IT UNIVERSITY OF COPENHAGEN

DEPARTMENT
COMPUTER SCIENCE

RESEARCH GROUP
NETWORKS, DATA, AND SOCIETY

Exploring Hedging Strategies for Multiple Players in GeoGuessr: A Study of Centrality in Geospatial Networks

Thesis written by:
Christian H. P. Larsen
hecl@itu.dk

Supervisor:
Associate Professor
Michael Szell

June, 2023

Abstract

The online browser-based game GeoGuessr has increased in popularity over recent years and professionals are going viral with videos displaying their extraordinary geolocating abilities. Especially the multiplayer game mode intrigues from a computational optimization perspective, as team members have to agree upon a hedging strategy for their guesses. This thesis argues for and implements five different hedging strategies: *Random Guess in Network*, *Random Guess in Convex Hull*, *Divide and Conquer*, *Closeness Centrality*, and *K-Means Clustering*. All hedging strategies are tested with 100.000 iterations on the playing fields of Bornholm and Malta. Results show from the tests, that the *K-Means Clustering* hedging strategy is superior to the other proposed hedging strategies with up to 151.2% by a measure of the mean distance to the target node for the winning guess. Furthermore, the thesis discusses the computational constraints of the presented hedging strategies and the limitations of applying the k-means algorithm to geospatial data. The application of in particular the algorithmic procedures of the *K-Means Clustering* hedging strategy within domains apart from GeoGuessr is discussed, and a customized version of it is used to solve the warehouse location problem.

*Gratitude is offered to OpenStreetMaps
for providing data and enabling the
open source community.*

*I am thankful and grateful
for the trust and cooperation offered by
Associate Professor Michael Szell.*

*Thank you for the endless support
Marius, Henrik, Ursula & Kaj.*

Contents

Acronyms and Definitions

Arena The region or (collection of) countries, in which the target is located

Geolocating The act of translating imagery into a location

Hedging strategy The use of a strategy to place guesses on the world map

OSM OpenStreetMap

Target The location of the place the players have to guess

Plausible region The region or (collection of) countries, in which the target is believed to be located

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This chapter starts with an introduction to the online game GeoGuessr in section 1.1, which briefly explains the playing rules of the different game modes and clarifies the one game mode researched in this thesis. Followed by that, the general motivation for the thesis is given in section 1.2, which also touches upon the recent increase in popularity of the game. The research questions aimed to explore and answer in the thesis are presented in section 1.3. Finally, section 1.4 offers a summary of the contributions made to both the field of optimizing results in GeoGuessr and more generally to the studies of centrality in geospatial networks.

1.1 GeoGuessr

GeoGuessr [2] is an online browser-based geography game, that lets players compete in a variety of game modes. Options for both single-player, multiplayer, and team-based games are available on the platform.

During each round, the players are presented with some location that is covered by Google Street View [6]. From the imagery of this ground view, the players will pinpoint a guess on a world map, where they believe the location to be. For the different game modes, points are added or subtracted based on the distance from the guess to the actual location.

Common for all game modes is the capability to choose the *arena*, which limits the country, region, or some other characteristic of the locations offered by Google Street View.

The game mode investigated in this thesis, which from this point on will be regarded as the game itself, is a multiplayer team-based version. In this game mode, two teams consisting of three players each, are competing against each other. Both teams start with an equal amount of points, and the losing team of each round gets subtracted points equivalent to the difference in distance of their best guess compared to the best guess of the winning team. A team wins when the other team reaches zero or negative points.

1.2 Motivation

When first introduced in 2013, GeoGuessr saw its first spike in interest [1]. As reported in [1] based on data collected from the mainstream social media platforms, GeoGuessr's popularity after its launch however declined quickly. It was during the COVID-19 pandemic, the game had its renaissance and became a popular game among social media personalities [1]. During this time, more game modes were made available, and short video clips went viral online of players showing off their skills.

For some game modes, making use of a hedging strategy makes sense. Even when playing single-player, it is often more reasonable to place a guess in a plausible region rather than guessing in the outskirts of that region and risking a higher distance to the target.

In the multiplayer team-based version of the game used as the basis for the research in this thesis, the use of hedging strategies is essential for winning tournaments. A clip from YouTube shows the GeoGuessr streamer Rainbolt playing a tournament with teammates [24]. They quickly identify Argentina, Mexico, or the United States of America to be the possible countries of the target, however in order to cover the region in a suitable way they agree on a hedging strategy [24]:

"Yeah, it's Arg, Mexico, or US."
"Alright, I'm going Mexico somebody go US."

Since the players have to place their guesses within a short time limit (typically within 30 seconds), the hedging strategies have to be agreed upon fairly quickly. In the clip, the players eventually decide to place two guesses in the United States of America and one in Mexico to cover their plausible region the best.

The precision of the guesses is therefore not as important as the general accuracy, as only the best guess will be taken into consideration each round. In other words, it is often more suitable for the teams to agree on a hedging strategy rather than having all team members place guesses individually.

1.3 Research Questions

The optimal hedging strategy is non-trivial for at least two reasons: 1) the underlying space of possible locations, i.e. the street network on which Google takes Street View images, is potentially heterogeneously distributed over the arena, and 2) the minimum of multiple player guesses wins.

In order to define a hedging strategy and be able to classify it as *good* or *better* than alternative hedging strategies, also naive hedging strategies have to be defined and tested similarly to the believed more optimal hedging strategy. Furthermore, the setting - in this case, the *arena* - in which the test is taking place has to be reflected upon.

Summarized, this thesis seeks to explore, implement, test, and reflect upon one or more hedging strategies while answering the questions: *"How can the optimal hedging strategy for multiple players in GeoGuessr be defined?"* and *"Which algorithms and calculations can be used to implement such hedging strategies?"*.

1.4 Contributions

The research of this thesis contributes to the task of geolocating in the domain of GeoGuessr by offering pre-computed guesses for a given arena. Five hedging strategies are offered, and each solves the task with a unique set of algorithmic procedures.

More generally, the research explores the notion of centrality in geospatial networks and presents applications in domains apart from GeoGuessr. Among others, it uses a customized version of a proposed hedging strategy to solve the warehouse location problem.

A GitHub repository is offered to interested readers, that documents and exemplifies the steps from fetching the networks to generating the guesses.

Chapter 2

Background

In this chapter, the context for the thesis is set and put into perspective by analyzing and presenting existing research in section 2.1. To understand the data being processed in the thesis offered by OpenStreetMap [14], section 2.2 will clarify its form and attributes. Centrality in networks can be defined and used for many different applications, which is why section 2.3 will define its characteristics in this thesis. In particular, section 2.3.3 will explore the k-means algorithm.

2.1 Related Work

This section will present existing work and research done on the topic of GeoGuessr. The usage of GeoGuessr in educational settings is described in section 2.1.1. From there on, the existing research concentrates on optimizing the winning chances and investigating how human players have become experts. In section 2.1.2, the usage of trained neural networks is employed to play the game with the goal of outperforming humans. The human performance is explored in section 2.1.3, which among others enlightens how professionals have become expert players. An altered and simplified version of the game has been subject to a research paper, that is presented in section 2.1.4. Finally, section 2.1.5 presents a previous attempt by the author of this thesis to create a hedging strategy based on a static analysis of a geospatial network.

2.1.1 Utilization in Education

Both [64] and [46] report positive feedback on the use of GeoGuessr in educational settings. Different computer games are compared and ranked in [64]. The study concludes GeoGuessr to be the highest rated due to its focus on the relations of nature, humans, and economic conditions.

The research conducted in [46] was mainly focused on GeoGuessr. This study eventually concludes, that there *"should be given more place for games in syllabuses, books and activities"* [46].

2.1.2 Machine Learning Techniques

In [68], a deep neural network was developed that given an image from Google Street View somewhere in the United States of America will geolocate the most likely state. This implies, that their model is only applicable when the arena of GeoGuessr is limited to images within the borders of the United States of America.

A neural network is trained with a data set consisting of 2500 samples for each of the 50 states. For each state, the samples are selected based on the population density. Thereby a claim is made that population density has a correlation to the coverage offered by Google Street View: *[...] density-based sampling allows us to achieve a good spread of locations [... and] primes the model for more densely populated regions* [68]. Each sample consists of four images, as all the cardinal directions are covered by Google Street View and therefore accounted for in the data set. *50States10K* [68] is the name of the data set with a total of 500.000 collected images made available for open-source research.

DeepGeo performs best in the states of Hawaii and Alaska with a precision of 95.9% and 91.2%. This is likely due to the uniqueness of their landscapes [68]. However, the performance in other states such as Virginia and Texas goes as low as 9.2% and 10.4%. Furthermore, the neural network makes no attempts to place a guess within the state.

A neural network is also developed and relied upon in [70] to increase the performance of geolocation. It is stated, that *"geolocation is an extremely challenging task since many photos offer only few, possibly ambiguous, cues about their location"* [70]. Their approach is based on the assumption that geolocation is a classification problem.

The model called *PlaNet*, is trained with 126 million photos. These photos have been scraped from the internet, and all consist of supplementary EXIF data attached to them. As also mentioned in the paper, their *"dataset is [...] extremely noisy, including indoor photos, portraits, photos of pets, food, products and other photos not indicative of location"* [70].

It is chosen in [70] to split the world into cells containing an equal amount of photos. Thereby, *"sparsely populated areas are covered by larger cells and densely populated areas are covered by finer cells"* [70].

Results show, that *PlaNet* outperforms similar models and *"even reaches super-human performance"* [70]. However, *PlaNet* makes no further effort to place a guess within a cell, as it simply uses the center of it.

Similar to [68] and [70], [59] tries to categorize imagery into a known set of areas. To do this, [59] uses *"a combination of computer vision, machine learning and text retrieval methods to compute a ranking of likely countries of the location shown"* [59]. In other words; given an image from Google Street View, their model called *Country Guesser* will output a list of countries paired with the computed likelihood of the target being within their borders.

The research in [59] determined that 110 countries were sufficiently covered by Google Street View to qualify for being a part of the model. To estimate the solar position, [59] investigates the brightness of parts of the image. Furthermore, [59] uses character detection in combination with language libraries in Python to infer linguistic origin. The coloration of the image, possibly car license plates, and identification of general objects is also included in the geolocating procedure of [59].

It is reported in [59] that from a total of 220 guesses, the country of the target was ranked 14.7 on average, meaning that on average over 14 countries were more likely according to the model to be the country of the target.

2.1.3 Human Capabilities

In [58] the research focuses on geolocation capabilities from a human perspective. In order to investigate this aspect of geolocation, 15 participants were included in the study.

Some of the signs and traits the participants used for making their guesses included studying architecture, identifying languages, inferring driving rules, positioning the sun, finding road signs, and recognizing landscapes [58].

As for models trained to geolocate, also participants can get misled by laying too much weight on certain traits that point in the wrong direction. [58] calls this symptom *fixation.*

2.1.4 Voronoi Game

An altered and simplified version of GeoGuessr is studied in [36], which argues to have designed similarities with the game studied in [36] and the Voronoi Game. As summarized in [31], the *"Voronoi game is a geometric model of competitive facility location problem played between two players. [...] Payoff of each player is defined by the quantity of users served by all of its facilities. The objective of each player is to maximize their respective payoff"* [31].

In the first version of the game presented in [36], two players, $Player_A$ and $Player_B$, are competing to place a guess closest to the target, T. It is assumed, that $Player_A$ has placed all it's guesses $(a_1)..(a_n)$ before $Player_B$ has placed any of its guesses $(b_1)..(b_n)$. At the time $Player_B$ has to place its guesses, the best guess from $Player_A$ with the notion of a is known. At any time, $Player_B$ knows the difference in distance between its best guess so far to T and the distance from a to T.

In the second version of the game, the two players, $Player_A$ and $Player_B$, take turns in placing a guess. Now both players know the difference in distance to T for their respective best guesses a and b.

Lastly, a version of the game is defined in a one-dimensional setting. As also stated in [36], this version *"becomes trivial"* [36], as it resembles a challenge in binary search more than an actual strategy for outplaying and opponent in geolocating.

2.1.5 Static Analysis

Similar to the approaches designed, developed, and tested presented in this thesis, the research project [49], which was the pilot project on this topic prior to this thesis, describes procedures that will place guesses based on a static analysis of a geospatial network.

The research conducted limits the arena to the island of Bornholm and suggests three different approaches. All results have the conditions of a team consisting of three players defining a hedging strategy. An overview of Bornholm can be found in appendix A.

The first hedging strategy places all its guesses randomly in the geospatial network. Results show, that the mean distance to target for the best guess when running 1.000.000 iterations is 11.58 km [49].

The second hedging strategy places all its guesses within the convex hull of the geospatial network. Results show, that the mean distance to the target for the best guess when running 1.000.000 iterations is 11.45 km [49].

Finally, the hedging strategy called *Divide and Conquer*, which places its guesses based on a static analysis of the geospatial network, shows results for the mean distance to the target for the best guess of 11.13 km when running 1.000.000 iterations. The improvement of this strategy compared to the random strategies is therefore respectively 3.9% and 2.8% [49].

In this thesis, the ideas of [49] are among others extended to exploring further hedging strategies with algorithmic procedures including machine learning techniques.

2.2 OSM and Geospatial Data

OpenStreetMap [14] is an open-source geographic database, that is maintained by its volunteers and users. Besides storing and offering geographic data free of charge, it allows for various integrations and uses of its service. The data offered by OpenStreetMap consists of the entities listed in Tab. 2.1.

Geospatial data in general has many facets. One definition of *geospatial* by the dictionary Merriam-Webster suggests the following: *"consisting of, derived from, or relating to data that is directly linked to specific geographical locations"* [10]. In many ways, a data set from OpenStreetMap follows the suggested definition:

1. The data consists of nodes directly linked to specific geographical locations, which are in essence the building blocks.

2. Derived from the nodes are ways, relations, and members. These meta-objects inherit and build upon the fundamental physical linkage of the nodes and describe structures and connections.

3. Finally, the tags relate to the data, by supplying additional meta-data to the nodes and the relations of the nodes.

Table 2.1: Structure of OSM data

Entity	**Description**
Tag	A tag consists of a key-value pair, that describes some attribute of a node, way, relation, or member. For example, some nodes have the tag of `<k="addr:housenumber" v="8"/>` when describing a building. Ways typically have tags describing, what vehicles can drive on the road, in example `<k="bicycle" v="yes"/>`. Finally, relations have tags that describe the type of the relation, in example `<k="building" v="yes"/>`.
Node	Nodes most importantly consist of the attributes `id`, `latitude`, and `longitude`. Besides possibly having tags attached, they are described with other attributes such as `visible="true"`, `timestamp="2012-08-04T11:35:42Z"`, `user="Ernst Poulsen"`, and `version="1"`.
Way	Ways in OSM consist of a series of node references, attributes, and possibly tags. Attributes for ways are similar to the ones for nodes, except they do not contain latitude and longitude, as these are contained in the referenced nodes.
Relation	A relation in an OSM data set consists of an ordered list of nodes, ways, and members. It describes a greater characteristic of a collection of entities, such as natural areas, buildings, and routes.
Member	Members exist within relations and represent other relations. A relation can therefore participate in multiple other relations. Members contain the attributes of `type`, `ref`, and `role`. A `type` can for example have the value of `way` or `relation`. The `ref` is the reference to the `id` of the relation and the role can have values such `inner`, `outer`, `backward`, and `forward`.

2.3 Centrality

2.3.1 Measures of centrality

Centrality measures are essential to understanding macro- and micro-specific properties of geospatial networks, and networks in general. In [69], a study on bus routes identifies its network to be the type of *"complex networks [that] exhibit a heterogeneous nature"* [69]. These types of networks are intricate to analyze, as every node can be of multiple importance: *"The importance of a node is largely affected and reflected by the topological structure of the network to which it belongs. A node's importance is highly correlated with its capacity to impact the behaviours of its surrounding neighbours"* [69].

The different measures of centrality defined in [69] are mostly concerned and only relevant to networks of heterogeneous complexity. Degree centrality, strength centrality, eigenvector centrality, Katz centrality, PageRank centrality, and betweenness centrality are identified in [69]. However, closeness centrality *"defined as the reciprocal of the mean* *geodesic distance from a node to all the other nodes in the network"* [69] has a potential application in the static analysis required for defining hedging strategies.

The types of centrality relevant to this thesis focus on the macro-specific properties of geospatial networks and thereby exclude measures that are reliant on node-specific analysis. Eigenvector centrality can, even though its primary applications lie within finding *"influential nodes in a network"* [63], be customized in a way to analyze distance importance. However, as with closeness centrality, also eigenvector centrality yields traits, that makes its use impractical in the research context of this thesis: the clustering of centralities. This phenomenon is explained and investigated further in section 2.3.2.

For calculating the closeness centrality of a node in a network, the following equation is used [44][75]:

$$C_c(p_k) = \frac{(n-1)}{\sum\limits_{i=1}^{n} d(p_i, p_k)} \tag{2.1}$$

Eq. (2.1) describes a measurement of closeness centrality C_c for a given node p_k, that is calculated by dividing the number of all nodes subtracted 1, $n-1$, with the sum of distances $\sum\limits_{i=1}^{n} d(p_i, p_k)$ from p_k to every other node $p_1..p_n$.

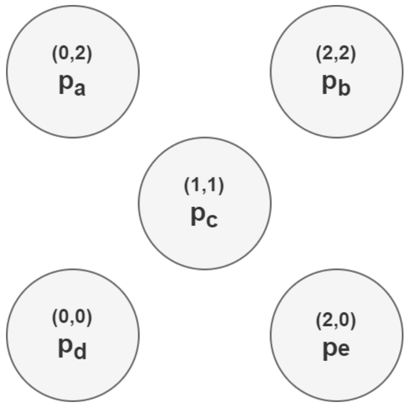
As also described in [57] and [56], the distance function is primarily understood as a Shortest Path implementation. However, the use of the Euclidean distance formula as the distance function is Eq. (2.1) is possible as well.

The Euclidean distance formula in a two-dimensional coordinate system is defined for two nodes, i and j, in Eq. (2.2) [37].

$$d(i,j) = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2} \tag{2.2}$$

In Fig. 2.1, a simple network of five nodes in a two-dimensional Cartesian coordinate system is visualized. Note that, since the distance measure is Euclidean distance, detached from graph distance, the placement of links is irrelevant. In this formulation, this centrality metric only cares about the spatial distances of nodes from each other. The nodes, $p_a..p_e$, all have their names and coordinates labeled.

Figure 2.1: Closeness centrality study in a simple network



When using Eq. (2.1) on the five nodes using the Euclidean distance defined in Eq. (2.2) as the distance function, the measurements resulting from the calculation are as shown in Tab. 2.2.

Table 2.2: Result of closeness centrality study in a simple network

Node	**Closeness centrality score**
p_a	0.485
p_b	0.485
p_c	0.707
p_d	0.485
p_e	0.485

2.3.2 Clustering of Centralities

As defined in [66], *"[t]he closeness centrality of a vertex is based on the total distance between one vertex and all other vertices, where larger distances yield lower closeness centrality values"* [66]. From this definition follows, that the likelihood of having the most central nodes as neighbors or in the relative near distance is high.

This phenomenon is investigated and reported as two figures, Fig. 2.2 and Fig. 2.3. The three nodes with the highest score of closeness centrality for the geospatial network of Bornholm are made visible in Fig. 2.2 as dots with the colors red, green, and blue. The figure shows, that all three nodes are in the center of the island close to each other. Fig. 2.3 visualizes this phenomenon by coloring the edges of the geospatial network also according to their score of closeness centrality. The figure indicates with a brighter edge-coloring, that the majority of highly scored edges are found towards the center of the island.

Both figures, Fig. 2.2 and Fig. 2.3, have been implemented using the NetworkX library function `closeness_centrality` [11].

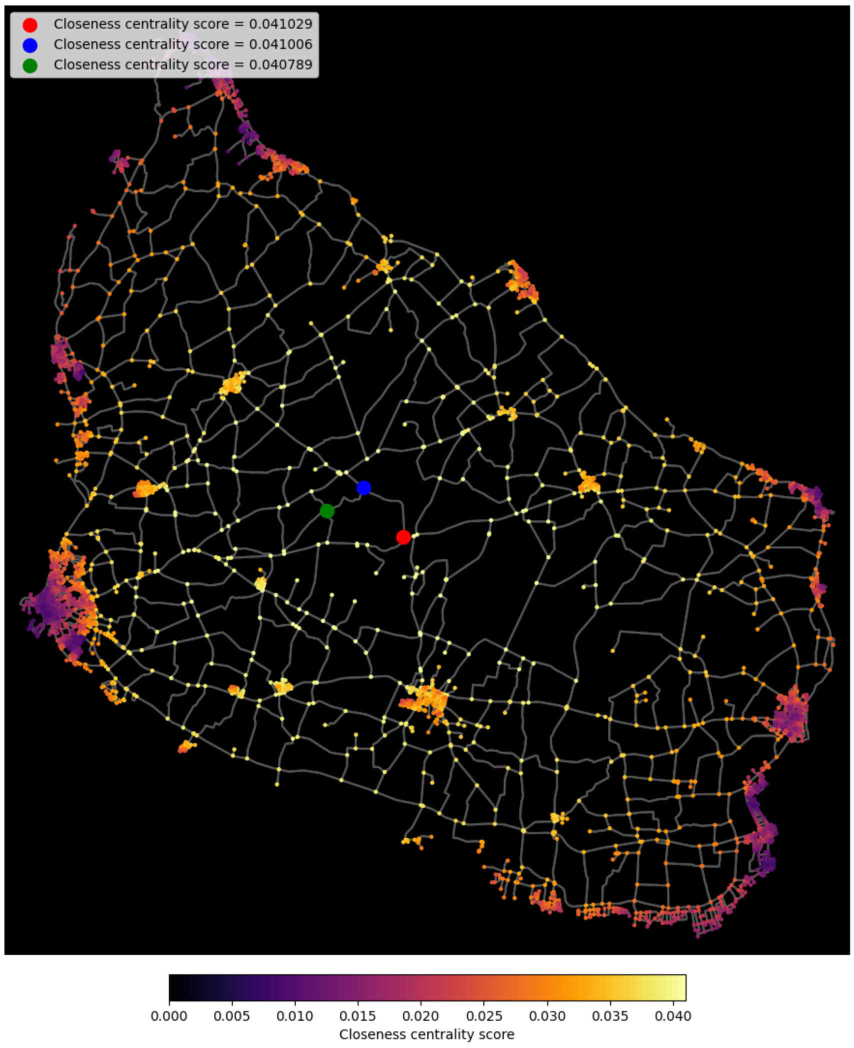
Implementation details and code can be found in the GitHub repository [23] for investigation and recreation of all results and plots presented in this thesis.

2.3.3 K-Means

As described and visualized in section 2.3.2, using off-the-shelf standardized centrality measures for the purpose of developing multiplayer hedging strategies in GeoGuessr is not useful since the multiple guesses should be spread so that they cover the arena more homogeneously to score a better minimum distance estimate on average. Other methods should be explored to approach the task from new perspectives.

In [60], research has been conducted that uses parallel k-means clustering to investigate geospatial vegetation data. This was done by studying a LIDAR data set and clustering different vegetation structures [60].

Figure 2.2: Closeness centrality shown by nodes for Bornholm



Closeness centrality score = 0.041029
Closeness centrality score = 0.041006
Closeness centrality score = 0.040789
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
Closeness centrality score

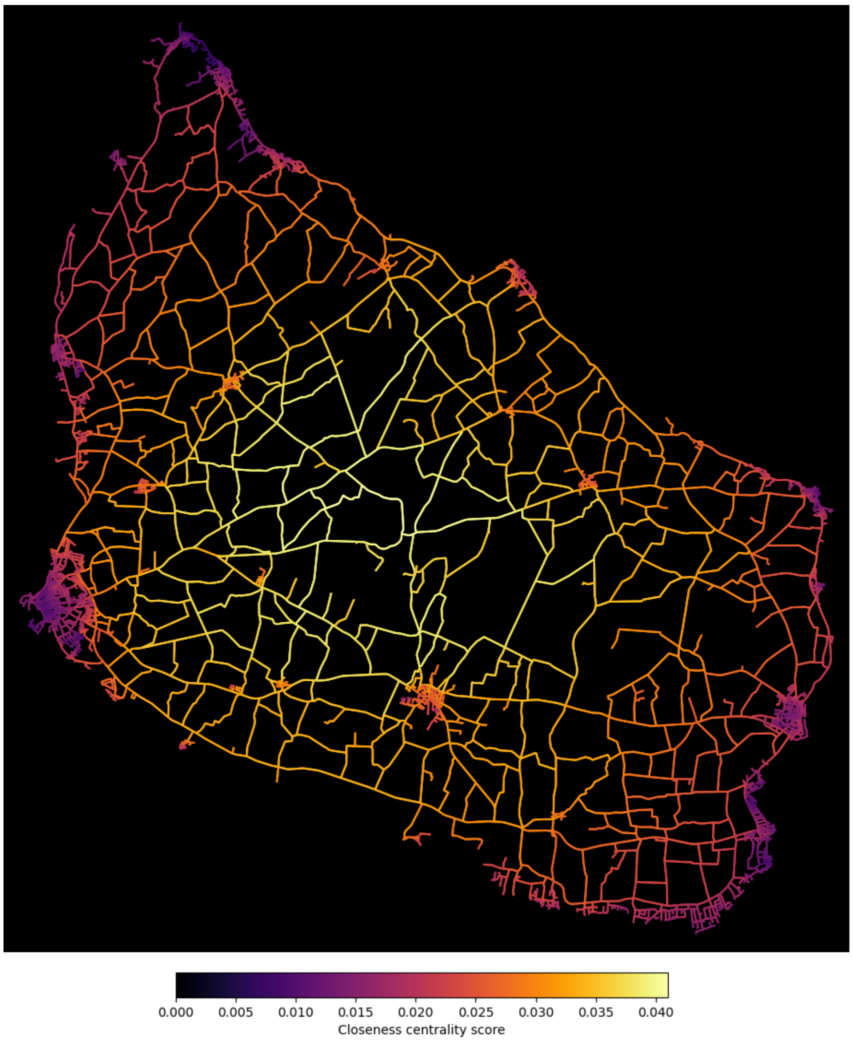
In [39] the k-means algorithm has been used with geospatial data, where the aim of the research was to better the traffic control of a city in Cyprus.

[47] uses a modified k-means algorithm to investigate and suggest an optimized public transportation network. A conclusion from the study underlines the synergy that appears when *"apply[ing] different methods (e.g. machine learning as well) for geospatial data analysis"* [47].

The k-means algorithm belongs to the: *"[...] family of algorithms [that] iteratively optimise a model (of the dataset under consideration) as K clusters. This cluster model comprises a membership element, assigning each member of the dataset to one of the clusters, and a centroid element, which describes each cluster"* [67]. Many variations exist of the k-means algorithm, whereas the most well-known is based on minimizing the distance d_{v,c_v} of a node v to its assigned cluster c.

In other words, it will cluster a data set into K clusters by assigning each data point to one of these.

Figure 2.3: Closeness centrality shown by edges for Bornholm



The main properties of a general k-means algorithm are demonstrated in Alg. 1 [30][35]. It alternates between assigning points to clusters and updating the centroids.

Algorithm 1 K-means Algorithm

```
Input
    $p_1, p_2, p_3, ..., p_n \in$ set of points $P$
    $C_1, C_2, C_3, ..., C_k \in$ set of clusters $C$
1: Choose random centers $c_1, c_2, c_3, ..., c_k$ for each cluster in $C$
2: repeat
3:     $C_1, C_2, C_3, ..., C_k \leftarrow \emptyset$
4:     for each $p \in P$ do
5:         Let i = $\text{argmin}_{i=1,2,3,...k}\ ||p - c_i||^2$
6:         $C_i \leftarrow C_i \cup \{p\}$
7:     for $i = 1$ to $k$ do
8:         if $P_i \neq \emptyset$ then $c_i = \frac{1}{|C_i|} \sum_{p \in C_i} p$
9: until centers do not change
```

As the pseudo-code in Alg. 1 is somewhat simplified, different parameters and additional optimizations should be explained.

First, one has to understand, that the result of each iteration of the algorithm is better than or equal to the clustering provided by the previous iteration. A parameter that therefore has a high correlation with both the running time and the correctness of the result is the number of iterations [27].

To test both the relation of iterations and the correctness of the resulting k-means clusters and their centroids and the libraries needed to conduct such experiments, a preliminary implementation is developed.

The k-means algorithm and random data generation function from `sklearn` are used for this investigation [22][19]. A data set consisting of 100 points randomly placed around a center is generated with a seed, S_{points}. Next, a list of centroids is initialized as `None`.

For the first iteration of the k-means clustering algorithm, three centroids are placed randomly within the distribution of the points in the data set using a seed, $S_{centroids}$. For the following iterations, the existing centroids are used as a basis for the local optimization procedure of the algorithm. For each iteration, the coordinates of the centroids are saved.

To explore the correlation of iterations and the correctness of the resulting k-means clustering, 1000 iterations of the algorithm are performed. This test is initialized with the same seeds, S_{points} and $S_{centroids}$, as the previous experiment.

Fig. 2.4a, Fig. 2.4b, Fig. 2.4c, and Fig. 2.4d, show the clustering and the centroids of the experiments after one, two, three, and 1000 iterations of the k-means algorithm on an identical data set and initial centroid placement using the same seeds. The results from the tests visualized in Fig. 2.4a, Fig. 2.4b, Fig. 2.4c, and Fig. 2.4d are shown in Tab. 2.3, in which both the coordinates of the centroids and the distance of cluster centroids to the final clustering centroids after 1000 iterations are recorded.

Table 2.3: Result of k-means test with various iterations

Cluster	**Iteration**	**Coordinates**$_{centroid}$	**Distance**$_{finalCentroid}$
A	1	(1.515, 2.643)	0.354
	2	(1.534, 2.369)	0.130
	3	(1.491, 2.285)	0.071
	1000	(1.423, 2.301)	
B	1	(0.116, 1.249)	0.471
	2	(0.206, 1.037)	0.252
	3	(0.273, 0.961)	0.152
	1000	(0.394, 0.869)	
C	1	(1.889, 0.874)	0.404
	2	(1.982, 0.712)	0.271
	3	(2.114, 0.683)	0.140
	1000	(2.253, 0.699)	

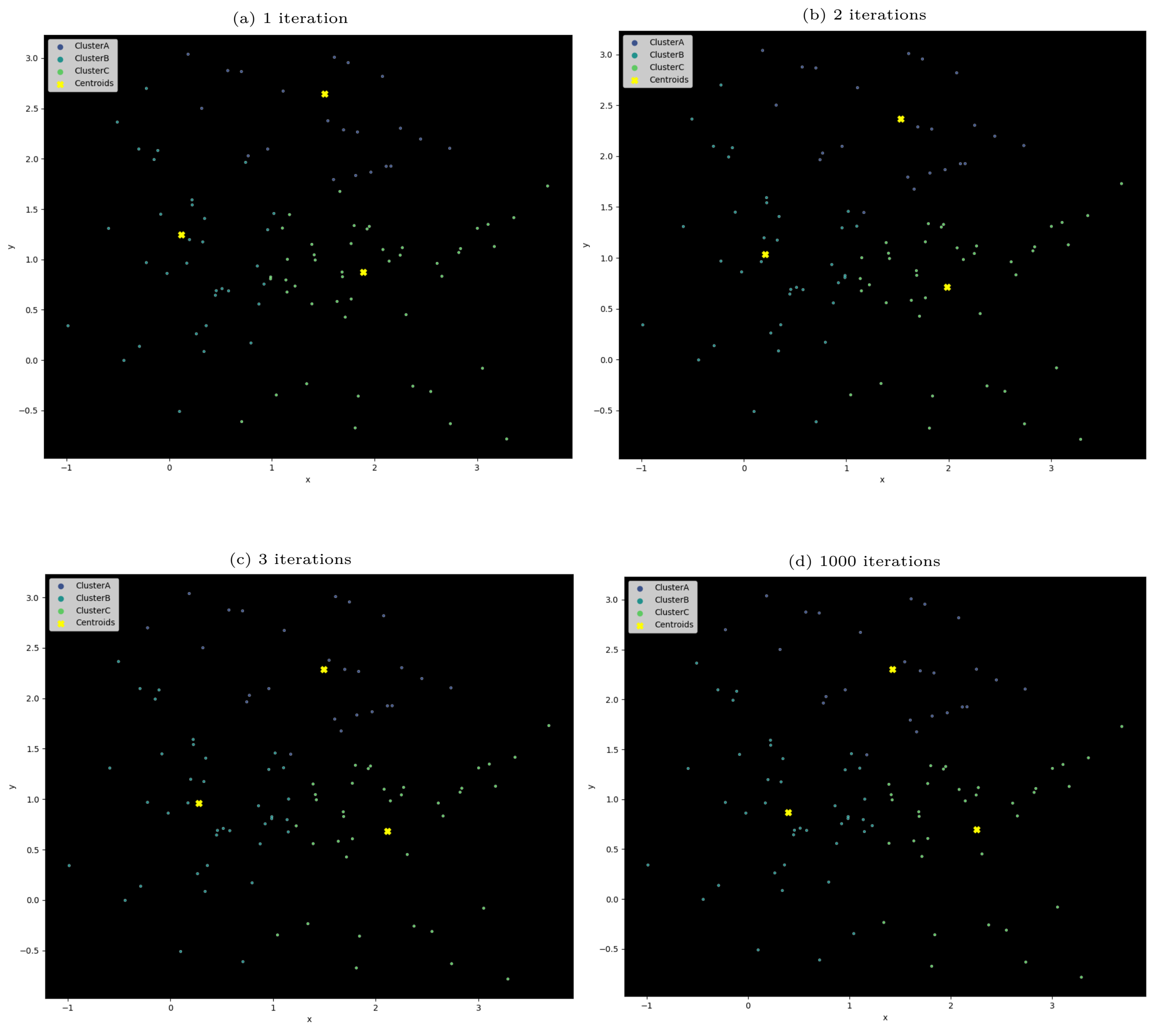
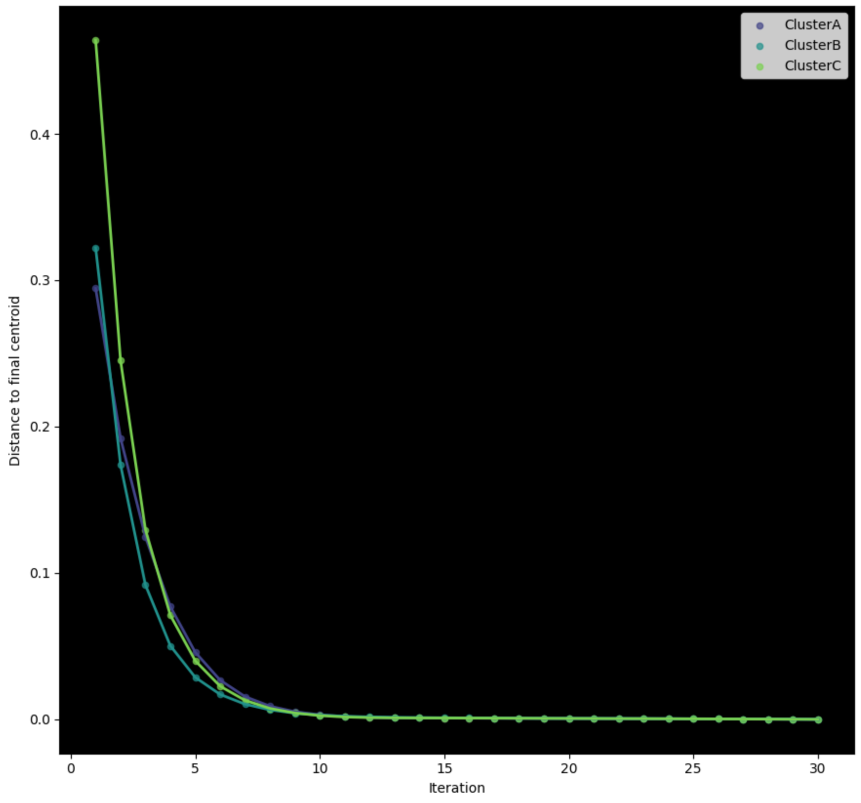


Figure 2.4: K-means clustering after various iterations

To further understand and visualize the correlation between the correctness of a k-means clustering and the number of executed iterations, an additional experiment is conducted.

For the test visualized in Fig. 2.5, 1.000.000 sample points are randomly generated, and three clusters are sought. In less than 15 iterations, the k-means algorithm has produced clusters with centroids that are only corrected by a little fraction for the following iterations.

Figure 2.5: Correlation between k-means clustering of 1.000.000 random sample points and the correctness



The low number of iterations needed for the conducted experiment in Fig. 2.5 has multiple causes. One is the initial positioning of the centroids, as also described in [29], [51], [74], and [34]. Another is the entropy of the randomly generated data set.

In general, the k-means algorithm has shortcomings. The running time is complex, as every iteration calculates the minimum distance for each point to a centroid and afterwards recalculates the position of the centroids. Much research has been made to optimize the k-means algorithm, for example [76] and [62]. Therefore many implementations of the k-means algorithm have several break conditions: stop the algorithm whenever the improvements are below a threshold, stop the algorithm if the centroids do not move, or even just a hardcoded maximum number of iterations.

Chapter 3

Methods and Implementation

This chapter and its sections will introduce and explain five different hedging strategies, from which the test results are reported later in chapter 4. The arenas used and the placement of targets within these are described in section 3.1 with code snippets that correspond to the actual implementation. From this point on, implementation details are described in a pseudo-code style. The first hedging strategy, presented in section 3.2, is of a random character as it randomly places guesses on the network. Section 3.3 also suggests a random approach, as the hedging strategy places guesses within the convex hull of the network. The hedging strategy developed in [49] is presented in section 3.4, and places its guesses based on an analysis of the network. Section 3.5 argues for and describes a hedging strategy based on closeness centrality. Finally, section 3.6 presents a hedging strategy based on a k-means clustering approach.

3.1 Arena and Target

3.1.1 Arenas: Bornholm and Malta

The tests will be conducted in two arenas, Bornholm and Malta. Appendix A and B show the maps of the arenas, whereas appendix D and E show the driveable geospatial networks. Furthermore, the tests are conducted with three guesses for each round, meaning that the team consists of three players.

The island of Bornholm is a Danish island in the Baltic Sea with a geospatial network mostly present in harbor cities [7]. Malta is a country in the Mediterranean Sea mainly consisting of three bigger islands Malta, Gozo, and Comino [9].

These test arenas were selected due to their small size, allowing fast computations, and due to the non-trivial shape of Malta which is split into three islands.

Both arenas are fetched using the `osmnx` library [18]. However, when fetching Malta by using the function `graph_from_place` only the main island is returned [16]. Furthermore, when using the function `graph_from_bbox`, only one of Malta's islands is returned [15]. Therefore Malta is fetched by its three main islands individually and composed using the function `compose_all` from the `networkx` library [13][12].

Due to optimization on time when debugging code and running tests, the networks are saved after being fetched. The code used for fetching, saving, and composing the networks is shown in Cod. 3.1.

Code snippet 3.1: Fetch, save, and load the geospatial networks for Bornholm and Malta

```python
import osmnx as ox
import networkx as nx

#Fetch and save Malta
gozo = ox.graph_from_bbox(north=36.0831, south=36.0036, east=14.3523,
    west=14.1825, network_type="drive")
comino = ox.graph_from_bbox(north=36.0002 ,south=35.8039, east=14.5782,
    west=14.3183, network_type="drive")
malta = ox.graph_from_bbox(north=36.0209, south=36.0036, east=14.3513,
    west=14.3162, network_type="drive")
ox.save_graphml(gozo, filepath="./savedGraphs/gozo.graphml")
ox.save_graphml(comino, filepath="./savedGraphs/comino.graphml")
ox.save_graphml(malta, filepath="./savedGraphs/malta.graphml")

#Fetch and save Bornholm
bornholm = ox.graph_from_place("Bornholm", network_type="drive")
ox.save_graphml(bornholm, filepath="./savedGraphs/bornholm.graphml")

#Load networks into networkx.MultiGraph. State the arena
arena = "malta" #can be either "malta" or "bornholm"
graph = nx.MultiGraph

if arena == "bornholm":
    graph = ox.load_graphml("./savedGraphs/bornholm.graphml")
else:
    gozo = ox.load_graphml("./savedGraphs/gozo.graphml")
    comino = ox.load_graphml("./savedGraphs/comino.graphml")
    malta = ox.load_graphml("./savedGraphs/malta.graphml")
    graph = nx.compose_all([gozo, comino, malta])
```

3.1.2 Placement of the Target

The target should be placed randomly in the driveable geospatial network of the arena. However, each node or edge cannot simply be chosen randomly correctly without weighing its actual representation. In other words, an edge between two nodes can be of arbitrary length, meaning that the random selection of some node in the network should account for this. Furthermore, the samples should be able to choose some interpolation of the edges rather than choosing just their endpoints (i.e., the network nodes).

To do exactly this, the targets are randomly created using the function `sample_points` from the `osmnx.utils_geo` library [17]. Working further with the source code shown in Cod. 3.1, retrieving a sample node as the target is done as shown in Cod. 3.2.

Code snippet 3.2: Retrieve sample node as the target

```
from osmnx.utils_geo import sample_points

#Translate the graph into geospatial network with latitude and longitude
graph.graph['crs'] = "epsg:3857"

#Get the coordinates from a geopandas.GeoSeries object
def retrieveLatAndLong(geo):
    return (geo.x, geo.y)

#Retrieve sample node as target
target = retrieveLatAndLong(sample_points(graph, 1))
```

3.2 Random Guess in Network

3.2.1 Motivation

The hedging strategy *Random Guess in Network* has two contributions to the exploratory part of this thesis and should therefore be tested.

First, it can be regarded as a measure of, how well or badly the other non-random hedging strategies are performing compared to this random hedging strategy. Thereby it produces a point of reference, or *null model*, that defines whether or not a non-random strategy has a rewarding effect.

Second, the strategy itself can be argued for, as the guesses it produces follows the distribution of the geospatial network if implemented correctly. Imagine a small circular island, where 80% of the network exists in and around a harbor city. The remaining 20% of the driveable geospatial network is on the opposite shore of the island in and around a smaller harbor city. Statistically, the target and each guess would have the same probability of being placed in either the bigger city or the smaller city, respectively 20% and 80%. To some extent of intuition, this is not a bad hedging strategy for all arenas.

3.2.2 Implementation

The implementation of this random hedging strategy requires two steps. First, three random guesses have to be generated somewhere in the network of the arena. This is done by using the same function as in section 3.1.2, `sample_points` [17].

Second, the distances from each sampled node should be calculated to the target. This is done by using the Haversine formula that calculates the distance between two nodes in a geospatial network [65]. The implementation is shown in Alg. 2 and a visualization of the algorithm is shown in Fig. 3.1 and Fig. 3.2.

Algorithm 2 Hedging Strategy: Random Guess in Network

```
Input
    $g_1, g_2, g_3 \in$ set of guesses $G$ within $arena_{network}$
    $t$ the target within $arena_{network}$
1: Let best guess so far, $g_{min} \leftarrow null$
2: Let minimum distance so far, $dist_{min} \leftarrow int_{max}$
3: for each $g \in G$ do
4:     Let $dist = haversine(g, t)$
5:     if $dist < g_{min}$ then $g_{min} = g, dist_{min} = dist$
6: return $g_{min}, dist_{min}$
```

Figure 3.1: Random Guess in Network - Bornholm

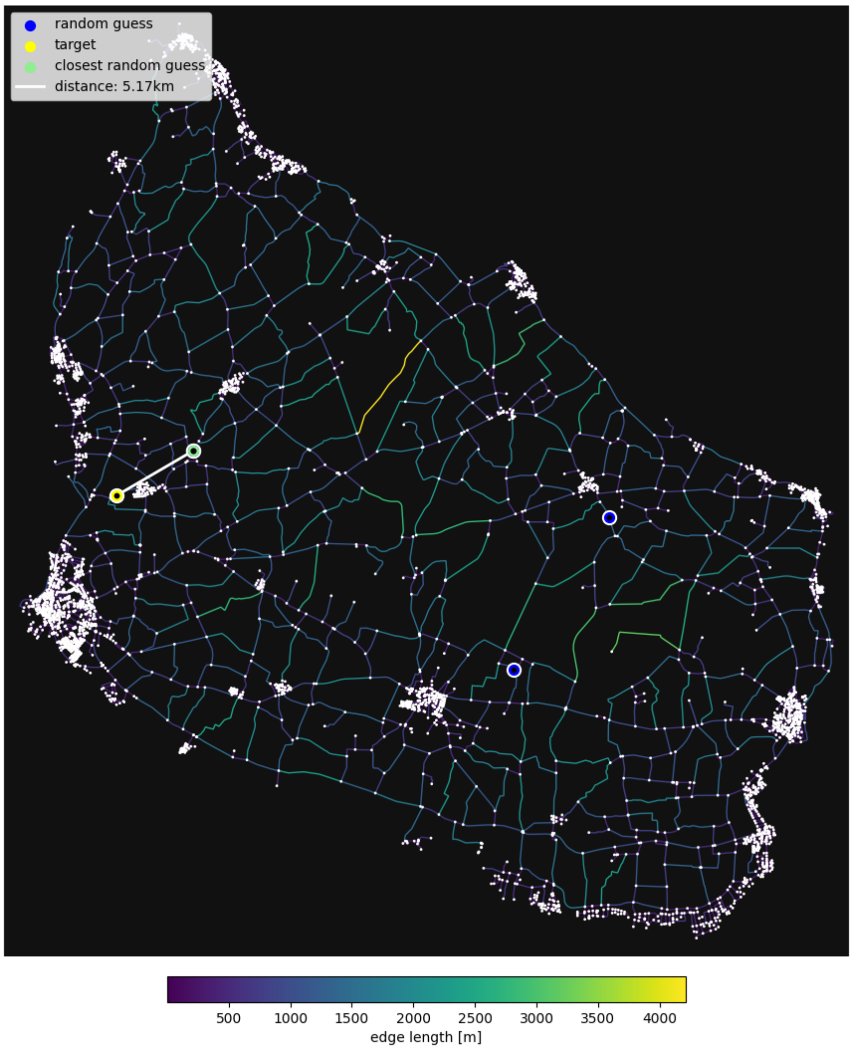
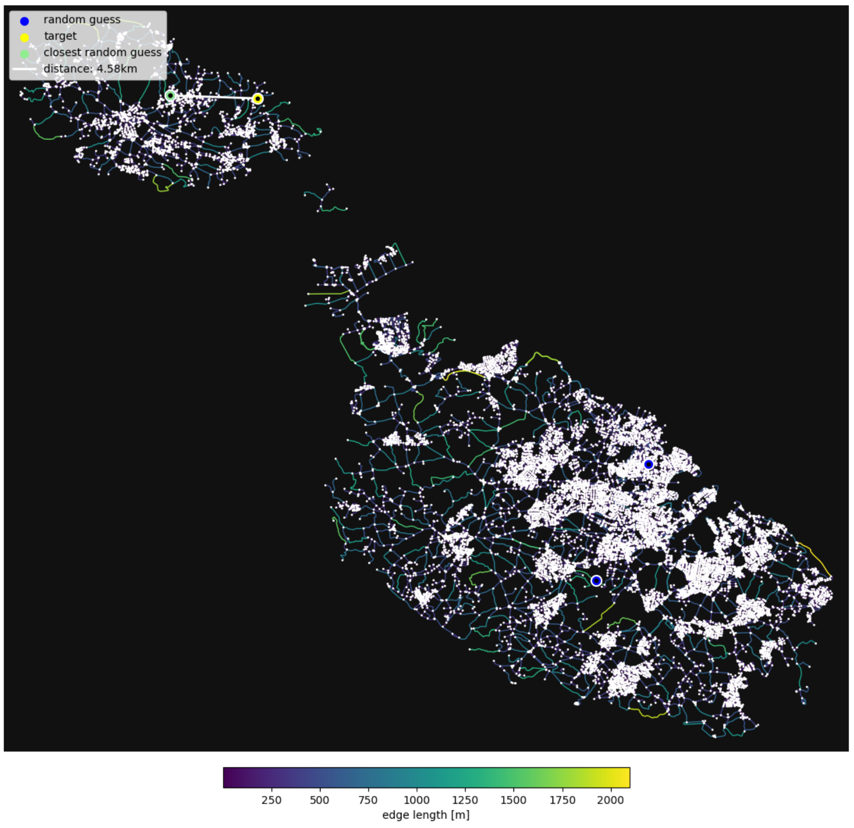


Figure 3.2: Random Guess in Network - Malta



3.3 Random Guess in Convex Hull

3.3.1 Motivation

Like the hedging strategy *Random Guess in Network* places guesses randomly, also the hedging strategy *Random Guess in Convex Hull* employs a random approach for guessing. However, *Random Guess in Convex Hull* places guesses randomly distributed within the convex hull of the network instead. The convex hull of a network is the minimum convex polygon that includes all the nodes [32].

Besides being another random approach that can be used as a measure for identifying more optimal hedging strategies, also *Random Guess in Convex Hull* can be argued for having a potential value in certain arenas.

Imagine an arena consisting of two identically shaped circular islands, $island_A$ and $island_B$. Both $island_A$ and $island_B$ have 50% of the total network of the arena evenly distributed. The distance between the two islands from the nearest shore of each respectively is equal to the radius of both $island_A$ and $island_B$.

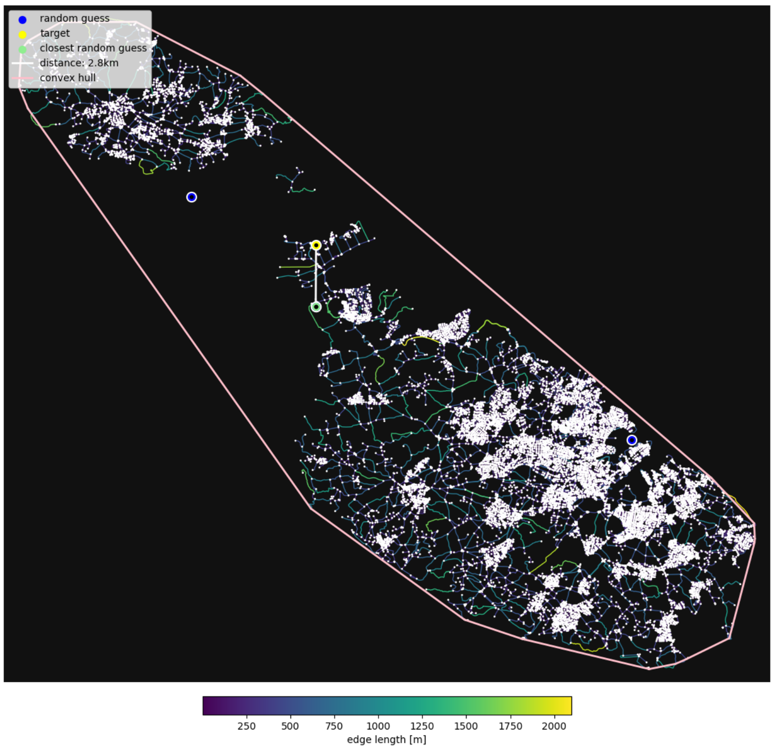
Using the other random hedging strategy, *Random Guess in Network*, would place all guesses on the islands as only they have a driveable network. However, intuitively placing one of the three guesses in the middle of the sea that is dividing the two islands would be a viable hedging of the arena.

3.3.2 Implementation

The implementation of *Random Guess in Convex Hull* requires some preliminary computations. The three randomly generated guesses should be within the convex hull of the arena. Therefore, the convex hull of the arena should be computed initially to determine whether or not a guess is within the boundaries. For generating the convex hull polygon, the function `convex_hull` is used by GeoPandas [5][4].

After determining the convex hull and generating three random guesses within it, the forthgoing procedure is very similar to Alg. 2. The implementation of *Random Guess in Convex Hull* is shown in Alg. 3 and a visualization of the algorithm is shown in Fig. 3.4 and Fig. 3.3.

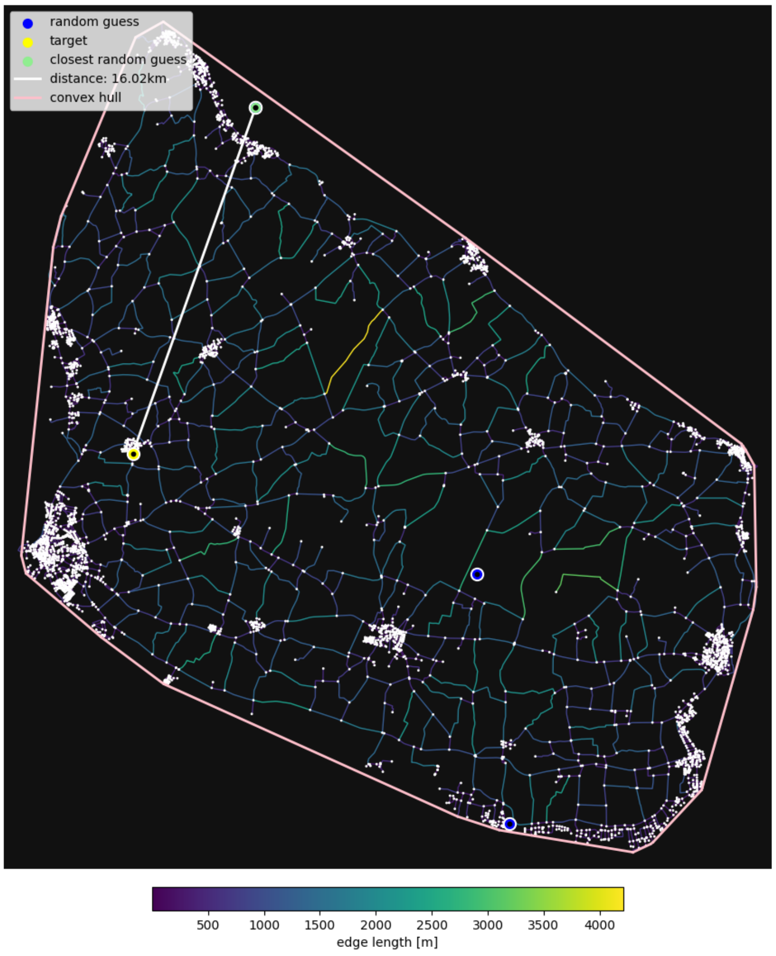
Figure 3.3: Random Guess in Convex Hull - Malta



Algorithm 3 Hedging Strategy: Random Guess in Convex Hull

```
Input
    $g_1, g_2, g_3 \in$ set of guesses $G$ within $arena_{convexHull}$
    $t$ the target within $arena_{network}$
1: Let best guess so far, $g_{min} \leftarrow null$
2: Let minimum distance so far, $dist_{min} \leftarrow int_{max}$
3: for each $g \in G$ do
4:     Let $dist = haversine(g, t)$
5:     if $dist < g_{min}$ then $g_{min} = g, dist_{min} = dist$
6: return $g_{min}, dist_{min}$
```

Figure 3.4: Random Guess in Convex Hull - Bornholm



3.4 Divide and Conquer

3.4.1 Motivation

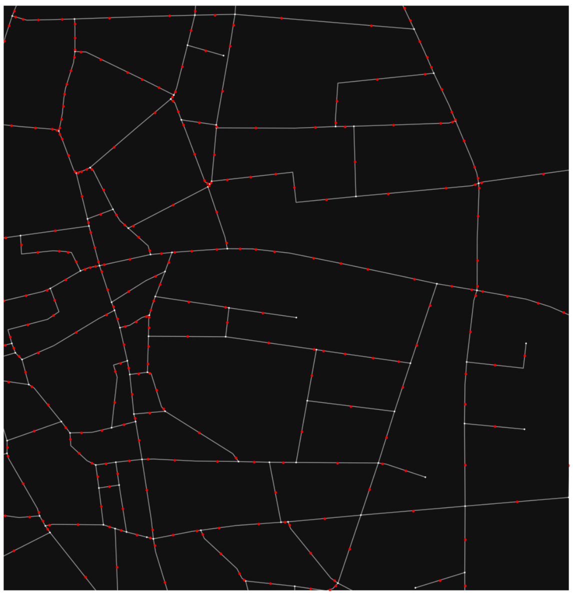
A hedging strategy called *Divide and Conquer* is described in [49], that places its guesses based on an analysis of the network. Instead of placing guesses randomly, it divides the network into n equal subnetworks, where n is the number of guesses. In each subnetwork, the midpoint is calculated and eventually used as a guess.

This hedging strategy intuitively has strengths that the random hedging strategies do not possess, as it places its guesses based on a data-driven rationale. However, as reported in [49], the applied graph traversal together with the use of midpoint as guesses are primitive solutions, as *"most of the bigger cities lie near the coast [of Bornholm]"* [49], and thereby simplify aspects of the challenge.

3.4.2 Implementation

Several data translations and computations are required in the hedging strategy *Divide and Conquer*. First, the network has to be abstracted into a set of points that are generated in the center of each edge. The points should, besides holding the values of latitude and longitude, also contain information on their weight, which is equal to the length each center point represents. This abstraction is shown in Fig. 3.5.

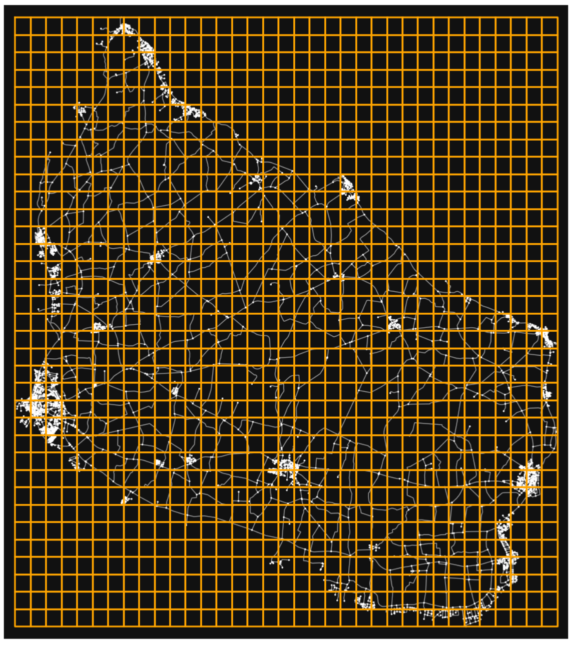
Figure 3.5: Center points of edges



The next step requires dividing the network into a grid and assigning each cell of the grid all of the center points, that are within it. The configuration in the implementation for the tests is set to $(35, 35)$, meaning that both the height and the width of the grid are split into 35 cells.

This subprocedure of the algorithm is shown in Fig. 3.6a and Fig. 3.6b for Bornholm and Malta respectively.

(a) Bornholm



(b) Malta

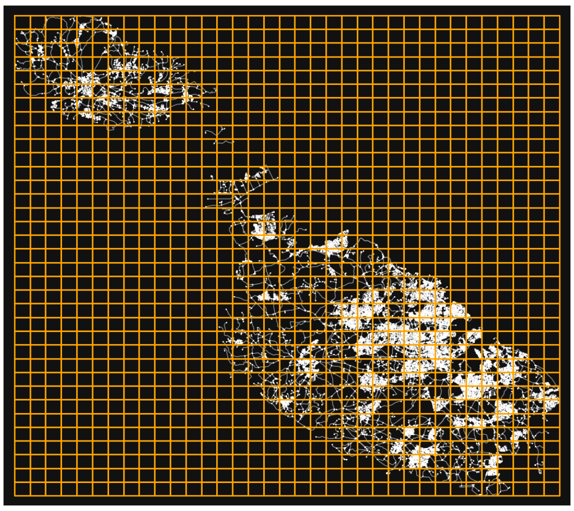


Figure 3.6: Grid division of the arenas

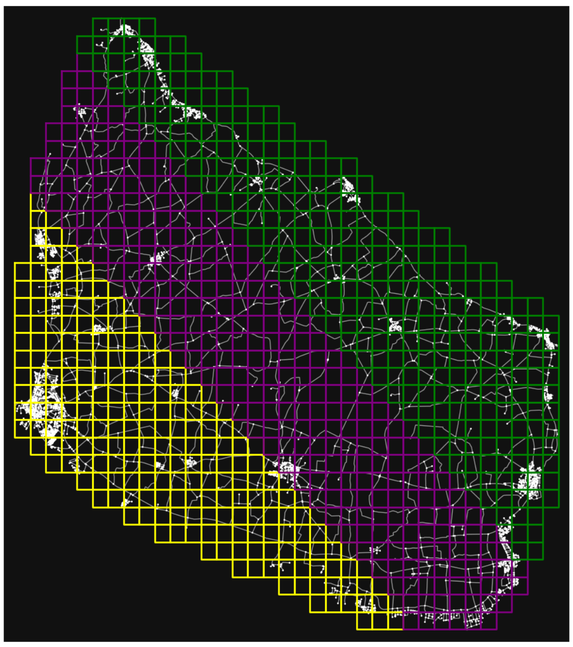
Now all the weighted points are iterated through to place each in one of the cells of the grid. By this point, all the grid cells contain both the weighted points and the total sum of their weights.

A traversal is now executed, that diagonally runs through all cells and associates each cell with a bigger area. There should be n bigger areas, where n is the number of guesses. During the traversal, a cell gets associated with a bigger area, $area_x$ if the cell is within the convex hull of the arena and the bigger area's current total weight is not greater or equal to the total weight of the network divided by n, as the following assertion must hold: $weight_{area} \leq \frac{weight_{total}}{n}$.

As described in [49], the diagonal direction of the traversal seemed intuitively the worst of the two possible directions, as the direction followed the shape of Bornholm. However, this claim was not supported by tests. Therefore both diagonal traversal directions will be tested in this thesis.

Fig. 3.7a and Fig. 3.7b show the resulting bigger areas when traversing with the two diagonal directions in the arena of Bornholm, whereas Fig. 3.8a and Fig. 3.8b show the resulting bigger areas when traversing with the two diagonal directions in the arena of Malta.

(a) Diagonal direction A



(b) Diagonal direction B

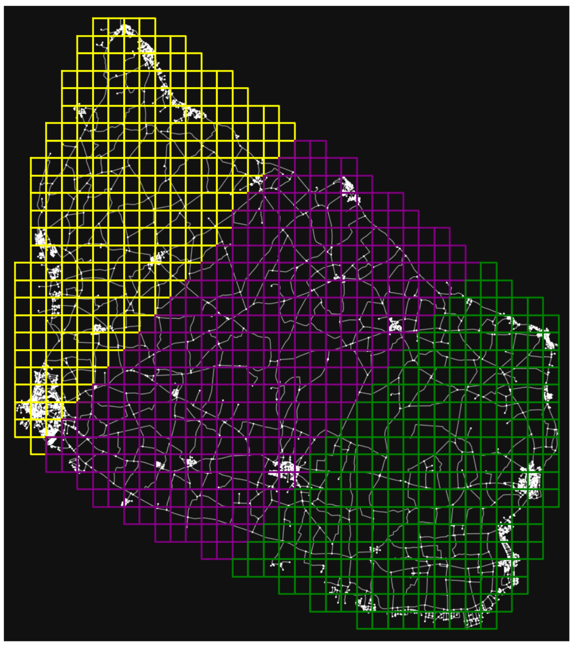


Figure 3.7: The n bigger areas of Bornholm

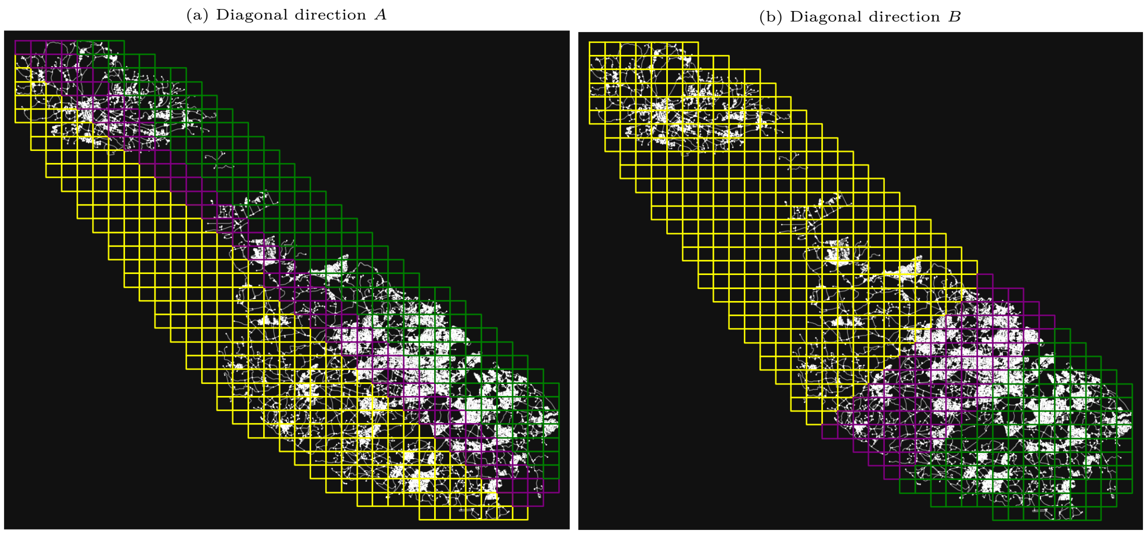
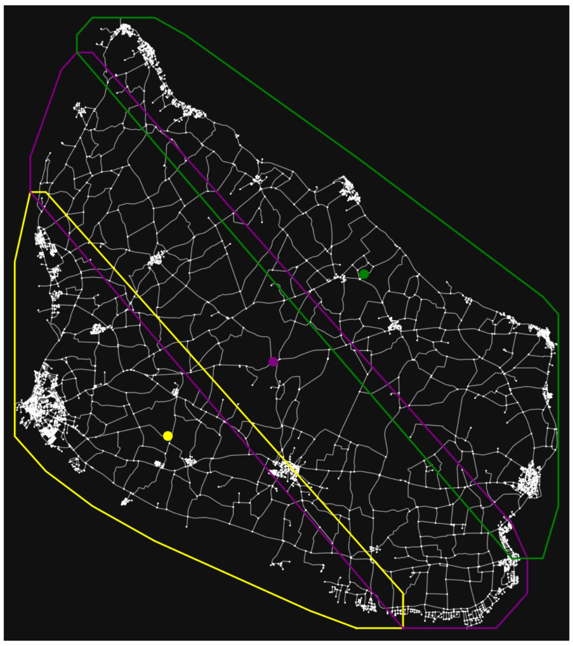


Figure 3.8: The n bigger areas of Malta

The next and final subprocedure of the hedging strategy explored in [49] defines the guesses after three steps. First, all the corner points of the grid cells of the n bigger areas are added to n collections. Second, the convex hulls of these n collections are now generated using the function `convex_hull` [4]. Third, the *geometric centroids*, which in this case is equivalent to the midpoint, is calculated using the function `centroid` from GeoPandas [3]. These n geometric centroids are now regarded as the n guesses of the *Divide and Conquer* hedging strategy.

Fig. 3.9a and Fig. 3.9b show the resulting convex hulls and geometric centroids when traversing with the two diagonal directions in the arena of Bornholm, whereas Fig. 3.10a and Fig. 3.10b show the resulting convex hulls and geometric centroids when traversing with the two diagonal directions in the arena of Malta. The pseudo-code for the *Divide and Conquer* hedging strategy is shown in Alg. 4.

(a) Diagonal direction A



(b) Diagonal direction B

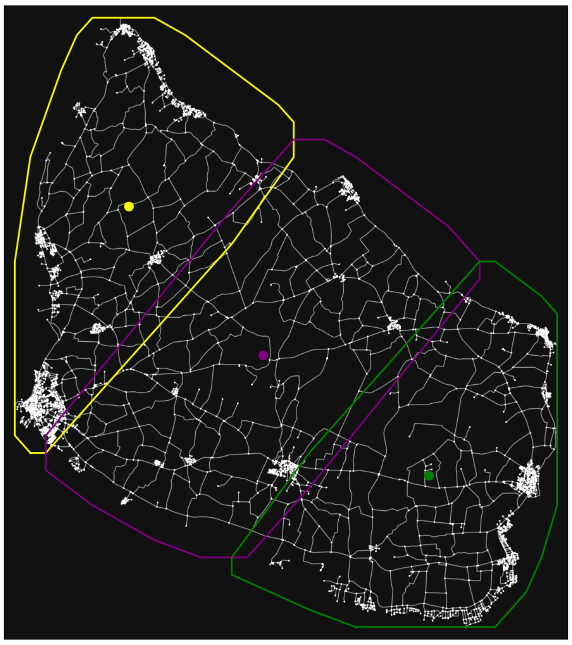
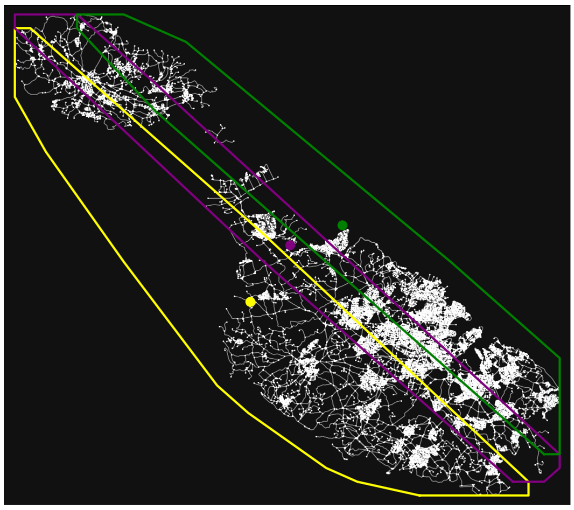


Figure 3.9: The n convex hulls and geometric centroids of Bornholm

(a) Diagonal direction A



(b) Diagonal direction B

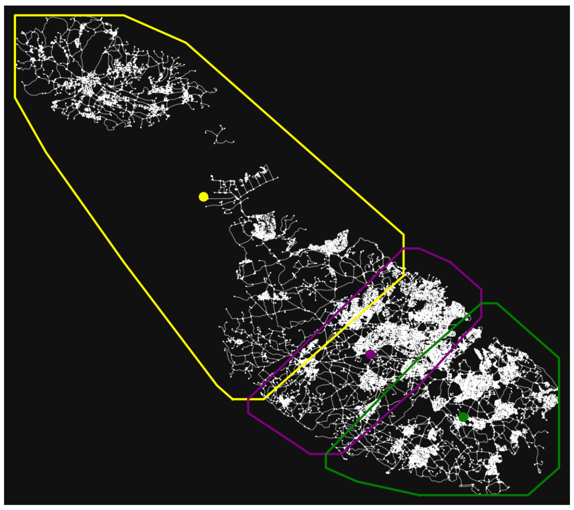


Figure 3.10: The n convex hulls and geometric centroids of Malta

Algorithm 4 Hedging Strategy: Divide and Conquer

```
Input
    The target, $t$, within $arena_{network}$
    The convex hull of the network, $convex_{hull}$
    The list of weighted points, $P_{weighted}$
    The collection of empty grid cells, $G$
    The list of empty bigger areas, $A$
    The total weight of the network, $weight_{total}$
 1: Let best guess so far, $g_{min} \leftarrow null$
 2: Let minimum distance so far, $dist_{min} \leftarrow int_{max}$
 3: An integer to determine the current index starting at 0, $i$
 4: for each $p \in P_{weighted}$ do
 5:     Find the cell $c$ of $G$ that includes $p$
 6:     Add $p$ to $c$ and add the weight of $p$ to the total weight of $c$
 7: for each $c \in G$ do (diagonal traverse)
 8:     if $c \in convex_{hull}$ then
 9:         if $weight(A[i]) < \frac{weight_{total}}{n})$ then $A[i] = A[i] \cup c$
10:         else $i++$, $A[i] = A[i] \cup c$
11: for each $a \in A$ do
12:     Let $g = geometric_{centroid}(a)$
13:     Let $dist = haversine(g, t)$
14:     if $dist < g_{min}$ then $g_{min} = g, dist_{min} = dist$
15: return $g_{min}, dist_{min}$
```

3.5 Closeness Centrality

3.5.1 Motivation

As mentioned in section 2.3.2, some challenges emerge when using closeness centrality to define a hedging strategy, caused by the risk of clustering of the most central nodes in the network. However, using closeness centrality within a subnetwork seems like a valid approach. As for the hedging strategy *Random Guess in Network*, this hedging strategy will therefore also place its guesses on the network of the arena.

In the hedging strategy *Divide and Conquer*, the final guess within the n resulting divisions of the arena is equal to the geometric centroid of the convex hull depicting the division. This hedging strategy, *Closeness Centrality*, will use the measure of closeness centrality to further adjust the guesses within these n convex hulls.

As the results in section 4.3 show when traversing with the two different diagonal directions in *Divide and Conquer*, one direction is found to outperform the others. Therefore, direction B is employed for this hedging strategy.

3.5.2 Implementation

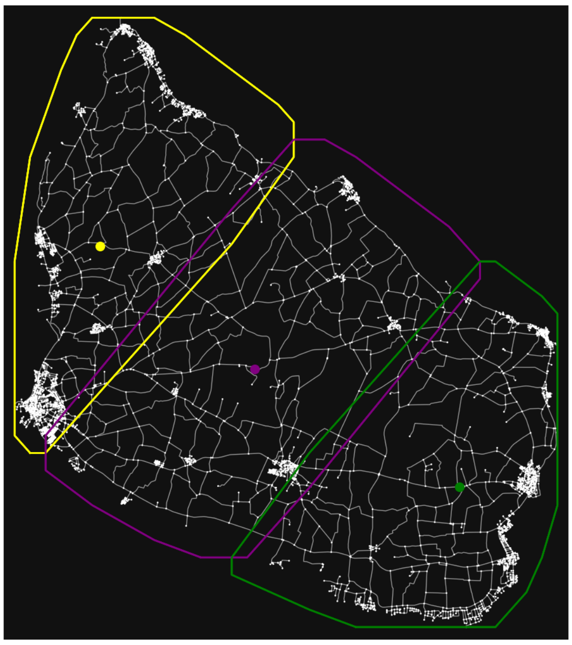
Many computational steps of this hedging strategy are identical to the ones of *Divide and Conquer*. Until the point when *Divide and Conquer*, as shown in Alg. 4 on line 12, uses the geometric centroids as guesses, the two hedging strategies are alike.

At this step, the *Closeness Centrality* hedging strategy calculates the node with the highest closeness centrality within each subnetwork. However, three problems occur when using the `closeness_centrality` function from `networkx` [11]. First, it requires an actual network, which is difficult to fetch from `osmnx` given the difficulties described in 3.1. Second, it uses as the measure of the distance the shortest path from each node to all others, where the measure of distance preferably should be the haversine. Third, the implementation should account for the weight a node represents.

Therefore, the closeness score is calculated with a custom function, that sums all haversine distances from each node to all others within every subnetwork. The nodes used for this calculation are abstractions of edges, given that they represent the centers and lengths. Given a time complexity of $O(N^2)$ for the distance calculations, the developed implementation makes it possible to simplify the number of center points by clustering neighbors.

Fig. 3.11a and Fig. 3.11b show the resulting convex hulls and the nodes within these with the highest score of closeness centrality. The pseudo-code for the *Closeness Centrality* hedging strategy is shown in Alg. 5.

(a) Bornholm



(b) Malta

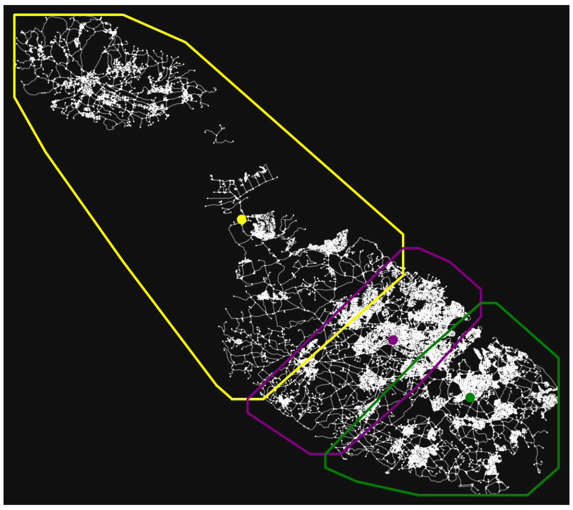


Figure 3.11: The n convex hulls and their nodes with the highest score of closeness centrality

Algorithm 5 Hedging Strategy: Closeness Centrality

```
Input
    The target, $t$, within $arena_{network}$
    The convex hull of the network, $convex_{hull}$
    The list of weighted points, $P_{weighted}$
    The collection of empty grid cells, $G$
    The list of empty bigger areas, $A$
    The total weight of the network, $weight_{total}$
1: Let best guess so far, $g_{min} \leftarrow null$
2: Let minimum distance so far, $dist_{min} \leftarrow int_{max}$
3: An integer to determine the current index starting at 0, $i$
4: for each $p \in P_{weighted}$ do
5:     Find the cell $c$ of $G$ that includes $p$
6:     Add $p$ to $c$ and add the weight of $p$ to the total weight of $c$
7: for each $c \in G$ do (diagonal traverse)
8:     if $c \in convex_{hull}$ then
9:         if $weight(A[i]) < \frac{weight_{total}}{n})$ then $A[i] = A[i] \cup c$
10:        else $i++$, $A[i] = A[i] \cup c$
11: Let $CC_{min} = []$
12: for each $a \in A$ do
13:     Let $CC = empty\ map$
14:     $N_a \leftarrow$ the weighted center points within $a$
15:     for each $n \in N_a$ do
16:         Let $CC[n] = 0$
17:         for each $m \in N_a$ do
18:             $CC[n] = CC[n] + haversine(n, m) \cdot weight_m$
19:     Append $min(CC)$ to $CC_{min}$
20: for each $g \in CC_{min}$ do
21:     Let $dist = haversine(g, t)$
22:     if $dist < g_{min}$ then $g_{min} = g, dist_{min} = dist$
23: return $g_{min}, dist_{min}$
```

3.6 K-Means Clustering

3.6.1 Motivation

As described in 2.3.3, the k-means clustering algorithm is a procedure for clustering entries in a data set, in which each iteration will either optimize the current clustering or terminate the process.

For the application of the k-means clustering algorithm within the domain of geospatial hedging, both opportunities and challenges arise. Unlike the implemented non-random hedging strategies so far - *Divide and Conquer* and *Closeness Centrality* -

this approach is not bound to a traversal of the network for dividing it. The k-means algorithm will cluster the network as this is a fundamental feature of the algorithm, whereas both *Divide and Conquer* and *Closeness Centrality* rely on a diagonal division of the network. Furthermore, whereas the other non-random hedging strategies divide the network into equal-weighted parts, the k-means algorithm will optimize the clustering natively.

Imagine an arena consisting of three equally weighted and evenly distributed islands in the shape of an equilateral triangle. One island is in the south, whereas the two other islands are located respectively northwest and northeast of it. This arena will challenge the diagonal traversal technique, meaning both directions A and B. The *Divide and Conquer* is challenged, as two guesses would be placed in the middle of the ocean between the same two islands, i_A and i_B, whereas the last guess would be placed in the middle of the third island, i_C. The *Closeness Centrality* hedging strategy would place two guesses on the shores of i_A and i_B, either both guesses on one island or one guess on each island, whereas the last guess would be placed in a central node of the last island, i_C.

The k-means algorithm would recognize by itself, that each island should be considered a cluster. Hence, the k-means algorithm intuitively seems like a versatile and promising approach.

3.6.2 Implementation

The implementation of the *K-Means Clustering* hedging strategy starts with a dilemma, as two approaches are considered.

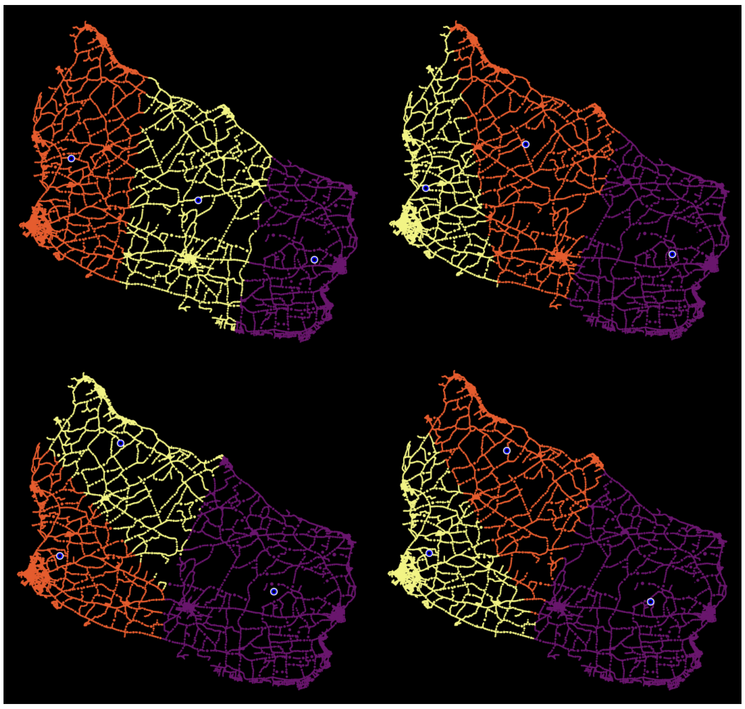
One approach builds on a custom build k-means algorithm developed for spatial applications, whereas the other approach utilizes an existing function `KMeans` from the `sklearn` library [22].

A third approach using `DBSCAN` was also considered [20]. However, it does not allow to predefine the number of clusters and was therefore dismissed.

The custom build implementation is more correct in terms of distance measures, as it uses the haversine function to determine the distance from nodes of a cluster to its centroid.

However, as visualized in Fig. 3.12, the initial selection of centroids is essential for the resulting clustering. Fig. 3.12 shows four different stable possible clusterings of Bornholm, that are achievable with different initial randomly chosen centroids.

Figure 3.12: Four different stable clusterings of Bornholm with the custom build k-means algorithm



Testing the four different clusterings visualized in Fig. 3.12 with 100.000 iterations shows, that they are not equally good hedging strategies.

Tab. 3.1 summarizes the test results from which can be concluded, that the clustering in the top left corner of Fig. 3.12 is superior to the others.

Table 3.1: Results of the four different clusterings of Bornholm resulting from the custom k-means implementation with 100.000 iterations

Position	**Measure**	**All [km]**	**Winning [km]**
top left	Mean	20.057	7.517
	σ	12.083	2.964
top right	Mean	20.192	7.783
	σ	12.096	3.478
bottom left	Mean	20.214	8.070
	σ	11.695	3.629
bottom right	Mean	20.143	7.989
	σ	11.657	3.589

The other approach utilizes the `KMeans` function from the `sklearn` library [22]. This function does not support distances to be calculated with other functions than the Euclidean distance formula. This means, that the resulting clusterings are not entirely correct.

However, the `KMeans` function supports multiple other parameters. Besides choosing the number of clusters and the maximum iteration count, `sklearn` offers the option to select an initialization parameter called `init` [22]. When passing the parameter value *k-means++* instead of *random*, the algorithm *"selects initial cluster centroids us-*

ing sampling based on an empirical probability distribution of the points' contribution to the overall inertia" [22]. When selecting this parameter value, the resulting clusterings have been observed to be near identical after each execution of the clustering algorithm.

Furthermore, running the k-means algorithm with the previously mentioned parameter on the arena of Bornholm has shown to result in a clustering very similar to the best-performing clustering documented in Tab. 3.1. After considering the trade-offs for each implementation, the choice falls on using `sklearns`'s function `KMeans`.

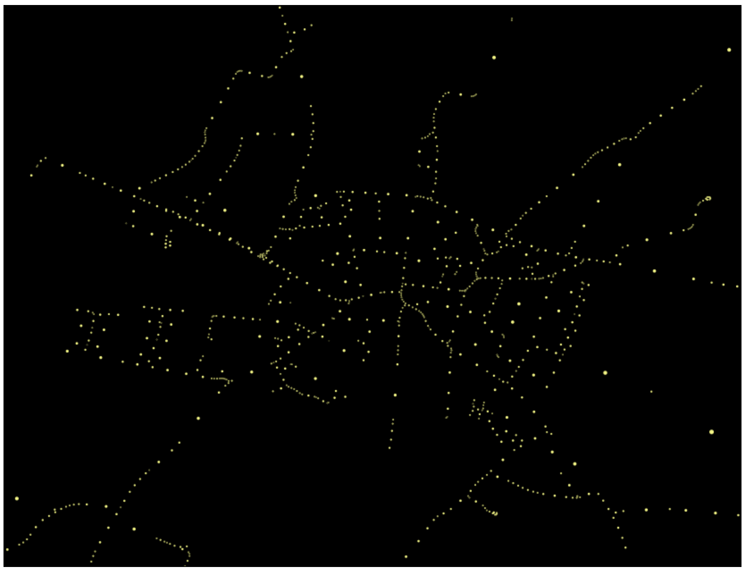
The implementation starts with, as also shown in Alg. 4 and Alg. 5, a computation of the weighted center points of all edges in the network of the arena.

Followed by this computation comes the initialization of `KMeans` [22]. It is given as parameters the number of clusters, the method for initial clustering selection, and the maximum number of iterations: `[3, 'k-means++', 1000]`.

Now the weighted center points can be fitted to the k-means algorithm configuration by using the `fit` function [21]. The function is given both the tuples consisting of the latitude and longitude of the weighted center points and their weights.

To visualize and confirm that the weight of each center point is correctly computed, Fig. 3.13 is created. It shows an inspection of a part of the arena's clusters and the different sizes of the nodes that correspond to their respective weights.

Figure 3.13: An inspection of the nodes of a cluster showing their sizes correctly corresponding to their weights



Now that the data set consisting of the weighted center points has been fitted, the cluster's centroids and their corresponding nodes can be retrieved. The centroids of the clusters will be the guesses.

Fig. 3.14a and Fig. 3.14b show the resulting clusters and their centroids of Bornholm and Malta. Alg. 6 shows the pseudo-code of the hedging strategy's algorithm.

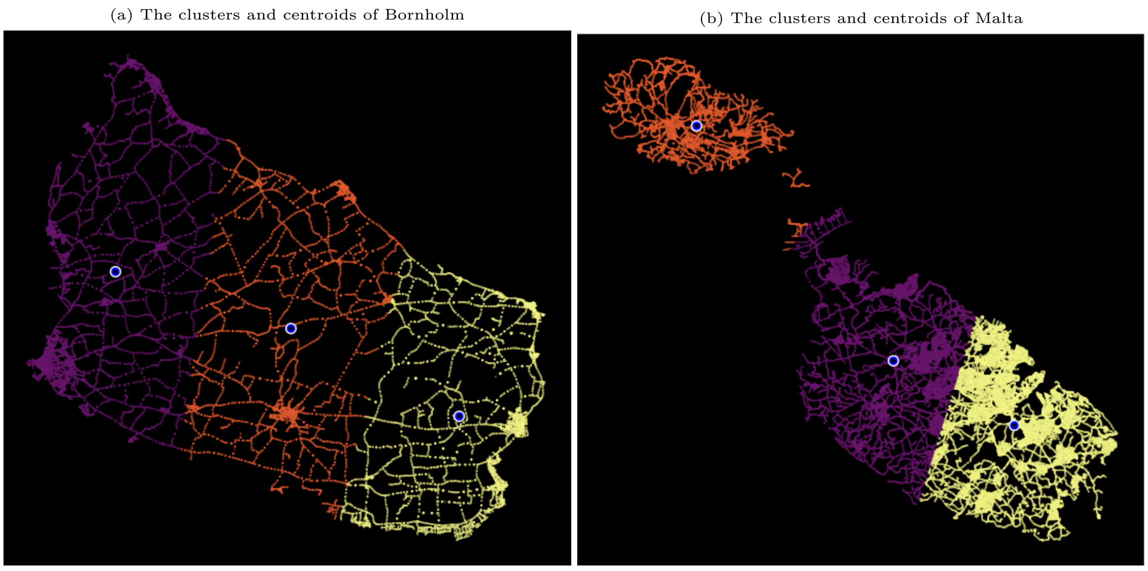

(a) The clusters and centroids of Bornholm
(b) The clusters and centroids of Malta

Figure 3.14: The clusters and centroids of the arenas when using the hedging strategy *K-Means Clustering*

Algorithm 6 Hedging Strategy: K-Means Clustering

```
Input
    The target, $t$, within $arena_{network}$
    The list of weighted points, $P_{weighted}$
1:  Let best guess so far, $g_{min} \leftarrow null$
2:  Let minimum distance so far, $dist_{min} \leftarrow int_{max}$
3:  Let $N$ be the nodes of $P_{weighted}$
4:  Let $W$ be the weights of $P_{weighted}$
5:  Let k-means, $k\_means \leftarrow KMeans(3, \text{'k-means++'}, 1000)$
6:  Let the weighted fit, $k\_means_{fit} \leftarrow k\_means.fit(N, W)$
7:  Let $C$ be the centroids of $k\_means_{fit}$
8:  for each $g \in C$ do
9:      Let $dist = haversine(g, t)$
10:     if $dist < g_{min}$ then $g_{min} = g, dist_{min} = dist$
11: return $g_{min}, dist_{min}$
```

Chapter 4

Results

This chapter will show and summarize the results of the five different hedging strategies presented and discussed in sections 3.2-3.6. The results for the individual hedging strategies are presented in sections 4.1-4.5, whereas an overview of the results of all hedging strategies is offered in section 4.6.

4.1 Random Guess in Network

4.1.1 Results

The results for the hedging strategy *Random Guess in Network* are shown in Tab. 4.1. Both the arenas - Bornholm and Malta - have been subjects to the test.

Tests have been conducted with three different numbers of iterations (1.000, 10.000, and 100.000) and both the means and standard deviations were measured and documented for all guesses and the winning guesses.

Table 4.1: Results of hedging strategy *Random Guess in Network* with various iterations

Iterations	**Arena**	**Measure**	**All [km]**	**Winning [km]**
1.000	Bornholm	Mean	21.029	11.608
		σ	11.804	7.393
	Malta	Mean	13.887	7.508
		σ	10.344	6.551
10.000	Bornholm	Mean	20.921	11.553
		σ	11.887	7.392
	Malta	Mean	13.661	7.346
		σ	10.141	6.362
100.000	Bornholm	Mean	20.934	11.572
		σ	11.855	7.420
	Malta	Mean	13.684	7.298
		σ	10.215	6.357

4.1.2 Summary

The test results for the hedging strategy *Random Guess in Network* with 100.000 iterations show, that the mean distance to the target node for all guesses on the arena of Bornholm is 20.934km with a standard deviation of 11.855km. On the arena of

Malta, the mean distance to the target node for all guesses is 13.684km with a standard deviation of 10.215km.

For the winning guesses on Bornholm, the mean distance to the target node is 11.572km with a standard deviation of 7.420km. On the arena of Malta, the mean distance to the target node for the winning guesses is 7.298km with a standard deviation of 6.357km.

4.2 Random Guess in Convex Hull

4.2.1 Results

The results for the hedging strategy *Random Guess in Convex Hull* are shown in Tab. 4.2. Both the arenas - Bornholm and Malta - have been subjects to the test.

Tests have been conducted with three different numbers of iterations (1.000, 10.000, and 100.000) and both the means and standard deviations were measured and documented for all guesses and the winning guesses.

Table 4.2: Results of hedging strategy *Random Guess in Convex Hull* with various iterations

Iterations	**Arena**	**Measure**	**All [km]**	**Winning [km]**
1.000	Bornholm	Mean	20.268	11.573
		σ	11.222	6.922
	Malta	Mean	15.893	8.393
		σ	9.976	5.861
10.000	Bornholm	Mean	20.037	11.495
		σ	10.977	6.911
	Malta	Mean	15.655	8.164
		σ	9.784	5.520
100.000	Bornholm	Mean	20.012	11.425
		σ	11.062	6.963
	Malta	Mean	15.740	8.238
		σ	9.800	5.623

4.2.2 Summary

The test results for the hedging strategy *Random Guess in Convex Hull* with 100.000 iterations show, that the mean distance to the target node for all guesses on the arena of Bornholm is 20.012km with a standard deviation of 11.602km. On the arena of Malta, the mean distance to the target node for all guesses is 15.740km with a standard deviation of 9.800km.

For the winning guesses on Bornholm, the mean distance to the target node is 11.425km with a standard deviation of 6.963km. On the arena of Malta, the mean distance to the target node for the winning guesses is 8.238km with a standard deviation of 5.623km.

4.3 Divide and Conquer

4.3.1 Results

The results for the hedging strategy *Divide and Conquer* are shown in Tab. 4.3. Both the arenas - Bornholm and Malta - have been subjects to the test.

Tests have been conducted with three different numbers of iterations (1.000, 10.000, and 100.000) and two directions for the diagonal traversals: A and B. Both the means and standard deviations were measured and documented for all guesses and the winning guesses.

Table 4.3: Results of hedging strategy *Divide and Conquer* with various iterations and two diagonal traversal directions

Iterations	**Arena**	**Measure**	**All [km]**	**Winning [km]**
1.000	Bornholm A	Mean	17.131	11.158
		σ	8.034	5.690
	Bornholm B	Mean	18.668	8.081
		σ	10.686	3.195
	Malta A	Mean	12.626	10.767
		σ	5.647	5.548
	Malta B	Mean	13.182	5.352
		σ	9.098	3.056
10.000	Bornholm A	Mean	17.258	11.171
		σ	8.059	5.620
	Bornholm B	Mean	18.681	8.148
		σ	10.625	3.109
	Malta A	Mean	12.740	10.905
		σ	5.527	5.461
	Malta B	Mean	12.959	5.199
		σ	8.996	3.028
100.000	Bornholm A	Mean	17.226	11.167
		σ	8.056	5.676
	Bornholm B	Mean	18.713	8.084
		σ	10.674	3.123
	Malta A	Mean	12.740	10.915
		σ	5.518	5.461
	Malta B	Mean	12.956	5.210
		σ	8.980	3.016

4.3.2 Summary

The test results for the hedging strategy *Divide and Conquer* with 100.000 iterations using traversal direction A show, that the mean distance to the target node for all guesses on the arena of Bornholm is 17.226km with a standard deviation of 8.056km.

On the arena of Malta, the mean distance to the target node for all guesses using traversal direction A is 12.740km with a standard deviation of 5.518km.

The test results for the hedging strategy *Divide and Conquer* with 100.000 iterations using traversal direction B show, that the mean distance to the target node for all guesses on the arena of Bornholm is 18.713km with a standard deviation of 10.674km. On the arena of Malta, the mean distance to the target node for all guesses using traversal direction B is 12.956km with a standard deviation of 8.980km.
For the winning guesses on Bornholm, the mean distance to the target node using traversal direction A is 11.167km with a standard deviation of 5.676km. On the arena of Malta, the mean distance to the target node using traversal direction A for the winning guesses is 10.915km with a standard deviation of 5.461km.

For the winning guesses on Bornholm, the mean distance to the target node using traversal direction B is 8.048km with a standard deviation of 3.123km. On the arena of Malta, the mean distance to the target node using traversal direction B for the winning guesses is 5.210km with a standard deviation of 3.016km.

4.4 Closeness Centrality

4.4.1 Results

The results for the hedging strategy *Closeness Centrality* are shown in Tab. 4.4. Both the arenas - Bornholm and Malta - have been subjects to the test.

Tests have been conducted with three different numbers of iterations (1.000, 10.000, and 100.000) and both the means and standard deviations were measured and documented for all guesses and the winning guesses.

The diagonal traversal direction B was used for the tests because this direction showed better results than direction A as shown in section 4.3.1.

Table 4.4: Results of hedging strategy *Closeness Centrality* with various iterations

Iterations	**Arena**	**Measure**	**All [km]**	**Winning [km]**
1.000	Bornholm	Mean	19.350	7.679
		σ	11.525	2.883
	Malta	Mean	12.147	5.538
		σ	8.801	4.189
10.000	Bornholm	Mean	19.499	7.707
		σ	11.583	2.893
	Malta	Mean	11.960	5.293
		σ	8.574	4.024
100.000	Bornholm	Mean	19.501	7.657
		σ	11.624	2.903
	Malta	Mean	12.069	5.356
		σ	8.634	4.078

4.4.2 Summary

The test results for the hedging strategy *Closeness Centrality* with 100.000 iterations show, that the mean distance to the target node for all guesses on the arena of Bornholm is 19.501km with a standard deviation of 11.624km. On the arena of Malta, the mean distance to the target node for all guesses is 12.069km with a standard deviation of 8.634km.

For the winning guesses on Bornholm, the mean distance to the target node is 7.657km with a standard deviation of 2.903km. On the arena of Malta, the mean distance to the target node for the winning guesses is 5.365km with a standard deviation of 4.078km.

4.5 K-Means Clustering

4.5.1 Results

The results for the hedging strategy *K-Means Clustering* are shown in Tab. 4.5. Both the arenas - Bornholm and Malta - have been subjects to the test.

Tests have been conducted with three different numbers of iterations (1.000, 10.000, and 100.000) and both the means and standard deviations were measured and documented for all guesses and the winning guesses.

Table 4.5: Results of hedging strategy *K-Means Clustering* with various iterations

Iterations	**Arena**	**Measure**	**All [km]**	**Winning [km]**
1.000	Bornholm	Mean	19.185	7.389
		σ	11.491	2.883
	Malta	Mean	15.340	4.453
		σ	11.088	1.960
10.000	Bornholm	Mean	19.342	7.465
		σ	11.523	2.801
	Malta	Mean	15.279	4.362
		σ	11.137	1.930
100.000	Bornholm	Mean	19.362	7.477
		σ	11.531	2.831
	Malta	Mean	15.282	4.345
		σ	11.146	1.939

4.5.2 Summary

The test results for the hedging strategy *K-Means Clustering* with 100.000 iterations show, that the mean distance to the target node for all guesses on the arena of Bornholm is 19.362km with a standard deviation of 11.531km. On the arena of Malta, the mean distance to the target node for all guesses is 15.282km with a standard deviation of 11.146km.

For the winning guesses on Bornholm, the mean distance to the target node is 7.477km with a standard deviation of 2.831km. On the arena of Malta, the mean distance to the target node for the winning guesses is 4.345km with a standard deviation of 1.939km.

4.6 Collected results

4.6.1 Results

The results for the tests with 100.000 iterations for all five different hedging strategies are shown in Tab. 4.6. Furthermore, as the hedging strategy *Divide and Conqer* has been tested with two traversal directions, *A* and *B*, Tab. 4.6 includes the results for both.

Because Tab. 4.6 only summarizes the results for the tests with 100.000 iterations, the first half of the rows concern the results for the arena of Bornholm, whereas the last half of the rows relate to the tests conducted in the arena of Malta.

4.6.2 Summary

As Tab. 4.6 only offers a collected overview for both arenas with no sorting or declaration of interrelationships in terms of relative improvements, Tab. 4.7 is created. To fewer the amount of rows and values the standard deviations have been excluded from Tab. 4.7. Furthermore, Tab. 4.7 only shows the values for the winning guesses.

However, the hedging strategies have been sorted by their mean distance to the target node of the winning guesses, and the increase in distance to the target node for each hedging strategy compared to the best hedging strategy is calculated and shown in percentage.

The test results for all hedging strategies using 100.000 iterations in Tab. 4.7 shows, that the best hedging strategy for both the arena of Bornholm and Malta is *K-Means Clustering.*

In the network of Bornholm, *K-Means Clustering* is 2.407% better that the second-best hedging strategy, *Closeness Centrality*, and 54.768% better than the worst hedging strategy, *Random Guess in Network.*
In the network of Malta, *K-Means Clustering* is 19.908% better that the second-best hedging strategy, *Divide and Conquer B*, and 151.208% better than the worst hedging strategy, *Divide and Conquer A*.

For both the arena of Bornholm and Malta, the upper half consists of the hedging strategies *Closeness Centrality*, *Divide and Conquer B*, and *K-Means Clustering*, whereas the lower half consists of the hedging strategies *Divide and Conquer A*, *Random Guess in Convex Hull*, and *Random Guess in Network.* The percentage gap between

Table 4.6: Results of all hedging strategies with 100.000 iterations

Arena	**Hedging strategy**	**Measure**	**All [km]**	**Winning [km]**
Bornholm	Random Guess in Network	Mean	20.934	11.572
		σ	11.855	7.420
	Random Guess in Convex Hull	Mean	20.012	11.425
		σ	11.062	6.963
	Divide and Conquer A	Mean	17.226	11.167
		σ	8.056	5.676
	Divide and Conquer B	Mean	18.713	8.084
		σ	10.674	3.123
	Closeness Centrality	Mean	19.501	7.657
		σ	11.624	2.903
	K-Means Clustering	Mean	19.362	**7.477**
		σ	11.531	2.831
Malta	Random Guess in Network	Mean	13.684	7.298
		σ	10.215	6.357
	Random Guess in Convex Hull	Mean	15.740	8.238
		σ	9.800	5.623
	Divide and Conquer A	Mean	12.740	10.915
		σ	5.518	5.461
	Divide and Conquer B	Mean	12.956	5.210
		σ	8.980	3.016
	Closeness Centrality	Mean	12.069	5.356
		σ	8.634	4.078
	K-Means Clustering	Mean	15.282	**4.345**
		σ	11.146	1.939

these two halves is 45.361% for Bornholm and 44.695% for Malta. This gap between the upper and lower halves of the test results for Bornholm and Malta is the largest intermediary gap for Bornholm and the second-largest intermediary gap for Malta. Only the gap between *Random Guess in Convex Hull* and *Divide and Conquer A* is larger for Malta.

Table 4.7: Sorted results of all hedging strategies with 100.000 iterations with the percentages of relative distance increasements for the mean of best guesses

Arena	**Hedging strategy**	**Mean of winning guess**	**Relative increase to best strategy**
Bornholm	K-Means Clustering	7.477	
	Closeness Centrality	7.657	2.407%
	Divide and Conquer B	8.084	8.084%
	Divide and Conquer A	11.167	49.351%
	Random Guess in Convex Hull	11.425	52.802%
	Random Guess in Network	11.572	54.768%
Malta	K-Means Clustering	4.345	
	Divide and Conquer B	5.210	19.908%
	Closeness Centrality	5.356	23.268%
	Random Guess in Network	7.298	67.963%
	Random Guess in Convex Hull	8.238	89.597%
	Divide and Conquer A	10.915	151.208%

Chapter 5

Discussion

In this chapter, the implemented and tested hedging strategies will be discussed in section 5.1. It offers remarks on the results of the individual arenas as well as comments on general observations and tendencies. In section 5.2, the research on hedging strategies and applied algorithmic procedures will be related to existing work for the purpose of exploring synergies. Limitations for the presented research are discussed in section 5.3, where computational limitations, limitations for the k-means algorithm, the validity of OSM data, and the result replication for arbitrary arenas will be discussed. The applicability of the conducted research in other domains is discussed and explored in section 5.4. Finally, section 5.5 provides suggestions for future research.

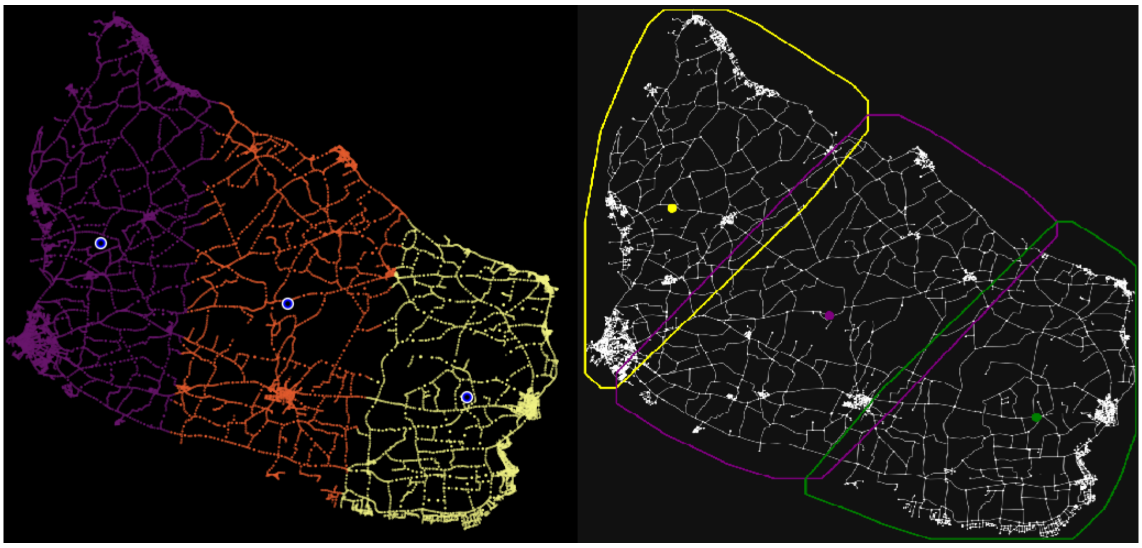
5.1 Comparison of Results

The hedging strategy *K-Means Clustering* is found to be the best hedging strategy for both Bornholm and Malta. Compared to the second-best hedging strategy for Bornholm, the *Closeness Centrality* hedging strategy, the *K-Means Clustering* is 2.407% better. For the arena of Malta, the *K-Means Clustering* hedging strategy is 19.908% better than the second-best hedging strategy *Divide and Conquer B*.

Besides manifesting a common winning hedging strategy, the *K-Means Clustering* hedging strategy, the comparison of the two best hedging strategies also displays an interesting observation: the second-best hedging strategy for Bornholm is only 2.407% less optimal than the winning hedging strategy, whereas for Malta this difference is 19.908%.

Reasoning for these results can be obtained visually by studying both the divisions and resulting guesses thereof for the hedging strategies, *K-Means Clustering* and *Closeness Centrality* on Bornholm, as shown in Fig. 5.1.

Figure 5.1: An inspection of the two best hedging strategies for Bornholm, *K-Means Clustering* (left) and *Closeness Centrality* (right)



Even though the divisions of the network for the two hedging strategies have been obtained from fundamentally different computations, both the sub-networks and the guesses are very similar.

Recalling from Alg. 5, the basis for the divisions of the network has a rationale in a fairness principle of each division comprising of $\frac{1}{3}$ of the total weight of the network, whereas the divisions created by Alg. 6 have no such requirements.

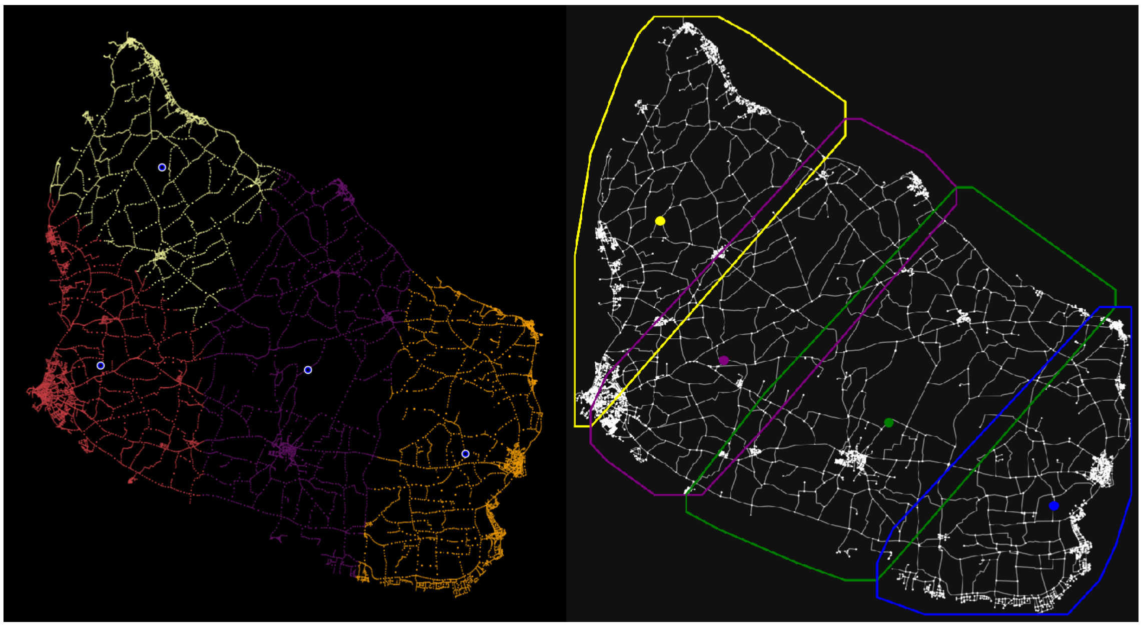
The *Closeness Centrality* hedging strategy has its traversal direction dictate the greater shape of the diagonals of the resulting convex hulls. The *K-Means Clustering* hedging strategy has no such predefined traversal technique, however the resulting divisions of the two hedging strategies are remarkably alike. Also for the *k-Means Clustering* hedging strategy, the resulting divisions have a slight diagonal feature, that is in the same diagonal direction as the traversal direction B used for the *Closeness Centrality* hedging strategy.

Placing the guesses within the divisions - the clusters for *K-Means Clustering* and the convex hulls for *Closeness Centrality* - also share similarities and differences. Both hedging strategies place the guesses based on a search for a minimum total distance within the division. However, the *K-Means Clustering* hedging strategy is not bound to place its guess in the network, whereas the *Closeness Centrality* hedging strategy has this condition.

Comparing the three guesses resulting from the two best hedging strategies from left to right, the following distance differences are calculated: 2.363km, 3.495km, and 1.183km. Recollecting that the two hedging strategies use both different algorithmic techniques to divide the network and place the guesses, these minor differences are remarkable.

This similarity between the guesses from *K-Means Clustering* and *Closeness Centrality* should be considered as an isolated case, as changing n, the number of guesses, will result in non-similar guesses. This hypothesis is tested and visualized in Fig. 5.2.

Figure 5.2: An inspection of the two hedging strategies, *K-Means Clustering* (left) and *Closeness Centrality* (right), for Bornholm with $n = 4$



As the *K-Means Clustering* hedging strategy is not procedurally bound to traverse in diagonals when creating clusters as it utilizes the k-means algorithm, the divisions and guesses differentiate substantially with $n = 4$.

The reasoning for the substantial distance difference between the best hedging strategy, *K-Means Clustering*, and the second-best hedging strategy, *Divide and Conquer B*, for the arena of Malta is predominantly based on the algorithmic procedures for dividing the network.

Unlike the *Divide and Conquer B* hedging strategy, the *K-Means Clustering* hedging strategy is not bound to divide the network of the arena into divisions of equal weight. This is due to the native clustering procedure of the k-means algorithm. Therefore, as also visualized in Fig. 3.10b, the northwestern island of Malta, Gozo, becomes its own cluster due to its sub-networks distance to the rest of the network.

In order for the hedging strategy *Divide and Conquer B* to obtain such an isolation of Gozo, the only parameter to change is n, the number of guesses. For all hedging strategies *Divide and Conquer A*, *Divide and Conger B*, and *Closeness Centrality* there exists a direct correlation between the sizes of the divisions in terms of weight and the number of guesses, as the network is divided into n equally weighted sub-networks. Furthermore, the diagonal traversal direction should be selected accurately to fit the purpose.

If the hedging strategy *Divide and Conquer B* is to isolate Gozo in one division as the hedging strategy *K-Means Clustering* does, it would require the parameter n to be 7 as visualized in Fig. 5.3.

Figure 5.3: An inspection of the two hedging strategies, *K-Means Clustering* (left) and *Divide and Conquer B* (right), for Bornholm with $n = 7$

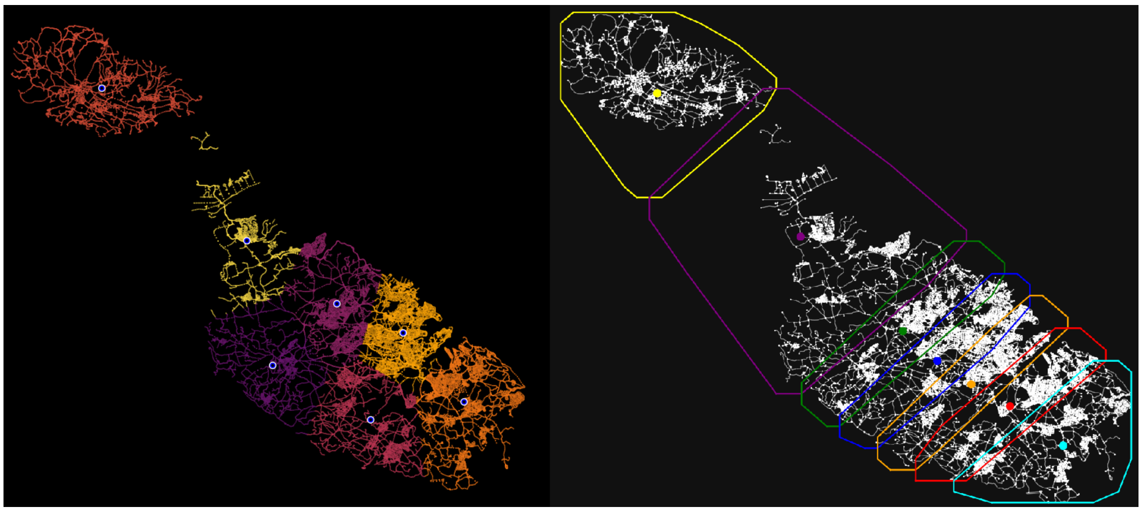


Fig. 5.3 shows that if $n = 7$ the sub-network on the island of Gozo is isolated into a division when using the hedging strategy *Divide and Conquer B* on Malta. However, when setting the parameter $n = 7$ for the *K-Means Clustering* hedging strategy, the island of Gozo is still isolated. This underlines the native qualities of the k-means algorithm.

Another interesting observation of the results can be found in the second-best and third-best hedging strategy for the two arenas. For the arena of Bornholm, the second-best hedging strategy is *Closeness Centrality*, and the third-best hedging strategy is *Divide and Conquer B*. The opposite ranking is the case for Malta, where the second-best hedging strategy is *Divide and Conquer B*, and the third-best hedging strategy is *Closeness Centrality*.

As observable on the guesses on Malta placed by the winning hedging strategy, *K-Means Clustering*, the northwestern guess is located on the island of Gozo. Comparing the northwestern guesses on Malta of the two hedging strategies *Divide and Conquer B* and *Closeness Centrality*, it can be concluded that *Divide and Conquer B* places its guess more northwest than *Closeness Centrality*. The other two guesses don't differentiate nearly as much.

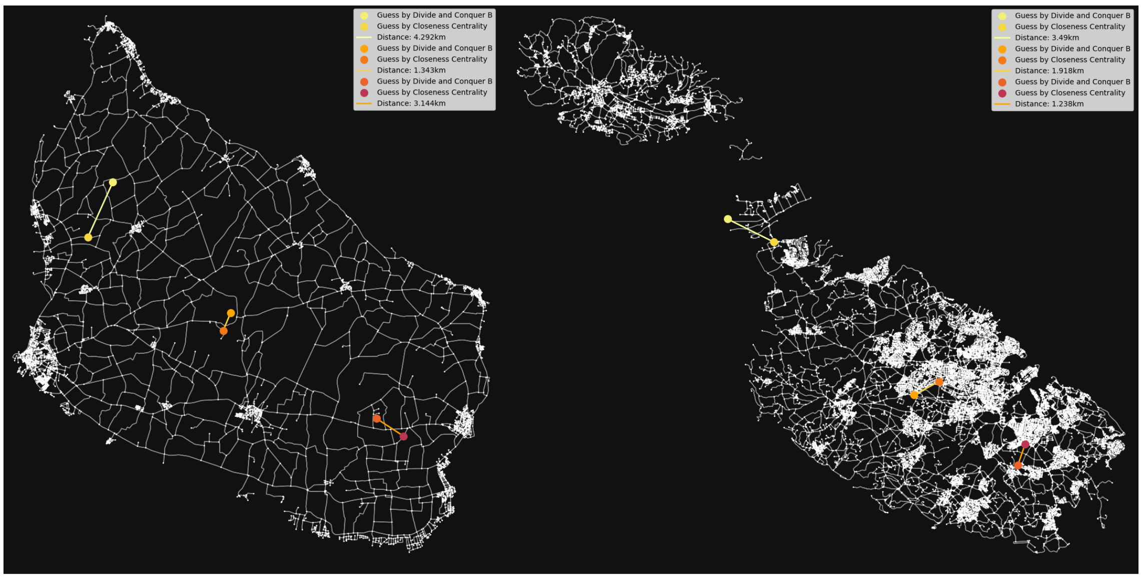
Computing the distance differences for the guesses on Malta by *Divide and Conquer B* and *Closeness Centrality* from left to right gives the following results: 3.490km, 1.918km, and 1.238km. The calculations show, that the biggest distance difference between the guesses is found for the most northwestern guess, which supports the hypothesis of an optimal guess, as demonstrated by the hedging strategy *K-Means Clustering*, lies on the island of Gozo.

In the case of Bornholm, the ranking is the opposite: the second-best hedging strategy is *Closeness Centrality* and the third-best hedging strategy is *Divide and Conquer B*. This is because the algorithmic procedure of the underlying divisioning of the network is the same, however the inter-placement strategy for the guesses within these divisions differs. The hedging strategy *Divide and Conquer B* places its guesses within the divisions based on the midpoints, also known as the geometric centroids, whereas the hedging strategy *Closeness Centrality* places its guesses based on the most central node given by the closeness centrality score.

As no big area of sea with no network is dividing the collected subnetwork within the divisions, the higher precision based on a centrality analysis given by the closeness centrality score is outperforming the more simplistic approach of using the midpoints. The distance differences for the guesses from left to right are calculated with the following results: 4.292km, 1.343km, and 3.144km.

Fig. 5.4 shows a comparison of the two hedging strategies, *Divide and Conquer B* and *Closeness Centrality*, on both the arenas of Bornholm and Malta.

Figure 5.4: An inspection of the second-best and the third-best hedging strategies, *Closeness Centrality* and *Divide and Conquer B*, for Bornholm and Malta



Guess by Divide and Conquer B
Guess by Closeness Centrality
Distance: 4.292km
Guess by Divide and Conquer B
Guess by Closeness Centrality
Distance: 1.343km
Guess by Divide and Conquer B
Guess by Closeness Centrality
Distance: 3.144km
Guess by Divide and Conquer B
Guess by Closeness Centrality
Distance: 3.49km
Guess by Divide and Conquer B
Guess by Closeness Centrality
Distance: 1.918km
Guess by Divide and Conquer B
Guess by Closeness Centrality
Distance: 1.238km

The guesses made by the hedging strategies *Divide and Conquer B* and *Closeness Centrality* are seemingly alike, as Fig. 5.4 illustrates.

However, the minor differences discussed are evidently decisive enough to rank the hedging strategies differently for the two arenas.

As also discussed in [49], choosing a traversal direction for the hedging strategies *Divide and Conquer*, that does not follow the shape of the arena, is a better choice. However, [49] does not test this hypothesis.

This thesis has shown by implementation and thorough testing, that the claim holds. For the arena of Bornholm when comparing the results for the mean of best guesses after 100.000 iterations, the *Divide and Conquer B* hedging strategy using traversal direction *B* is 45.361% better than the hedging strategy *Divide and Conquer A* using traversal direction *A*.

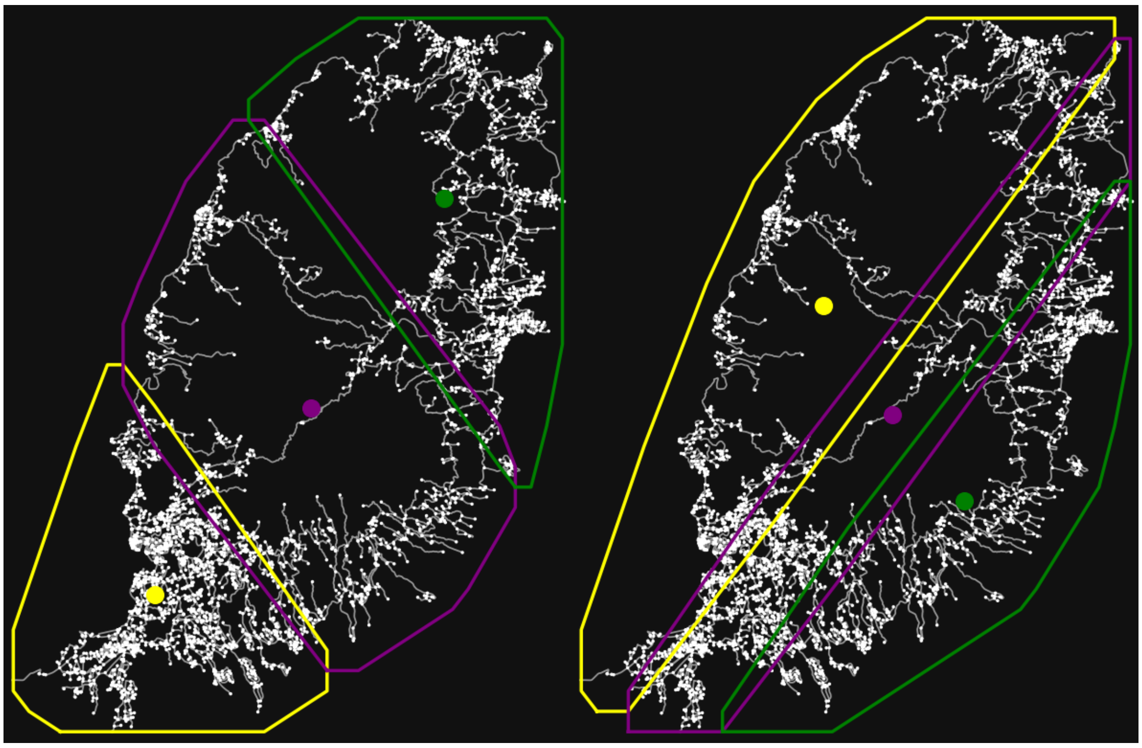
When comparing the mean of best guesses after 100.000 iterations for the arena of Malta, the *Divide and Conquer B* hedging strategy using traversal direction B is 131.300% better than the hedging strategy *Divide and Conquer A* using traversal direction A. The increase in relative distance difference is presumably due to the narrower shape of Malta compared to Bornholm.

As the results show in Tab. 4.7, the hedging strategies *Divide and Conquer B* and *Closeness Centrality*, which are both using the same diagonal traversal procedure for divisioning the network, are ranked within the second-best and third-best hedging strategy. These results do not make the hedging strategies competitive with the hedging strategy *K-Means Clustering* however, as of now they still require a manual judgment on which traversal direction to put in use.

The native qualities of the hedging strategy *K-Means Clustering* is capable of divisioning - in this case, clustering - the network on its own without any preconditions of inputs or parameters declaring the shape of the arena.

Examples can be found, where the use of traversal direction A outperforms traversal direction B. Fig. 5.5 shows two images of the main island of Grenada. The left image uses the hedging strategy *Divide and Conquer A*, whereas the image to the right uses the same hedging strategy but with traversal direction B.

Figure 5.5: A comparison of traversal direction A (left) and B (right) on the main island of Grenada



Unlike the shape of Bornholm and Malta, the main island of Grenada, as also shown in appendix C and F, has a diagonal shape parallel to traversal direction B, which makes traversal direction A an intuitively better choice.

This hypothesis is explored, and therefore the hedging strategies *Divide and Conquer A* and *Divide and Conquer B* are tested with 100.000 iterations.

The mean distance to the target for the winning guess of *Divide and Conquer A* is 2.991km with a standard deviation of 1.297km, whereas the mean distance to the target for the winning guess of *Divide and Conquer B* is 4.975km with a standard deviation of 2.290km.

Tests have shown, that for the main island of Grenada, unlike Bornholm and Malta, using traversal direction *A* implies better results. For the case of Grenada specifically, using the diagonal traversal *A* compared to direction *B* yields 66.332% better results in combination with the general *Divide and Conquer* approach.

5.2 Synergy with Other Research

The research presented in this thesis has the assumption, that the players know in which region or country the target is located.

Even though this assumption is not far-fetched, as seen in [24] due to the tells and techniques described in among others [58], an automated way of narrowing the arena would statistically yield better results for the presented hedging strategies in sections 3.2-3.6.

Exactly this task is described by [68], [70], and [59], in which each study also proposes a solution.

As described in section 2.1.2, [68] proposes a neural network that is trained with images from the United States of America. Each state contributes an equal amount of test data. However, [68] offers no following guidance on how to precisely place a guess within a region after the classification problem has been solved.

In [70] a model is proposed that, based on the data used for its training, divides the entire world into envelopes of uneven sizes. For example, small envelopes exist especially in certain parts of Europe, because a substantial part of the scraped imagery has EXIF data disclosing a European location. As with [68], also [70] has no technique for placing guesses within a division. The model simply chooses the: *"the center of the highest confidence [...] cell as its guess"* [70].

Correspondingly [59] proposes a model that is similar to [68] and [70]. However, instead of narrowing the arena down to one state of the United States of America or a cell of the entire world, [59] aims to output the correct country.

At this point, the research and developed hedging strategies presented in this thesis have the ability to contribute to the published research. As the *K-Means Clustering* hedging strategy has proved to be outperforming the other proposed hedging strategies on both Bornholm and Malta, it could be applied to either the states of the United States of America, the cells described in [70] or the countries of the world and precalculate guesses.

Another possibility for combining the models presented in [68], [70], and [59] with the hedging strategies developed in this thesis is to test all hedging strategies for either each state of the United States of America, the cells described in [70] or the countries of the world and choose the winning strategy. Although the *K-Means Clustering* hedging strategy has proven to be the winning strategy for the arenas of Bornholm and Malta, it can not be precluded that there exist arenas, for which one of the other proposed hedging strategies has better results.

All the models proposed by [68], [70], and [59] eventually produce a ranking of the most plausible regions being either a state, a cell, or a country. To possibly obtain a better result, a cross-research twofold strategy can be comprised. First, take the three most plausible regions. Second, compose a collected network of these regions, create an arena, and use one of the hedging strategies proposed in this thesis.

If the three most plausible regions are not neighboring, the most promising hedging strategy would presumably be *K-Means clustering*, as it would create a cluster and guess for each region. If the plausible regions consist of neighboring states, cells, or countries, all hedging strategies should be tested.

5.3 Limitations

5.3.1 Computational Limitations

All the hedging strategies require the prerequisites of a fetched network that has been parsed into a graph. However, following this step, the computational complexity varies for each of them.

The random hedging strategies, *Random Guess in Network* and *Random Guess in Convex Hull* require the least computational efforts. The convex hull of a set of nodes, N, can be computed with the time complexity of $O(N\ log\ N)$, and placing a random guess either in the network or within a polygon is possible in constant time [26].

The hedging strategy *Divide and Conquer*, regardless of the diagonal traversal direction, uses further data preprocessing and calculations for generating its guesses.

Initially, the data structure representing roads is iterated through to extract all nodes. Followed by this iteration, the nodes are mapped to their respective grid cells. Assuming N to be the number of translated weighted center points of the network, this operation has the time complexity of $O(N)$.

Thereafter the convex hulls are calculated for each of the n greater divisions, where n is the number of guesses. Calculating the convex hull of a set of points has the time complexity of $O(N\ log\ N)$, where N is the number of points [26]. The calculation of the convex hulls uses the points of the bounding boxes of the grid cells in each of the n greater divisions.

The geometric centroid has to be extracted for each of the n convex hulls with time complexity of $O(p)$, where p is the number of points for each of the polygons [38].

As for the hedging strategy *Divide and Conquer*, also *Closeness Centrality* initially iterates through the data structure representing the roads of the network in order to store all of the center points of the edges and their weights. This data preprocessing step has the time complexity of $O(N)$, where N is the number of translated weighted center points in the network.

Followed by that, the convex hulls are computed with time complexity $O(b \; log \; b)$, where b is the number of points of the bounding boxes of the grid cells [26].

Now, unlike the *Divide and Conquer* hedging strategy, where the geometric centroids are computed, the *Closeness Centrality* hedging strategy computes for each center point within the convex hulls the total weighted distance to all other center points.

The widely adopted version of the closeness centrality measure, which uses a shortest path algorithm to determine the closeness score, can be optimized in certain ways. A general text-book implementation would require calculating the shortest path from each node to all other nodes [41]. However, the *Closeness Centrality* hedging strategy is not interested in the shortest path, as it requires the score based on the weighted haversine distance. The calculation of the closeness centrality scores has a time complexity of $O(N^2)$, where N is the number of weighted points and thereby equals the number of edges, E, of the network.

As for the other non-random hedging strategies, also the *K-Means Clustering* hedging strategy requires data preprocessing, that translates the edges of the network into weighted center points with the time complexity of $O(N)$, where N is the number of translated center points.

Hereafter, the k-means algorithm is employed. The task is NP-hard, yet the calculation of approximations can be done feasibly. An agreed-upon time complexity of the k-means algorithm has been challenging to find, partly because the actual running time of the algorithm is greatly dependent on the input and initial clustering [55] [28].

Some procedures, however, can be estimated. Each iteration, $1..i$ requires the calculation of the distance from each node to each centroid. Subsequently, the centroids are recalculated with time complexity equal to the number of points for each cluster [72]. Given that N is the number of weighted center points and k is the number of guesses, the time complexity of the used k-means algorithm can be expressed as $O(i \; k \; N)$.

As both the hedging strategies *Closeness Centrality* and *K-Means Clustering* are rapidly increasing in computational complexity by both the size of the network and other parameters, simplifications of the studied networks can be constructed to keep the computations in a feasible state.

The implementation of the *Closeness Centrality* hedging strategy shows, that such a simplification is offered as a parameter [23]. The simplification parameter, s, groups s nearby weighted center points, by creating a new replacement point that has the center coordinates of the s center points paired with the sum of the s center points' weights.

Using the simplification parameter s will thereby reduce the input size by a factor of approximately s. It will not exactly result in a reduction factor equal to itself, as edges can be comprised of a number of nodes, that modulo s leaves a remainder.

5.3.2 K-means Limitations

Most implementations of the k-means algorithm do not promise a global optimum, as the task of computing this is infeasible for most data sets. Instead, the algorithm converges into local optima. The clustering and resulting centroids are in many cases changing for each calculation [42]. Arguing that the hedging strategy *K-Means Clustering* is an optimal solution, therefore, becomes unattainable.

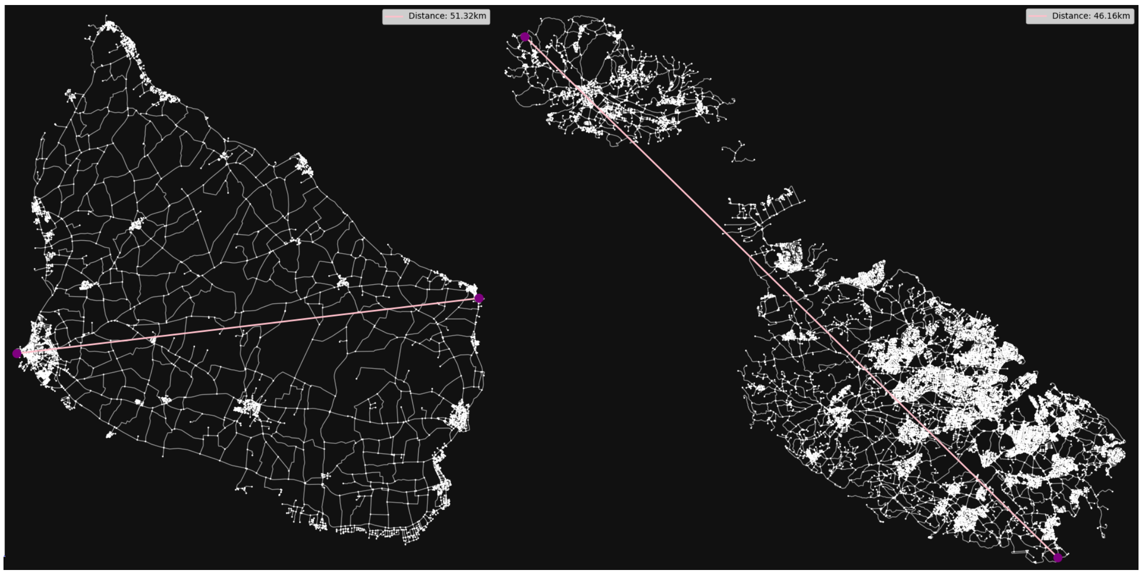
The initial selection of centroids is influential on the final clustering, as *"the kmeans clustering algorithm greatly depends upon the correctness of the initial centroids"* [73]. However, with clever procedures for placing the initial centroids, as described by [73], [43], and many others, a consistently acceptable final clustering can be obtained.

As seen in Fig. 3.12 that displays the final clusters when the initial centroids are chosen randomly, the outcome varies greatly. This was the predominant reason for choosing the k-means function by `sklearn`, as it offers the possibility to optimize the initial centroid positions.

Because the *K-Means Clustering* hedging strategy uses a k-means algorithm that calculates Euclidean distances from the nodes to the centroids, some margin of error is introduced. The curvature of the Earth requires the use of a haversine distance calculation between two points [71].

To investigate the potential size of the introduced error margin, the greatest possible span between two nodes is identified. This process is visualized in Fig. 5.6.

Figure 5.6: An inspection of the maximum span between two nodes in the networks of Bornholm (left) and Malta (right)



As seen in Fig. 5.6, the greatest distance between two nodes, n_1 and n_2, in each of the networks is respectively 51.32km for Bornholm and 46.16km for Malta. An examination is conducted for each arena, where the mean of the coordinates is saved as m_{coords}.

As a control experiment, exp_{euc}, the total Euclidean distance from n_1 to n_2 is calculated and saved as euc_{total}. Now the half-distance, euc_{half} is calculated from arbitrarily

n_1 or n_2 to m_{coords}. The results show, that euc_{half} is exactly $\frac{1}{2}$ of euc_{total}.

Similar to exp_{euc}, an experiment is conducted, exp_{hav}, that initially calculates the total haversine distance from n_1 to n_2 and saves the result as hav_{total}. The result, hav_{total}, equals the distances shown in Fig. 5.6. Now the half-distance, hav_{half} is calculated from arbitrarily n_1 or n_2 to m_{coords}. The results show, that hav_{half} is not exactly $\frac{1}{2}$ of hav_{total}, but is slightly off. The half-distances accord to respectively the half +/- a small corrective. This is due to the curvature of the Earth. For Bornholm, the distance is 0.006% off for the half distance, and for Malta, the distance is 0.014% off for the half distance.

In other words, using Euclidean distance measurements as compensation for using haversine distance measurements introduces a small error margin. However, for larger regions, especially close to the poles (Russian, Canada, etc.), distortions may be significantly larger.

The potential error margin introduced by using Euclidean distance calculations in the clusters can also be tested on the final clustering by examining whether or not the nodes of a cluster are actually closest to the assigned centroid.

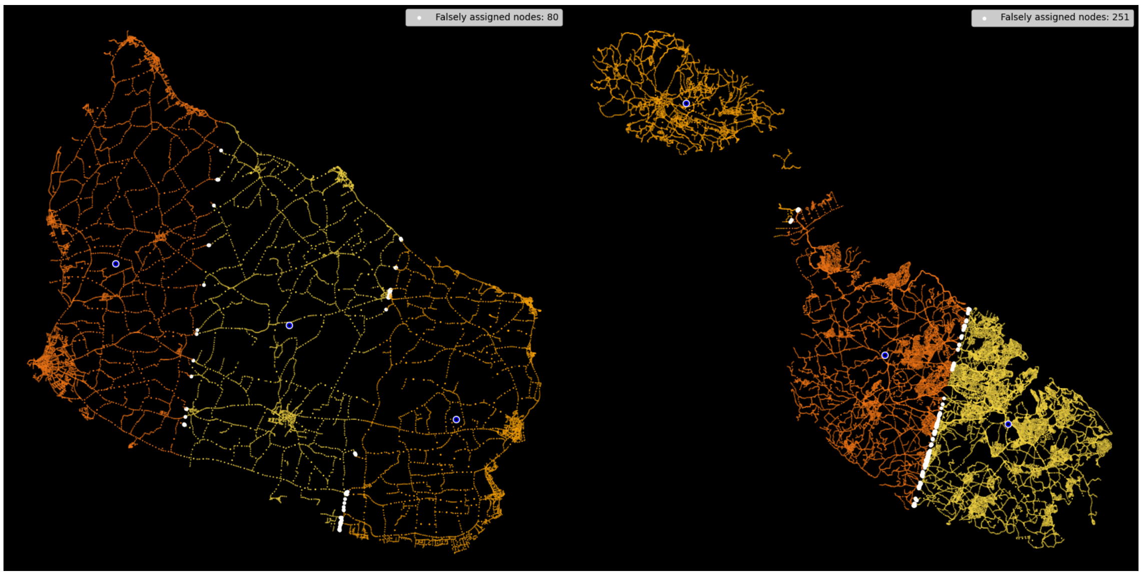
This control procedure can be implemented for the solution with time complexity of $O(n\ N)$, where n is the number of guesses and N is the total number of nodes.

Fig. 5.7 shows this investigation. It visualizes with the highlighted white circles, that the nodes assigned to the wrong centroids are located close to the dividing borders of the clusters.

A total of 80 wrongly assigned nodes are found for the resulting clusters of Bornholm, which is equal to 0.306% of the total amount of nodes.

For Malta, a total of 251 wrongly assigned nodes are found for the resulting clusters, which is equal to 0.296% of the total amount of nodes.

Figure 5.7: Nodes assigned to the wrong centroids during clustering in the networks of Bornholm (left) and Malta (right)



5.3.3 Validity of OSM Data

The imagery presented in GeoGuessr stems from Google Street View and thereby also uses the underlying geospatial data by Google. The geospatial data used and parsed into geospatial networks in this thesis originates from OSM, which presents two possible impediments to data correctness.

First, since the data sources used in GeoGuessr and for this thesis do not share the same origin, there might be potential misalignments for the networks.

Second, the validity of OSM data itself has been questioned by multiple sources. [48] states, that *"OSM faces the challenge of ensuring data quality"*, since the contributors and data collectors are largely non-professionals. Also [25] states, that *"this kind of contribution mechanism is usually associated with data quality issues"*.

The contribution model allows users to submit various and elaborate descriptions as documented in Tab. 2.1, however *"it is not possible to access ground-truth spatial data"* [61].

Both [54] and [50] try to compare the geospatial data from OSM with other data providers and true positional data. [50] concludes, that the *"[q]uality differs locally, and even in a single town the different aspects of quality may vary"*, while the studies of [54] *"shows highest correlation between OSM/[TomTom] and their true position, but the [TomTom] dataset has less distortion than OSM"*.

The isolated use of OSM data with the tested hedging strategies and their underlying algorithmic procedures does not suffer from a potential data quality issue. However, using the hedging strategies within the domain of GeoGuessr, that is using a different data source, might introduce small inaccuracies.

5.4 Applicability in Other Domains

Apart from applications within the game GeoGuessr, several other domains might benefit from using the proposed hedging strategies presented in section 3.2-3.6, especially the *K-Means Clustering* hedging strategy and customized varieties of it.

The warehouse location problem has been investigated and studied in multiple published research, among others in [33], [52], [40], and [45]. In short, solutions to the problem aims to *"choose the location of the warehouse[s] or distribution center[s] to maximize efficiency"* [53].

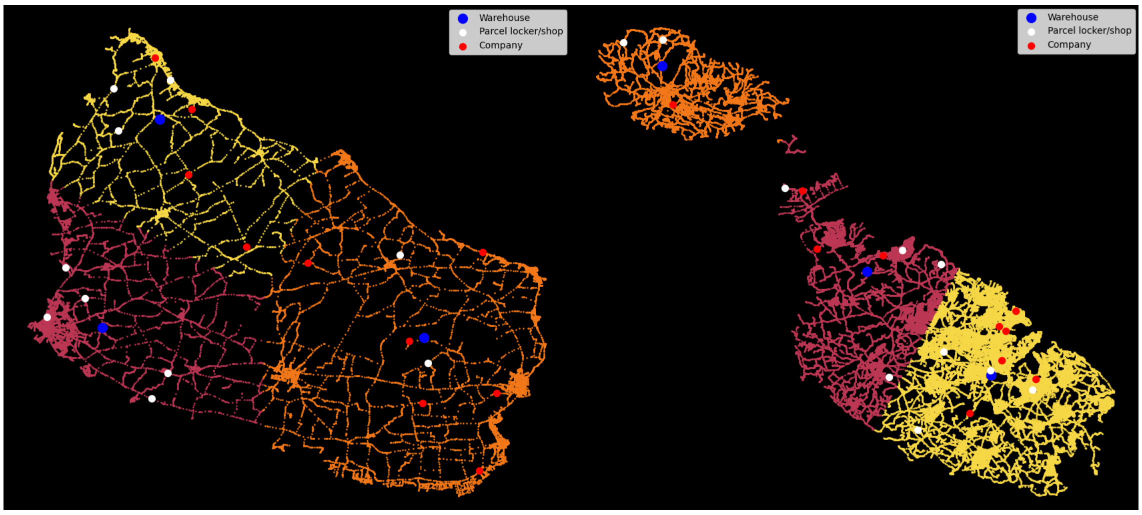
The *K-Means Clustering* hedging strategy presented in this thesis can be applied to the problem, however some customizations of both the network and the hedging strategy have to be implemented.

First, the network should instead of representing driveable roads represent possible delivery destinations. These possible delivery destinations are both private households, companies, and most importantly parcel lockers and partnered parcel shops.

Second, the customized *K-Means Clustering* hedging strategy should weight these possible delivery destinations individually based on their type. The frequency of delivering packages to a private household address is lower than the frequency of delivering packages to companies, parcel lockers, and partnered parcel shops.

The resulting clustering should therefore account for both the distance to the potential delivery destinations and their types. Fig. 5.8 shows a customization of the *K-Means Clustering* hedging strategy, that aims to solve a warehouse location problem of size $n = 3$, where n is the number of warehouses.

Figure 5.8: A solution to the warehouse location problem for Bornholm (left) and Malta (right)



Compared to a weighting based on the length of the road section each node represent, the resulting clusters and centroid placements resulting from the *K-Means Clustering* hedging strategy are significantly different.

The weighting used for the implementation of the warehouse location problem is based on the type of the node. For the test visualized in Fig. 5.8, a normal node (center-point) representing a private household, is given the weight $w = 0.1$. Nodes representing companies are weighted with $w = 500$. Finally, nodes representing parcel lockers and partnered parcel shops have the weight $w = 1000$. These weights are customizable parameters in the implementation [23]. The warehouses are placed based on the resulting centroids from the weighted k-means clustering.

Customizing the *K-Means Clustering* hedging strategy to find a solution to the warehouse location problem demonstrates, that the research of this thesis indeed has geospatial applications apart from GeoGuessr.

Also within the domain of electric vehicle charging, the proposed hedging strategy *K-Means Clustering* can be customized to find locations for charging stations. In this case, the network and the weighting of its nodes would not have to be translated, because the network should resemble driveable roads.

When running the algorithm on a network, the longest distance from a node to its assigned centroid, max_{dist} can be stored during the execution. Following a given maximum distance requirement, req_{min}, the number of clusters n, representing the number of charging stations, can be incremented until max_{dist} adheres to req_{min}.

The same customization for a possible application within the positioning of electrical vehicle charging stations can be used within the domain of telecommunication infrastructure.

Where road infrastructure is established, people either live or travel. From this follows, that the entirety of a road network must be covered by cell sites for its residing population or visiting travelers to be online.

The algorithm can be executed $1..n$ times, where n is the number of cell sites, until the maximum distance from a node to its assigned centroid, max_{dist}, ensures coverage.

5.5 Future Research

As described in section 5.2, the research presented in this thesis is compatible with the models from [68], [70], and [59].

The models narrow the location of the target down to a state, area, or country, whereas the proposed hedging strategies place guesses based on an analysis of the network within these convolutions.

In particular, future research should implement the *K-Means Clustering* hedging strategy in the extension of [68], [70], and [59] for optimized geolocation.

The players of GeoGuessr usually know, in which country the target is located. Therefore, future research might take up the task to compute guesses for all the countries of the world, that are covered by Google Street View, using the *K-Means Clustering* hedging strategy.

Such an effort would produce a list of pre-calculated guesses for each country. Following the creation of the list, the guesses resulting from the *K-Means Clustering* hedging strategy can be compared to guesses made by the professional players of GeoGuessr.

Chapter 6

Conclusion

The problem of geolocating has been extensively studied from two perspectives. First, much research has been centered around human performance by investigating and mapping the reflections of professionals. Second, many neural networks and models have been created and trained with the purpose of geolocating from imagery. However, this thesis uses analytical methods and algorithmic procedures to precompute centralities in a geospatial network.

Five different hedging strategies were implemented and tested: *Random Guess in Network*, *Random Guess in Convex Hull*, *Divide and Conquer*, *Closeness Centrality*, and *K-Means Clustering*. The results showed, that the *K-Means Clustering* hedging strategy was 2.4% better than the second-best hedging strategy and 54.8% better than the worst hedging strategy in the arena of Bornholm. For Malta, the *K-Means Clsutering* hedging strategy was 19.9% better than the second-best hedging strategy and 151.2% better than the worst hedging strategy. The results were discussed, and the assumption of the correlation between the diagonal traversal direction and the shape of the arena was tested and confirmed on the main island of Grenada.

The discussion of the computational limitations revealed that optimizations and simplifications are needed for two of the hedging strategies, *Closeness Centrality* and *K-Means Clustering*, if the network size reaches a critical size. Such a simplification procedure of the geospatial network was successfully implemented and tested for the *Closeness Centrality* hedging strategy. Limitations when applying the k-means algorithm with geospatial data were discussed and in particular, the margin of error introduced by using Euclidean distances instead of haversine distances was investigated and reported.

Possible applications of the research and in particular the *K-Means Clustering* hedging strategy were explored. The *K-Means Clustering* was customized to solve the warehouse location problem by introducing new attributes to the nodes of the network. The potential applications within the domains of telecommunication infrastructure and electric vehicle charging stations were discussed.

Future research lies in using the *K-Means Clustering* hedging strategy to precompute guesses for each country of the world, that is covered by Google Street View. An effort should also be made to combine the work presented in this thesis with existing research and geolocating models.

Bibliography

[1] *"BrandWatch"*. URL: https://www.brandwatch.com/email/brandwatch-bulletin-0098-geoguessr/ (visited on 04/12/2023) (cit. on pp. 8, 9).

[2] *"GeoGuessr"*. URL: https://www.geoguessr.com/ (visited on 04/12/2023) (cit. on p. 8).

[3] *"GeoPandas centroid"*. URL: https://geopandas.org/en/stable/docs/user_guide/geometric_manipulations.html#GeoSeries.centroid (visited on 04/24/2023) (cit. on p. 32).

[4] *"GeoPandas convex_hull"*. URL: https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoSeries.convex_hull.html (visited on 04/24/2023) (cit. on pp. 28, 32).

[5] *"GeoPandas"*. URL: https://geopandas.org/en/stable/index.html (visited on 04/24/2023) (cit. on p. 28).

[6] *"Google Street View"*. URL: https://www.google.dk/intl/en/streetview/ (visited on 04/12/2023) (cit. on p. 8).

[7] *"Map of Bornholm"*. URL: https://www.openstreetmap.org/#map=11/55.1593/14.8734 (visited on 04/12/2023) (cit. on pp. 23, 70).

[8] *"Map of Grenada (main island)"*. URL: https://www.openstreetmap.org/search?query=grenada#map=12/12.1154/-61.6882 (visited on 05/09/2023) (cit. on p. 72).

[9] *"Map of Malta"*. URL: https://www.openstreetmap.org/search?query=malta#map=12/35.9377/14.3517 (visited on 04/17/2023) (cit. on pp. 23, 71).

[10] *"Merriam-Webster"*. URL: https://www.merriam-webster.com/dictionary/geospatial (visited on 04/14/2023) (cit. on p. 14).

[11] *"NetworkX Closeness Centrality"*. URL: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.closeness_centrality.html (visited on 04/12/2023) (cit. on pp. 17, 35).

[12] *"NetworkX compose_all"*. URL: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.operators.all.compose_all.html#compose-all (visited on 04/24/2023) (cit. on p. 23).

[13] *"NetworkX"*. URL: https://networkx.org/documentation/stable/ (visited on 04/24/2023) (cit. on p. 23).

[14] *"OpenStreetMap"*. URL: https://www.openstreetmap.org/ (visited on 04/12/2023) (cit. on pp. 11, 14).

[15] *"OSMnx graph_from_bbox"*. URL: https://osmnx.readthedocs.io/en/stable/osmnx.html?highlight=graph_from_place#osmnx.graph.graph_from_bbox (visited on 04/24/2023) (cit. on p. 23).

[16] *"OSMnx graph_from_place"*. URL: https://osmnx.readthedocs.io/en/stable/osmnx.html?highlight=graph_from_place#osmnx.graph.graph_from_place (visited on 04/24/2023) (cit. on p. 23).

[17] *"OSMnx sample_points*. URL: https://osmnx.readthedocs.io/en/stable/osmnx.html?highlight=sample_points#osmnx.utils_geo.sample_points (visited on 04/24/2023) (cit. on pp. 24, 25).

[18] *"OSMnx"*. URL: https://osmnx.readthedocs.io/en/stable/index.html (visited on 04/24/2023) (cit. on p. 23).

[19] *"sklearn data generation"*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html (visited on 04/12/2023) (cit. on p. 20).

[20] *"sklearn DBSCAN"*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html (visited on 04/24/2023) (cit. on p. 37).

[21] *"sklearn fit"*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.fit (visited on 04/24/2023) (cit. on p. 39).

[22] *"sklearn k-means"*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html (visited on 04/12/2023) (cit. on pp. 20, 37–39).

[23] *"Thesis GitHub repository"*. URL: https://github.com/MissingClosingBracket/GeoGuessrThesis (visited on 04/17/2023) (cit. on pp. 17, 57, 61).

[24] *"YouTube"*. URL: https://youtu.be/kv8-oaI6Avk?t=82 (visited on 04/12/2023) (cit. on pp. 9, 55).

[25] Ahmed Loai Ali et al. "Ambiguity and plausibility: managing classification quality in volunteered geographic information". *Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems.* 2014, pp. 143–152 (cit. on p. 60).

[26] Donald C. S. Allison and MT Noga. "Some performance tests of convex hull algorithms". In: *BIT Numerical Mathematics* 24 (1984), pp. 2–13 (cit. on pp. 56, 57).

[27] Renato Cordeiro de Amorim. "An Empirical Evaluation of Different Initializations on the Number of K-Means Iterations". *Advances in Artificial Intelligence.* Springer Berlin Heidelberg, 2013, pp. 15–26 (cit. on p. 20).

[28] David Arthur and Sergei Vassilvitskii. “How slow is the k-means method?” *Proceedings of the twenty-second annual symposium on Computational geometry*. 2006, pp. 144–153 (cit. on p. 57).

[29] Andleeb Aslam et al. “Improving K-Mean Method by Finding Initial Centroid Points”. *2020 22nd International Conference on Advanced Communication Technology (ICACT)*. 2020, pp. 624–627 (cit. on p. 22).

[30] Merhad Ay et al. “FC-Kmeans: Fixed-centered K-means algorithm”. In: *Expert Systems with Applications* 211 (2023), p. 118656 (cit. on p. 19).

[31] Sayan Bandyapadhyay et al. “Voronoi game on graphs”. In: *Theoretical Computer Science* 562 (2015), pp. 270–282 (cit. on p. 13).

[32] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. “The quickhull algorithm for convex hulls”. In: *ACM Transactions on Mathematical Software (TOMS)* 22.4 (1996), pp. 469–483 (cit. on p. 27).

[33] William J Baumol and Philip Wolfe. “A warehouse-location problem”. In: *Operations research* 6.2 (1958), pp. 252–263 (cit. on p. 60).

[34] Surbhi Bhatia. “New improved technique for initial cluster centers of K means clustering using Genetic Algorithm”. *International Conference for Convergence for Technology-2014*. 2014, pp. 1–4 (cit. on p. 22).

[35] Johannes Blömer et al. “Theoretical analysis of the k-means algorithm–a survey”. In: *Algorithm Engineering: Selected Results and Surveys* (2016), pp. 81–116 (cit. on p. 19).

[36] Álvaro Gutiérrez Cáceres. “Strategy in Geoguessr: Games of competitive Euclidean geometry” (2022) (cit. on p. 13).

[37] Shuang Chen, Junli Li, and Xiuying Wang. “A Fast Exact Euclidean Distance Transform Algorithm”. *2011 Sixth International Conference on Image and Graphics*. 2011, pp. 45–49 (cit. on p. 16).

[38] Matthew Díaz and Joseph O’Rourke. “Algorithms for computing the center of area of a convex polygon”. In: *The Visual Computer* 10.8 (1994), pp. 432–442 (cit. on p. 56).

[39] Loukas Dimitriou and Paraskevas Nikolaou. “Dynamic partitioning of urban road networks based on their topological and operational characteristics”. *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. IEEE. 2017, pp. 457–462 (cit. on p. 18).

[40] Zvi Drezner, Carlton Scott, and Jing-Sheng Song. “The central warehouse location problem revisited”. In: *IMA Journal of Management Mathematics* 14.4 (2003), pp. 321–336 (cit. on p. 60).

[41] Christina Durón. “Heatmap centrality: a new measure to identify super-spreader nodes in scale-free networks”. In: *Plos one* 15.7 (2020), e0235690 (cit. on p. 57).

[42] Murat Erisoglu, Nazif Calis, and Sadullah Sakallioglu. "A new algorithm for initial cluster centers in k-means algorithm". In: *Pattern Recognition Letters* 32.14 (2011), pp. 1701–1705 (cit. on p. 58).

[43] Aleta C Fabregas, Bobby D Gerardo, and Bartolome T Tanguilig III. "Enhanced initial centroids for k-means algorithm". In: *International Journal of Information Technology and Computer Science* 1 (2017), pp. 26–33 (cit. on p. 58).

[44] Linton C Freeman et al. "Centrality in social networks: Conceptual clarification". In: *Social network: critical concepts in sociology. Londres: Routledge* 1 (2002), pp. 238–263 (cit. on p. 16).

[45] Xuehong Gao. "A location-driven approach for warehouse location problem". In: *Journal of the Operational Research Society* 72.12 (2021), pp. 2735–2754 (cit. on p. 60).

[46] Mustafa Girgin. "Use of games in education: GeoGuessr in geography course". In: *International Technology and Education Journal* 1.1 (2017), pp. 1–6 (cit. on p. 11).

[47] Alexey Golubev et al. "Geospatial data generation and preprocessing tools for urban computing system development". In: *Procedia Computer Science* 101 (2016), pp. 217–226 (cit. on p. 18).

[48] Peyman Hashemi and Rahim Ali Abbaspour. "Assessment of logical consistency in OpenStreetMap based on the spatial similarity concept". In: *OpenStreetMap in GIScience: experiences, research, and applications* (2015), pp. 19–36 (cit. on p. 60).

[49] Christian Hedegaard Pliess Larsen. ""Exploring Hedging Strategies for Multiple Players in GeoGuessr"" (2022) (cit. on pp. 13, 14, 23, 29, 31, 32, 53).

[50] Marco Helbich et al. "Comparative spatial analysis of positional accuracy of OpenStreetMap and proprietary geodata". In: *Proceedings of GI_Forum* 4 (2012), p. 24 (cit. on p. 60).

[51] Dongyang Jiang, Wei Zheng, and Xiaoqing Lin. "Research on selection of initial center points based on improved K-means algorithm". *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology.* 2012, pp. 1146–1149 (cit. on p. 22).

[52] Leon Kaufman, Marc Vanden Eede, and Pierre Hansen. "A plant and warehouse location problem". In: *Journal of the Operational Research Society* 28.3 (1977), pp. 547–554 (cit. on p. 60).

[53] Nina Kudláčková and Jan Chocholáč. "Warehouse location problem in context of delivery time shortening". *MATEC Web of Conferences. Vol. 134 (2017): 18th International Scientific Conference-LOGI 2017.* EDP Sciences. 2017 (cit. on p. 60).

[54] Ina Ludwig, Angi Voss, and Maike Krause-Traudes. "A Comparison of the Street Networks of Navteq and OSM in Germany". In: *Advancing geoinformation science for a changing world* (2011), pp. 65–84 (cit. on p. 60).

[55] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. "The planar k-means problem is NP-hard". In: *Theoretical Computer Science* 442 (2012), pp. 13–21 (cit. on p. 57).

[56] Natarajan Meghanathan. "Use of Centrality Metrics to Determine Connected Dominating Sets for Real-World Network Graphs". *2015 12th International Conference on Information Technology - New Generations.* 2015, pp. 243–248 (cit. on p. 16).

[57] Natarajan Meghanathan and Raven Lawrence. "Centrality analysis of the United States network graph". *3rd International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences (EEECOS 2016).* 2016, pp. 1–6 (cit. on p. 16).

[58] Sneha Mehta, Chris North, and Kurt Luther. "An exploratory study of human performance in image geolocation tasks". *HCOMP 2016 GroupSight Workshop on Human Computation for Image and Video Analysis.* Vol. 308. 2016 (cit. on pp. 13, 55).

[59] Tim Menzner, Florian Mittag, and Jochen L Leidner. "Which Country Is This? Automatic Country Ranking of Street View Photos". *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III.* Springer. 2023, pp. 275–280 (cit. on pp. 12, 13, 55, 56, 62).

[60] Richard Tran Mills et al. "Parallel k-means clustering of geospatial data sets using manycore CPU architectures". *2018 IEEE international conference on data mining workshops (ICDMW).* IEEE. 2018, pp. 787–794 (cit. on p. 17).

[61] Peter Mooney, Padraig Corcoran, and Adam C Winstanley. "Towards quality metrics for OpenStreetMap". *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems.* 2010, pp. 514–517 (cit. on p. 60).

[62] Shi Na, Liu Xumin, and Guan Yong. "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm". *2010 Third International Symposium on Intelligent Information Technology and Security Informatics.* 2010, pp. 63–67 (cit. on p. 22).

[63] Fereshteh-Azadi Parand, Hossein Rahimi, and Mohsen Gorzin. "Combining fuzzy logic and eigenvector centrality measure in social network analysis". In: *Physica A: Statistical Mechanics and its Applications* 459 (2016), pp. 24–31 (cit. on p. 16).

[64] Wojciech Pokojski et al. "Educational computer games in geography". In: *Edukacja biologiczna i środowiskowa* 1 (2017), pp. 50–56 (cit. on p. 11).

[65] C Carl Robusto. "The cosine-haversine formula". In: *The American Mathematical Monthly* 64.1 (1957), pp. 38–40 (cit. on p. 25).

[66] Fátima Ferrão dos Santos. "Text Data Mining in a Geospatial Metadata Catalog" (2015) (cit. on p. 17).

[67] Robert Stanforth, Evgueni Kolossov, and Boris Mirkin. “Hybrid k-means: Combining regression-wise and centroid-based criteria for QSAR”. In: *Selected Contributions in Data Analysis and Classification* (2007), pp. 225–233 (cit. on p. 18).

[68] Sudharshan Suresh, Nathaniel Chodosh, and Montiel Abello. “DeepGeo: Photo localization with deep neural network”. In: *arXiv preprint arXiv:1810.03077* (2018) (cit. on pp. 11, 12, 55, 56, 62).

[69] Yujing Wang et al. “Analysing the spatial configuration of urban bus networks based on the geospatial network analysis method”. In: *Cities* 96 (2020), p. 102406 (cit. on pp. 15, 16).

[70] Tobias Weyand, Ilya Kostrikov, and James Philbin. “PlaNet-photo geolocation with convolutional neural networks”. *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*. Springer. 2016, pp. 37–55 (cit. on pp. 12, 55, 56, 62).

[71] Edy Winarno, Wiwien Hadikurniawati, and Rendy Nusa Rosso. “Location based service for presence system using haversine method”. *2017 international conference on innovative and creative information technology (ICITech)*. IEEE. 2017, pp. 1–4 (cit. on p. 58).

[72] Jyoti Yadav and Monika Sharma. “A Review of K-mean Algorithm”. In: *Int. J. Eng. Trends Technol* 4.7 (2013), pp. 2972–2976 (cit. on p. 57).

[73] Madhu Yedla, Srinivasa Rao Pathakota, and TM Srinivasa. “Enhancing K-means clustering algorithm with improved initial center”. In: *International Journal of computer science and information technologies* 1.2 (2010), pp. 121–125 (cit. on p. 58).

[74] Qilong Yuan, Haibo Shi, and Xiaofeng Zhou. “An optimized initialization center K-means clustering algorithm based on density”. *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. 2015, pp. 790–794 (cit. on p. 22).

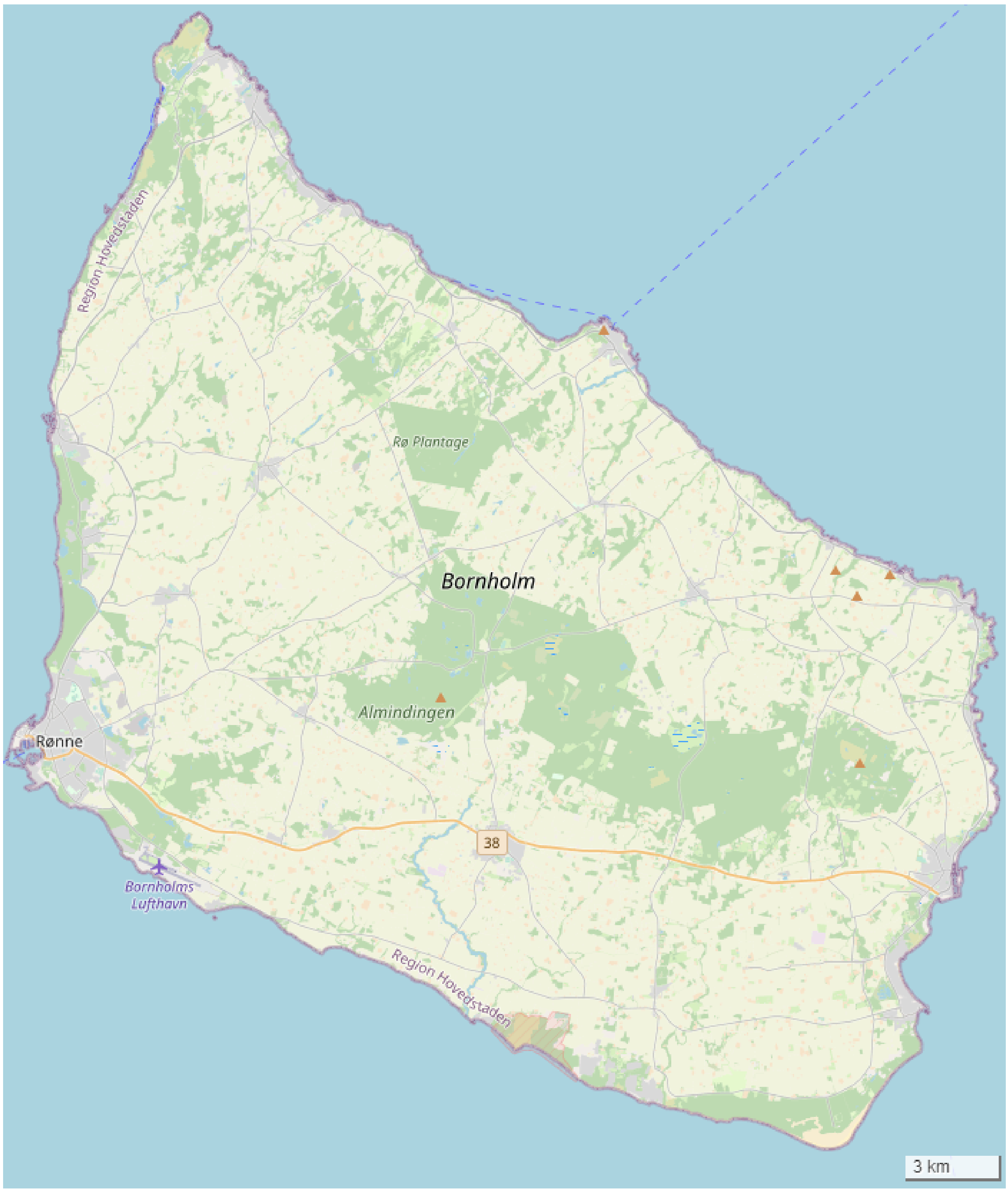
[75] Junlong Zhang and Yu Luo. “Degree centrality, betweenness centrality, and closeness centrality in social network”. *2017 2nd international conference on modelling, simulation and applied mathematics (MSAM2017)*. Atlantis press. 2017, pp. 300–303 (cit. on p. 16).

[76] Lanlan Zhang et al. “Improvement of K-means algorithm based on density”. *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. 2019, pp. 1070–1073 (cit. on p. 22).

Appendices

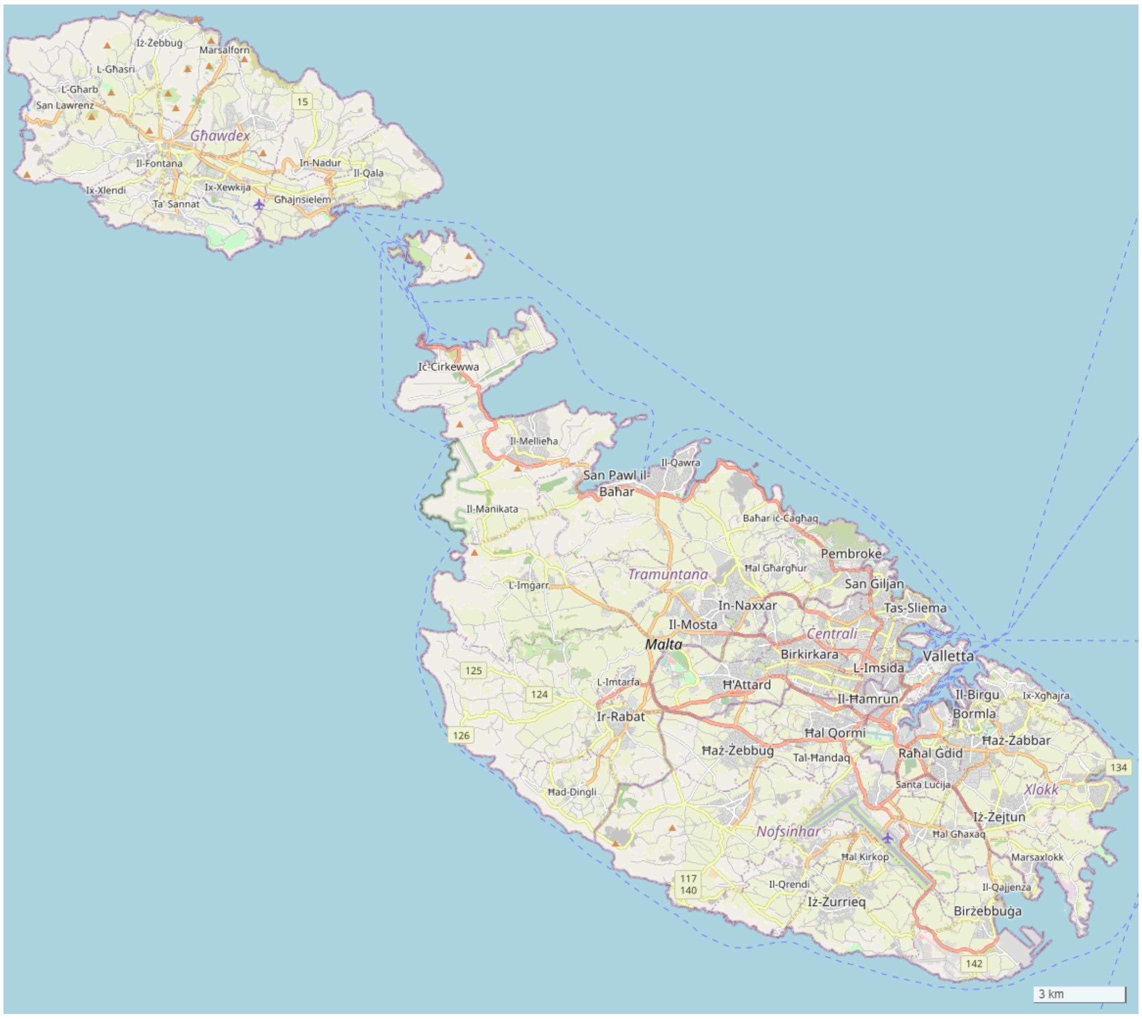
A Map of Bornholm

Source: [7]



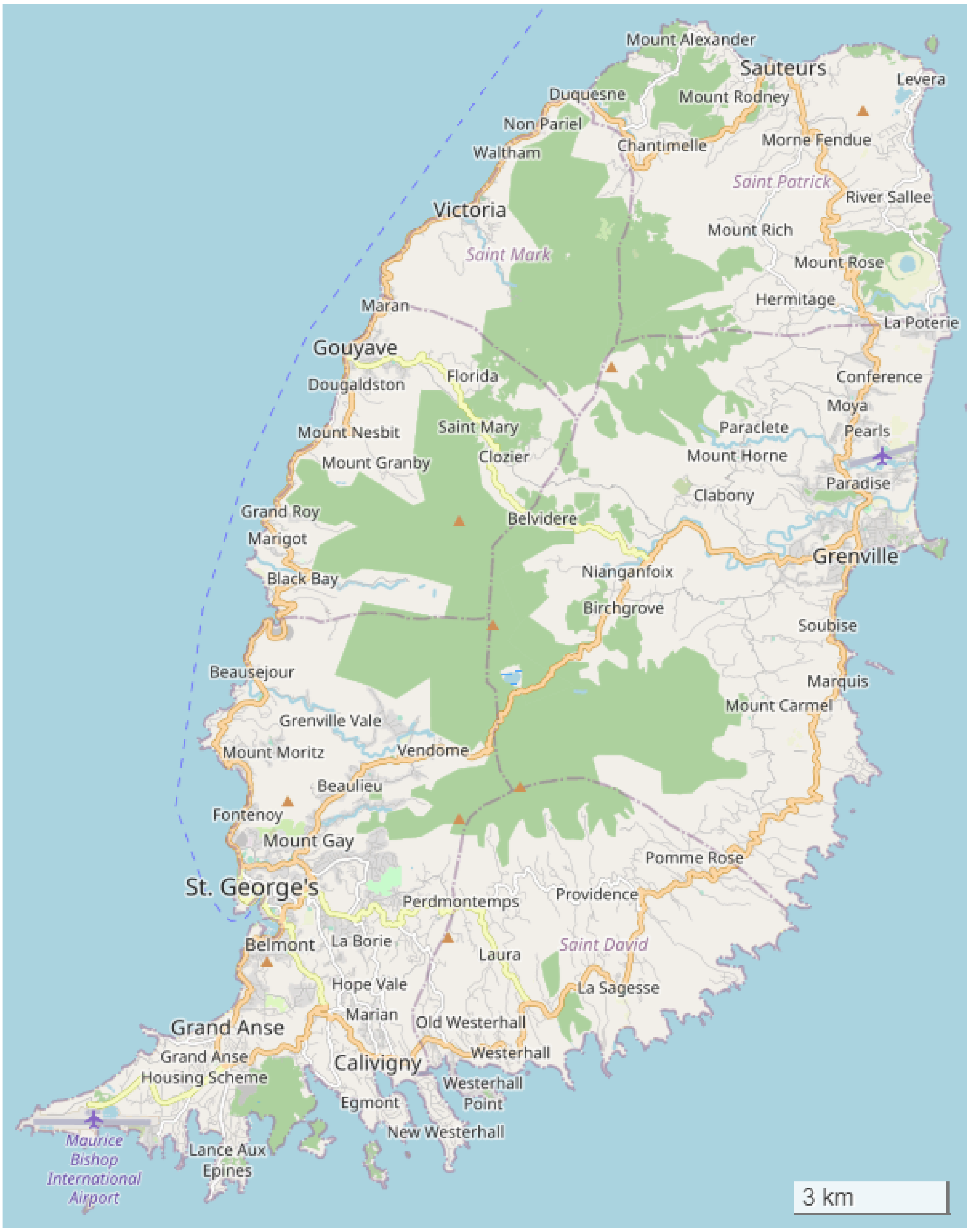
B Map of Malta

Source: [9]

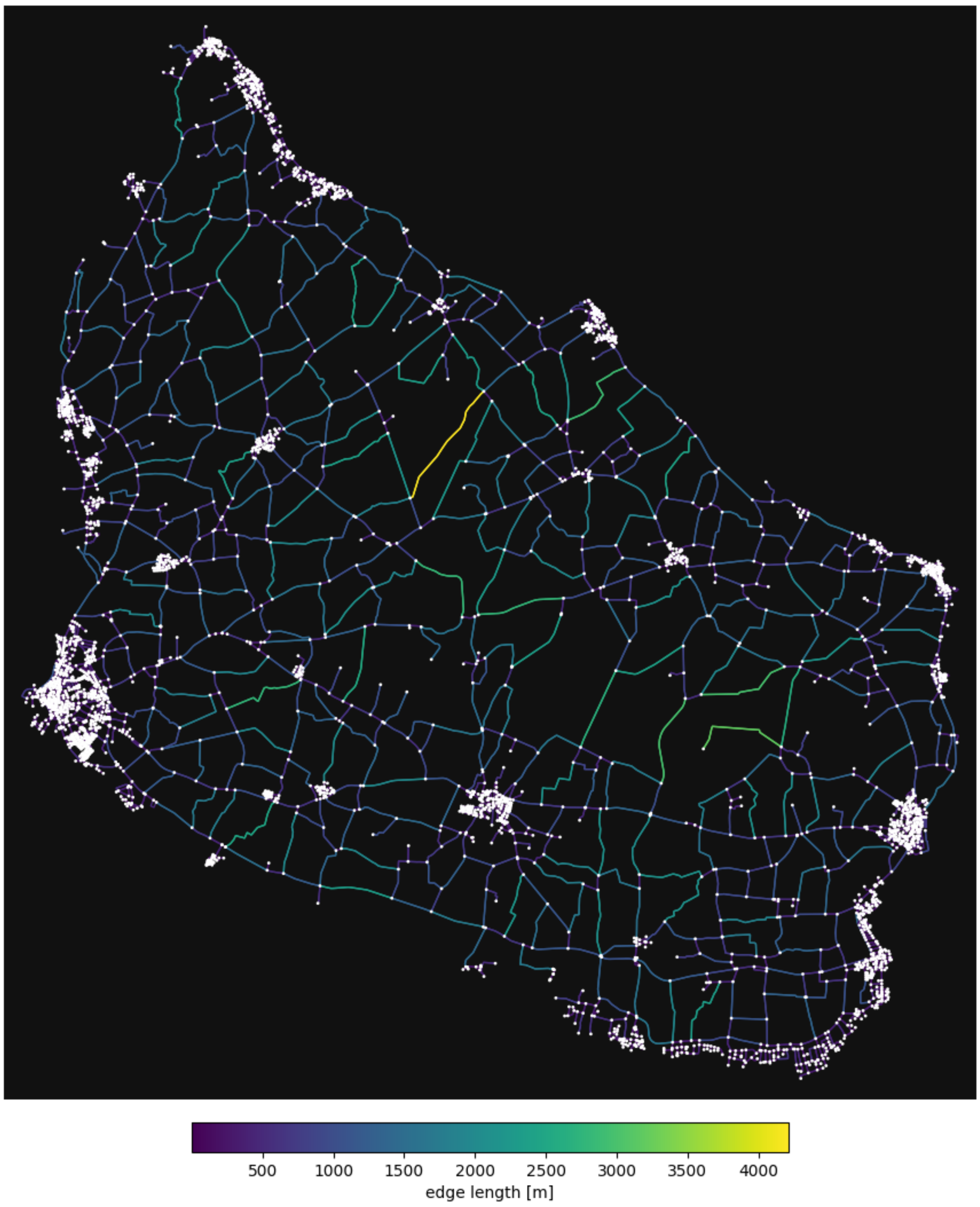


C Map of Grenada (main island)

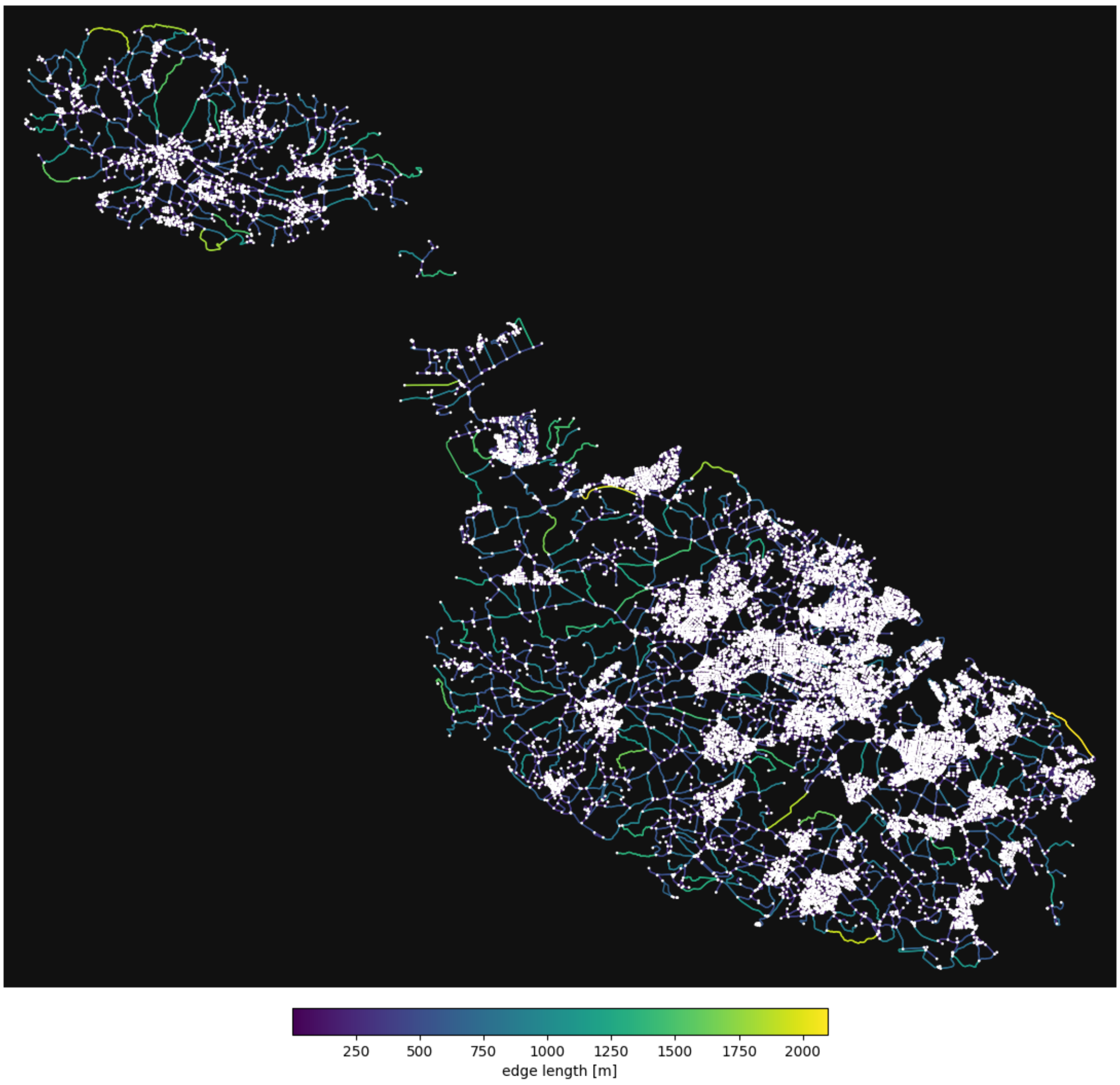
Source: [8]



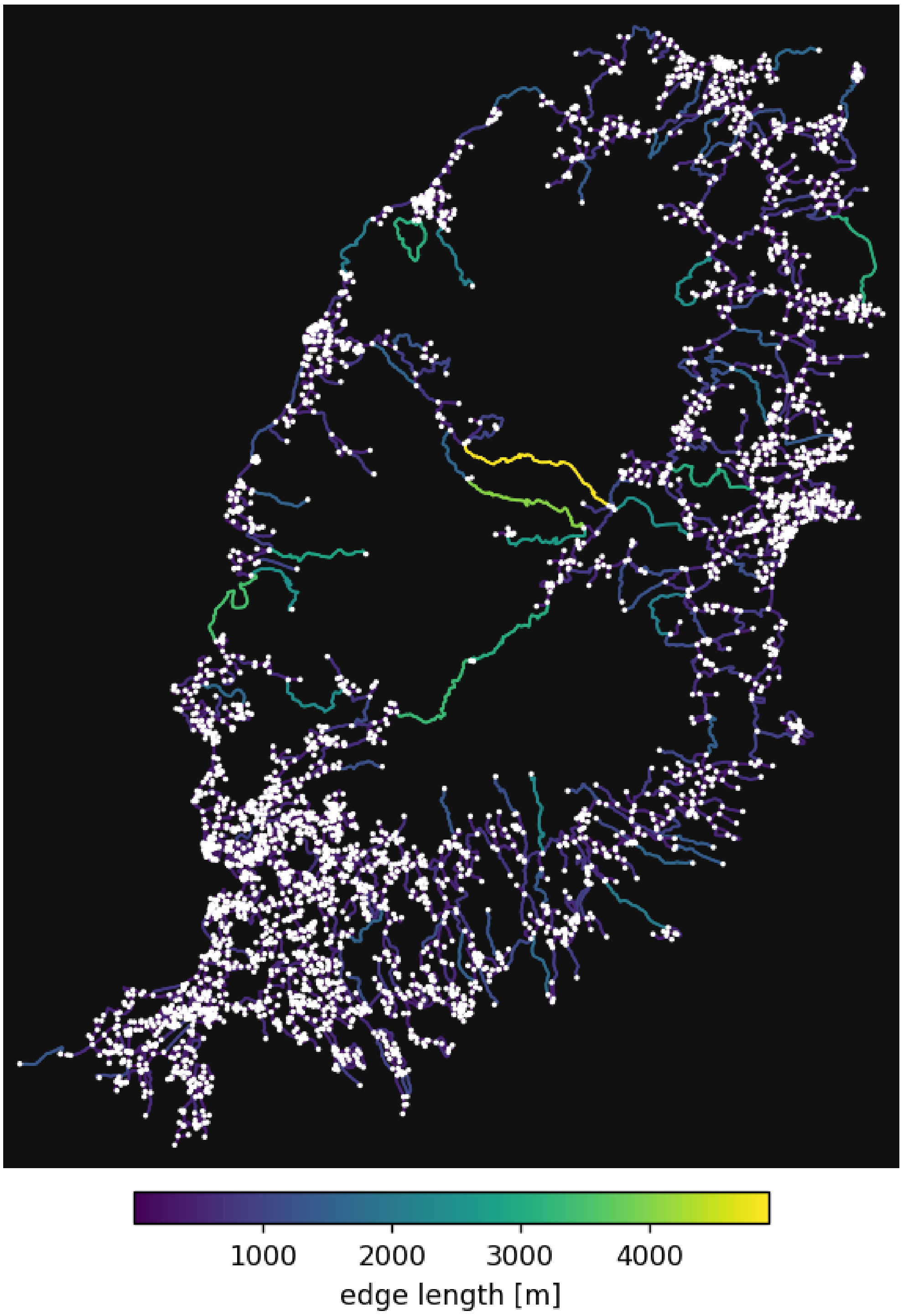
D Map of Bornholm: Driveable Network



E Map of Malta: Driveable Network



F Map of Grenada (main island): Driveable Network



1000
2000
3000
4000
edge length [m]