2013

Finding Open Locating Dominating Sets on Infinite and Finite Graphs

Allison Oldham
College of William and Mary

Finding Open Locating Dominating Sets on Infinite and Finite Graphs

Allison Oldham

Department of Mathematics,

College of William and Mary,

Williamsburg, VA 23187-8795, USA

Email: aloldham@email.wm.edu

May 9, 2013

Abstract

An Open Locating Dominating set is a subset of vertices of a graph such that (i) every vertex in the graph has at least one neighbor in the Open Locating Dominating set and (ii) no two vertices in the graph have the same set of neighbors in the Open Locating Dominating set. Such a set can provide insight into the structure of the graph as well as give network control or increase network robustness. We give an Integer Linear Program which finds Open Locating Dominating sets on finite graphs. Finally, we prove that the Open Locating Dominating density of the Infinite Triangular Grid is 4/13.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

We often use graphs as a theoretical tool to represent actual networks. These could be physical networks such as routers or factories, or they could be more conceptual, like friendship networks or publication networks. On any of these networks, it may be useful to have some concept of security and the ability to detect failures at the nodes of the network. These failures will be defined in different ways for each kind of network but regardless of the precise network application, the ability to detect failures is crucial for the network to be able to function effectively. Open Locating Dominating sets provide us with this ability. If we represent any of these networks as a graph and find an Open Locating Dominating set on that graph, we will have the ability to detect a failure at any of the nodes. If we endow the nodes that belong in the Open Locating Dominating set with the ability to detect failures at any of it's neighbors, whether it be with physical software or some more cerebral method, we will be able to detect failures on the entire network. Whether or not the specific node that has failed can be identified relies on the specific type of network and the kind of detection mechanism used on each node. We will explain this in detail for the example network types given above.

First, consider a network of routers, passing information amongst themselves. If one of these routers ceases to function properly, the network will at best pass information more slowly and at worst stop passing information completely. As such, the ability to detect failures at any of the routers is critical. Suppose we can install software that allows each router to detect a failure at each of its neighbors and, furthermore, return the specifics of which neighbor(s) failed. If we represent this network of routers as a graph where the vertices represent the routers and there is an edge between two routers if they are connected, then finding an Open Locating Dominating set on this graph would specify the routers on which the software should be installed. Not only would any failure of any router be detected, but furthermore, a failed router could be pinpointed and fixed in a timely manner.

Next, consider a group of factories, or any group of related buildings, which require security, perhaps all owned by the same company. Suppose that these factories have the ability to communicate with one another and furthermore, it's possible to install security systems which have the ability to detect break-ins at the factories with which they communicate. Then, if we model this network of factories as a graph where the vertices are the factories and there is an edge between two factories if they are able to communicate with one another, then finding an Open Locating Dominating set on this graph, will provide us with exactly the buildings in which we should install the security system. Since such security is cost prohibitive, finding the smallest Open Locating Dominating set would minimize the cost while still providing security for every factory.

Finally, consider the more conceptual example of a friendship network. To build the graph representation of a friendship network, let each vertex represent a person and let there be an edge between two people if they are friends. While these networks may have begun purely as a thought exercise to help conceptualize graphs, they have become a reality, and a programming problem to tackle with the advent of social media like Facebook. What might security or robustness look like on such a network? Let us

consider the example of a false or malicious rumor spreading through a school. Let us consider the belief of that rumor as a *failure*, and assume that a person can detect this failure in any of their friends. Then by finding an Open Locating Dominating set on the friendship network made up of all the students at this school, we identify the students who believe the rumor and the fallacy of the rumor can be explained to them.

Clearly, Open Locating Dominating sets can be used to increase network robustness or security. Furthermore, it may be of use to find the smallest possible Open Locating Dominating set to minimize effort or cost. As such, most of this research is centered around finding the smallest possible Open Locating Dominating set on a variety of graphical structures.

We also seek to find the size of the smallest Open Locating Dominating sets, both on finite graphs with a specific structure, as well as on a particular class of infinite graphs. There is a significant difference between the structure of Open Locating Dominating sets on finite graphs and those on infinite graphs. These differences largely stem from the boundaries of the finite graphs and how they are dealt with in constructing the Open Locating Dominating set.

1.2 Introduction

1.2.1 Graphs

Formally, a graph is a pair of sets. The set, V, is a set of unique items called *vertices* and a set, E, containing pairs of vertices called *edges*. Often we represent a graph pictorially rather than with these two sets. The vertex for the graph below is $V = \{1, 2, 3, 4\}$ and the edge set is $E = \{\{1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$.

Note that neither the two vertices that make up a pair, nor the pair itself, need to be unique. That is, we can allow be more than one edge between the same two vertices as well as an edge from a single vertex to itself. When we allow these things, we have what

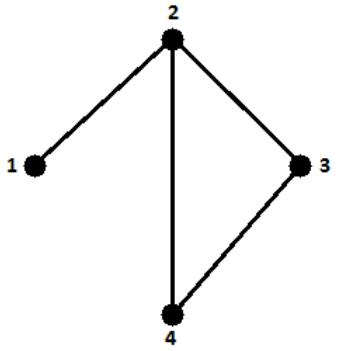

Figure 1.1: Graph

is called a *multi-graph*. When we restrict the edge set to unique and distinct pairs, we have a *simple graph*. We restrict our study in this paper to simple graphs.

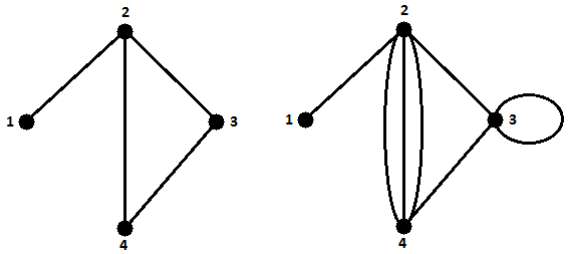

Figure 1.2: A Simple Graph And A Multigraph

Any vertex that shares an edge with a vertex, v, is called a *neighbor* of v. The *degree* of a vertex is the number of neighbors that v has. In the multigraph above, for example, the degree of vertex 2 is 5. If every vertex in a graph has the same degree, that graph is called *regular*. The *open neighborhood* of a vertex v is denoted $N(v)$ and is a set containing all of the neighbors of v. The *closed neighborhood* of v is denoted $N[v]$ and is a set containing all of the neighbors of v as well as v itself.

A *path* between two vertices, v and w, is a list of distinct vertices, x_i, $(v, x_1, x_2, ..., x_m, w)$ such that each x_i and x_{i+1} share an edge. The length of such a path is $m+1$, the number

of edges traversed. We define the distance between two vertices v and w, $d(v, w)$, as the length of the shortest path between v and w. Note that the distance between v and itself is 0.

Note, also, that neither the vertex nor the edge set must be finite. Consider the infinite triangular grid, with which we work extensively. The infinite triangular grid has an infinite vertex set and each vertex has a degree of 6, so it is regular. As you can see below, each face of the grid is a triangle, hence the name. Since each vertex has a non-zero degree, the edge set is clearly also infinite.

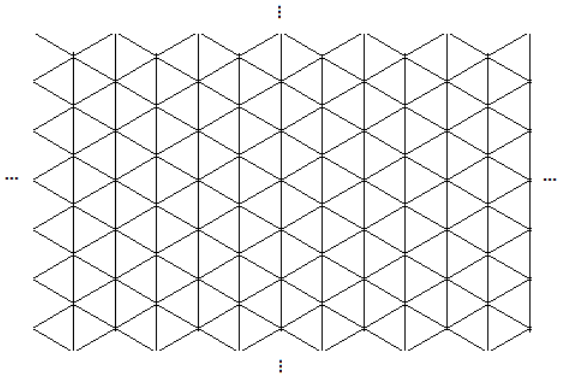

Figure 1.3: The Infinite Triangular Grid

1.2.2 Open Locating Dominating Sets

This paper is focused on finding Open Locating Dominating Sets on both infinite and finite graphs. An *Open Locating Dominating Set*, hereafter referred to as *OLD sets* or *OLDs*, is defined as a subset, S, of vertices that satisfy the following conditions:

- For every vertex, v, $S \cap N(v) \neq \emptyset$.
- For any vertices v and w such that $v \neq w$, $(N(v) \cap S) \neq (N(w) \cap S)$.

So for every vertex, v, in the graph, v has some vertex adjacent to it in the OLD and no two vertices have the same adjacent vertices in the OLD. The idea behind OLD sets is that every vertex has some vertex watching it and there are no two vertices being watched by exactly the same vertices. We can easily see how this structure would be useful for the security problems presented in section 1.1.

As we saw in the example of factory security, it can be desirable to design an OLD with the smallest number of vertices. For any graph, G, the *Open Locating Dominating Number*, denoted $OLD(G)$, is the size of the smallest possible OLD on that graph. Below we have have an example of a graph with an OLD Number of 3. The red are an OLD set.

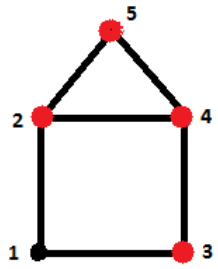

Figure 1.4: The Smallest OLD Set

Extending this measure of the Open Locating Dominating Number, we define the *Open Locating Dominating Density*, denoted $OLD\%(G)$, as

$$OLD\%(G) = \min_{v \in V(G)} \{ \lim_{k \to \infty} \frac{|N_k[v] \cap S|}{|N_k\{v\}|} | S \ is \ an \ OLD \ for \ G \}$$

Where $N_k[v] = \{w \in V(G) : d(v, w) \leq k\}$. That is $N_k[v]$ is the set of all vertices in G that are within k distance of v. We can think of this as the percent of vertices of a graph which are included in the OLD set.

1.2.3 The Discharging Method

A method which is often used in graph theory proofs, the discharging method, lends itself well to Open Locating Dominating Sets. The discharging method works as follows:

first, we assign an initial charge to each vertex in the graph, then the vertices give or receive charge amongst themselves according to predetermined rules, and after this, the vertices have a final amount of charge. We use this final charge distribution to gain insight into some aspect of the graph, in our case, the size of the OLD set for that graph. Below is a small example, displaying the discharging process. We initially assign all of the red nodes a charge of 1 and the black nodes a charge of 0. The discharging rule is: all red nodes give 1/6 of their charge to any adjacent black nodes. After the discharging, every node has a charge of 1/6.

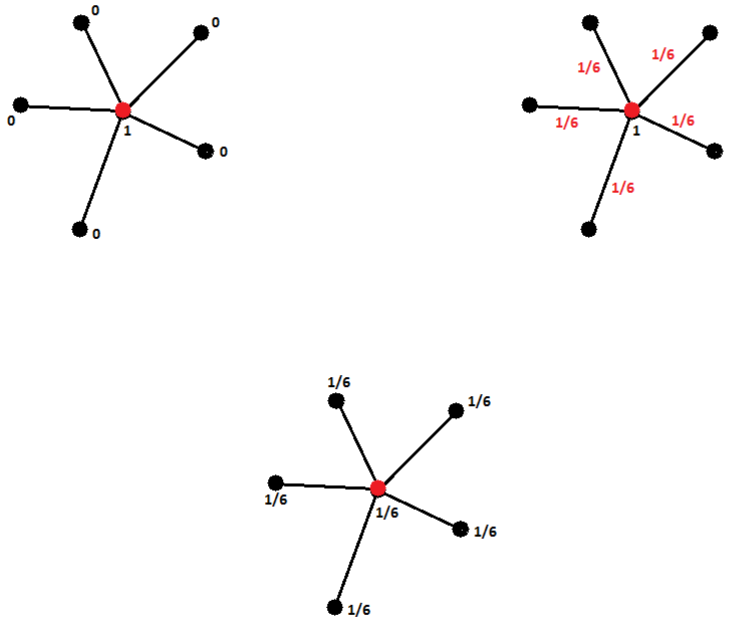

Figure 1.5: Discharging

The idea behind discharging is that, if we pick our initial charges well, the final charges should give us an idea of the number of some specific type of vertex on the graph. It is especially useful, when the structure you are measuring involves some concept of *taking*

care of all of the vertices, as is the case for OLDs. So for our uses, we assign initial charges of one to any vertex in the OLD set and 0 to any vertex not in the OLD set. After we allow the vertices to discharge, according to rules that coincide with the structure of OLDs, then the final charge should tell us the OLD percentage, or at least a lower bound for the OLD percentage. We describe this method in greater detail in section 3.1.

Chapter 2

Integer Linear Program

We formulated an Integer Linear program to find the smallest possible OLD on a given finite graph. First, we represent the graph, G, using its adjacency matrix, $A(G)$. Let $n = |V(G)|$. Then $A(G)$ is an $n \times n$ matrix with elements $a_{i,j}$ such that $a_{i,j} = 1$ if there is an edge between vertices i and j and $a_{i,j} = 0$ if there is not an edge between vertices i and j.

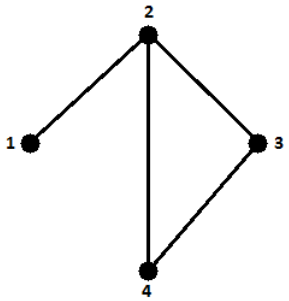

Figure 2.1: A Simple Graph

So for our example simple graph from above, the adjacency matrix would be:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Let $x = [x_1 \dots x_n]$ be a vector of size n which indicates which vertices of G are in the OLD; $x[i] = 1$ if vertex i is in the OLD and 0 otherwise. So to find the OLD on G which contains the fewest number of vertices, we need to minimize the number of 1's in v, so we want to minimize the sum of the elements of x, $\sum_{i=1}^{n} x[i]$.

We require constraints that ensure that the vertices we select to put in the OLD satisfy the conditions of an OLD. First, we must ensure that every vertex of G has a neighbor in the OLD. Let A_i denote the ith column of the adjacency matrix and that each of these columns describes $N(i)$ for every vertex, i. If we multiply the adjacency matrix A by the OLD vector, x, we obtain an $n \times 1$ matrix, that is, a vector of size n. Since each 1 in the jth row of A represents a neighbor of vertex j, when one of these 1's is multiplied by a 1 in x, it means that that neighbor is in the OLD. When we sum up all of these 1×1 multiplications, we have the number of neighbors of j in the OLD.

Now, we must ensure that no two vertices have the same neighbors in the OLD. We do this using Hadamard multiplication, or element-wise multiplication of matrices, denoted by $\circ$. Since A_i and x are both $n \times 1$, we may take the Hadamard product of A_i and x to obtain another $n \times 1$ binary vector. This vector, $A_i \circ x$ describes exactly which of vertex i's neighbors are in the OLD. If $(A_i \circ x)[j] = 1$, then that means both $A_i[j]$ and $x[j]$ are 1, thereby indicating that vertex j is both in the OLD and a neighbor of vertex i. So if we compute $A_i \circ x$ for each vertex in G, and compare $(A_i \circ x)$ and $(A_j \circ x)$ for every pair of vertices i and j, we require that these two vectors not be equal to ensure that no two vertices i and j share the same neighbors in the OLD. This is equivalent to requiring that the difference between the two vectors is not the 0-vector, that is, that none of the 1's in

$A_i \circ x$ or $A_j \circ x$ match up. Therefore, we need $(A_i \circ x) - (A_j \circ x) \neq \vec{0}$.

As a result, we have the following integer linear program

$$\min \sum_{k=1}^{n} x_k$$

subject to

$$Ax \geq \vec{1}$$

$$(A^i \circ x) - (A^j \circ x) \neq 0 \ \forall \ i, j$$

$$x \in \{0,1\}^n$$

Below is the AMPL code for this integer linear program, which will provide a solution as long as the adjacency matrix of the graph is not prohibitively large.

```
"OLD.mod"

set NODES;     # number of nodes in simple undirected graph
set ROWPAIRS;  # number of pairs of nodes two edges apart

param Adj {NODES, NODES} >= 0;   # adjacency matrix
param A2 {ROWPAIRS, NODES} >= 0; # absolute value of difference of Adj rows, 2 edges apart

var x {NODES} >= 0;

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 2covering {k in ROWPAIRS}:  # every node 2 edges away must be covered #

  sum {j in NODES} A2[k,j] * x[j] = 1;
```

Chapter 3

OLD Number and Density Results

3.1 The OLD Density of the Infinite Triangular Grid

We now turn to the task of finding the OLD density of the Infinite Triangular grid, denoted Tr. Since this is a graph that consists of an infinite number of vertices, the number of vertices in the OLD set is necessarily also infinite, otherwise, we would not satisfy the first condition to form an OLD set: that every vertex have a neighbor in the OLD set. So we want to find the proportion of vertices of Tr which are included in the OLD which uses the smallest proportion of vertices possible.

3.1.1 Previous Results

Previously, it has only been possible to give bounds on the OLD density of Tr. First, [2] gives a lower bound for any graph which is regular: if G is countable infinite and regular of degree r, then $OLD\%(G) \geq \frac{2}{1+r}$. As we stated above, Tr is regular of degree 6, so from this given lower bound we have $OLD\%(Tr) \geq \frac{2}{7}$.

Seo and Slater give a construction with a density of $\frac{1}{3}$ in [2].

The outlined pattern repeats over the whole grid. Since for every 3 vertices in the outlined triangle pattern, 1 is included in the OLD, we have a density of $\frac{1}{3}$. Therefore,

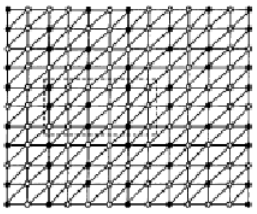

Figure 3.1: An OLD Set With Density 1/3

we are able to say that

$$\frac{1}{3} \geq OLD\%(G) \geq \frac{2}{7}$$

A tighter upper bound is given in [1] by Honkala.

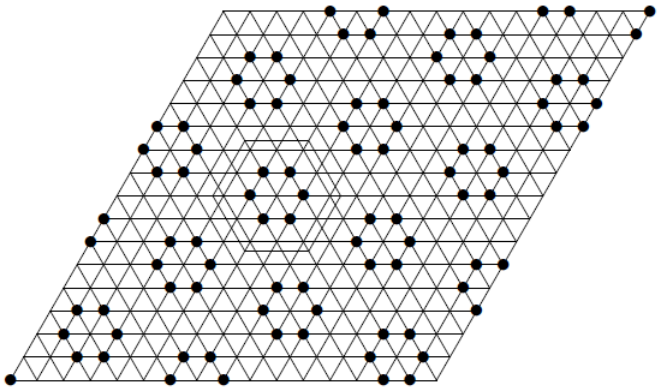

Figure 3.2: A Better Bound On The OLD Density

In the outlined hexagon, there are 19 vertices and 6 of them are included in the OLD set. So now we have an upper bound of $\frac{6}{19} < \frac{1}{3}$.

3.1.2 Upper Bound

We present a construction which gives an upper bound on the OLD density of the infinite triangular grid.

Lemma. For the infinite triangular grid, Tr, we have $OLD\%(Tr) \leq \frac{4}{13}$.

Proof. Consider the following construction:

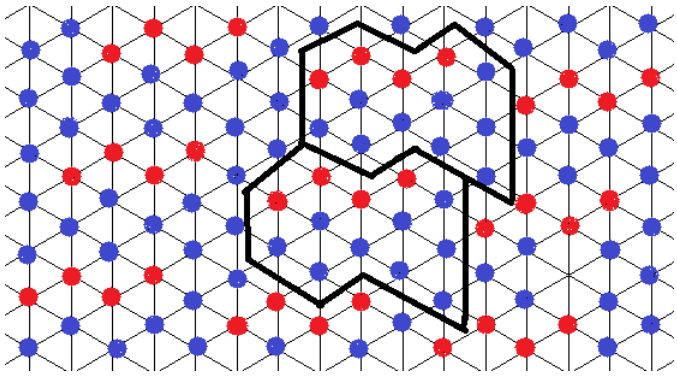

Figure 3.3: An OLD Set with Density 4/13

In the outlined area, there are 13 vertices and 4 of them, the red ones, are included in the OLD set. We need to make sure that this structure is, indeed, an OLD set. First, consider the vertices included in the potential OLD. Every vertex has at least one neighbor in the OLD; the two red vertices on the end of the red path have one neighbor each and they are not the same neighbor and the two other red vertices have two neighbors each in the OLD and again these are not the same two therefore both conditions for an OLD set are satisfied for the vertices in the potential OLD.

Consider the following graphic where vertices which are not included in our potential OLD set are color-coded by the number of neighbors they have in the OLD. For any vertex, v, let us denote this number as $deg_O(v)$. Every green node, g, has $deg_O(g) = 2$ and the two neighbors are in different 4-clusters in the potential OLD. Every orange node, o, has $deg_O(o) = 2$ and the two neighbors form a 2-cluster. Every purple node, p, has $deg_O(p) = 3$ and all three neighbors are a cluster. Finally, every pink node, f, has $deg_O(f) = 1$. Since all the nodes of Tr fall into one of these categories, we can conclude that every vertex in Tr has a neighbor in the OLD since for each v not in the potential OLD, $deg_O(v) > 0$. Since every color category has a different number of neighbors in the

OLD except for the orange and green nodes which both have two, we need only look at these two sets of vertices to ensure that no vertices have the same set of neighbors in the OLD. However, since the two neighbors of the green vertices are in different red 4-clusters, they cannot be the same as the 2-cluster-neighbors of the orange vertices. Therefore, no two vertices which are not included in the potential OLD have the same neighbors in the potential OLD.

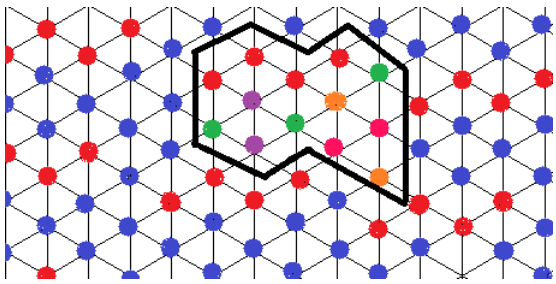

Figure 3.4: An OLD Set with Density 4/13 with Vertices Color-Coded

Finally, we must make sure that for v in the potential OLD and u not in the potential OLD, that v and u do not share the same neighbors in the potential OLD. We must only check vertices which have the same number of neighbors in the potential OLD. First, we consider the two outer red vertices and the pink vertices. Since for each pink node, the one neighbor which is included in the potential OLD is an outer red node, they cannot share the same neighbors in the potential OLD, since a vertex cannot be its own neighbor in a simple graph. Now, consider the two inner red vertices, and the green or orange vertices. Since the inner red vertices must necessarily have neighbors in the 4-cluster and the green vertices' neighbors are in two different 4-clusters, these two groups cannot share the same neighbors in the potential OLD. Each orange vertex is adjacent to one inner red vertex. Again, because a vertex cannot be its own neighbor, the inner red vertex to which the orange vertex is adjacent cannot share the same neighbors in the potential OLD as the orange vertex. The other inner red vertex, to which the orange vertex is not adjacent, has, as one of it's neighbors, an outer red vertex, which is not adjacent to

the orange vertex. Therefore, these too do not share the same neighbors in the potential OLD. Therefore we can conclude that no two vertices of Tr share the same neighbors in the potential OLD. Hence, the conditions of an OLD set are met and our potential OLD is officially an actual OLD.

Since we have a construction which is an OLD set which gives an OLD density of $\frac{4}{13}$, we know that we do not need a density higher than this and $\frac{4}{13}$ is therefore an upper bound on the OLD density of the infinite triangular grid. □

3.1.3 Lower Bound and Equality

We now show that not only is the OLD density of the infinite triangular grid bounded from above by $\frac{4}{13}$, but it is also bounded from below by $\frac{4}{13}$ and it is therefore exactly $\frac{4}{13}$.

Theorem. For the infinite triangular grid, Tr, $OLD\%(Tr) = \frac{4}{13}$.

Proof. We proceed by showing that $\frac{4}{13}$ is a lower bound as well as an upper bound. Suppose that we have an OLD set on Tr; let us call it S. Further, suppose that this OLD set achieves a density on Tr which is equal to Tr's OLD density. We need to show that this OLD has a density of at least $\frac{4}{13}$ in Tr.

Note that nowhere in this OLD is there an isolated vertex, that is, there is no vertex included in the OLD for which all six of its neighbors are not in the OLD. This is because that vertex would have no neighbor in the OLD and therefore the first condition for an OLD would be broken.

A *cluster* is defined as any subset of vertices such that any two vertices in the cluster are connected by a path. This is the same as a connected component An $n-$*cluster* is a cluster consisting of exactly n vertices. We sometimes wish to count the number of neighbors a vertex has which are in a cluster, C. We denote this $deg_C(v)$.

Therefore, every vertex included in the OLD must be part of a n-cluster with $n \geq 2$. So now we show that for every possible cluster size, $OLD\%(Tr) = \frac{4}{13}$. We do this using the discharging method. We assign an initial charge of 1 to any vertices in S and 0 to

any vertices not in S. Our goal is to show every vertex has a charge of at least $\frac{4}{13}$. Our discharging rules are as follows, where $\alpha = \frac{4}{13}$:

1. Every vertex gets $\frac{\alpha}{k}$ charge from each of the k clusters to which it is adjacent.

2. A vertex which gains a charge of β from a cluster and is adjacent to l vertices in that cluster gains $\frac{\beta}{l}$ from each of those l vertices.

Now we show that every vertex will have a charge of at least $\frac{4}{13}$. We break this down into two cases: clusters of size less than five, and clusters of size greater than or equal to five.

Case 1: Let C be an n-cluster with $n < 5$. We consider three possible cluster sizes: $n = 2$, $n = 3$, and $n = 4$. This will result in all the vertices not in the cluster having a final charge of $k\frac{\alpha}{k} = \alpha$. If we count up the charge on the vertices in the cluster after the discharging, we determine what values of α ensure that there is a non-zero amount of charge left on these vertices in the cluster. If we let $\alpha = \frac{4}{13}$ and there is a non-negative charge left on the vertices in the cluster, then we have a density of no less than $\frac{4}{13}$.

Note that all we know about the structure of S is that the n vertices in the cluster are in S and any neighbors of these vertices which aren't in the cluster are necessarily not in S, or else the cluster would be bigger than n. Our assumptions don't tell us anything about vertices which are farther than a distance of 1 away from any vertices in the cluster. However, we can use the conditions for an OLD to reason about these vertices for the specific cluster structures.

Case 1a: Let $n = 2$, so we have two adjacent vertices in S which are not adjacent to any other vertices in S.

Consider the two sets of green vertices. For each of these sets of three green vertices, there is a red vertex which is not adjacent to them. This red vertex and the three green vertices share a single vertex in the cluster, namely the other red vertex in the cluster. Since all four of these vertices share the same set of vertices in the OLD, it must be

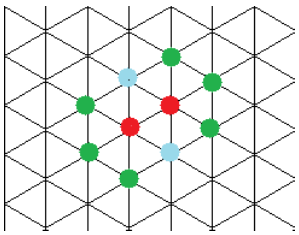

Figure 3.5: Case 1a

the case that three of these four must be adjacent to other clusters in the OLD. Since it would violate the definition of a cluster for it to be the red vertex, it must be the three green vertices that are adjacent to some other cluster. So we know that each of the green vertices are adjacent to at least two clusters. Next consider the two light blue vertices, which are both adjacent to the whole 2-cluster. Again, these two vertices share the same neighbors in the OLD, so it must be the case that one of them is adjacent to at least one other cluster. So of the 8 vertices that surround the 2-Cluster, 7 of them must be adjacent to at least one other cluster. So these 7 are adjacent to 2 clusters and they gain $\frac{\alpha}{2}$ from the 2-Cluster. The last of the 8 vertices may only be adjacent to this 2-Cluster, so it gains α from the 2-Cluster. Since each vertex in the cluster, is by definition, adjacent to only one cluster, each of those vertices gain α from the 2-Cluster. So let us count the total charge on the 2-Cluster after the discharging occurs.

$$2 - 2(\alpha) - 7(\frac{\alpha}{2}) - \alpha = 2 - \frac{13}{2}\alpha$$

Let $\alpha = \frac{4}{13}$. Then this becomes

$$2 - \frac{13}{2}\frac{4}{13} = 2 - 2 = 0$$

Since this is non-negative, we can conclude that for all 2-Clusters, we have a density of at least $\frac{4}{13}$.

Case 1b: Let $n = 3$. The only way for this to occur is if we have three adjacent

vertices in a triangle. We cannot have three in a path because then the two outer vertices would share the same neighborhood in the OLD, specifically the middle vertex.

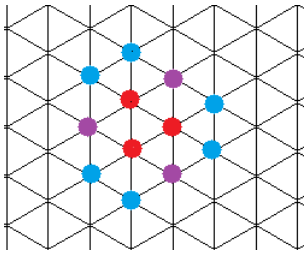

Figure 3.6: Case 1b

Consider one of the vertices in the triangle. It has two neighbors in the OLD, specifically the other two vertices in the cluster. Note that there is another vertex, one of the purple vertices, which has this same set of neighbors in the OLD, the vertex across from our original vertex in the triangle. Since these two share the same neighbors in the cluster, it must be that one of these two vertices must be adjacent to another cluster. Since it can't be the one in the cluster, it must be the purple vertex. Furthermore, there are two vertices, the blue vertices, adjacent to our original vertex which only have it as a neighbor in the cluster. Since these two vertices have the same neighbor in the OLD, it must be true that at least one of them is adjacent to another cluster. Since this is true for all three of the vertices in the cluster, we have a final count of 6 vertices that are adjacent to 2 total cluster and 3 vertices which are only adjacent to this cluster. So if we count the total charge still on the 3-Cluster after discharging, we have

$$3 - 3\alpha - 3\alpha - 6\frac{\alpha}{2} = 3 - 9\alpha$$

Let $\alpha = \frac{4}{13}$

$$3 - 9\frac{4}{13} = \frac{3}{13}$$

Since we have a non-negative amount of charge left on the cluster after the discharging, we know that in the case of 3-clusters, we have a density of at least $\frac{4}{13}$.

Case 1c: Let $n = 4$. Unlike the previous cases there are several possible structures for 4-Clusters. The following two structures do not satisfy the definition of an OLD, so we do not need to ensure that they will have enough charge after discharging:

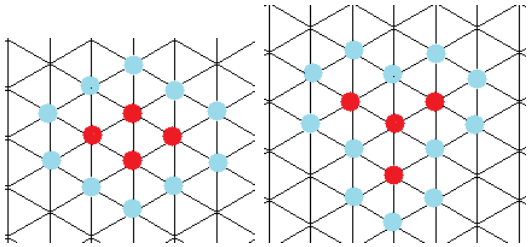

Figure 3.7: 4-Cluster Structures Which Are Not Permitted

In both cases, these 4-Clusters could not be in an OLD because the outer vertices share the inner vertex(ices) as their only neighbors in the OLD. So we need only consider the following cases of 4-Cluster:

Case 1c(i): Consider the following 4-cluster structure:

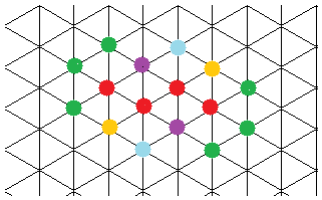

Figure 3.8: Case 1c(i)

Consider either of the end vertices in the cluster. All of the adjacent green vertices have only that vertex as their neighbor in the OLD, so at least two of them must be adjacent to another cluster. We cannot determine anything about either the yellow or purple vertices, so we must assume they are adjacent only to this cluster. However, each of the blue nodes are adjacent only to one of the inner red vertices in the cluster. The two

outer red vertices are also only adjacent to these same inner vertices, so we must conclude that these blue vertices must each be adjacent to at least one other cluster. So now we know that there are 6 vertices which we must assume are only adjacent to this cluster, and 6 vertices which are adjacent to 2 total clusters. Our count for the amount of charge on this cluster after discharging is

$$4 - 4\alpha - 6\alpha - 6\frac{\alpha}{2} = 4 - 13\alpha$$

If we let $\alpha = \frac{4}{13}$,

$$4 - 13\frac{4}{13} = 0$$

Since we have a non-negative charge left on the cluster after discharging, we conclude that this 4-Cluster structure has a density of at least $\frac{4}{13}$.

Case 1c(ii): Consider the following 4-cluster structure:

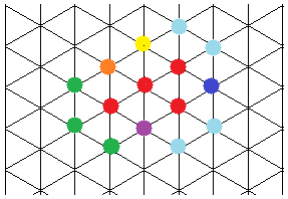

Figure 3.9: Case 1c(ii)

The three green vertices all share the outer red vertex in the cluster as their only neighbor in the OLD, so at least two of them must be adjacent to another cluster. We cannot say anything about the purple, orange, or dark blue vertices, so we must assume that they are only adjacent to this cluster. The yellow vertex and the red vertex which isn't in line with the rest of the cluster both share the same two vertices in the cluster, so the yellow vertex must be adjacent to at least one other cluster. The two sets of light blue vertices each share one of the red vertices of the cluster as their only neighbor in the

OLD so one vertex in each set of light blue must be adjacent to at least one other cluster. From this, let us count the charge on the cluster after discharging:

$$4 - 4\alpha - 6\alpha - 5\frac{\alpha}{2} = 4 - \frac{25\alpha}{2}$$

Let $\alpha = \frac{4}{13}$.

$$4 - \frac{25\alpha}{2} = \frac{2}{13}$$

Again, this non-negative charge left on the cluster tells us that for this cluster structure we have a density of at least $\frac{4}{13}$.

Case 1c(iii): Consider the following 4-cluster structure.

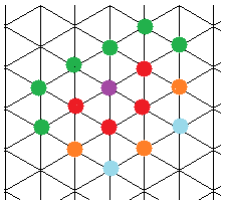

Figure 3.10: Case 1c(iii)

For each red vertex at the end of the cluster, there are three green vertices which have that red vertex as their only neighbor in the OLD, so at least two of the three green vertices for each of these red end-vertices must be adjacent to another cluster. Furthermore, each of these end vertices have only their adjacent red inner-vertex as their only neighbor in the OLD, and this is also true of each light blue vertex, so we know that each light blue vertex is also adjacent to another cluster. We cannot say anything about either the purple vertex or the orange vertices, so we must assume that they are only adjacent to this cluster. From this, we count the amount of charge left on the cluster after discharging:

$$4 - 4\alpha - 6\alpha - 6\frac{\alpha}{2} = 4 - 13\alpha$$

Letting $\alpha = \frac{4}{13}$,

$$4 - 13\alpha = 4 - 13\frac{4}{13} = 0$$

Since this is non-negative, we conclude that for this cluster structure we have a density of at least $\frac{4}{13}$.

Case 1c(iv): Consider the the following 4-cluster structure.

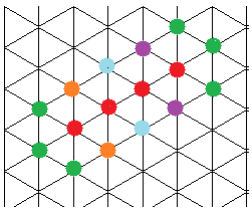

Figure 3.11: Case 1c(iv)

For each red vertex on the end of the path, there are three green vertices which only have that red vertex as their only neighbor in the OLD, so at least two of three of the green vertices on each side must be adjacent to some other cluster. Then each pair of the purple, blue, and orange vertices share the same two red vertices in the path as their only neighbors in the OLD, so in each pair there must be one vertex which is adjacent to another cluster. So we have on the cluster after discharging,

$$4 - 4\alpha - 5\alpha - 7\frac{\alpha}{2} = 4 - \frac{25\alpha}{2}$$

Letting $\alpha = \frac{4}{13}$,

$$4 - \frac{25\alpha}{2} = \frac{2}{13}$$

Since this is non-negative, we know that this cluster structure has a density of at least $\frac{4}{13}$. So we know that for every 4-Cluster structure possible in an OLD, we have a density of at least $\frac{4}{13}$. We have shown that for any $n < 5$, any n-cluster in an OLD will have a

density of at least $\frac{4}{13}$ on the infinite triangular grid. Next, we show that this is true for $n \geq 5$.

Case 2: Consider a k-cluster, C, with $k \geq 5$. This cluster starts out with a total charge of k since our initial charge set up assigns a charge of 1 to each vertex included in the OLD. Then after discharging, this cluster must retain $k\alpha$ for itself. So the amount of charge left on the cluster to take care of adjacent vertices is $k - k\alpha = k(1-\alpha)$. For each of the k vertices there is $\frac{k(1-\alpha)}{k} = 1 - \alpha$ available to cover any adjacent vertices not in the cluster. If we let $\alpha = \frac{4}{13}$ then this is $1 - \frac{4}{13} = \frac{9}{13}$. So each vertex in the cluster has $\frac{9}{13}$ charge to give out.

Note that in the previous case, when a vertex outside of the OLD was forced to have some other neighbor in the OLD we could assume that this vertex was in a different cluster because we specified the size of the cluster. But with general clusters of size $k \geq 5$, we cannot assume this. In fact we must assume the worst case, which is that these forced neighbors are in fact in the same cluster as the original neighbor.

Case 2a(i): Suppose that $deg_c(v) > 1$ and it is not adjacent to any vertices, u, such that $deg_C(u) = 1$, where u, $v \in C$. In the figure below, one such v is circled in purple.

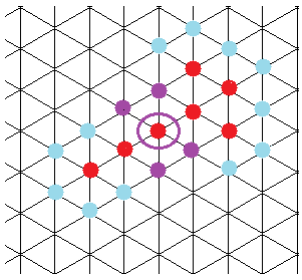

Figure 3.12: Case 2a(i)

For each such vertex, v, there are four neighbors, shown in purple, which are not in the cluster that this vertex must cover with its charge. There are two pairs of vertices which share the same neighborhood in the OLD. Therefore, one purple vertex in each pair must be adjacent to some other vertex of the OLD; let us assume it is also in this cluster.

Then each of these two purple vertices are adjacent to three vertices in this cluster and therefore v must contribute $\frac{\alpha}{3}$ to each of these two vertices. For the other two purple vertices, they may only be adjacent to two vertices in the cluster and v must therefore contribute $\frac{\alpha}{2}$ to each of these two. For all four neighbors, the total charge that must be given out is at most $2\frac{\alpha}{3} + 2\frac{\alpha}{2} = \frac{5\alpha}{3}$. If we let $\alpha = \frac{4}{13}$, then this is $\frac{20}{39}$. Since this is less than $\frac{9}{13}$, we know that we don't need to worry about this kind of vertex in the cluster.

Case 2a(ii): Consider another such v, circled in purple below:

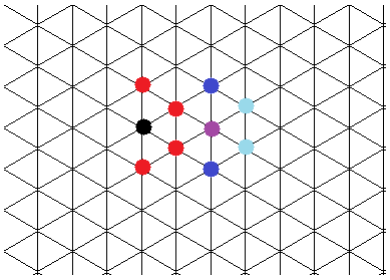

Figure 3.13: Case 2a(ii)

The red vertices are included in the OLD set and the blue ones are not. The vertex, v, must therefore provide enough charge for all four blue vertices. The two dark blue vertices are adjacent to the two shown red vertices and some other vertex in the OLD. Let us assume that this other vertex is part of this cluster. Therefore, each of these dark blue vertices require $\frac{\alpha}{3}$ from v. The light blue vertices both share the neighborhood of v and one of them must, therefore, be adjacent to some other vertex in the OLD, and let us again assume it is part of this cluster. From this we obtain a total charge given out by v of

$$2\frac{\alpha}{3} + \frac{\alpha}{2} + \alpha = \frac{13\alpha}{6}$$

If we let $\alpha = \frac{4}{13}$ then we have a total charge given out of $\frac{2}{3} < \frac{9}{13}$. So this kind of vertex is also not a problem.

Case 2c: Now suppose that the vertex, v, is adjacent to exactly one vertex, u, such that $deg_C(u) = 1$. In the figure below such a v is circled in purple and u is circled in

green.

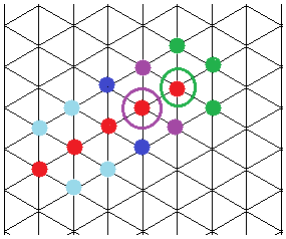

Figure 3.14: Case 2b

Let us count the charge that these two vertices must give out. The purple circle vertex, v, must cover half of the charge given to the two dark blue vertices because they are shared by the next red vertex. These two blue vertices share the same two red vertices in the OLD so at least one must be adjacent to another vertex in the OLD. Together v and u must cover the two purple vertices, which also share the same two vertices in the OLD, so at least one must be adjacent to some other vertex in the OLD. Finally, the 3 green vertices all share the end red vertex as a neighbor in the OLD, so at least two of them must be adjacent to another vertex in the OLD. In order to take care of the worst case, let us assume any other forced neighbor in the OLD are in the same cluster. So together these two vertices must give out

$$\begin{gathered}(\frac{\alpha}{2}+\frac{\alpha}{2}+\frac{\alpha}{3}+\frac{\alpha}{3}) \\ (\frac{\alpha}{2}+\frac{\alpha}{2}+\alpha+\frac{\alpha}{2}+\frac{\alpha}{3}) \\ =4\alpha+\frac{\alpha}{2}=\frac{9\alpha}{2}\end{gathered}$$

If we let $\alpha = \frac{4}{13}$, then this becomes $\frac{18}{13}$. Since this is not greater than $2\frac{9}{13} = \frac{18}{13}$, we know that for this type of vertex we have a density of at least $\frac{4}{13}$.

Note that we do not consider the case of a vertex, v, in the cluster with $deg_c(v) > 1$ and two neighbors u and w such that $deg_c(u) = deg_c(w) = 1$. This is because this case cannot happen for the following reasons.

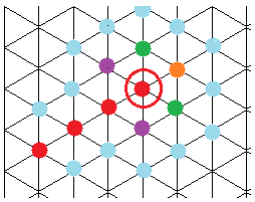

Figure 3.15: A Cluster Structure That Is Not Permitted

First, such a vertex v would have to be at the end of a path of vertices in the cluster or it would not be possible for v to have neighbors with $deg_c(u) = 1$ because they would be forced to be adjacent to other vertices in the cluster. So let v be the circled red vertex in the picture. The two neighbors cannot include either of the two purple vertices because they are adjacent to other vertices in the cluster, making their degree with respect to the cluster greater than 1. This implies that the two neighbors must be some combination of the green vertices and the orange vertex. It cannot be a green vertex or an orange vertex because then $deg_c(orange) = deg_c(green) = 2$. So we are left only with the option of the two green vertices. However, if this is the case, then these two green vertices only have the vertex v as a neighbor in the OLD and, therefore, one of them must be adjacent to some other vertex in a the OLD. Note however, that these green vertices are *in* the cluster so if one of them is adjacent to another vertex in the OLD, then that extends the cluster and thereby makes the degree of this green vertex with respect to the cluster greater than 1. Therefore we cannot have this construction.

Since we have a density of at least $\frac{4}{13}$ for every cluster size, we know that the density of an OLD on the infinite triangular grid must be at least $\frac{4}{13}$.

Since we know that it must be at least $\frac{4}{13}$ and we showed with our example that it need not be greater than $\frac{4}{13}$, we know that the density of the smallest possible OLD on the infinite triangular grid is exactly $\frac{4}{13}$.

□

3.2 Conclusion

Open Locating Dominating sets can be a useful tool in real life network models. We provided an Integer Linear Program to find OLDs on finite graphs. We also showed that the density of the smallest possible OLD on the infinite triangular grid is $\frac{4}{13}$ by first giving a construction which showed that $\frac{4}{13}$ is an upper bound. Then we used the discharging methods and broke down a potential OLD into different cluster sizes to show that $\frac{4}{13}$ is a lower bound.

3.3 Acknowledgements

I would like to thank my advisers, Professor Gexin Yu and Professor Rex Kincaid. I would also like to thank the CSUMS research program for providing me with funding to begin work on this research in the summer of 2012.

Bibliography

[1] Honkala, Iiro., *An Optimal Strongly Identifying Code in the Infinite Triangular Grid*, University of Turku., Turku, Finland. 2009.

[2] Seo, Suk J. and Slater, Peter J., *Open Neighborhood Locating-Dominating Sets*, Australian Journal of Combinatorics. 2010.

[3] West, Douglas B., *Introduction to Graph Theory*, Prentice Hall, Inc., 1996.