $v_3 \in N(v_1), v_3 \notin N(v_2)$. Since \mathcal{D} is feasible, both vertices are still covered and there still must exist some codeword v_s which separates v_1 and v_2 . This separation is unchanged by adding v_3 . Since \mathcal{D} is feasible, \mathcal{D}' is also feasible in all cases, and it is clear adding codewords to a feasible OLD set will yield another feasible OLD set. Therefore, if there exists a feasible OLD set of size $k' \leq k$, there also exists a feasible OLD set of size k . ■

Let $r = 1$. The following decision problem is NP-complete:

Name: Open Locating-Dominating Set (OLD set)

Question: Is there an OLD set $\mathcal{D} \subset V$ of size at most k ?

Instance: A connected graph $G = (V, E)$ and an integer $k \leq |V|$.

Proof: First, OLD set is \in NP. Given a solution, its accuracy may be verified in polynomial time. To check that every vertex is covered, one must examine each vertex's neighborhood to verify at least one neighbor is a codeword. This is polynomial in $|V|$, and could approach $|V^2|$ in the worst case, for a complete

¹No graph with $|V| < 3$ has a feasible OLD set that satisfies $|\mathcal{D}| < |V|$.

graph. To check that no two vertex neighborhoods have the same intersection with \mathcal{D} , each vertex must be examined pairwise with all other vertices, requiring $\sum_{i=1}^{|V|} (i-1) = \frac{|V|^2 - |V|}{2}$ comparisons, each comparison involving a maximum of $\mathcal{D} \leq |V|$ individual pair checks. The overall time to verify is $\frac{1}{2}(|V|^3 - |V|^2) + |V|^2$, which is polynomial in $|V|$.

Next, the 3SAT problem is reduced to the OLD set problem, by means of a construction that is polynomial in the size of the 3SAT. For every variable $x_i \in X$ in the 3SAT, construct a graph $G_{x_i} = (V_{x_i}, E_{x_i})$ as follows:

$$V_{x_i} = \{a_i, b_i, c_i, d_i, e_i, f_i, x_i, \bar{x}_i\}$$

$$E_{x_i} = \{(a_i, b_i), (b_i, c_i), (c_i, x_i), (c_i, \bar{x}_i), (x_i, \bar{x}_i), (x_i, d_i), (\bar{x}_i, d_i), (d_i, e_i), (e_i, f_i)\}$$

$$|V_{x_i}| = 8, |E_{x_i}| = 9$$

For each clause $c_j \in \Theta$, construct a graph $G_{c_j} = (V_{c_j}, E_{c_j})$ as follows:

$$V_{c_j} = \{\theta_j, \beta_j, \gamma_j\}$$

$$E_{c_j} = \{(\theta_j, \beta_j), (\beta_j, \gamma_j)\}$$

To this, add edges from θ_j to each variable u_{j1}, u_{j2}, u_{j3} as they appear in the clause c_j .

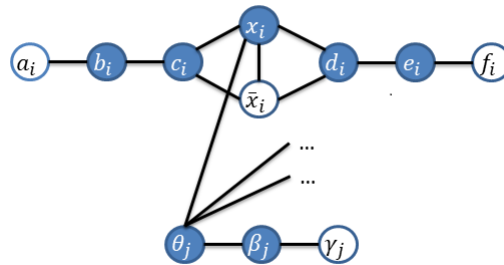


Figure 2.1: Construction of the G_{x_i} and G_{c_j} Graphs

$$G = (V, E) : V = \left(\bigcup_{i=1}^n V_{x_i} \right) \cup \left(\bigcup_{j=1}^m V_{c_j} \right) \quad (2.1)$$

$$E = \left(\bigcup_{i=1}^n E_{x_i} \right) \cup \left(\bigcup_{j=1}^m V_{c_j} \right) \cup \left(\bigcup_{\substack{j=1 \\ k=\{1,2,3\}}}^m (\theta_j, u_{jk}) \right) \quad (2.2)$$

The OLD set on G will be as follows: $b_i, c_i, d_i, e_i, \in \mathcal{D}, \forall i = 1, \dots, n$, and $\theta_j, \beta_j \in \mathcal{D}, \forall j = 1, \dots, m$. If $x_i = \text{TRUE}$, $x_i \in \mathcal{D}$, otherwise $\bar{x}_i \in \mathcal{D}$. This construction is polynomial in $n + m$, the size of 3SAT, since $|V| = 8n + 3m$, and $|E| = 9n + 5m$. $k = |\mathcal{D}| = 5n + 2m$.

Theorem 2.1 *The 3SAT problem evaluates to true if and only if there exists a feasible OLD set, \mathcal{D} , of size at most k on G .*

Proof: \mathcal{D} must be a feasible OLD set of size at most $k = 5n + 2m$, and it is a minimum cardinality OLD set on G .

Examination of $G_{x_i}, \forall x_i \in X, i = 1, \dots, n$, shows the following: to cover a_i , vertex b_i must be included in \mathcal{D} . Similarly, $e_i \in \mathcal{D}$ to cover f_i . To cover b_i , either a_i , or c_i must be in \mathcal{D} . Similarly, to cover e_i , either d_i , or f_i must be in \mathcal{D} . To separate c_i and d_i from a_i and f_i , respectively, either x_i or \bar{x}_i must be in \mathcal{D} . This will also separate x_i and \bar{x}_i . Note that including a_i or c_i is arbitrary, as is d_i or f_i , but $|V_{x_i}| \cap \mathcal{D} \geq 5$ to ensure a feasible OLD set. Choose c_i and d_i , noting that one is needed to ensure coverage for x_i and \bar{x}_i .

Examination of $G_{c_j}, \forall c_j \in \Theta, j = 1, \dots, m$ shows that $\beta_j \in \mathcal{D}$ to cover θ_j and γ_j . Either θ_j or γ_j must be in \mathcal{D} to cover β_j . Inclusion of θ_j as a codeword is necessary to separate x and \bar{x} . We will choose θ_j . Note that $|V_{c_j} \cap \mathcal{D}| \geq 2$.

This construction yields an OLD set on G of size $|\mathcal{D}| = 5n + 2m$, consisting of

$\theta_j, \beta_j, \forall j = 1, \dots, m$ and b_i, c_i, d_i, e_i and x_i or \bar{x}_i , whichever evaluates to TRUE, $\forall i = 1, \dots, n$.

Examine the claim that if there exists a feasible OLD set, \mathcal{D} of size at most $k = 5n + 2m$, as shown above, then Υ can be satisfied. The inclusion of x_i xor \bar{x}_i ensures a valid truth assignment for the 3SAT problem, as each variable or its negation will evaluate to TRUE. If neither evaluate to TRUE, then c_i and d_i are not separated from a_i and f_i respectively. If both evaluate to TRUE, then $|V_{x_i} \cap \mathcal{D}| = 6$ and $|\mathcal{D}| > 5n + 2m$. A brief examination shows that all vertices are covered, and all vertices in V_{x_i} are separated. Since $N(\gamma_j) = \{\beta_j\}$, to separate θ_j and γ_j , θ_j must be covered by a vertex other than β_j . Therefore, one of the literals connected to θ_j must be included in \mathcal{D} . Because of the inclusion condition for x_i and \bar{x}_i , this is equivalent to at least one literal evaluating to TRUE for each clause. This constitutes a satisfying solution to Υ .

Conversely, if Υ can be satisfied, then there exists a feasible OLD set, \mathcal{D} , on G of size at most k . Consider the construction of \mathcal{D} presented above, of size k . If Υ can be satisfied, at least one literal in each clause must evaluate to TRUE. Since each literal x_i or \bar{x}_i that evaluates to true is in \mathcal{D} , each θ_j vertex, $j = 1, \dots, m$ will be connected to at least one vertex in \mathcal{D} other than β_j . This ensures separation for θ_j and $\gamma_j, \forall j = 1, \dots, m$. We have previously shown that an OLD set of size k guarantees coverage of all vertices and ensures separation of all vertices except θ_j and γ_j . Thus, if Υ can be satisfied, then there exists a feasible OLD set \mathcal{D} on G of size at most $k = 5n + 2m$.

The contrapositive of this second claim also yields insight: If there does not exist a feasible OLD set \mathcal{D} on G of size at most k , then Υ cannot be satisfied. As shown, a \mathcal{D} of size k covers all vertices and ensures separation of all vertices except for the pairs θ_j and $\gamma_j, \forall j = 1, \dots, m$. If there does not exist a feasible OLD set, then for

at least one $j \in J$, θ_j and γ_j cannot be separated. (i.e. no vertex in \mathcal{D} separates $OS(\theta_j)$ and $OS(\gamma_j)$). Since θ_j is connected to the three literals in clause c_j , and any literal that evaluates to TRUE is in \mathcal{D} , $u_{j1}, u_{j2}, u_{j3} \notin \mathcal{D}$. Therefore, clause c_j will evaluate to FALSE, and Υ cannot be satisfied.

Thus, the OLD set problem is in NP, and the problem and solution can be reduced in polynomial time to/from the 3SAT problem, known to be NP-complete. Therefore, the OLD set problem is NP-complete. ■

2.2 OLD set ILP Formulation

In the spirit of the massive corpus using integer linear programming (ILP) for network location problems, including a significant contribution from Daskin’s text on Network Location Theory [10], integer programs are a promising method by which to identify open locating-dominating sets. Some programming formulations have been explored [43, 36], but are mainly conceptual in nature without specific, usable formulations. Use of integer programming to directly identify OLD sets of minimum cardinality on arbitrary graphs was first explored in 2014 [38].

2.2.1 Pre-constructed ILP Formulation

The initial ILP formulation,² referred to as the “pre-constructed formulation” takes as input an undirected graph $G = (V, E)$, represented by the graph’s adjacency matrix, \mathbf{A} , with elements $\alpha_{i,j}$. Generally, $\alpha_{i,j} = 1$ represents coverage, i.e. two vertices being within the specified coverage radius, r . For this chapter, $r = 1$ and the \mathbf{A} is equivalent to the adjacency matrix, though this will not be the case in later

²Work in this section was originally presented in [38]

chapters.

$$\alpha_{i,j} = \begin{cases} 1 & \text{if } d(i,j) \leq r \equiv (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

The formulation uses a pre-constructed vertex-pair matrix, β , in the locating constraint. The β matrix relies on a pre-processing step to compare the shared neighborhoods of each vertex pair in the graph, similar to the concepts in [43, 36]. Vertex pairs with shared neighbors are included in the set $\chi = (i,j) \quad \forall (i,j) \in V : d(i,j) \leq 2$. Note that if the shortest path distance is greater than 2, the vertices share no neighbors and need not be considered. $|\chi| \leq \frac{n^2-n}{2}$ though the worst case rarely arises in practice. The β matrix is $|\chi| \times n$, indexed by $l = (i,j) \in \chi$ and $k \in V$.

$$\beta_{l,k} = \begin{cases} 1 & \text{if } k \in N(i) \text{ or } N(j) \text{ but not both} \\ 0 & \text{otherwise} \end{cases}$$

The pre-constructed formulation is as follows:

$$\min \sum_{j \in V} x_j \tag{2.3}$$

$$\text{s.t. } \sum_{j \in V} \alpha_{i,j} x_j \geq 1 \quad \forall i \in V \tag{2.4}$$

$$\sum_j \beta_l x_j \geq 1 \quad \forall l \in \chi \tag{2.5}$$

$$\text{where } x_j = \begin{cases} 1 & \text{if } j \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

The correctness of this formulation is easily established. The objective function

seeks to minimize the number of vertices included in the set. The dominating constraint, 2.4, ensures that each vertex $i \in V$ has at least one neighbor in the OLD set. The locating constraint, 2.5, ensures that, for every vertex pair with shared neighbors, at least one vertex that separates i and j is included in the OLD set. Thus, every vertex is covered and separated from every other, and the formulation produces a minimum cardinality OLD set.

2.2.2 Dynamic ILP Formulation

The primary drawback to the formulation presented in equations 2.3-2.6 is the reliance on specific preprocessing in generating the β matrix. The step is quite cumbersome and increases the storage size of the problem. The preprocessing also limits the overall flexibility of the model. As further explored in chapter 3, extending the formulation to cover dynamic situations requires a method that does not rely on such a heavy preprocessing step. A better formulation,³ referred to as the “dynamic formulation”, eliminates the preconstructed β matrix altogether and uses an $n \times n$ matrix, Ω .

$$\Omega_{i,j} = \begin{cases} 1 & \text{if } 0 < d(i,j) \leq 2 \\ 0 & \text{otherwise} \end{cases}$$

The associated locating constraint uses the \mathbf{A} matrix, also $n \times n$, already needed for the dominating constraint. Use of the Ω matrix decreases the worst case problem storage size by a factor of n . Generation of the Ω matrix from the graph’s adjacency matrix is via a small computer program, whose pseudo code is included in appendix B. The code utilizes the Floyd-Warshall [1] algorithm to generate the

³Work in this section was originally presented in [37]

shortest path distances between each vertex pair, then assigns the appropriate value to $\Omega_{i,j}$ based on the distance between i and j . The dynamic formulation is as follows:

$$\min \sum_{j \in V} x_j \quad (2.7)$$

$$\text{s.t. } \sum_{j \in V} \alpha_{i,j} x_j \geq 1 \quad \forall i \in V \quad (2.8)$$

$$\sum_k (\alpha_{i,k} - \alpha_{j,k})^2 x_k \geq \Omega_{i,j} \quad \forall i, j \in V \quad (2.9)$$

$$\text{where } x_j = \begin{cases} 1 & \text{if } j \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases} \quad (2.10)$$

The objective function is the same as before and seeks to minimize the number of vertices included in the OLD set. This formulation again uses the \mathbf{A} matrix to satisfy the dominating constraint, 2.8: for every vertex v , at least one neighbor must be a codeword. To satisfy the locating constraint, 2.9, it uses the previously introduced Ω matrix. If $\Omega_{i,j} = 0$ (the vertices share no neighbors, or $i = j$) then no constraint is posed regarding selection of vertices in the OLD set since all x_j values can equal 0. If $\Omega_{i,j} = 1$, meaning i and j share at least one neighbor, then the constraint will function in a similar manner to 2.5: The left hand side (LHS) of 2.9 seeks a vertex that appears in the neighborhood of one vertex i or j , but not both, as in the β matrix. The constraint demands at least one distinguishing vertex be included in \mathcal{D} for each pair of vertices with shared neighbors. $(\alpha_{i,k} - \alpha_{j,k})^2$ will equal 1 where the vertex v_k is in $N(v_i)$ xor $N(v_j)$. The LHS is the sum of binary components, so the range is discrete and ≥ 0 . Thus, the formulation will produce

the minimum cardinality feasible OLD set, if one exists.

The objective function could be easily supplemented with costs, c_j , to reward or penalize inclusion of specific vertices in the OLD set. Consider $\sum_j c_j x_j$. Conceptually this could represent varied costs for placing sensors at certain locations.

2.3 OLD set Results and Examples

Implementation of the ILP in equations 2.7-2.10 was completed using AMPL (*A Mathematical Programming Language*). AMPL is a commercial modeling language designed for optimization problems [13]. The application supports numerous open-source and commercially available solvers. The primary solver used for this work is Gurobi. “Gurobi’s outstanding performance has been demonstrated through leadership in public benchmark tests and dramatic improvement in solve times year after year”[21]. Gurobi supports linear and convex quadratic optimization in continuous and integer variables, and uses primal and dual simplex and interior-point (barrier) for continuous problems; and advanced branch-and-bound with presolve, feasibility heuristics and cut generators for integer problems [21]. Examples of AMPL model and data files used throughout this work are included in Appendix A.

2.3.1 OLD sets on Randomly Generated Graphs

To gain a feel for the ILP’s performance on “real world” graphs, it was tested it on eight randomly generated, 100 vertex, graphs. Three basic constructs were explored: Geometric, Erdős-Reyni, and Scale Free.

Random geometric graphs place each vertex at a random location on a unit square, and connect (establish an edge between) two vertices if they are within a specified distance of each other [25]. Erdős-Reyni graphs establish edges based on a specified probability, p , i.e. each vertex pair $v_i, v_j \in V(G)$ has a probability, p , of having an edge $(v_i, v_j) \in E(G)$ [25]. Scale-free graphs, also known as power-law and preferential attachment, are assigned connections based on preferential attachment. Vertices are added one at a time, and connected to a single existing vertex with a probability proportional to the existing number of edges of that vertex[2]. These graphs follow power-law distributions, with a small number of high degree vertices, and a large number of small degree vertices. Graph generation was further controlled by two factors: eigenvalues (λ), and the Randic Index.

The eigenvalue used is the first non-zero eigenvalue, λ_2 , of the Laplacian matrix⁴ of the graph, assuming the eigenvalues are labeled $\{\lambda_1, \lambda_2, \dots\}$. Spectral theory shows the smallest eigenvalue, λ_1 , is equal to zero for graph applications [8]. λ_2 , then, offers insight into the connectedness of the graph. A graph with a small λ_2 value contains weakly connected components. A high λ_2 value represents a graph with strongly connected components, and generally a smaller graph diameter[8]. (Graph diameter is the max distance between any two vertices in the graph: $\max d(i, j) \forall (i, j) \in V$.) If one considers a process traversing the graph, the λ_2 value similarly gives insight into the graph's likelihood to synchronize. A high λ_2 value represents a graph that is likely to synchronize (many paths between components), while a low value represents a graph that is unlikely

⁴The Laplacian matrix, \mathbf{L} , for a graph is defined as $\mathbf{L}=\mathbf{D}-\mathbf{A}$, where \mathbf{D} is a diagonal matrix with the degree of each vertex on the diagonal and zeros elsewhere, and \mathbf{A} is the graph's adjacency matrix. The resulting matrix has the vertex's degree on the diagonal, and -1 for each (i, j) entry if i and j are adjacent.

to synchronize (few connections between components). Varying this parameter allowed examination of differences between highly connected graphs and graphs with less connected components.

The Randic Index, S , is defined as $S = \sum_{u,v \in E(V)} deg_u \cdot deg_v$, where deg_v denotes the degree of vertex v [25]. It represents the extent to which high degree vertices are connected to other high degree vertices (high S values) or to which high degree vertices are connected to low degree vertices (low S values).

To generate six of the graphs, a random graph generator was tweaked to maximize or minimize the eigenvalue of the graph's Laplacian matrix. For the geometric graphs, the generator was also used to generate two graphs that maximized or minimized the Randic-index.

The eight graphs studied are as follows:

1. Geometric, Maximizing λ_2
2. Geometric, Minimizing λ_2^*
3. Geometric, Maximizing S
4. Geometric, Minimizing S^*
5. Erdős-Renyi, Maximizing λ_2
6. Erdős-Renyi, Minimizing λ_2
7. Scale-Free, Maximizing λ_2^*
8. Scale-Free, Minimizing λ_2^*

(Entries marked with an asterisk had no feasible solution)

The main results are summarized in Table 2.1, comparing the pre-constructed formulation (using the the pre-constructed β matrix, with row elements, χ , to capture vertex pairs with shared neighborhoods) and the dynamic formulation (using the Ω matrix to identify vertex pairs with shared neighbors). All computations were completed using AMPL with a Gurobi 8.1 solver running on a Scientific Linux OS with AMD FX-8350 eight-core 4.0GHz processor. The total number of constraints for any program under the pre-constructed formulation was $100 + |\chi|$. Constraints under the dynamic formulation were $100 + 10,000$. The solve time and the number of simplex and branch and cut iterations required to identify the optimal set \mathcal{D} are also reported for each formulation. The instances denoted by * had no feasible solutions, for reasons discussed in section 2.3.4.

2.3.2 Comparison and Observations

For graphs on which there were feasible OLD sets, $|\mathcal{D}| \approx \frac{V}{3}$. This approximate relationship held for graphs of various generation rules, average vertex degrees, and max vertex degrees. Table 2.1 lists run-times and other solution data points for each of the graphs with feasible solutions, and offers a comparison between the pre-constructed and dynamic formulations. As illustrated in figure 2.2, the run times for the two constructs was nearly identical and there was no consistency to which construct ran faster for a particular graph. Figures 2.3 and 2.4 show the OLD sets on select Geometric and Erdős-Rényi graphs.

2.3.3 Computational Size Limitations of the ILP

To explore the upper limit of a direct approach, additional larger graphs were generated and tested. A Geometric construction with 1,000 vertices with a 0.07 con-

| | | Geo. Max E | Geo. Max S | E-R Max E | E-R Min E |
|----------------------------------|------------------------------|---------------|---------------|--------------|--------------|
| Optimal Set Size $ \mathcal{D} $ | | 38 | 26 | 31 | 35 |
| Pre-Constructed | Constraints | 1618 | 2812 | 1479 | 837 |
| | Constraints (post pre-solve) | 1081 | 92 | 289 | 797 |
| | $ \mathcal{X} $ | 1565 | 2712 | 1379 | 737 |
| | Simplex | 478 | 34 | 216,454 | 2,689 |
| | Br. & Cut | 1 | 1 | 7,930 | 271 |
| | Solve Time (s) | 0.053 | 0.37 | 53.8 | 0.843 |
| Dynamic | Constraints | 10,100 | 10,100 | 10,100 | 10,100 |
| | Constraints (post pre-solve) | 2,095 | 173 | 2,858 | 1,499 |
| | Simplex | 457 | 37 | 154,721 | 3,635 |
| | Br. & Cut | 1 | 1 | 5,743 | 354 |
| | Solve Time (s) | 0.058 | 0.37 | 45.7 | 0.871 |

Table 2.1: ILP Results on Randomly Generated Graphs

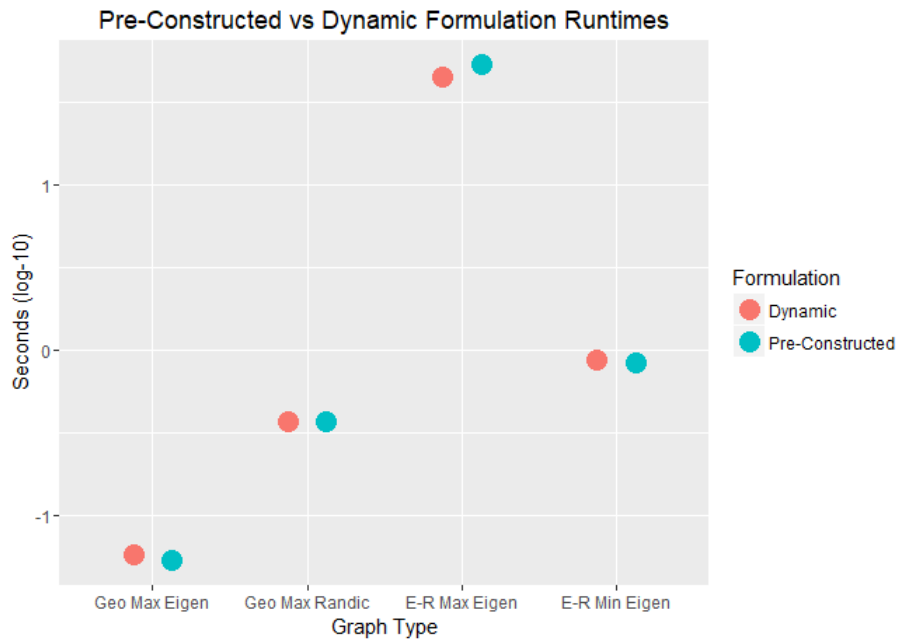


Figure 2.2: Plot of Pre-Constructed vs. Dynamic Run Times

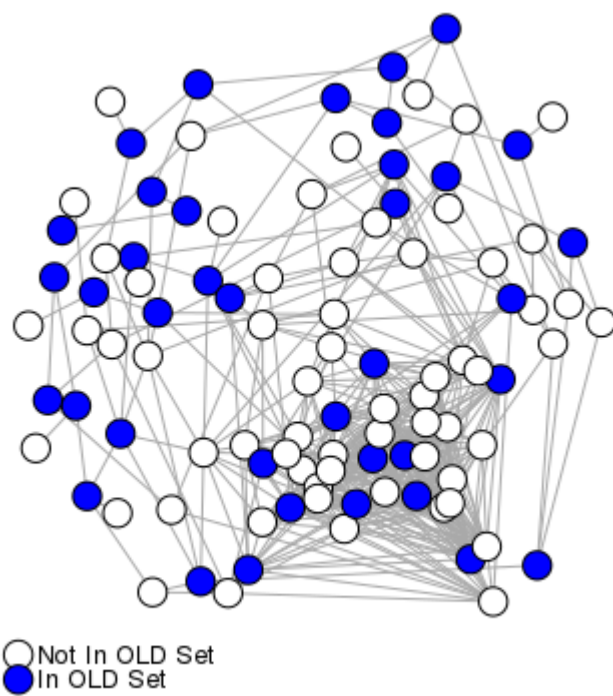


Figure 2.3: OLD set on 100 Vertex Geometric Graph (Max λ_2)

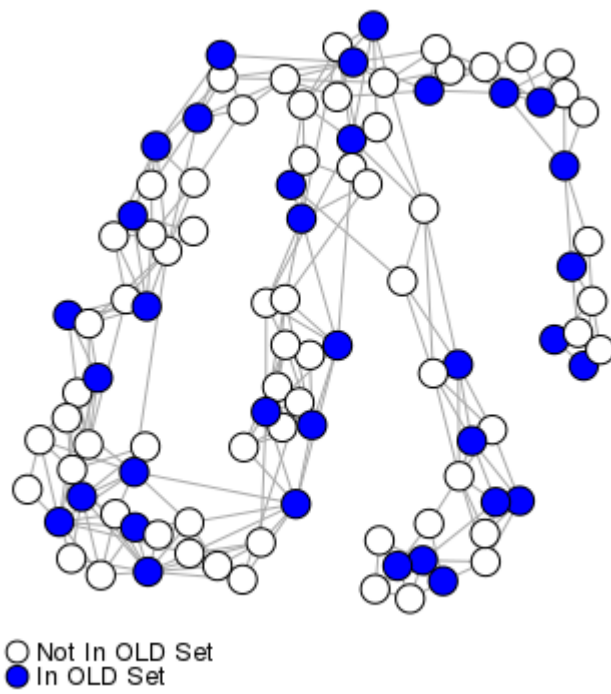


Figure 2.4: OLD set on 100 Vertex Erdős-Rényi Graph (Min λ_2)

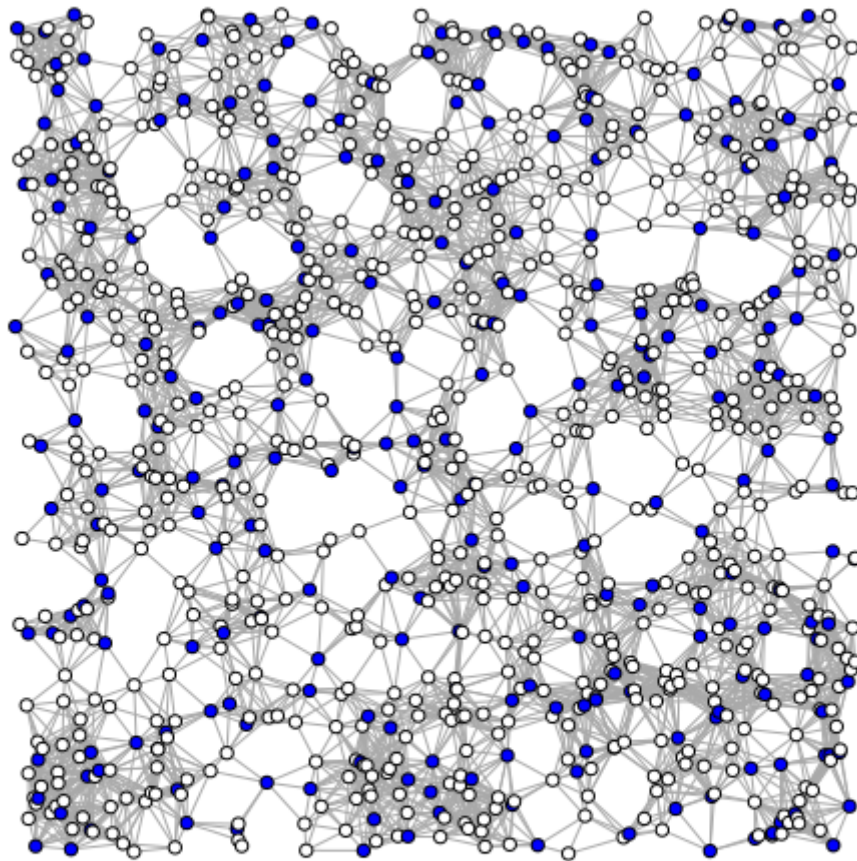
nection radius was terminated after 269,417s (almost 75 hours), having explored 6,519,613 nodes of the branch and bound. This process involved 1,490,325,779 simplex iterations, and produced a best feasible solution of 260 and a best bound of 258, gap 0.7692%. The graph and best feasible solution are shown in figure 2.5. The solver identified a solution of 260 sensors within the first 10 minutes, and closed the gap to under 2% in less than 2 hours. It took the remaining three days to try to close the remaining 2%, demonstrating a typical property of an NP-complete integer programming problem. Additional techniques, such as a relaxation technique like Lagrangian Relaxation, or a heuristic method such as Tabu Search or Simulated Annealing would be necessary to effectively generate good solutions. These solutions could be used on their own if they did not pose excess cost to the decision maker, or could be used as warm-starts for a solver to hone in on the optimal solution.

2.3.4 Admissibility and Inadmissibility

Four of these graphs contained no feasible OLD sets. The reason for this is straightforward. Observe that if $v_i, v_j \in V$ and $N(v_i) = N(v_j)$ then $N(v_i) \cap \mathcal{D} = N(v_j) \cap \mathcal{D}$. A graph with such a construction will never admit an OLD set, as it is impossible to satisfy equation 1.2, the separation constraint. Most of the vertex pairs that prohibit construction of a feasible OLD set stem from a single structure, that of a vertex with multiple leaves. This gives rise to a second observation:

For a graph, G , if $v_i \in V(G)$ has more than one neighbor of degree 1, then no OLD set exists in G .

The reason for the infeasibility is clear if one examines a star graph, shown in figure 2.6. The star graph has a central hub connected to multiple leaves, which



○ Not In OLD Set

● In OLD Set

Figure 2.5: Best Feasible OLD set on a 1,000 Vertex Geometric Graph

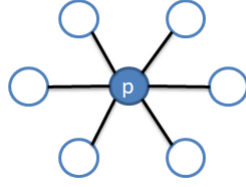


Figure 2.6: Star Graph

are vertices of degree 1. Consider a hub, vertex p , of degree $\deg(p) \geq 3$, with r leaves, $2 \leq r < \deg(p)$. These leaves are all of degree 1, with p being their only neighbor by definition. Since $N(v_i) = p \forall i \in r$, then $N(v_i) = N(v_j) \forall i, j \in r$. Consider the following lemma and theorem:

Lemma 2.2 *In a feasible OLD set, for every pair of vertices $v_1, v_2 \in V, v_1 \neq v_2, \exists$ some $v_3 \in V : v_3 \in \mathcal{D}, v_3 \in N(v_1), v_3 \notin N(v_2)$.*

Proof: Suppose there does not exist a vertex $v_3 \in \mathcal{D} : v_3 \in N(v_1), v_3 \notin N(v_2)$. Then, $N(v_1) \cap \mathcal{D} = N(v_2) \cap \mathcal{D}$. But this contradicts (1.2) in the definition of an OLD set. ■

Theorem 2.2 *If a graph has two or more vertices with the same neighborhoods ($N(v_1) = N(v_2), v_1 \neq v_2$), it has no feasible OLD sets.*

Proof: Assume there is a feasible OLD set, $\mathcal{D} \subseteq G$ and there exist some vertices $v_1, v_2 \in V : N(v_1) = N(v_2), v_1 \neq v_2$. Since \mathcal{D} is feasible, lemma 2.2 provides: $\exists v_3 : v_3 \in N(v_1), v_3 \notin N(v_2)$. But then $N(v_1) \neq N(v_2)$, which contradicts the initial assumption. ■

By Theorem 2.2, the star graph admits no feasible OLD sets. Other graphs that do not have feasible OLD sets by theorem 2.2 include P_3 , a path with three vertices, as seen in figure 3.2, and C_4 , a cycle on four vertices, arranged and

connected in a square pattern. The geometric graph which minimized the Randic-index, where vertices of high degree are connected to vertices of low degree generates a pattern with nodes of high degree connected to many leaves. This pattern violates Theorem 2.2, and does not admit an OLD set. Scale-Free, or power-law, graphs also tend to include this leaf structure and as such are poor candidates for OLD sets. The omission of these graphs from consideration limits applications for OLD sets since many graphs, such as those representing social networks, fall under this category [2]. This pattern is also called a “hub-and-spoke”: figure 2.7 depicts such a pattern, where v_1 is the hub, and v_2, v_3 and v_4 are the leaves. The prevalence of such patterns in real world graphs gives rise to the question of how to identify a set that most closely meets the requirements of an OLD set on graphs that do not fully admit feasible OLD sets.

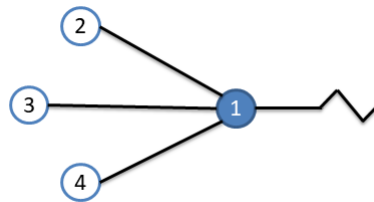


Figure 2.7: Hub-and-Spoke Pattern

Chapter 3

Maximum Covering OLD set

As discussed in previous chapters, not all graphs admit identifying codes [4, 12] or OLD sets [33, 38]. However, there may still be times when an OLD set construct is desired to inform sensor placement or other efforts. The question then arises if it is possible to identify a subgraph that does admit an OLD set. That is, can one find a set of vertices that is twin-free and where all vertices can be covered and separated? This chapter explores and answers this question, drawing heavily from standard network location theory and the concept of maximum covering sets. The resulting construct is a Maximum Covering OLD set.¹ The ILP presented in Chapter 2 is modified to identify maximum covering OLD sets, and preliminary results are explored.

3.1 IP Formulation

If the locating condition of the OLD set is relaxed, infeasibility can be avoided. Consider a construct where any vertex that is included, or “covered,” satisfies both the dominating and locating constraints, but any vertex that is excluded, “not covered,” bears no effect on the selection of codewords. This behavior could come from a preprocessing step to analyze twin-free areas of the graph, or selection of a twin-free area could be a function of the model and the optimal solution. This sort of functionality is found in maximum covering sets, discussed below, and allows for easy inclusion of trade off parameters and other modeling features, and is the desirable approach.

Let the set of vertices that are covered be called \mathcal{C} . The natural objective is to maximize $|\mathcal{C}|$, while adhering to the requirements of the OLD set for any

¹Work in this chapter was presented in 2016 at the Operations Research Conference in Hamburg, Germany, and appears in the conference proceedings [37].

covered vertex. The result is a “maximum covering” OLD set. This is similar to the problem that frequently arises in network location theory literature of a similar name. Daskin’s text on network location provides an excellent examination of the topic [10]. There are three primary cases where this construct is useful. The first is on a graph that does not admit an OLD set (for reasons previously discussed). The second is a scenario in which the number of sensors that may be placed is limited, either to some fixed, apriori number, or by resource limitations such as cost. The third case explores the tradeoff between coverage and the number of sensors required to attain the coverage; one may consider weighted parameters to help shape a tradeoff curve for a given application. Note that this is a fundamental change in the concept of domination: it is no longer necessary that every vertex have a neighbor that is a codeword.

A considerable challenge of this construction, however, is the need for the vertices to be considered dynamically within an IP. If a vertex is uncovered, it poses no separation requirement with its pairwise neighbors and such constraints may be ignored. With the pre-constructed ILP (equations 2.3-2.6) presented in chapter 2.2 that used the β matrix, such dynamic consideration is impossible. As shown below, the dynamic formulation, using the Ω matrix, can be extended to provide this behavior in a single-stage IP.

The key difference in the maximum covering OLD set is that vertices that are not covered have no bearing on selection of codewords. As such, any dominating or locating constraint that involves such a vertex should be ignored, by having the right hand side of the constraint set to zero. This must be dynamic to allow the “selection” of the covered set, \mathcal{C} , within the optimization formulation itself. This is

accomplished by the introduction of a new variable y_i where

$$y_i = \begin{cases} 1 & \text{if vertex } i \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

For flexibility, the IP below includes weighting parameter, γ , in the objective function to facilitate future tradeoff explorations. The addition of the y_i decision variable leads to a quadratically constrained IP with a linear objective function and binary decision variables.

$$\min \gamma \sum_{j \in V} c_j x_j - (1 - \gamma) \sum_{i \in V} b_i y_i \quad (3.1)$$

$$\text{s.t. } \sum_{j \in V} \alpha_{i,j} x_j \geq y_i \quad \forall i \in V \quad (3.2)$$

$$\sum_k (\alpha_{i,k} - \alpha_{j,k})^2 x_k \geq \Omega_{i,j} y_i y_j \quad \forall i, j \in V \quad (3.3)$$

$$\sum_{j \in V} x_j \leq P \quad \text{optional} \quad (3.4)$$

$$x_j \in 0, 1 \quad (3.5)$$

$$y_i \in 0, 1 \quad (3.6)$$

To ensure correctness, the objective and constraints must again be examined to ensure the desired behavior. To maximize the coverage, ($\max |\mathcal{C}|$), the objective is set to $\max \sum_i y_i$. As is frequently seen in optimization approaches, maximizing a variable is equivalent to minimizing the negative of that variable². $\max \sum_i y_i \equiv$

²Consider maximizing the variable x and minimizing $-x$, where $x \in (1, 100)$. In both cases, the optimal value is $x = 100$: For the maximization, the objective is 100. For the minimization, the objective is -100

$\min(-\sum_i y_i)$. This minimization formulation may be directly combined with the minimal $|\mathcal{D}|$ objective to introduce a multiobjective optimization. To allow greater flexibility, the weighting parameter γ is included to control the tradeoff between the cardinality objectives. Weighting/cost parameters c_j and b_i are also included to reflect potential costs of establishing a sensor at a particular location, or the importance of covering a specific vertex.

The new dominating constraint, equation (3.2), demonstrates the desired dynamic nature: $\sum_j \alpha_{i,j} x_j \geq y_i \forall i$. When $y_i = 0$, the constraint right hand side (RHS) is zero and functionally removes any constraint on the selection of codewords, x_j , stemming from that vertex v_i . When $y_i = 1$, the RHS acts as before, requiring selection of at least one codeword adjacent to y_i to satisfy the dominating constraint.

The locating constraint, equation (3.3) is more difficult and introduces a non-linear term, but follows directly from the previous section with the introduction of the Ω matrix. The locating constraint must be enforced when all of the following conditions are met:

1. vertices v_i and v_j share at least one neighbor ($\Omega_{i,j} = 1$)
2. vertex v_i is included $\in \mathcal{C}$ ($y_i = 1$)
3. vertex v_j is included $\in \mathcal{C}$ ($y_j = 1$)

If these three are met, then, by lemma 2.2, at least one facility location must distinguish $OS(v_i)$ and $OS(v_j)$. To ensure this, the RHS should equal 1, which will force the LHS to take on a value ≥ 1 , thus ensuring some vertex v_k that separates v_i and v_j is included in the OLD set. This will satisfy the locating constraint. If any of the three are NOT met, then $\text{RHS} \rightarrow 0$ and will impose no constraint on

codeword selection. The RHS $\Omega_{i,j}y_iy_j$ yields this desired behavior. The last two variables introduce the non-linear term, though the RHS is still binary.

An optional constraint could be added to govern the maximum number of facilities to be placed, $\sum_j x_j \leq P$, as is traditionally found in the network location theory “fixed-P” problem.

3.2 Results and Examples

Below are preliminary results and visualizations to further demonstrate this construct.

Maximal Covering OLD set on a 5 Vertex Graph

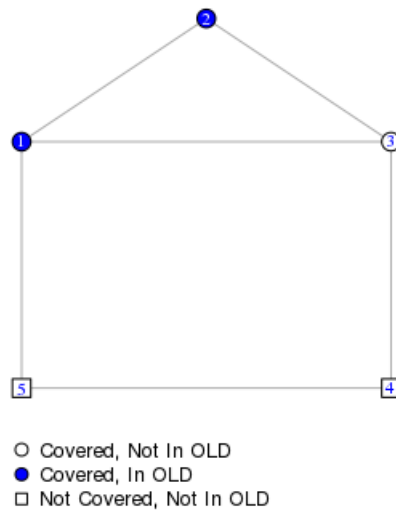


Figure 3.1: Maximum Covering OLD set, $P = 2$

The fixed-P covering set on the original 5 vertex graph demonstrates the set \mathcal{C} . By limiting the number of facilities to 2, not all 5 vertices can be covered. The

| i | $\mathcal{D} = \{1, 3\}$ | $N(v_i) \cap \mathcal{D}$ | | | |
|-----|--------------------------|---------------------------|--------------------------|--------------------------|--------------------------|
| | | $\mathcal{D} = \{1, 2\}$ | $\mathcal{D} = \{1, 4\}$ | $\mathcal{D} = \{1, 5\}$ | $\mathcal{D} = \{4, 5\}$ |
| 1 | v_3 | v_2 | - | v_5 | v_5 |
| 2 | v_1, v_3 | v_1 | v_1 | v_1 | - |
| 3 | v_1 | v_1, v_3 | v_1, v_4 | v_1^* | v_4 |
| 4 | v_3^* | - | - | v_5^* | v_5^* |
| 5 | v_1^* | v_1^* | v_1^* | v_1^* | v_4 |

Table 3.1: Enumeration of Maximum Covering OLD set on a 5 Vertex Graph

IP solution identifies 3 vertices that can be covered by an OLD set with $|\mathcal{D}| = 2$. The correctness can be verified by examining $N(v_i) \cap \mathcal{D}$. Note that due to the symmetry of the graph, several alternate solutions are not explicitly listed. For instance $\{1, 2\}$ is the same as $\{2, 3\}$, and $\{1, 4\}$ is functionally the same as $\{3, 5\}$.

Three Vertex Graph with No OLD set

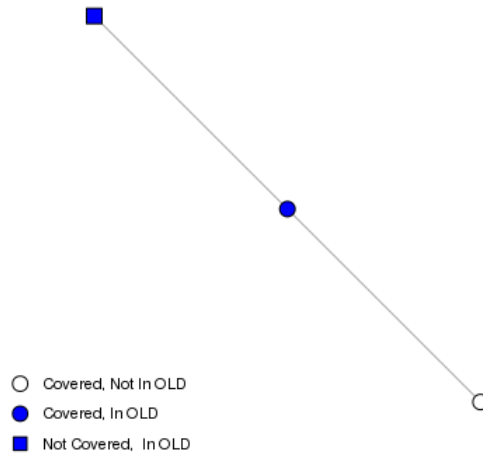


Figure 3.2: Maximal Covering Set on a 3 Vertex Path, P_3

This formulation is a small scale example of a case where a traditional OLD set would be infeasible as there would be no possible distinction between the two end vertices (they have the same neighborhood). The maximum covering OLD set permits a feasible solution by excluding one vertex from coverage. It is interesting to note that the solver chose a vertex $\notin \mathcal{C}$ as a codeword. There is an alternate optimal solution where both codewords are in $\in \mathcal{C}$. If it is desirable that only vertices that are covered be codewords, one may add a simple constraint to the IP: $x_i \leq y_i \forall i \in V$. If vertex i is not covered ($y_i = 0$), then $x_i \rightarrow 0$.

15 Vertex Graph with Fixed-P

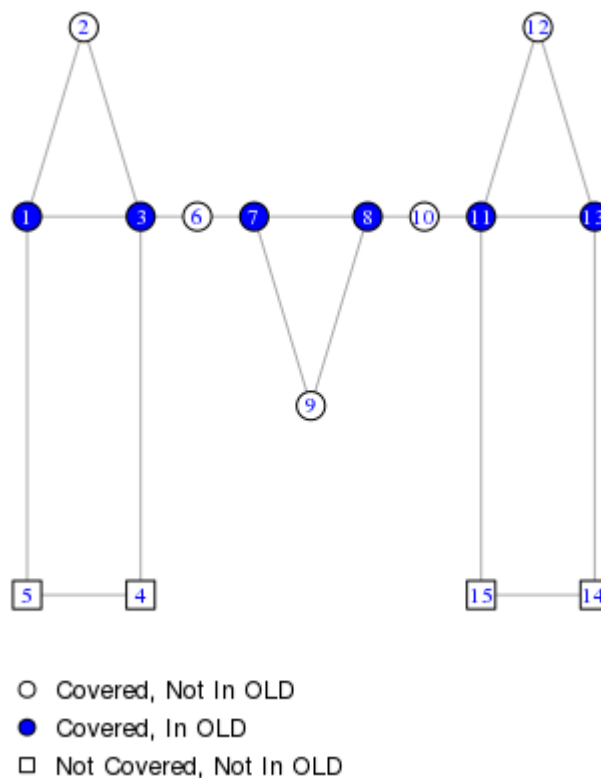


Figure 3.3: Maximum Covering OLD set on a 15 Vertex Graph, P=6

This graph is a small expansion of the initial 5 vertex graph, using a fixed- $P=6$ maximum covering OLD set construct, which yields $|\mathcal{C}| = 11$. The results highlight an interesting tradeoff between $\min |\mathcal{D}|$ and $\max |\mathcal{C}|$. Of note, there are numerous alternate optimal solutions. v_1 and v_3 , both codewords, have been included in \mathcal{C} instead of v_4 and v_5 , by using the constraint detailed in Section 3.2. The original listed optimal solution included v_4 and v_5 instead. Further, if the fixed- P constraint were relaxed to allow a slightly larger OLD set, the graph could be completely covered. $|\mathcal{D}| = 8$ provides full coverage, $|\mathcal{C}| = 15$.

Maximum Covering OLD set on a Scale-Free Graph

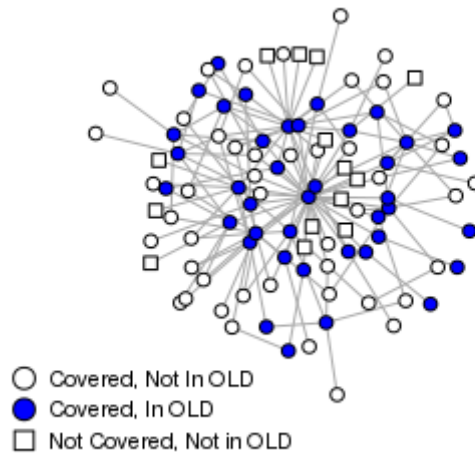


Figure 3.4: Maximum Covering Set on a 100 Vertex Scale-Free Graph

Scale-Free graphs by their nature rarely have feasible OLD sets, and were a primary motivation for development of the Maximum Covering formulation. Figure 3.4 shows the results of a Maximum Covering OLD set on a 100 vertex Scale-Free graph. Near the top center of figure 3.4 is a blue vertex (codeword) with four

neighbors above, all white, one circle and three squares. These neighbors all have a degree of one, and are of the traditional star, or hub-and-spoke, construct found in Scale-Free graphs. Since these four have identical neighborhoods, there is no way to define \mathcal{D} in a way that separates these vertices. However, the maximum covering OLD set formulation selects one of these four vertices for inclusion in \mathcal{C} and excludes the other three.

3.3 Linearization

As discussed in section 3.1, the Maximum Covering formulation introduces a non-linear constraint, specifically in the right hand side (RHS) of the locating constraint. Note the RHS is still $\in \{0, 1\}$, as it is the product of three binary terms. However, this constraint can be “linearized” by introducing a new variable $z_{i,j}$ and three new constraints, resulting in an ILP.

$$\min \gamma \sum_{j \in V} c_j x_j - (1 - \gamma) \sum_{i \in V} b_i y_i \quad (3.7)$$

$$\text{s.t.} \quad \sum_{j \in V} \alpha_{i,j} x_j \geq y_i \quad \forall i \in V \quad (3.8)$$

$$\sum_k (\alpha_{i,k} - \alpha_{j,k})^2 x_k \geq \Omega_{i,j} z_{i,j} \quad \forall i, j \in V \quad (3.9)$$

$$z_{i,j} \leq y_i \quad \forall i, j \in V \quad (3.10)$$

$$z_{i,j} \leq y_j \quad \forall i, j \in V \quad (3.11)$$

$$z_{i,j} \geq y_i + y_j \quad \forall i, j \in V \quad (3.12)$$

$$\sum_{j \in V} x_j \leq P \quad \text{optional} \quad (3.13)$$

$$x_j \in 0, 1 \quad (3.14)$$

$$y_i \in 0, 1 \quad (3.15)$$

$$z_{i,j} \in 0, 1 \quad (3.16)$$

However, as illustrated in table 3.2, run times for the linearized models are significantly slower than the original formulation. (Both models were run with AMPL and a Gurobi 8.1 solver.) Cursory results suggest the lower number of variables and constraints help the original formulation run faster. During these instances, the AMPL Gurobi the solver was able to explore more nodes and use more techniques such as branch-and-cut to hone its search for the optimal solution. Due to the power and sophistication of modern solvers such as Gurobi and others that can handle similar non-linear constraints, there is little advantage in converting formulations to linear programs or integer linear programs, at least not when still dealing with binary constraints.

| | Vertices | Linear Model | | Quadratic Model | |
|-----------------------|----------|---------------|---------|-----------------|----------------------|
| | | Run Time(sec) | Simplex | Run Time(sec) | Simplex ² |
| Geometric\$scale-Free | 10 | 0.0545 | 95 | 0.0362 | 8 |
| | 50 | 3.722 | 5,954 | 0.284 | 125 |
| | 100 | 9.231 | 18,271 | 0.509 | 178 |
| | 250 | 6,039 | 331,730 | 4.191 | 3,428 |
| | 10 | 0.0661 | 59 | 0.0434 | 16 |
| | 50 | 1.203 | 7,393 | 0.323 | 219 |
| | 100 | 12.89 | 29,070 | 1.752 | 641 |
| | 250 | 838.7 | 435,253 | 36.46 | 13,945 |

Table 3.2: Run Times for Linearized Model vs. Quadratic Model

²This is a non-LP Simplex method, commonly used in commercial solvers. These methods date back the late 1950's and 1960's and include work by Wolfe [41] and Van de Panne [39].

Chapter 4

Mixed Weight OLD sets: Coverage

Radius > 1

Many scenarios arise that are best modeled with sensors that have a stronger coverage radius than $r = 1$, e.g. computer systems may be able to detect second order anomalies, reflected by a coverage radius $r = 2$, and radio signal sensors may be placed at higher elevation or underwater, amounting to an increased coverage radius. The underlying problems fall into two categories: homogeneous systems, in which the coverage radii of every vertex is the same, and heterogeneous systems, in which a range of coverage radii may be used at any given vertex. The goal in both of these is still to find a minimum cardinality or cost set that satisfies both the locating and dominating constraints. The former is quite similar to OLD sets where $r = 1$. The latter is quite different, and is called a Mixed Weight OLD set (MWOLD set).¹

4.1 Homogeneous OLD sets

Homogeneous OLD sets with $r > 1$ are very similar in construct and behavior to OLD sets with $r = 1$. The construct of the \mathbf{A} matrix remains the same as that found in Section 2.2.1, but now r can take on values greater than 1. The “adjacency” values are relative to the coverage radius. The Ω matrix is similarly relative to r .

$$\alpha_{i,j} = \begin{cases} 1 & \text{if } 1 \leq d(i, j) \leq r \\ 0 & \text{otherwise} \end{cases}$$

$$\Omega_{i,j} = \begin{cases} 1 & \text{if } 1 \leq d(i, j) \leq 2r \\ 0 & \text{otherwise} \end{cases}$$

¹Work in this chapter was first presented in 2017 at the International Symposium on Locational Decisions (ISOLDE) XIV, in Toronto, Canada, by the author

The problem formulation is the same as found in section 2.2.1 and is solved in the same manner.

4.1.1 R-Complete Graphs

Homogeneous graphs with higher r values and Mixed Weight graphs bring the concept of complete graphs into consideration. While complete graphs exist, they are not common in large, everyday applications. When considering a coverage radius greater than 1, it is much easier to create functionally complete graphs where, due to a high coverage radius, all vertices could cover all others. Formally:

Given a graph G , let $\hat{d} = \max_{i,j \in V} d(i,j)$. If $r \geq \hat{d}$, then G is an r -complete graph.

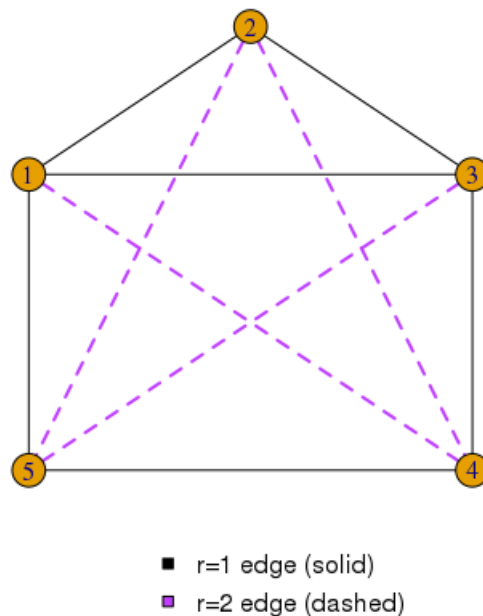


Figure 4.1: R-Complete 5 Vertex Graph, $r \leq 2$

To examine this further, recall the 5 vertex problem presented in Chapter 1. Figure 4.1 shows this construct with a coverage radius $r = 2$. The original edges

are shown in solid black lines. One may consider notional “edges” existing between vertices that are distance two away from each other. ($d(i, j) = 2$). These notional edges covered with the larger coverage radius are shown in dashed purple. For example, edges exist between vertices 5 and 1, as well as 1 and 2. Therefore, considering a coverage radius $r = 2$, there exists a notional edge between vertices 5 and 2. Under this increased coverage radius, all vertices are connected to each other, forming what is known as a complete graph. Since this required an increased coverage radius, it is presented here under the name “r-complete”.

Importantly, all complete and r-complete graphs admit feasible OLD sets.

Theorem 4.1 *If a graph ($n \geq 2$) is complete (or r-complete), then it admits an OLD set.*

Proof: By definition, the open neighborhood adjacency matrix of a complete (or r-complete) graph is as follows:

$$\begin{matrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 0 \end{matrix}$$

If a graph satisfies both the dominating and locating conditions, it will admit an OLD set. There are two necessary and sufficient conditions for a graph to admit an OLD set: each vertex must have at least one neighbor, and no two vertices may have the same neighborhood. If a vertex does not have at least one neighbor, it cannot have a neighbor in the set. If two vertices have the same set of neighbors, there is no way to distinguish the two. If a graph meets both of these requirements, it is possible to construct an OLD set. Complete graphs meet these conditions:

- In a complete graph, each vertex has $n - 1 \geq 1$ neighbors.

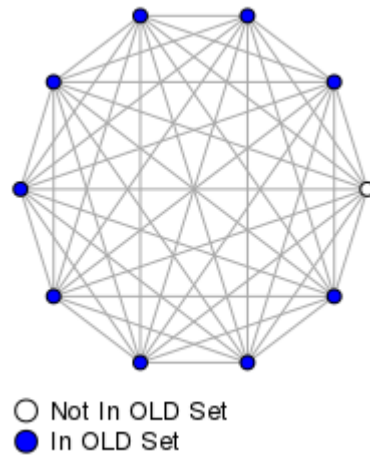


Figure 4.2: OLD set on a 10 Vertex Complete Graph

- For any vertex i in a complete graph, its neighbors are $V \setminus i$. For any two vertices i and j , where $i \neq j$, the neighborhoods are $V \setminus i$ and $V \setminus j$, which are distinct since $i \neq j$. Therefore, a complete graph satisfies both of the required conditions for a feasible OLD set to exist.

Since complete graphs will always meet these two conditions, complete (or r-complete) graphs admit OLD sets. ■

To better illustrate this, consider the set $\mathcal{D} = V$ as the OLD set, where each vertex, v_i , is covered by $\mathcal{D} \setminus v_i$. The optimal OLD set solution in a complete graph is any set of $|V| - 1$ vertices, where exactly one vertex v_j does not have a sensor. Figure 4.2 shows an OLD set on a 10 vertex complete graph. All vertices $v_i \neq v_j$ are covered by $\mathcal{D} \setminus v_i$, while v_j is covered by \mathcal{D} . Dropping any additional sensor would leave two vertices indistinguishably covered by all sensors and would be infeasible.

It should be noted complete graphs, or R-complete graphs, do not admit identifying codes. This can be seen in a complete graph's closed neighborhood adjacency matrix, where all entries are 1. Every row of the matrix is the same, and it

is impossible to distinguish any two vertices.

4.2 Heterogeneous OLD sets

Heterogeneous, or MWOLD sets, are constructs in which sensors placed at vertices may have varying coverage ranges. These were first introduced by Kincaid, Yu, and Givens in 2016 [15]. The locating and dominating constraints of an MWOLD set remain constant in concept: each vertex must have one neighbor in the set and no two vertices may have the same neighbors in the set. The varying coverage radius, however, requires a closer examination of the outgoing and incoming balls, relying more heavily on the incoming ball as every possible neighbor may have a different outgoing ball, and the ball may change during the computation if a different radius is selected for the neighboring vertex. Because of this, the incoming ball is fluid during the computations, and must be the focus of the IP.

4.2.1 MWOLD set ILP Formulation

The ILP formulation is similar to the formulations presented previously, but is indexed over both the vertices in the graph, V , and the range of potential coverage strengths, R . The maximum coverage strength considered should never exceed $n-1$ where $|V| = n$. It can be reduced, practically, to the maximum shortest path distance between any two vertices in the graph. This value is not inherently obvious from an adjacency matrix, but can be calculated during the construction of the coverage matrices. As a rule, $1 \leq r \leq R \leq \max d(i, j) \in G$. Since each vertex may have a sensor of any strength $r \leq R$, it is necessary to compute “adjacency”

matrices for each possible strength $r \in R$, for use in the domination constraints. Because they extend beyond adjacent vertices, these matrices are called “coverage matrices” and are represented by \mathbf{A}_r , for a given coverage radius $r \in R$. The construction results in a three dimensional matrix, with elements $\alpha_{r,i,j}$ of dimension $R \times n \times n$. Each of the $r, n \times n$ matrices contains a 1 in position i, j if a sensor of strength r , placed at vertex i would cover vertex j and a 0 otherwise.

$$\alpha_{r,i,j} = \begin{cases} 1 & \text{if } d(i, j) \leq r \\ 0 & \text{otherwise} \end{cases}$$

The large number of coverage matrices, each with n^2 entries, leads to the large problem sizes and subsequent memory issues described in section 4.3.1.

By contrast, a single Ω matrix is required for the MWOLD formulation. The Ω matrix can either follow an R-Complete construct (zeros on the diagonal, ones everywhere else) or can be based on a value of twice the maximum coverage radius, which can be calculated during the initial processing. The Ω matrix was introduced as a computational convenience primarily to avoid the heavy preprocessing step of identifying row pairs that have shared neighbors and tailoring the formulation and constraints to that list. In a MWOLD setting, the Ω matrix is set conservatively to include every pair of vertices that could be covered under maximum coverage strengths $r = R$. There is no harm in including extra vertex pairs in the constraints, as the very act of their domination will satisfy the locating constraint. An Ω matrix of all ones (zeros on the diagonal) will accurately solve any OLD set problem, but with slightly less efficiency. The loss of efficiency rises from the extra locating constraints whose right hand sides are non-zero, and therefore cannot be easily discounted during a presolve. A constraint with a right hand side

equal to zero can easily be discounted as the left hand side is the sum of binary terms which by definition are greater than or equal to zero. Such constraints pose no impact to the selection of vertices to be included in the OLD set.

A parameter W_r is included to represent the cost of a sensor with a given strength. This could reflect a stronger antenna, or a more involved procedure to obtain a higher coverage level. The parameter c_j represents a cost for placing a sensor at a given location. One might consider an easily accessible site in contrast to a remote location, or a sensor site that is privately vs. publicly owned. The binary variable $x_{r,j}$ equals 1 if a sensor of strength r is placed at location j , and 0 otherwise. Naturally a constraint is added to ensure no more than one sensor is placed at any given location (Equation 4.4). The locating and dominating constraints are of the typical OLD set formulation, except now summed over r as well to account for the potential sensor strengths that could be placed at any location.

The Integer Linear Programming Formulation of the heterogeneous MWOLD set is as follows, with $i, j, k \in V, r \in R$:

$$\min \sum_{r,j} (W_r + c_j)x_{r,j} \quad (4.1)$$

$$\text{s.t.} \sum_{r,j} \alpha_{r,i,j}x_{r,j} \geq 1 \quad \forall i \in V \quad (4.2)$$

$$\sum_{r,k} (\alpha_{r,k,i} - \alpha_{r,k,j})^2 x_{r,k} \geq \Omega_{i,j} \quad \forall i, j \in V \quad (4.3)$$

$$\sum_r x_{r,j} \leq 1 \quad \forall j \in V \quad (4.4)$$

$$x_{r,j} \in \{0, 1\} \quad (4.5)$$

4.2.2 MWOLD Correctness

The objective function for a MWOLD set, equation 4.1, is a natural extension of the one for homogeneous OLD set formulations, and accounts for the cost of a certain strength sensor, W_r , the cost of placing a sensor at a certain vertex, c_j , and whether a sensor of strength r is placed at vertex j , represented by the binary $x_{r,j}$. The goal is to minimize this objective, summed across r and j which is desired.

The dominating constraint, equation 4.2, ensures that the incoming ball of every vertex i is covered by at least one sensor of strength r at vertex j . That $d(i, j) \leq r$ is guaranteed by the construct of $\alpha_{r,i,j}$.

The locating constraint, equation 4.3 uses the $\Omega_{i,j}$ matrix to govern which vertex pairs need to be separated. For pairs that are too far apart or where $i = j$, $\Omega = 0$ and imposes no constraint. The left hand side of the equation ensures that the incoming balls of i and j differ by at least one neighbor.

4.2.3 MWOLD set Examples

| Vertex | $\alpha_{1,i,j}$ | | | $\alpha_{2,i,j}$ | | |
|--------|------------------|---|---|------------------|---|---|
| | 1 | 2 | 3 | 1 | 2 | 3 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 2 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | 0 | 1 | 0 | 1 | 1 | 0 |

Table 4.1: Coverage Matrices for 3 Vertex Path, P3

To further illustrate the functionality of the MWOLD set, two results are shown and explained: A three vertex path, and a 10 vertex tree.

Consider the 3 vertex path, P_3 , shown in figure 4.3. For this model, all sensor strength and location costs were equal, and the ILP sought to minimize the

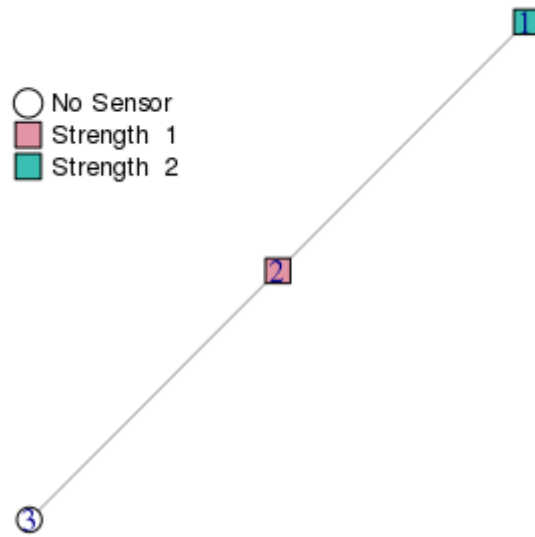


Figure 4.3: MWOLD results on a 3 Vertex Path, P_3

total numbers of sensors needed. Recall from Chapter 3 this graph does not admit a OLD set across all three vertices, but adding a range of sensor strengths overcomes this limitation. Considering domination for P_3 , it is clear at least two sensors are needed to satisfy the dominating constraint. A sensor at vertex 2 provides coverage for vertices 1 and 3, while at least one of these vertices must also have a sensor to cover vertex 2.

For P_3 , the enumeration of the constraint for vertex pair $(1, 3)$ is as follows:

$$\begin{array}{ll}
 (\alpha_{1,1,1} - \alpha_{1,1,3})^2 x_{1,1} + & \text{strength-1 sensor at vertex 1} \\
 (\alpha_{1,2,1} - \alpha_{1,2,3})^2 x_{1,2} + & \text{strength-1 sensor at vertex 2} \\
 (\alpha_{1,3,1} - \alpha_{1,3,3})^2 x_{1,3} + & \text{strength-1 sensor at vertex 3} \\
 (\alpha_{2,1,1} - \alpha_{2,1,3})^2 x_{2,1} + & \text{strength-2 sensor at vertex 1} \\
 (\alpha_{2,2,1} - \alpha_{2,2,3})^2 x_{2,2} + & \text{strength-2 sensor at vertex 2} \\
 (\alpha_{2,3,1} - \alpha_{2,3,3})^2 x_{2,3} \geq \Omega_{1,3} & \text{strength-2 sensor at vertex 3}
 \end{array}$$

Because both vertices 1 and 3 are at a distance of one from vertex 2, a sensor at vertex 2 offers no separation. This can also be seen by examining the first and third rows of $\alpha_{1,i,j}$ in table 4.1, which are the same. Therefore, a strength-2 sensor is required at either vertex 1 or vertex 3 to offer separation.

To illustrate the concept on a larger graph, figure 4.4 shows the ILP solution on a randomly generated tree with 10 vertices. The optimal solution has five sensors, three of strength 2 and two of strength 3. Table 4.2 lists the shortest path distance matrix (the header denotes if a vertex has a sensor, and its strength). It is then clear from an examination of the incoming ball of each vertex listed in table 4.3 that each vertex has at least one neighbor in the set and no two vertices are covered by the same set of neighbors. Proving that this solution is the minimum cardinality MWOLD set is far more complicated, and would require full enumeration of all possible sets with $|\mathcal{D}| = 4$, at each possible strength. Noting that the greatest distance between two vertices in this graph is 5, and therefore strengths 6-10 need not be considered, there are 3,150,000 possible solutions of size $|\mathcal{D}|=4$ which would need to be checked for feasibility to determine if a better solution existed.

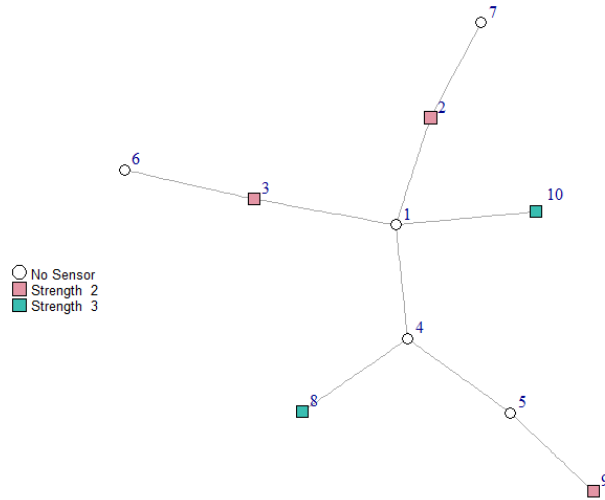


Figure 4.4: MWOLD set Results on a 10 Vertex Tree

| v_{x_r} | 1 | 2 ₂ | 3 ₂ | 4 | 5 | 6 | 7 | 8 ₃ | 9 ₂ | 10 ₃ |
|-----------|---|----------------|----------------|---|---|---|---|----------------|----------------|-----------------|
| 1 | 0 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 1 |
| 2 | 1 | 0 | 2 | 2 | 3 | 3 | 1 | 3 | 4 | 2 |
| 3 | 1 | 2 | 0 | 2 | 3 | 1 | 3 | 3 | 4 | 2 |
| 4 | 1 | 2 | 2 | 0 | 1 | 3 | 3 | 1 | 2 | 2 |
| 5 | 0 | 3 | 3 | 1 | 0 | 4 | 4 | 2 | 1 | 3 |
| 6 | 0 | 3 | 1 | 3 | 4 | 0 | 4 | 4 | 5 | 3 |
| 7 | 0 | 1 | 3 | 3 | 4 | 4 | 0 | 4 | 5 | 3 |
| 8 | 0 | 3 | 3 | 1 | 2 | 4 | 4 | 0 | 3 | 3 |
| 9 | 0 | 4 | 4 | 2 | 1 | 5 | 5 | 3 | 0 | 4 |
| 10 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 0 |

Table 4.2: Shortest Path Distance Matrix for the 10 Vertex Tree

Note, if a solution with $|\mathcal{D}| < 4$ exists, then a solution where $|\mathcal{D}| = 4$ exists, as shown in lemma 2.1, so it would be enough to show that any solution smaller than the ILP result was feasible.

| v | B^+ |
|----|---------------|
| 1 | {2, 3, 8, 10} |
| 2 | {3, 8, 10} |
| 3 | {2, 8, 10} |
| 4 | {2, 3, 9, 10} |
| 5 | {8, 9, 10} |
| 6 | {3, 10} |
| 7 | {2, 10} |
| 8 | {10} |
| 9 | {8} |
| 10 | {2, 3, 8} |

Table 4.3: Incoming Ball $B^+(v)$ for 10 Vertex Tree

4.3 Complexity

Complexity of OLD set with fixed $r > 1$

In [5], Charon et al. offer a NP-Completeness proof for identifying codes when the sensor coverage radius is greater than 1. NP-Completeness for the homogeneous OLD set construct, where $r_{v_i} = \hat{r} \quad \forall i \in V$ follows simply from the $r = 1$ construct presented in Chapter 2. The adjacency matrix is reconstructed to reflect an edge wherever the distance $d(i, j) < r$. This takes $\frac{n^2}{2}$ steps, checking for each (i, j) pair where $i < j$, and making the symmetric change for both $\alpha_{i,j}$ and $\alpha_{j,i}$. Once the new adjacency matrix and graph are constructed, the instance may be treated like an $r = 1$ graph. In [15], Givens offers a proof for the MWOLD set problem being NP-Complete.

4.3.1 Run Time Analysis

Run times for 10 random graphs are shown in table 4.4. The homogeneous problems stayed relatively compact across the various sizes tested. This is consistent with a homogeneous OLD set construct with $r > 1$ being easily reduced to an $r = 1$ instance, and follows similar run times as seen in Chapter 2. Results on graphs

| | Homogeneous | | | | |
|---------------------|---------------|--------|--------|--------|---------|
| Graph Size | 10 | 50 | 100 | 250 | 500 |
| Max Distance $d(G)$ | 5 | 11 | 13 | 17 | 18 |
| Max Coverage Radius | 10 | 10 | 10 | 10 | 10 |
| Variables | 10 | 50 | 100 | 250 | 500 |
| Constraints | 110 | 2,550 | 10,100 | 62,750 | 250,500 |
| OLD set Size D | 9 | 48 | 93 | 227 | 454 |
| Run Time (s) | 0.00426 | 0.0503 | 0.0534 | 1.41 | 17.2 |
| | Heterogeneous | | | | |
| Graph Size | 10 | 50 | 100 | 250 | 500 |
| Max Distance $d(G)$ | 5 | 11 | 13 | 17 | 18 |
| Max Coverage Radius | 9 | 11 | 13 | 17 | 18 |
| Variables | 90 | 550 | 1,300 | 4,250 | 9,000 |
| Constraints | 120 | 2,600 | 10,200 | 63,000 | 250,500 |
| OLD set Size D | 5 | 20 | 40 | 107 | - |
| Max Sensor Strength | 3 | 10 | 13 | 12 | - |
| Run Time (s) | 0.257 | 10.4 | 229.24 | 6,308 | - |

Table 4.4: Run Time Analysis for Homogeneous and Mixed Weight OLD set

with a homogeneous set of sensors, $r = 10$, are shown in table 4.4. Run times, and more importantly memory constraints, for Heterogeneous Mixed Weight OLD set problems grew substantially with problem size. Constructs with various maximum strengths were tested, including $r \leq 10$, $r \leq \max_{i,j \in V} d(i,j)$. For brevity, only the results where $r = \max_{i,j \in V} d(i,j)$ are listed in table 4.4, as these are most applicable. (Note, the table uses the notation $\max d(G)$ to mean $\max_{i,j \in V} d(i,j)$, or the maximum distance between any two vertices in G). Results are not shown for the 500 vertex Mixed Weight problem, as it was too large to load into a system with 64GB RAM, either through AMPL or through a solver running outside of AMPL.

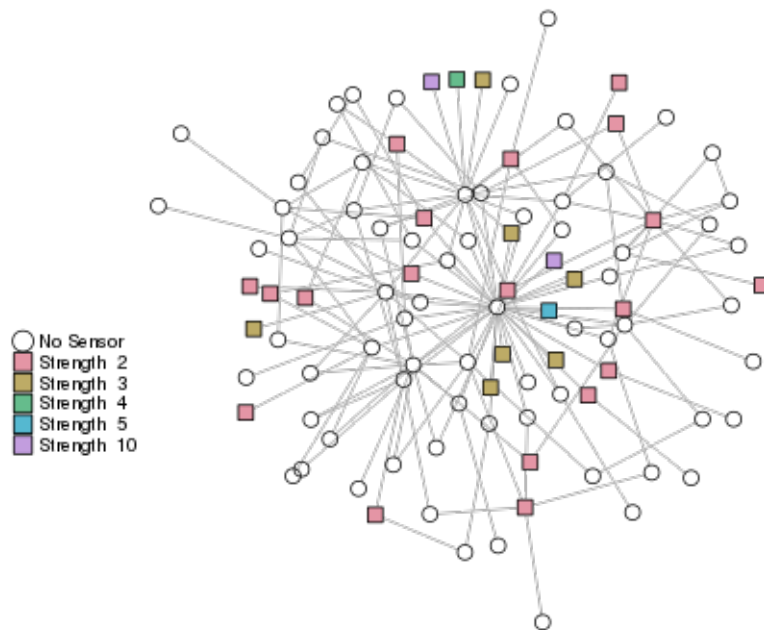


Figure 4.5: MWOLD set Results on a 100 Vertex Scale-Free Graph

4.4 Results and Examples

A MWOLD set of minimum cardinality may be identified using the formulation presented above with modern solver packages such as Gurobi. Most advanced solvers are equipped with algorithms for branch and bound techniques, branch and cut, cutting planes, and other powerful methods of attacking integer programming problems.² Figure 4.5, as an example, shows the minimum cardinality MWOLD set result on a 100 vertex random Scale-Free graph, with a maximum sensor strength of 10 and all sensor costs equal. This was solved using Gurobi 6.5 with a run time of 45 seconds. The minimum cardinality MWOLD set uses 30 sensors: 19 of coverage radius $r = 2$; seven of $r = 3$, one of $r = 4$, one of

²As noted by Professor Michael Trick of Carnegie Mellon University in a recent INFORMS podcast, solvers have advanced to the point where intricate presolving techniques and advanced approaches, external to the solver, are no longer necessary as they were five to eight years ago. It is now just as efficient for most problems to formulate the program and the solver work the solution[23].

$r = 5$, and two of $r = 10$. This formulation can be carried out on any graph for any application, computational resources permitting. This formulation also works for identifying codes, but needs significant modification to work on closed neighborhood locating-dominating sets, as discussed below in Chapter 5. For identifying codes, one simply uses an \mathbf{A} matrix with ones on the diagonal instead of zeros.

Chapter 5

Terrorist Network Analysis

Terrorism has long plagued societies, and its study, in efforts to learn about it and combat it, is a continued area of great interest. There are many groups and institutions dedicated specifically to researching terrorism, including the National Consortium for the Study of Terrorism and Responses to Terrorism (START) at the University of Maryland. One area of interest is how to most efficiently monitor terrorist networks, and key players in them, to ensure authorities learn of attacks prior to their being carried out. The use of surveillance, undercover agents, and confidential informants can be crucial to alert authorities of a planned attack. However, this is not without cost. Surveillance takes man-hours and expensive equipment, and sometimes includes court orders, wire taps, and cyber methods by trained agents. Similarly, developing confidential informants takes time and significant investment and often yields less than reliable results. Nevertheless, authorities continue to use these methods as the best way to identify and thwart attacks. Any method that could help limit or hone the expense by identifying an optimal set of targets would be a great benefit.

It is reasonable to consider that activity by someone being monitored could also indicate that an associate has become active in planning an attack. If authorities had a graph of the terrorist network, these associations would be indicated by edges between operatives. Further, one might consider that a more robust surveillance package or a more loyal, and well established, confidential informant might indicate activity several times removed. This could be through “friends-of-friends” communication, or second hand chatter that an informant might relay. This gives rise to a mixed weight notion for surveillance and informants.

If the only goal of authorities was to ensure knowledge of a pending attack without needing to know the potential perpetrators, authorities might implement a dominating set. A dominating set would ensure all members of the network are

within the coverage radius of a network member under surveillance. Based on the assumption that signs of the planning would be visible directly or indirectly through members under surveillance, a dominating set would ensure authorities would learn of pending attacks. However, if authorities were also interested in immediately identifying the potential perpetrator of an attack, then a identifying code, locating-dominating set, or an OLD set would be appropriate and provide the desired information.

The notion of using identifying codes, locating-dominating sets, and specifically OLD sets, to identify activity in criminal or terrorist networks was first proposed in [38], in 2014. Recently, Sen et al. published two works [31, 3] that explored the use of identifying codes to analyze terrorists networks. These two works describe an ILP concept and implement a graph coloring method that correctly verifies satisfaction of the locating constraint. Howeverm they do not present a method to directly identify surveillance targets.

This chapter explores the use of identifying codes, locating-dominating sets, and OLD sets on terrorist networks. It examines ideal strategies for the authorities, or an “intelligence agent” (used interchangeably), as they look to monitor a network to learn of attacks and identify the potential perpetrator of an attack. It identifies the “targets,” or members of the network that should be placed under surveillance or turned into confidential informants, depending on the construct being discussed. It presents modified formulations where necessary, demonstrates the use of weights and costs to shape the output¹, and provides results on two

¹After speaking with several colleagues in Homeland Security Investigations, The Coast Guard Investigative Service, the FBI, and the Royal Canadian Mounted Police, it became apparent that using real weights, costs, and associated information would be too sensitive for this work, and could reveal too much about counter-network practices currently in use throughout law enforcement. While not classified, the material would cross the thresholds of Sensitive But Unclassified (SBU), Law Enforcement Sensitive (LES), and For Official Use Only (FOUO). To avoid such complications, the work presented here is hypothetical in nature, and meant to illustrate how weighting

real world terrorist networks: the network behind the September 11th attacks on the U.S. and the network behind the Paris bombing in 2015.

This chapter includes five sections, each exploring a different aspect of monitoring criminal or terrorist networks, and using different assumptions. These aspects and assumptions lend themselves to different locating-dominating approaches. First, an identifying code construct is used to explore placing network members under surveillance. Next, an open locating-dominating set construct is used to account for potential efforts on the part of network members to hide their involvement in an attack. Combining these two, a strongly identifying code formulation is presented to make the model more robust and less dependent on assumed member behavior. Next, a locating-dominating set construct is developed to model the use of confidential informants, who can act as three-state sensors and explicitly tell their handlers if they've been tasked to carry out an attack. Finally, a model is presented which combines identifying codes and locating-dominating sets in a novel fashion as a decision support tool for investigators and intelligence personnel.

5.1 Terrorist Networks

This chapter explores two networks: the Al Qaida network behind the September 11, 2001 attacks, and the Islamic State of Iraq and the Levant (ISIL) network behind the attacks in Paris on November 13, 2015. Each of these attacks were catastrophic events that required significant planning and coordination. Similar to the assertions in [31, 3], it is assumed if authorities had been monitoring these networks or the actors in them, signs of the pending attacks might have been

and cost systems could work, if developed accurately for a real world criminal network.

detected, allowing authorities to learn of the pending activity and identify the perpetrators. The following sections explore various monitoring constructs pertaining to different assumptions about the behavior of the actors in these networks, and different tools available to the authorities. The two networks of individuals who carried out these attacks were extrapolated from [31] and [3] to ensure consistency with the results in that article and to provide a sound basis of comparison. In each, an actor is shown as connected to his/her known first order associates.

The major assumption of this section, the same that underlies the two papers by Basu and Sen, et al in [31, 3], is that the network is known before hand. There are cases in practice where this assumption holds: Certain organized or established criminal networks or gangs for instance, are often known in detail to authorities. This assumption does not always hold, and considerable effort is often spent trying to determine the underlying structure of a network. To best examine these locating-dominating tools, the assumption is made in this chapter that the network structures and associations were known to authorities prior to the attacks.

5.1.1 Paris Attacks

The eight coordinated attacks in Paris on November 13, 2015, were carried out by ISIL and claimed the lives of at least 19 people, injuring 140. The attacks were reportedly a retaliation for France's support of the U.S. military efforts in Iraq and Syria [11]. The component of ISIL in Europe that planned and carried out these attacks contains 10 individuals and is shown in figure 5.1.

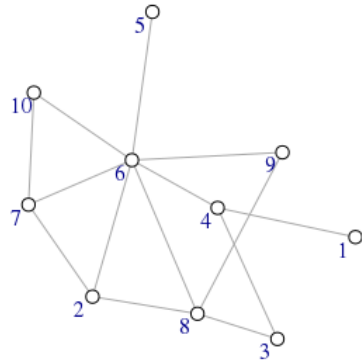


Figure 5.1: ISIL Europe Network - Paris Attacks

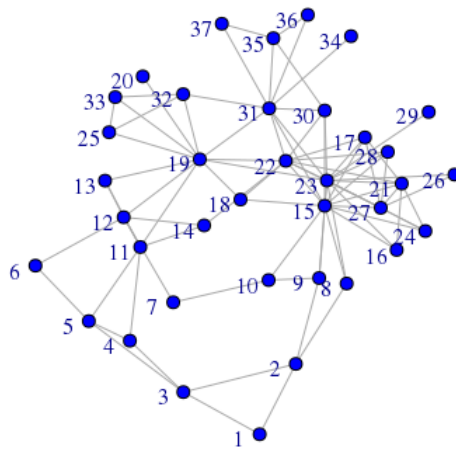


Figure 5.2: Al Qaida Network - 9-11 Attacks

5.1.2 9-11 Attacks

The four coordinated attacks on the east coast of the U.S. on September 11, 2001 were carried out by members of Al-Qaida. The attacks used hijacked aircraft flown into their targets, and claimed the lives of more than 2,767 people, injuring more than 16,000 others [11]. The network of Al Qaida operatives involved in these attacks contains 37 people, and is shown in figure 5.2.

5.2 Surveillance: Identifying Codes

For initial examination, an identifying code (IC) construct and ILP was used. Let the set of vertices in the identifying code be denoted as \mathcal{I} . An identifying code on a criminal or terrorist network can be thought of as set of surveillance targets. It is assumed that when a terrorist within the coverage range of a member under surveillance becomes active in planning an attack, the member under surveillance will show certain activities and behaviors that will tip off authorities who are watching. Each vertex that is in \mathcal{I} will be placed under surveillance so that when an attack is being planned by a member of the network, authorities will learn of the attack, and they will be able to identify the potential perpetrator immediately based on the unique set of surveillance targets that show signs of activity. Initially, a coverage radius of 1 is used, representing attack preparations by a target or their first order associates. This also indicates if the person under surveillance were the one planning to carry out the attack, the authorities would be able to note the activity in a similar manner as if the target's associates were planning the attack - i.e. the target is assumed not to be intentionally hiding their activity due to being under surveillance, but also does not give direct indication that they

themselves are the attacker. It is worth reiterating the depth of this assumption - the behavior of one planning an attack is assumed to be indistinguishable from the behavior of a member under surveillance whose first order associate is planning an attack. If this assumption does not hold, e.g. if authorities believe they could immediately identify a perpetrator if that person were under direct surveillance, then a locating-dominating set would be appropriate. Identify codes are used here to reflect general planning activity: site surveillance, gathering materials, etc. These could easily be carried out by the attack planner or associates, yielding credence to the notion that the activities of perpetrator and associates would be indistinguishable which supports the use of identifying codes. The ILP for an identifying code is the same as that for an OLD set presented in Chapter 2, except that the adjacency matrix is based on the closed neighborhood (1's on the diagonals), rather than the open.

$$\alpha_{i,j} = \begin{cases} 1 & \text{if } d(i, j) \leq r \\ 0 & \text{otherwise} \end{cases}$$

$$\Omega_{i,j} = \begin{cases} 1 & \text{if } 1 \leq d(i, j) \leq 2 \\ 0 & \text{otherwise} \end{cases}$$

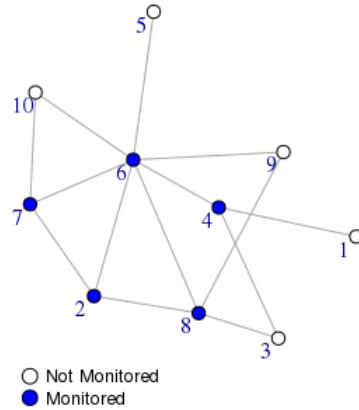


Figure 5.3: Minimum Identifying Code for Paris Attack Network

$$\min \sum_{j \in V} x_j \quad (5.1)$$

$$\text{s.t. } \sum_{j \in V} \alpha_{i,j} x_j \geq 1 \quad \forall i \in V \quad (5.2)$$

$$\sum_k (\alpha_{i,k} - \alpha_{j,k})^2 x_k \geq \Omega_{i,j} \quad \forall i, j \in V \quad (5.3)$$

$$\text{where } x_j = \begin{cases} 1 & \text{if } j \in \mathcal{I} \\ 0 & \text{otherwise} \end{cases} \quad (5.4)$$

5.2.1 Identifying Code: Paris Network

The minimum IC for the Paris network is five targets, $\mathcal{I} = \{2, 4, 6, 7, 8\}$. This is depicted in figure 5.3. This result is consistent with the result in [3], though that article does not easily identify which persons should be placed under surveillance.

5.2.2 Identifying Code: 9-11 Network

The September 11th Al Qaida network is more challenging because two vertices are twins. While not leaves (degree one vertices, examined in Chapter 2), these two vertices (25 and 33) are both connected to each other and the same two other vertices (19 and 32), though the latter two have other connections. In essence, these four vertices form a complete subgraph. As discussed in Chapter 4, complete graphs admit OLD sets and locating-dominating sets, but do not admit identifying codes. To overcome this challenge, a modified maximum covering formulation is used as presented in Chapter 3. The maximum covering then would indicate which potential perpetrators would tip off authorities and whose activity would be missed. An appropriate weighting scheme would be critical in this context. First, the γ value balances the “cost” of resource expenditure vs the risk of an event going undetected². Second, the b_i values in 5.5 should weight those individuals considered highest risk to ensure likely activity is not missed. This could result in lower weights on confirmed lower risk targets, reflected in smaller b_i values, but with the constant caution that high risk targets could hide their activity to appear benign at first observation.³ The quadratically constrained IP for identifying codes below is unchanged from Chapter 3, but as in the previous section operates on the closed neighborhood adjacency matrix rather than the open.

²While not employed in this exploratory analysis, the “fixed-P” constraint could also be beneficial if an agency had fixed resources and could only monitor a certain number of individuals.

³While by no means an actual law enforcement case, the pop culture reference of Keyser Soze from “The Usual Suspects” illustrates this point. While police question the seemingly benign, crippled con man Kint, they are being played by the actual mastermind disguising his role and leading them astray.

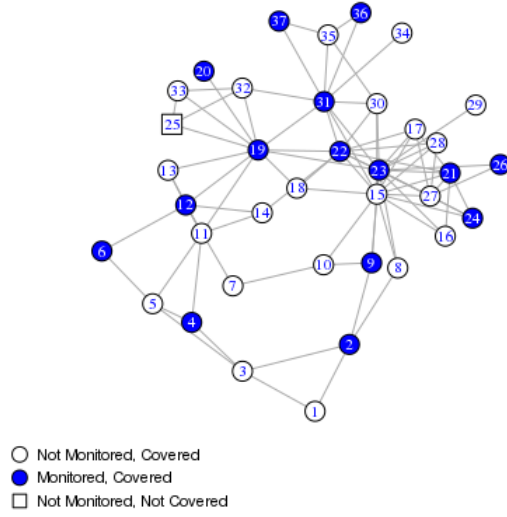


Figure 5.4: Minimum Identifying Code for 9-11 Attack Network

$$\min \gamma \sum_{j \in V} c_j x_j - (1 - \gamma) \sum_{i \in V} b_i y_i \quad (5.5)$$

$$\text{s.t.} \quad \sum_{j \in V} \alpha_{i,j} x_j \geq y_i \quad \forall i \in V \quad (5.6)$$

$$\sum_k (\alpha_{i,k} - \alpha_{j,k})^2 x_k \geq \Omega_{i,j} y_i y_j \quad \forall i, j \in V \quad (5.7)$$

$$x_i \leq y_i \quad \forall i \in V \quad (5.8)$$

$$\sum_{j \in V} x_j \leq P \quad \text{optional} \quad (5.9)$$

$$x_j \in \{0, 1\} \quad (5.10)$$

$$y_i \in \{0, 1\} \quad (5.11)$$

The minimum identifying code, required for the Al Qaida 9-11 network was $\mathcal{I} = \{2, 4, 6, 9, 12, 19, 20, 21, 22, 23, 24, 26, 31, 36, 37\}$. $|\mathcal{I}| = 15$, which is again consistent with the results in [31]. Similarly, those authors identified that both vertices 25

and 33 could not be separated, and their solution was to ignore 33. Initially, the IP solution presented here monitored, but did not cover, vertex 33. However, this makes little sense in context, and equation 5.8 was added to require that subjects being monitored also be covered.

5.3 Hiding Activity: OLD sets

It may be the case that a target under surveillance will suspect or know that they are being watched, and may take measures to hide their own involvement in a forthcoming attack to prevent monitoring. This is known as counter-surveillance. However, attacks still require planning and it is likely the case that a member of the network who is not directly planning the attack might still carry out detectable planning activities. Through surveillance of these inactive members, authorities might still see the indications of a pending attack by an inactive member's associates. Under these assumptions, an OLD set is appropriate to identify the targets that should be watched. In an OLD set, if a target were actively planning, they would not give indications if being directly monitored, but actions could be detected through a member's associates, within a specified coverage range. OLD sets of both $r = 1$ and $r \leq \max d(G)$, the maximum inter-vertex distance, are considered below. To ensure the model is robust against twins, the non-linear max-covering OLD set formulation from section 3.1 is used. The adjacency matrix is based on the open neighborhood:

$$\alpha_{i,j} = \begin{cases} 1 & \text{if } 1 \leq d(i, j) \leq r \\ 0 & \text{otherwise} \end{cases}$$

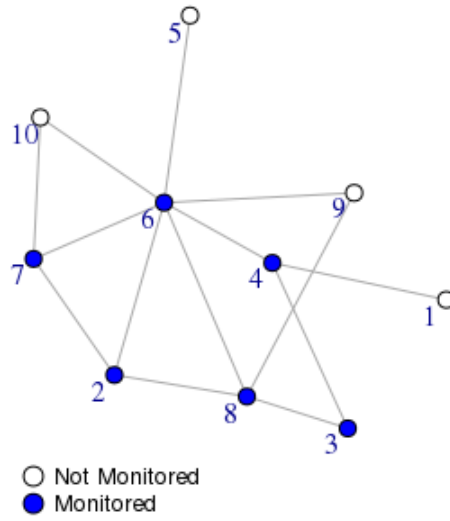


Figure 5.5: OLD set for Paris Attack Network

$$\min \gamma \sum_{j \in V} c_j x_j - (1 - \gamma) \sum_{i \in V} b_i y_i \quad (5.12)$$

$$\text{s.t.} \quad \sum_{j \in V} \alpha_{i,j} x_j \geq y_i \quad \forall i \in V \quad (5.13)$$

$$\sum_k (\alpha_{i,k} - \alpha_{j,k})^2 x_k \geq \Omega_{i,j} y_i y_j \quad \forall i, j \in V \quad (5.14)$$

$$x_i \leq y_i \quad \forall i \in V \quad (5.15)$$

$$\sum_{j \in V} x_j \leq P \quad \text{optional} \quad (5.16)$$

$$x_j \in \{0, 1\} \quad (5.17)$$

$$y_i \in \{0, 1\} \quad (5.18)$$

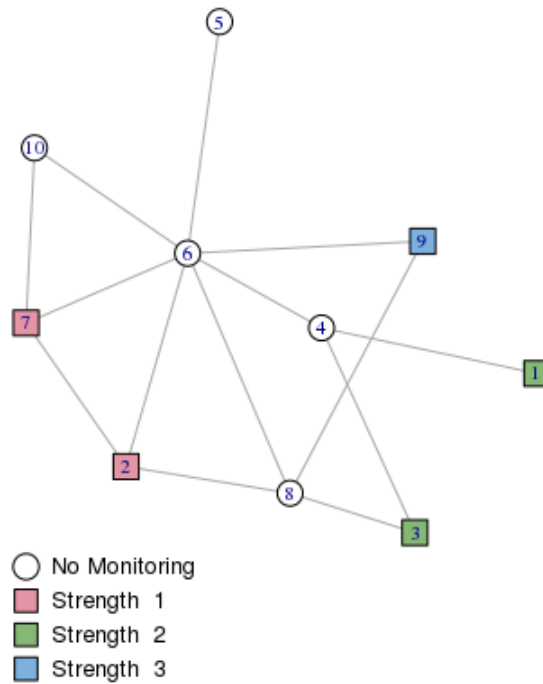


Figure 5.6: MWOLD set for Paris Attack Network

5.3.1 OLD and MWOLD sets: Paris Network

The OLD set for the Paris Network, shown in figure 5.5, indicates six targets for monitoring: $\mathcal{D} = \{2, 3, 4, 6, 7, 8\}$. It includes the five targets found in the identifying code solution from section 5.2.1, and adds vertex 3 to separate vertices 7 and 8, which otherwise would share neighbors 2 and 6 in the open construct. This network has no twins, and all network members are covered. This IP used no weights for surveillance costs or coverage costs. It must be acknowledged, the identified target set represents a significant portion of the network, and placing such a large portion under surveillance could be prohibitive in practice.

The MWOLD solution for the Paris network, shown in figure 5.6, uses five targets, one at strength 3, two at strength 2, and two at strength 1. This IP used no weights to model surveillance costs, coverage costs, or surveillance strength.

If costs of deeper surveillance are negligible, this solution offers a savings over the traditional OLD construct. When this model was rerun with any reasonable surveillance strength costs (strength 1 = 1, strength 2 \geq 1.25, strength 3 \geq 1.5), then the homogeneous solution with $r = 1$, above, is an alternative optimal, or outright optimal, solution.

5.3.2 OLD and MWOLD sets: 9-11 Attack Network

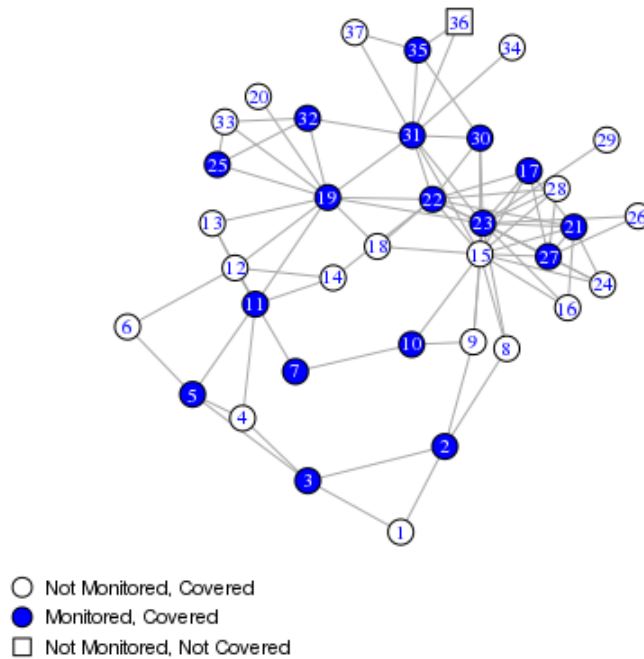


Figure 5.7: Max-Covering OLD set for 9-11 Attack Network

The OLD set for the 9-11 network, shown in figure 5.9, requires 17 surveillance targets, an increase of two from the identifying code solution above. The OLD set for the 9-11 attack network required a max covering construct due to the twin relationships between vertices 36 and 37. While these two vertices are not leaves, as discussed in section 2.3.4, each vertex is connected to 31 and 35, offering no sep-

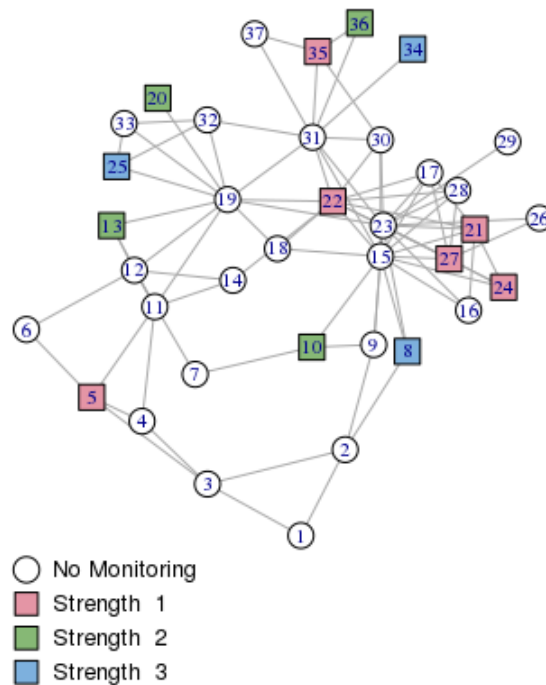


Figure 5.8: MWOLD set for 9-11 Attack Network

aration in an open construct. This differs from the closed construct and identifying code result presented, in section 5.2.2 where this vertex pair posed no difficulty. In a closed construct, each vertex could cover itself, which offered a separation because the two vertices were not connected to each other. In contrast, recall from section 5.2.2, vertices 25 and 33 were twins in a closed construct, as part of a complete subgraph. Again referencing results in section 4.1.1, complete graphs always admit OLD sets, but they do not admit identifying codes. In the open construct, these vertices pose no difficulty. Due to the assumed counter-surveillance, if vertex 25 is active, the targets at 19 and 32 will indicate activity. If vertex 33 is active, targets at 19, 25, and 32 will be active, providing the required separation between those vertices.

Figure 5.8 shows a MWOLD set for the 9-11 network, where surveillance costs are equal for all strengths. This represents, for illustrative purposes, the result

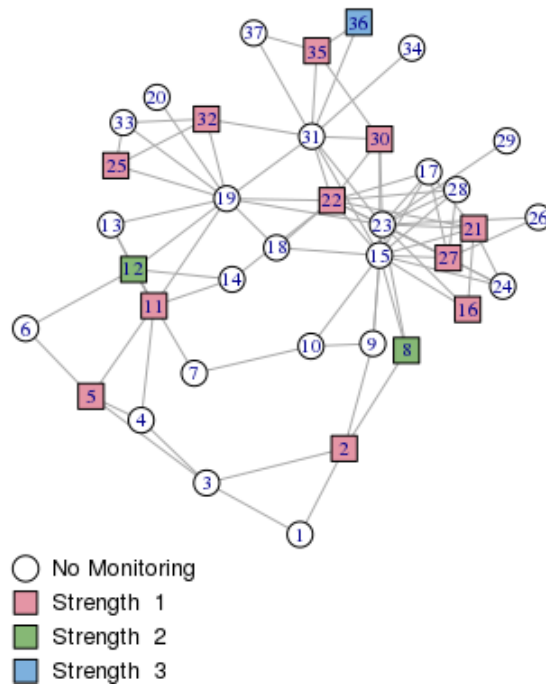


Figure 5.9: Weighted MWOLD set for 9-11 Attack Network

when there is no extra cost from watching a target closely enough to pick up on second, third, or higher order connection activity. The IP was formulated to include up to strength 5 surveillance, but the solution used nothing above strength 3. The solution contains three strength 3 targets, four strength 2 targets, and six strength 1 targets, for a total of 13 targets. The added strength of surveillance offers a savings of four targets. While not explicitly explored in this work, having fewer overall targets might be beneficial in terms of counter-detection, or the authorities' desire to keep their monitoring covert. In this sense, the added cost of fewer targets might outweigh the associated cost of monitoring a target more closely. It could also be the case that watching a target more closely offers greater counter-detection risk, and watching more targets superficially could be more beneficial.

Figure 5.9 adds extra costs for deeper surveillance on targets. Strength:cost values are: 1:1, 2:2, 3:2.5, 4:3, 5:3.5. The solution uses one strength 3 target,

and only two strength 2 targets, but 11 strength 1 targets, for a total of 14 targets. It requires one additional target from the unweighted MWOLD solution shown in figure 5.8, but is optimal given the provided weights.

5.4 Robustness: Strongly Identifying Codes

In the previous sections, strong assumptions were made regarding the detectability of activity if the perpetrator were directly under surveillance. If the perpetrator's activity were detectable with confidence, then an identifying code is appropriate. If authorities could be certain the perpetrator would hide his or her activity, then an open locating-dominating set is appropriate. However, it is likely the case that authorities cannot be certain about the potential detectability, as such certainty is extremely rare in law enforcement and intelligence operations. This suggests that a strongly identifying code, presented briefly in Chapter 1 might be most appropriate. Recall a strongly identifying code must satisfy the locating constraint for the open and closed neighborhood constructs simultaneously. As demonstrated in section 5.3.2, a feasible locating set on an open network is not necessarily feasible on the closed network, or vice versa. The dominating constraint however, need only be solved over the open construct, as a feasible dominating set on the open construct remains feasible when examining the closed construct.

Lemma 5.1 *A feasible dominating set on an open construct of a graph is also feasible on the closed construct of the same graph.*

Proof: Let D be a feasible dominating set for the open neighborhood construct of a graph G . Let \mathbf{A} , elements $\alpha_{i,j}$ represent the adjacency matrix for G , where $\alpha_{i,j} = 1$ if i, j are within the specified coverage radius. The diagonals of $\mathbf{A} = 0$

due to the open neighborhood construct. Let $x_j = 1$ if vertex j is a member of D , $x_j = 0$ otherwise. Since \mathcal{D} is a feasible dominating set, the following equation is true:

$$\sum_{j \in V} \alpha_{i,j} x_j \geq 1 \quad \forall i \in V$$

This can also be stated: for any vertex $v_i \in V$ there exists at least one vertex $v_j \in D$ within the coverage radius of v_i . When examining the closed neighborhood construct of G , the diagonals of $\mathbf{A} = 1$, but the remainder of the matrix is unchanged. For any vertex $v_i \in V$, the vertex v_j is still within the coverage radius and satisfies the dominating requirement. Therefore, D remains a feasible dominating set on the closed construct of G . ■

To model the strongly identifying code, two adjacency matrices are needed: \mathbf{A} for the open neighborhood, elements $\alpha_{i,j}$, and $\hat{\mathbf{A}}$ for the closed, elements $\hat{\alpha}_{i,j}$.

$$\alpha_{i,j} = \begin{cases} 1 & \text{if } 1 \leq d(i,j) \leq r \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{\alpha}_{i,j} = \begin{cases} 1 & \text{if } d(i,j) \leq r \\ 0 & \text{otherwise} \end{cases}$$

The IP is as follows:

$$\min \gamma \sum_{j \in V} c_j x_j - (1 - \gamma) \sum_{i \in V} b_i y_i \quad (5.19)$$

$$\text{s.t.} \sum_{j \in V} \alpha_{i,j} x_j \geq y_i \quad \forall i \in V \quad (5.20)$$

$$\sum_k (\alpha_{i,k} - \alpha_{j,k})^2 x_k \geq \Omega_{i,j} y_i y_j \quad \forall i, j \in V \quad (5.21)$$

$$\sum_k (\hat{\alpha}_{i,k} - \hat{\alpha}_{j,k})^2 x_k \geq \Omega_{i,j} y_i y_j \quad \forall i, j \in V \quad (5.22)$$

$$x_i \leq y_i \quad \forall i \in V \quad (5.23)$$

$$\sum_{j \in V} x_j \leq P \quad \text{optional} \quad (5.24)$$

$$x_j \in 0, 1 \quad (5.25)$$

$$y_i \in 0, 1 \quad (5.26)$$

5.4.1 Strongly Identifying Code: Paris Network

The strongly identifying code for the Paris network, shown in figure 5.10, contains six surveillance targets. Interestingly, the strongly identifying set is the same set of targets as the OLD set on this network. This is not always the case and is merely a coincidence for this particular network structure.

5.4.2 Strongly Identifying Code: 9-11 Network

The strongly identifying code solution for the 9-11 network, shown in figure 5.11, contains 18 targets, more than either the OLD set solution (17) or the identifying code solution (15). This is expected, as additional sensors are needed to satisfy the simultaneous locating constraints. Each of these previous solutions identified

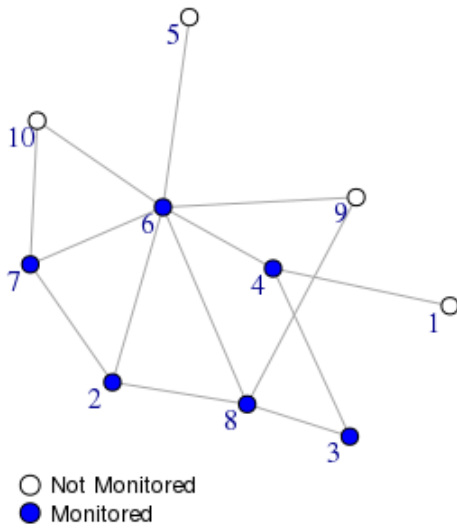


Figure 5.10: Strongly Identifying Code for Paris Attack Network

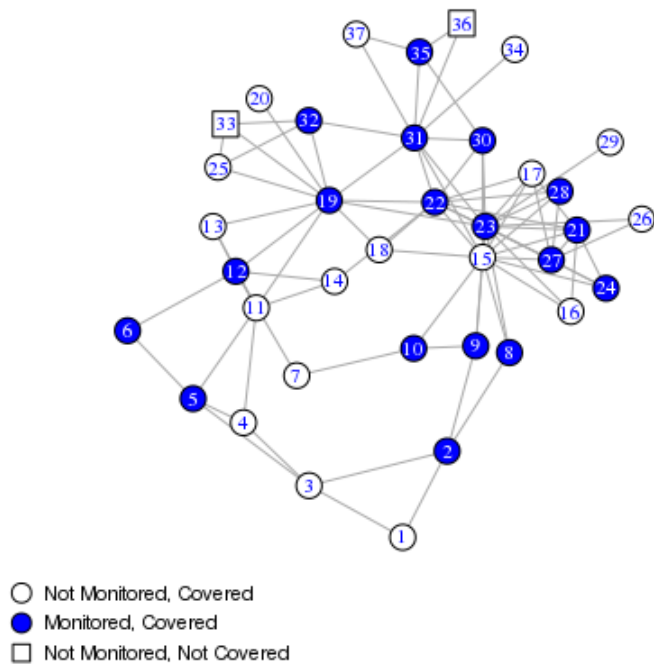


Figure 5.11: Strongly Identifying Code for 9-11 Attack Network

twin-vertices, which precluded full coverage of the network. The strongly identifying code, subject to the constraints of both OLD sets and identifying codes, must leave two vertices uncovered: vertex 33, as discussed in the identifying code solution, and vertex 36, as discussed in the OLD set solution.

5.5 Confidential Informants: Locating-Dominating Sets

Confidential informants (CI) play a key role in investigations and counter terrorism efforts [28]. According to the FBI's 2005 special report, a confidential informant is "any individual who provides useful and credible information to a Justice Law Enforcement Agency (JLEA) regarding felonious criminal activities and from whom the JLEA expects or intends to obtain additional useful and credible information regarding such activities in the future." An advantage to a confidential informant is the relationship developed between the case agent and the informant is such that the case agent will know immediately if their informant is to be the perpetrator of the attack. In this sense, when placing confidential informants, one is interested in a locating-dominating set (LDS) construct. This holds because the informants represent 3-state sensors that can independently identify when they are to be the source of the event, but otherwise give a typical yes-no response if they sense an event but are not the source. To use this concept, the ILP formulations presented earlier must be adapted, including the addition of a non-linear constraint. Recall the locating and dominating constraints of an open locating-dominating set or identifying code must hold for every vertex in the graph, except those in the LDS. Let the locating-dominating set be denoted by \mathcal{L} .

$$\text{Locating: } N[v] \cap \mathcal{L} \neq \emptyset \quad \forall v \in V \setminus \mathcal{L}.$$

$$\text{Dominating: } N[v_i] \cap \mathcal{L} \neq N[v_j] \cap \mathcal{L} \quad \forall v_i, v_j \in V \setminus \mathcal{L}.$$

The IP for a LDS is as follows:

$$\min \sum_{j \in V} c_j x_j \tag{5.27}$$

$$\text{s.t. } \sum_{j \in V} \alpha_{i,j} x_j \geq 1 - x_i \quad \forall i \in V \tag{5.28}$$

$$\sum_k (\alpha_{i,k} - \alpha_{j,k})^2 x_k \geq \Omega_{i,j} (1 - x_i)(1 - x_j) \quad \forall i, j \in V \tag{5.29}$$

$$x_j \in 0, 1 \tag{5.30}$$

5.5.1 Locating Dominating IP: Correctness

The objective function is similar to previously presented formulations, and seeks to minimize the cardinality of the locating-dominating set, subject to weighting parameters. The left hand sides of the equations 5.27-5.29 are the same as presented previously for OLD sets and identifying codes and function in the same manner. It is necessary that this IP only applies constraints to vertices that are not included in the LDS, denoted \mathcal{L} , since the constraints apply to each vertex in $G \setminus \mathcal{L}$. In a similar fashion to the IP presented in chapter 3, where the variable y_i was used to effectively turn a particular constraint off dynamically during the optimization, the right hand sides of 5.28 and 5.29 do the same. For the dominating constraint, if a vertex i is included in the LDS, the right hand side will go to zero, and impose no restriction. If the vertex is not included in the LDS, x_i will be zero, the right hand side will be one, and the constraint will be imposed. For the locating constraint, equation 5.29, if either of the vertices i or j are included in the LDS, the right hand side will go to zero and impose no constraint for that vertex pair. As

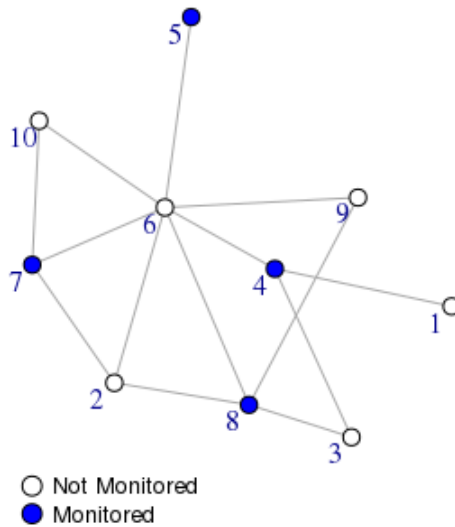


Figure 5.12: Locating-Dominating Set for Paris Attack Network

established in [4], all graphs admit locating-dominating sets (trivially, $\mathcal{L} = G$), so a maximum covering construct is not mathematically necessary. However, section 5.6 sets forth reasons why one might want to limit the number of confidential informants in a network, and demonstrates an appropriate formulation to ensure this behavior.

5.5.2 Locating-Dominating Set: Paris Network

The LDS for the Paris attack network is shown in figure 5.12. $\mathcal{L} = \{4, 5, 7, 8\}$, and $|\mathcal{L}| = 4$, one less than the identifying code construct. Vertices 4, 7, and 8 were also surveillance targets under the identifying code construct, but interestingly the LDS includes vertex 5, which is a lone vertex with a degree of one. This sort of outsider, or less connected member of the group, might be easier to turn into a confidential informant.

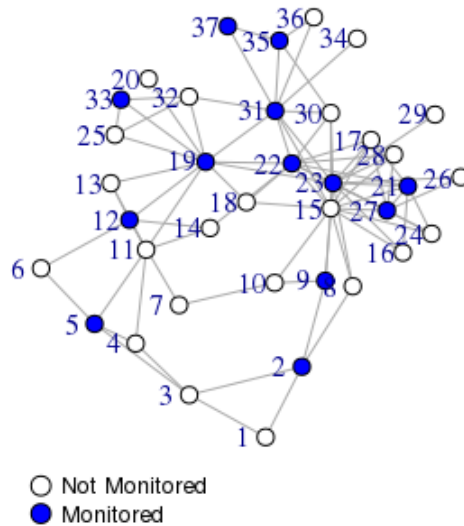


Figure 5.13: Locating-Dominating Set for 9-11 Attack Network

5.5.3 Locating Dominating Set: 9-11 Network

The LDS for the 9-11 Attack network is shown in figure 5.13. $\mathcal{L} = \{2, 5, 9, 12, 19, 21, 22, 23, 27, 31, 33, 35, 37\}$, and $|\mathcal{L}| = 13$, two fewer than required under the identifying code construct. However, attempting to establish 13 confidential informants in a network of 38 individuals is problematic, likely exposes authorities to significant counter-detection risk, and is potentially infeasible from a practical standpoint.

5.6 Combined Decision Support Tool for Terrorist Network Monitoring

Due in large part to the theoretical nature of previous identifying code, locating-dominating, and OLD set work, each situation has been examined independently and assumed to be correct for the discussed application. There do not appear to

be any results in the literature that explore various constructs on a single application, attempt to identifying the most appropriate construct for a given application, or combine multiple constructs into a single analysis or tool. However, in the context of monitoring terrorist networks, a combined tool is most appropriate for the decision maker. An IP that could identify both targets for surveillance as well as the ideal persons to turn into confidential informants would best assist the intelligence officer(s) by providing an optimized strategy to monitor a particular network.

Let $\alpha_{r,i,j}$ represent the coverage matrix elements for coverage radius r , as defined in previous chapters. Let $x_{r,j} = 1$ represent vertex j being placed under surveillance of strength r in a mixed weight construct as defined in section 5.3. Let $y_{r,j} = 1$ represent a confidential informant with strength r at vertex j . Let P represent the maximum number of confidential informants allowed for the network. W_r^{surv} is a cost parameter for the strength of surveillance (larger r values represent deeper penetration and are more costly to establish); W_r^{ci} is a similar cost parameter for how deeply a CI can provide information (a more deeply connected CI will presumably take additional resources to turn and maintain). C_j^{surv} is the cost of putting a specific vertex/member under surveillance, and C_j^{ci} is the cost of turning a specific member to be a CI.

$$\min \sum_{j \in V} \sum_{r \in R} (W_r^{surv} + C_j^{surv})x_{r,j} + (W_r^{ci} + C_j^{ci})y_{r,j} \quad (5.31)$$

$$\text{s.t.} \sum_{j \in V} \sum_{r \in R} \alpha_{r,i,j}(x_{r,j} + y_{r,j}) \geq 1 - \sum_{r \in R} y_{r,i} \quad \forall i \in V \quad (5.32)$$

$$\begin{aligned} & \sum_{r \in R} \sum_{k \in V} (\alpha_{i,k} - \alpha_{j,k})^2 (x_{r,k} + y_{r,k}) \\ & \geq \Omega_{i,j} (1 - \sum_{r \in R} y_{r,i}) (1 - \sum_{r \in R} y_{r,j}) \quad \forall i, j \in V \end{aligned} \quad (5.33)$$

$$\sum_{r \in R} (x_{r,j} + y_{r,j}) \leq 1 \quad \forall j \in V \quad (5.34)$$

$$\sum_{r \in R} y_{r,j} \leq P \quad \forall j \in V \quad (5.35)$$

$$x, y \in \{0, 1\}^n \quad (5.36)$$

5.6.1 Combined IP: Correctness

The objective function 5.31 seeks to minimize the total cost, which is desired. If all weights/costs are equal, then the model will seek a feasible solution with the fewest number of targets (surveillance and confidential informant combined), which follows logically. The dominating constraint 5.32, is similar to previous dominating constraints, however one must consider a vertex's proximity to either a vertex under surveillance or to a confidential informant. For each vertex in V , the proximity is considered against every other vertex at all possible weights. This is done through examination of the incoming ball of each vertex. The right hand side ensures that each vertex, i , has a least one neighbor in the set, unless vertex i is a confidential informant. If i is a confidential informant of any strength, then $\sum_{r \in R} y_{r,i} = 1$, and the right hand side will go to zero, imposing no constraint for

vertex i . The locating constraint is a similar adaptation to previous locating constraint equations. Each vertex pair $i, j \in V \setminus L$ must have at least one neighbor in the set, either a surveillance target or a confidential informant, that separates the pair and ensures a unique intersection with the set. Again, this constraint is not required if either i or j is a confidential informant, but is still required if one or both is simply under surveillance. The right hand side of the equation will go to zero if either i or j is an informant and pose no restriction on the model. If neither i nor j are informants, and i and j have shared neighbors, then the right hand side will equal 1, and function as normal. Equation 5.34 ensures no more than a single method (surveillance, confidential informant) and a single weight is selected for each vertex. Equation 5.35 offers a “fixed-P” construct as presented in Chapter 3, and optionally provides that no more than a specified number P of confidential informants are in a set for a single terrorist network, as too many informants could raise suspicion or end up reporting on each other unknowingly.

5.6.2 Combined Monitoring Plan: Paris Network

The Paris Attack Network monitoring solution, shown in figure 5.14, is based on even weights and costs and a maximum of one confidential informant. Using more than one informant in such a small network seems to pose too great a risk for counter-detection. The solution identifies 5 targets. One confidential informant, of strength 1, three surveillance targets of strength 1, and one surveillance target of strength 2. This is the same number of targets as the identifying code and the MWOLD set. It appears that the network is too small for the single confidential informant to significantly reduce the total number of surveillance targets.

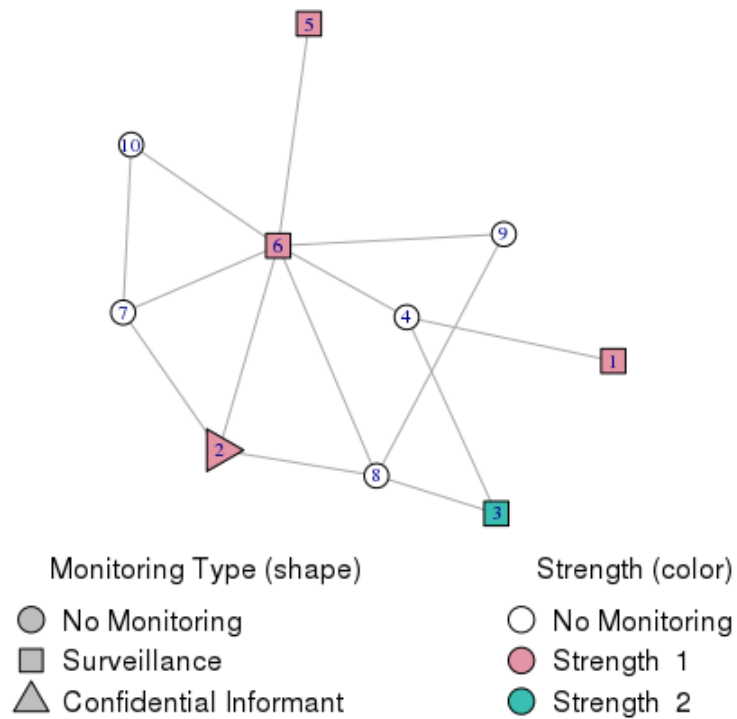


Figure 5.14: Combined Monitoring Solution for Paris Attack Network

5.6.3 Combined Monitoring Plan: 9-11 Network

For the 9-11 Attack network, the model was limited to a maximum of two confidential informants, to prevent counter detection and compromise of the informants. The strength weights are all set to 1, for both surveillance and confidential informants. The optimal result, shown in figure 5.15, uses two confidential informants: one of strength 1 and one of strength 2. It places 11 people under surveillance: six at strength 1, and 5 at strength 2. This is the same overall number of targets as the MWOLD set solution and the locating-dominating set solution.

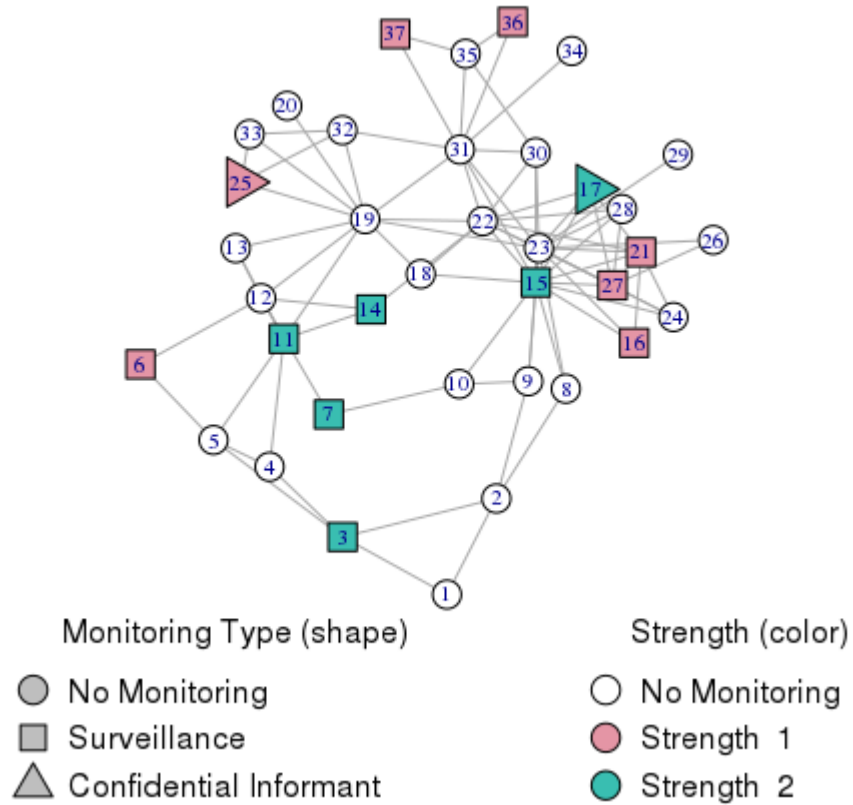


Figure 5.15: Combined Monitoring Solution for 9-11 Attack Network

| Construct | Monitoring | Notes |
|---------------------------|--------------|--|
| Identifying Code | Surveillance | All activity detectable |
| OLD/MWOLD set | Surveillance | Perpetrator activity hidden against direct surveillance |
| Strongly Identifying Code | Surveillance | Robust against hidden activity assumptions |
| Locating-Dominating Set | CI | CI can self report, indicate knowledge of associate activity |
| Combined | Both | |

Table 5.1: Terrorist Network Monitoring - Formulation Types

| | IC | OLD | MWOLD | SIC | LDS | Combined |
|------------------|----|-----|-------|-----|-----|----------|
| Paris: # Targets | 5 | 6 | 5 | 6 | 4 | 5 |
| 9-11: # Targets | 15 | 17 | 13 | 18 | 13 | 13 |

Table 5.2: Terrorist Network Monitoring - Comparison

5.7 Conclusions

While this work does not offer a typical hypothesis and conclusion, it offers significant observations about the behavior and applications of location-domination concepts. For each of the networks, the number of required targets ranged from about one-third to about one-half of the total number of actors in the network. This number seems extremely high given typical government agency and law enforcement resourcing. It is more likely that a construct needs to be developed to identify a cascading subgraph approach, where 4-5 members (in a 40-50 person network) could be monitored, and activity would trigger authorities to identify an active subgroup of the network, and reidentify 4-5 new targets in the subgroup to monitor for additional activity, continuing this process until the perpetrator is identified. The long planning lead times for coordinated attacks or criminal activity seem to make this potential approach feasible, and worth of consideration in future work.

Table 5.2 shows the number of targets required for each solution as a measure of performance. Strongly identifying codes seem to have the worst performance, which is logical given their dual constraint requirement. The mixed weight systems, both MWOLD and the mixed-weight combined approach had the best performance, though weight was not significantly considered due to the difficulties establishing reasonable weighting structures without disclosing sensitive information. The locating-dominating set constructs had equally good performance, but relied on an extremely high number of informants for a given network. These

results are interesting theoretically, but highly impractical.

In both of the terrorist networks examined in this chapter, a single vertex was identified for monitoring in every construct. For the Paris network, this was vertex 7; for the 9-11 network, vertex 21. Neither of these vertices were the highest degree vertex in their respective graphs, nor did they offer the greatest number of separations (instances where that vertex was in the ball of one vertex of a vertex pair, but not the other). This suggests that development of greedy algorithms for identifying near-optimal sets may be difficult, as there is no clear link between the major quantitative characteristics of a vertex and its likelihood of being included in a set.

Nevertheless, this chapter does demonstrate the feasibility of identifying codes, open locating-dominating sets, mixed weight open locating-dominating sets, locating-dominating sets, and the new combined construct for real world applications, such as monitoring a terrorist or criminal network. Each solution presented in this chapter was identified in less than 60 seconds using AMPL and a Gurobi solver on a standard Linux desktop, indicating that real world applications are computationally reasonable. With further development, these tools could prove valuable in the nation's ongoing struggle against these subversive groups.

Appendix A

AMPL Code: Models and Example Data Files

AMPL was used exclusively for modeling and solving the OLD sets in this paper. Listed below for reference are the various models used and example data files for each model. The models included are:

- Pre-constructed OLD set, with β matrix. Data file for a 5-vertex house (figure 1.2).
- Dynamic OLD set with Ω matrix. Data file for a 5-vertex house.
- Maximum Covering OLD set. Data file for three vertex path. (figure 4.3).
- Mixed Weight OLD set. Data file for 10-vertex tree (figure 4.4).
- Combined Monitoring Tool. Data set for Paris Network (figure 5.14).

Model for pre-constructed OLD set:

```
set NODES;
set ROWPAIRS;

param Adj {NODES, NODES} >= 0;  # adjacency matrix
param Beta {ROWPAIRS, NODES} >= 0;
# abs value of difference of Adj rows, 2 edges apart

var x {NODES} binary;
minimize Objective: sum {j in NODES} x[j];

subject to Covering {i in NODES}:
# every adjacent node must be covered
    sum {j in NODES} Adj[i,j] * x[j] >= 1;

subject to Separating {k in ROWPAIRS}:
# every node 2 edges away must be covered
    sum {j in NODES} Beta [k,j] * x[j] >= 1;
```

Data File for 5-vertex House. fig 1.2

```
data;   ### DATA STARTS HERE ###
```

```
set NODES := 1 2 3 4 5;
```

```
set ROWPAIRS := 14 24 25 35;
```

```
param Adj : 1 2 3 4 5 :=
```

```
1    0 1 1 0 1
```

```
2    1 0 1 0 0
```

```
3    1 1 0 1 0
```

```
4    0 0 1 0 1
```

```
5    1 0 0 1 0;
```

```
param Beta : 1 2 3 4 5 :=
```

```
14   0 1 0 0 0
```

```
24   1 0 0 0 1
```

```
25   0 0 1 1 0
```

```
35   0 1 0 0 0;
```

Model for Dynamic OLD set.

```
set NODES;

param Adj {NODES, NODES} >= 0; # adjacency matrix
param Omega {NODES, NODES} >= 0;
# matrix indicating shared neighborhoods

var x {NODES} binary; # x=1 if vertex is in the OLD set

minimize Objective: sum {j in NODES} x[j];

subject to Dominating {i in NODES}:
# every node must be covered
    sum {j in NODES} Adj[i,j] * x[j] >= 1;

subject to Locating {i in NODES, j in NODES}:
# Omega(i,j)=1 if d(i,j)<2*r, 0 otherwise. Indicates
# shared neighbors. Every pair with shared neighbors
# must have at least one distinguishing neighbor in the set.
    sum {k in NODES} (Adj[i,k]-Adj[j,k])^2 * x[k] >= Omega[i,j];
```

Data File for 5-vertex House with Omega

```
data; ##### DATA STARTS HERE #####
```

```
set NODES := 1 2 3 4 5;
```

```
param Adj: 1 2 3 4 5 :=  
1 0 1 1 0 1  
2 1 0 1 0 0  
3 1 1 0 1 0  
4 0 0 1 0 1  
5 1 0 0 1 0;
```

```
param Omega: 1 2 3 4 5 :=  
1 0 1 1 1 1  
2 1 0 1 1 1  
3 1 1 0 1 1  
4 1 1 1 0 1  
5 1 1 1 1 0;
```

Model for Maximum Covering OLD set

```
set NODES;

param Adj {NODES, NODES} binary; # adjacency matrix
param Omega {NODES, NODES} >= 0; # shared neighborhoods
param Gamma = 1; #Weighting parameter for set size
param Zi = 10; #Weighting parameter for uncovered vertices

var x {NODES} binary;
var y {NODES} binary;

minimize Objective:Gamma*sum{j in NODES}x[j] - Zi*sum{k in NODES}y[k];

subject to Dominating {i in NODES}:
# every included vertex must be covered
    sum {j in NODES} Adj[i,j] * x[j] >= y[i];

subject to Locating {i in NODES, j in NODES}:
# every pair of covered vertices must be separated
    sum {k in NODES} (Adj[i,k]-Adj[j,k])^2*x[k]>= Omega[i,j]*y[i]*y[j];
```

Data File for 3 vertex Triad, figure 3.2

```
data; ##### DATA STARTS HERE #####
```

```
set NODES := 1 2 3;
```

```
param Adj : 1 2 3:=  
            1 0 1 1  
            2 1 0 0  
            3 1 0 0;
```

```
param Omega: 1 2 3:=  
            1 0 1 1  
            2 1 0 1  
            3 1 1 0;
```

Model for Mixed Weight OLD set

```
set NODES;
set Radii;
param Adj {Radii, NODES, NODES} >= 0; # adjacency matrix
param Omega {NODES, NODES} >= 0; # shared neighborhood
param Weight {Radii}; # weighting function for sensor strengths
var x {Radii, NODES} binary;

minimize Objective:sum {r in Radii, j in NODES}x[r,j]*Weight[r];

subject to Dominating {i in NODES}:
# every adjacent node must be covered
    sum {r in Radii, j in NODES} Adj[r,i,j] * x[r,j] >= 1;

subject to Locating {i in NODES, j in NODES}:
# every pair of nodes must have at least one sensor
# in one neighborhood but not the other
    sum{r in Radii,k in NODES}(Adj[r,k,i]-Adj[r,k,j])^2*x[r,k]
    >=Omega[i,j];

subject to OnlyOneSensor {j in NODES}:
#No more than one sensor may be placed at a given vertex
    sum {r in Radii} x[r,j] <= 1;
```


Data File for 10-vertex Scale Free, Mixed-Weight $R = 9$

```
data; ##### DATA STARTS HERE #####
```

```
set NODES := 1 2 3 4 5 6 7 8 9 10;
```

```
set Radii := 1 2 3 4 5;
```

```
param Weight:=
```

```
1 2
```

```
2 2
```

```
3 2
```

```
4 2
```

```
5 2;
```

```
param Adj:=
```

```
[1,*,*]: 1 2 3 4 5 6 7 8 9 10:=
```

```
1      0 1 1 1 0 0 0 0 0 1
```

```
2      1 0 0 0 0 0 1 0 0 0
```

```
3      1 0 0 0 0 1 0 0 0 0
```

```
4      1 0 0 0 1 0 0 1 0 0
```

```
5      0 0 0 1 0 0 0 0 1 0
```

```
6      0 0 1 0 0 0 0 0 0 0
```

```
7      0 1 0 0 0 0 0 0 0 0
```

```
8      0 0 0 1 0 0 0 0 0 0
```

```
9      0 0 0 0 1 0 0 0 0 0
```

```
10     1 0 0 0 0 0 0 0 0 0
```

```
[2,*,*]: 1 2 3 4 5 6 7 8 9 10:=
```

| | |
|----|---------------------|
| 1 | 0 1 1 1 1 1 1 1 0 1 |
| 2 | 1 0 1 1 0 0 1 0 0 1 |
| 3 | 1 1 0 1 0 1 0 0 0 1 |
| 4 | 1 1 1 0 1 0 0 1 1 1 |
| 5 | 1 0 0 1 0 0 0 1 1 0 |
| 6 | 1 0 1 0 0 0 0 0 0 0 |
| 7 | 1 1 0 0 0 0 0 0 0 0 |
| 8 | 1 0 0 1 1 0 0 0 0 0 |
| 9 | 0 0 0 1 1 0 0 0 0 0 |
| 10 | 1 1 1 1 0 0 0 0 0 0 |

[3,*,*]: 1 2 3 4 5 6 7 8 9 10:=

| | |
|----|---------------------|
| 1 | 0 1 1 1 1 1 1 1 1 1 |
| 2 | 1 0 1 1 1 1 1 1 0 1 |
| 3 | 1 1 0 1 1 1 1 1 0 1 |
| 4 | 1 1 1 0 1 1 1 1 1 1 |
| 5 | 1 1 1 1 0 0 0 1 1 1 |
| 6 | 1 1 1 1 0 0 0 0 0 1 |
| 7 | 1 1 1 1 0 0 0 0 0 1 |
| 8 | 1 1 1 1 1 0 0 0 1 1 |
| 9 | 1 0 0 1 1 0 0 1 0 0 |
| 10 | 1 1 1 1 1 1 1 1 0 0 |

[4,*,*]: 1 2 3 4 5 6 7 8 9 10:=

| | |
|---|---------------------|
| 1 | 0 1 1 1 1 1 1 1 1 1 |
| 2 | 1 0 1 1 1 1 1 1 1 1 |

```

3      1 1 0 1 1 1 1 1 1 1
4      1 1 1 0 1 1 1 1 1 1
5      1 1 1 1 0 1 1 1 1 1
6      1 1 1 1 1 0 1 1 0 1
7      1 1 1 1 1 1 0 1 0 1
8      1 1 1 1 1 1 1 0 1 1
9      1 1 1 1 1 0 0 1 0 1
10     1 1 1 1 1 1 1 1 1 0

```

[5,*,*]: 1 2 3 4 5 6 7 8 9 10:=

```

1      0 1 1 1 1 1 1 1 1 1
2      1 0 1 1 1 1 1 1 1 1
3      1 1 0 1 1 1 1 1 1 1
4      1 1 1 0 1 1 1 1 1 1
5      1 1 1 1 0 1 1 1 1 1
6      1 1 1 1 1 0 1 1 1 1
7      1 1 1 1 1 1 0 1 1 1
8      1 1 1 1 1 1 1 0 1 1
9      1 1 1 1 1 1 1 1 0 1
10     1 1 1 1 1 1 1 1 1 0

```

param Omega: 1 2 3 4 5 6 7 8 9 10:=

```

1      0 1 1 1 1 1 1 1 1 1
2      1 0 1 1 1 1 1 1 1 1
3      1 1 0 1 1 1 1 1 1 1
4      1 1 1 0 1 1 1 1 1 1

```

5 1 1 1 1 0 1 1 1 1 1
6 1 1 1 1 1 0 1 1 1 1
7 1 1 1 1 1 1 0 1 1 1
8 1 1 1 1 1 1 1 0 1 1
9 1 1 1 1 1 1 1 1 0 1
10 1 1 1 1 1 1 1 1 1 0 ;

Model for Combined Monitoring Tool

```
set NODES;

set Radii;

#adjacency matrix
param Adj {Radii, NODES, NODES} >= 0;

# matrix indicating shared neighborhoods
param Omega {NODES, NODES} >= 0;

# weighting function for depth/range/strengths (WEIGHT)
param Surv_Depth_Cost {Radii};

# weighting function for CI depth/range/strength
param CI_Depth_Cost {Radii};

# cost of placing a specific vertex under surveillance
param Surv_Node_Cost {NODES};

# cost of turning specific vertex to confidential informant
param CI_Node_Cost {NODES};

param MaxCI;

## note: Omega reflects either r-complete or is generated using
##2R+1 if a maximum r value is known.

var x {Radii, NODES} binary;
var y {Radii, NODES} binary;

minimize Objective: sum {r in Radii, j in NODES} (x[r,j] *
  (Surv_Depth_Cost [r] + Surv_Node_Cost [j]) + y[r,j] *
  (CI_Depth_Cost [r] + CI_Node_Cost [j]));
```

every node that is not a CI must be covered

subject to Dominating {i in NODES}:

$$\sum \{j \text{ in NODES}\} \sum \{r \text{ in Radii}\} (\text{Adj}[r,i,j] * (x[r,j] + y[r,j])) \geq 1 - \sum \{r \text{ in Radii}\} y[r,i];$$

#every pair of nodes must have at least one sensor in one neighborhood but not the

subject to Locating {i in NODES, j in NODES}:

$$\sum \{k \text{ in NODES}\} \sum \{r \text{ in Radii}\} ((\text{Adj}[r,k,i] - \text{Adj}[r,k,j])^2 * (x[r,k] + y[r,k])) \geq \text{Omega}[i,j] * (1 - \sum \{r \text{ in Radii}\} y[r,i]) * (1 - \sum \{r \text{ in Radii}\} y[r,j]);$$

subject to OnlyOneSensor {j in NODES}:

$$\sum \{r \text{ in Radii}\} (x[r,j] + y[r,j]) \leq 1;$$

subject to MaxCIAAllowed:

$$\sum \{r \text{ in Radii}\} \sum \{j \text{ in NODES}\} y[r,j] \leq \text{MaxCI};$$

Data for Paris Network Combined

data;

DATA STARTS HERE

set NODES := 1 2 3 4 5 6 7 8 9 10;

set Radii := 1 2 3;

param MaxCI:= 1;

param Surv_Depth_Cost:=

1 1

2 1

3 1;

param Surv_Node_Cost:=

1 1

2 1

3 1

4 1

5 1

6 1

7 1

8 1

9 1

10 1;

```
param CI_Depth_Cost:=
```

```
1 1  
2 1  
3 1;
```

```
param CI_Node_Cost:=
```

```
1 1  
2 1  
3 1  
4 1  
5 1  
6 1  
7 1  
8 1  
9 1  
10 1;
```

```
param Adj:=
```

```
[1,*,*]: 1 2 3 4 5 6 7 8 9 10:=
```

```
1    1 0 0 1 0 0 0 0 0 0  
2    0 1 0 0 0 1 1 1 0 0  
3    0 0 1 1 0 0 0 1 0 0  
4    1 0 1 1 0 1 0 0 0 0  
5    0 0 0 0 1 1 0 0 0 0  
6    0 1 0 1 1 1 1 1 1 1  
7    0 1 0 0 0 1 1 0 0 1
```



```
8    0 1 1 0 0 1 0 1 1 0
9    0 0 0 0 0 1 0 1 1 0
10   0 0 0 0 0 1 1 0 0 1
```

[2,*,*]: 1 2 3 4 5 6 7 8 9 10:=

```
1    1 0 1 1 0 1 0 0 0 0
2    0 1 1 1 1 1 1 1 1 1
3    1 1 1 1 0 1 0 1 1 0
4    1 1 1 1 1 1 1 1 1 1
5    0 1 0 1 1 1 1 1 1 1
6    1 1 1 1 1 1 1 1 1 1
7    0 1 0 1 1 1 1 1 1 1
8    0 1 1 1 1 1 1 1 1 1
9    0 1 1 1 1 1 1 1 1 1
10   0 1 0 1 1 1 1 1 1 1
```

[3,*,*]: 1 2 3 4 5 6 7 8 9 10:=

```
1    1 1 1 1 1 1 1 1 1 1
2    1 1 1 1 1 1 1 1 1 1
3    1 1 1 1 1 1 1 1 1 1
4    1 1 1 1 1 1 1 1 1 1
5    1 1 1 1 1 1 1 1 1 1
6    1 1 1 1 1 1 1 1 1 1
7    1 1 1 1 1 1 1 1 1 1
8    1 1 1 1 1 1 1 1 1 1
9    1 1 1 1 1 1 1 1 1 1
```

```
10    1 1 1 1 1 1 1 1 1 1;
```

```
param Omega: 1 2 3 4 5 6 7 8 9 10:=
```

```
1      0 1 1 1 1 1 1 1 1 1
```

```
2      1 0 1 1 1 1 1 1 1 1
```

```
3      1 1 0 1 1 1 1 1 1 1
```

```
4      1 1 1 0 1 1 1 1 1 1
```

```
5      1 1 1 1 0 1 1 1 1 1
```

```
6      1 1 1 1 1 0 1 1 1 1
```

```
7      1 1 1 1 1 1 0 1 1 1
```

```
8      1 1 1 1 1 1 1 0 1 1
```

```
9      1 1 1 1 1 1 1 1 0 1
```

```
10     1 1 1 1 1 1 1 1 1 0;
```

Appendix B

Pseudo Code for Matrix Generation

Data: Graph Size (V), Max Coverage Radius (R), Adj. Matrix (A)

Result: Coverage (C) and Omega (Ω) Matrices, AMPL data file

Generate shortest path distance matrix using Floyd-Warshall Algorithm

(*algorithm 2, below*). Let $\mathbf{d} \leftarrow$ shortest path matrix;

for $r = 1$ to V **do**

for $i = 1$ to V **do**

for $j = 1$ to V **do**

if $d(i, j) < r$ **then** $C_r(i, j) = 1$;

else $C_r(i, j) = 0$;

end

end

 Write C_r to .dat file;

end

%Write Omega Matrix;

for $i = 0$ to V **do**

for $j = 0$ to V **do**

if $d(i, j) < R + 1 \parallel i \neq j$ **then** $\Omega(i, j) = 1$;

else $\Omega(i, j) = 0$;

end

end

% Write Omega to .dat file;

Algorithm 1: Matrix Generation Pseudocode
Floyd-Warshall Algorithm[1]

Data: Adjacency Matrix(**a**)

Result: Shortest Path Distance Matrix (**d**)

```
;
% initialize;
if  $a(i,j)=0$  then
    |  $A(i,j) = M$  % big  $M \sim \infty$ ;
else
    |  $A(i,j) = a(i,j)$ ;
end
for  $k=0$  to  $V$  do
    |
    | for  $i=0$  to  $V$  do
    | | for  $j=0$  to  $V$  do
    | | | if  $A(i, k) + A(k, j) < A(i, j)$  then
    | | | |  $A(i, j) = A(i, k) + A(k, j)$ ;
    | | | | % if path from i to k to j is shorter than i to j, update with
    | | | | shorter path distance;
    | | | end
    | | end
    | end
end
```

Algorithm 2: Floyd-Warshall Algorithm

Bibliography

- [1] RAVINDRA K. AHUJA, THOMAS L. MAGNANTI, AND JAMES B. ORLIN. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.
- [2] ALBERT-LÁSZLÓ BARABÁSI, RÉKA ALBERT, AND HAWOONG JEONG. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(1–4):69–77, 6/15 2000.
- [3] KAUSTAV BASU, CHENYANG ZHOU, ARUNABHA SEN, AND VICTORIA HORAN GOLIBER. A novel graph analytic approach to monitor terrorist networks. *arXiv preprint arXiv:1902.02836*, 2019.
- [4] I. CHARON, O. HUDRY, AND A. LOBSTEIN. Identifying and locating-dominating codes: Np-completeness results for directed graphs. *Information Theory, IEEE Transactions on*, 48(8):2192–2200, Aug 2002.
- [5] IRÈNE CHARON, OLIVIER HUDRY, AND ANTOINE LOBSTEIN. Minimizing the size of an identifying or locating-dominating code in a graph is NP-hard. *Theoretical Computer Science*, 290(3):2109–2120, 2003.
- [6] IRÈNE CHARON, IIRO HONKALA, OLIVIER HUDRY, AND ANTOINE LOBSTEIN. The minimum density of an identifying code in the king lattice. *Discrete Mathematics*, 276(1–3):95 – 109, 2004. 6th International Conference on Graph Theory.
- [7] MUSTAPHA CHELLALI, NADER JAFARI RAD, SUK JAI SEO, AND PETER JAMES SLATER. On open neighborhood locating-dominating in graphs. *Electronic Journal of Graph Theory and Applications (EJGTA)*, 2(2):87–98, 2014.
- [8] FAN RK CHUNG. Lectures on spectral graph theory. *CBMS Lectures*, 1996. University of Pennsylvania.
- [9] DANIEL W. CRANSTON AND GEXIAN YU. A new lower bound on the density of vertex identifying codes for the infinite hexagonal grid. *The Electronic Journal of Combinatorics*, 16, 2009.

- [10] MARK S DASKIN. *Network and discrete location: models, algorithms, and applications*. John Wiley & Sons, 2011.
- [11] NATIONAL CONSORTIUM FOR THE STUDY OF TERRORISM AND RESPONSES TO TERRORISM (START). Global terrorism database, 2018. Retrieved from <https://www.start.umd.edu/gtd>.
- [12] FLORENT FOUCAUD, ELEONORA GUERRINI, MATJAŽ KOVŠE, REZA NASERASR, ALINE PARREAU, AND PETRU VALICOV. Extremal graphs for the identifying code problem. *European Journal of Combinatorics*, 32(4):628 – 638, 2011.
- [13] ROBERT FOURER, DAVID M. GAY, AND BRIAN W. KERNIGHAN. *AMPL: A Modeling Language for Mathematical Programming*. Thomson Learning, 2 edition, 2003.
- [14] MICHAEL R. GAREY AND DAVID S. JOHNSON. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
- [15] R. M. GIVENS, R. K. KINCAID, WEIZHEN MAO, AND GEXIN YU. Mixed-weight open locating-dominating sets. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, March 2017.
- [16] FRANK HARARY AND RA MELTER. On the metric dimension of a graph. *Ars Combin*, 2(191-195):1, 1976.
- [17] TERESA W. HAYNES, STEPHEN T. HEDETNIEMI, AND PETER J. SLATER. *Fundamentals of Domintion in Graphs*. Marcel Dekker, Inc., 1998.
- [18] TERESA W HAYNES, MICHAEL A HENNING, AND JAMIE HOWARD. Locating and total dominating sets in trees. *Discrete Applied Mathematics*, 154(8):1293–1300, 2006.
- [19] IIRO HONKALA, TERO LAIHONEN, AND SANNA RANTO. On strongly identifying codes. *Discrete Mathematics*, 254(1):191 – 205, 2002.
- [20] VICTORIA HORAN, STEVE ADACHI, AND STANLEY BAK. A comparison of approaches for finding minimum identifying codes on graphs. *Quantum Information Processing*, 15(5):1827–1848, 2016.
- [21] AMPL OPTIMIZATION INC. Gurobi for ampl.
- [22] M. G. KARPOVSKY, K. CHAKRABARTY, AND L. B. LEVITIN. On a new class of codes for identifying vertices in graphs. *IEEE Transactions on Information Theory*, 44(2):599–611, Mar 1998.

- [23] ASHLEY KILGORE, SHELDON JACOBSON, MICHAEL TRICK, AND WALT DEGRANGE. Resoundingly human, march 2019. INFORMS Podcast, 3 2019.
- [24] REX KINCAID, ALLISON OLDHAM, AND GEXIN YU. Optimal open-locating-dominating sets in infinite triangular grids. *Discrete Applied Mathematics*, 193:139–144, OCT 1 2015.
- [25] SARAH JOYCE KUNKLER. Finding the minimum randic index. *Honors thesis*, 2012. College of William & Mary.
- [26] TERO LAIHONEN. Optimal codes for strong identification. *European Journal of Combinatorics*, 23(3):307 – 313, 2002.
- [27] ANTOINE LOBSTEIN. Watching systems, identifying, locating-dominating and discriminating codes in graphs. <http://perso.telecom-paristech.fr/lobstein/debutBIBidetlocdom.pdf>, January 2017.
- [28] FEDERAL BUREAU OF INVESTIGATION. The federal bureau of investigation’s compliance with the attorney general’s investigative guidelines, special report, 2005.
- [29] B.S. PANDA AND ARTI PANDEY. Algorithmic aspects of open neighborhood location-domination in graphs. *Discrete Applied Mathematics*, 216:290–306, 2017.
- [30] ARTI PANDEY. Open neighborhood locating-dominating set in graphs: Complexity and algorithms. *14th International Conference on Information Technology*, 2015.
- [31] ARUNABHA SEN, VICTORIA HORAN GOLIBER, CHENYANG ZHOU, AND KAUSTAV BASU. Terrorist network monitoring with identifying code. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 329–339. Springer, 2018.
- [32] SUK J. SEO AND PETER J. SLATER. Open neighborhood locating-dominating in trees. *Discrete Applied Mathematics*, 159(6):484–489, MAR 28 2011.
- [33] PETER SLATER AND SUK J. SEO. Open neighborhood locating-dominating sets. *Australian Journal of Combinatorics*, 46, 2010.
- [34] PETER J. SLATER. Domination and location in acyclic graphs. *Networks*, 17(1):55–64, 1987.
- [35] SIBEL BILGE SONUÇ. *Multi-Stage MIP Formulation and Exact Solution Approaches for Combinatorial Network Problems*. PhD thesis, University of Florida, 2016.

- [36] JUKKA SUOMELA. Approximability of identifying codes and locating-dominating codes. *Information Processing Letters*, 103(1):28–33, 2007.
- [37] D. BLAIR SWEIGART AND REX KINCAID. Maximum covering formulation for open locating-dominating sets. In *Operations Research Proceedings, 2016*, pages 259–266, 2016.
- [38] D. BLAIR SWEIGART, JULIA PRESNELL, AND REX KINCAID. An integer program for open locating dominating sets and its results on the hexagon-triangle infinite grid and other graphs. *2014 Systems and Information Engineering Design Symposium (SIEDS)*, 2014.
- [39] C VAN DE PANNE. Programming with a quadratic constraint. *Management Science*, 12(11):798–815, 1966.
- [40] ROBERT J. VANDERBEI. *Linear Programming: Foundations and Extensions*. Springer, Princeton, NJ, 3 edition, 2008.
- [41] PHILIP WOLFE. The simplex method for quadratic programming. *Econometrica: Journal of the Econometric Society*, pages 382–398, 1959.
- [42] YI-CHUN XU AND REN-BIN XIAO. Identifying code for directed graph. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, volume 2, pages 97–101. IEEE, 2007.
- [43] YI-CHUN XU AND REN-BIN XIAO. Solving the identifying code problem by a genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(1):41–46, 2007.