Highlights

A Generalization of Distance Domination

Alicia Muth, E. Dov Neimand

- We generalize the concept of distance domination in graphs.
- We present an algorithm that, under the generalization, finds a minimal failure set for a tree.

A Generalization of Distance Domination

Alicia Muth, E. Dov Neimand

[a]*Departments of Mathematics and Computer Engineering, Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ, United States*

Abstract

Expanding on the graph theoretic ideas of k-component order connectivity and distance-ℓ domination, we present a quadratic-complexity algorithm that finds a tree's minimum failure-set cardinality i.e. the minimum cardinality any subset of the tree's vertices must have so that all clusters of vertices further away than some ℓ do not exceed a cardinality threshold. Applications of solutions to the expanded problems include choosing service center locations so that no large neighborhoods are excluded from service, while reducing the redundancy inherent in distance domination problems.

Keywords: domination, neighbor connectivity, distance-domination, tree graphs

1. Introduction

For standard graph-theoretic notation and terminology, we refer to the book "Graphs and Digraphs" authored by Chartrand, Lesniak, and Zhang [2].

Given an order n graph $G = (V, E)$, Haynes et al. in [5] define a set $S \subseteq V$ as a **distance-ℓ dominating set** if every vertex of V is within distance ℓ of at least one vertex in S.

A **k-component order connectivity** set of a graph G is the set $S \subseteq V$ such that all components of the graph induced by $V \setminus S$ have order less than k.

A **k-component order neighbor connectivity** set of a graph G is the set $S \subseteq V$ such that all components of the graph induced by $V \setminus N_1[S]$ have order less than k.

The **closed-ℓ neighborhood** of a vertex $v \in V$, denoted $N_\ell[v]$, is the set

$$N_\ell[v] = \{u \in V \mid d(u, v) \leq \ell\}.$$

A set F is a **distance-ℓ dominating set** if $N_\ell[F] = V$. The **distance-ℓ domination number**, $\gamma_{\leq \ell}(G)$, is the minimum cardinality of a distance-ℓ dominating set in G.

Applications of distance domination include the optimization of the placement of resources. For example, placement of bus stops to minimize the number of blocks any individual in a given area may have to walk; or the placement of radio stations, which individually can cover a particular radius, to minimize the number of stations while still covering an entire area [1]. Additionally, there are opportunities for this parameter to take on even more properties. For example, it is possible to use weighted vertices to represent the population of a block; or digraphs to represent one-way streets.

As distance domination problems, these placements may lead to significant overlap due to the requirement that all vertices be within ℓ distance of a distance dominating set. Our extension to k-component order neighbor connectivity allows for particular subsets of a region to be omitted from coverage, provided they are 'small enough.' These results have cost-saving benefits as they allow for minimum installations while still providing optimum service for all but some smaller clusters. This parameter could be used to minimize the number of installations, and therefore the budget while keeping in mind the maximum distance needed to travel.

For relevant works preceding ours, in [8], Luttrell's dissertation fully introduces k-component order neighbor connectivity. Doucette et al. expand on Luttrell's work in [3]. Gross et al. [4] give us the k-component order neighbor connectivity definition we use here. They denote the cardinality of the smallest such S with $\kappa_{nc}^{(k)}(G)$. More information regarding the various domination-inspired parameters can be found in Luttrell et al [7].

2. Initial Results

First introduced is the definition of the vulnerability parameter, k, that is the focus of the paper.

Below, except where stated otherwise, we use $G = (V, E)$ to refer to a simple order-n graph.

Definition 1. A set S is a **distance-ℓ, k-component order, neighbor connective failure set** if the subgraph of G induced by $V \setminus N_\ell[S]$ leaves all components with order less than k.

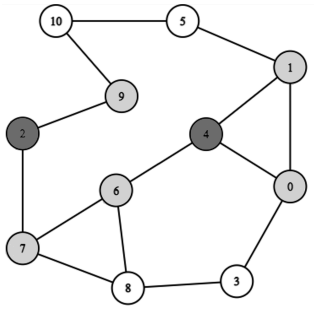
Definition 2. The **distance-ℓ, k-component order connectivity number** of a graph, $\lambda_\ell^{(k)}(G)$, is the minimum cardinality of a distance-ℓ, k-component failure set for G. We require the component threshold, k, always has $1 \leq k \leq n$, and the distance $\ell \geq 0$. Below, we use k as a component threshold and l as a distance.

Where $k = 1$ then the problem becomes one of distance domination. When $\ell = 0$ it becomes a problem of k-component order connectivity.

For distance-ℓ, k-component order neighbor connectivity, a vertex $v \in V$ is said to **fail** itself and each of its ℓ-adjacent neighbors. That is, v fails all $w \in N_\ell[v]$.

For $F \subseteq V$, a component H of $V \setminus N_\ell[F]$ is **failed** if H has order less than k.

Figure 1:
A Minimum Distance-1, 3-Component Failure Set



A graph G is **failed** by a set F if each component of the subgraph of G induced by $V \setminus N_\ell[F]$ has order less than k.

In Figure 1 we offer an example of a minimum failure set with $k = 3$ and $\ell = 1$. The darker nodes are those in the failure set, the lighter nodes are within a distance of 1 from nodes in the failure set, and the white nodes make up components with sizes less than 3.

This parameter was first introduced in the dissertation of A. Muth using the notation $\kappa_{nc,\leq\ell}^{(k)}(G)$ where the subscript nc stands for neighbor connectivity[9].

For simplicity, we refer to a distance-ℓ, k-component order neighbor connectivity failure set as a **failure set** and if there is no failure set with smaller cardinality, a **minimum failure set**.

We consider the relationship between the cardinality of a minimum failure set, F, and the values of k and ℓ. As the order of surviving components increases, the cardinality of the minimum failure set decreases.

Proposition 3. *Let j and k be component thresholds with $j \leq k$. Then*

$$\lambda_\ell^{(k)}(G) \leq \lambda_\ell^{(j)}(G).$$

Similarly, as the distance increases, minimum failure-set cardinality may decrease.

Proposition 4. *Let m and ℓ be distances with $m \leq \ell$. Then,*

$$\lambda_{\ell}^{(k)}(G) \leq \lambda_{m}^{(k)}(G).$$

Notice that the next two propositions follow directly from the definition of the various parameters and the preceding results.

Proposition 5. *The distance domination number, $\gamma_{\leq \ell}(G)$, of a graph is $\lambda_{\ell}^{(1)}(G)$. Propositions 3 and 4 combine for the following:*

$$\lambda_{\ell}^{(k)}(G) \leq \lambda_{\ell}^{(1)}(G) = \gamma_{\leq \ell}(G)$$

.

Proposition 6. *For distance $\ell = 1$, a set F is a distance-1, k-component order neighbor connectivity failure set, if and only if it is a k-component order neighbor connectivity failure set, i.e.,*

$$\lambda_{\ell}^{(k)}(G) \leq \lambda_{1}^{(k)}(G) = \kappa_{nc}^{(k)}(G).$$

If the maximum distance between any two vertices of the graph, the diameter, is less than ℓ then any nonempty failure set will fail the graph.

Proposition 7. *If $0 < \operatorname{diam}(G) \leq \ell$, then $\lambda_{\ell}^{(k)}(G) = 1$.*

Proposition 8. *Let G be disconnected and $G = \bigcup_{i=1}^{N} G_i$. Then the order of a minimum failure set of the graph G is the sum of the orders of the minimum failure sets for each G_i, i.e.,*

$$\lambda_{\ell}^{(k)}(W) = \sum_{i=1}^{N} \lambda_{\ell}^{(k)}(W_i).$$

3. A Minimum Failure Set Algorithm

Finding a minimum failure set for distance domination on an arbitrary simple graph is an NP-hard problem, [6]. The distance domination problem is a distance-ℓ 1-component order, neighbor connectivity failure set problem, and is therefor NP-hard. It follows that the more generic k-component problem is NP-hard as well. Consequently, we limit the scope of our algorithmic efforts to trees.

We use $T = (V, E)$ to indicate a rooted tree arbitrarily rooted at r, and for some $v \in V$ we use $T_v = (V_v, E_v)$ to denote the subtree of T rooted at v. A node v's neighbor is the parent if it is on the unique path from v to r, and a child, $c \in C_v$, if it is not.

In order to construct an algorithm that finds a minimum failure set, we need to compute the order of a tree component. We do this as follows.

Definition 9. Let $v \in V$ and $F \subseteq V$. If there exists a $w \in F$ such that $v \in N_\ell[w]$ we say that component order $\text{comporder}(F, v) = 0$. If v is not in $N_\ell[F]$ with children C_v, we recursively define $\text{comporder}(F, v) = 1 + \sum_{c \in C_v} \text{comporder}(F, c)$. Where F is unambiguous, we omit it.

Lemma 10. *A component K of the graph induced by $V \setminus F$ has an order of* $\text{comporder}(v)$ *where v is the root of the smallest subtree containing K.*

Proof. The component K is a tree itself and the component order of the root r_K of K is the order of K. Proof by induction on the order of K is immediate. □

Remark 11. Consider Algorithm 1. We prove in Theorem 21 that the algorithm constructs a minimum failure set F. The algorithm takes in a vertex $v \in V$ as input. Any vertex is acceptable, and every vertex of V gets called by the recursive action of the algorithm. The user of the algorithm only calls the algorithm on r, initializing $F \leftarrow \emptyset$ when doing so. For implementation details beyond those presented here, see [10].

Example 12. We will run the algorithm on a small example problem with $\ell := 1$ and $k := 1$. The root of our graph will be $r = a$. We'll set a's children to a^2, ab and ac. We'll set a^2's children to a^3 and a^2b. We'll set ab's child to aba and aba's child to aba^2. See Figure 2.

Always before calling the algorithm on the root, we set $F \leftarrow \emptyset$. That is to say, the failure set is empty, no vertices have been marked for failure. We call Algorithm 1 on $v \leftarrow r = a$. On Line 1 we immediately call Algorithm 1 on a^2, ab, and ac. We'll look at each of these in turn and, without loss of generality, begin with $v \leftarrow a^2$. On Line 1 we call Algorithm 1 on a^2's children and proceed to $v \leftarrow a^3$.

Since a^3 has no children, we proceed to Line 2 with $\text{comporder}(a^3) \leftarrow 1$. On Line 3 we note that $a^3 \neq r$ and proceed to Line 4. Since a^3 has no 1-generation descendants (children), the condition is false, and we complete Algorithm 1 on a^3.

Algorithm 1: Selects Vertices for Failure

Input:

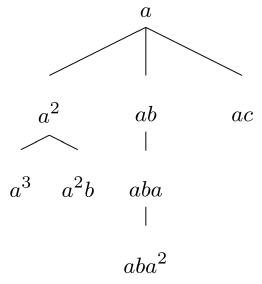
- Constants $k, \ell \in \mathbb{N}$ as described above.
- A vertex v of a tree T with root r.
- A global $F \subseteq V$. If $v = r$, then $F \leftarrow \emptyset$.

Output:

- $F \subseteq V$ is the failure set after the algorithm has run on r.

1 **foreach** c child of v **do** Run Algorithm 1 on c
2 Set component orders for T_v.
3 **if** $v = r$ **and** $\exists u \in N_\ell[r]$ such that comporder$(u) \geq k$ **then** Add v to F.
4 **else if** $\exists u$ such that u is an ℓ-generation descendent of v with comporder$(u) \geq k$ **then**
5 Add v to F.

Figure 2: Example 12



We resume Algorithm 1's progress for $v = a^2$ and proceed to call Algorithm 1 on the next child of a^2, namely a^2b. Algorithm 1 runs on a^2b exactly as it did on a^3 so we omit the details.

Once Line 1 is completed for $v = a^2$ we continue to Line 2 where comporder$(a^3) \leftarrow 1$, comporder$(a^2b) \leftarrow 1$ and comporder$(a^2) \leftarrow 1 + 1 + 1 = 3$. On Line 3 we note that $a^2 \neq r$ so we proceed to Line 4 and check if there exists a 1-generation descendent of a^2 with a component order greater than or equal to 1, and find, without loss of generality, that a^3 has comporder$(a^3) \geq 1$. Therefore we add a^2 to F, equivalently $F \leftarrow F \cup \{a^2\}$. The vertices a^3, a^2b, a^2, and a are all in $N_\ell[a^2]$ and near a failed vertex. We have completed Algorithm 1's run on a^2 and return to $v = a$.

Algorithm 1 was on Line 1 for $v = a$, and now proceeds to call Algorithm 1 on ab, which in turn immediately calls Algorithm 1 on aba calling it on aba^2. Since $v = aba^2$ is a leaf, the algorithm runs just as it did on a^3. On

completion of $v = aba^2$ we return to $v = aba$. With aba having no other children, the algorithm proceeds to Line 2 setting comporder$(aba^2) \leftarrow 1$ and then comporder$(aba) \leftarrow 1 + 1 = 2$. Checking the condition on Line 3 we note that $aba \neq r$, and on Line 4 we find that aba does have a 1-generation descendent, aba^2 with comporder$(aba^2) \geq k = 1$, so we add aba to F giving $F = \{a^2, aba\}$. The algorithm completes its run on aba and we return to the run $v = ab$.

The algorithm has completed its recursive calls to all of ab's children, so we proceed to Line 2. Since aba^2, aba, and ab are all in $N_1[aba]$ they have component orders of 0. Proceeding to lines 3 and 4, the conditions are not met and the algorithm completes its run on ab, returning to a.

On Line 1, a has one last child, ac. We call Algorithm 1 on ac, which is a leaf so the algorithm runs as it did on a^3, and we continue to Line 2 for $v = a$. Note, every vertex in the subtrees rooted at a^2 and ab is in the neighborhood of some $f \in F$ so all those vertices have component order 0, as does a itself. All that remains is comporder$(ac) \leftarrow 1$. On Line 3 we note $a = r$ and $ac \in N_1[a]$ with comporder$(ac) \geq 1$, so we fail a and skip checking the **else if** condition. The algorithm is complete leaving $F = \{a, a^2, aba\}$ and $V \subseteq N_1[F]$.

If the Algorithm has only run on a subtree rooted at v, and that subtree contains a component with order at least k, that component must be failed by some vertex outside the subtree. The following Lemma and Theorem guarantee this.

Lemma 13. *Let $v \in V$. If Algorithm 1 has run on v, and there exists a $u \in V_v$ with* comporder$(F, u) \geq k$*, then $u \in N_{\ell-1}[v]$.*

Proof. We will prove by induction. For the base case, we will apply Algorithm 1 to a leaf, $v \in V$. Let $u \in V_v = \{v\}$. We have $u = v$ with comporder$(u) = 1$. If $k = 1$, then for any $\ell \geq 1$, we have $u \in \{v\} \subseteq N_{\ell-1}[v]$, the desired result. If $\ell = 0$ then the **else if** condition on line 4 was met and v was added to the failure set, insuring comporder$(v) = 0$. If $k > 1$, then the desired result is achieved trivially.

For the inductive assumption, Line 1 calls Algorithm 1 on all the children of v. We assume that for any child c of v, if $u \in V_c$ and comporder$(u) \geq k$ then $d(u, c) < \ell$. Equivalently, $d(u, v) < \ell + 1$. If $d(u, v) < \ell$, or if there is no such u, then the desired result is achieved, so we will assume that before the algorithm proceeds to Line 2, their exists a u such that

$d(u, v) = \ell$ and comporder$(u) \geq k$. Line 4, or if $v = r$ Line 3, fails v resetting comporder$(u) \leftarrow 0$. Once the algorithm has run, no such u exists. □

Proposition 14. *Let Algorithm 1 have run on r. The set $F \subseteq V$ is a failure set for T.*

Proof. By Lemma 13 we have that any $v \in V$ with comporder$(v) \geq k$ is in $N_{\ell-1}[r]$. Algorithm 1 Line 3 guarantees that any such v was assigned comporder$(v) \leftarrow 0$. □

Next, we would like to show that for any arbitrary failure set of the tree, a surjective mapping may be constructed from that failure set, to the set constructed by the Algorithm.

Algorithm 2 plays a roll in our proof that the failure set F, created by Algorithm 1, is a minimum failure set. It is not meant to be used for any other purpose. When called on r, Algorithm 2 builds a mapping $M \mid W \to V$. Before Algorithm 2 is called on r, this mapping is initialized to the identity function. The mapping is a global variable whose value changes without being explicitly returned or passed. With each recursive call to Algorithm 2, M may be modified resulting with its final value a surjective mapping onto F (Lemma 20).

Definition 15. For a mapping $M \mid W \to V$, the image of M is denoted $M(W)$. We also define $M_v(W) := (M(W) \cap V_v) \setminus \{v\}$ after Algorithm 2 has run on all of v's children.

Definition 16. For convenience, if the statement on line 4 is reached and for all w, $M(w)$ is reset to v's parent p, we say that v**'s fail is moved up**.

Lemma 17. *Let Algorithm 2 run on $v \in V$. If W is a failure set, then $M(W)$ is a failure set.*

Proof. Let's assume W is a failure set. The starting value of M is the identity function, so we need only verify that when M is changed, on Line 4, no new components of order greater than or equal to k are created. Note, comporder$(M_v(W), u) \leq$ comporder$(M(W), u)$, so by considering comporder$(M_v(W), u)$ instead of comporder$(M(W), u)$ there is no risk of creating a new component of order greater than or equal to k.

The $M(W)$ components of T that may be affected by moving some v's fail can be divided into two categories. Those to whom the only path to v is

Algorithm 2: Constructs a Mapping to F

Input:

- $v \in V$
- $W \subseteq V$
- Global $M \mid W \to V$. If $v = r$, then $M \leftarrow$ the identity function.

1 **foreach** c child of v **do** Run Algorithm 2 on c.
2 Set component orders for T_v with the failure set $M_v(W)$.
3 **if** $\forall u : u$ is an ℓ-gen. descendent of v, comporder$(M_v(W), u) < k$, $v \neq r$ **then**
4 $\quad \forall w \in W : M(w) = v$, reset $M(w) \leftarrow v$'s parent.

through p, the parent of v, and those through whom the only path is through a child of v.

Vertices connected to v through p can have their $M(W)$-component order only reduced when v's fail is moved up, either to 0 because those vertices are in $N_\ell[p]$, or to less than what they were because a descendent was set to component order 0.

A vertex u connected to v through c, a child of v, can fall into 3 disjoint sub categories: $d(u, v) < \ell$, $d(u, v) = \ell$, and $d(u, v) > \ell$. Any $u \in N_{\ell-1}[v]$ still has comporder$(M(W), u) = 0$ after the change since $u \in N_\ell[p]$. Any u with $d(u, v) > \ell$ remains unchanged in component order since u's descendants are unchanged in their component orders.

We will now consider a vertex u with $d(u, v) = \ell$, that is, an ℓ-generation descendent of v. The algorithm looks at the component order of u as though v and its ancestors, are not in $M(W)$, which tells us what the component order will be if v's fail were moved up, as $u \in N_\ell[v] \setminus N_\ell[p]$. If the component's order will be greater than k, then the condition on Line 3 is false and the statement on Line 4 is never reached. Consequently, v's fail is not moved up. If we'll have comporder$(u) < k$, then v's fail is moved up. In either case, all components continue to have order less than k. □

Lemma 18. *Let W be a failure set, and Algorithm 2 have run on $v \in V$. If for some u that is an ℓ-generation descendent of v, we have* comporder$(M_v(W), u) \geq k$*, then $v \in M(W)$.*

Proof. Lemma 17 tells us $M(W)$ is a failure set, so there exists $g \in M(W)$ such that $\text{comporder}(M_v(W) \cup \{g\}, u) < \text{comporder}(M_v(W), u)$. It follows that $g \in (V \setminus V_v) \cup \{v\}$ and there exists a w that is a descendent of v, with $w \in N_\ell[g]$. For all such w, all paths from g to w go through both v and u. Therefore, $d(w, g) \geq d(u, g) \geq d(u, v) = \ell$. Equalities exist only when $w = u$ and $g = v$, the desired result. □

Definition 19. Let F_v be the failure set generated by Algorithm 1 when run on all the descendants of v.

Lemma 20. *Let $W \subseteq V$ be a failure set and Algorithm 2 have run on r and W generating M. Let Algorithm 1 have run on r generating F.*

1. $M(W) \subseteq F \cup \{r\}$, *and*
2. $M(W) \supseteq F$.

Proof. We will prove with induction. For the base case, let $v \in V$ be leaf.

($1 \subseteq$) Let $v \in M(W)$. If we falsely assume $v \neq r$ then the condition in Algorithm 2 Line 3 is trivially true and Line 4 insures that v is not in $M(W)$, a contradiction. We assume $v = r$. Then $v \in F \cup \{r\}$.

($2 \supseteq$) Let $v \in F$. If $\ell = 0$, then $k = 1$, and since W is a failure set with $N_0(W) = W$, we have $v \in W$. Note that Algorithm 2 does not move v up, giving $v \in M(W)$. If $\ell \geq 1$, there are no ℓ-generation decedents of v. We arrive at $v = r$ and $k = 1$. Since W is a failure set, $W = \{v\}$. Under these circumstances, Algorithm 2 did not move v's fail up and $M = id \Rightarrow M(W) = W = F$.

For the inductive hypothesis, we assume that for any non-leaf vertex $v \in V$, if algorithms 1 and 2 have run on a descendent u, then if $u \in M(W) \iff u \in F$, or equivalently, $F_v = M_v(W)$.

($1 \subseteq$) Let $v \in M(W)$ after Algorithm 2 has run on v. This means v's fail was not moved up and there is a $w \in W$ such that $M(w) = v$. The condition in Algorithm 2 Line 3 was false. If $v \neq r$, there exists u that is an ℓ-generation descendent of v with $\text{comporder}(M_v(W), u) \geq k$. Since $\text{comporder}(F_v, u) = \text{comporder}(M_v(W), u)$, we may conclude that when Algorithm 1 Line 4 arrived at v, the condition was true and v was added to F achieving the desired result. If $v = r$ then the desired result is immediate.

($2 \supseteq$) Let $v \in F$. If $v \neq r$, then Algorithm 1 found a u that is an ℓ-generation descendent of v with $\text{comporder}(F_v, u) \geq k$. By the inductive assumption, $\text{comporder}(M_v(W), u) \geq k$. By Lemma 18, $v \in M(W)$. If

$v = r$ then Algorithm 1 failed r because there exists a $u \in N_\ell[r]$ such that comporder$(F_r, u) \geq k$. Using the inductive assumption together with Lemma 17, we conclude that there exists a $g \in M(W) \setminus M_r(W)$ that lowers the component order of u. The only element that could be is r. We conclude $v = r \in M(W)$. □

We arrive at our final results.

Theorem 21. *The set F of all vertices marked as failed after Algorithm 1 has been called on r is a minimum failure set for T.*

Proof. Lemma 14 gives F is a failure set. Lemma 20.2 tells us that for any failure set W, there exists a surjective mapping from W to F, $|F| \leq |W|$. □

Theorem 22. *The computational complexity of Algorithm 1 run on r is $O(n^2)$.*

Proof. All n descendants of r are called by Line 1. The complexity is $O(n \cdot k)$ where k is the complexity of each vertex's operations. It remains to compute k.

The complexity of Line 2 is at most $O(n)$ since each vertex has at most n descendants over which the recursive component order computation iterates.

Lines 3, 4, and 5, along with the **then** statement in 3 are similarly $O(n)$. This gives the complexity of Algorithm 1 as $O(n \cdot (n + n + n + n + n)) = O(n^2)$. □

Acknowledgments

Thanks to the Stevens/Seton Hall Graph Theory Group for their continued support.

References

[1] K. Ameenal Bibi, A. Lakshmi, and R. Jothilakshmi. Applications of distance-2 dominating sets of graphs in networks. *Advances in Computation Sciences and Technology*, 10:2801–2810, 2017.

[2] Gary Chartrand, Linda Lesniak, and Ping Zhang. *Graphs and Digraphs*. Chapman and Hall, Boca Raton, 2011.

[3] Alexis Doucette, Alicia Muth, and Charles Suffel. An exceptional invariant k-component order neighbor connectivity: a generalization of domination alteration sets in graphs. *Congr. Numer.*, 230:167–190, 2018.

[4] Daniel Gross, Monika Heinig, Lakshmi Iswara, L William Kazmierczak, Kristi Luttrell, John T Saccoman, and Charles Suffel. A survey of component order connectivity models of graph theoretic networks. *WSEAS Transactions on Mathematics*, 12:895–910, 2013.

[5] Teresa W. Haynes, Stephen T. Hedetniemi, and Peter J. Slater. *Fundamentals of domination in graphs*. CRC Press, Boca Raton etc., 1998.

[6] Toresa W Haynes, Stephen T Hedetniemi, and Peter J Slater. Domination in graphs (advanced topics) marcel dekker publications. *New York*, 1998.

[7] Kristi Lutrell, Iswara Lakshmi, L William Kazmierczak, Daniel Gross, John T Saccoman, and Charles Suffel. The relationship between neighbor-connectivity, component order connectivity, and neighbor-component order connectivity. *Congressus Numerantium*, 212:15–30, 2012.

[8] Kristi Luttrell. *On the neighbor-component order connectivity model of graph theoretic networks*. PhD thesis, Stevens Institute of Technology, 2013.

[9] Alicia Muth. *A Generalization of Distance Domination*. PhD thesis, Stevens Institute of Technology, 2022.

[10] E. Dov Neimand. Code Written for An Algorithm for Trees a Generalization of Distance Domination, 1 2022. URL: `https://github.com/KayakDov/An-Algorithm-for-Trees-a-Generalization-of-Distance-Domination`.