Multiple domination models for placement of electric vehicle charging station in road networks

Pako Kesiamang

Supervisor: Dr.A.Roux

1 Abstract

Electric vehicles have been developed to reduce carbon emissions. However, their limited range and the need for frequent recharging presents significant challenge. Strategically placing charging stations is essential to enhance the usability of Electric Vehicles (EVs). In this paper, we model the locations of charging stations on the Stellenbosch road network as a multiple dominations model to ensure that charging stations are accessible from all locations on the road network. To achieve an optimal solution, we utilize heuristics such as greedy methods, probabilistic method and the combined approach that integrates both methods. We compare our results with other research conducted in Dublin and Boston [8].

2 Introduction

The automobile industry's dramatic increase in the number of cars on the road has caused an increase in carbon emissions globally, which increases the environmental concern for the future. Technologies have been developed to make it possible to manufacture both low-emission electric and hybrid vehicles that are environment-friendly. While electric cars are an eco-friendly alternative, their limited range and frequent need for battery charging present significant challenges when compared to fuel-powered vehicles [1]. Thus, ensuring accessible charging stations are available is vital to making electric cars a practical option. Our project seeks to pinpoint the most effective locations to install charging stations along the Stellenbosch road network.

2.1 How to place charging station and its challenges

The placement of the charging stations may be modelled as a facility location problem. Determining the optimal placement and quantity of charging stations in a given road network is a complex optimization task that involves multiple objectives and constraints [8]. The optimal placement of charging stations can contribute to minimizing the amount of deviation drivers must take from their intended route to recharge their vehicles, but there might be other constraints, including:

- the number of charging stations must be small enough,
- the distance between consecutive stations must be within the vehicle's cruising range; and
- the charging stations must be adequate to prevent bottlenecks [8].

2.2 How to solve this problem

In order to solve the facility location problem, we will use a street network and convert it into a connected graph. Each vertex will represent a road intersection or a dead-end road and the edges of the graph will represent the roads themselves. The weight of each edge is the distance between the intersection and dead ends on the roads. Using this graph, we will create a *reachability graph* where each vertex represents an intersection or dead end in the road network. Where two vertices are connected if the distance between them is below a certain threshold. This reachability graph will show which points in the network are accessible to each other within the specified thresholds.

Using this graph we will make use of a multiple domination model to determine the optimal locations for charging stations so that a vehicle at every point in the network can reach one or more charging stations before running out of power. To do this, we set the threshold in the reachability graph to the distance the vehicle can travel on the remaining battery [8].

We will use three heuristic optimization algorithms to find an optimal placement. First, we will deploy a greedy algorithm, followed by a probabilistic heuristic and then a combined approach, to identify a multiple dominating set of the graph. We will then evaluate the efficiency of these algorithms by determining the optimal placement of charging stations in Stellenbosch. Finally, we will compare our findings with those obtained for Dublin and Boston in the study of [8] to draw meaningful conclusions.

3 The reachability graph

3.1 Road network

The *road network* is represented as a graph $G_{RN} = (V_{RN}, E_{RN})$, where V_{RN} includes all road intersections and dead ends as vertices, and E_{RN} is the set of edges representing roads connecting two intersections (dead ends), with weights equal to the road segment lengths. That is, for $e \in E_{RN}$ the weight $w(e)$ is the segment length. In order to find the road network, OpenStreetMap (OSM) is utilized to acquire road network data. OSM was chosen because it is an open and free source widely recognized for volunteer geographical information [4].

The OSM database comprises of points (referred to as vertices), lines, and polygons (known as areas) [4]. OSM data extraction and road network generation are generated by OSMNx, a Python package converting OSM data into a network graph [6]. The road network data of Stellenbosch was acquired from OSM on 24 April 2024 and the graph G_{RN} in Figure 1 was generated by OSMnx.

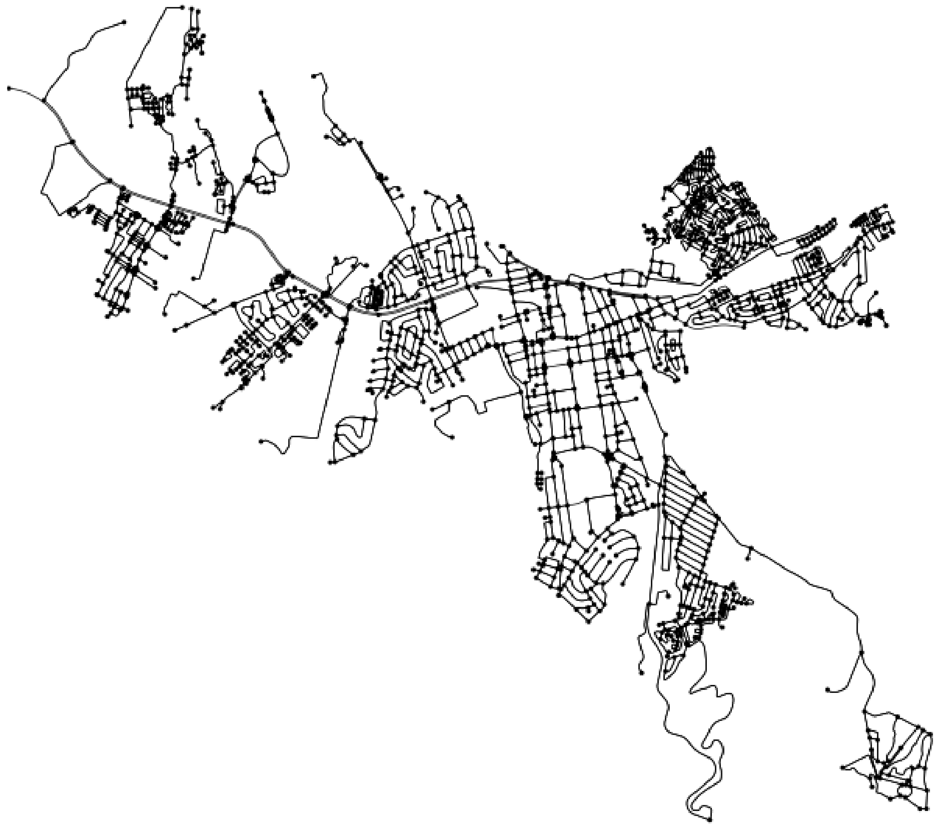


Figure 1: The road network of Stellenbosch

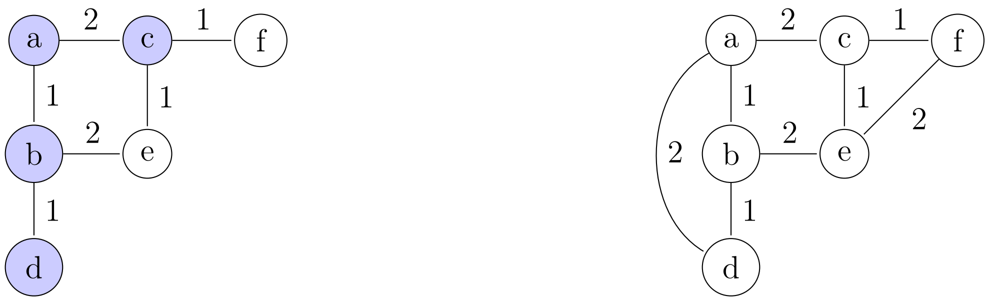
3.2 The reachability graph

In a *graph* $G = (V, E)$, two vertices $u, v \in V$ are *adjacent* if there exists an edge $uv \in E(G)$. The number of vertices n in G is called the *order*, while the number of edges in G is its *size* m. A *path* between two vertices u and v is a sequence of vertices, $P = (u = v_0, v_1, ..., v_i = v)$ such that v_{i+1} is adjacent to v_i and all the vertices of P are distinct. The length of the path P is the number of edges in the path. The *minimum distance* $d_G(u, v)$ between any two vertices u and v in G is the shortest path between u and v in G. If there exists a path in G from vertex u to vertex v, then vertex v is connected to vertex u in G. The *open neighbourhood* of a vertex $v \in V(G)$ is the set of $N_G(v) = \{u : uv \in E(G)\}$ of vertices adjacent to v in G and the closed neighbourhood is the set $N_G[v] = \{v\} \cup N_G(v)$. The *degree of a vertex* $v \in V$ is the number of vertices in the open neighbourhood of v and is denoted by $deg_G(v)$. The minimum and maximum vertex degrees of graph G are denoted by $\delta = \delta(G)$ and $\Delta = \Delta(G)$, respectively.

The *reachability graph* $G_R = (V_R, E_R)$ has a set of vertices $V_R = (v_1, v_2, ..., v_n)$ which concide with the vertex set of G_{RN}, and two vertices u and v are adjacent if their distance

$d_{G_{RN}}(u, v) \leq t$ for some distance t. This reachable graph G_R is suitable for modelling a scenario where a driver is concerned about battery degradation and needs to divert to a charging station accessible from his current location [3]. In other words, each point $v \in V(G_R)$ should be able to reach some neighbouring locations in $N_G(v)$ with the remaining battery.

We shall use a modified form of Dijkstra Algorithm 1 to determine all vertices in a road network that are reachable within a distance t from a source (initial vertex) $v_0 \in V_R$. This algorithm takes a road network G_{RN}, source vertex v_0, threshold t as input and produces the set of all reachable vertices within the distance t from v in the reachability graph G_R. This is done for every vertex. The algorithm updates two sets during the execution: the *frontier* F containing vertices and their distance at the current stage from the source and R the *reachable set* that stores the reachable vertices within the threshold of distance t. The algorithm starts by initiating $F = \{(v, 0)\}$ and setting the distance between v and any other u vertex to $D(u) = \infty$. Next, it extracts $(v, 0)$ from F and adds v to R. For every vertex u in the open neighbourhood of v, it adds $(u, w(v, u))$ to the frontier. It then extracts any of the elements in F say, (u, k), $k = w(u, v)$, and if $k > t$, the algorithm breaks. Otherwise, it checks if u is not already in R and adds u to the reachable set. Next, for every vertex w in the open neighbourhood of u, it adds $(w, D(w))$ into the frontier F, where $D(w)$ is updated to $D(u) + w(u, v)$ if it is smaller than the current value of $D(w)$. This process is repeated until F is empty.



(a) G_{RN} (b) G_R

Figure 2: (a) An example road network G_{RN} illustrating the modified Dijkstra's algorithm where the reachable nodes are shaded with $t = 2$. (b) The reachability graph G_R.

Consider an example of a road network G_{RN} in Figure 2a, with a set of vertices $V_{G_{RN}} = (a, b, c, d, e, f)$ and the weight of the edges as shown. Consider the source vertex a in G_{RN} and let the threshold be $t = 2$. The algorithm initiates the frontier as $F = \{(a, 0)\}$, and sets the reachable set to $R = \emptyset$. First, extract $(a, 0)$ from F and add a to the reachable set. Thus $R = \{a\}$. Since $N_{G_{RN}}(a) = \{b, c\}$, we add $(b, w(ab) = 1)$ and $(c, w(ac) = 2)$ to the frontier F. The new frontier is $F = \{(b, 1), (c, 2)\}$. Next, we extract $(b, 1)$ from the frontier F. Since $D(b) \leq 2$, the algorithm add b to the reachable set, thus $R = \{a, b\}$. For the neighbors of b not in R, e and d, it adds $(e, D(b) + w(be) = 3)$ and $(d, D(b) + w(bd) = 2)$ to the frontier. The new frontier is $F = \{(c, 2), (e, 3), (d, 2)\}$.
Next, it extract $(d, 2)$ from the frontier. Since $D(d) \leq 2$, add d to the reachable set. Thus, $R = \{a, b, d\}$, but d has no neighbors except b, which is already in the reachable set. The new frontier is $F = \{(c, 2), (e, 3)\}$. Next, it extract $(e, 3)$ from the frontier F, but since $D(e) \geq 2$ we can not add it to the reachable set. Thus, a new frontier is $F = \{(c, 2)\}$ and

the reachable set is $R = \{a, b, d\}$. Finally, it extract $(c, 2)$ from the frontier and add c to the reachable set. Since $D(c) \leq 2$, For the neighbors of c (except a), e and f, it add $(e, D(c)+w(ce) = 3)$ and $(f, D(c)+w(cf) = 3)$ to the frontier, thus $F = \{(e, 3), (f, 3)\}$. At this point, since both the distance of $d(a, e)$ and $d(a, f)$ exceed 2, the algorithm terminates. The reachable set is $R = \{a, b, c, d\}$. Repeating Algorithm 1 for the remaining vertices of G_{RN} we form the reachability graph G_R, shown in Figure 2b.

Algorithm 1: Modified Dijkstra's Algorithm to Find Reachable Nodes

```
Input: A Road network Graph $G_{RN}$, source (initial vertex) $v$, and threshold $t$
Output: Reachable nodes
frontier ← {(v, 0)} (starting $v$ and distance, 0)
reachable ← $\emptyset$
$D(v) \leftarrow \infty$ (a set store distances to each vertex)
$D[v] \leftarrow 0$
while frontier is not empty do
    (u, d) ← extract and remove a vertex from the frontier;
    if $d > t$ then
        break
        else
        reachable ← add $u$ to reachable set
        foreach $N(u) \backslash R$, $x$ do
            edge weight  $w(u, x)$ ← distance from $u$ to $x$
            if $D(u) + w(u, x) < D[x]$ then
                $D[x] \leftarrow D(u) + d_{ux}$
                add $(x, D[x])$ to frontier
            end
        end
    end
end
return reachable
```

Figure 3 illustrates the vertices adjacent to the blue vertex in the reachability graph G_R, corresponding to the Stellenbosch road network. The adjacent vertices are highlighted in red, with a distance threshold of $t = 1$ km.

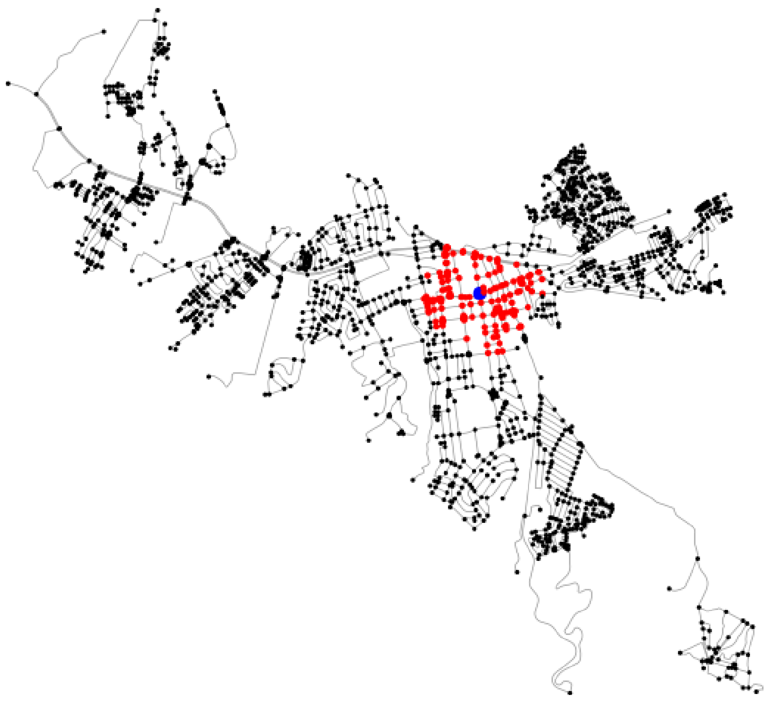


Figure 3: The red vertices represent the neighbourhood of the blue vertex in the reachability graph of Stellenbosch, with a threshold distance of $t = 1$km.

4 Domination

In a graph G, a vertex $v \in V(G)$ dominates itself and each of its adjacent vertices. A *dominating set* D of a graph G is a set of vertices of G such that each vertex not in D is adjacent to at least one vertex of D. In a graph G, *minimum dominating set* G is a dominating set of *minimum cardinality*. The cardinality of a minimum dominating set of G is called the *dominating number* and is denoted by $\gamma(G)$. A minimum dominating set is called the γ-set of G. A facility location is an optimization problem that aims to find an optimal placement of facilities so that every vertex(representing location) in the graph is adjacent to at least one facility. If a facility is placed on each vertex of a dominating set, every vertex in the graph will be adjacent to a facility. Therefore a minimum dominating set will provide an optimal placement of facilities [8].

In the following theorem, lower and upper bounds for the dominating number of a graph, each expressed in terms of the graph's order and its maximum degree, is given.

Theorem 1 (Chartrand and Zhang [2])**.** *If G is a graph of order n, then*

$$\frac{n}{1 + \Delta(G)} \leq \gamma(G) \leq n - \Delta(G).$$

Consider the reachability graph G in Figure 4, with a set of vertices $V = (a, b, c, d, e, f, g, h, i)$ representing the possible locations to place a charging station. The following sets are all dominating sets of G in Figure 4.,$D_1 = \{a, f, d, i, e\}$, $D_2 = \{b, c, h, g\}$ and $D_3 = \{h, b, d\}$.

The graph G, in Figure 4 has order $n = 9$ and maximum degree $\Delta = 4$. It follows from Theorem 1 that $2.2 \leq \gamma(G) \leq 5$. Since $\gamma(G)$ is an integer, $3 \leq \gamma(G) \leq 5$.

The minimum number of vertices that dominate G is 3. From Theorem 1, $\gamma(G) = |D| \geq 3$. Since $\{b, d, h\}$ is a dominating set, it follows that $\gamma(G) = 3$.

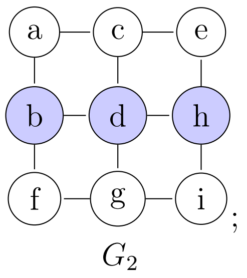


Figure 4: The reachability graph G_2, where the shaded vertices represent a minimum dominating $\gamma(G)$ set of G_2.

Given integer $k \geq 1$, a *k-dominating set* of a graph G is a subset $D \subseteq V(G)$ such that each vertex not in G is adjacent to at least k vertices in D [11]. The concept of k-domination refers to a type of multiple domination in graphs [4]. In the reachability graph, a k-dominating set will provide the location of the charging station such that a vehicle at any vertex can reach k charging station on the reachability graph. The *k-domination number* $\gamma_k(G)$ is the minimum size of a k-dominating set of G [11]. Thus, $\gamma_1(G) = \gamma(G)$, and $\gamma_{k_1}(G) \leq \gamma_{k_2}(G)$ when $k_1 \leq k_2$.

Theorem 2 (Fink and Jacobson [7]). *Let G be a graph of order n with maximum degree Δ and m edges and let $k \leq \Delta$ be a positive integer, then*

(i) $\gamma_k(G) \geq \frac{kn}{k+\Delta}$.

(ii) $\gamma_k(G) \geq n - \frac{m}{k}$.

Consider the same example of road network G in Figure 2. Let $k = 1$, then each vertex in $V \backslash D$ must be adjacent to at least one vertex in D. Since $\gamma_1(G) = \gamma(G)$, it follows that $\gamma_1(G_1) = 3$. In the following theorem, two lower bounds of k-domination are given.
Using Theorem 2(i), it follows that $\gamma_2(G) \geq 3$. However, the minimum number of vertices that dominates all other vertices twice of G is 4. To show this, assume $\gamma_2(G) = 3$. If $d \notin D$, every vertex in D is adjacent to at most 3 vertices in $V \backslash D$. If $|D| = 3$, then the vertices in D sends at most 9 edges to the vertices in $V \backslash D$.
However, the 6 vertices in $V \backslash D$ must each have at least 2 neighbors in D showing that we need at least 12 edges between $V \backslash D$ and D. This is a contradiction and so $d \in D$. Again assume that $|D| = 3$. The other two vertices dominate at most 2 vertices in $V \backslash D$ and we therefore have at most 4 edges between $D \backslash \{d\}$ and $V \backslash D$. But the remaining vertices in $V \backslash D$ each required to be dominated at least once, showing we need at least 6 edges between $D \backslash \{d\}$ and $V \backslash D$, a contradiction. It follows that $\gamma_2(G) = |D| \geq 4$. Since $\{c, b, h, g\}$ is a 2-dominating set, it follows that $\gamma_2(G) = 4$.

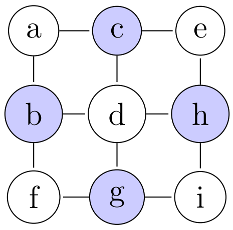


Figure 5: The reachability graph G_2, where the shaded vertices represent the minimum 2-dominating set of G_2 of cardinality of $\gamma_2(G) = 4$.

The following upper bound makes use of a probabilistic argument [3, 9]. It constructs a dominating set by first including each vertex with probability p into the set A. It then adds all the vertices not already k-dominated to A to form a k-dominating set. The bound then follows from the expected value of the formed dominating set.

We first introduce some basic probabilistic concepts. The *(finite) probability space* is the set Ω together with function $\mathbf{Pr}$, where Ω is a finite set called the *sample space* of the probability space and $\mathbf{Pr}$ is the *probability distribution* function that maps a subset of Ω to the interval [0,1] such that $\mathbf{Pr}(\Omega) = 1$ [10]. In a probability space, an *event* is a subset $A \subseteq \Omega$ [10]. The *probability* of an event $A \subseteq \Omega$ is given by

$$\mathbf{Pr}(A) = \sum_{G \in A} \mathbf{Pr}(G).$$

A random variable X is a function that maps the elements of the sample space Ω onto the real numbers $\mathbb{R}$, $X : \Omega \longrightarrow \mathbb{R}$ [12]. The *expected value* of a random variable X is expressed as

$$E(X) = \sum x\mathbf{Pr}(X = x).$$

If $X_1, .., X_2$ are random variables on a probabilistic space Ω and $c_1, ..., c_2$ are real number, then *linearity expectation* states that

$$E(c_1X_1 + ... + c_nX_n) = c_1E(X_1) + ... + c_nE(X_n). \tag{1}$$

Theorem 3 (Corcoran et al. [4]). *For every graph G with $\delta \geq k$*

$$\gamma_k(G) \leq \left(1 - \frac{\delta_0}{b_{k-1}^{1/\delta_0}(1+\delta_0)^{1+\frac{1}{\delta_0}}}\right) n;$$

where $\delta_0 = \delta - k + 1$, $b_{k-1} = \binom{\delta}{k-1}$ and $0 \leq k-1 \leq \delta$.

Proof: Let $G = (V, E)$ be a graph of order n with a minimum degree $\delta \geq k$. Consider a probability space Ω, where each element represents a possible subset of the vertex set $V(G)$. Since there are n vertices, there are 2^n possible subsets in this space.
For each vertex $v \in V(G)$, we select δ vertices from $N(v)$ and denote the resulting set by

$N'(v)$. Let $A \subseteq V(G)$ be a random set that is formed by choosing each vertex $v \in V(G)$ independently with probability p, where

$$p = 1 - \frac{1}{\sqrt[\delta_0]{b_{k-1}(1+\delta_0)}}.$$

For $m = 0, 1, ..., k-1$, let $B_m = \{v_i \in V(G) \backslash A : |N'(v_i) \cap A| = m\}$. The set D is defined as:

$$D = A \cup \left(\bigcup_{m=0}^{k-1} B_m \right).$$

This set D is a k-dominating set.

Let X be a random variable defined as $X(A) = |A|$. The indicator variable X_v, a random variable that takes only two possible values $X_v(A) = 1$ if $V \in A$ and 0 otherwise. For each vertex $v \in V(G)$ we have $E(X_v) = p$.

The size of set A is denoted by $|A|$, and we have

$$E(X(A)) = \sum_{v \in V} E(X_v) = \sum_{v \in V} \mathbf{Pr}(v \in A).$$

Hence, the expected value of cardinality of D is given by

$$E(X(D)) = E(|D|) \leq E\left(|A| + \sum_{m=0}^{k-1} |B_m| \right).$$

By the linearity of expectation (1), we have

$$E\left(|A| + \sum_{m=0}^{k-1} |B_m| \right) = E(|A|) + \sum_{m=0}^{k-1} E(|B_m|).$$

The expected value of $|A|$ is given by

$$E(|A|) = \sum_{v \in V} \mathbf{Pr}(v \in A) = \sum_{v \in V} p = np,$$

and that of $|B_m|$ is

$$E(|B_m|) = \sum_{i=1}^{n} \mathbf{Pr}(v_i \in B_m) = \sum_{i=1}^{n} \mathbf{Pr}(v \notin A \text{ and } v \text{ has } m \text{ neighbors in } A).$$

For each $v_i \in V(G)$, the set B_m consists of vertices that are not in A but have exactly m neighbors in A. The number of ways we can select m neighbors from δ neighbors can be calculated using the combinatorial coefficient $\binom{\delta}{m}$. The probability that $v \in A$ is $(1-p)$ and therefore we have

$$E(|B_m|) = \sum_{i=1}^{n}(1-p)\binom{\delta}{m}p^m(1-p)^{\delta-m}.$$

Now, the expected size of D becomes

$$\begin{aligned} E(|D|) &\leq pn + \sum_{i=1}^{n}(1-p)\binom{\delta}{m}p^m(1-p)^{\delta-m}b_m n \\ &= pn + (1-p)^{\delta-k+2}n\sum_{m=0}^{k-1}p^m(1-p)^{k-m-1}b_m. \end{aligned} \tag{2}$$

Furthermore, for $0 \leq m \leq k-1$,

$$b_m = \binom{\delta}{m} \leq \binom{\delta-m}{\delta-k+1}\binom{\delta}{m} = \binom{k-1}{m}\binom{\delta}{k-1} = \binom{k-1}{m}b_{k-1}. \tag{3}$$

Substituting equation (3) back into equation (2), we have

$$\begin{aligned} E(|D|) &\leq pn + (1-p)^{\delta_0+1}nb_{k-1}\sum_{i=1}^{n}\binom{k-1}{m}p^m(1-p)^{k-m-1} \\ &= pn + (1-p)^{\delta_0+1}nb_{k-1}. \end{aligned} \tag{4}$$

We have to select value of p such that minimizes the size of the expected value of D. By equating the derivative of the maximum $E(|D|)$ with respect to p, we have $\frac{d}{dp}(pn + (1-p)^{\delta_0+1}nb_{k-1}) = 0$. That is $0 = n + (\delta_0+1)(1-p)(-1)nb_{k-1}$, and so

$$p = 1 - \frac{1}{\sqrt[\delta_0]{b_{k-1}(1+\delta_0)}}.$$

Substituting p into (4), we have

$$\begin{aligned} E(|D|) \leq n\left(p + (1-p)^{\delta_0+1}b_{k-1}\right) &= n\left(1 - \frac{1}{\sqrt[\delta_0]{b_{k-1}(1+\delta_0)}} + \frac{1}{(\sqrt[\delta_0]{b_{k-1}(1+\delta_0)})^{\delta_0+1}}\right) \\ &= n\left(1 - \frac{\delta_0}{b_{k-1}^{\frac{1}{\delta_0}}(1+\delta_0)^{\frac{1}{\delta_0}+1}}\right). \end{aligned}$$

Since $E(|D|)$ is the expected cardinality of a k-dominating set there exist a k-dominating set of cardinality at most $E(|D|)$. It follows that

$$\gamma_k(G) \leq n \left(1 - \frac{\delta_0}{b_{k-1}^{\frac{1}{\delta_0}}(1+\delta_0)^{\frac{1}{\delta_0}+1}}\right),$$

which completes the proof of Theorem 3.

The upper bounds of Theorem 3 is known to be one of the best bounds for γ_k when $1 \leq k \leq \delta$ [3]. However, this bound tends to be less effective when working with the reachability graph G_R of the road networks. This is due to the fact that, in certain cases, the reachability graph G_R of the road network does not satisfy the condition that minimum degree $\delta \geq k$. Hence, instead of using the minimum degree of the reachability graph, we use the average vertex degree of the reachability graph G_R to compute probability p, as done in [3]. The average vertex degree is denoted by

$$\overline{d}(G_R) = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{5}$$

where d_i is the degree of vertex v_i, and n is the number of vertices in the graph. Since $\overline{d}(G_R) \geq \delta$, we can adapt the bound in Theorem 3 to

$$\gamma_k(G) \leq \left(1 - \frac{\overline{d}_0}{b_{k-1}^{\frac{1}{\overline{d}_0}}(1+\overline{d}_0)^{\frac{1}{\overline{d}_0}+1}}\right) n;$$

where $\overline{d}_0 = \overline{d}(G_R) - k + 1$, $b_{k-1} = \binom{\overline{d}(G_R)}{k-1}$ and $0 \leq k - 1 \leq \overline{d}(G_R)$.

5 Heuristics for determining k-domination sets

There is no polynomial time algorithm that exist to solve the k-domination problem for general graphs. Unless the graph G is reasonably small, one can not use an exact algorithm to compute an optimal solution in a reasonable time [3]. When working with a large graph, a heuristics method may be used to find a k-dominating set of a small cardinality in a reasonable time. In this paper, we use three heuristics optimization algorithms to find a k-dominating, set for the corresponding reachability graph G_R. The three algorithms that will be used are a *greedy method*, *probabilistic method* and *the combined approach* that integrates both methods. Since these algorithms might not return a minimal k-dominating set, we employ the *minimal k-dominating set* algorithm to reduce the proposed solutions to k-minimal dominating sets.

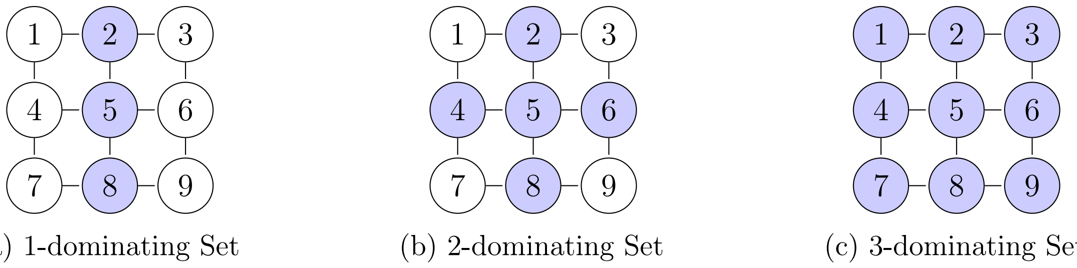
5.1 Greedy k-domination set method

A greedy makes locally optimal decisions in the hope that those decisions will lead to globally optimal decisions. However, this does not always give the best possible result. To determine a k-dominating set $D \subset V(G)$, we use the k-dominating greedy algorithm introduced in [3]. Algorithm 2 takes for input the reachability graph $G_R = (V, E)$, a positive integer k and an initial set $D \subset V(G)$, and creates a k-dominating set D of G_R. Then the algorithm checks if there is a vertex $v \in V(G_R) \backslash D$ that does not have at least k neighbors and adds it to U. From $V(G_R) \backslash D$ it chooses a vertex v that maximises the intersection of $N(v)$ and U. This process is repeated until $U = \emptyset$.

Algorithm 2: Greedy k-dominating set

```
**Input:** A reachability graph $G_R$, positive integer $k$ and $D \subset V(G)$
**output:** A $k$-dominating set $D$ of $G_R$
**while** $|\{v \in V(G_R) \setminus D : |N(v) \cap D| < k\}| > 0$ **do**
    Set $U = \{v \in V(G_R) \setminus D : |N(v) \cap D| < k\}$
    Find $u = \arg\max_{v \in V(G_R) \setminus D} |N(v) \cap U|$
    Set $D = D \cup \{u\}$
**end**
```

Consider the reachability graph G_2, in Figure 6 and apply Algorithm 2 with $D = \{2\}$ and k-values equal to $1, 2$, and 3.



(a) 1-dominating Set (b) 2-dominating Set (c) 3-dominating Set

Figure 6: Examples of k-dominating sets of graph G_2 where the shaded vertices represent the k-dominating set, for (a) $k = 1$, (b) $k = 2$, and (c) $k = 3$.

To illustrate how the greedy algorithm works consider the graph G_2 in Figure 6(a) with a set of vertices $V(G_2) = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Let $k = 1$ and choose $D = \{2\}$. Then $V(G_2) \backslash D = \{1, 3, 4, 5, 6, 7, 8, 9\}$. The algorithm starts by taking the set $D = \{2\}$ and for each vertex $v \in V(G_2) \backslash D$ adds it to set U if $|N(v) \cap D| < 1$. Since the only vertices adjacent to vertex 2 are the vertices 1,3 and 5, the rest of the vertices are added to U. Thus $U = \{4, 6, 7, 8, 9\}$.
Next the algorithm choose v from $V(G_2) \backslash D$ that maximise the intersection of $N(v)$ and U.

From Table 1 the neighbourhood of vertex 5 maximizes the intersection with U and is therefore added to D, resulting in $D = D_1 = \{2, 5\}$. Same as before: since vertices 7 and 9 are the only vertices not dominated by D, we have $U = \{7, 9\}$. Next, the algorithm chooses the vertex v in $V(G_2) \backslash D$, that maximizes the intersection of $N(v)$ and U. The only vertices that are adjacent to a vertex in U are $\{4, 6, 7, 8, 9\}$ and only vertex 8 is adjacent to both vertices 7 and 9. Therefore, the algorithm chooses 8 and set $D = D_2 = \{2, 5, 8\}$. Since D_2 dominates every vertex at least once the algorithm breaks.
The greedy algorithm will not always produce a minimal k-dominating set. For example, by looking at the 2-dominating set illustrated in Figure 6(b), the 2-dominating set is

v	$N(v) \cap U$
1	1
3	1
4	1
5	3
6	1
7	2
8	2
9	2

Table 1: The intersection of $N(v)$ and set U for each vertex v in $V(G_2)\backslash D$.

$\{2, 4, 5, 6, 8\}$, but vertex 5 can be removed from set D and the new remaining set is still a 2-dominating set of G_2.

5.2 Probabilistic Method

The probabilistic method uses a reachability graph G_R to generate a random set of vertices, where each vertex is chosen with a certain probability. The following algorithm, introduced in [4], takes a reachability graph G_R as an input, and a positive integer k (where $k \leq \overline{d}(G_R)$) and it produces a k-dominating set D of G_R. This algorithm is based on the upper bound of Theorem 3 [3].

The algorithm starts by computing probability p. It then creates a set A, by independently choosing each vertex $v \in V(G_R)$ with probability p. The algorithm then initializes an empty set B. Then for each vertex v in $V(G_R)\backslash A$, if v does not have k neighbors in A it is added to B. The algorithm then creates a k-dominating set $D = A \cup B$. Note that D may not be the minimal k-dominating set.

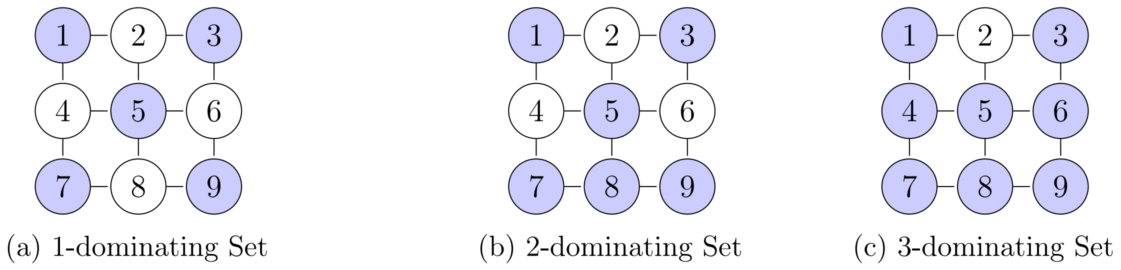


Figure 7: Examples of k-dominating sets of graph G_2 where the shaded vertices represent the k-dominating set, for (a) $k = 1$, (b) $k = 2$ and (c) $k = 3$.

Consider the reachability graph G_2, in Figure 7. When Algorithm 3 is applied to graph G_2 in Figure 7 it produces the shaded vertices for $k = 1$, $k = 2$ and $k = 3$ in (a), (b) and (c), respectively. Consider Figure 7(b) when $k = 2$, the algorithm begins by computing probability p and initiates A as the empty set. Each vertex in $V(G_2)$ is added to A with probability p. Based on Figure 7(b), the 2-dominating set is $D = \{1, 3, 5, 7, 8, 9\}$. Let's assume vertices 1, 3, 7, 8, and 9 were selected to be in set A.

Algorithm 3: Randomized k-dominating set

```
Input: A reachability graph $G_R$, and an integer $k$, $\overline{d}(G_R) \leq \delta$
Output: A $k$-dominating set $D$ of $G_R$
Compute the probability $p = 1 - \frac{1}{\sqrt[\delta_0]{b_{k-1}(1+\delta_0)}}$;
Initialize set $A = \emptyset$;                    (* Form a set $A \subseteq V(G_r^t)$ *)
foreach vertex $v \in V(G_r^t)$ do
    with the probability $p$, decide whether $v \in A$, otherwise $v \notin A$; (*this forms a
     subset $A \subseteq V(G_r^t)$ *)
end
Initialize $B = \emptyset$
for each vertex $v \in V(G_r^t) \setminus A$ do
    if $|N(v) \cap A| < k$ then
        $v$ is dominated by less than $k$ vertices of $A$
        add $v$ into $B$; (* this forms a subset $B \subseteq V(G_r^t) \setminus A$ *)
    end
end
Set $D = A \cup B$;            (*$D$ is a $k$-dominating set in $G_r^t$ *)
return $D'$
```

v	$\|N(v) \cap A\|$
2	2
4	2
5	1
6	2

Table 2: The intersection of $N(v)$ and set A for each vertex v in $V(G_2)\backslash A$.

Next, for each vertex $v \in V(G_2)\backslash A$, the algorithm adds v to B if $|N(v) \cap A| < 2$. From Table 2 vertex 5 is the only vertex not dominated twice, resulting in $B = \{5\}$. Since each of the vertices 2, 4, and 6 have 2 neighbors in A they are not added to set B. The algorithm then creates a set D, $D = \{1, 3, 7, 8, 9\} \cup \{5\} = \{1, 3, 5, 7, 8, 9\}$, which is necessarily a 2-dominating set.

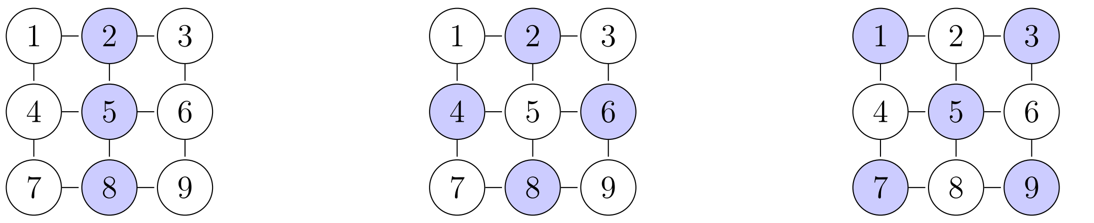
5.3 Minimal k-dominating Set

The k-dominating set D found by Algorithms 2 and 3 is not always a minimal k-dominating set. To ensure that the k-dominating set is the minimal k-dominating set of the reachable graph G_R, we use Algorithm 4 [3]. The aim of this algorithm is to reduce the k-dominating set by removing vertices in D that are not necessary in the k-dominating set. Given a reachability graph G_R and a k-dominated set, the algorithm creates a minimal k-dominating set D of G_R.
The algorithm begins by creating ordered set list $L = \{v_1, \ldots, v_n\}$ such that the number of neighbors v_i not in D should not be greater than the number of neighbors of $D\backslash v_{i+1}$ not in D. For each vertex v_i in the ordered list L, the algorithm checks whether removing v_i from D results in k-dominating set, If $D\backslash v_i$ is a k-dominating set then the vertex v_i is

removed from D.

```
Algorithm 4: Minimal $k$-Dominating Set
  Input: A reachability grap $G_R$ and a $k$-dominating set $D$ of $G_R$
  Output: A minimal $k$-dominating set $D$ of $G_R$
  arrange the $v_i$ in $D$ as $L = (v_1, \ldots, v_{|D|})$ such that $v_i \in D$ and
   $|N(v_i) \setminus D| \leq |N(v_{i+1}) \setminus D|$
  for $i = 1$ to $|D|$ do
      if $D \setminus \{v_i\}$ is a $k$-dominating set of $G_r^t$ then
          $D \leftarrow D \setminus \{v_i\}$
      end
  end
```

To demonstrate the functionality of Algorithm 4, we apply it to the k-dominating sets derived from Algorithm 2 (section 5.1), for the reachability graph G_2 shown in Figure 6. Consider the reachability graph G_2, in Figure 8 and apply Algorithm 4 with k-dominating sets and k-values equal to $1, 2$, and 3.



(a) 1-dominating Set (b) 2-dominating Set (c) 3-dominating Set

Figure 8: Examples of minimal k-dominating sets of graph G_2 where the shaded vertices represent the minimal k-dominating set, for (a) $k = 1$, (b) $k = 2$, and (c) $k = 3$.

Consider the graph G_2 in Figure 8 (c) with a set of vertices $V(G_2) = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Let $k = 3$ and 3-dominating set D= $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The algorithm begins by sorting vertices in the set D based on the number of vertices they have that are not in D as set L, ensuring that $|N(v_i) \setminus D| \leq |N(v_{i+1}) \setminus D|$. Since $D = V$, the list L can have any order. Let $L = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Next, for each vertex v_i in L, the algorithm checks whether removing v_i from D still results in a 3-dominating set of G_2, if it does it removes v_i from D.

First, the algorithm removes vertex 1 from set D. Then $D_1 = D\backslash\{1\} = \{2, 3, 4, 5, 6, 7, 8, 9\}$, now the algorithm check whether D_1 satisfies 3-dominating set condition. Since vertex 1 is not in D_1, it must have at least 3 neighbors. Therefore, D_1 is not a 3-dominating set of G_2 and vertex 1 cannot be removed and set D remains unchanged.

Next, the algorithm removes vertex 2 from set D. Then $D_2 = D\backslash\{2\} = \{1, 3, 4, 5, 6, 7, 8, 9\}$ is still a 3-dominating set of G_2, vertex 2 is removed and D_2 is set as the new set D.

This process is repeated for every vertex v_i in L. This process results in the minimal 3-dominating set $D = \{1, 3, 5, 7, 9\}$ of G_2.

5.4 Combined Approach

The combined approach is a combination of the probabilistic method (section 5.2) with the greedy algorithm (section 5.1) to create a new k-dominating set. It uses the probabilistic method to create a set $A \subseteq V(G)$ choosing a vertex $v \in V(G)$ independently with probability p and rather than adding v to set B, A is used as the initial set, together with the greedy algorithm to generate the k-dominating set.

Algorithm 5: Combined Approach

```
**Input:** A reachability graph $G_R$, and an integer $k$, $\overline{d}(G_R) \leq \delta$
**Output:** A $k$-dominating set $D$ of $G_R$
Compute the probability $p = 1 - \frac{1}{\sqrt[\delta_0]{b_{k-1}(1+\delta_0)}}$;
Initialize set $D = \emptyset$;                (* Form a set $D \subseteq V(G_r^t)$ *)
**foreach** *vertex* $v \in V(G_r^t)$ **do**
    with the probability $p$, decide whether $v \in D$, otherwise $v \notin D$; (*this forms a
      subset $D \subseteq V(G_r^t)$ *)
**end**
**while** $|\{v \in V(G_R) \setminus D : |N(v) \cap D| < k\}| > 0$ **do**
    Set $U = \{v \in V(G_R) \setminus D : |N(v) \cap D| < k\}$
    Find $u = \arg\max_{v \in V(G_R) \setminus D} |N(v) \cap U|$
    Set $D = D \cup \{u\}$ (*$D$ is a $k$-dominating set in $G_r^t$ *)
**end**
**return** $D$
```

6 Experimental evaluation

6.1 Data

This section presents an empirical evaluation of the proposed method for computing k-domination sets in the Stellenbosch road network graph G_{RN}. The road network, illustrated in Figure 9(a), was obtained from OpenStreetMap as described in section 3.1. The road network graph G_{RN} of Stellenbosch consists of 1920 vertices and 2588 edges.
To evaluate the performance of the algorithms, we computed the reachability graph G_R for different thresholds t, ranging from 1 km to 3 km.
Table 3 displays the number of vertices in the reachability graph G_R, the threshold distance t and the corresponding number of edges for each value of t. Additionally, Table 4 present the average degree $\overline{d}(G_R)$ and probability p for each value of t and k, which is essential parameters in the probabilistic and combined approach algorithms.

Figure 9(a) below illustrates the Stellenbosch road network, and Figure 9(b), 9(c), and 9(d) highlights the neighbourhood of a vertex in the reachability graph as discussed in section 3.2, the red vertices represent the neighbourhood of the blue vertex in the reachability graph G_R with threshold distance $t = 1$ km, $t = 2$ km, and $t = 3$ km, respectively.

t km	No of vertices of G_R	No of edges of G_R
1 km	1920	95636
2 km	1920	248 717
3 km	1920	457 207

Table 3: The threshold distance t, the number of vertices in the reachability graph G_R of Stellenbosch and the corresponding number of edges for each value of t.

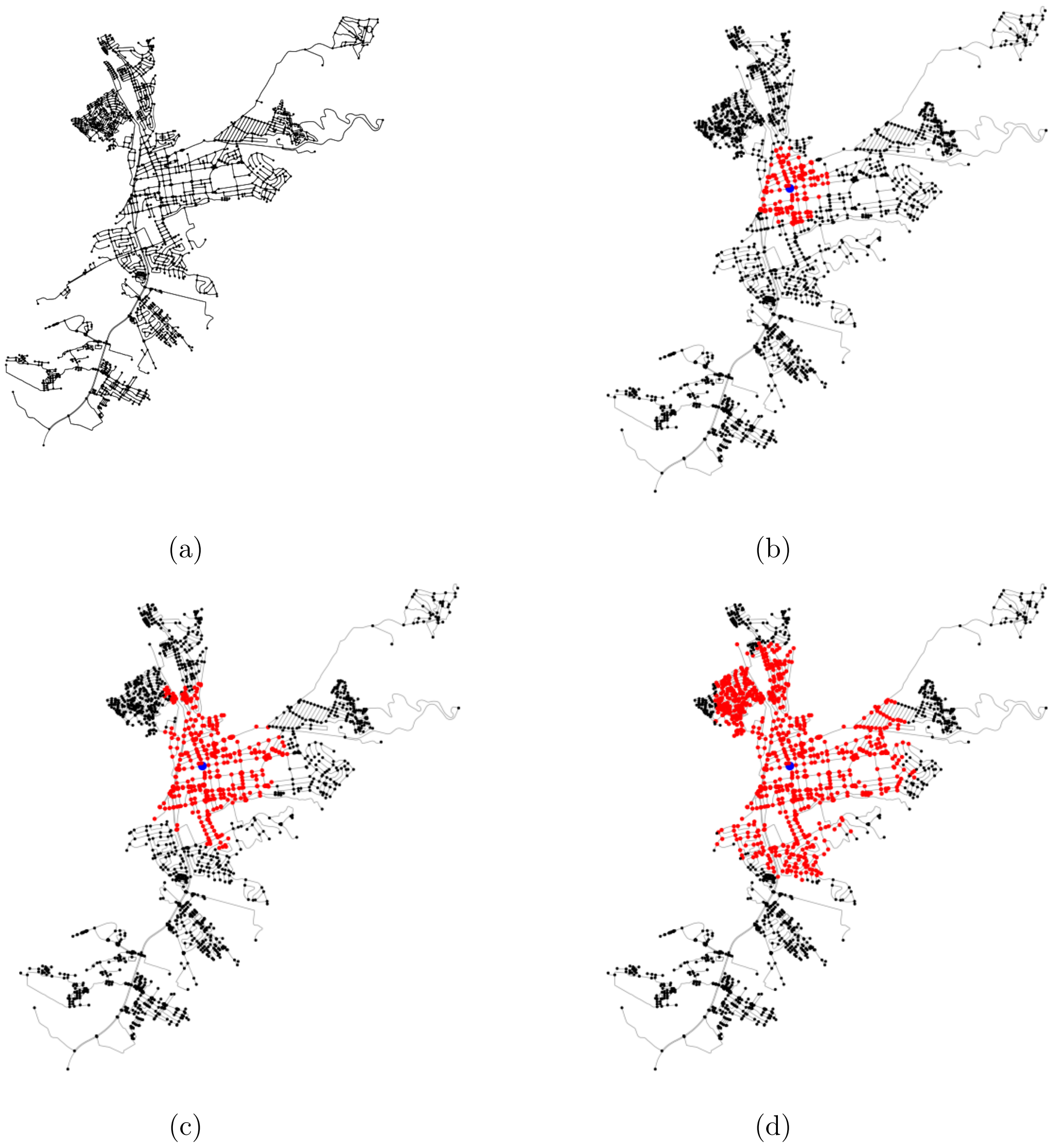


Figure 9: (a) The road network of Stellenbosch; The red vertices represent the neighbourhood of the blue vertex in the reachability graph for (b) $t = 1$ km, (c) $t = 2$ km and (d) $t = 3$ km.

t	k	Average degree $\overline{d}(G_R)$	Probability p
1 km	1	99	0.04545
	2	99	0.08951
	3	99	0.12608
2 km	1	259	0.02124
	2	259	0.04216
	3	259	0.06025
3 km	1	476	0.01287
		476	0.02563
		476	0.03685

Table 4: Average degree $\overline{d}(G_R)$ and probability p for each threshold distance t and k.

6.2 Algorithm Performance

In this subsection, we analyse the performance of the greedy, probabilistic and combined approach for determining the k-dominating set on the road network G_{RN} of Stellenbosch. To evaluate the effectiveness of each method, we computed the reachability graph G_R using the threshold distances of $t = 1, 2$ and 3km. To assess the influence of the minimal algorithm, we ran the greedy, probabilistic, and combined algorithms 10 times independently, followed by the minimal algorithm. For the greedy algorithm, the initial vertex was selected randomly for each iteration, and the average results for these algorithms were used.
The results are summarized in Table 5, which compares the performance of each algorithm with and without the minimal algorithm. Additionally, the table includes the results of the minimal algorithm (section 5.3) to observe how it performs independently. For the minimal algorithm, we started with all the vertices $V(G_R)$ in k-dominating set D.

t km	k	Greedy Method		Probabilistic		Combined		Minimal
		Alone	Minimal	Alone	Minimal	Alone	Minimal	Alone
1	1	48.4	41.8	211.7	56.4	1610.8	47.5	52
	2	1805.9	92	278.5	103.4	1609.6	88	98
	3	1810.9	134	346.5	146.6	1624.7	127.3	142
2	1	20.1	16.1	116.9	23.2	613.9	19.7	21
	2	1912.0	40	128.8	39.1	1430.4	33.7	41
	3	1912	57	174.1	54	1614.9	48.2	58
3	1	851.4	10.1	77.1	13.4	666.1	10.6	13
	2	853.7	20.7	98.7	22	557.7	21.1	23
	3	856.4	27.1	106.2	31.2	634.1	30.9	33

Table 5: Comparison of the greedy, probabilistic and the combined approach with and without the minimal algorithm, along with the minimal algorithm alone.

Although the probabilistic and the combined approach begins similarly when constructing a k-dominating set, the combined approach produce large k-dominating set compared to the probabilistic method. For instance, in the probabilistic method for $k = 1$ and $t = 1$, the expected number of k-dominating set is $211, 7$ while for the combined approach is $1610, 8$. This is due to the greedy refinement phase in the combined approach, which often introduces unnecessary vertices, leading to a big dominating set size despite the initial steps being similar to the probabilistic method.

Interestingly, in most cases, the probabilistic method independently produces smaller k-dominating sets compared to the greedy method and the combined approach alone. Comparing all the methods with the minimal algorithm alone, the probabilistic with minimal algorithm produces relatively similar results as the minimal algorithm alone, while the greedy and the combined approach methods with minimal produce smaller k-dominating sets.

6.3 Running time

In this section, we analyze the running time of the greedy, probabilistic and combined approach. This analysis aims to assess the efficiency of each algorithm when computed independently and with minimal algorithms. To provide a better insight into the efficiency of each algorithm, we measure the running time of each algorithm for threshold distance $t = 1$ and for $k = 1$, 2 and 3. Table 6 presents the running time of each algorithm with and without the minimal algorithm.

t km	k	Greedy Method		Probabilistic		Combined		Minimal
		Alone	Minimal	Alone	Minimal	Alone	Minimal	Alone
1	1	18.00 sec	25.46 sec	0.02 sec	1.13 sec	16.9 sec	23.71 sec	8.79 sec
	2	19.01 sec	25.66 sec	0.02 sec	1.59 sec	15.36 sec	22.23 sec	8.58 sec
	3	19.93 sec	26.51 sec	0.1 sec	2.26 sec	14.09 sec	20.76 sec	8.43 sec

Table 6: Running time of the greedy, probabilistic and combined approach with and without the minimal algorithm, along with the minimal algorithm alone for $t = 1$ and $k = 1$, 2 and 3.

The results in Table 6 show consistent running times for all algorithms across different values of k. The greedy, probabilistic and minimal algorithm running time increases slightly when k increases. For the combined approach, time decreases slightly when k increases. As suspected, when the minimal algorithm is applied to each method, there is a noticeable increase in the running time. For instance, in the probabilistic method for $k = 1$, the running time increases from 0.02 seconds to 1.13 seconds after applying the minimal algorithm. The minimal algorithm checks whether each vertex in the set is necessary. Since the probabilistic algorithm produces a smaller solution, the difference in time when the minimal algorithm is added is significantly reduced. The probabilistic algorithm is the most computationally efficient, both when computed alone and when combined with the minimal algorithm. This is because the probabilistic algorithm adds all non-dominated vertices to the set at once, whereas the other algorithms consider each vertex individually.

As shown in Table 5 and 6, the minimal algorithm produces relatively similar k-dominating sets compared to the probabilistic method and it is significantly more computationally expensive compared to the probabilistic method. In general, it is more practical to prioritize the method which is computationally cheaper if both methods produce the same results. The minimum algorithm is best used as a post-processing step to further optimize the solution generated by more effective methods such as greedy, probabilistic and the combined approach, since when used independently it results in a less optimal solution.

6.4 Comparative Discussion

In this section, we compare our results to those obtained for the cities of Dublin and Boston, as described in [3]. In their work, they modelled the road networks G_{RN} of both Dublin and Boston cities within the rectangular region and they computed the reachability graph G_R with a threshold distance set to 3.0 km for both cities' road networks G_{RN}.

Table 7 displays the length and width of the rectangular regions in which the road networks G_{RN} of both cities are contained and Stellenbosch. It also includes the number of vertices and the number of edges for all road networks G_{RN}, along with the number of edges in the reachability graph G_R, computed using the threshold distance of 3.0 km.
Furthermore, we calculate the density of each road network. The density of a graph is given by $\frac{2m}{n(n-1)}$, which is the ratio of edges to all possible edges [5]. This gives a basis for comparing the structure and complexity of the road networks across the three road networks.

City	**Region (width & length)**	**No. of Vertices in G_{RN}**	**No. of Edges in G_{RN}**	**No. of Edges in G_R**	Density of G_{RN}
Dublin	(29.5 km, 24.6 km)	55,162	64,437	54,306,700	0.0000423
Boston	(15.5 km, 12.1 km)	21,542	31,112	23,052,466	0.0001344
Stellenbosch	(—-, —-)	1920	2588	457,207	0.0014

Table 7: Dimensions and Properties of Road Networks and Reachability Graphs for Dublin, Boston and Stellenbosch

The results shows that the density of all three road networks are very low. Stellenbosch road network has a higher density, but even then, it still remains sparse. The road network of Dublin is incredibly large, both in terms of the number of edges, vertices and less connected compared to Boston.

For each reachability graph G_R with the threshold distance of 3 km, the greedy method was used to compute a single k-dominating set, while for the probabilistic method, ten k-dominating sets were computed for $k = 1, 2, 4$. From these ten runs, the smallest k-dominating set was selected and compared to the greedy method k-dominating set.
Table 8 presents the smallest k-dominating set from their study for both cities, computed using the greedy method and the probabilistic method, alongside the results of Stellenbosch obtained using the probabilistic, greedy, and combined approach for $k = 1, 2$ and 4.

Network	Boston		Dublin		Stellenbosch		
Algorithm	Greedy	Probabilistic	Greedy	Probabilistic	Greedy	Probabilistic	Combined
$k = 1$	32	31	110	111	9	12	10
$k = 2$	64	56	214	215	21	21	20
$k = 4$	122	115	413	411	43	38	37

Table 8: Comparison of the smallest k-dominating sets computed by the greedy, probabilistic, and combined approaches for the road networks of Boston, Dublin, and Stellenbosch.

In their findings, for Boston and Dublin they found out that in four out of the six cases, the probabilistic method produced a smaller k-dominating set compared to the greedy method. Compared to our case, Stellenbosch, the probabilistic outperforms the greedy for $k = 2$ and 4. The combined approach outperforms both the probabilistic and greedy methods for all values of k, this suggests that the combined approach leverages both the benefits of random exploration and systematic selection.
In our findings, the probabilistic method produces a smaller k-dominating set for $k = 2$ and 4(two out of three cases) compared to the greedy method and for $k = 1$ the greedy method produces a smaller k-dominating set. Our results correlate to their findings when comparing only the greedy method and the probabilistic method. When comparing all three methods, the combined approach produces a smaller k-dominating set for all cases. In general, all of those values are very close to each other, the only place where there is significant difference is for $k = 4$ for Stellenbosch and Boston, and for $k = 2$ for Boston.

7 Conclusion

In this paper, we explored the application of graph theory, specifically the multiple domination model to optimize the placement of electric vehicle charging stations within the Stellenbosch road network. We employed a range of heuristics methods, including the greedy, probabilistic, and the combined approach and we compared our results to those obtained for Dublin and Boston in [3].

Based on our analysis (section 6.2 and 6.4), for a dominating set ($k = 1$), for any distance threshold t, the greedy method performs well compared to the other two methods. That is, for a case where a vehicle can reach at least one charging station in the road network without running out of power. On the other hand when $k > 1$, for the k-domination problem, the combined approach demonstrates superior performance, and the greedy method again produces a smaller k-dominating set for $k = 3$. While the probabilistic algorithm is (marginally) worse than the other methods, the average running time is better than the other methods. Since it is the quickest algorithm, it is suitable for application that require a quick solution. We also seen that the minimal k-dominating set plays a big role in getting a smaller k-dominating set. In a recent study by Corcoran and Gagarin [3], a new method for solving k-domination model in the road network was proposed, i.e., *beam search*. It would be interesting to evaluate how their algorithm performs in comparison to ours.

For a possible direction in this area of research, we can explore the integration of the existing charging station into the optimal process, instead of starting with empty k-dominating sets(or starting with random vertex like in the greedy method), we could begin with the existing charging station as our initial k-dominating sets. This will be the development of a more practical model that focuses on expanding the current network.
Furthermore, the probabilistic method for constructing k-dominating sets can be improved. In the standard probabilistic approach (section 5.2), the set B is constructed separately from the initial probabilistic set A, based on the vertices that are not yet k-dominated. However, instead of creating a new set of B, we may update and expand the existing set A directly. This method may produce k-dominating set of a smaller cardinality. In the

future work, we could explore this method by testing it on large networks or different types of network to further evaluate its efficiency.

Acknowledgment

First and foremost, I would like to thank God for granting me strength, guidance, and perseverance throughout this journey. I would like to express my deepest appreciation for my supervisor, Dr. A. Roux from Stellenbosch University, for her invaluable feedback, guidance, and essential suggestions, which greatly enhanced the quality of my work.

A special thanks to my friends, family, and colleagues for their unwavering support and encouragement. I am also profoundly grateful to Mulilo Renewable for funding my studies, which made this work possible.

References

[1] Alanazi, F. [2023], 'Electric vehicles: Benefits, challenges, and potential solutions for widespread adaptation', *Applied Sciences* **13**(10).
URL: *https://www.mdpi.com/2076-3417/13/10/6016*

[2] Chartrand, G. and Zhang, P. [2005], *A First Course in Graph Theory*, Dover Publications, Mineola, NY, chapter 13, pp. 361–365.

[3] Corcoran, P. and Gagarin, A. [2021], 'Heuristics for k-domination models of facility location problems in street networks', *Computers Operations Research* **133**, 105368.
URL: *https://www.sciencedirect.com/science/article/pii/S030505482100143X*

[4] Corcoran, P., Mooney, P. and Bertolotto, M. [2013], 'Analysing the growth of openstreetmap networks', *Spatial Statistics* **3**, 21–32.
URL: *https://www.sciencedirect.com/science/article/pii/S2211675313000031*

[5] Darlay, J., Brauner, N. and Moncel, J. [2012], 'Dense and sparse graph partition', *Discrete Applied Mathematics* **160**(16), 2389–2396.
URL: *https://www.sciencedirect.com/science/article/pii/S0166218X12002399*

[6] Dingil, A. E., Schweizer, J., Rupi, F. and Stasiskiene, Z. [2018], 'Road network extraction with osmnx and sumopy', *EPiC Ser. Eng* **2**, 111–117.

[7] Fink, J. F. and Jacobson, M. S. [1985], n-domination in graphs, *in* Y. Alavi and A. T. White, eds, 'Graph Theory with Applications to Algorithms and Computer Science', Wiley, Kalamazoo, Michigan, pp. 282–300.

[8] Gagarin, A. and Corcoran, P. [2018], 'Multiple domination models for placement of electric vehicle charging stations in road networks', *Computers Operations Research* **96**, 69–79.
URL: *https://www.sciencedirect.com/science/article/pii/S0305054818300832*

[9] Gagarin, A., Poghosyan, A. and Zverovich, V. [2013], ‘Randomized algorithms and upper bounds for multiple domination in graphs and networks’, *Discrete Applied Mathematics* **161**(4), 604–611. Seventh International Conference on Graphs and Optimization 2010.
URL: *https://www.sciencedirect.com/science/article/pii/S0166218X11002423*

[10] Henning, M. A. and Vuuren, J. H. V. [2020], *Graph and Network Theory: An Applied Approach Using Mathematica*, 1st edn, Springer.

[11] Lan, J. K. and Chang, G. J. [2013], ‘Algorithmic aspects of the k-domination problem in graphs’, *Discrete Applied Mathematics* **161**(10), 1513–1520.
URL: *https://www.sciencedirect.com/science/article/pii/S0166218X13000395*

[12] Poghosyan, A. [2020], The Probabilistic Method for Upper Bounds in Domination Theory, Phd thesis, University of the West of England, Bristol.