Reinventing the Territory Optimizer

What makes a Balanced Aesthetic?

Derek van den Elsen

2580100

September 12, 2018

Master Thesis Report Business Analytics

VU, Faculty of Science:

De Boelelaan 1085
1081 HV Amsterdam
The Netherlands

Irving van Heuven van Staereling
René Bekker

ORTEC B.V.:

Houtsingel 5
2719 EA Zoetermeer
The Netherlands

Arjan Hennink

(This page intentionally left blank)

Preface

This internship report is written as part of the Master's degree Business Analytics at the Vrije Universiteit Amsterdam. This is a two-year multidisciplinary program, aimed at improving business processes by applying a combination of methods based on mathematics, computer science and business management. The Master's program is concluded with a six-month individual internship at a business or research institute. The main objective of this internship is to research and analyze a specific problem affecting the business of the host organization. I conducted my research at host organization ORTEC at the ORTEC Optimization Technology department under the routing team. The work within this report concerns improvements to the Territory Optimizer, which is a module within ORTEC Tactical Routing. The Territory Optimizer geographically groups clients in balanced clusters to simplify a periodic vehicle routing problem.

The past six months have been a very challenging and invaluable experience. ORTEC has taught me a lot every single day and the process has been particularly wonderful. Foremost, I would like to thank my ORTEC supervisor, Arjan Hennink, who has been incredibly involved throughout this process and has served as a mentor to me at ORTEC. Second, I would like to thank Irving van Heuven van Staereling, under the wing of Rob van der Mei, for helping me through any academic pitfalls as my academic supervisor and first reader. Third, I would like to give thanks to Renè Bekker for functioning as my second reader. Finally, I would like to thank all my colleagues at ORTEC that welcomed me with open arms and made me feel right at home. Special shoutout to my fellow interns Danny and Liana. The memories to the board game nights, karaoke and even a trip to Romania won't be forgotten, but cherished dearly.

Derek van den Elsen - Zoetermeer, September 2018

(This page intentionally left blank)

Summary

This Master's Thesis brings us into the realm of weight balanced clustering with geographical distances. This can be applied in the periodic vehicle routing problem by clustering clients, assigning trucks to clusters in advance and then running a routing algorithm. ORTEC's Territory Optimizer is responsible for this clustering and the main goal of this thesis is to provide a state-of-the-art alternative to it. Clients of a specific truck should be located close together geographically, as to minimize the travel time necessary to visit each client. Similarly, giving trucks a relatively equal amount of weight to carry is preferred to huge workload imbalances. This brings us the main objectives of balance and a good aesthetic.

Consequently, this report will discuss several validation measures for both geographical clustering quality and balancedness. A self-made geographical clustering validation measure is the neighbourhood ratio, which checks what fraction of points in the neighbourhood of a point is in the same cluster. Balancedness is measured through the standard deviation, minimum and maximum of the cluster size distribution in relation to the mean.

A combination of several heuristics working together is used within this report to solve the weight balanced clustering with geographical distances. Of prime importance is Charity Search, which aims to improve balance by transferring a client from the most overloaded cluster to the most underloaded cluster, while maintaining a nice geographical aesthetic. Charity Search finds a path in clusters using Breadth-First Search and then continuously transfers the client with the least average distance to the subsequent cluster on that path.

Twenty two real-life distinct datasets are analyzed and prepared to test the aforementioned method. The main data analysis result is that there is typically not an insignificant number of extremely large clients. Notwithstanding, Charity Search takes between an instant and 10 minutes for the largest cases. For all cases the balance requirements were also reached even when they were as tight as 1% and the visual pictures of the map generally showed little to no flaws, which is supported by excellent neighbourhood ratios in the 0.80's with 0.8275 as average for all cases.

In conclusion this thesis develops an iteration-based, fast, intuitive, setting-appropriate alternative in Charity Search to ORTEC's current solution to the weight balanced clustering problem with geographical distances that can work also in conjunction with other methods.

Contents

Chapter 1

Introduction

ORTEC is one of the world's leaders in optimization software and analytics solutions. ORTEC makes businesses more efficient, more predictable and more effective, turning complex challenges into easy-to-use solutions. ORTEC serves clients in almost every industry. They deliver solutions on a global scale, always underpinned by local know-how and services with offices strategically located across the continents.

One of ORTEC's products is ORTEC Tactical Routing (OTR) that solves the periodic vehicle routing problem over a long planning horizon with tons of optional additional constraints like spatial dimension limitations on the vehicle, vehicle type acceptance for clients and lunch breaks for drivers. A solution to this problem is a full logistic plan over what truck goes where over several months. OTR solves this problem into steps, starting with the Territory Optimizer, which allocates resources that can stand for trucks or distribution centers for instance towards orders. This allocation will result in specific geographical territories or clusters that are meant to decompose the overall problem in load balanced digestible chunks. Each territory will be assigned a resource and will solely be served by that resource. This is necessary, because the periodic vehicle routing problem is complex and it also has the added benefit that clients will deal with the same distributor, tightening connections with the customer. The subsequent modules: Visit Day Optimizer and Route Scheduling are standalone and work in tandem with the territories provided by the Territory Optimizer to create a full planning, detailing where every resource goes and at what time within their assigned territory.

The main purpose of this thesis is to overhaul the Territory Optimizer, because the current one is not behaving optimally. On first glance the territories in Figure 1.1 that are from a real customer case look of near equal size and are definite separate entities, which is exactly as desired. Note that the green is from two different territories, because OTR is limited in the number of differing colors it can show.

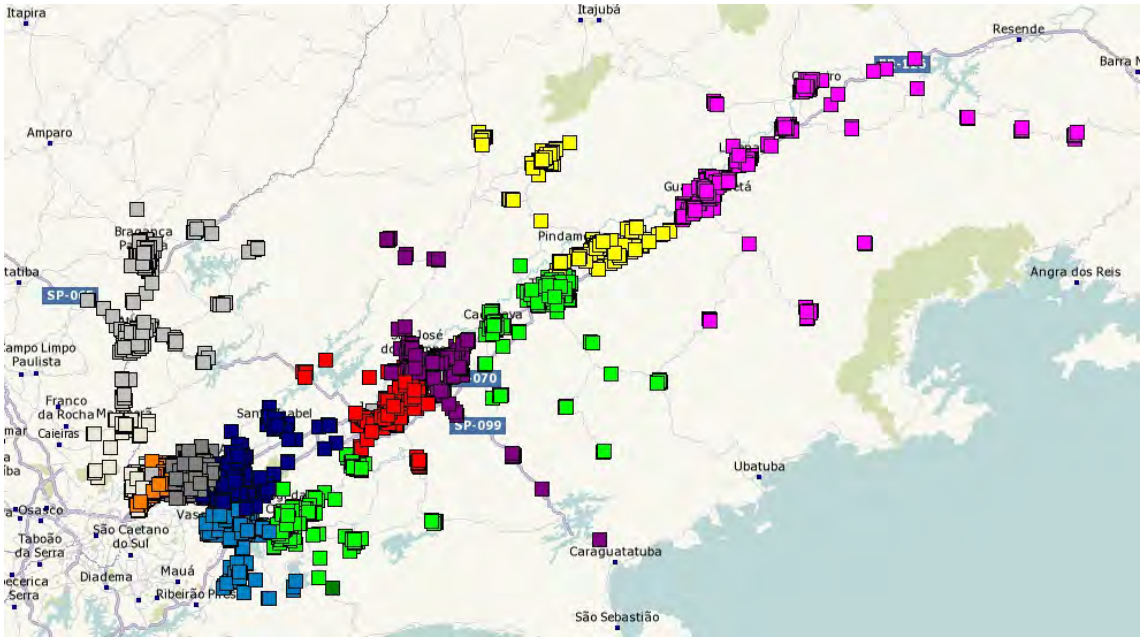


Figure 1.1: Result Territory Optimizer

On a closer look in Figure 1.2 however, we find out some yellow stragglers are nowhere near their territory that is off-screen and are surrounded by many members of other territories. This is hard to explain to prospective buyers of the software and is a mistake that OTR does not recognize yet.

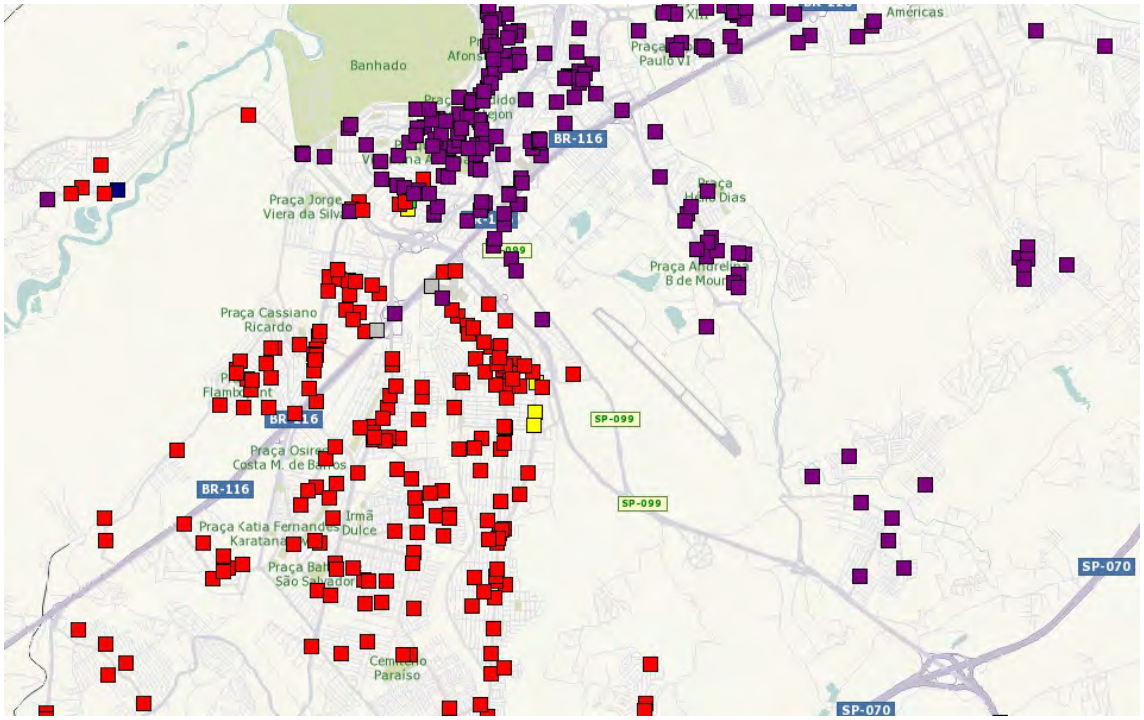


Figure 1.2: Zoom Shot Result Territory Optimizer

Hence the primary goal of this thesis is to find out how to efficiently make load balanced geographical clusters of weighted demand nodes in a logistics setting. It is ambiguous and subjective what

constitutes a good solution as it is hard to quantify when a clustering is good and ultimately that that clustering leads to a good solution for the periodic vehicle routing problem.

This thesis commences by going deeper into the problem statement. Second, a literature review is given about the Vehicle Routing Problem and general clustering separated into partitional, hierarchical, density-based and grid-based clustering archetypes. Some applications of balanced clustering are also discussed. Third, the methodology section discusses the set of key performance indicators used throughout this research, briefly the original ORTEC approach and a set of algorithms, namely Metachrosis, Stretchy Search and Charity Search that aim to improve on ORTEC's approach. Their interplay is also visualized in a flowchart. Fourth, A data analysis delves into the available data. How it is cleaned, a global overview and a more specific examination of three cases, Poland, USA II and Stockholm are present here. Fifth, results follow with a similar layout to the data analysis. Last, a conclusion summarizing the main points and main results made within this thesis.

Chapter 2

Problem Statement

The problem is weight-balanced clustering in a geographical plane. How to make certain groupings, such that the points within groupings are close together and far away from points in other groupings in terms of distance over the road, such that the combined weight of any grouping is approximately equal to other groupings. These two objectives are competing, so we constrain the solution space to only produce balanced solutions. The metric that is supposed to model the quality of the clustering and the overall aesthetic is subjective and is obscured in the objective function $f(X, D)$, where X is the point assignment and D the real pairwise distance matrix. Minimal overlapping clusters is cited to be one of the primary goals. As input we have a set of points with a weight, w_i, attached to each point. We also have the number of points as N and the number of groupings as M. b is the maximum relative percentage deviation from the cluster size mean that represents our balance requirements. As output we have an assignment of all points to some cluster.

$$\min_x Z = f(X, D)$$

$$\text{s.t.} \sum_{j=1}^{M} x_{ij} = 1 \qquad \forall i = 1, 2, .., N$$

$$\frac{\sum_{i=1}^{N} w_i}{M} \cdot (1 - \frac{b}{100}) \leq \sum_{i=1}^{N} w_i \cdot x_{ij} \leq \frac{\sum_{i=1}^{N} w_i}{M} \cdot (1 + \frac{b}{100}) \qquad \forall j = 1, 2, ..., M$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j$$

N	number of points
M	number of clusters
x_{ij}	= 1 if client i is assigned to cluster j, = 0 otherwise
b	balance margin in percentages
w_i	weight of point i

We are not interested in statistical clustering, where obtaining a model that follows the data-generating process is important, but merely the grouping of data. Finding out why a client would lie in a certain place and where it could have been in an alternate universe, where inherent randomness would roll differently, is a philosophical thought question that is not of our concern. Our data being geographical coordinates is not largely an issue to apply most of the standard clustering techniques successfully. The validation of said clustering is however much different and the measures used do not rely on "unseen" or generated new data. We merely need internal instead of external measure criteria, which will be further described in subsection 4.1.1

The problem within this thesis is specifically distinct from traditional clustering, because, for instance, the clustering is only done in two dimensions. The distance metric is of utmost importance and should be taken as the main drive, as geographical boundaries are not taken into account in just x and y coordinates. In many other clustering problems, finding an appropriate distance metric is actually a big part of the problem. Especially when dealing with the 'curse of dimensionality', where a high number of dimensions provides difficulties for computation and storage of distance measures.

Another thing to consider is that the pairwise distances are our only available metric, so the plane as a whole is relatively unknown. This is on a practical level the case, because repeated calls to the ORTEC Map and Route server are computationally expensive and not feasible on such a large scale. This is precisely so, because the number of pairwise distances that need to be calculated is quadratic in the number of locations. We deal with geographical boundaries like bodies of water, mountains, constrained infrastructure that make this vastly different from a simple Euclidean plane. This implies that for instance, an approach with Voronoi diagrams as seen in Figure 2.1 is impossible, as we do not know the distance from any arbitrary point to our known points. A Voronoi diagram is a partition of a space such that each part is closest to the point on which that part was constructed, so every line segment is closest to two points and every line intersection is closest to three points. That could usually be helpful to cluster accordingly, but we do not have this method available.

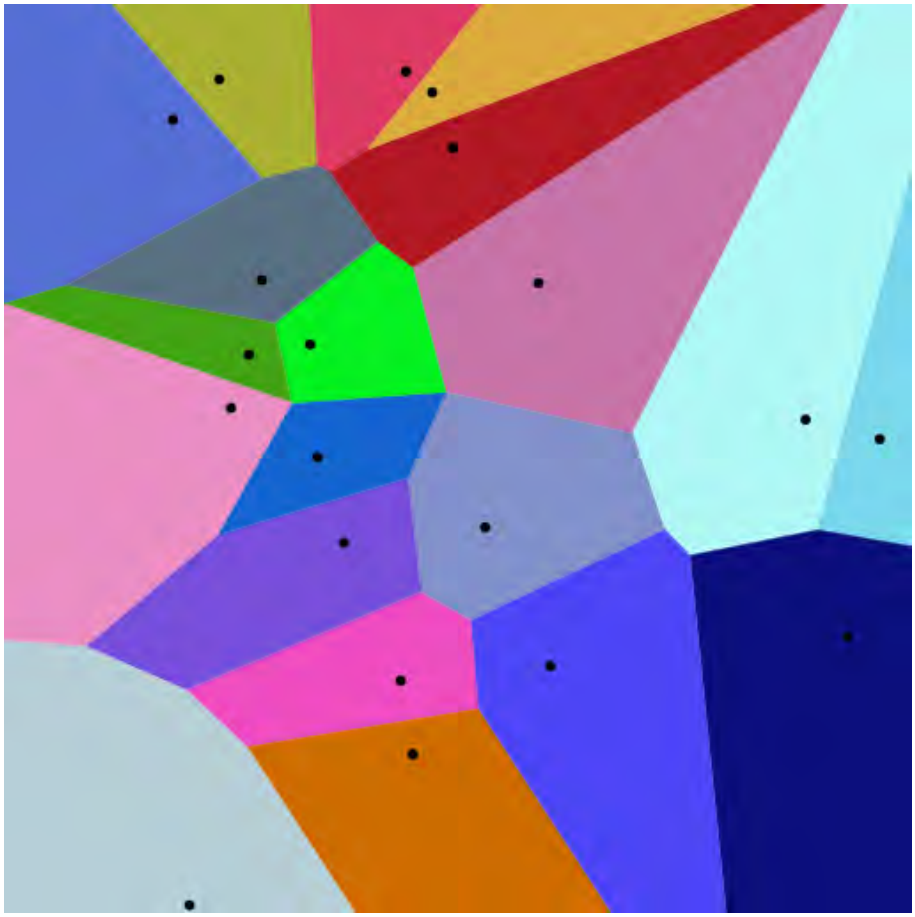


Figure 2.1: Euclidean distance Voronoi diagram

Furthermore, the points in the plane may be spread out in any which manner, giving us varying point group densities that do not always appear in traditional clustering, where observations are typically gathered from the same, or at least similar, data-generating process. These differing point group densities are obviously incorporated in most of our data, as we have big concentrated cities and small rural villages that have contrasting needs. The difference in point group densities, makes focusing just on the distance a bad strategy, since you will encounter precisely less points in a less dense area for the same amount of distance by definition. Finally, the weight distribution can be of any arbitrary form, giving us either near unweighted balanced clustering or a more skewed distribution, where the difference is more stark.

Chapter 3

Literature Review

This literature review will first briefly talk about the Vehicle Routing Problem, because the Territory Optimizer fits the solution paradigm of Cluster-first/route-second to solve the Vehicle Routing Problem and is responsible for the clustering part. Second, it will go into general clustering, where first some applications are named and then 4 different paradigms are discussed in turn, namely partitional, hierarchical, density-based and grid-based clustering. Third, it will give some practical applications of why balanced clustering in particular is an interesting problem.

3.1 VRP

The Vehicle Routing Problem (VRP) was first described in 1959 and has since then exploded with a myriad of extensions (Dantzig and Ramser, 1959). The goal of the classical VRP is to design a set of delivery routes for a fleet of vehicles from a central depot to a set of demand points each having known service requirements that minimizes the total distance covered by the vehicle fleet. All vehicles have fixed capacity and must start and finish their routes at the central depot. Each customer is served by only one vehicle, which must satisfy that customer's entire demand in one visit.

Solution strategies for the VRP can be roughly divided into 5 categories (Bowerman et al., 1994). Cluster-first/route-second are methods that first assign demand nodes into clusters and then make individual routings per cluster. Route-first/cluster-second are strategies that first make a giant tour that visits every demand node exactly once and then break up this tour into routes that satisfy the capacity constraints of the vehicles. Savings/insertion are procedures that make an initial configuration of nodes and incrementally improve upon this configuration by looking at alternatives and picking the one that yields the largest increase in objective function. Improvement/exchange are methods that operate on some set of initial routes and intend to improve them through swapping arcs. Finally we have mathematical programming, which typically optimally solves a relaxation of the linear program. One formulation of a linear program is listed next:

$$\min_x Z = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{V} c_{ij} \cdot x_{ijk}$$

$$\begin{aligned}
\text{s.t.} \sum_{i=1}^{N}\sum_{k=1}^{V} x_{ijk} &= 1 && \text{for } j = 1, 2, ..., N-1 \\
\sum_{i=1}^{N} x_{ihk} - \sum_{j=1}^{N} x_{hjk} &= 0 && \text{for } k = 1, 2, ..., V \text{ and } \quad h = 1, 2, ..., N \\
\sum_{i=1}^{N} Q_i \cdot \sum_{j=1}^{N} x_{ijk} &\leq P_k && \text{for } k = 1, 2, ..., V \\
\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij} \cdot x_{ijk} &\leq T_k && \text{for } k = 1, 2, ..., V \\
\sum_{j=1}^{N-1} x_{Njk} &\leq 1 && \text{for } k = 1, 2, ..., V \\
y_i - y_j + N x_{ijk} &\leq N - 1 && \text{for } 1 \leq i \neq j \leq N-1, \quad 1 \leq k \leq V \\
x_{ijk} &= 0 \text{ or } 1 && \forall i, j, k
\end{aligned}$$

V	number of vehicles
P_k	capacity of vehicle k
T_k	maximum cost allowed for a route of vehicle k
Q_i	demand at node i
c_{ij}	cost of travelling from location i to location j
y_i	arbitrary real number for location i
x_{ijk}	= 1 if pair (i,j) is in the route of vehicle k, = 0 otherwise

The first two constraints ensure that each customer is served by one and only one vehicle and routes are continuous. The third constraint embodies that vehicle capacity is not exceeded. Fourth constraint bounds total routes. Fifth constraint ensures vehicle availability is not exceeded. Sixth constraint are the subtourbreaking constraints. Finally the decision variables must be binary (Kulkarni and Bhave, 1985).

3.2 Clustering

Clustering is ubiquitous and occurs in many areas as for example: biology, medicine, business, computer science, social science and robotics. In biology, clustering is applied to establish a sibling relationship of a population from genetic data for example (Chou et al., 2011). In medicine, on Positron-emission tomography (PET) scans, cluster analysis is one of the tools in differentiating

between different categories of tissues in a three-dimensional image, which can be employed to detect abnormal brain structures for instance (Filipovych et al., 2011). In business, clustering can facilitate searching through business processes in repositories (Ordoñez et al., 2017). In computer science, one application of clustering is in forensic image source acquisition from mobile phones (Villalba and Corripio, 2015). In social sciences, several crime data sets like details of every terrorist attack since 1970, a Chicago police dataset with details of every drug-related incident over a month and a dataset describing members of a Hezbollah terror network within the U.S. were clustered together and appropriately visualized to discover patterns (Skillicorn and Leuprecht, 2015). In robotics, clustering is used to detect breakpoints in laser scanning for the navigation problem (Fernandez et al., 2010).

Clustering can most definitely not be solved using complete enumeration, where all possible clusterings are compared, aside from extremely limited sample sizes. Not only is computing what constitutes a decent clustering quite expensive, there are the following number of distinct possible clusterings, where n is the number of points to be clustered and k the number of clusters necessary (Jain and Dubes, 1988) :

$$S(n,k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^{k-i}\binom{k}{i}i^n$$

$S(n,k)$ are also referred to as Stirling numbers of the second kind. For more context, dividing 10 points over 4 clusters can be done in 34.105 distinct ways or $S(10,4) = 34105$. This number rapidly expands with increasing n as there are approximately 11.259.666.000 possible partitions when dividing 19 points over 4 clusters or $S(19,4) = 11.259.666.000$, so we must clearly find other smarter ways to construct solutions.

Clustering is very broadly applied and has many possible solution methodologies. Clustering algorithms can be mainly divided in partitional, hierarchical, density-based and grid-based clustering (Varghese et al., 2013).

3.3 Partitional

Partitional clustering algorithms initially partition the dataset creating the same number of partitions as clusters necessary and improve upon this partition. Three sub-categories can be made: k-means, k-medoids and probabilistic clustering (Berkhin, 2006).

3.3.1 K-means

One of the most popular and oldest clustering algorithms, k-means, is a partition clustering algorithm. K-means attempts to minimize the within-cluster sum of squares or variance, which is equivalent to minimizing the pairwise in-cluster squared deviation and maximizing the between-cluster sum of squares (Kriegel and Schubert, 2016). This problem is however unfortunately NP-hard in many of its possible problem formulations (Drineas et al., 1999).

Most commonly it is solved by initializing a set of centers and assigning every point to its closest center. A center can be seen as a gravitational point containing all the mass of a cluster. After

one iteration the centroids are recalculated until convergence is reached and the centers do not change a lot anymore. An illustration demonstrating the algorithm is shown in Figure 3.1. It occurs so frequently that this is usually referred to as the k-means algorithm or alternatively Lloyd's algorithm. Sometimes, Voronoi diagrams are used to assign points to clusters and this utilizes the whole geometric space, instead of merely a discrete set of points (Lloyd, 1982).

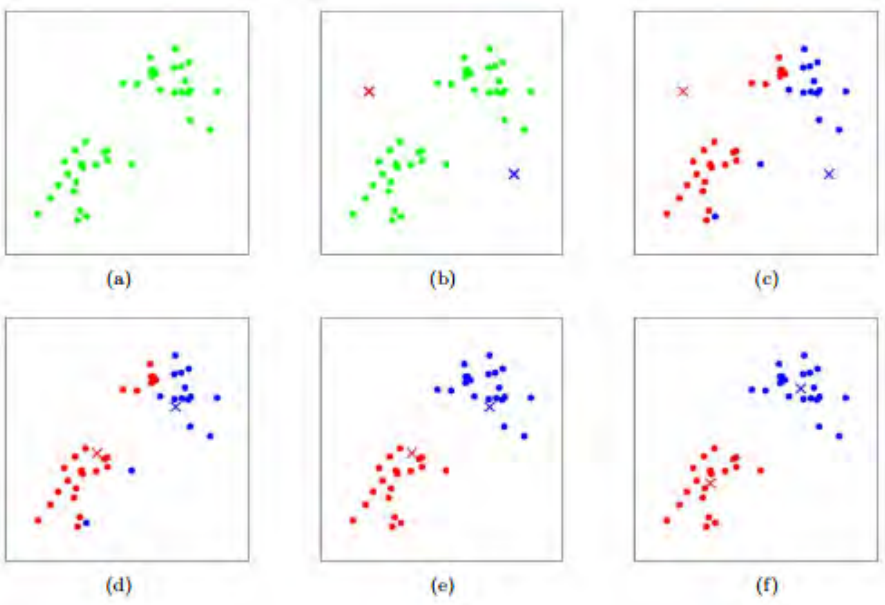


Figure 3.1: K-means stages

The initialization schemes can also vary. Prevalent is Forgy's method, where k observations are randomly selected from the data to be the first centers or random partition, where the first clusters are made by purely random selection and the centers become the centroids of those initially selected random clusters (Elkan and Hamerly, 2002). Another option is Farthest-first traversal, where the first center is chosen randomly and every iteration the point that is farthest from all other currently selected centers is added (Rosenkrantz et al., 1977). Finally, we have kmeans++, where the initial center is chosen randomly and then new centers are added iteratively. Each new center is chosen from all points that are left with a weighted probability distribution proportional to the distance to a point's closest center, skewing the probability towards points further away from centers. This ensures the initial centers are more spread out (Arthur and Vassilvitskii, 2006).

K-means is not particularly well suited for our problem as there is nothing that ensures balance. In actual practice it is frequently the case that it produces clusters that are near-empty and by complement clusters that are very big. Centers anywhere in the plane are inherently flawed with our problem statement, as we only have the pairwise distances between each point. A centers approach by itself is already flawed, as we deal with geographical boundaries that obscure a conventional distance method like Euclidean distance, with which this algorithm typically works. Its simplicity, similar need to an input for the number of clusters, low computational complexity and ubiquitous appearance in the clustering literature do make it an attractive option to draw inspiration from.

3.3.2 K-medoids

K-medoids is a partitional algorithm, unsurprisingly, very similar to k-means. The main difference is that the centers can now only be represented by actual observations instead of any arbitrary point in the space and are called medoids. This is a lot more appropriate than k-means when intermediate points in the space do not mean anything like for instance with categorical variables.

One of the most basic and well-known algorithms for k-medoids is Partitioning Around Medoids (PAM). PAM is a greedy search that finds a local optimum. It has similar initialization schemes to k-means and afterwards assigns every point to its closest medoid. While the sum of pairwise distances is going down, it attempts to swap every medoid with a non-medoid. If the swap does not result in a lower sum of pairwise distances it does not not go through with the swap and goes on to the next medoid. The algorithm terminates after no positive swap can be found for all medoids or a different stopping criterion is reached (Kaufman and Rousseeuw, 1990).

Clustering LARge Applications (CLARA) draws a sample of the dataset and runs PAM on that to find the appropriate medoids. After finding the medoids on the sample, the unsampled leftovers are simply clustered to their closest medoids. There is also Large Applications based on RANdomized Search (CLARANS), which samples in the solution space instead of the original observations space and randomly finds a number of neighbours instead of a complete local search (Raymond and Han, 1994).

K-mediods is near identically suited to our problem as k-means is, however it does not hold the drawback that any arbitrary point in the plane can be a center, since we only have access to the interpoint distances. Intermediate points in the plane do have more meaning than for instance categorical variables for which k-medoids is frequently used. There is significantly more computational complexity involved in the classic k-medoids algorithms however.

3.3.3 Probabilistic

In the probabilistic viewpoint of clustering, the data is considered to be a sample independently drawn from a mixture of several probability distributions (McLachlan and Basford, 1988). It is assumed that data is generated by randomly picking a model j with probability $\pi_j, j = 1, ..., k$. Areas around the mean of a certain distribution naturally represents a cluster, so a cluster's characteristics are akin to the distribution's mean, variance etc. Some point x belongs only to a singular cluster with probability $Pr(C_j|x)$, then the overall likelihood is:

$$Pr(X|C) = \prod_{i=1}^{n} \sum_{i=1}^{k} \pi_j Pr(x_i|C_j)$$

The log-likelihood serves as an objective function and the expectation maximization algorithm (EM) is designed for this purpose. EM works by alternating between performing an expectation (E) step, which estimates $Pr(x|C_j)$ using the current estimate for the parameters and a maximization (M) step, which computes the parameters maximizing the expected log-likelihood found on the E step. These estimates are then used to determine the distribution of the latent variables in the next E step. This iterative process continues until the log-likelihood converges (Dempster et al., 1977).

Probabilistic clustering is an extremely ill-fitting approach towards our problem, because our clustering does not value finding out the data-generating process, but merely wants to produce balanced groupings that have points in the same grouping closely together and far away from points in other groupings.

3.4 Hierarchical

Hierarchical clustering produces a hierarchy of clusters within a dendogram, one of which is visualised in Figure 3.2. Hierarchical clustering can mainly be done in an agglomerative (bottom-up) or divisive (top-down) manner. In agglomerative clustering every point starts as a cluster and clusters are iteratively combined until there is one big singular cluster. In divisive clustering all points start as a big cluster and clusters are recursively split in two separate clusters until all points are a separate cluster.

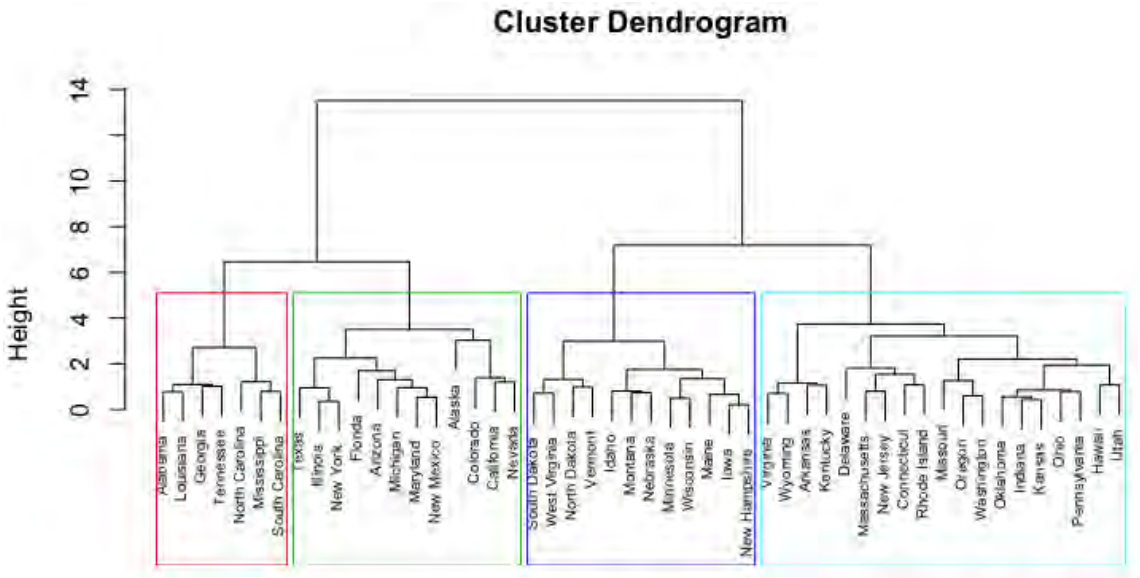


Figure 3.2: Dendogram US States

The height of each node represents the dissimilarity. The longer it takes for a merge to occur the less similar the two children nodes are. A horizontal cut at any point in the dendogram can give a certain number of clusters. If the number of clusters is not known beforehand, it is generally a good idea to horizontally cut, wherever there is a large jump to the next split. The cut for 4 clusters as indicated by the boxes is shown in Figure 3.3.

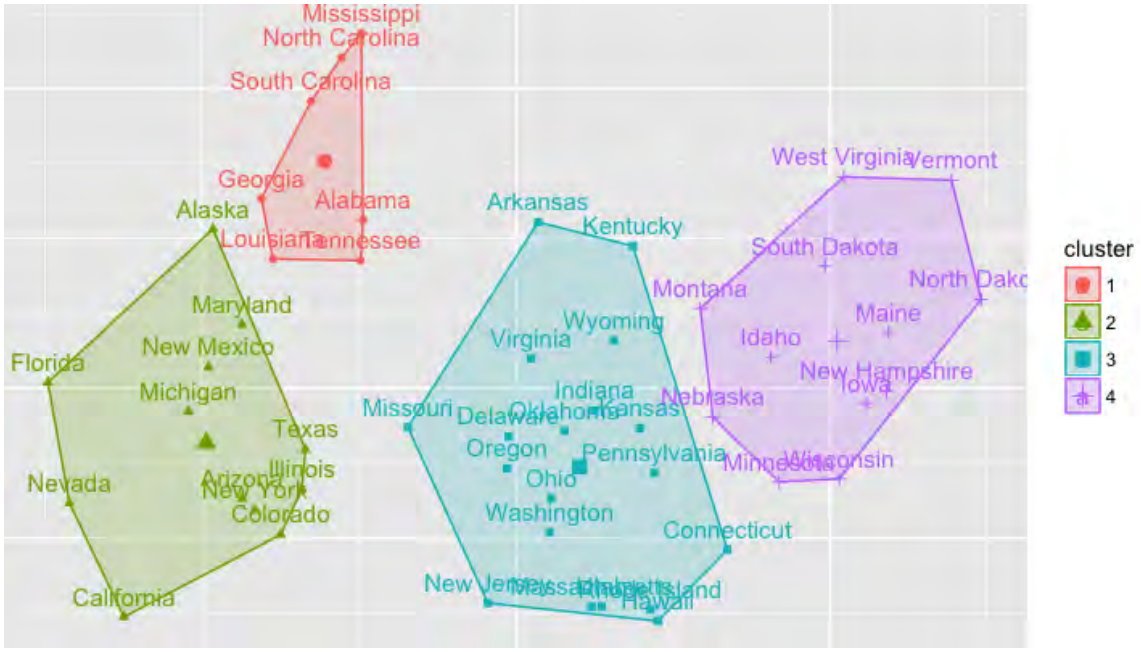


Figure 3.3: US States Clustering, k = 4

3.4.1 Agglomerative

In agglomerative clustering, every point starts as a cluster and they are merged iteratively until only a big cluster remains. This is done greedily by selecting the lowest distance at that iteration. The distance between singular points is usually well-defined as Euclidean distance or a similar metric, however the distance between multiple points to multiple points is less straightforward. To accommodate this, we have the Lance-Williams formula:

$$d(C_i \cup C_j, C_k) = \alpha_i d(C_i, C_k) + \alpha_j d(C_j, C_k) + \beta d(C_i, C_j) + \gamma |d(C_i, C_k) - d(C_j, C_k)|$$

Certain parameter sets representing a certain link type are showcased in Table 3.1 (Murtagh and Contreras, 2011). The last three link types are distinct in that they are distances between some geometrically defined central point.

Table 3.1: Lance-William parameters

Link Type	α_i	β	γ
Single	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete	$\frac{1}{2}$	0	$\frac{1}{2}$
Group Average	$\frac{\lvert C_i \rvert}{\lvert C_i \rvert + \lvert C_j \rvert}$	0	0
Weighted Average	$\frac{1}{2}$	0	0
Centroid	$\frac{\lvert C_i \rvert}{\lvert C_i \rvert + \lvert C_j \rvert}$	$-\frac{\lvert C_i \rvert \lvert C_j \rvert}{(\lvert C_i \rvert + \lvert C_j \rvert)^2}$	0
Median	$\frac{1}{2}$	$-\frac{1}{4}$	0
Minimum Variance (Ward, 1963)	$\frac{\lvert C_i \rvert + \lvert C_k \rvert}{\lvert C_i \rvert + \lvert C_j \rvert + \lvert C_k \rvert}$	$-\frac{\lvert C_k \rvert}{\lvert C_i \rvert + \lvert C_j \rvert + \lvert C_k \rvert}$	0

These parameter sets might look daunting, but some are straightforward if portrayed differently. In words, single linkage implies that clusters are connected through the minimum value of all distances between their individual points. Single linkage corresponds to $d(C_i \cup C_j, C_k) = \min\{d(C_i, C_k), d(C_j, C_k)\}$ proof:

$$d(C_i \cup C_j, C_k) = \frac{1}{2}d(C_i, C_k) + \frac{1}{2}d(C_j, C_k) - \frac{1}{2}|d(C_i, C_k) - d(C_j, C_k)|$$

if $d(C_i, C_k) < d(C_j, C_k)$

$$d(C_i \cup C_j, C_k) = \frac{1}{2}(d(C_i, C_k) + d(C_j, C_k) - d(C_j, C_k) + d(C_i, C_k))$$

$$d(C_i \cup C_j, C_k) = \frac{1}{2}(2 \cdot d(C_i, C_k)) = d(C_i, C_k)$$

if $d(C_j, C_k) < d(C_i, C_k)$

$$d(C_i \cup C_j, C_k) = \frac{1}{2}(d(C_i, C_k) + d(C_j, C_k) - d(C_i, C_k) + d(C_j, C_k))$$

$$d(C_i \cup C_j, C_k) = \frac{1}{2}(2 \cdot d(C_j, C_k)) = d(C_j, C_k)$$

if $d(C_i, C_k) = d(C_j, C_k)$

$$d(C_i \cup C_j, C_k) = \frac{1}{2}(d(C_i, C_k) + d(C_i, C_k) - 0) = d(C_i, C_k)$$

□

Naive approaches to implementing the several linkage types, where the full distance matrix is simply updated and recalculated for the particular merge happening, leads to $O(n^3)$ running time and $O(n^2)$ space requirements. However more efficient implementations exist that have $O(n^2)$ running time and an $O(n)$ space requirements. SLINK does so for single linkage (Sibson, 1972). CLINK for complete linkage (Defays, 1977). The following algorithm for Group average linkage (Voorhees, 1986). Finally, Weighted Average, Centroid and Median are contained here (Day and Edelsbrunner, 1984).

Clustering Using REpresentatives (CURE) is a more sophisticated hierarchical agglomerative clustering algorithm that is more robust to outliers or non-spherically shaped natural clusters than traditional algorithms. It represents clusters through a specific number of fixed points, aptly called representatives, that are generated through selecting points that are spread out well and then shrinking them towards the center by a certain fraction. The clusters whose representative points are closest to each other are merged at every iteration. Having multiple representatives guards against non-spherically shaped natural clusters and shrinking them towards the center counters the disturbance caused by outliers. Furthermore, CURE maintains scalability through a combination of random sampling and partitioning in steps (Guha et al., 2001). Robust Clustering algorithm for Categorical Data (ROCK) is, as the name suggests, designed for categorical data. It is very much like CURE, but uses a different measure for calculating the distance to deal with the different nature of the data (Guha et al., 2000).

3.4.2 Divisive

There are far less divisive hierarchical clustering algorithms relative to agglomerative hierarchical clustering algorithms. This mostly has to do with the computational expense, which in the agglomerative case boils down to $\frac{n(n-1)}{2}$ pairs to compare per step, whereas in the divisive case, there are $2^{n-1}-1$ possible divisions per step to consider. The former is quadratic, whereas the latter is exponential, which is increasingly more important for larger n.

There are two important decisions to make in a divisive clustering algorithm. What cluster will be split during an iteration and how will it be split. The former is typically either split in a prespecified order, randomly, on highest cardinality or highest load or based on a intracluster distance measure. It is important that this quality is monotone, meaning that a subcluster will not have a higher value than its parent. This is in particular true for the diameter or maximum distance within a cluster, which DIvisive ANAlysis (DIANA) uses as splitting criterion.

DIANA starts of all points as one big cluster. At each iteration DIANA calculates the average distance to other objects in a cluster. The one with maximum average distance starts of a splinter group and is removed from the original cluster. The average distance is recalculated for all remaining points in the original cluster, to the splinter group and also the other points in the original cluster. The difference between these average distances is computed and the maximum difference joins the splinter group, as long as it is a positive difference. If no such positive difference exists, the iteration is complete and this division will be made. The next cluster that will be split is the one that has the largest diameter. This process continues until all clusters are singletons (Kaufman and Rousseeuw, 1990).

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is another hierarchical algorithm, but it is specifically designed to handle large volumes of data. BIRCH only requires a single scan of the whole database. It saves local information, indicating what cluster a point belongs to, in a height-balanced tree and updates it as it encounters more data. The balancedness mention refers to the tree that is made and is largely different from the context within this thesis, where we mean the size of the clusters having balance restrictions (Zhang et al., 1996).

Hierarchical clustering is very much suited towards our problem needs, as the focus is entirely on the pairwise distance measure, which is of prime importance, as the geographical boundaries are accounted for accordingly in that. Its complexity is sufficient for our problem and it is an easy to understand approach. Varying point group densities might pose a problem for this approach, yielding highly imbalanced solutions. Modifications to make it more balanced are immediately apparent, however the question is if they can make it balanced it enough, while maintaining the appealing geographical layout. The linkage functions used within hierarchical clustering also have an intuitive appeal, however several are not suited towards our problem structure, like the Centroid linkage function.

3.5 Density-Based

In density-based clustering, clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density or noise. These regions can have an arbitrary shape and the points inside regions may be arbitrarily distributed.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is the primogenitor of density-based clustering algorithms. DBSCAN has two main parameters: ϵ and `MinPoints`. ϵ dictates when a point is considered close enough to another and belongs to their neighborhood. `MinPoints` determines the number of points necessary to be in a point's neighborhood before it is considered dense enough. A point is a core point if it has more than `MinPoints` that are less than ϵ away. Otherwise it is a border point. DBSCAN defines clusters as one core point combined with all points that are density reachable from it with respect to the two parameters ϵ and `MinPoints`. The exact notion of density reachable is better explored in the paper itself. DBSCAN starts by simply selecting a random point. If that is a border point, pick a new random point until a core point has been found. Upon finding a core point, form a cluster with all points that are density reachable from this point with respect to parameters ϵ and `MinPoints`. Continue this procedure until all the data has been clustered or declared as noise (Ester et al., 1996).

Generalized Density-Based Spatial Clustering of Applications with Noise (GDBSCAN) is, straightforwardly, a generalization of the aforementioned DBSCAN. It allows clustering of more extensive figures as opposed to points in space and it can attach a weight to certain observations to determine density as opposed to simple cardinality (Sander et al., 1998).

Ordering Points To Identify the Clustering Structure (OPTICS) is another generalization of DBSCAN that largely abstracts away the ϵ parameter. OPTICS creates an ordering of the points that allows the extraction of clusters with arbitrary values for ϵ. It produces a reachability graph as seen on a synthetic dataset in Figure 3.4, which has the ordering the algorithm produces on the x-axis and the reachability distance on the y-axis. The reachability distance of a point is the smallest distance such that the point is density-reachable from a core point. Clusters can be identified as valleys in the reachability graph and noise is in between clear valleys. What constitutes a valley depends on the choice of ϵ (Ankerst et al., 1999).

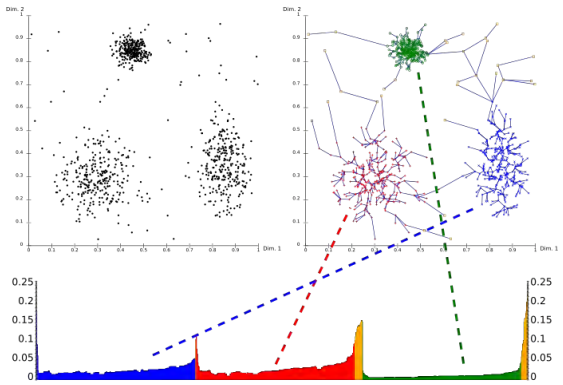


Figure 3.4: Reachability Graph

Density-Based clustering is not suited for our problem for several reasons. The number of clusters it finds is not controlled. It may attribute certain points as outliers who have no cluster affiliation and in our case every point needs to be assigned a cluster. Additionally, the `MinPoints` and ϵ parameters need to be predetermined somehow. There is no notion of balance and not a convenient way to incorporate it either. The premise of density-based clustering is that fault lines between clusters are simple transitions of point group densities, which means that enormously crowded cities will always be one cluster, instead of the several clusters required. At best, Density-Based clustering gives us some insight in, where the densities vary specifically, but this is usually already quite clear visually.

3.6 Grid-Based

Grid-Based clustering algorithms typically quantize the space into a discrete number of cells and continue to do all their operations on the quantized space. This usually leads to low computation times, because it depends on the number of cells and not the number of points. All grid-based approaches unfortunately suffer from the modifiable areal cell problem (Openshaw, 1977). This problem arises in terms of scaling and then aggregating information. The problem is in determining the appropriate number and size of cells to represent the data. There are an infinite amount of ways this can be done. In general, if a very coarse quantization is applied, more points will be applied to the same cell, and more importantly there is a higher probability that points from different clusters will be assigned the same cell. This results in merging of multiple naturally individual clusters. Conversely, if a very fine quantization is applied, many cells will have no, or very few points, so that also proper clusters cannot be identified. Aggregation refers to the process of summarizing the information of all points in a particular cell.

STatistical INformation Grid-based (STING) is a method that divides the spatial area in rectangles using a hierarchy. The hierarchical structure is showcased in Figure 3.5. Per cell the following attribute-dependent information is saved: mean, standard deviation, minimum, maximum, estimated distribution type. Clustering is performed in a top-down manner, starting with the root. Relevant cells are determined through the statistical information and only the paths through relevant cells are followed down the tree. The clusters are formed at the leaf level by performing a breadth-first search that merges cells based on whether the average density of the area is greater than some parameter and their proximity. It is interesting to note that this procedure approximates DBSCAN in the limit of making cells arbitrarily small (Wang et al., 1997).

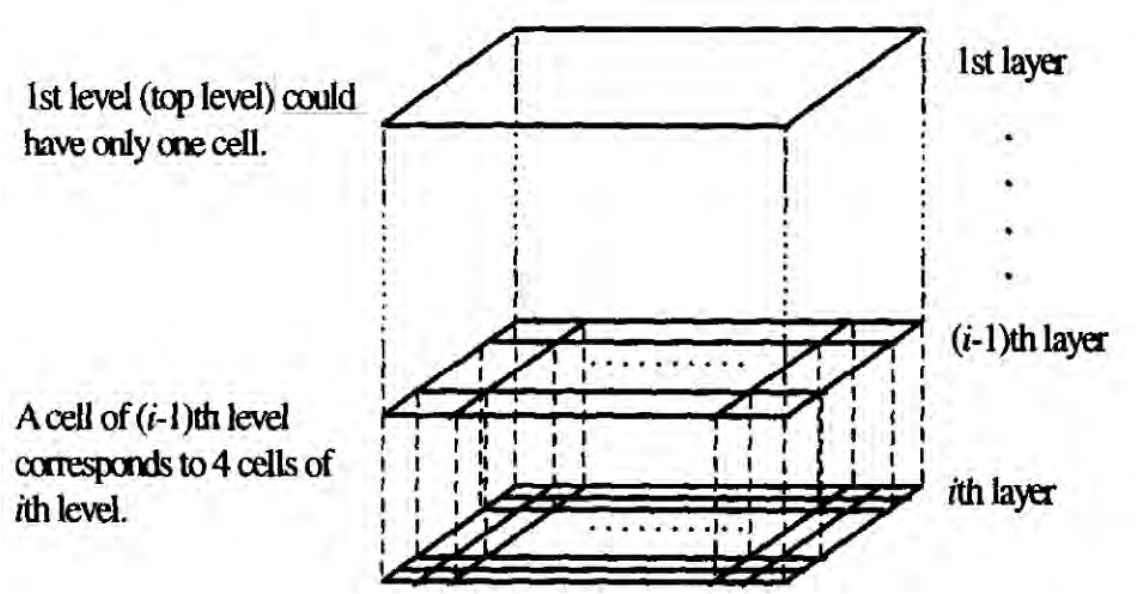


Figure 3.5: Hierarchical Structure

WaveCluster is a multi resolution algorithm which finds clusters in very large spatial databases. It summarizes the data by imposing a multidimensional grid structure to the data space. Towards this purpose, it transforms the original data by applying wavelet transform and then finding the dense regions in the new space (Sheikholeslami et al., 2000).

CLustering In QUEst (CLIQUE) partitions the dataspace into non-overlapping sufficiently dense rectangular units. These dense units are then connected to form clusters. Given the required parameters and a set of data points, CLIQUE can find clusters in all subspaces of the original data space and is able to present a minimal description for each cluster (Agrawal et al., 1998).

MOSAIC, referring to small subclusters making up a larger more sophisticated cluster, is a clustering algorithm that utilizes Gabriel graphs to determine the neighborhood of clusters. A Gabriel graph is a graph in the Euclidean plane, where vertices are connected if the closed disc, of which the line segment between the vertices is the diameter, contains no other vertices (Gabriel and Sokal, 1969). One such graph is visualized in Figure 3.6. MOSAIC uses agglomerative clustering to greedily merge the small subclusters, already obtained through another representative based algorithm, locally optimizing some given fitness function like the Silhouette Coefficient (Choo et al., 2011).

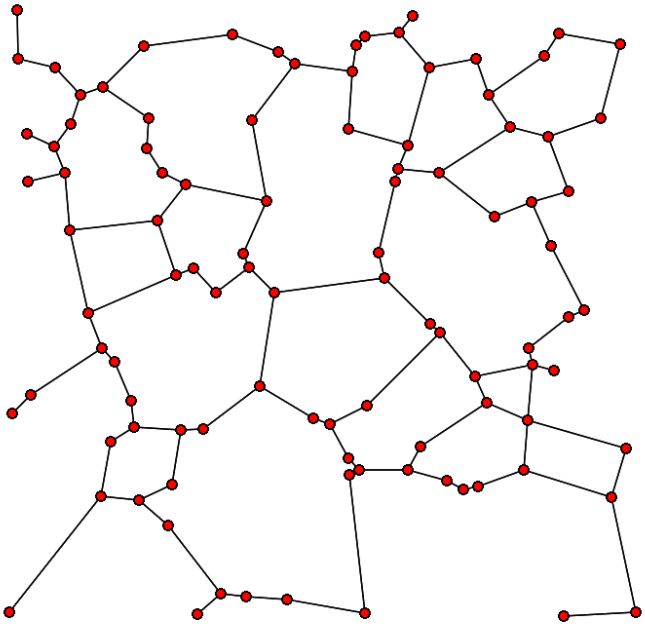


Figure 3.6: Gabriel Graph

Grid-Based clustering is not easily translatable to our problem definition, because it basically scales down the problem to gain a good solution locally, which is then scaled up again, however our balance objective is global. Usually grid-based clustering is applied whenever the full sample space is too vast, however that is also not the case in our problem structure. The grid cells created can also not be of an uniform size, as the plane is not even. Geographical boundaries are the caveat in that regard.

3.7 Balanced Clustering

In general, the natural clusters in the data may be of widely varying sizes. Moreover, this variation may not be known beforehand and balanced solutions may not be important. Typically however having empty or very small clusters is undesirable. Conversely having near all-encompassing clusters is equally fruitless. For several real-life applications, having a balancing requirement helps in making the clusters actually useful and actionable. Thus this imposition largely comes from the associated business needs rather than from the inherent properties of the data. Indeed, it may be specified simply based on the end-goals even without examining the actual data. Some of the uses of balanced clustering are now described with applications in Direct Marketing, Document Clustering, Category Management, Energy Aware Sensor Networks and Logistics.

3.7.1 Direct Marketing

A direct marketing campaign generally divides the customer base into several segments of roughly equal size or equal estimated revenue generation. This estimation can be done based on for example demographical characteristics, market basket analysis or purchase history on the web. It is important

that it is near equal, so that the same amount of sales workforce, marketing budget or attention can be allocated to each segment. These clusters can usually represent a specific user archetype that simplifies decisions regarding up-selling, cross-selling or promotional opportunities (Strehl and Ghosh, 2003).

3.7.2 Document Clustering

Generating topic hierarchies from an enormous corpus of documents is not without its challenges in clustering appropriately. Balancing immensely facilitates browsing or navigation by avoiding highly skewed hierarchies, with uneven depth in different parts of the hierarchy tree or leaf nodes that have a widely varying number of documents. Similar principles apply when grouping articles in a web site (Lynch and Horton, 2016), portal design, and creation of domain specific ontologies by hierarchically grouping concepts (Baeza-Yates and Ribeiro-Neto, 1999).

3.7.3 Category Management

Category management is a process that entails managing product categories as business units and adapting them to satisfy customer needs ensuring that individual store characteristics are taken into account. Especially for large retailers, one of the core operations within category management is to group products into categories of a specified size such that they correspond to units of floor or shelf space. Another important application of category management is to cluster affiliated stock keeping units in bundles of equivalent revenue or profit. In both applications the clustering needs to be updated to reflect promotions, differences in consumer behaviour, seasonal changes and whims of the market, so this is an on-going process (Gupta and Ghosh, 2001a,b)

3.7.4 Energy Aware Sensor Networks

Wireless sensor networks have the potential to be utilized in a variety of military, civil and scientific applications. An important restriction is that generally the sensors need to rely on solely the on-board energy supply and as such it is of paramount importance that the network's energy is efficiently managed in order to extend the life of the sensors. Commonplace individual sensors are unable to support long haul communication to reach remote command sites and thus require some local gateway or transmitter to transfer their data on their behalf. These gateways form the cluster heads or centers and it is also important that the distance between these cluster heads or the remote site is kept at a minimum to retain maximum efficiency, making this differ from usual clustering problems. It is viewed as desirable to have cluster heads serve a sensor network that consumes comparable amounts of power, as this makes the overall network more reliable, sustainable and scalable, hence having a balanced load per sensor head is advantageous (Ghiasi et al., 2002) (Gupta and Younis, 2003).

Chapter 4

Methodology

First, some key performance indicators (KPIs) that deal with clustering quality and balance are defined and discussed. Second, some background information on how ORTEC currently handles the Territory Optimizer is given. Third, several algorithms will be discussed starting with hierarchical clustering with size-sensitive distance. This is followed by center-based methods after which a method for finding cluster neighbourhoods is described. Consequently, Metachrosis, which is a local search function optimizing clustering quality, Stretchy Search, which is a local search function optimizing balance, while maintaining geography and Charity Search, which is an extension of Stretchy Search are explained and discussed. Finally, an overview of the complete algorithmic process is showcased.

4.1 KPIs

4.1.1 Clustering Quality

What constitutes a good or bad clustering? Typically a good cluster has its members be close or be similar to members of the same cluster and far away or be dissimilar to members of other clusters. The former is referred to as cohesion and a lower intracluster distance is equivalent with a better cohesion. The latter is defined as separation and a higher intercluster distance leads to a better separation. In a good clustering this would apply to a majority of the clusters. Singleton clusters and all-encompassing clusters are also special cases that need to be taken into account.

A really important application of clustering is pattern recognition or datamining, where generally extrinsic validation criteria are used. Extrinsic validation criteria require a priori knowledge of what cluster a point belongs to. Intrinsic validation criteria solely use the distance matrix in order to validate the clustering. Points belonging to the same cluster have low distance between them and points belonging to different clusters have high distance between them (Jain and Dubes, 1988).

List of Symbols:

n_C	number of clusters
n	number of clients
n_k	number of clients in cluster k
G_k	centroid of cluster k
L_i	location of client i
$d(L_i, L_j)$	distance between client i and client j
$\sigma(C_k)$	intracluster distance of cluster k
$\delta(C_k, C_{k'})$	intercluster distance between cluster k and k'

As distance is important for the cluster quality, we formally define it here for the 2 dimensions we deal with. The Minkowski or p-norm distance (Deza and Deza, 2016):

$$d(L_1, L_2) = (|L_{1_x} - L_{2_x}|^p + |L_{1_y} - L_{2_y}|^p)^{\frac{1}{p}}$$

for p = 2, this is the well known Euclidean distance.

$$d(L_1, L_2) = (|L_{1_x} - L_{2_x}|^2 + |L_{1_y} - L_{2_y}|^2)^{\frac{1}{2}}$$

for p = 1, this is known as Manhattan distance.

$$d(L_1, L_2) = |L_{1_x} - L_{2_x}| + |L_{1_y} - L_{2_y}|$$

for p tending to infinity, this is named Chebyshev distance.

$$d(L_1, L_2) = \max\{|L_{1_x} - L_{2_x}|, |L_{1_y} - L_{2_y}|\}$$

However, we are not dealing with points in a simple two-dimensional Cartesian coordinate system, but with real places on earth, which contrary to thankfully unpopular belief, is not flat, but an ellipsoid, which may be approximated through a sphere. It is hard to quantify the exact error that results from this, however we can be sure that it is too significant for our application. One more appropriate way to look at distance on a sphere is the Haversine distance.

$$\mathrm{d}(\mathrm{L}_1, L_2) = 2r_{earth} \arcsin\left(\sqrt{\sin^2\left(\frac{L_{2_{Lat}} + L_{1_{Lat}}}{2}\right) + \cos(L_{1_{Lat}})\cos(L_{2_{Lat}})\sin^2\left(\frac{L_{2_{Lon}} + L_{1_{Lon}}}{2}\right)}\right)$$

However, this is also not entirely accurate as the earth is not a perfect sphere. The "Earth radius" r_{earth} varies from 6356.752 km at the poles to 6378.137 km at the equator. More importantly, the radius of curvature of a north-south line on the earth's surface is 1% greater at the poles (≈ 6399.594 km) than at the equator (≈ 6335.439 km), so the Haversine formula cannot be guaranteed correct to be better than 0.5%, depending on latitude and direction of travel. (Veness, 2017) Instead, we will rely on ORTEC Map and Route to produce more accurate distances that not only incorporate the former, but also take into account geographical boundaries like rivers and mountains and existing infrastructure and speed regulations.

The following formulas are some ways of measuring the intracluster distance. Consecutively, we have the average distance towards the centroid, the diameter of the cluster and the normalized pairwise distance. A lower intracluster distance results into a better cohesion and an overall better clustering, so the aim is to minimize this.

$$\sigma_1(C_k) = \frac{1}{n_k} \sum_{i \in C_k} d(L_i, G_k)$$
$$\sigma_2(C_k) = \max_{i,j \in C_k,\, i \neq j} d(L_i, L_j)$$
$$\sigma_3(C_k) = \frac{1}{n_k \cdot (n_k - 1)} \sum_{i,j \in C_k,\, i \neq j} d(L_i, L_j)$$

The following formulas are some ways of measuring the intercluster distance. Consecutively, we have the maximum distance, the minimum distance, the average distance of all distances between the clusters, the distance between the two centroids and lastly the average distance towards each other's centroid. A higher intercluster distance results into a better separation and an overall better clustering, so the aim is to maximize this.

$$\delta_1(C_k, C_{k'}) = \max_{i \in C_k,\, j \in C_{k'}} d(L_i, L_j)$$
$$\delta_2(C_k, C_{k'}) = \min_{i \in C_k,\, j \in C_{k'}} d(L_i, L_j)$$
$$\delta_3(C_k, C_{k'}) = \frac{1}{n_k \cdot n_{k'}} \sum_{i \in C_k,\, j \in C_{k'}} d(L_i, L_j)$$
$$\delta_4(C_k, C_{k'}) = d(G_k, G_{k'})$$
$$\delta_5(C_k, C_{k'}) = \frac{1}{n_k + n_{k'}} \left(\sum_{i \in C_k} d(L_i, G_k) + \sum_{j \in C_{k'}} d(L_j, G_{k'}) \right)$$

A good clustering requires the combination of a good cohesion and separation and there are many different indices that combine the two in a singular performance measure. The following measures have been selected and implemented, as they are widely popular in the literature and capture different aspects (Desgraupes, 2013).

The Dunn index is the ratio of the minimum intercluster distance to the maximum intracluster distance. This is a worst case kind of measure as the intercluster distance should actually be maximized and the intracluster distance should be minimized. Several different measures for inter- and intracluster distance exist, so also several Dunn indices exist. The Dunn index is positively unbounded, so it does remain hard to make a definitive statement about the quality of a solution outside of an ordinal comparison (Dunn, 1974).

$$\text{Dunn Index } = \frac{\min_{k \neq k'} \delta(C_k, C_{k'})}{\max_k \sigma(C_k)}$$

The Davies-Bouldin index is the average taken over all clusters, maximum, over all other clusters, ratio of the sum of centroidal intracluster distances of both clusters to the centroidal intercluster distance between the clusters. Similarly to the Dunn Index, the Davies-Bouldin should be minimized

and is positively unbounded, so definitive statements about the cluster quality involving the Davies-Bouldin index remain difficult. An added benefit however is that the Davies-Bouldin index is defined locally per cluster, so it is possibly to assess individual clusters using this measure (Davies and Bouldin, 1979).

$$\text{Davies-Bouldin index} = \frac{1}{n_C}\sum_{k=1}^{n_C}\max_{k\neq k'}\left\{\frac{\sigma_1(C_k)+\sigma_1(C_{k'})}{\delta_4(C_k,C_{k'})}\right\}$$

The Silhouette Coefficient is, as opposed to the previous measures, defined per point. This is a huge advantage, because a bad silhouette coefficient can indicate what points specifically need to be looked at to improve the clustering. The Silhouette Coefficient is a combination of the centroidal intracluster distance and the minimum over all clusters of the averages over all distances toward other cluster points, aside from its original cluster. The first term is about cohesion and the second term looks at the "neighboring cluster" or the next best fit and embodies separation (Rousseeuw, 1987).

$$\delta(L) = \min_{k,L\notin C_k}\left\{\frac{1}{n_k}\sum_{i\in C_k} d(L,L_i)\right\}$$

$$\text{Silhouette Coefficient per location, } S(L) = \frac{\delta(L)-\sigma(C_{k,L\in C_k})}{\max\{\delta(L),\sigma(C_{k,L\ \in C_k})\}}$$

Taking the average over all points in a given cluster, gives us a Silhouette Coefficient per cluster.

$$\text{Silhouette Coefficient per cluster, } S(C_k) = \frac{1}{n_k}\sum_{i\in C_k} S(L_i)$$

Finally taking the average over all clusters, gives us a global measure.

$$\text{Silhouette Coefficient, } S = \frac{1}{n_C}\sum_{k=1}^{n_C} S(C_k)$$

The Silhouette Coefficient is appropriately normalized between -1 and 1, with 1 being a theoretically perfect clustering and -1 a theoretically disastrous clustering. A value of 0 indicates the point in particular is on the fence whether it should be in its original or neighboring cluster. The fact that it has proper bounds, really makes this measure more valuable than the previous ones, because it is easier to make value judgments.

The previous measures are not ideal for varying density. To accommodate this shortcoming, we establish the neighbourhood ratio, which is defined per point. It simply looks at how many closest points are in the same cluster and takes the ratio of the number of points checked. The closest points are defined as having minimum distance to the point of which the neighbourhood ratio is being computed. We somewhat arbitrarily check a certain number of neighbours, namely the number of points in the cluster that the point is currently in divided by four to make it robust against varying cluster sizes. Checking too many points makes the KPI lose its specificity and checking too few points makes the KPI lose its accuracy.

$$\text{Neighbourhood Ratio per location, } NR(L) = \frac{\text{\#points in cluster}}{\text{\#neighbours checked}}$$

As the Silhouette Coefficient, this measure can be extended to both the cluster and global level by simply taking averages.

$$\text{Neighbourhood Ratio per cluster, } NR(C_k) = \frac{1}{n_k} \sum_{i \in C_k} NR(L_i)$$

$$\text{Average Neighbourhood Ratio, } NR = \frac{1}{n_C} \sum_{k=1}^{n_C} NR(C_k)$$

OTR also uses two cluster quality measures that go by compactness. They are defined per cluster as the sum of all pairwise distances or the sum of all distances to the centerpoint. To adapt a version of this, we scale them by taking averages and then dividing them by the average distance to get a value that is typically between 0 and 1. Our centerpoints are defined as the client that has lowest average distance to all other points within the cluster. These new measures are branded scaled compactness and scaled centerpoint distance.

$$\text{Scaled Compactness, } SC = \frac{\frac{1}{n_C} \sum_{k=1}^{n_C} \sigma_3(C_k)}{\frac{1}{n_k^2} \sum_{i \in C_k, j \in C_k} d(L_i, L_j)}$$

$$\text{Scaled Centerpoint Distance, } SCD = \frac{\frac{1}{n_C} \sum_{k=1}^{n_C} \sigma_1(C_k)}{\frac{1}{n_k^2} \sum_{i \in C_k, j \in C_k} d(L_i, L_j)}$$

4.1.2 Balancedness

Our application requires that our cluster have near equal size or are balanced. First of, we define the size or weight of a cluster to be the sum of its members weights.

$$w(C_k) = \sum_{i=1}^{n_k} w(L_i)$$

The following measures are adapted from Banerjee and Ghosh, where we have made a weighted version. A natural candidate for measuring the dispersion is the appropriate standard deviation of the cluster sizes (SDCS). As we want clusters that have similar sizes, we would like this to be minimized

$$\text{SDCS } = \sqrt{\frac{1}{n_C} \sum_{k=1}^{n_C} \left(w(C_k) - \frac{\sum_{i=1}^{n_C} w(C_i)}{k} \right)^2}$$

A more worst-case metric is the ratio of minimum or maximum to the expected cluster size (RME). We would want this ratio as close to 1 as possible. At ORTEC, they put hard bounds on these specific KPIs that state the specific margin may not be higher or lower than a specific percentage. The same margin is used for the minimum and maximum (Banerjee and Ghosh, 2004).

$$\text{RME } = \frac{\min_k / \max_k \{w(C_k)\}}{\frac{\sum_{i=1}^{n_C} w(C_i)}{k}}$$

Note that these measures are wholly independent from the cluster quality measures selected before.

4.2 ORTEC Tactical Routing

ORTEC has a centerpoints-based approach towards finding good territories. First, centerpoints are picked randomly from all clients. Second, the problem is modeled as a modified transportation problem that is solved to get a "better" set of centerpoints.

The Transportation problem can be written down in the following way and can be solved to optimality in polynomial running time by a primal-dual algorithm (Ford and Fulkerson, 1956).

$$\begin{aligned} \min_x Z &= \sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij} \cdot x_{ij} \\ \text{s.t. } \sum_{j=1}^{N} x_{ij} &= a_i && \text{for } i = 1, 2, ..., M \\ \sum_{i=1}^{N} x_{ij} &= b_j && \text{for } j = 1, 2, ..., N \\ x_{ij} &\geq 0 && \forall i, j \end{aligned}$$

The modified version is shown below.

$$\begin{aligned} \min_x Z &= \sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij} \cdot x_{ij} \\ \text{s.t. } \sum_{j=1}^{M} x_{ij} &= 1 && \forall i = 1, 2, .., N \\ \frac{\sum_{i=1}^{N} w_i}{M} \cdot (1 - \frac{b}{100}) \leq \sum_{i=1}^{N} w_i \cdot x_{ij} &\leq \frac{\sum_{i=1}^{N} w_i}{M} \cdot (1 + \frac{b}{100}) && \text{for } j = 1, 2, ..., M \\ 0 \leq x_{ij} &\leq 1 && \forall i, j \end{aligned}$$

N	number of clients
M	number of territories
x_{ij}	= 1 if client i is assigned to centerpoint j, = 0 otherwise
c_{ij}	cost for assigning client i to centerpoint j
b	balance margin in percentages
w_i	weight of client i

Finally, the clients are allocated to the set of centerpoints with a relaxed Integer Linear Programming solution or a final run of solving the modified transportation problem.

Lastly, two heuristic local searches in the post-optimizer stage try to increase the quality of the clusters, while staying within the balance margins. The Lin-Kernighan heuristic tries to locally change a client to a different territory one at a time for all clients. The change is made if an improvement in the objective function would occur (Kernighan and Lin, 1970). The second heuristic attempts all swaps of clients or groups of clients up to a size of some parameter and applies the change if the objective function is improved. The objective function in this case is the compactness measure, which is the sum of pairwise within cluster distances. The heuristics can be repeated as a change might shift the situation in such a way that more local improvements are possible.

4.3 Algorithms

4.3.1 Hierarchical Clustering with Size-Sensitive Distance

As the name suggests, it is akin to Agglomerative Hierarchical clustering. The distance given by the linkage function is however adapted for whatever it is linking, giving a bonus to clusters of smaller size, to increase the likelihood that they will be linked. This size bonus needs to be appropriately normalized to avoid making balancing the sole drive in the greedy search, where Ω is the set of all the clients.

$$d(C_k, C_{k'})_{new} = d(C_k, C_{k'})_{old} \cdot \frac{w(C_k) + w(C_{k'})}{w(\Omega)}$$

In practice, amping this bonus up multiplicatively by some parameter, did not increase the balancedness. It is theorized that this is due to singular clients first becoming pairs of clients, then double pairs of clients, where finally extremely large groups of clusters will merge together paradoxically creating a big imbalance. Having the bonus be too small will simply result in no change from the previous situation however.

Another problem that pops up is that precisely adapting the distance sometimes may result in a cluster being able to "hop over" a cluster that is too full to another cluster that is sufficiently small, completely destroying the geographical outlook as a result. In some cases it might even lead to one cluster being completely enveloped by another as seen in Figure 4.1, where the green cluster is embedded in the pink cluster.

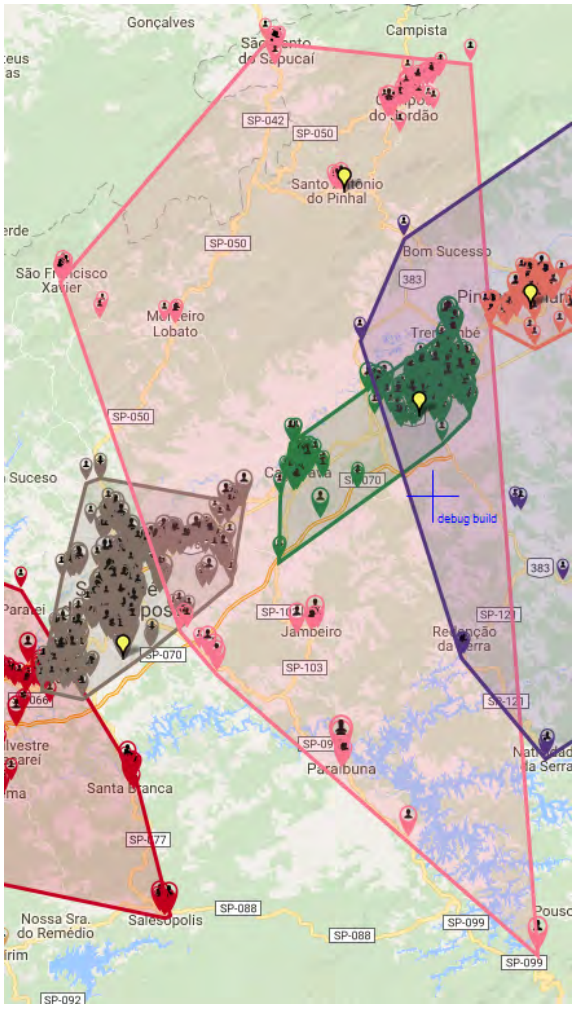


Figure 4.1: Enveloped Cluster

Some experimentation with different linkage functions, combinations of linkage functions and using different linkage functions over different time stages of the algorithm did not mediate these issues. Another disadvantage of having a solution being constructed that way is that it is hard to generalize and every parameter would need to be tuned on a case-by-case basis, so this method was not explored further beyond a possible initial construction heuristic.

4.3.2 Center Selection Functions

Several ways of finding points that are supposed to represent a group of data as a whole or be representative of that set have been researched, as this is one of the predominant ways that clustering algorithms work. All these ways rely on the centers being clients, as we only have access to the pairwise distances between those clients. Among those are k-means++ and Farthest-first traversal that have been mentioned earlier in the K-means section.

Circular exclusion works by randomly selecting centers from points that are currently at least some distance away from already selected centers. The default value at which that distance starts is $5 \cdot \frac{max\_distance}{nr\_clusters}$, where max_distance is the biggest distance from any point to any other point. If that distance is eventually too big, it will get continuously multiplied by 0.8, which is a parameter, until a

point becomes available again. It is a more guaranteed form of kmeans++ in the sense that points are guaranteed to be a certain distance away as opposed to being a certain distance away in expectations.

Furthermore, we have heavy centers, that simply returns the points with highest weight values attached to them. Certain algorithms had difficulties with heavy points on the border of a cluster, so having them be a center nudges this to be less of a problem.

Most of these distance-based center selection methods work great for traditional clustering, but produce bad solutions for instances with varying point density when balancing cluster loads is a concern. An uneven point density can also be expected in a geographical setup with cities typically having more and bigger clients than rural areas. For that reason another center selection method was created, which first produces a multitude of randomly made center selections. Consequently, all clients are assigned to their closest center point and the cluster size standard deviation is calculated. The selection with the lowest cluster size standard deviation is chosen as centers. The absolute distance does not play a role here. Only the relative distance is used when assigning the clients to their closest center point, so varying density does not lead to any issues with this approach, which is coined the Best of Random Center Selection Assignment.

Lastly, a form of k-means was also implemented, where centers were chosen by the aforementioned methods and consequently all points were added to their closest center, after which the centers would be recalculated. This recalculation was done by averaging the x and y coordinates of all points attached to that center and then finding the point with the closest Haversine distance to have that be the new center. Furthermore, a more balanced version was created that took the weighted average of the x and y coordinates instead, so bigger points have a larger impact on the new centers location. The differing centers at the start have a wildly variable impact on the clustering at the end of the algorithm as can be seen in Figure 4.2.

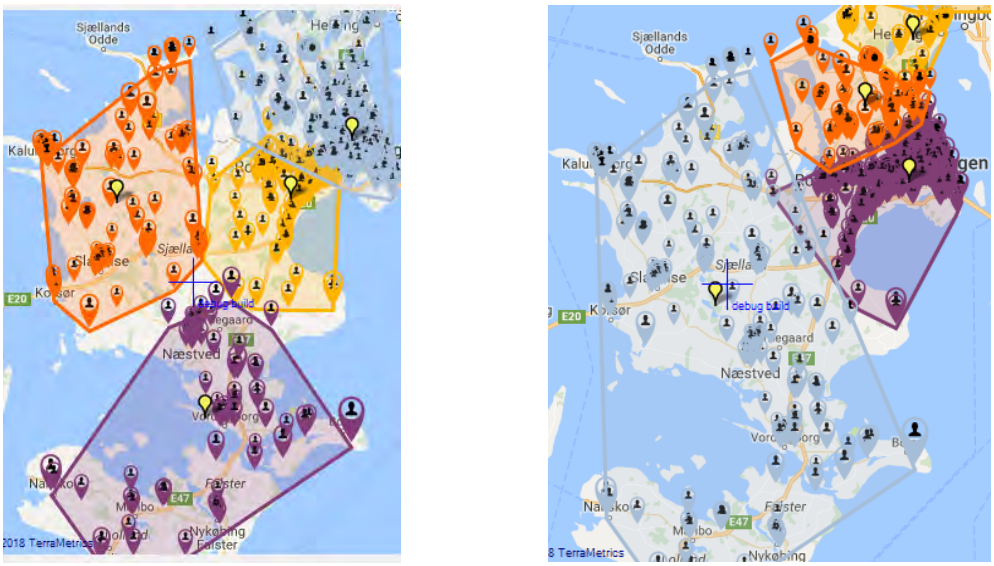


Figure 4.2: Results Differing Initial Centers K-means

4.3.3 Center-Based Heuristic Expansion

Center-Based Heuristic Expansion is an iterative algorithm that at each iteration will expand the center, whose load is at that stage the lowest. Centers expand by taking the nearest client within their territory, where nearest can be defined in terms of a linkage function, or the actual initial distance to the center. It was theorized that this could be improved by for instance altering the distance towards specific points based on certain attributes. Assigning bigger points first, would improve overall balance generally for instance. Giving points that are closer to differing clusters, than the one currently being expanded upon, a higher distance could also be beneficial.

Another two variations exist. One where clients are eligible to be expanded upon, if they do not belong to some center already or one where clients are always eligible to be expanded upon, but distances to that client increase multiplicatively by some multiplicatively growing parameter to deter reselection. The latter method ensures that clients are not becoming fixed and that makes the algorithm more dynamic overall. The solution resulting from this algorithm has the attractive property that it cannot deviate more than the weight of the biggest point from the optimal solution in the worst case possible. This approach was not pursued any further after unsatisfactory initial results came back, because this first prioritizes balance and then needs to be re-optimized towards a nice geographical outlook, which is a harder order than the other way around. It is easier to work towards better balance when geographical outlook is already optimized, as balance is less subjective of an objective.

4.3.4 Finding Cluster Neighbourhood

Similarly to finding center points, another thing that might prove useful is having a method to detect when two clusters are near each other and, maybe even more importantly, when they are not. One such method relies on first finding the two-dimensional convex hull of every cluster, which doubles up as a nice visualisation tool and possible measures designed around overlap in convex hulls. We achieve this with the monotone chain convex hull algorithm, which first sorts the points by x-coordinate, and in case of a tie, by y-coordinate and consequently, it constructs upper and lower hulls of the points and connects them (Andrew, 1979).

After we have the convex hulls of all clusters, we can detect the neighbourhood for every cluster. For a single cluster, we take the convex hull. For every point on that convex hull, we find the closest point that is not in that cluster. Any clusters such points belong to are added as neighbouring clusters and this process is repeated for all clusters. This does not necessarily have to be symmetric as in if cluster A is in the neighbourhood of cluster B that cluster B is in the neighbourhood of cluster A. See the following example in Figure 4.3, where green cluster A is connected to red cluster B, but cluster B is only attached to brown clusters C and not connected to cluster A. This situation might seem contrived, however it was a reality for the picture at subsection 4.3.1. The symmetry is important for future algorithms and is thus enforced afterwards.

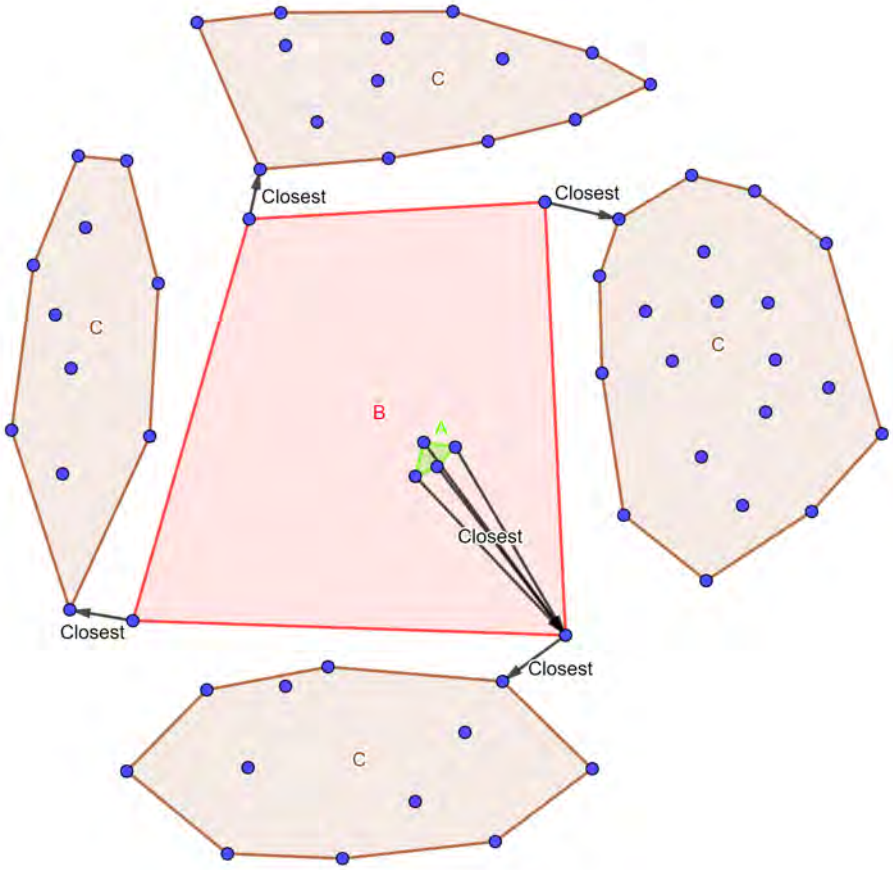
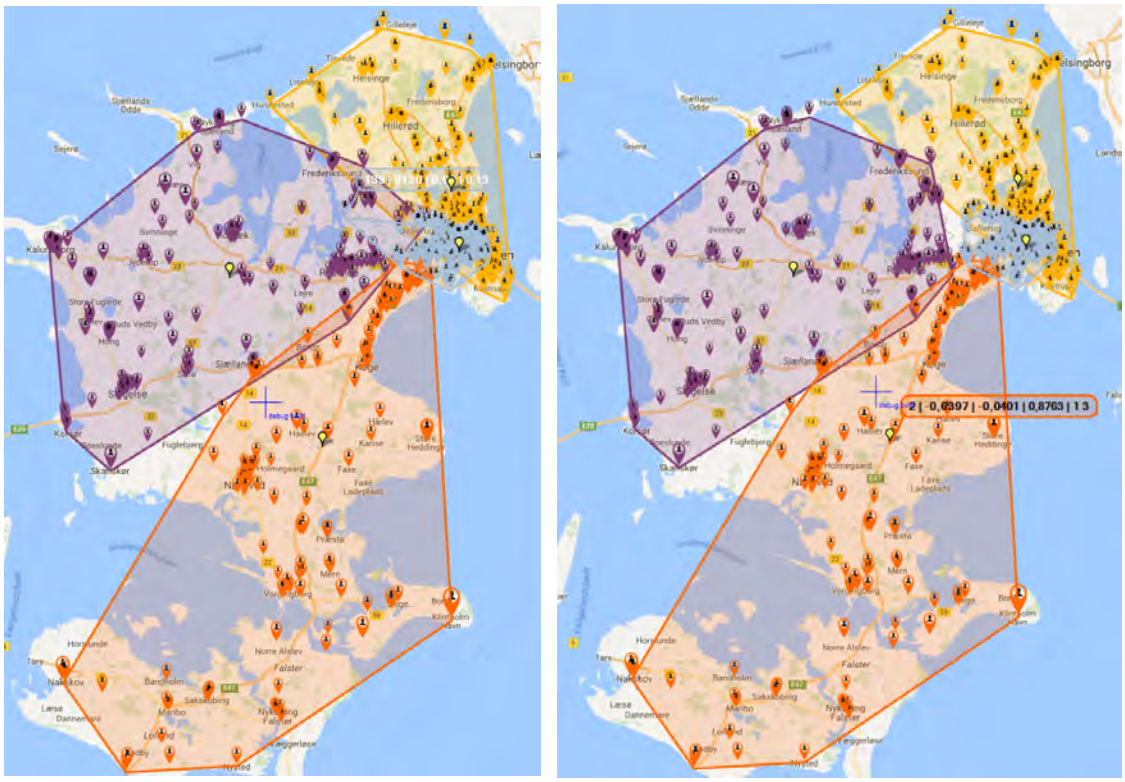


Figure 4.3: Non-guaranteed Symmetry Between Cluster Neighbourhoods

4.3.5 Metachrosis

Metachrosis refers to the ability of an animal like a chameleon to change its colour to blend in with the environment. The purpose of this local search function is to get rid of strange points in a clustering that the Territory Optimizer sometimes created and integrate them into the respective cluster that they are close to or even embedded in. Its purpose is to improve the geographical outlook with no regard for balance, so it may improve or worsen the balance. It changes all points with a neighbourhood ratio of 0.2 or less to the cluster that it has the most neighbours in common with. The cutoff is arbitrary and a parameter, but it makes some intuitive sense if you realize the fact that points on the boundary of a cluster will always have a neighbourhood ratio of at most 0.8 or so, unless there is the extreme case of all clusters being extremely far away from one another. An example of Metachrosis at work is shown in Figure 4.4, where the blue gray point in Figure 4.4a is now purple as seen in Figure 4.4b leading to a much nicer overall geographical outlook in a smoother separation between the purple and blue gray cluster. A similar function was developed that used the Silhouette Coefficient instead, but that proved to be less effective and was not picked up further.



(a) Before

(b) After

Figure 4.4: Metachrosis

4.3.6 Stretchy Search

Stretchy Search is a local search like function that takes in an imbalanced clustering that already has a nice geographical outlook and attempts to improve the balance while keeping the outlook the same or better via stretching the clusters. After each iteration, it is checked if the balancing requirements are met and the algorithm can stop attempting to find improvements. Stretchy Search shrinks clusters that are too large in size and swells clusters that are too small in size. The swelling situation is completely symmetric to shrinking. Stretchy Search first orders every cluster from highest discrepancy towards the mean to lowest. The rationale behind this is that the huge enormous and incredibly tiny clusters need to be fixed first.

Per the previous subsection, the neighbourhood of every cluster is known. Stretchy Search attempts to shrink clusters by passing of the point that has the lowest average distance to a different neighbouring cluster. This is only an attempt as there is a check in place that ensures that after the client would be passed of to the neighbouring cluster that that neighbouring cluster would not suddenly be the bigger cluster of the two, leading to an infinite loop of passing a client between the two clusters. In such a case, the attempt does not go through and the next most imbalanced cluster is considered for shrinking or swelling. It is conjecture that the previously forbidden exchange will be freed up again once other clusters start improving. This conjecture turned out to be false in the following situation that might occur, where every cluster triggers the balance check and nothing can be exchanged anymore. An example is shown in Figure 4.5, where A cannot shrink towards B, C cannot shrink towards B, G, cannot swell towards F, E cannot swell towards F, F cannot swell

towards D, B cannot shrink towards D and D cannot swell towards B. The exceptionally large clients in clusters B and D exacerbate the problem. The solution to this problem is to somehow bypass these intermediate problematic clusters, which is what the next section entails.

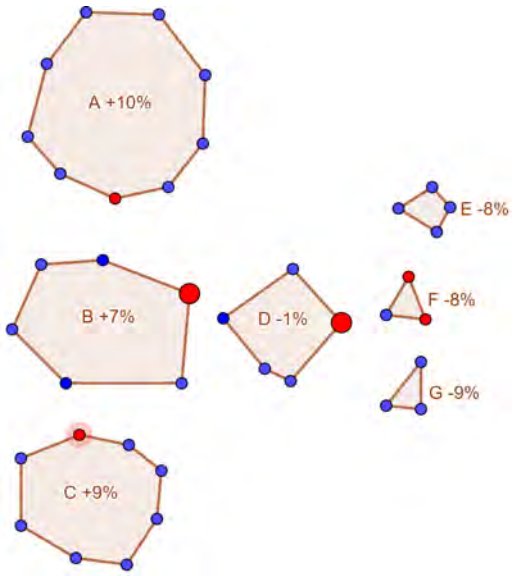


Figure 4.5: Counterexample Stretchy Search

4.3.7 Charity Search

The solution to the particular caveat that stops Stretchy Search from working is to to consider more than just the neighbouring clusters. Charity Search does this by finding a path in clusters from the most overloaded to the most underloaded cluster and then routing some load to that cluster, as is visualized in Figure 4.6, where ultimately cluster A should lose a point and cluster G should gain one.

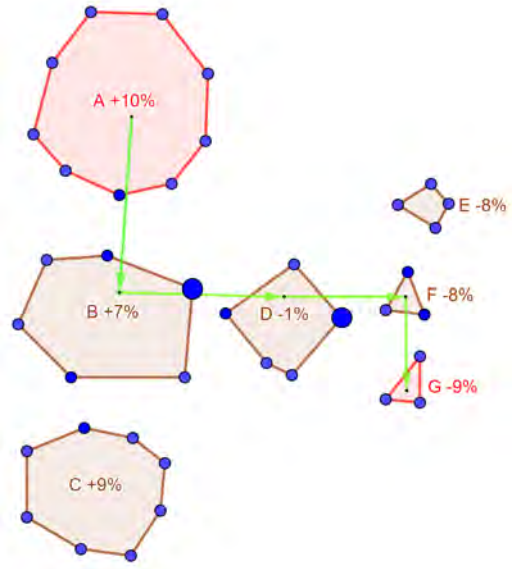


Figure 4.6: Charity Search

This path is found by utilizing Breadth-First Search, which is described below. Breadth-First Search can be used to find shortest paths in an unweighted graph. (Cormen et al., 2014) The fact that all clusters have known cluster neighbours is used. This is also where the symmetry of being cluster neighbours was important or the algorithm would not terminate.

```
BREADTH-FIRST SEARCH(G, SOURCE, SINK)
 1: Create queue object, Q, and Q.enqueue(source).
 2: Create visited array and add source.
 3: Create dictionary, parents, to save parent nodes
 4: while Q is not empty do
 5:     v = Q.dequeue()
 6:     for all neighbours w of v in Graph G do
 7:         if w is not visited then
 8:             Q.enqueue(w)
 9:             mark w as visited
10:             w.parent = v
11:         end if
12:         if v == sink then
13:             Q.clear()
14:             break
15:         end if
16:     end for
17: end while
18: return path from source to sink by backtracking parents from sink
```

After a path is found, each cluster transfers a point to the subsequent cluster on the path. The point that is given to the next cluster is the one with lowest average distance to all points in that cluster, hence we use inspiration here from the hierarchical linkage functions. This occassionally leads to clusters splitting other clusters in the middle, because the algorithm's only concern is with the former cluster and it does not take into account the latter one when the point switches clusters. Several approaches to attempt to remedy this problem were taken. First a single point kept being transferred, but now based on having the lowest partial silhouette coefficient, where the partial indicates that only the two clusters that are exchanging a point were used in the computation. The lowest partial neighbourhood ratio was chosen next as a candidate for exchange.

These single point transfer methods did not solve the problem without making new problems appear, so the next attempt was to incorporate multiple points. A frontier was now calculated, which attempted to appropriately handle one cluster by having the frontier be chosen in a certain way and then choose a point from the frontier that would be most beneficial for the other cluster. The frontier is computed by taking for all points, in the cluster that gains a point, the closest possible point that is not in the same cluster. If that closest possible point is currently a point in the cluster that loses a point it will become part of the frontier. This is necessary, because the closest possible point might also be part of a third party cluster. If the frontier is empty, because for instance a third party cluster takes all of the points, this restriction is bypassed and only the two clusters in question are considered to still always yield a non-empty frontier. One frontier is depicted in Figure 4.7, where the brown cluster wants to give one of its points to the green cluster.

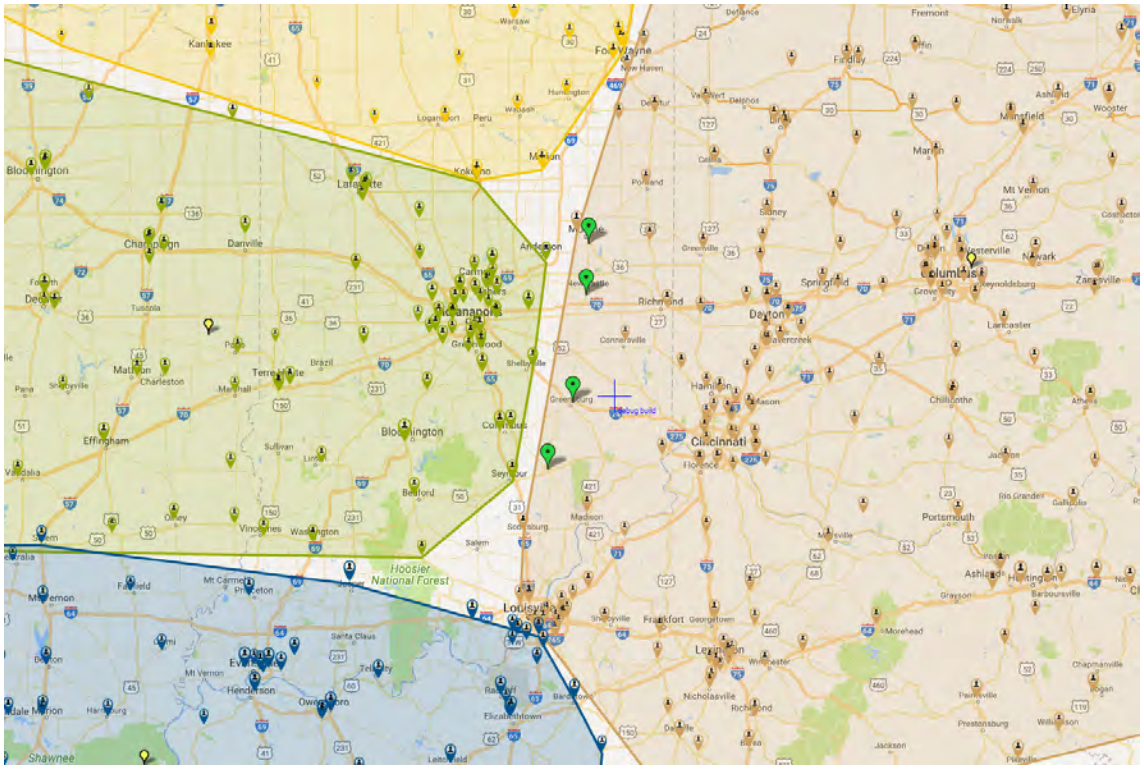


Figure 4.7: Frontier

Selection mechanisms on the frontier try to maximize the cluster quality of the cluster that loses a point or in the case of the figure, the brown cluster. Attempts on this were lowest neighbourhood ratio, lowest silhouette coefficient, highest average distance from all other points in the cluster, other farthest distance measures using linkage functions and combinations of linkage functions. None of them unilaterally performed better on all cases from both the perspective of the picture and the KPIs, so for simplicity's sake we stick with the single point average linkage transfer mentioned at the start. It is suspected that third party clusters interfere in such a way that measures based on just the clusters participating in the transfer are not sufficient in the general case.

Charity Search can not handle extremely big clients. Big is relative to both the margin desired and the mean cluster size, which is obtained from the total mass of all clients divided by the number of clusters necessary. These clients are fortunately an oddity and if they are sufficiently big, they can be made a cluster or even several clusters on their own and be removed from the data. If they however would occur and would cause an infinite loop in swapping between clusters continuously, Charity Search will run 100 iterations of Stretchy Search to flush out some weight and allow new transfers that were not previously possible. This solved all such problematic cases.

4.3.8 Algorithmic Process

The final algorithmic process is depicted in the flowchart in Figure 4.8. At the initialisation phase the data is read in and relegated to the relevant data structures. The Best of Random Center Selection Assignment described in subsection 4.3.2 runs with number of iterations equal to the number of clients there are for that particular case. Charity Search described in the previous section is ran next alternately with Metachrosis for 3 cycles in total. Charity Search has two possible stopping

criteria. Either 1000 iterations are reached or the balance falls within the margins. This 1000 is empirically chosen as a large number that is usually not reached. Every 10 iterations Charity Search checks if there is an improvement in balance and if there is not, it will run 100 iterations of Stretchy Search before continuing. This 100 was determined to be plentiful to escape local Charity Search optima, but scarce enough to not become the main solution method. After 3 cycles, the algorithm will terminate and a solution is reached, which can be visualized or have its KPIs be computed. This algorithmic process scheme has undergone a series of modifications based on the KPIs and the visualisation results, yielding this as most apt in the general case.

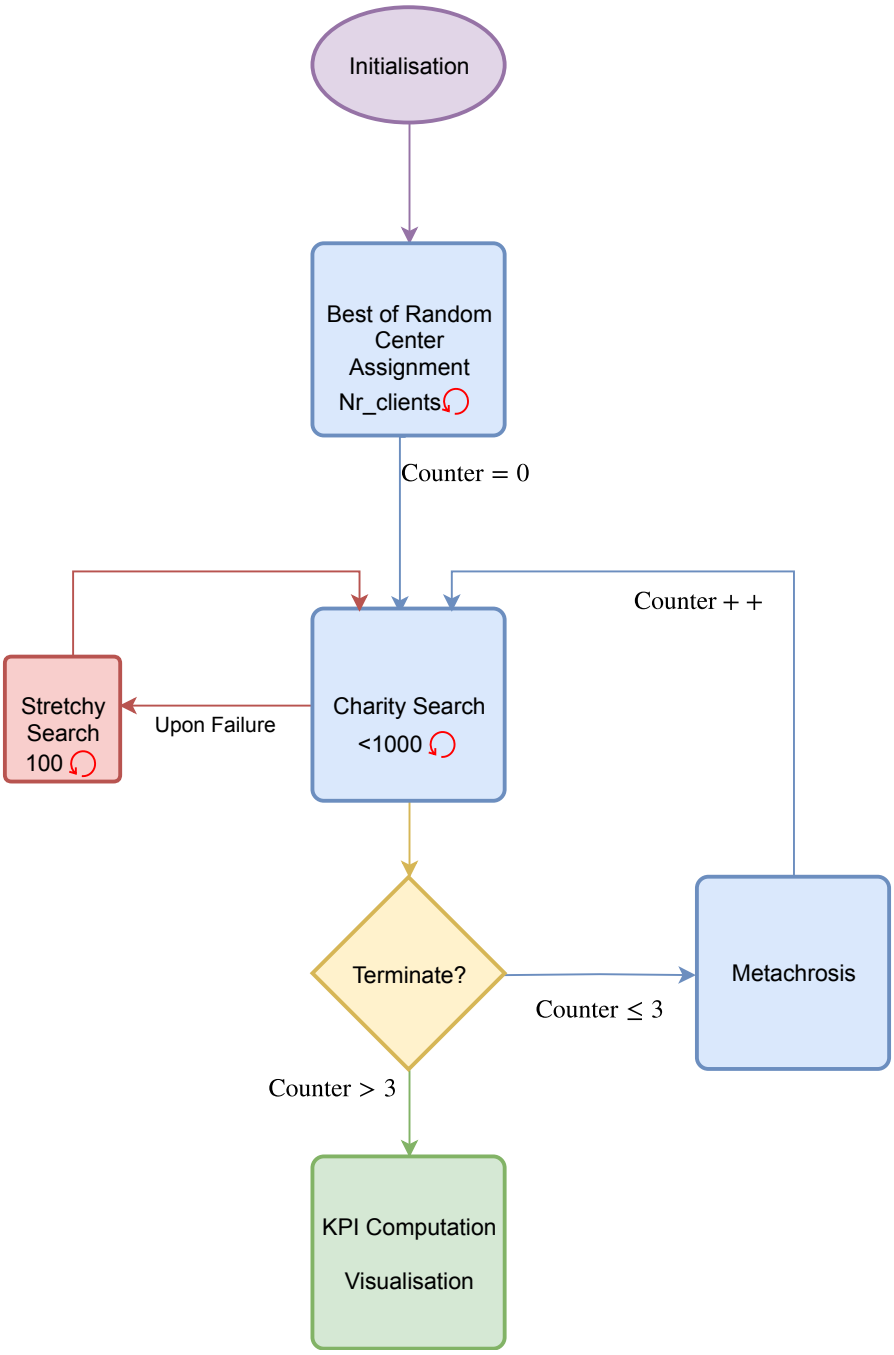


Figure 4.8: Algorithmic Process

Chapter 5

Data

This section takes a closer look at the real life data cases we utilize in validating our algorithms and coming to results. This data analysis is done to determine the quality and practical accuracy of the data, so that we may actually use it and gain insight in what the results mean. Some data characteristics may also be leveraged to create specialized algorithms that are better suited for this given data. First, we consider the cleaning steps that were made and the origin of the data. Second, we show a global overview of all data cases that are available and, last, we discuss 3 cases more in depth, namely cases that are based in Poland, the USA and Stockholm.

5.1 Data Cleaning

Data comes straight from workspaces of existing or past clients of ORTEC. It is therefore not usable right away for our purposes and requires some cleaning. OTR maps input data in the form of GPS coordinates or descriptions containing address, city, region, country, zip code, house number fields to a map. The input data is user-inputted and as such is very error-prone. It frequently occurs that an employee mismatches a field or puts irrelevant extra information that makes it hard for the algorithm to detect the important information. To fix this, it is required to manually look at outliers and remap or delete them, based on a case-by-case basis. In some cases, it occurs that a client is mapped in the depths of an ocean for instance, making it obvious that something needs to be done. Each mapping gets a certain Geo-score attached to how closely the mapped location resembles the input data, as a tool to identify these points. GPS coordinates are almost always correct however.

Another issue that occurs is that different clients can be attached to the same location ID. This relates to how to the mapping server assigns clients their location, where for example clients in the same street would be assigned the same coordinates. It would be hard to explain why the same building or complex would belong to multiple different territories unless the resources are very small, so typically these need to be of the same territory. We simple combine such clients into a singular bigger client representing a location, such that all clients represent a specific location.

5.2 Data Cases

Twenty two datasets were prepared for this undertaking and a global overview is showcased in Table 5.1. The company names are not listed for privacy concerns, but we do have a varied sample with the the following sectors - subsectors being included: Consumer Goods and Manufacturing - Beverage, Professional & Public Services - Hospitality, Professional & Public Services - Facility Services, Trade - Food Services and Trade - Retail. The number of clients ranges from 260 to 45.653, so we must have algorithms that can accommodate roughly 50.000 clients. Similarly, the number of resources available or the number of clusters that need to be created is given and ranges from 3 to 169. It is of note that more clients does not translate to more clusters, so the clients per cluster ratio is not constant. The specific balance criteria are also given, which may be estimated working time, kg, pallets, colli, the number of visits or a custom balance size. Finally, the margin that the company allows is listed. A margin of x% means that the company in question allows for a x% differential between the mean size of a cluster and the maximum and minimum size of a cluster. A higher margin means that the company values balanced loads less than the quality of the clustering. Three datasets will be discussed in depth next. The other cases can be viewed in Appendix A.

Table 5.1: Global Data Overview

Location	Company	Clients	Clusters	Balance Criteria	Margin
Denmark East I	A	1002	4	Working Time	8%
Denmark East II	A	6304	28	Working Time	8%
Denmark West	A	9912	50	Working Time	6%
Arlöv	A	3855	19	Working Time	5%
Göteborg	A	4884	22	Kg	10%
Jönköping	A	3884	20	Kg	20%
Stockholm	A	3924	23	Kg and Working Time	5% each
Roskilde	A	746	7	Working Time	5%
Texas I	B	777	16	Working Time	2%
Texas II	B	516	10	Working Time	5%
Netherlands	B	45653	38	Visits	5%
UK	B	23211	169	Working Time	5%
Poland	B	601	5	Working Time	5%
St. Louis I	B	1876	33	Working Time	5%
St. Louis II	B	1829	33	Working Time	5%
New York City	B	2846	45	Working Time	2%
Taiwan I	B	2143	14	Working Time	5%
Taiwan II	B	3770	22	Working Time	5%
Brazil	C	3659	12	Pallets	10%
Belgium	D	15831	39	Kg	10%
USA I	E	260	3	Colli	1%
USA II	E	5060	44	Balance Size	2%

5.2.1 Poland

In Table 5.1 some statistics about the client data are listed, focusing on the balancing criteria for this specific case. The whole distribution is visualized in Figure 5.2. This is akin to an exponential curve and this specific pattern appears for the majority of the dataset, where there are a lot of relatively small clients and progressively fewer medium sized clients and typically a relative large amount of outliers. The green bar visualizes the 99% quantile, so 1% of the largest clients is contained in that special bin.

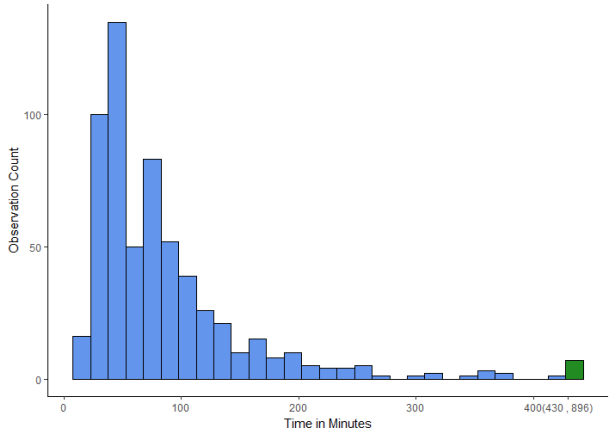


Figure 5.1: Poland Histogram

Table 5.2: Poland Statistics

Minimum Value	10
Maximum Value	896
Mean (Standard Deviation)	87.88 (86.00)
Observation Count	601
Unique Observation Count	189
Mode (count)	36 (21)
2nd Mode (count)	38 (15)
3rd Mode (count)	40 (15)

In Figure 5.2 the locations of clients are showcased on the map having a scale of a large city and its surroundings. The colours indicate the cluster that OTR has assigned them. There is a more dense epicenter in the middle of Poznań, with some less dense areas on the outskirts.

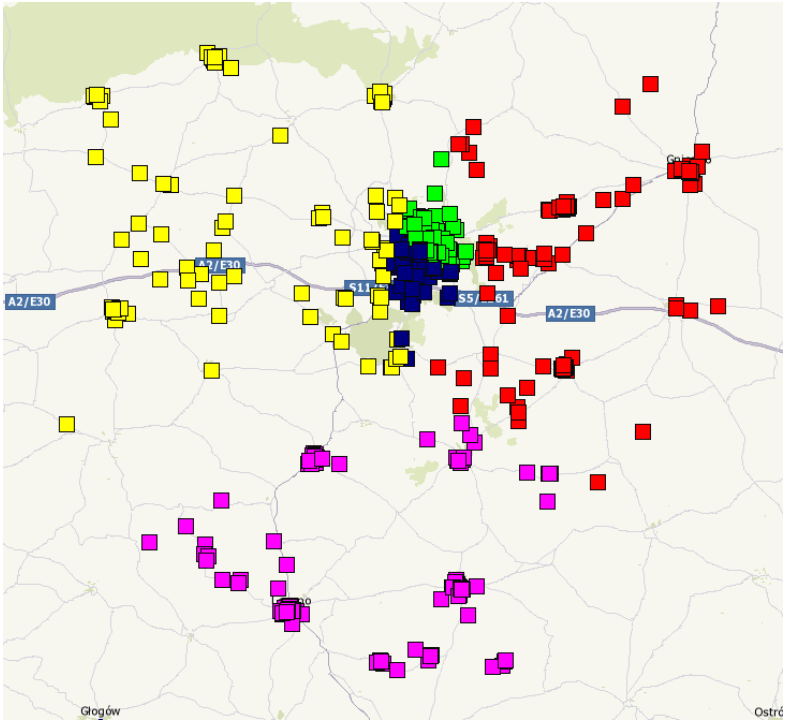


Figure 5.2: Poland OTR

5.2.2 USA II

Some statistics are again recorded in Table 5.3 and visualized in Figure 5.3. The balancing criteria in this case are special user-inputted values. Both USA datasets are the only datasets that do not follow the previously mentioned exponential-like curve, but instead show something more akin to a normal distribution. Once more there are substantial extreme values. A phenomenon detected in every dataset except USA I.

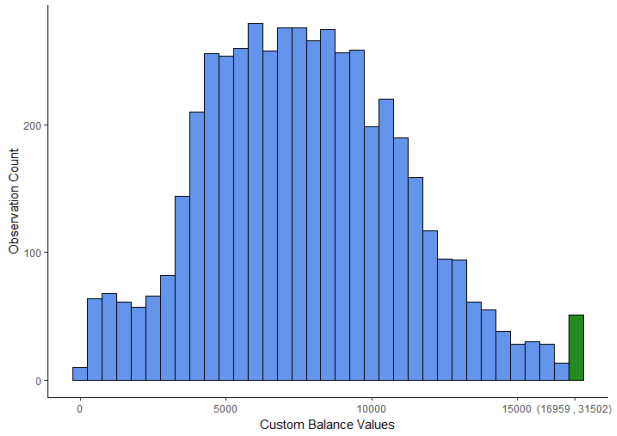


Figure 5.3: USA II Histogram

Table 5.3: USA II Statistics

Minimum Value	6
Maximum Value	31502
Mean (Standard Deviation)	7591 (3510.70)
Observation Count	5060
Unique Observation Count	4198
Mode (count) 2nd Mode (count) 3rd Mode (count)	9334 (6) 4699 (5) 4922 (4)

In Figure 5.4 the locations of clients are showcased on the map, which has a scale of a large part of a continent. The colours indicate the cluster that OTR has assigned them. OTR only has 12 colour options, so not every region is marked distinctly. The collection of points correspond almost directly to a population density map, so this result is expected with dense zones on the east and west coasts and a relatively lower density in the middle of the USA.

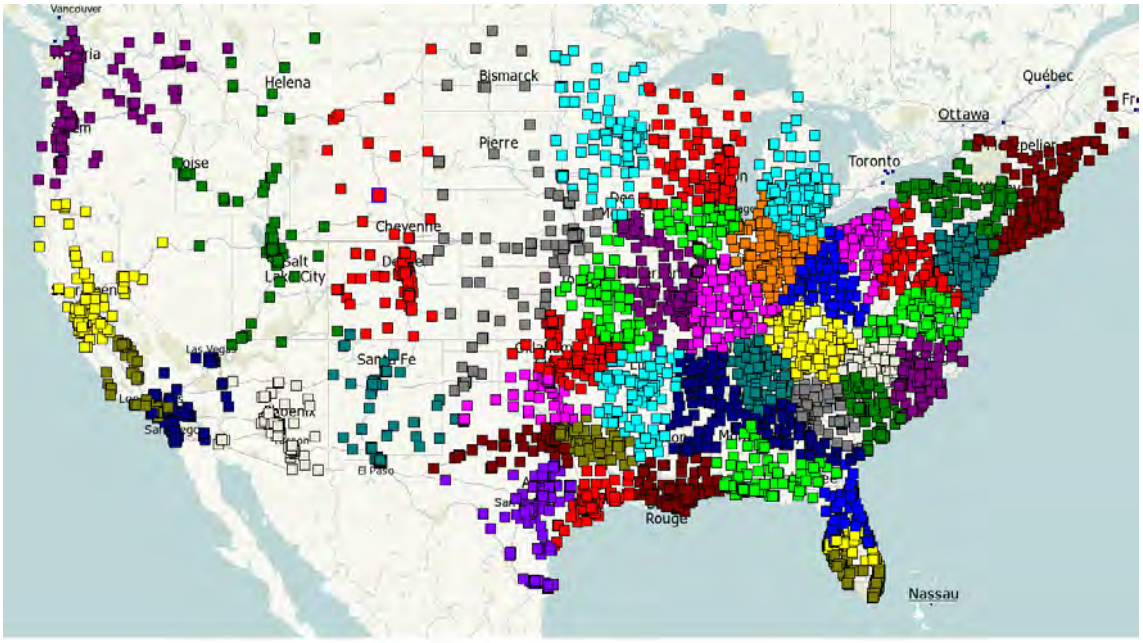


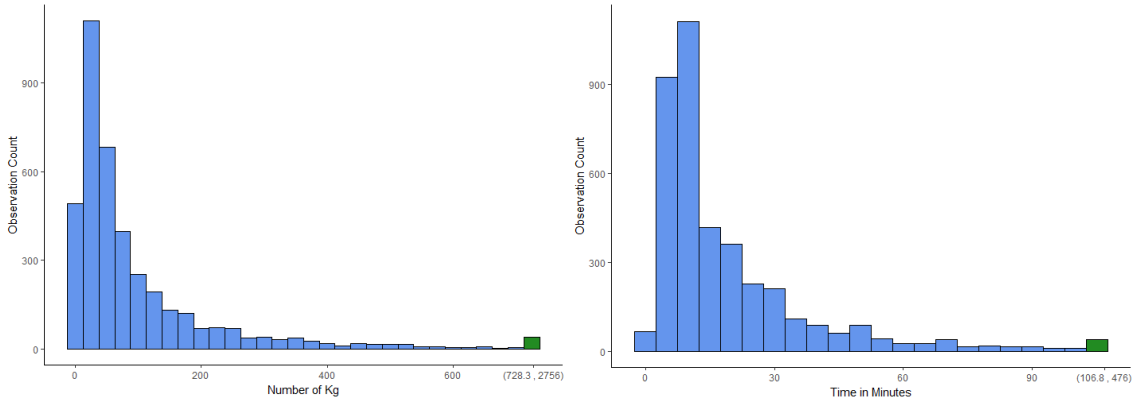
Figure 5.4: USA II OTR

5.2.3 Stockholm

This is the sole dataset where was balanced on multiple criteria, namely kg and estimated working time. Some statistics are once again recorded in Table 5.4 and visualized in Figures 5.5a and 5.5b. An exponential like curve can be detected once again.

Table 5.4: Stockholm

Statistic	Kg	Working Time in minutes
Minimum Value	0	2
Maximum Value	2756	476
Mean (Standard Deviation)	99.37 (154.31)	19.73 (23.70)
Observation Count	3924	3924
Unique Observation Count	487	125
Mode (count)	0 (93)	6 (442)
2nd most common (count)	13 (70)	8 (438)
3rd most common (count)	23 (69)	12 (339)



(a) Kg (b) Working Time

Figure 5.5: Stockholm Data

The previous distributions were remarkably similar, so one might wonder whether it is practical to balance on both criteria, if the criteria are almost the same. The correlogram of this dataset is shown in Figure 5.6 that showcases the correlation between kg and estimated working time is 0.77. It is expected that all these criteria are positively correlated as for example loading more kg, typically takes more unloading time, but if they are too correlated, it is pointless to balance on both, instead of just one. For this reason it is important to check the correlation when considering multiple balance criteria.

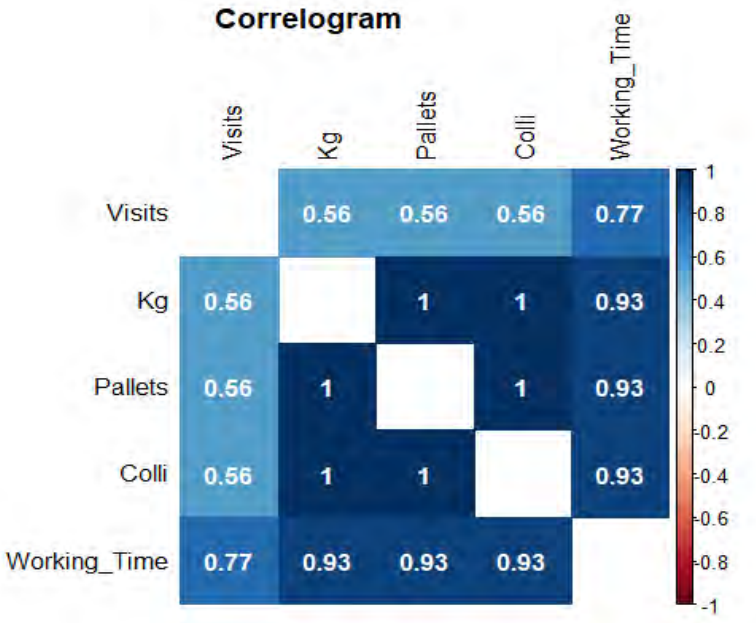


Figure 5.6: Correlogram

In Figure 5.7 the locations of clients are showcased on the map, which has a scale of a region of a country. The colours indicate the cluster that OTR has assigned them. This dataset is especially rich in water areas that obtrude on the usual 'as the crow flies' distances. These geographical obstructions can also favor some algorithms over others. The dense areas are Stockholm and Uppsala, becoming less dense going outward from their centers, as could be expected.

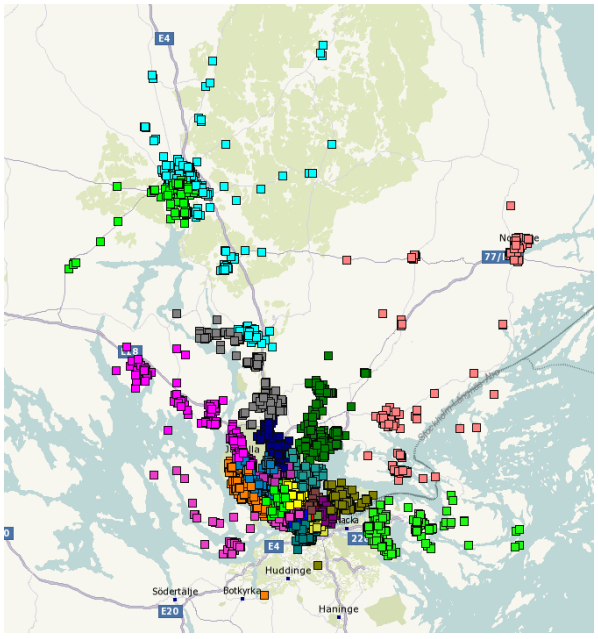


Figure 5.7: Stockholm OTR

Chapter 6

Results

First, an overview of the global results will be given in the form of a table of KPIs and global pictures of the geographical clustering that resulted from the algorithm. Second, a comprehensive lens is going over the specific cases handled in the data analysis section, namely the Poland, USA II and Stockholm cases.

6.1 Global

The algorithmic process scheme as depicted in subsection 4.3.8 has been run for every single case and the results are showcased in Table 6.1. A global picture of the map and individual KPI results are listed in Appendix A. In order the Ratio Minimum/Maximum to Expected balance size, Average Neighbourhood Ratio, Average Silhouette Coefficient, Scaled Compactness and full algorithm running time are shown here. Dunn indices were excluded for lack of explanatory value and the sheer number of possible combinations possibly obscuring other KPIs. Davies-Bouldin indices and the standard deviations of the cluster sizes are relegated to the appendix. The Denmark West and Belgium cases are both ran using Euclidean distances due to their size. All other cases use real distances as calculated by the ORTEC Map and Route server. The Netherlands and UK cases were eventually excluded due to implementation limitations and problmes with the particular data.

Table 6.1: Global Results

Location	RME	ANR	ASH	SC	Running Time (mm:ss)
Denmark East I	0,937 - 1,038	0,8625	0,2237	0,635	00:01
Denmark East II	0,955 - 1,055	0,8132	0,2343	0,227	02:51
Denmark West	0,953 - 1,048	0,8223	0,2144	0,127	11:12
Arlöv	0,962 - 1,023	0,8143	0,1736	0,215	00:54
Göteborg	0,920 - 1,073	0,8212	0,1207	0,202	01:50
Jönköping	0,854 - 1,134	0,8197	0,1944	0,197	00:57
Stockholm	0,972 - 1,034	0,8206	0,1970	0,246	01:08
Roskilde	0,965 - 1,035	0,7895	0,0556	0,481	00:05
Texas I	0,988 - 1,010	0,8338	0,1316	0,253	00:02
Texas II	0,972 - 1,040	0,8490	0,3152	0,323	00:00
Poland	0,975 - 1,032	0,8774	0,0263	0,610	00:00
St. Louis I	0,962 - 1,038	0,8125	0,1672	0,287	00:13
St. Louis II	0,962 - 1,037	0,7977	0,1828	0,284	00:16
New York City	0,987 - 1,015	0,7945	0,2278	0,172	01:03
Taiwan I	0,971 - 1,037	0,8476	0,2913	0,254	00:18
Taiwan II	0,973 - 1,029	0,8017	0,1811	0,267	01:00
Brazil	0,922 - 1,075	0,8751	0,2407	0,255	00:28
Belgium	0,928 - 1,079	0,7755	0,1649	0,166	10:40
USA I	0,995 - 1,004	0,9091	0,3813	0,349	00:00
USA II	0,984 - 1,015	0,8159	0,2649	0,135	02:39

For all cases the Ratio Minimum/Maximum to Expected balance size falls within the required boundaries of the set margin. As for the clustering quality KPIs, we have that the Average Neighbourhood Ratio is relatively high falling generally in the 0.8 range. This is principally decent, because points in the center of a well-made cluster typically have 1 as neighbourhood ratio and points near the border 0.6 to 0.7. The Average Silhouette Coefficient is at least positive for every single case, but is not particularly high. This is to be expected with our strict requirements of balancedness combined with varying point density. The Scaled Compactness is hard to interpret for similar reasons, but is one of the main OTR metrics and is included for that reason. The running times are excellent with around 10 minutes for the largest cases and near instant for the smaller cases. This falls in line with the expectation of a collection of heuristics.

6.2 Specific Cases

First, a timeline of the KPIs at each specific phase of the algorithmic process is given. Phase I is right after the best of random center selection assignment. Phase II is after an initial Metachrosis that cleans up any oddities from the best of random center selection assignment. Phase III is after the first Charity Search. Phase IV, V and VI stand for a cycle of Metachrosis plus Charity Search. Phase VI is then also the final result. The number of iterations that were ran per phase is given between brackets. Second, a global picture of the map plus some zoom shots of specific areas are shown that have interesting results. Third, a bar plot of the cluster size distribution is shown.

Fourth, a bar plot of the silhouette coefficients distribution and a plot of the neighbourhood ratios distribution are shown.

6.2.1 Poland

The KPIs over time for the Poland case are given in Table 6.2. The case is relatively small, so it requires few iterations and runs almost instantly. A significant improvement in the balance KPIs can be observed after phase II, which is where the Charity search first runs. It is noteworthy that the Ratio Maximum to Expected and Ratio Minimum to Expected both fall within the 5% margin from that point onward. The cluster KPIs remain relatively stable with only the Davies-Bouldin index showing a large improvement indicating that the most poor cluster got improved in that phase. The Silhouette Coefficient is low, but positive and the Neighbourhood Ratio is relatively high.

Table 6.2: Timeline KPIs Poland

KPI	I (601)	II (7)	III (12)	IV (7)	V (3)	**VI** (0)
Mean Size (Working Time)	-	-	-	-	-	**10563,20**
Standard Deviation	921,21	915,94	304,84	244,15	283,73	**283,73**
Ratio Maximum to Expected	1,090	1,082	1,034	1,026	1,032	**1,032**
Ratio Minimum to Expected	0,876	0,868	0,965	0,977	0,975	**0,975**
Average Neighbourhood Ratio	0,8541	0,8755	0,8590	0,8718	0,8774	**0,8774**
Average Silhouette Coefficient	0,0415	0,0422	0,0332	0,0276	0,0263	**0,0263**
Davies-Bouldin Index	5,4907	5,3411	5,5476	5,4532	3,9267	**3,9267**
Scaled Compactness	0,622	0,613	0,613	0,611	0,610	**0,610**
Scaled Centerpoint Distance	0,432	0,429	0,429	0,429	0,429	**0,429**
Cumulative Running Time	00:00:00	00:00:00	00:00:00	00:00:00	00:00:00	**00:00:00**

A global picture of the Poland case is shown in Figure 6.1. The yellow, purple and blue gray clusters seem well separated on a glance. The red and purple seem to have overlap, but in actuality this is just a by-product from highlighting the convex hull. A closer look at the middle is warranted.

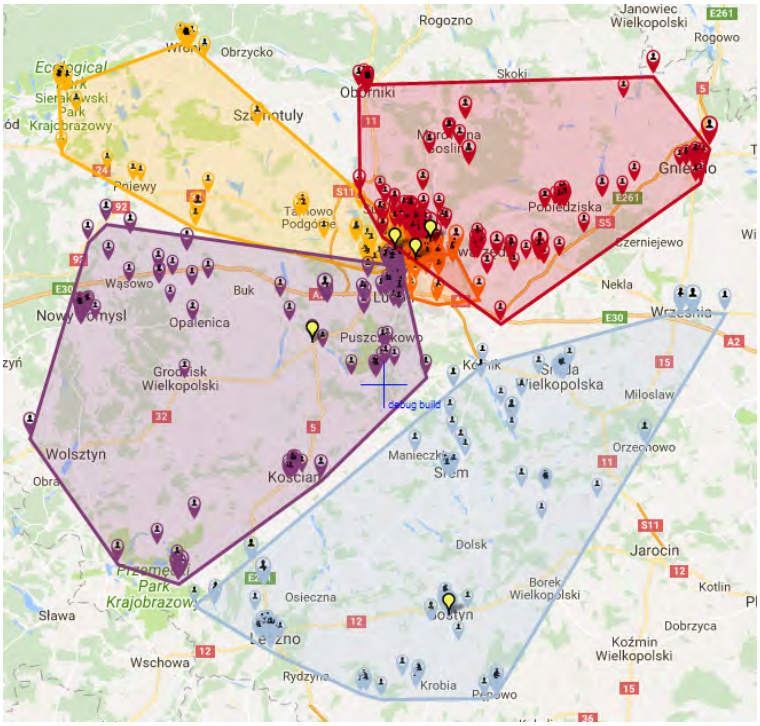


Figure 6.1: Poland Results Map

A zoom shot from the map is portrayed in Figure 6.2. There is a clear separation between the 4 highlighted clusters and the earlier suspicion that the red and orange cluster are separated is confirmed. The red convex hull point within the yellow cluster is also justified, because of the main road making that point closer than the two yellow points in the more rural areas. This is a prime example of the difference Euclidean and real distance can make and why the convex hull is a flawed, yet helpful tool.

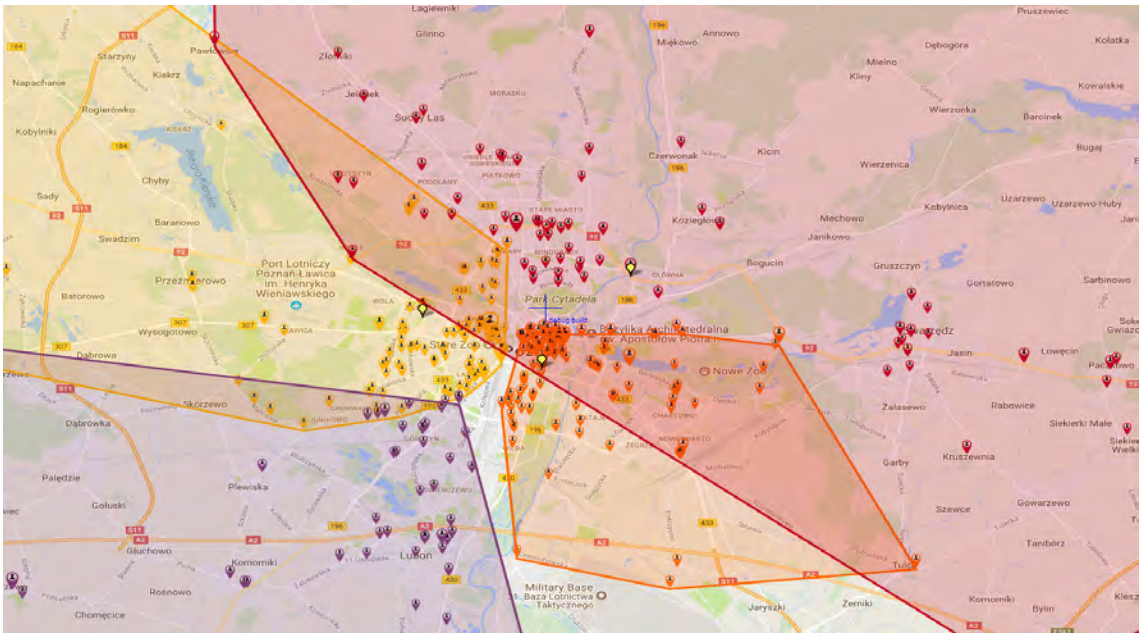


Figure 6.2: Poland Zoom Shot

A bar plot of the cluster size balance deviations for the Poland case is shown in Figure 6.3 to show a more complete picture of the cluster size distribution, as opposed to just the standard deviation and minimum and maximum. For this small case it is relatively unremarkable however, but we can note it to be between the margin of -5% and +5%. The colours correspond to the map pictures.

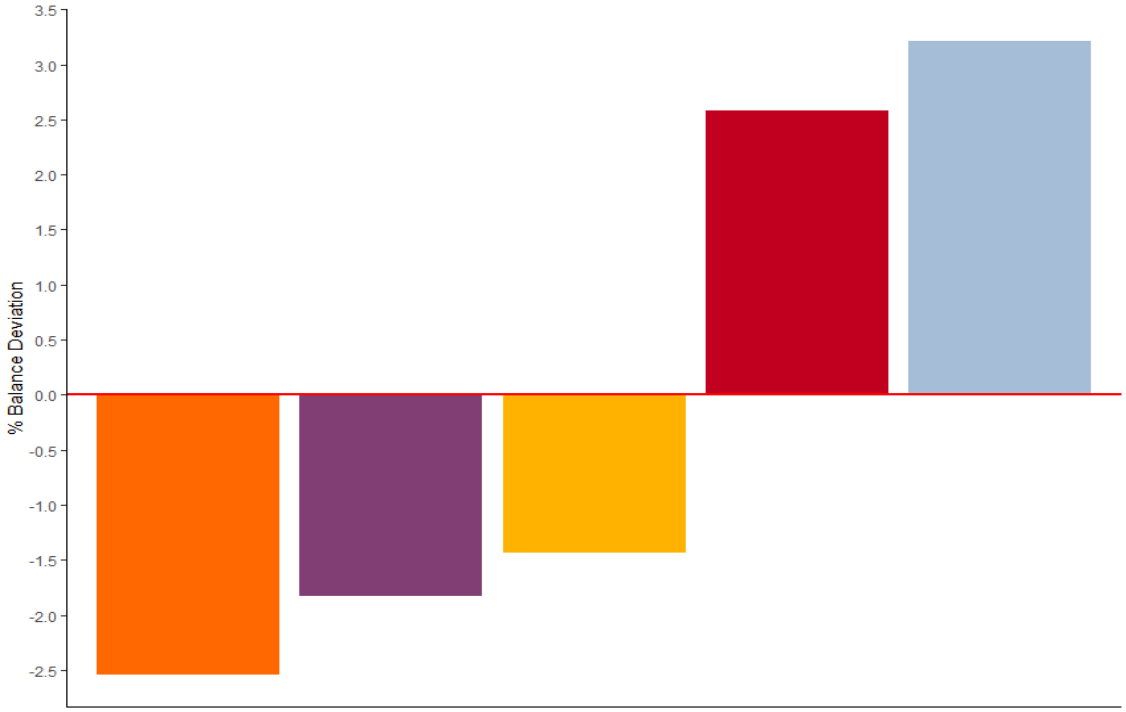
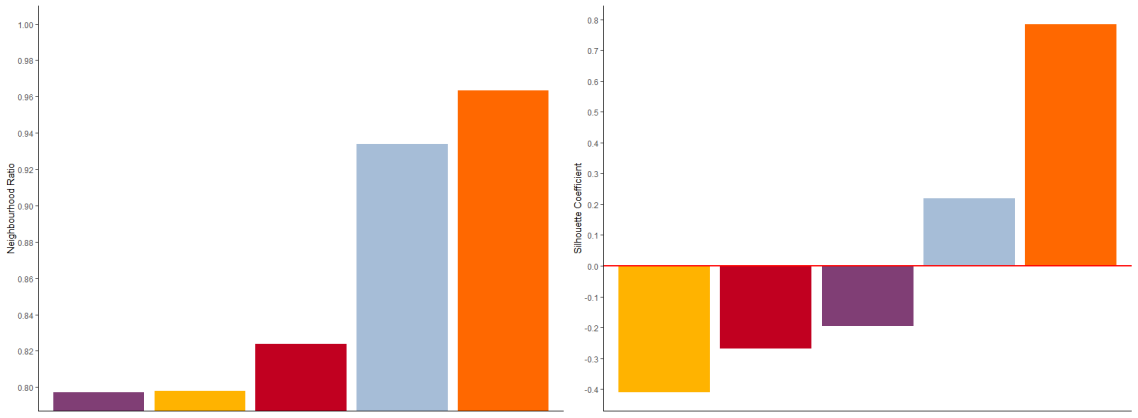


Figure 6.3: Poland Dispersion Cluster Sizes

Similarly, bar plots of the neighbourhood ratios and silhouette coefficients of the clusters are displayed in Figure 6.4. The neighbourhood ratios have an ordinary distribution, but a high frequency of the silhouette coefficients are negative.



(a) Neighbourhood Ratios

(b) Silhouette Coefficients

Figure 6.4: Poland Dispersion Cluster Measures

6.2.2 USA II

The KPIs over time for the USA II case are given in Table 6.3. This medium-sized case takes around 4 minutes to run completely. Note that calculating the intermediate KPIs also takes around 2 seconds each. The metachrosis typically runs instantly, whereas the Charity Search and Stretchy Search take up more significant time. Most time is however devoted to the Best of Random Center Selection Assignment, as is visible here, with almost $\frac{3}{4}$'th of the total time devoted to it with a little over 3 minutes. The jump in balance is a lot more pronounced here by going from 0.284 - 2.822 to 0.988 - 1.016, which falls within the strict 2% margin that is dictated for the USA II case. For this case the cycles starting from phase III reinforce the purpose for their existence as a significant improvement for the cluster KPIs can be observed. The Average Neighbourhood Ratio rises from 0.7998 to 0.8159 and a similar increase in the Average Silhouette Coefficient from 0.2525 to 0.2649 can be seen.

Table 6.3: Timeline KPIs USA II

KPI	I (5060)	II (43)	III (662)	IV (167)	V (100)	**VI** (72)
Mean Size (Custom Input)	-	-	-	-	-	**892336,36**
Standard Deviation	367646,03	377548,68	8060,06	9147,21	8831,47	**8383,55**
Ratio Maximum to Expected	2,822	2,865	1,016	1,015	1,016	**1,015**
Ratio Minimum to Expected	0,284	0,242	0,988	0,984	0,984	**0,984**
Average Neighbourhood Ratio	0,8583	0,8654	0,7998	0,8099	0,8134	**0,8159**
Average Silhouette Coefficient	0,3202	0,3255	0,2525	0,2602	0,2617	**0,2649**
Davies-Bouldin Index	0,4750	0,4694	1,6798	1,6898	1,6913	**1,6802**
Scaled Compactness	0,123	0,121	0,138	0,137	0,136	**0,135**
Scaled Centerpoint Distance	0,088	0,087	0,133	0,133	0,133	**0,133**
Cumulative Running Time	00:03:08	00:03:10	00:03:32	00:03:40	00:03:46	**00:03:52**

A global picture of the USA II case is shown in Figure 6.5. In the west-center part we see clusters that cover a large surface, as is to be expected from these less dense areas when dealing with balancing requirements. Conversely, very tight packed clusters cover the more populated areas to the east. The projection on the map does not respond to zoom perfectly, so it might appear that the the south eastern pink cluster protrudes upon the south eastern green cluster.

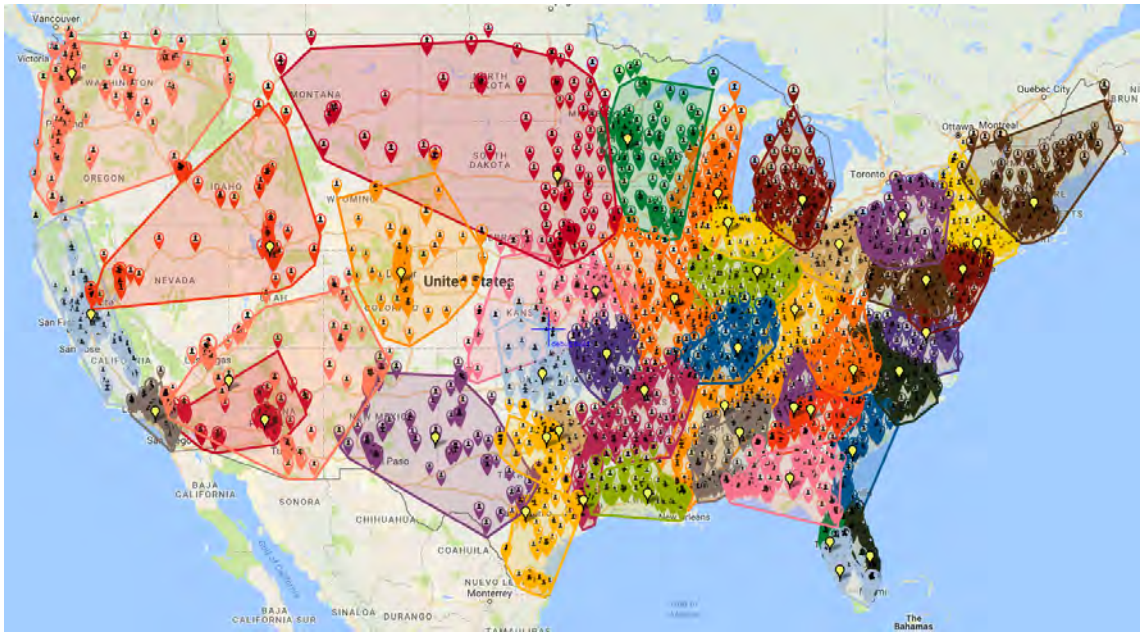


Figure 6.5: USA II Results Map

On closer inspection in the zoom shot in Figure 6.6, we can however see that fortunately this is not the case and there is a clear separation. The rest of this corner seems to be clustered adequately as well.

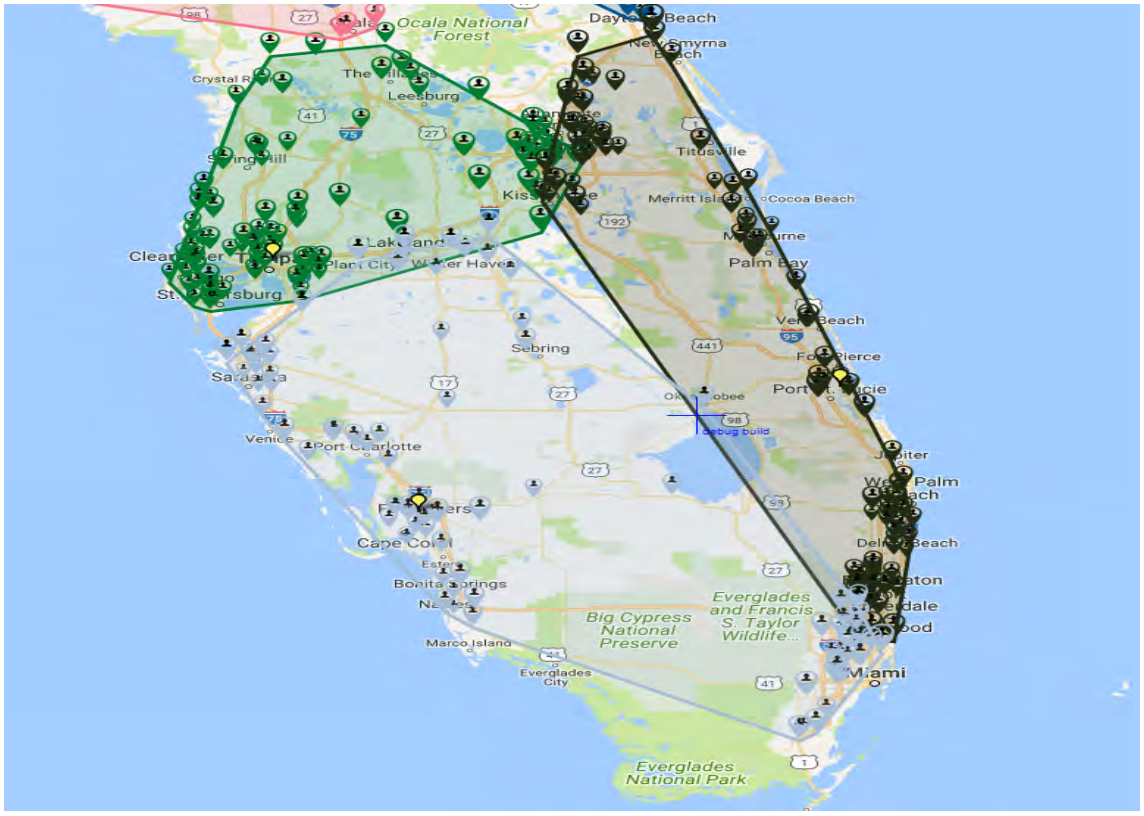


Figure 6.6: USA II Zoom Shot 1

To highlight another crucial difference in Euclidean and real distance, we zoom in on a large body of water, Lake Michigan. The orange, yellow and brown cluster perfectly surround it, instead of having half of a cluster be on one side of the lake and the other half on the other side. A scenario that is likely to occur when using Euclidean distances. The clusters that are showcased here are also ostentatiously well separated.

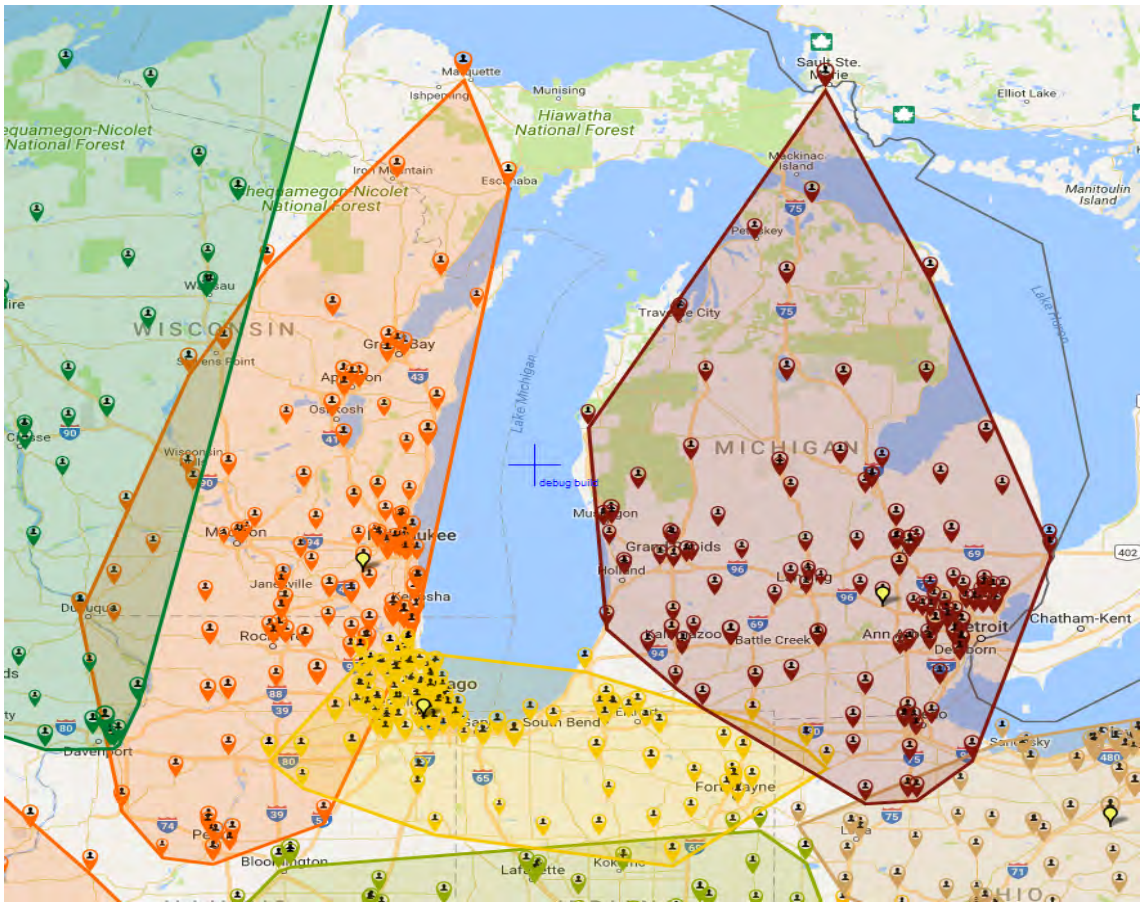


Figure 6.7: USA II Zoom Shot 2

There is also a very visible eyesore on the global map showcased in Figure 6.8. Fast heuristic algorithms carry their limits and this is a prime example. Do note that the majority of the red cluster's mass, near the yellow centroid, is close together and separated from the rest. The convex hull highlights the red stragglers more than is useful and the layout of the roads do favour this distribution between the pink and red cluster. Remarkable is that the pink cluster also holds the worst Silhouette Coefficient -0.2653 and a relatively bad Neighbourhood Ratio of 0.7545 validating the usefulness of the KPIs.

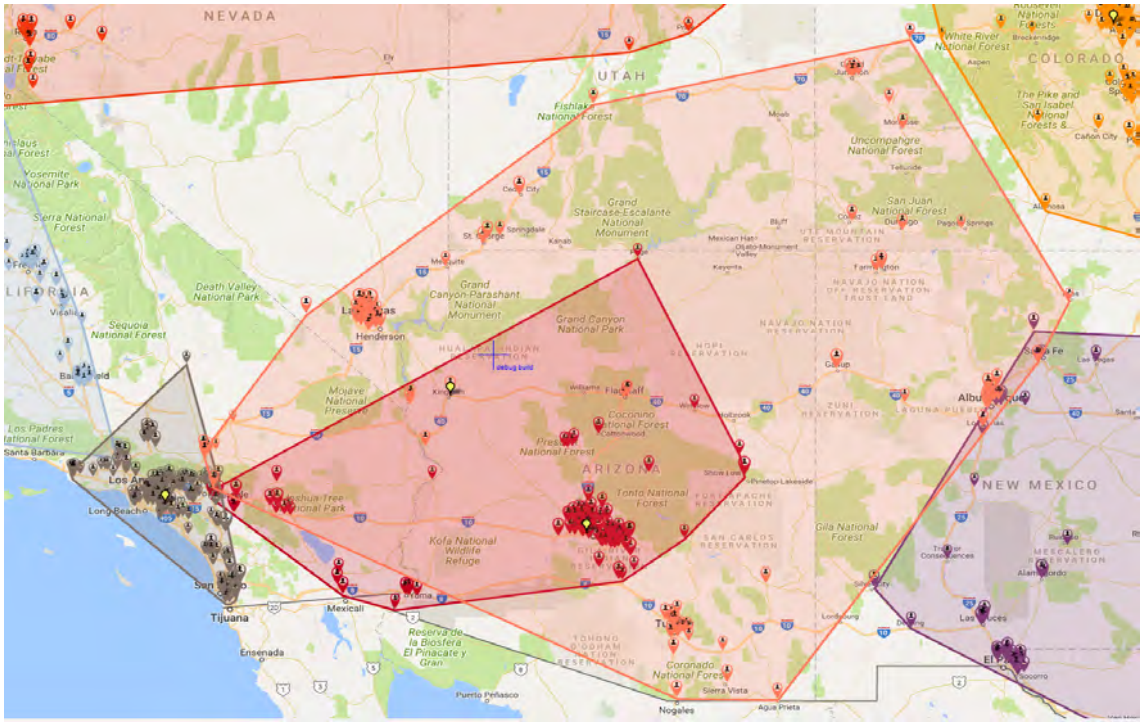


Figure 6.8: USA II Zoom Shot 3

The mass majority of clusters is however divided very well and the zoom shot shown in Figure 6.9 enforces that. All clusters have near perfect separation and the only visible overlap is because of the aforementioned convex hull visualisation limitations.

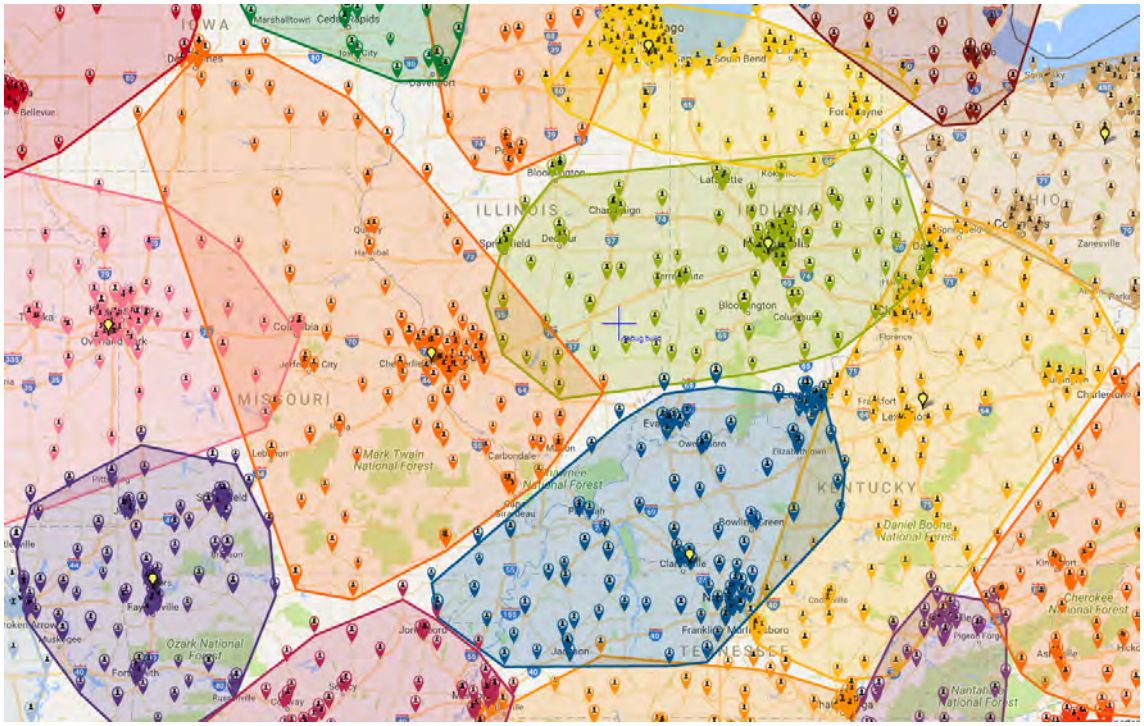


Figure 6.9: USA II Zoom Shot 4

A bar plot of the cluster size balance deviations for the USA II case is shown in Figure 6.10. It is notable that the bounds are between the desired margin of -2% and +2%. Nothing else particularly stands out.

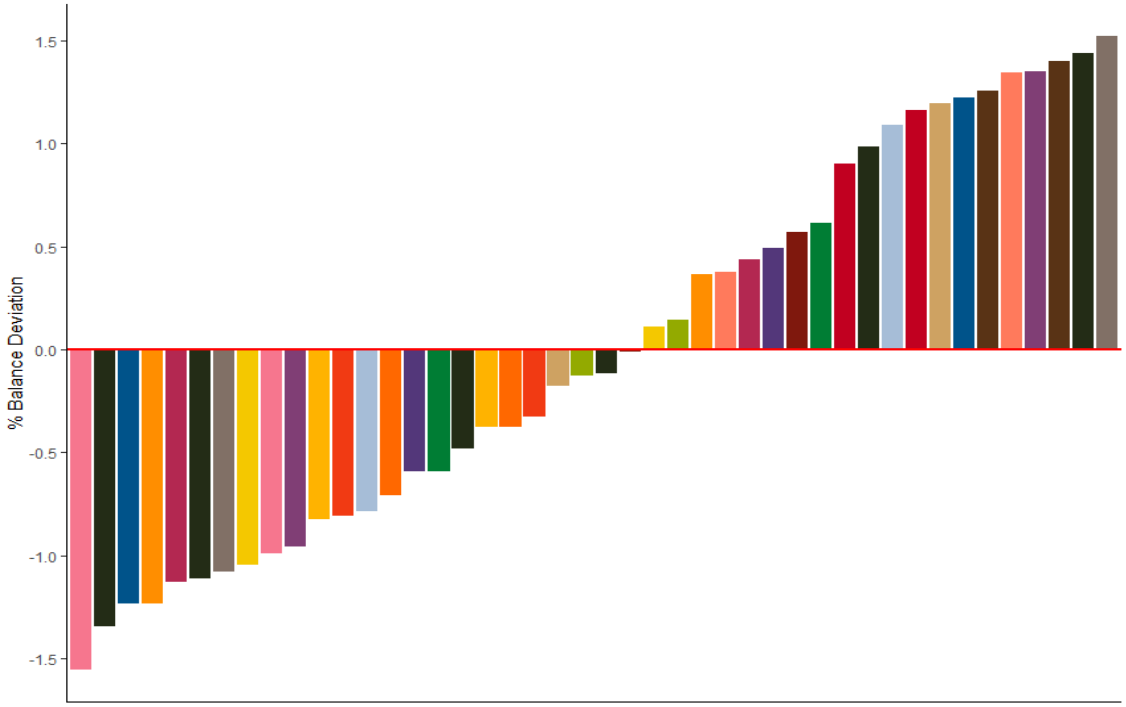
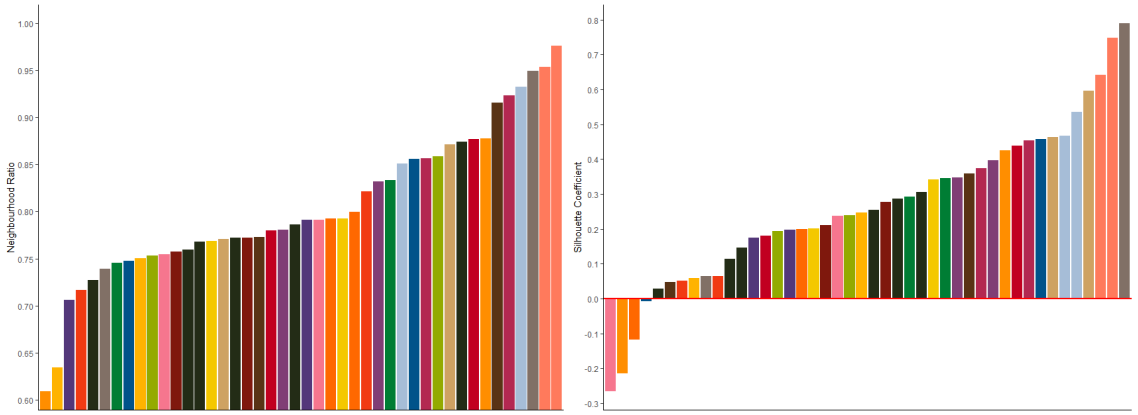


Figure 6.10: Dispersion Cluster Sizes

Bar plots of the neighbourhood ratios and silhouette coefficients of the clusters are displayed in Figure 6.11. There are few really low and really high neighbourhood ratios and more that are around 0.80. There are just 3 negative silhouette coefficients and 41 positive ones, which is positive for this case.



(a) Neighbourhood Ratios

(b) Silhouette Coefficients

Figure 6.11: USA II Dispersion Cluster Measures

6.2.3 Stockholm

The KPIs over time for the Stockholm case are given in Table 6.4. This medium-sized case takes around 2 minutes to run completely. Note the desired improvement in Average Neighbourhood Ratio from 0.8659 to 0.8748 after running Metachrosis. The balance requirements are once again met as being better than 5%. In this special case, where two balance requirements had to be fulfilled, both constraints were met. This is probably due to their high correlation. The main cluster KPIs end at respectably and respectively at 0.8206 and 0.1970. Noteworthy, is that the cycles once again improved them over time, while adhering to the balance constraint.

Table 6.4: Timeline KPIs Stockholm

KPI	I (3924)	II (30)	III (365)	IV (96)	V (125)	VI (68)
Mean Size (Kg + Working Time)	-	-	-	-	-	**10159,63**
Standard Deviation	3525,60	3503,35	324,86	293,54	87,98	**166,04**
Ratio Maximum to Expected	1,844	1,883	1,035	1,039	1,015	**1,034**
Ratio Minimum to Expected	0,384	0,384	0,960	0,964	0,985	**0,972**
Average Neighbourhood Ratio	0,8659	0,8748	0,8000	0,8120	0,8104	**0,8206**
Average Silhouette Coefficient	0,2583	0,2623	0,1905	0,1933	0,1919	**0,1970**
Davies-Bouldin Index	1,3987	1,4157	1,2939	1,3087	1,2804	**1,2647**
Scaled Compactness	0,222	0,221	0,252	0,248	0,249	**0,246**
Scaled Centerpoint Distance	0,152	0,152	0,183	0,183	0,184	**0,182**
Cumulative Running Time	00:01:05	00:01:07	00:01:27	00:01:33	00:01:44	**00:01:47**

A global picture of the Stockholm case is shown in Figure 6.12. A majority of tight clusters is near the center of Stockholm and a wealth of geographical boundaries can be spotted.

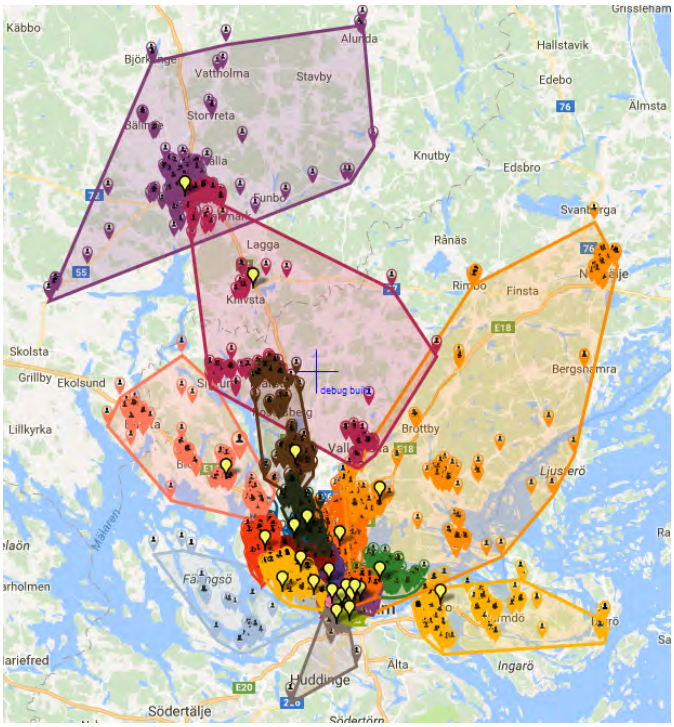


Figure 6.12: Stockholm Results Map

A zoom shot southwest shown in Figure 6.13 sheds some light on the apparent overlap. In actuality, it is actually perfectly connected with the mainland through the only land bridge and split very well in that sense.

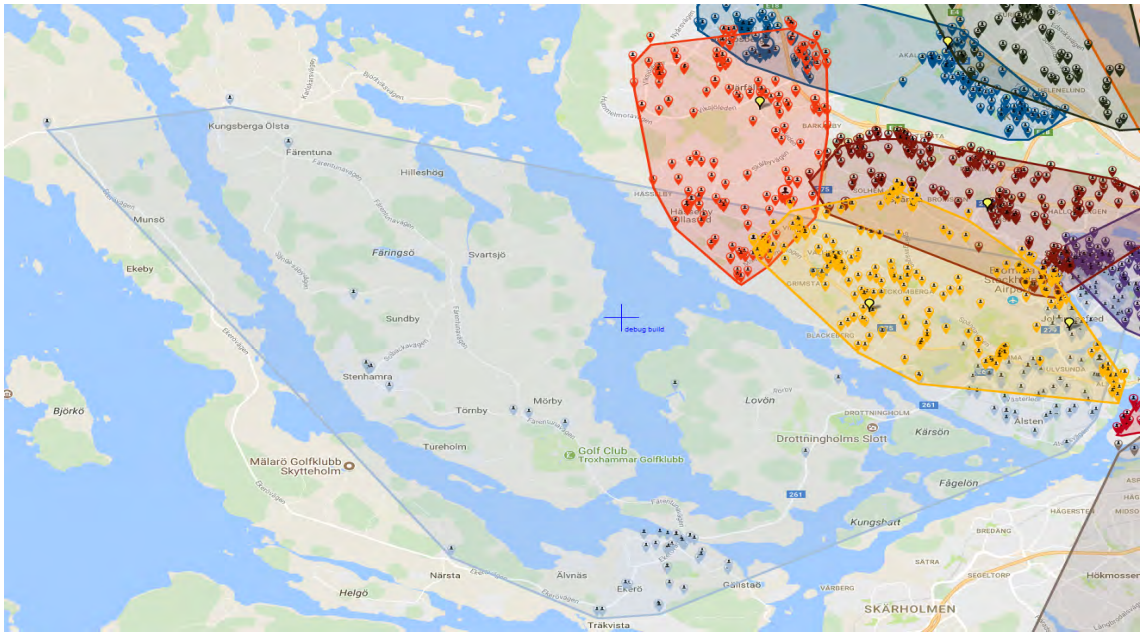


Figure 6.13: Stockholm Zoom Shot 1

A zoom shot near the middle is shown in Figure 6.14. This image really captures the large influence highway E4 has had on this particular clustering. This is exacerbated by the fact that the average linkage function within Charity Search tends to make clusters split in the middle of other clusters. Nevertheless the separation is still adequate, but this situation is still noteworthy.

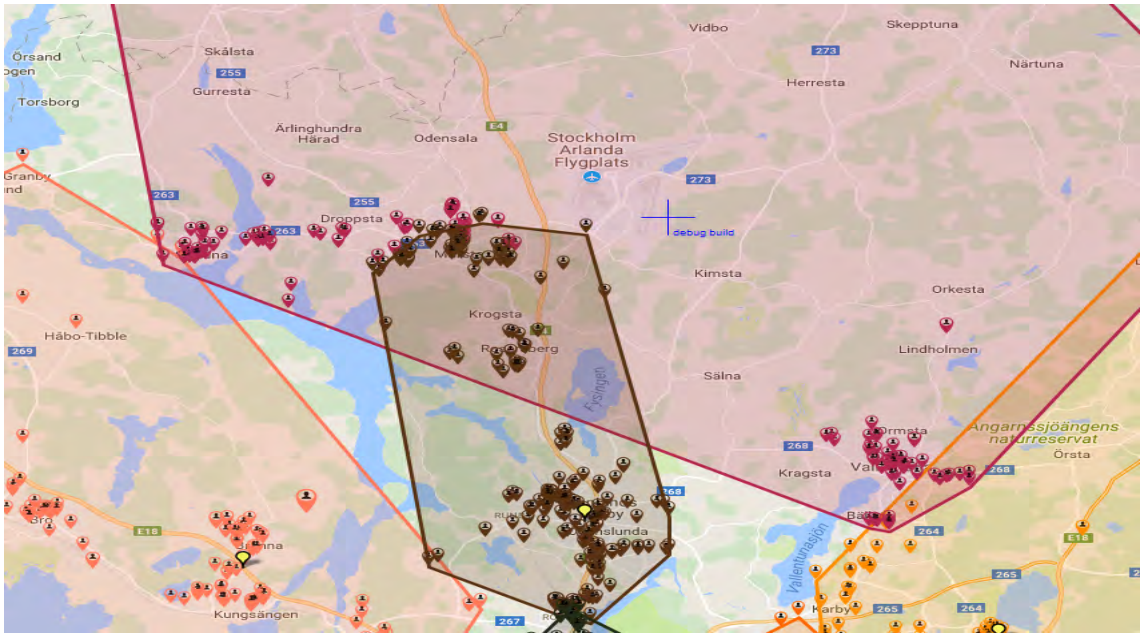


Figure 6.14: Stockholm Zoom Shot 2

A slightly zoomed in shot, which can be seen in Figure 6.15 features the diversity in point density that is present for almost all data cases. Obviously, this is due to varying population densities, but its impact on relying on just distance for algorithms or validation measures is immense and should be taken into account. Charity Search is largely indifferent between the scale of distance, which is precisely what is required.

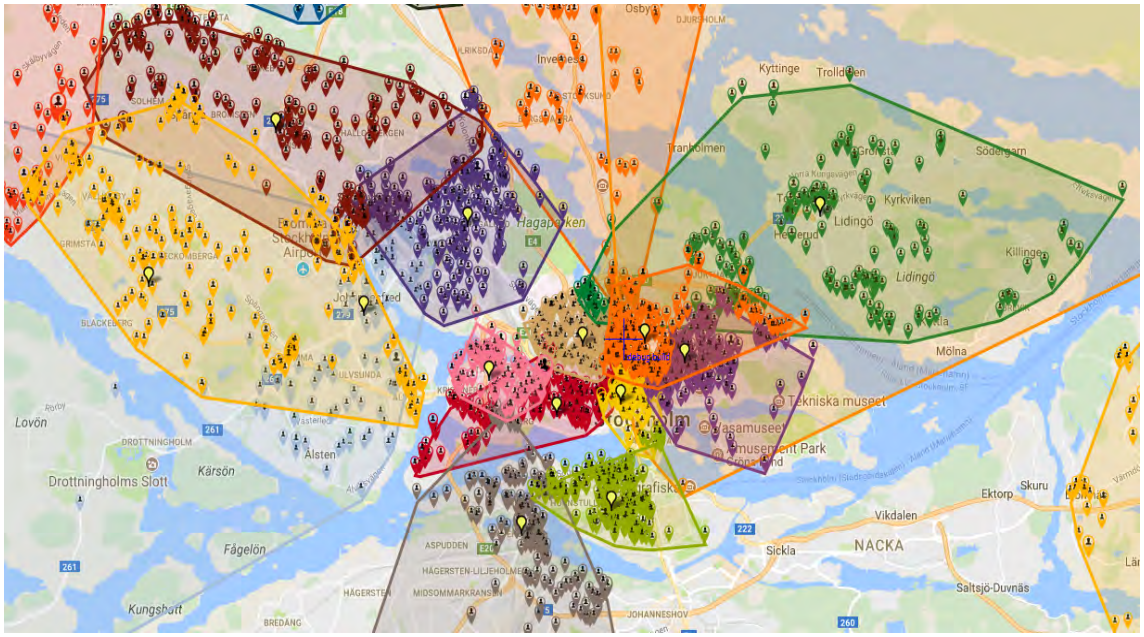


Figure 6.15: Stockholm Zoom Shot 3

A bar plot of the cluster size balance deviations for the Stockholm case is shown in Figure 6.16. The extremes are well within the required margin of -5% and +5%. This makes more abrupt jumps than the previous cases, which may be attributed to differing weight distributions.

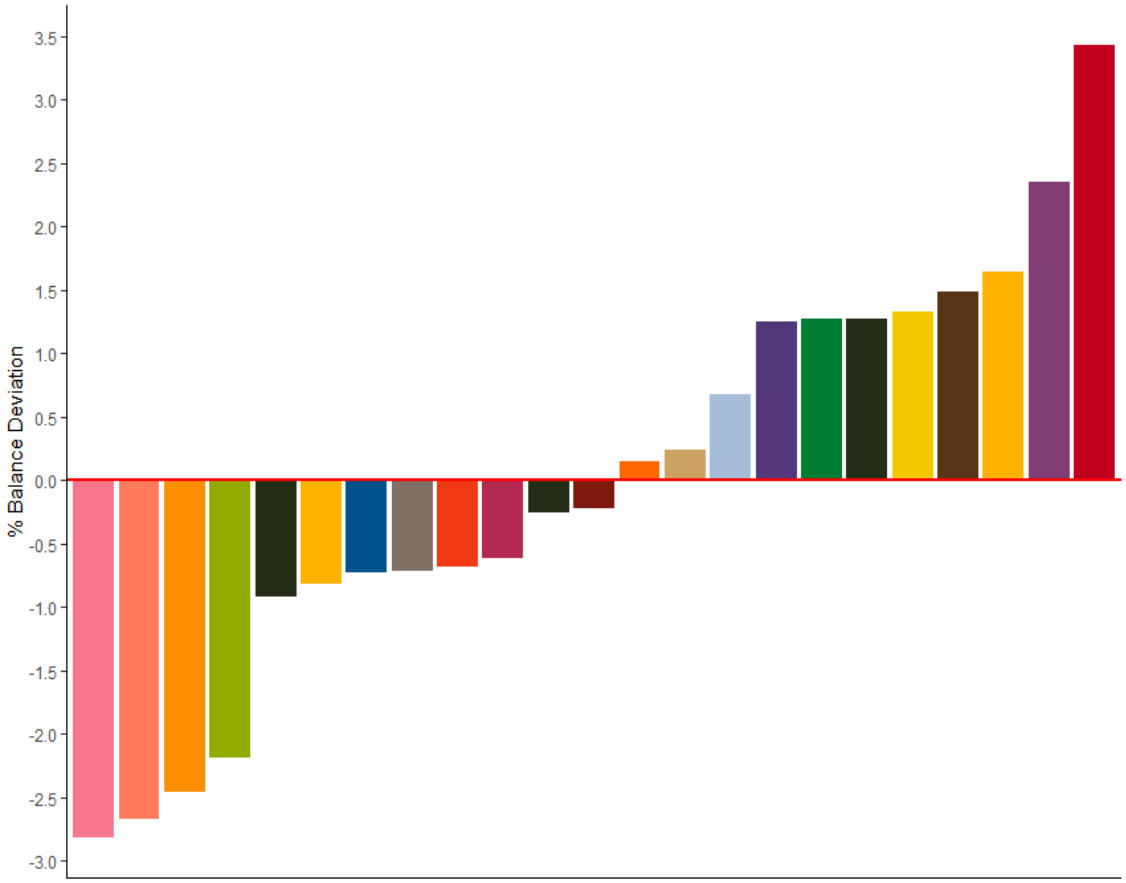
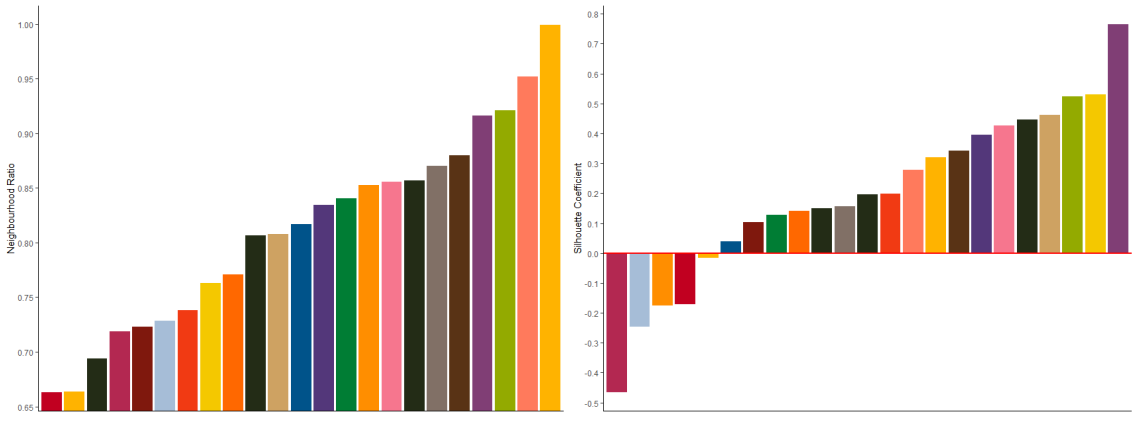


Figure 6.16: Dispersion Cluster Sizes

Bar plots of the neighbourhood ratios and silhouette coefficients of the clusters are displayed in Figure 6.17. The neighbourhood ratios do not show much variation in their dispersion. The silhouette coefficients show both an extreme negative and positive value indicating a great individual cluster and a terrible one.



(a) Neighbourhood Ratios

(b) Silhouette Coefficients

Figure 6.17: Stockholm Dispersion Cluster Measures

Chapter 7

Conclusion

This work strived to provide an alternative solution to the weight balanced clustering problem with geographical distances in an attempt to either replace or complement the current Territory Optimizer part of ORTEC Tactical Routing. A literature review revealed little related work, but did give a comprehensive overview of partitional, hierarchical, density-based and grid-based clustering. The Vehicle Routing Problem and balanced clustering applications were also briefly discussed.

The literature review revealed a wealth of distance based cluster quality measures that were not all equipped to handle the balance restriction combined with the varying point density present in a practical sense for our problem. Silhouette Coefficients were still deemed to have explanatory value and a custom made measure, the Neighbourhood Ratio, was developed. Balance measures were somewhat trivially found as the Ratios Minimum and Maximum to Expected cluster size plus the standard deviation of the cluster sizes.

Several unique methods were developed to tackle the weight balanced clustering problem with geographical distances. Charity Search reigned supreme, which is a heuristic local search method that transfers a client from the cluster with the most weight to the cluster with the least weight. Charity Search accomplishes this by first establishing neighbouring clusters for all clusters, after which a Breadth-First Search is ran to find the path from the most overloaded cluster to the most underloaded cluster. This path is followed by exchanging a client with the lowest average distance to the next cluster on the path. Ultimately, this improves the balance, while maintaining a nice geographical outlook.

An astounding twenty-two real-life datasets were prepared and analyzed of which twenty made it to testing. Eighteen of these used real distances, as some were deemed too large. Three cases: Poland, USA II and Stockholm were put under a microscope. The main data analysis result is that there is a small, yet not insignificant, pool of clients with an extremely large weight for nearly all cases. This fact had influenced the algorithmic progression and explained certain shortcomings of earlier algorithm incarnations.

The algorithmic scheme containing Charity Search led to extremely low running times with a maximum of 10 minutes for the largest case, yet excellent solutions simultaneously. All cases had their balance requirements fulfilled and showed positive results on the cluster KPIs like neighbour-

hood ratios within the 0.80's range. All visualisations showed few misplaced points giving a nice aesthetic, yet did not have none all the time. This is a result of the inescapable reality of using heuristics that general perfection is unobtainable. These points did make sense in the context of the algorithmic progression and were identifiable by their local KPIs.

Moreover, Poland, USA II and Stockholm's KPIs were systemically investigated over time with the conclusion that the construction heuristic took the majority of the running time and that Charity Search was highly effective at balancing the cluster sizes. Additional zoom shots of the map also enforced the ease with which Charity Search handles two main hurdles of this problem: geographical distances as opposed to Euclidean and varying point density. On top of that the dispersions of the cluster size, neighbourhood ratios and silhouette coefficients were researched by plotting the distribution.

Extensions to the current implementation that can relatively easily be added exist. For instance, true multiple balance limitations as opposed to handling it as a singular weighted objective. The guiding metric for the Charity Search should, in that case, instead be the current largest relative difference with the mean over all balance criteria. Having every cluster be of equal size is also something that may be relaxed to a concept akin to capacities or target weight. This could either be implemented as proportionally changing how much the weight of a client attributes to a cluster or changing the deviation from the mean metric to a deviation from the capacity directly.

Future research should focus on improving or replacing the Best of Random Center Selection Assignment, as it takes up the majority of computation time of the whole algorithmic scheme and is relatively simplistic. Small isolated groups of clients sometimes occur, because of the Charity Search's design using the average linkage function. This is an issue that is minor and only appears sparingly with frequent use of Charity Search. It does not arise if Charity Search is used as a complement instead of being the sole solution method. Nevertheless, it is worthy of a close inspection. Especially, since many fruitless attempts at fixing the issue were made within the scope of this thesis. Future research can also focus on exploring different avenues towards finding a solution to the weight balanced clustering problem like evolutionary computing or (integer) linear programming. A definitive identifyable tradeoff between cluster and balance KPIs is also desirable. Finally, the impact of the clustering on the ultimate vehicle routing solution should be quantified, but this also depends on the routing algorithm in place.

Recommendations to ORTEC would be to include the full algorithmic scheme, standalone Charity Search and standalone Metachrosis as additional options for the Territory Optimizer within ORTEC Tactical Routing. Our algorithmic scheme is generally an order of magnitude faster, but does not produce a perfect solution in general, so removing the prior Territory Optimizer is not ideal. Moreover, the visualisation tools produced within this thesis can be of value to ORTEC, because the weight dependent size of client markers is for instance more clear. "Bad" points can also be highlighted based on local KPIs to assist the user in decision-making. The convex hull and additional colours also provide a more enlightening aesthetic. It also is not reliant on an in-house loaded map, but can project using popular online mapping services like Google Maps or Bing Maps, provided that licenses are acquired, which are updated in real time requiring no manual updating. Lastly, the Silhoutte Coefficient and Neighbourhood Ratio can be listed as additional KPIs.

Appendix A

Cases

A.1 Denmark East I

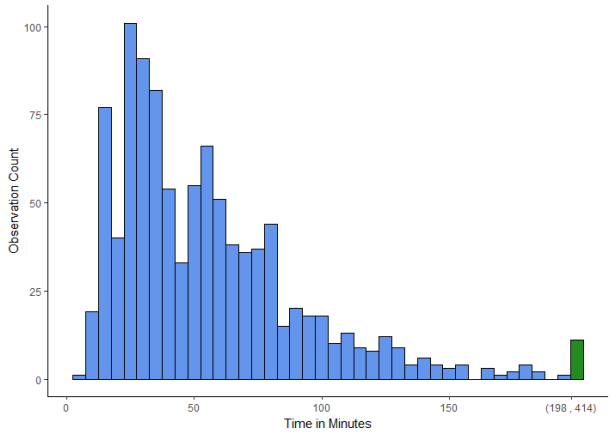


Figure A.1: Denmark East I Histogram

Table A.1: Denmark East I Statistics

Minimum Value	6
Maximum Value	414
Mean (Standard Deviation)	56.32 (40.24)
Observation Count	1002
Unique Observation Count	154
Mode (count) 2nd Mode (count) 3rd Mode (count)	24 (32) 54 (27) 26 (26)

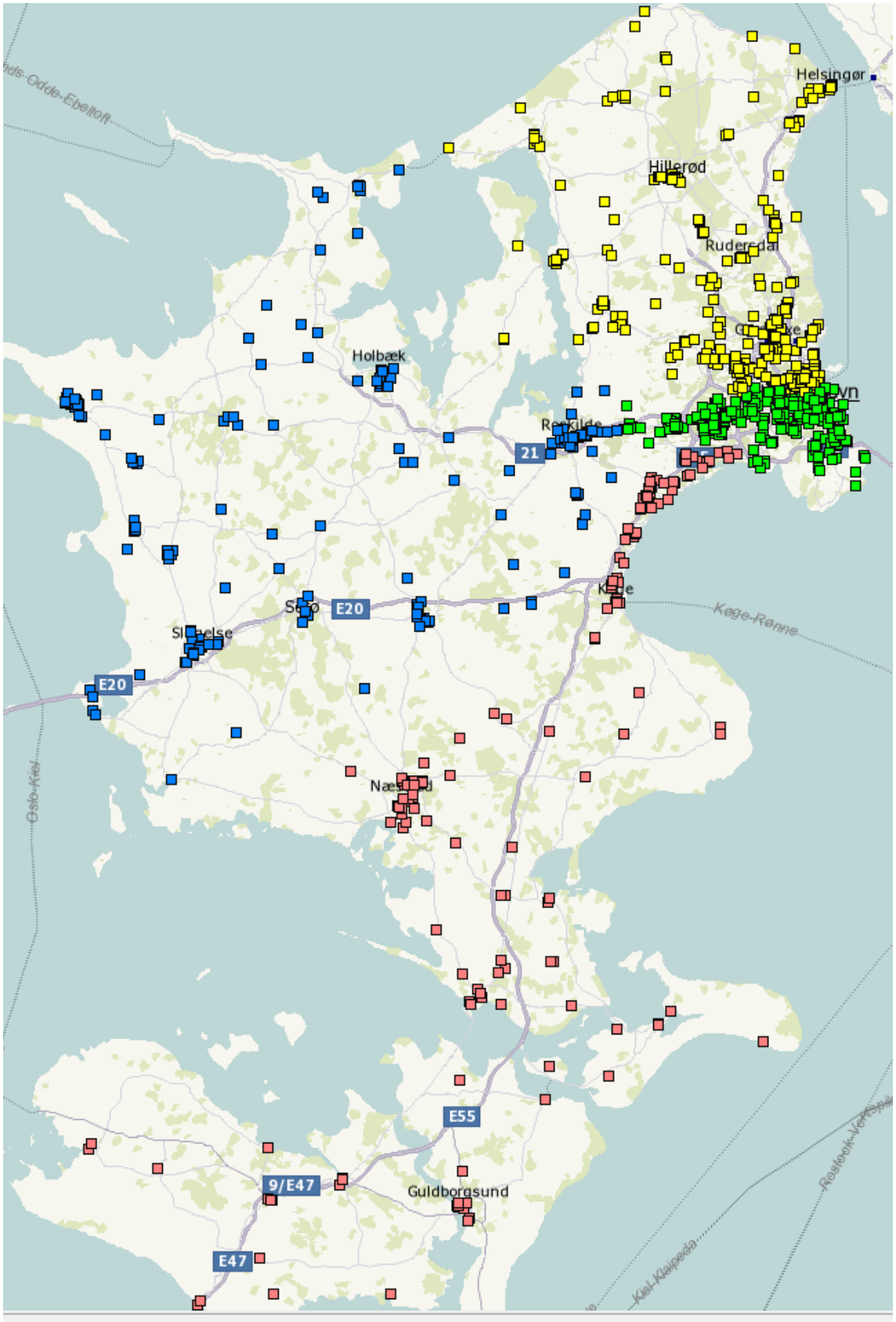


Figure A.2: Denmark East I OTR

Table A.2: Denmark East I Results KPIs

Mean Size (Working Time)	14107,25
Standard Deviation	622,22
Ratio Maximum to Expected	1,038
Ratio Minimum to Expected	0,937
Average Neighbourhood Ratio	0,8625
Average Silhouette Coefficient	0,2237
Davies-Bouldin Index	1,3037
Scaled Compactness	0,635
Scaled Centerpoint Distance	0,437
Running Time Total	00:00:02
Running Time Algorithm	00:00:01

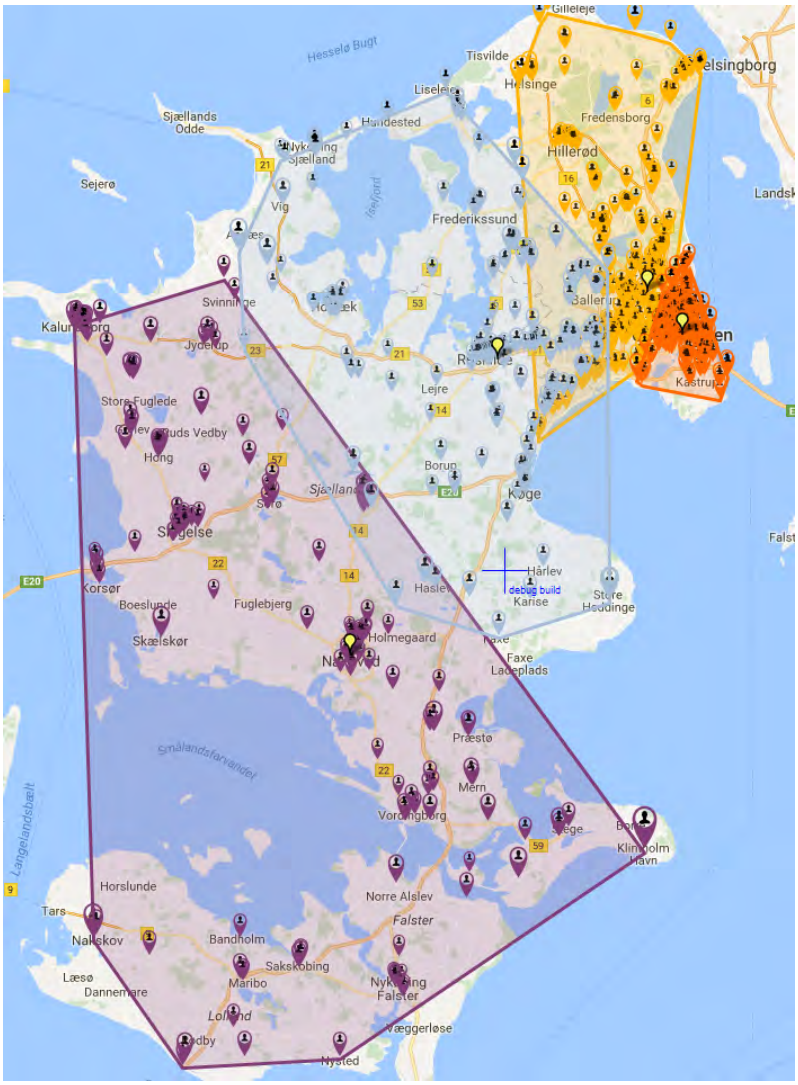


Figure A.3: Denmark East I Results Map

A.2 Denmark East II

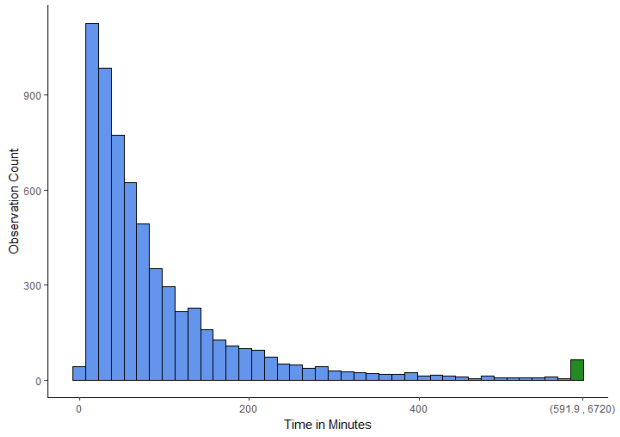


Table A.3: Denmark East II Statistics

Minimum Value	4
Maximum Value	6720
Mean (Standard Deviation)	96.32 (153.292)
Observation Count	6304
Unique Observation Count	500
Mode (count) 2nd Mode (count) 3rd Mode (count)	32 (215) 16 (206) 8 (177)

Figure A.4: Denmark East II Histogram

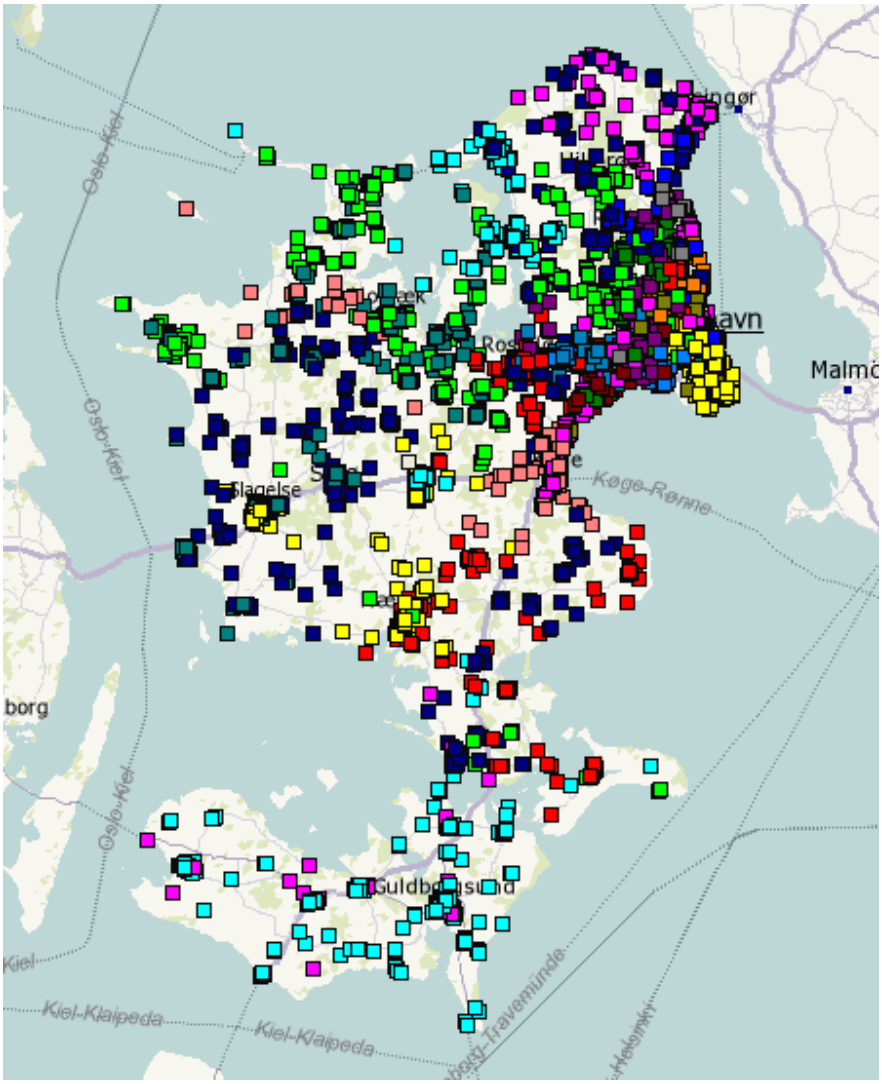


Figure A.5: Denmark East II OTR

Table A.4: Denmark East II Results KPIs

Mean Size (Working Time)	21664,75
Standard Deviation	744,95
Ratio Maximum to Expected	1,055
Ratio Minimum to Expected	0,955
Average Neighbourhood Ratio	0,8132
Average Silhouette Coefficient	0,2343
Davies-Bouldin Index	0,7304
Scaled Compactness	0,227
Scaled Centerpoint Distance	0,160
Running Time Total	00:03:46
Running Time Algorithm	00:02:51

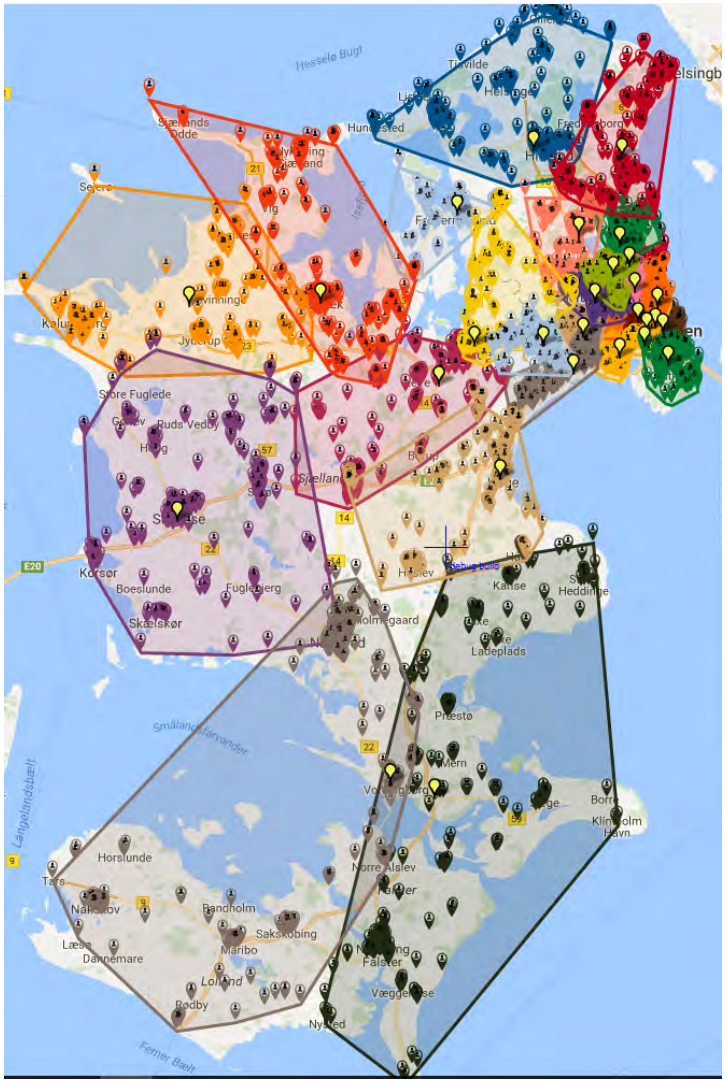


Figure A.6: Denmark East II Results Map

A.3 Denmark West

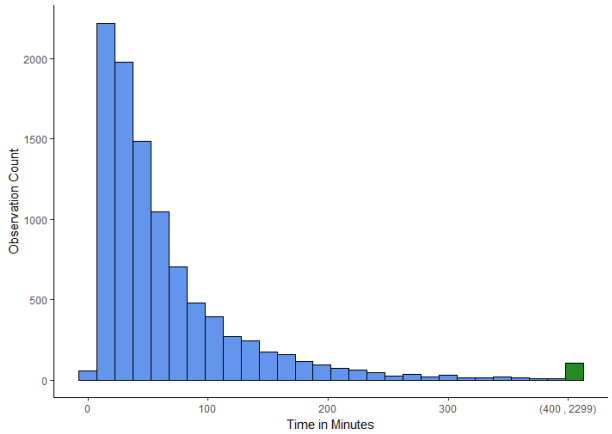


Figure A.7: Denmark West Histogram

Table A.5: Denmark West Statistics

Minimum Value	4
Maximum Value	2299
Mean (Standard Deviation)	68.36 (87.84)
Observation Count	9912
Unique Observation Count	437
Mode (count)	16 (387)
2nd Mode (count)	24 (310)
3rd Mode (count)	32 (254)

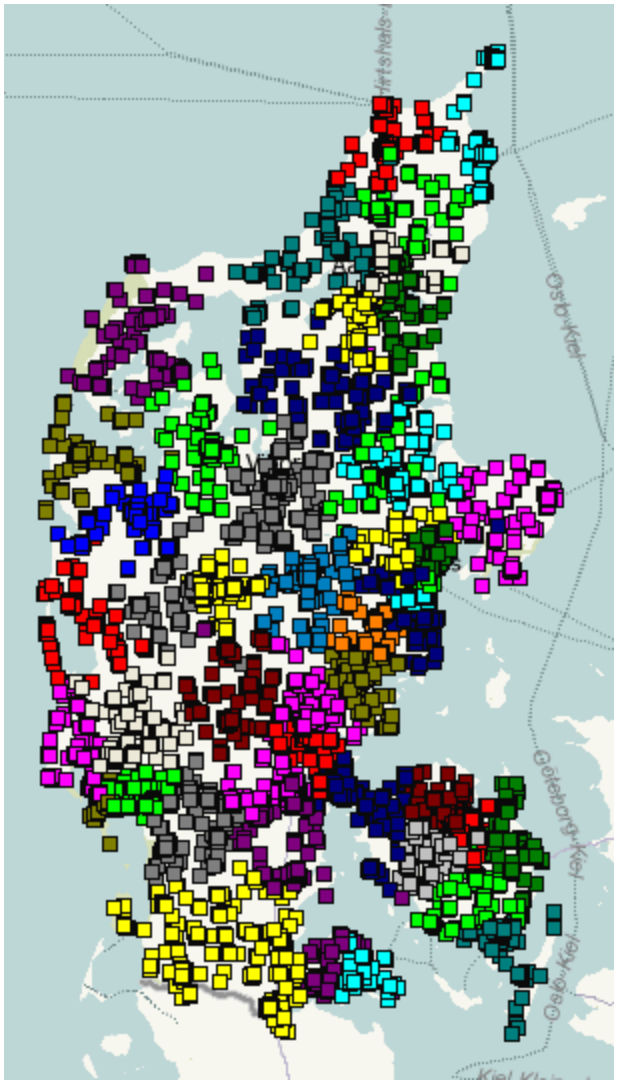


Figure A.8: Denmark West OTR

Table A.6: Denmark West Results KPIs

Mean Size (Working Time)	13552,32
Standard Deviation	481,73
Ratio Maximum to Expected	1,048
Ratio Minimum to Expected	0,953
Average Neighbourhood Ratio	0,8223
Average Silhouette Coefficient	0,2144
Davies-Bouldin Index	0,9500
Scaled Compactness	0,127
Scaled Centerpoint Distance	0,11
Running Time Total	00:12:04
Running Time Algorithm	00:11:12

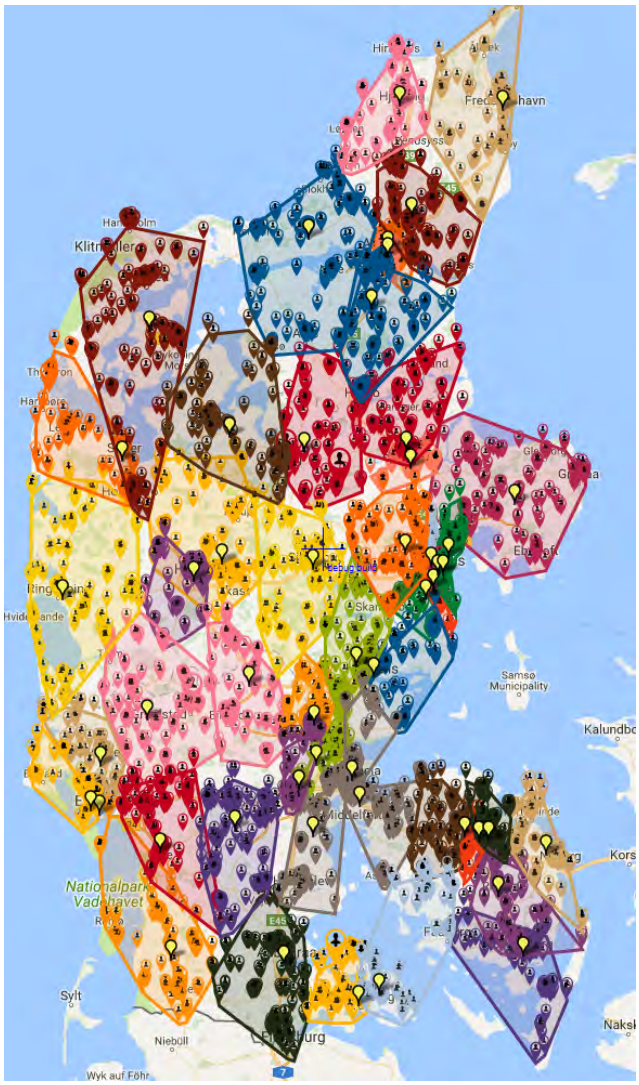


Figure A.9: Denmark West Results Map

A.4 Arlöv

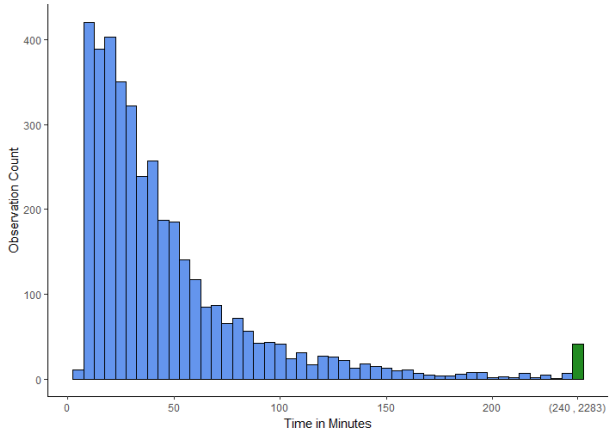


Figure A.10: Arlöv Histogram

Table A.7: Arlöv Statistics

Minimum Value	4
Maximum Value	2283
Mean (Standard Deviation)	48.14 (60.59)
Observation Count	3855
Unique Observation Count	243
Mode (count)	16 (163)
2nd Mode (count)	24 (116)
3rd Mode (count)	8 (112)

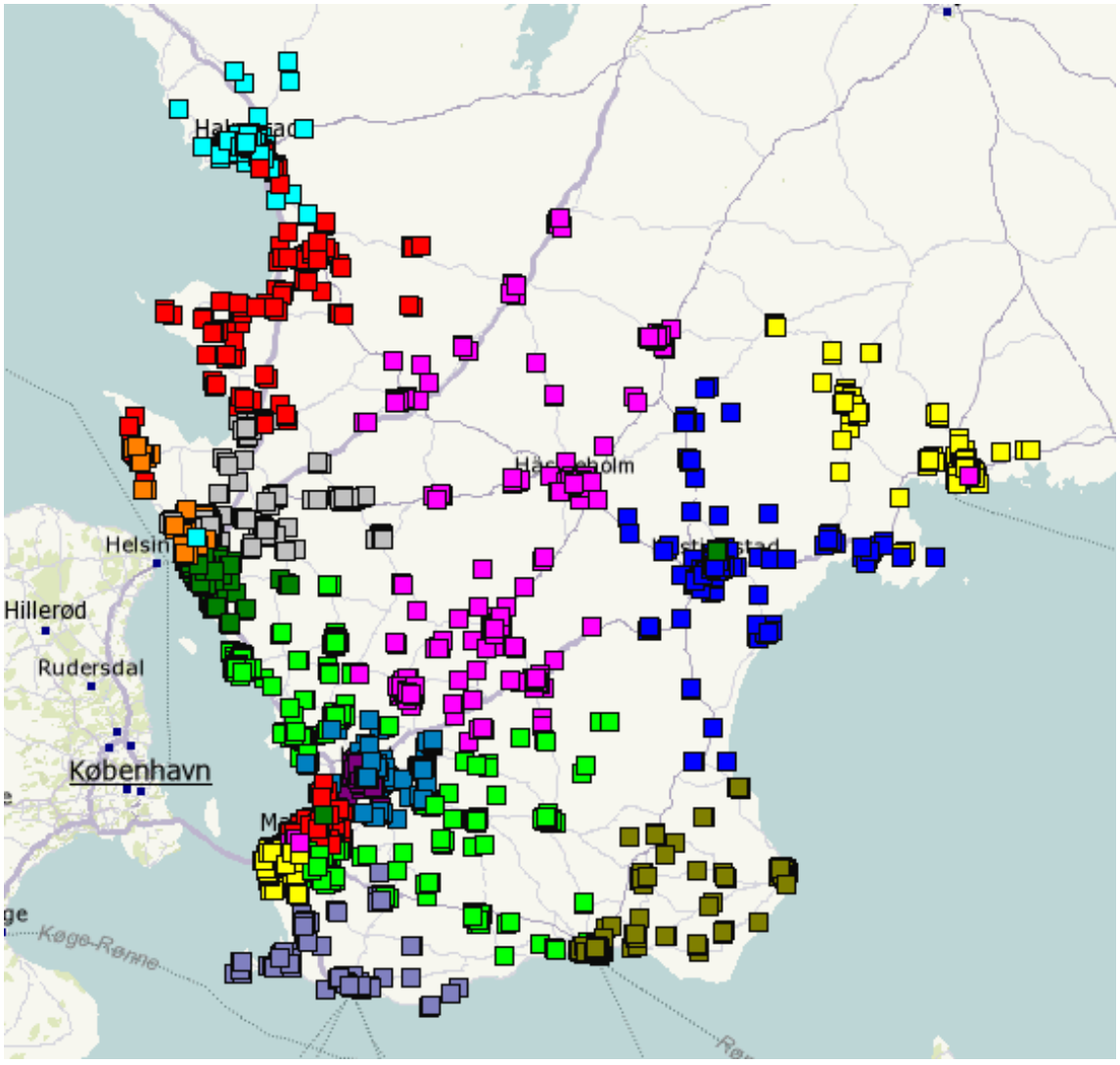


Figure A.11: Arlöv OTR

Table A.8: Arlöv Results KPIs

Mean Size (Working Time)	9767,63
Standard Deviation	229,47
Ratio Maximum to Expected	1,023
Ratio Minimum to Expected	0,962
Average Neighbourhood Ratio	0,8143
Average Silhouette Coefficient	0,1736
Davies-Bouldin Index	0,7730
Scaled Compactness	0,215
Scaled Centerpoint Distance	0,154
Running Time Total	00:01:11
Running Time Algorithm	00:00:54

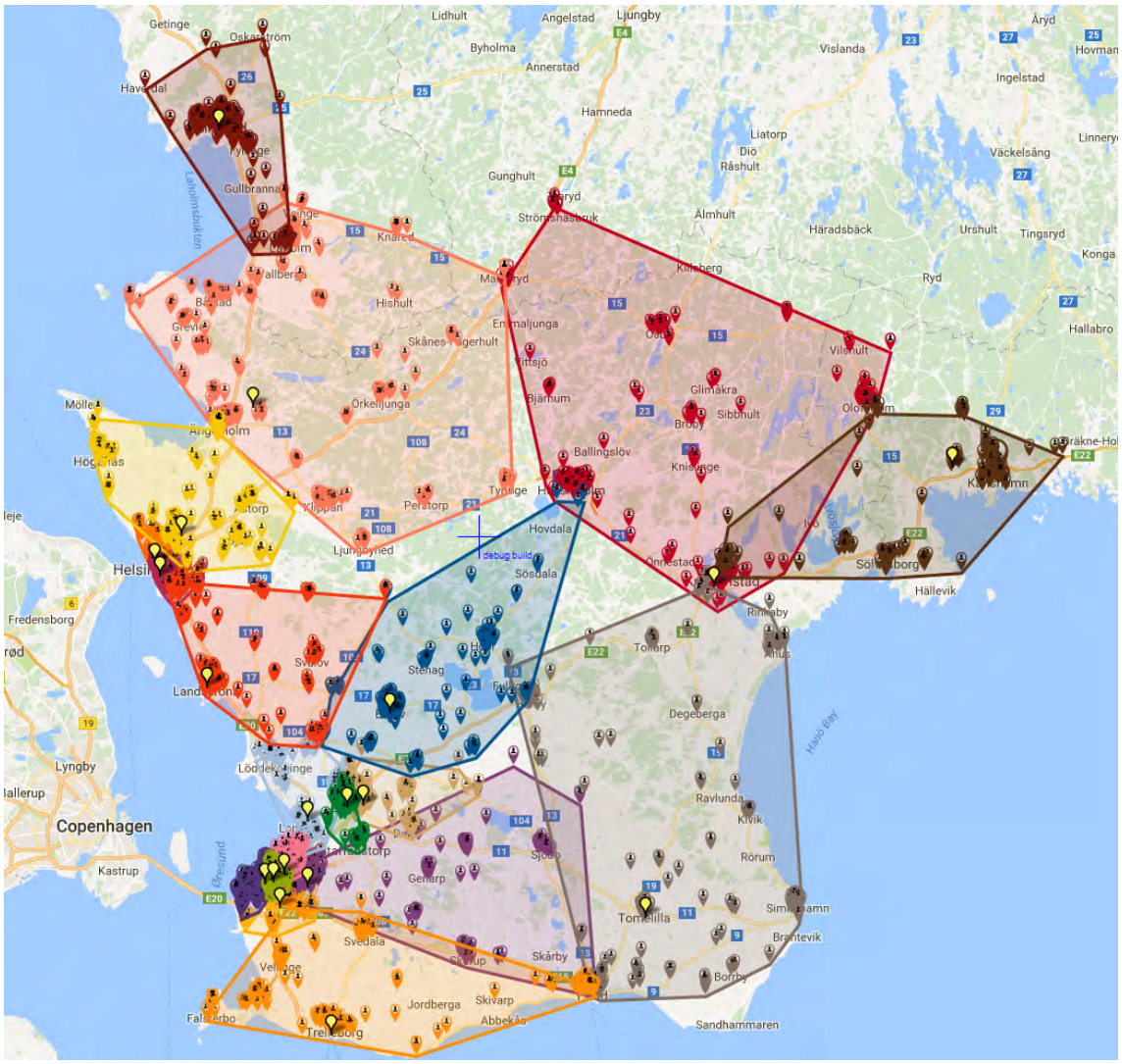


Figure A.12: Arlöv Results Map

A.5 Göteborg

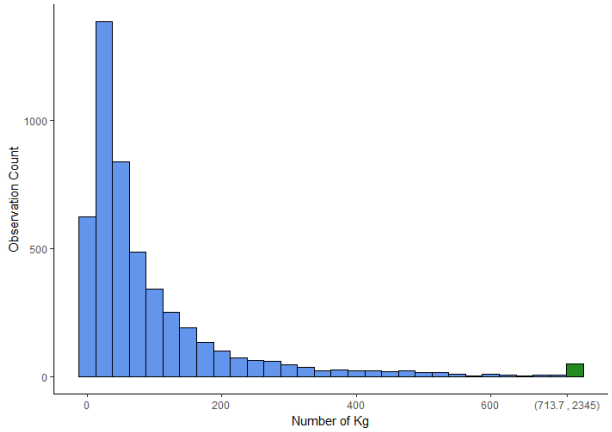


Figure A.13: Göteborg Histogram

Table A.9: Göteborg Statistics

Minimum Value	0
Maximum Value	2345
Mean (Standard Deviation)	97 (148.12)
Observation Count	4884
Unique Observation Count	525
Mode (count)	7 (114)
2nd Mode (count)	17 (112)
3rd Mode (count)	13 (98)

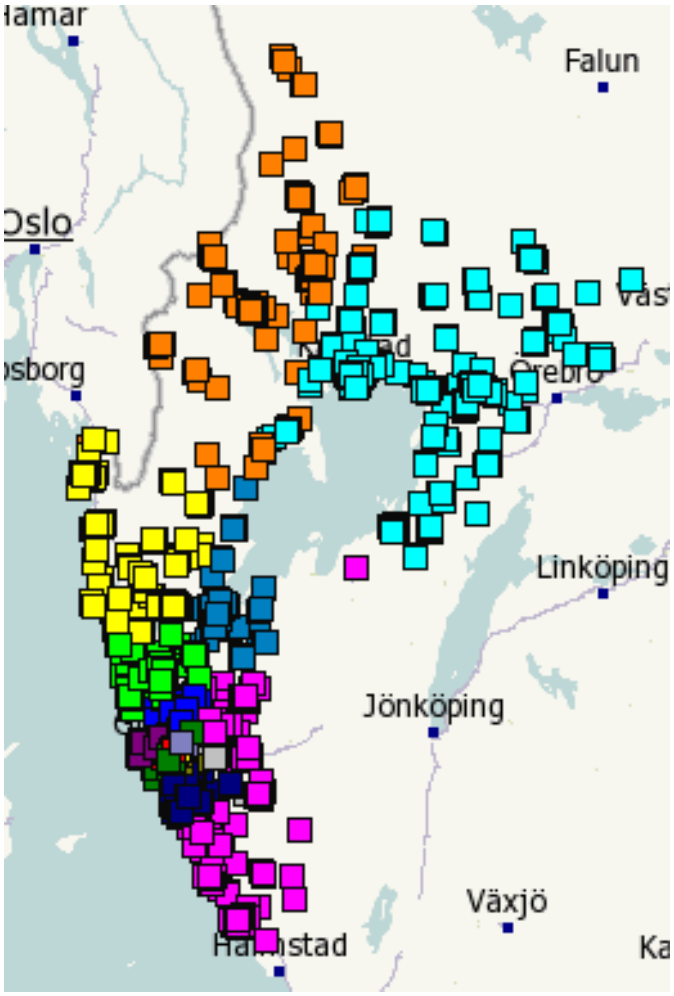


Figure A.14: Göteborg OTR

Table A.10: Göteborg Results KPIs

Mean Size (Kg)	21533,55
Standard Deviation	1272,09
Ratio Maximum to Expected	1,073
Ratio Minimum to Expected	0,920
Average Neighbourhood Ratio	0,8212
Average Silhouette Coefficient	0,1207
Davies-Bouldin Index	2,3505
Scaled Compactness	0,202
Scaled Centerpoint Distance	0,143
Running Time Total	00:02:22
Running Time Algorithm	00:01:50

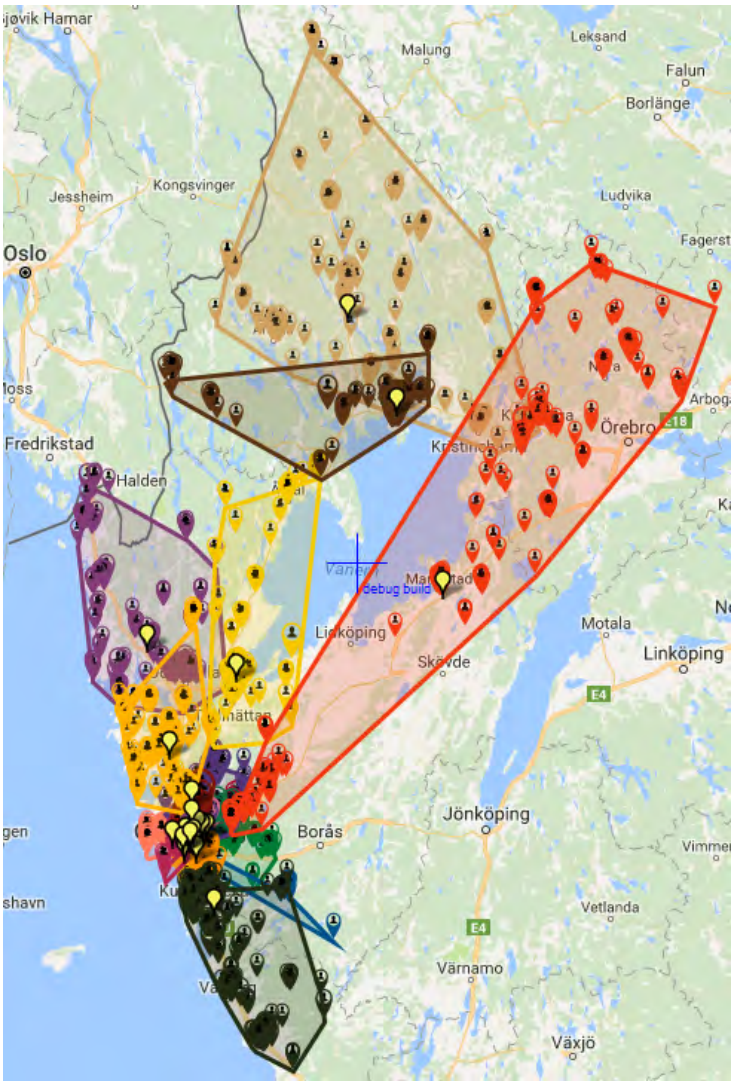


Figure A.15: Göteborg Results Map

A.6 Jönköping

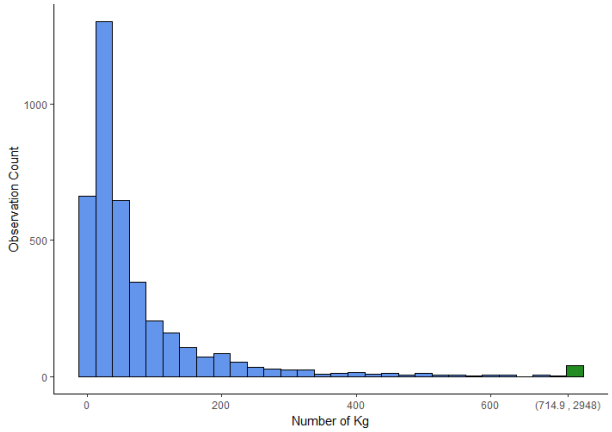


Figure A.16: Jönköping Histogram

Table A.11: Jönköping Statistics

Minimum Value	2
Maximum Value	2948
Mean (Standard Deviation)	80.25 (144.49)
Observation Count	3884
Unique Observation Count	423
Mode (count)	7 (130)
2nd Mode (count)	13 (104)
3rd Mode (count)	17 (102)

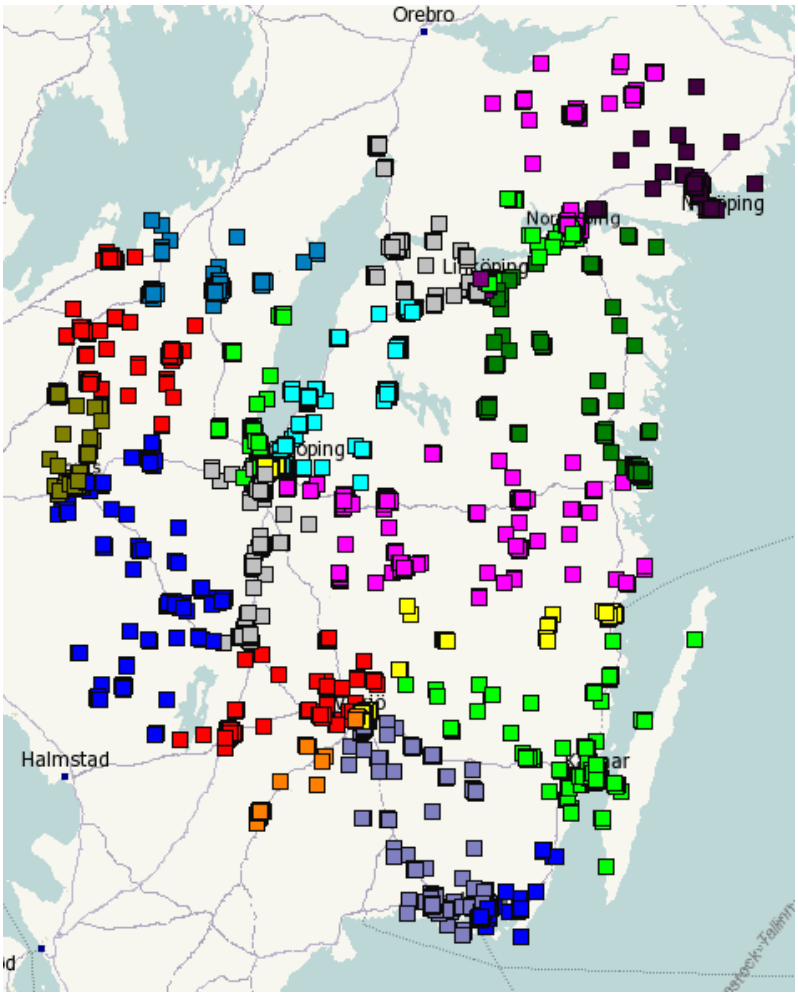


Figure A.17: Jönköping OTR

Table A.12: Jönköping Results KPIs

Mean Size (Kg)	15584,90
Standard Deviation	1469,67
Ratio Maximum to Expected	1,134
Ratio Minimum to Expected	0,854
Average Neighbourhood Ratio	0,8197
Average Silhouette Coefficient	0,1944
Davies-Bouldin Index	1,0341
Scaled Compactness	0,197
Scaled Centerpoint Distance	0,144
Running Time Total	00:01:14
Running Time Algorithm	00:00:57

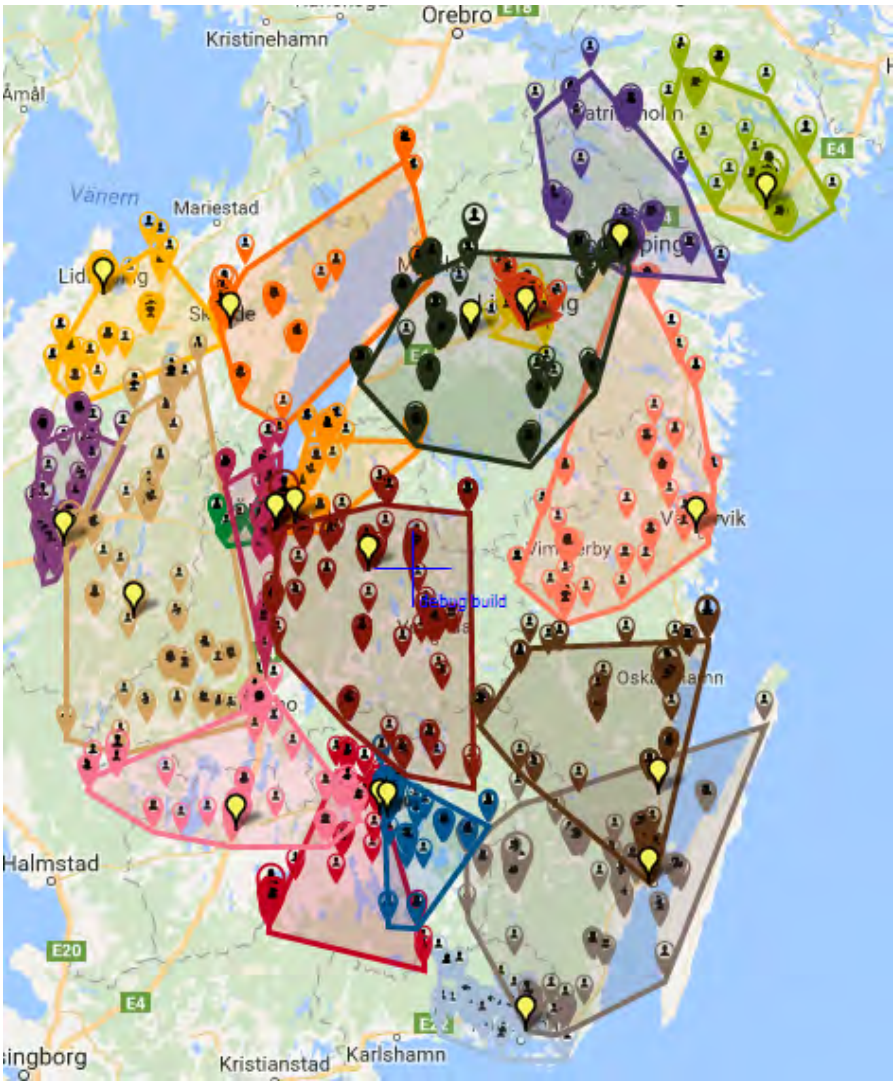


Figure A.18: Jönköping Results Map

A.7 Stockholm

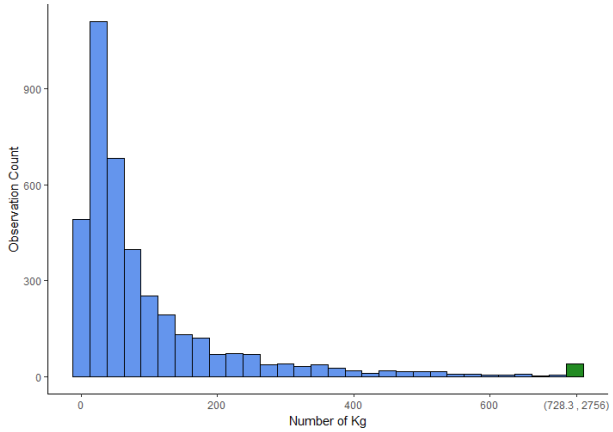


Figure A.19: Stockholm, Kg Histogram

Table A.13: Stockholm, Kg Statistics

Minimum Value	0
Maximum Value	2756
Mean (Standard Deviation)	99.37 (154.31)
Observation Count	3924
Unique Observation Count	487
Mode (count)	0 (93)
2nd most common (count)	13 (70)
3rd most common (count)	23 (69)

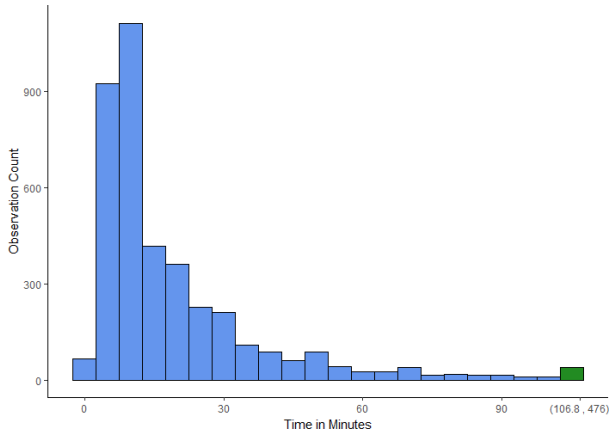


Figure A.20: Stockholm, Working Time Histogram

Table A.14: Stockholm, Working Time Statistics

Minimum Value	2
Maximum Value	476
Mean (Standard Deviation)	19.73 (23.70)
Observation Count	3924
Unique Observation Count	125
Mode (count)	6 (442)
2nd most common (count)	8 (438)
3rd most common (count)	12 (339)

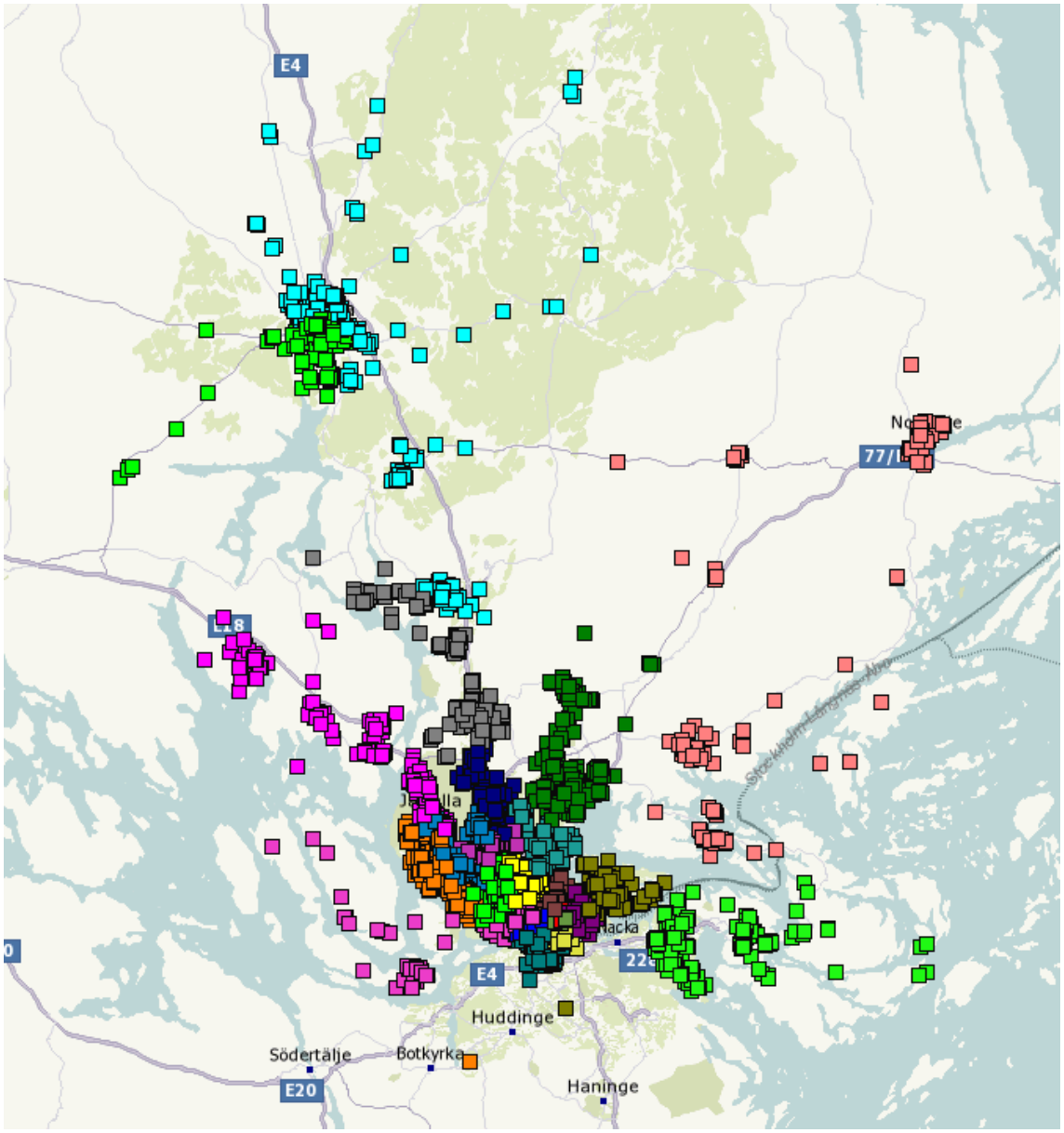


Figure A.21: Stockholm OTR

Table A.15: Stockholm Results KPIs

Mean Size (Kg and Working Time)	10159,63
Standard Deviation	166,04
Ratio Maximum to Expected	1,034
Ratio Minimum to Expected	0,972
Average Neighbourhood Ratio	0,8206
Average Silhouette Coefficient	0,1970
Davies-Bouldin Index	1,2327
Scaled Compactness	0,246
Scaled Centerpoint Distance	0,173
Running Time Total	00:01:25
Running Time Algorithm	00:01:08

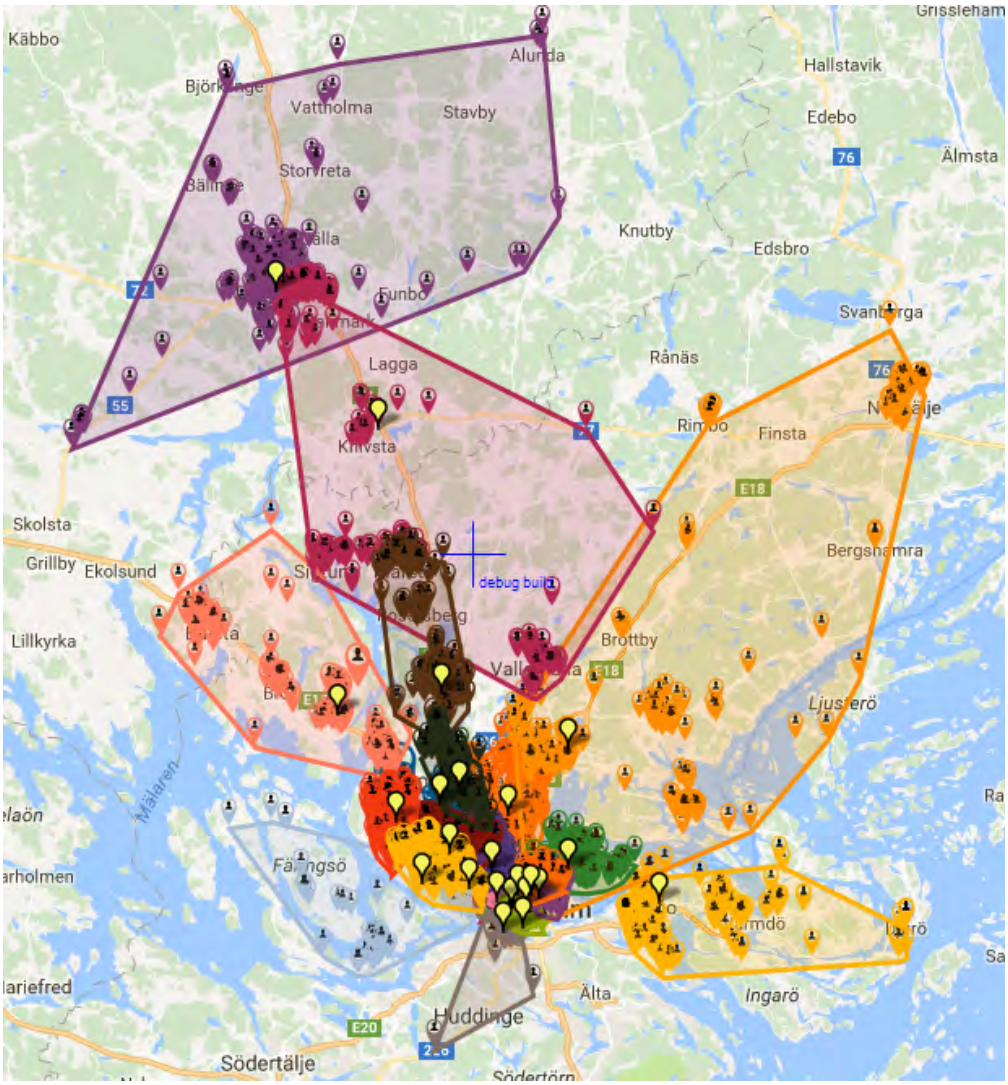


Figure A.22: Stockholm Results Map

A.8 Roskilde

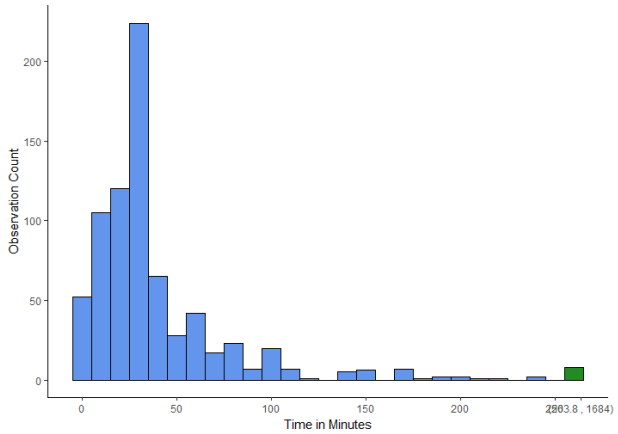


Figure A.23: Roskilde Histogram

Table A.16: Roskilde Statistics

Minimum Value	1
Maximum Value	1684
Mean (Standard Deviation)	44.88 (89.58)
Observation Count	746
Unique Observation Count	77
Mode (count)	28 (190)
2nd Mode (count)	24 (44)
3rd Mode (count)	4 (38)

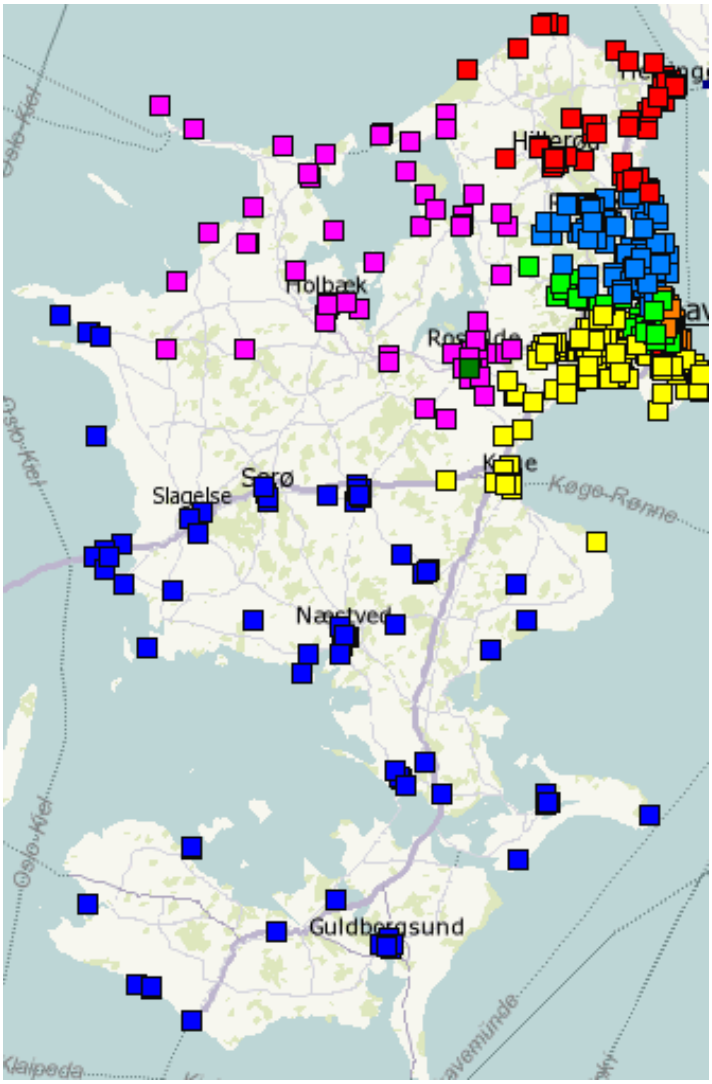


Figure A.24: Roskilde OTR

Table A.17: Roskilde Results KPIs

Mean Size (Working Time)	4782,86
Standard Deviation	119,27
Ratio Maximum to Expected	1,035
Ratio Minimum to Expected	0,965
Average Neighbourhood Ratio	0,7895
Average Silhouette Coefficient	0,0556
Davies-Bouldin Index	1,6282
Scaled Compactness	0,481
Scaled Centerpoint Distance	0,328
Running Time Total	00:00:06
Running Time Algorithm	00:00:05

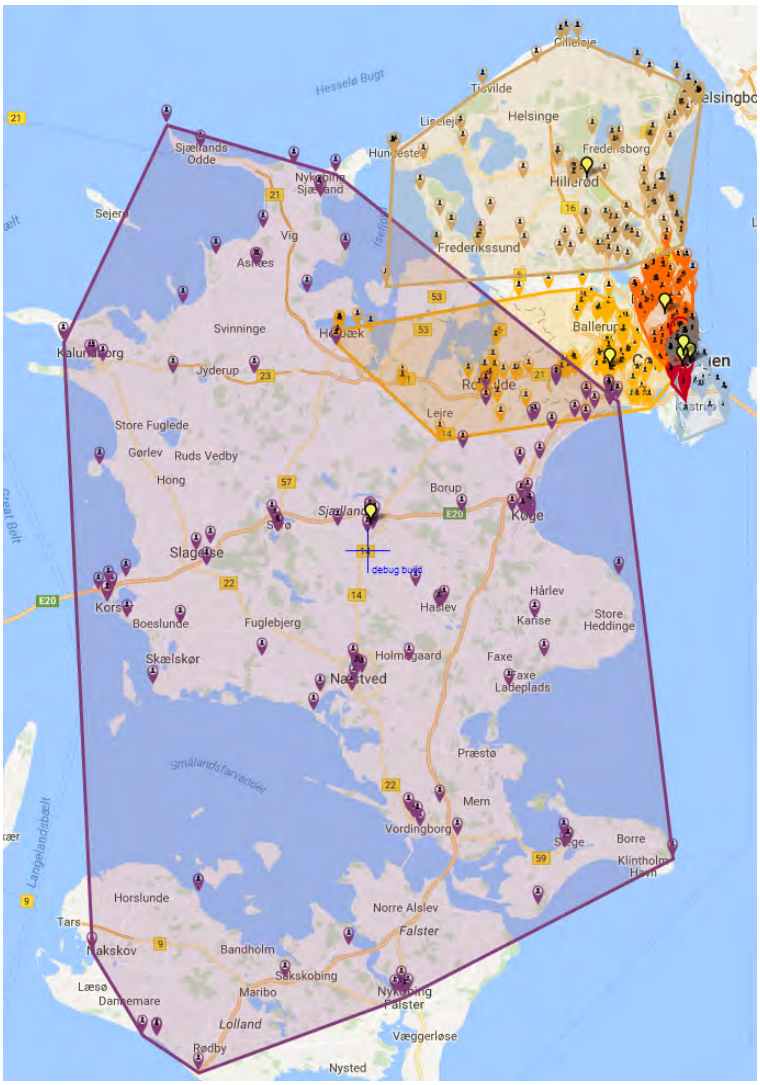


Figure A.25: Roskilde Results Map

A.9 Texas I

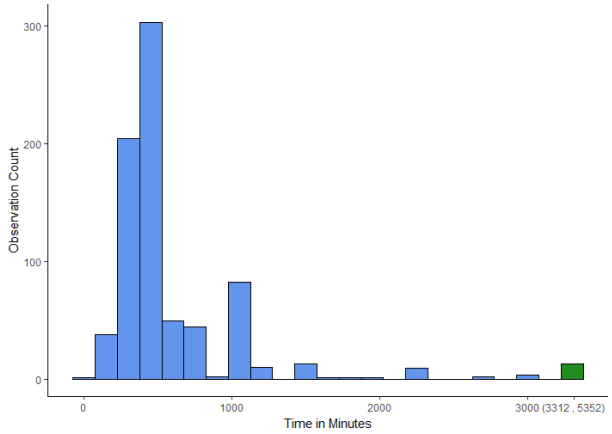


Figure A.26: Texas I Histogram

Table A.18: Texas I Statistics

Minimum Value	68
Maximum Value	5352
Mean (Standard Deviation)	612.3 (554.68)
Observation Count	777
Unique Observation Count	122
Mode (count)	504 (253)
2nd Mode (count)	252 (138)
3rd Mode (count)	1008 (57)

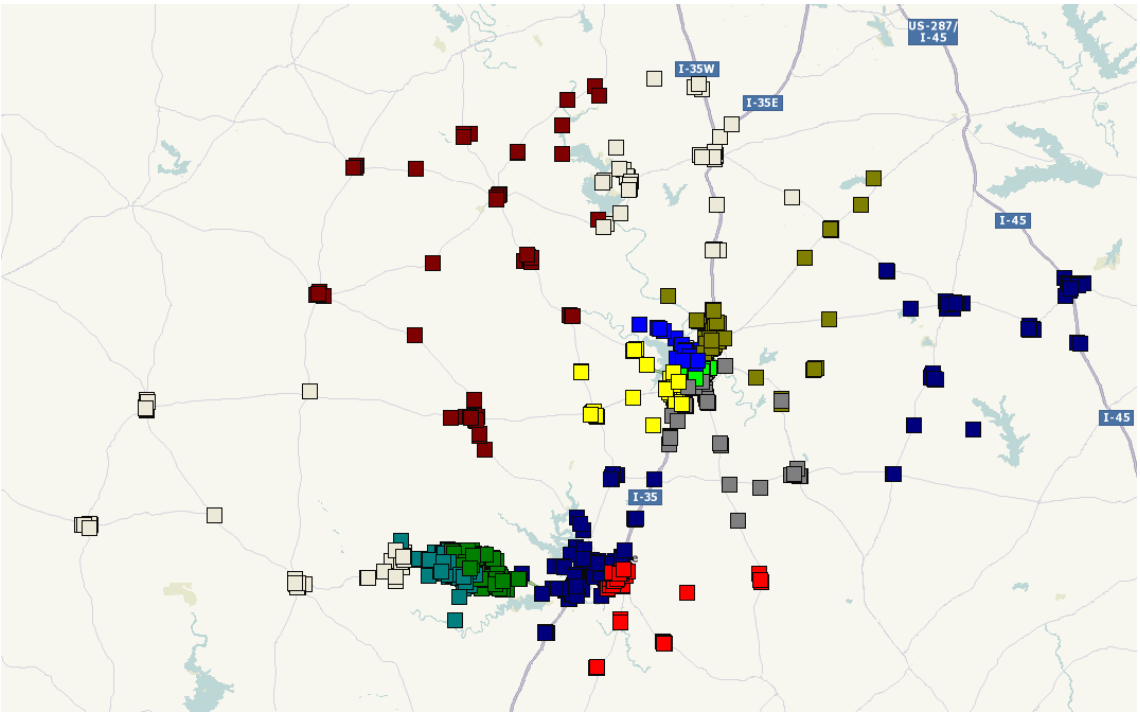


Figure A.27: Texas I OTR

Table A.19: Texas I Results KPIs

Mean Size (Working Time)	29732,88
Standard Deviation	190,22
Ratio Maximum to Expected	1,010
Ratio Minimum to Expected	0,988
Average Neighbourhood Ratio	0,8338
Average Silhouette Coefficient	0,1316
Davies-Bouldin Index	0,8744
Scaled Compactness	0,253
Scaled Centerpoint Distance	0,179
Running Time Total	00:00:03
Running Time Algorithm	00:00:02

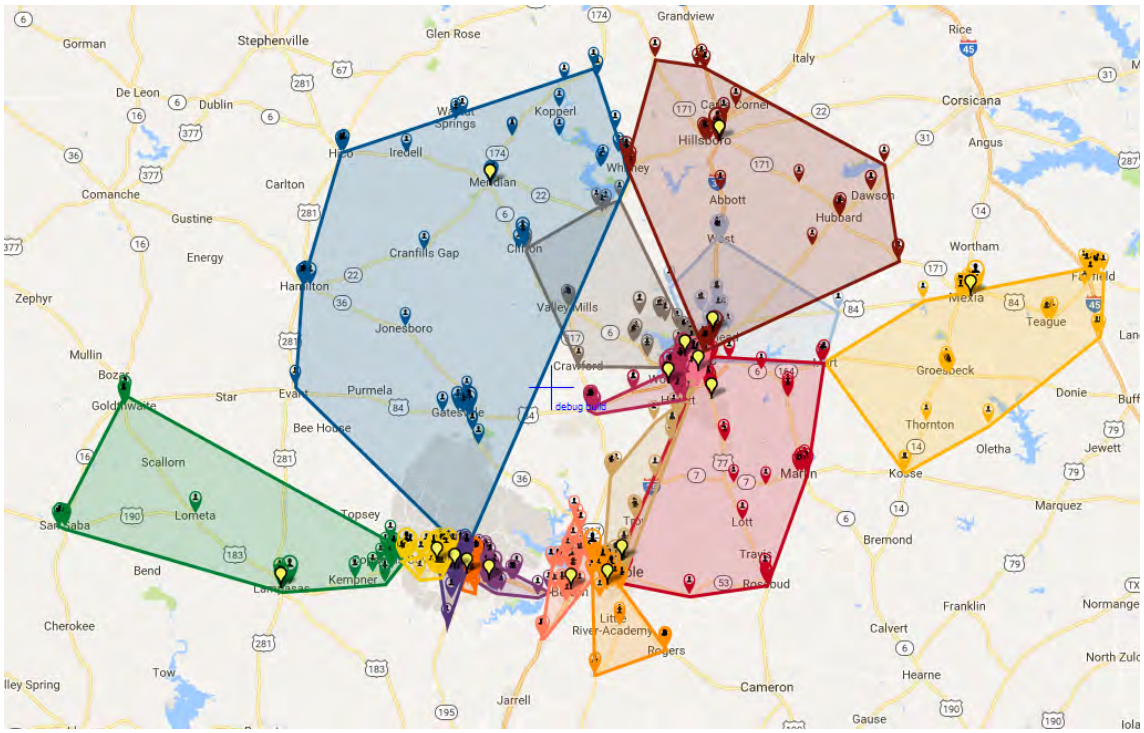


Figure A.28: Texas I Results Map

A.10 Texas II

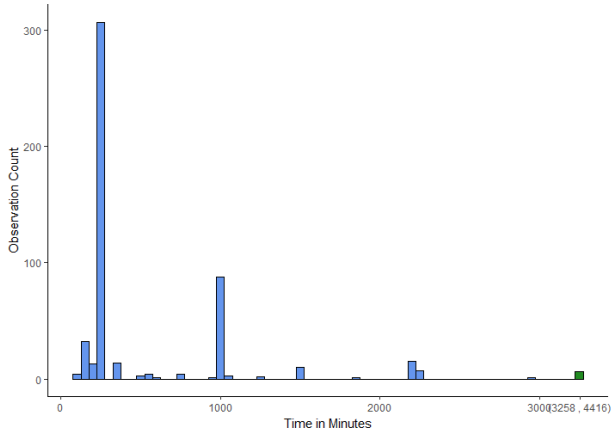


Figure A.29: Texas II Histogram

Table A.20: Texas II Statistics

Minimum Value	102
Maximum Value	4416
Mean (Standard Deviation)	548.3 (614.41)
Observation Count	516
Unique Observation Count	40
Mode (count) 2nd Mode (count) 3rd Mode (count)	252 (286) 1008 (85) 132 (31)

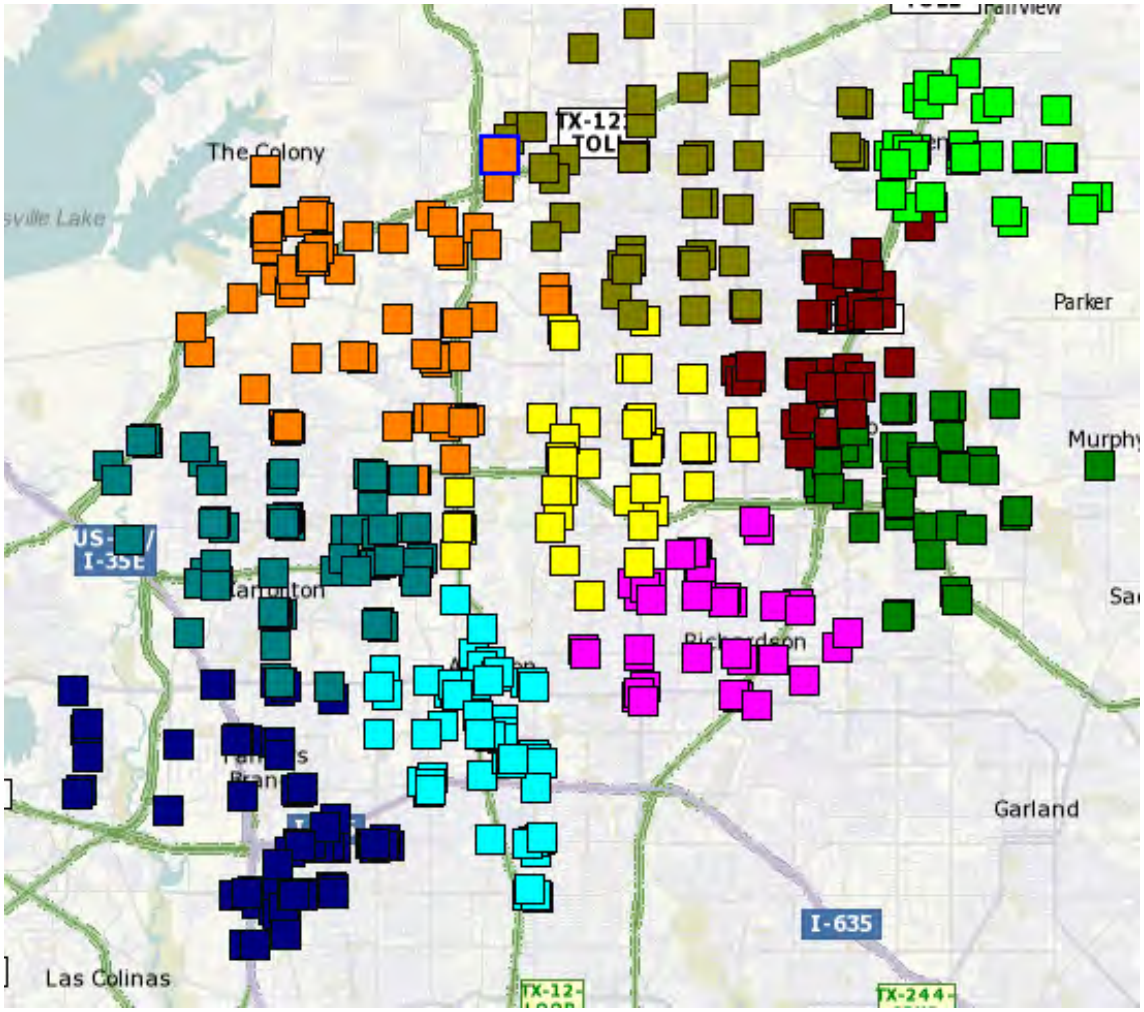


Figure A.30: Texas II OTR

Table A.21: Texas II Results KPIs

Mean Size (Working Time)	28273,20
Standard Deviation	549,60
Ratio Maximum to Expected	1,040
Ratio Minimum to Expected	0,972
Average Neighbourhood Ratio	0,8490
Average Silhouette Coefficient	0,3152
Davies-Bouldin Index	0,9090
Scaled Compactness	0,323
Scaled Centerpoint Distance	0,227
Running Time Total	00:00:00
Running Time Algorithm	00:00:00

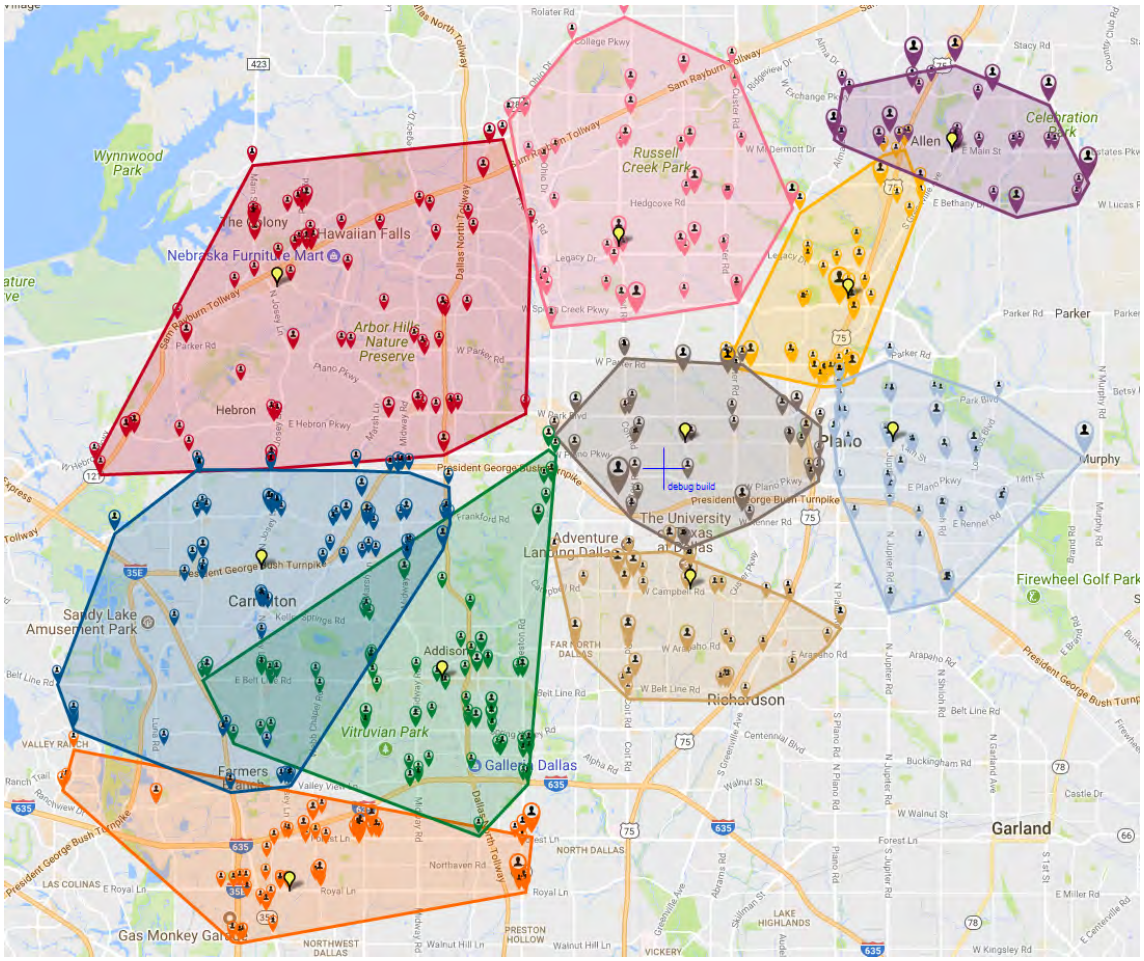


Figure A.31: Texas II Results Map

A.11 Netherlands

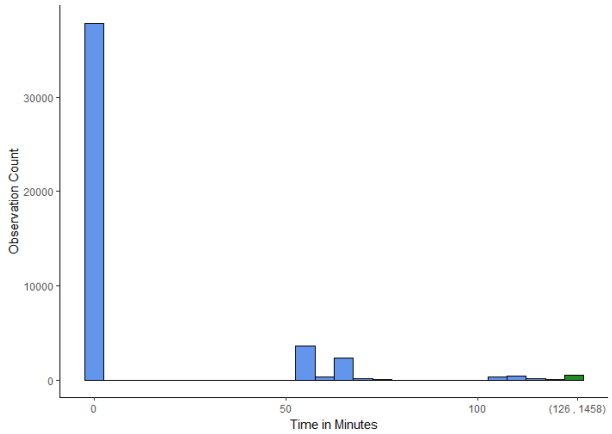


Figure A.32: Netherlands Histogram

Table A.22: Netherlands Statistics

Minimum Value	0
Maximum Value	1458
Mean (Standard Deviation)	12.77 (35.46)
Observation Count	45653
Unique Observation Count	91
Mode (count)	0 (37856)
2nd Mode (count)	56 (3366)
3rd Mode (count)	66 (2174)

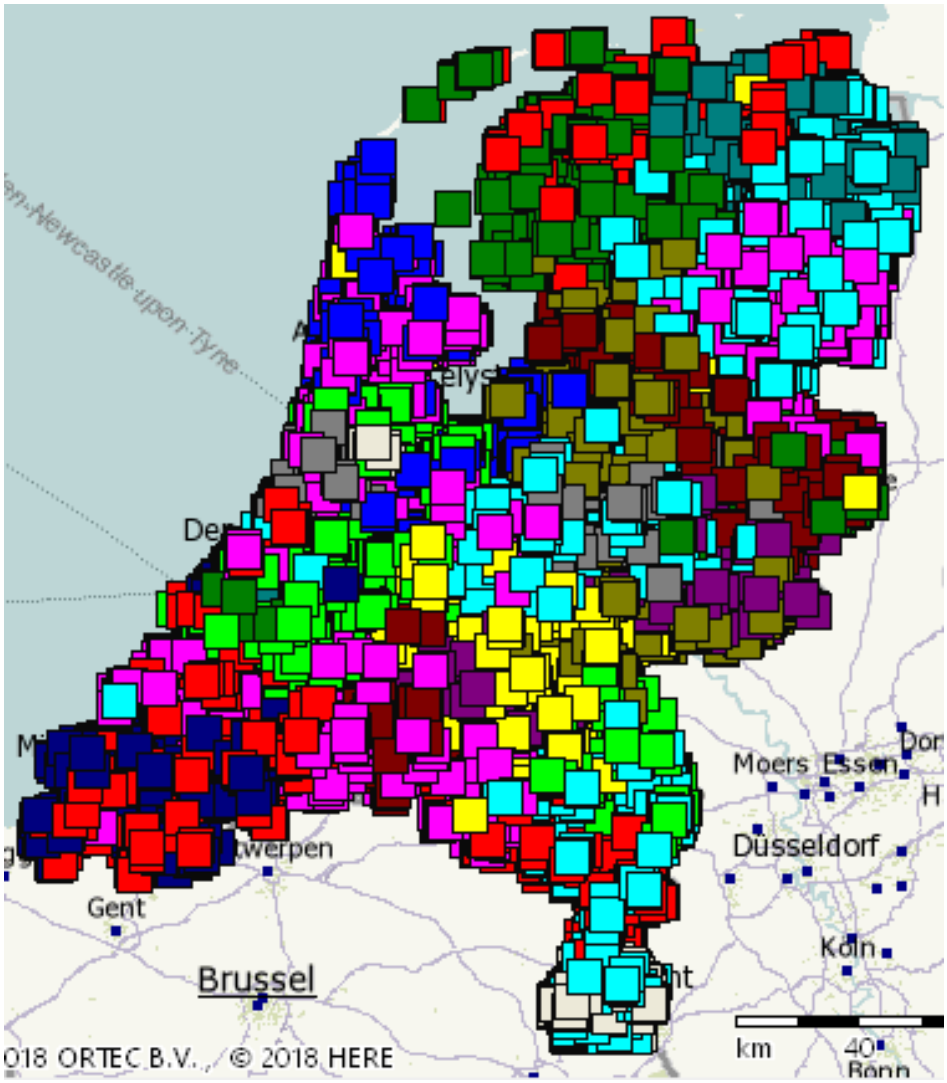


Figure A.33: Netherlands OTR

A.12 UK

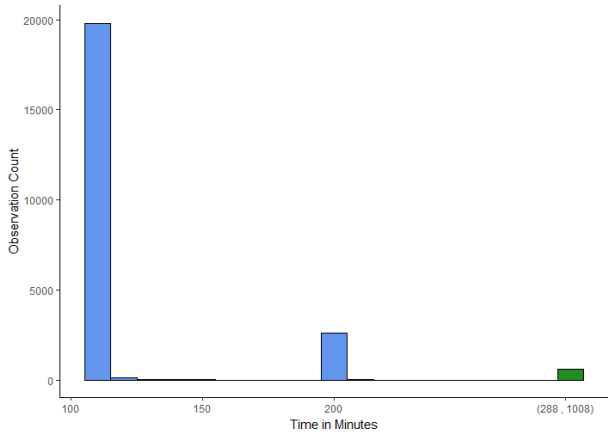


Figure A.34: UK Histogram

Table A.23: UK Statistics

Minimum Value	108
Maximum Value	1008
Mean (Standard Deviation)	123.5 (42.67)
Observation Count	23211
Unique Observation Count	70
Mode (count)	108 (19146)
2nd Mode common (count)	198 (2550)
3rd Mode (count)	288 (467)

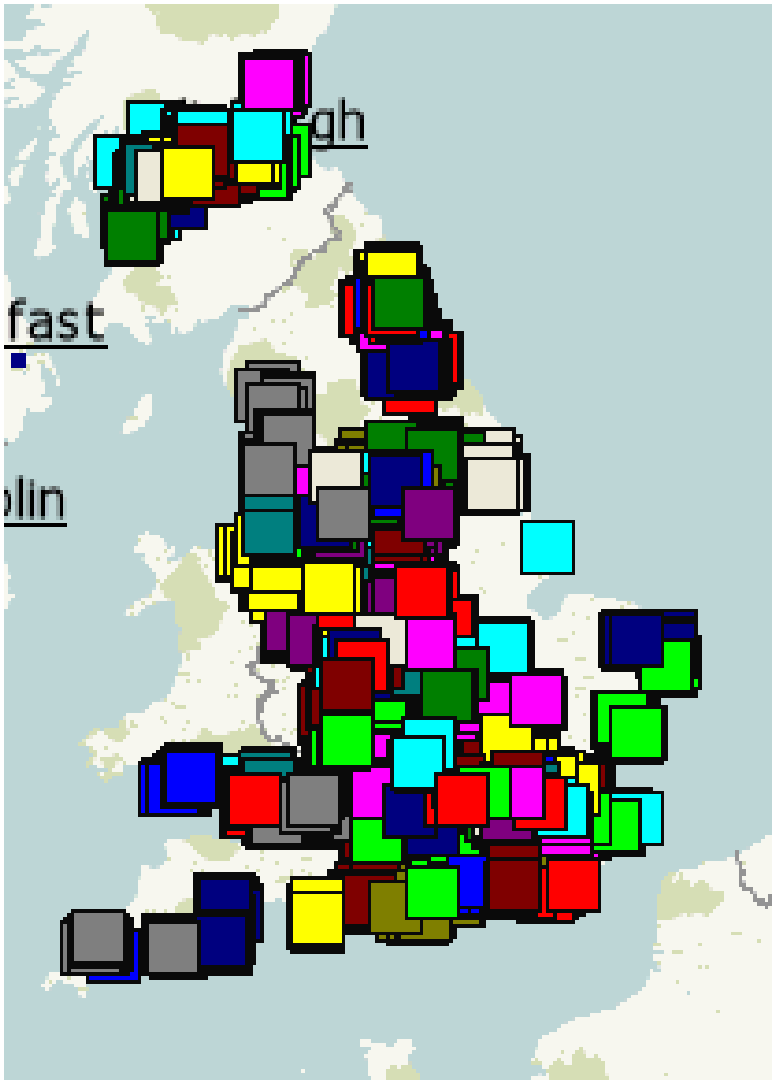


Figure A.35: UK OTR

A.13 Poland

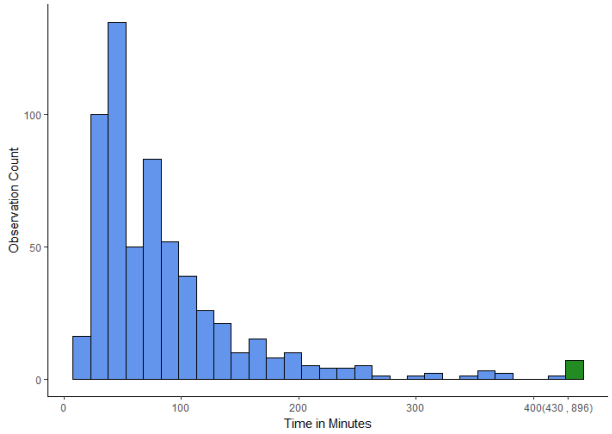


Figure A.36: Poland Histogram

Table A.24: Poland Statistics

Minimum Value	10
Maximum Value	896
Mean (Standard Deviation)	87.88 (86.00)
Observation Count	601
Unique Observation Count	189
Mode (count)	36 (21)
2nd most common (count)	38 (15)
3rd most common (count)	40 (15)

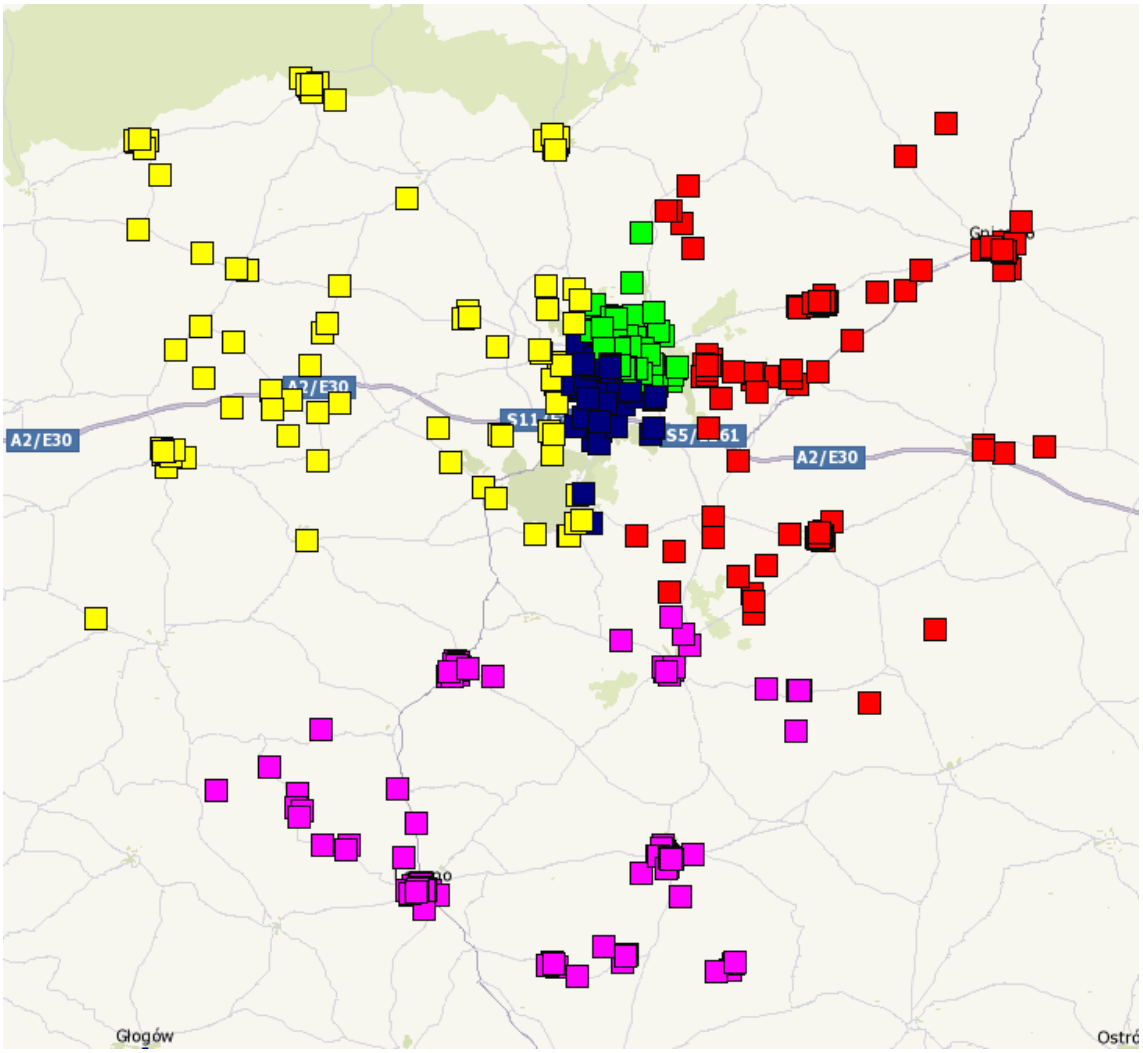


Figure A.37: Poland OTR

Table A.25: Poland Results KPIs

Mean Size (Working Time)	10563,20
Standard Deviation	283,73
Ratio Maximum to Expected	1,032
Ratio Minimum to Expected	0,975
Average Neighbourhood Ratio	0,8774
Average Silhouette Coefficient	0,0263
Davies-Bouldin Index	3,9267
Scaled Compactness	0,610
Scaled Centerpoint Distance	0,429
Running Time Total	00:00:00
Running Time Algorithm	00:00:00

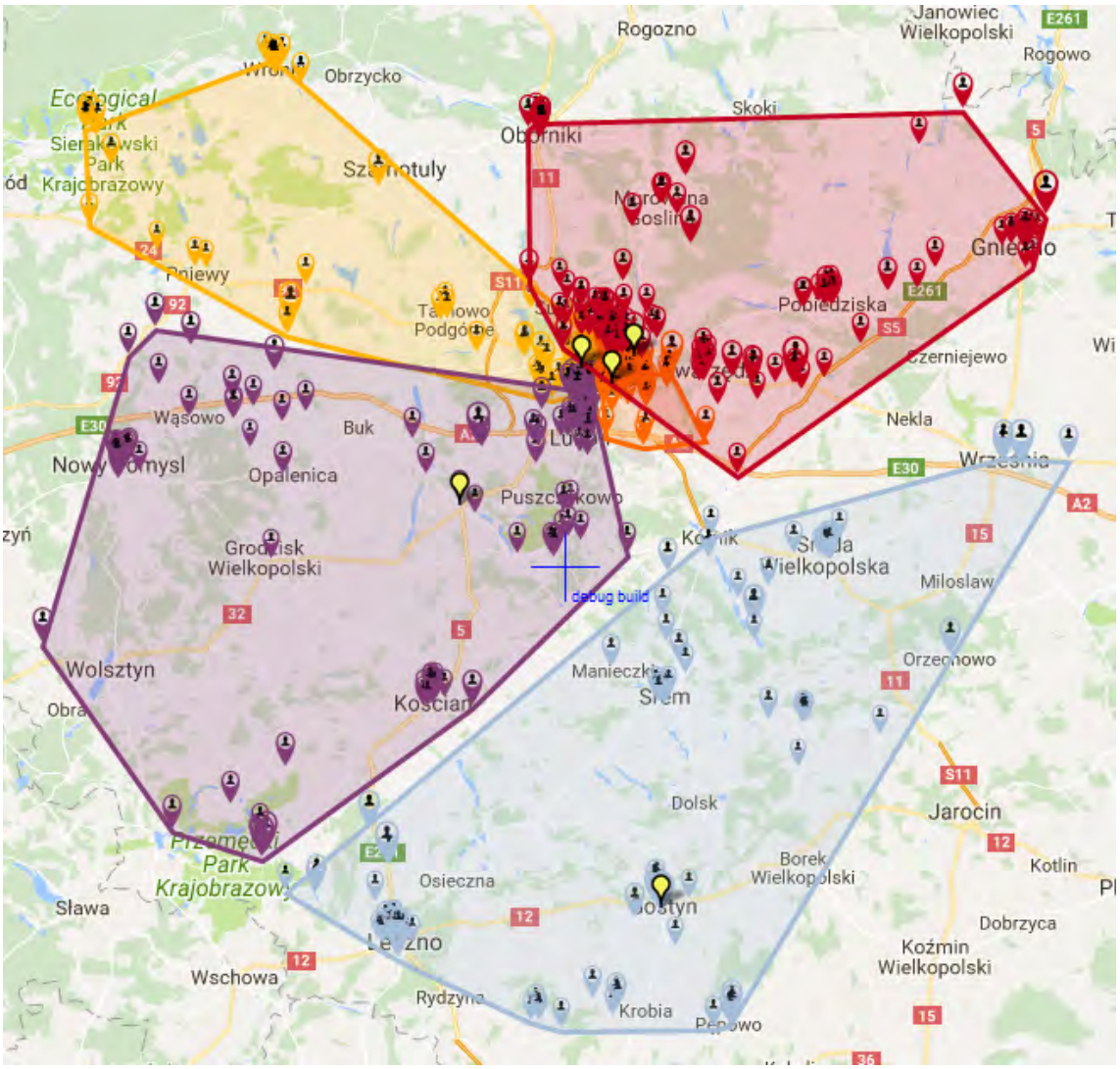


Figure A.38: Poland Results Map

A.14 St. Louis I

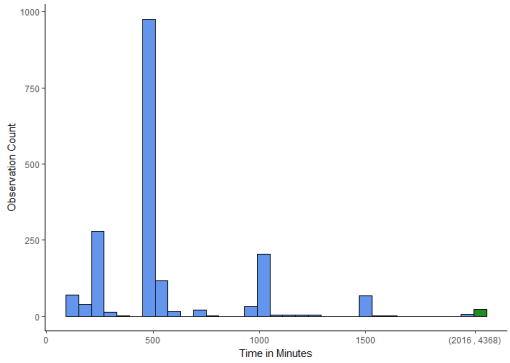


Figure A.39: St. Louis I Histogram

Table A.26: St. Louis I Statistics

Minimum Value	126
Maximum Value	4368
Mean (Standard Deviation)	580.8 (366.79)
Observation Count	1876
Unique Observation Count	104
Mode (count) 2nd Mode (count) 3rd Mode (count)	504 (936) 252 (232) 1008 (190)

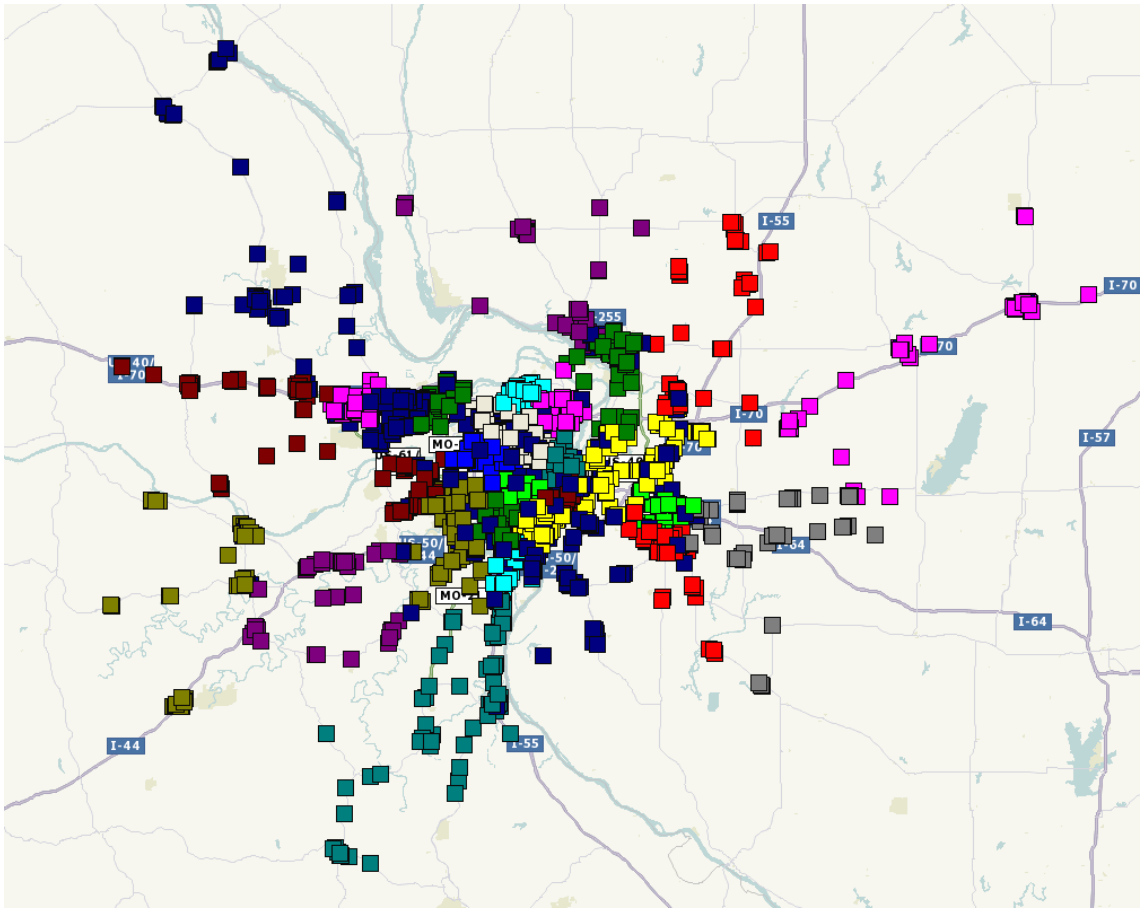


Figure A.40: St. Louis I OTR

Table A.27: St. Louis I Results KPIs

Mean Size (Working Time)	32986,55
Standard Deviation	821,28
Ratio Maximum to Expected	1,038
Ratio Minimum to Expected	0,962
Average Neighbourhood Ratio	0,8125
Average Silhouette Coefficient	0,1672
Davies-Bouldin Index	1,3036
Scaled Compactness	0,287
Scaled Centerpoint Distance	0,204
Running Time Total	00:00:16
Running Time Algorithm	00:00:13

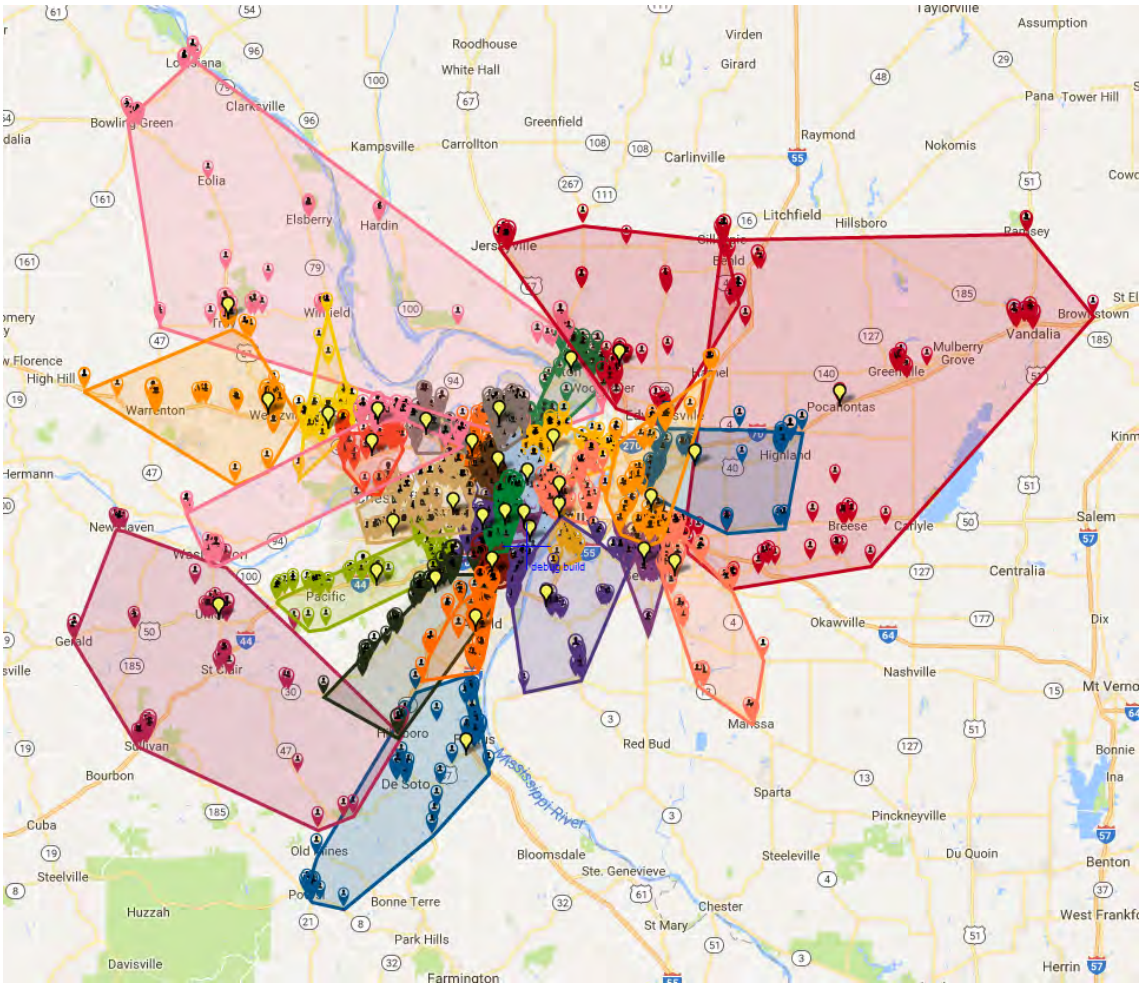


Figure A.41: St. Louis I Results Map

A.15 St. Louis II

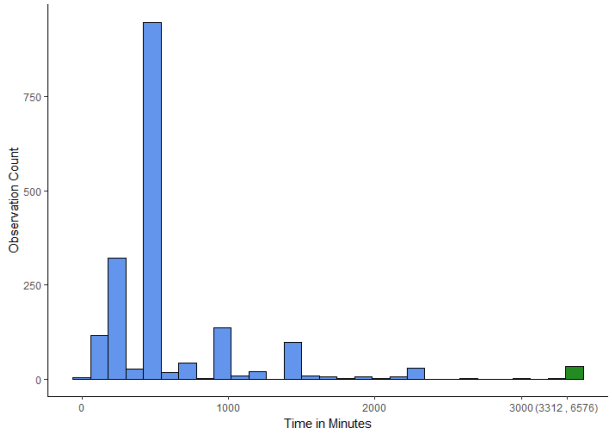


Figure A.42: St. Louis II Histogram

Table A.28: St. Louis II Statistics

Minimum Value	51
Maximum Value	6576
Mean (Standard Deviation)	654 (620.84)
Observation Count	1830
Unique Observation Count	116
Mode (count)	504 (897)
2nd Mode (count)	252 (222)
3rd Mode (count)	1488 (97)

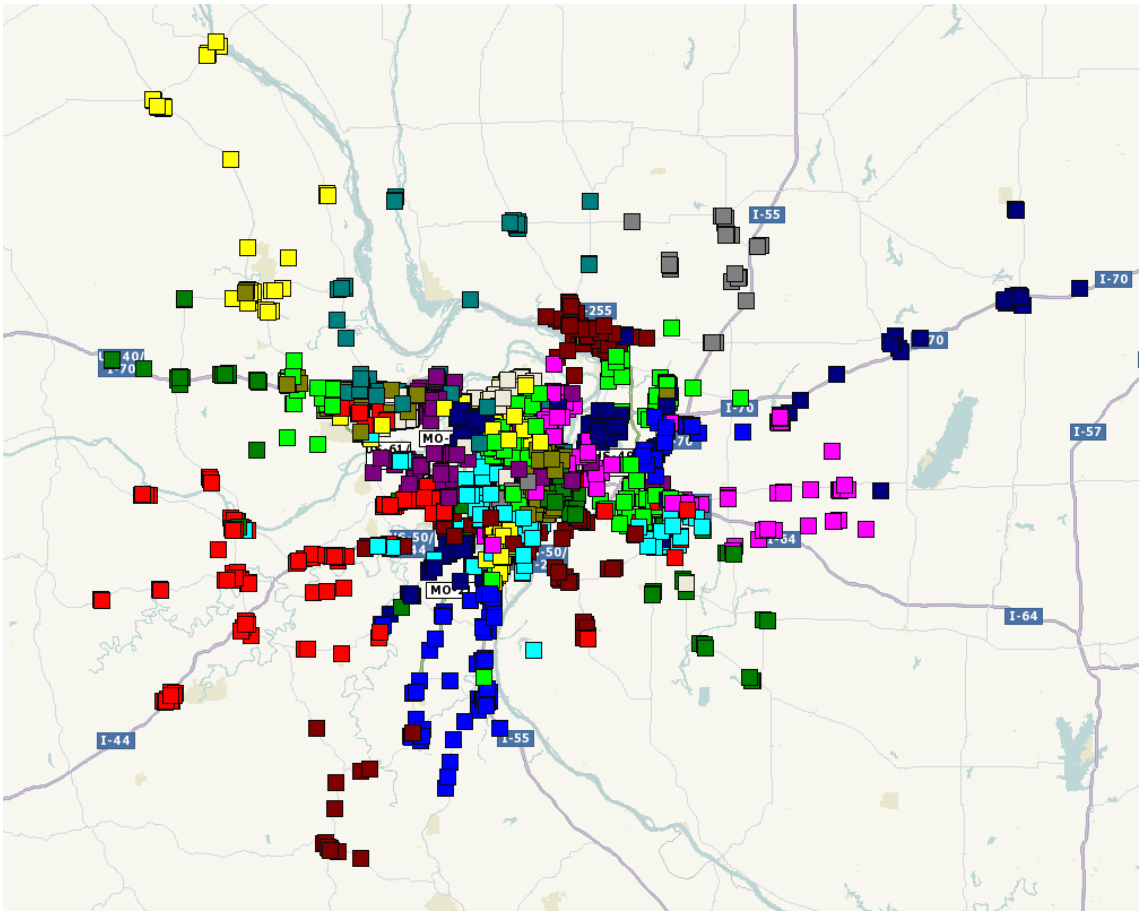


Figure A.43: St. Louis II OTR

Table A.29: St. Louis II Results KPIs

Mean Size (Working Time)	36268,85
Standard Deviation	831,58
Ratio Maximum to Expected	1,037
Ratio Minimum to Expected	0,962
Average Neighbourhood Ratio	0,7977
Average Silhouette Coefficient	0,1828
Davies-Bouldin Index	0,8630
Scaled Compactness	0,284
Scaled Centerpoint Distance	0,210
Running Time Total	00:00:20
Running Time Algorithm	00:00:16

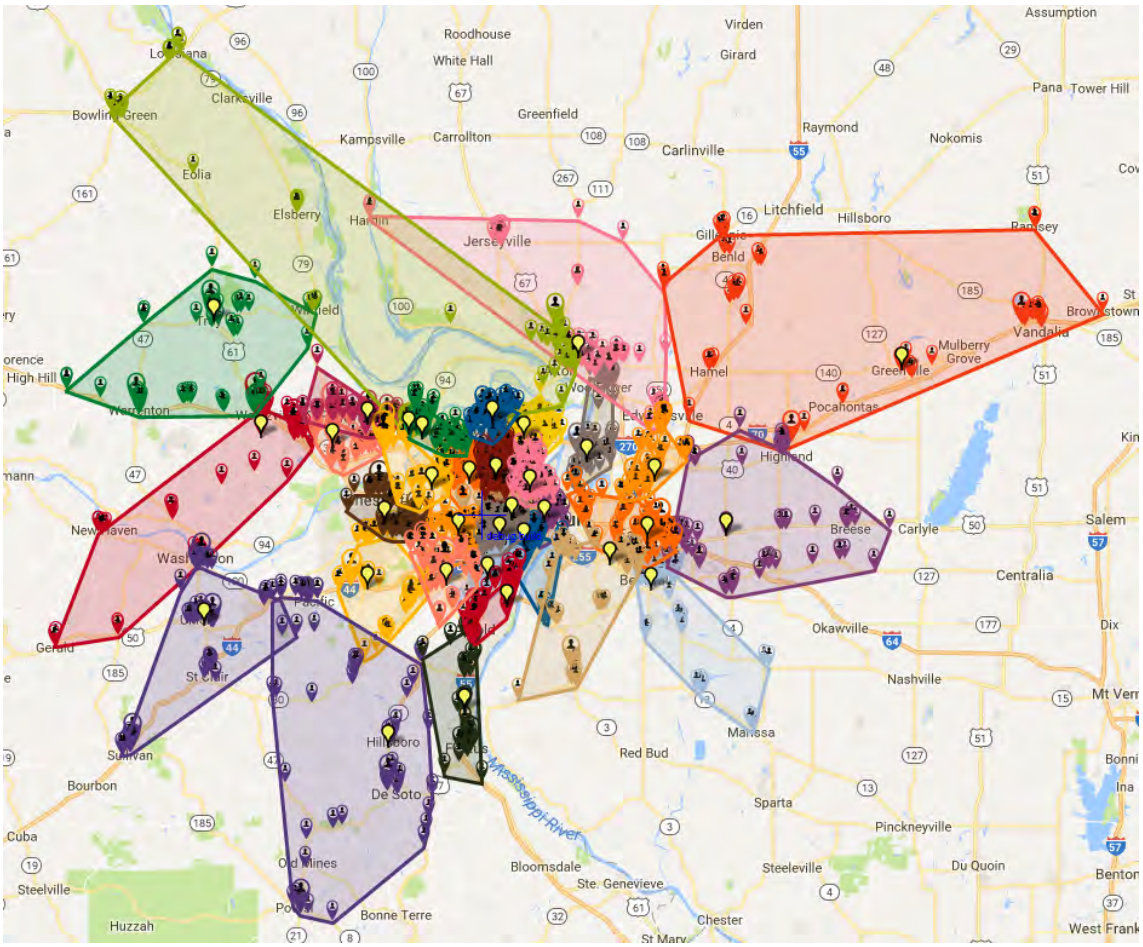


Figure A.44: St. Louis II Results Map

A.16 New York City

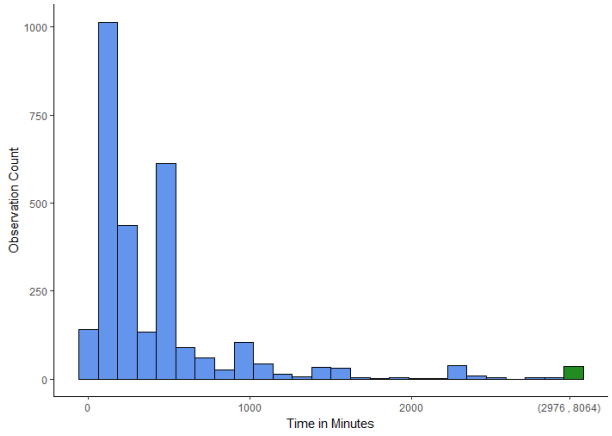


Figure A.45: New York City Histogram

Table A.30: New York City Statistics

Minimum Value	45
Maximum Value	8064
Mean (Standard Deviation)	450 (575.23)
Observation Count	2846
Unique Observation Count	199
Mode (count)	180 (460)
2nd Mode (count)	480 (402)
3rd Mode (count)	240 (331)

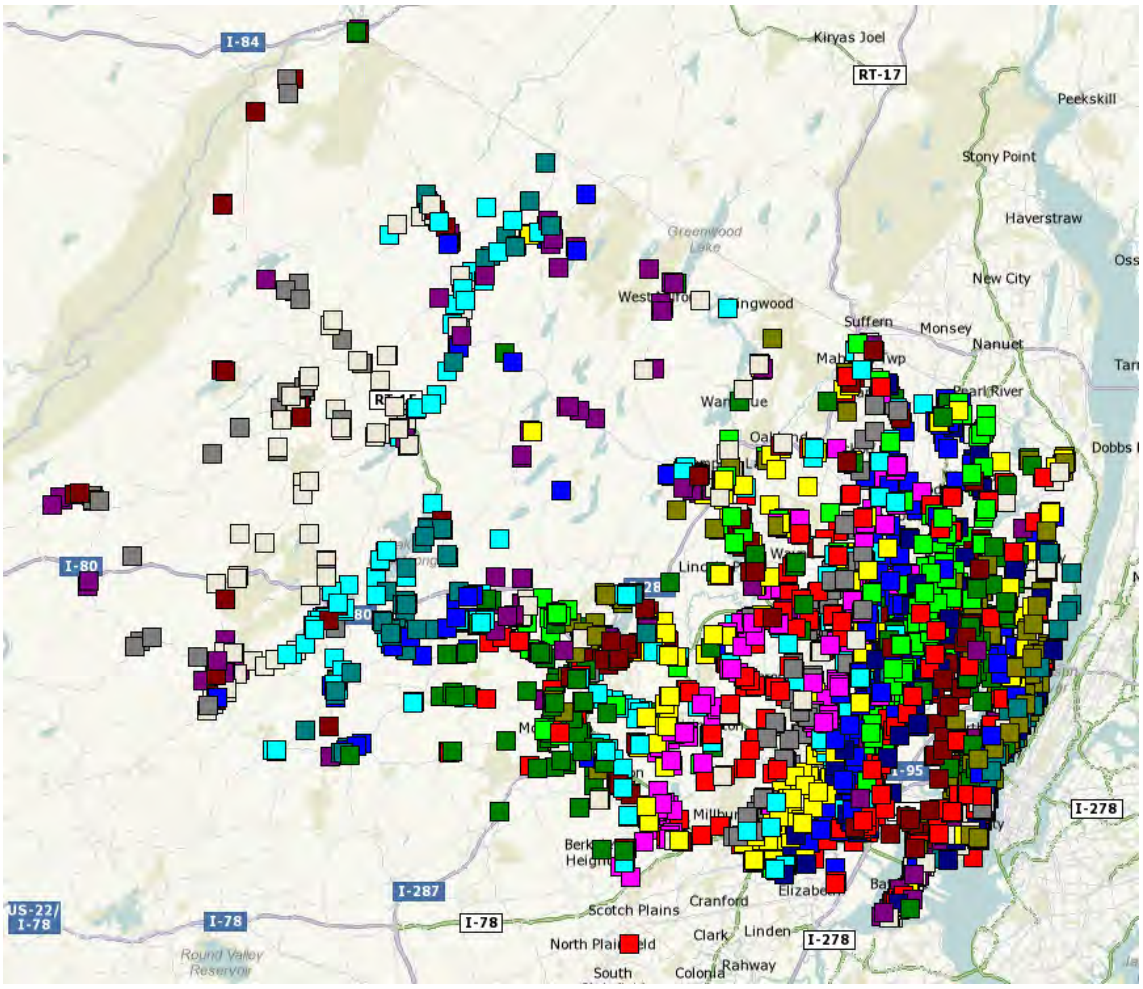


Figure A.46: New York OTR

Table A.31: New York City Results KPIs

Mean Size (Working Time)	28457,89
Standard Deviation	194,76
Ratio Maximum to Expected	1,015
Ratio Minimum to Expected	0,987
Average Neighbourhood Ratio	0,7945
Average Silhouette Coefficient	0,2278
Davies-Bouldin Index	1,3627
Scaled Compactness	0,172
Scaled Centerpoint Distance	0,121
Running Time Total	00:01:13
Running Time Algorithm	00:01:03

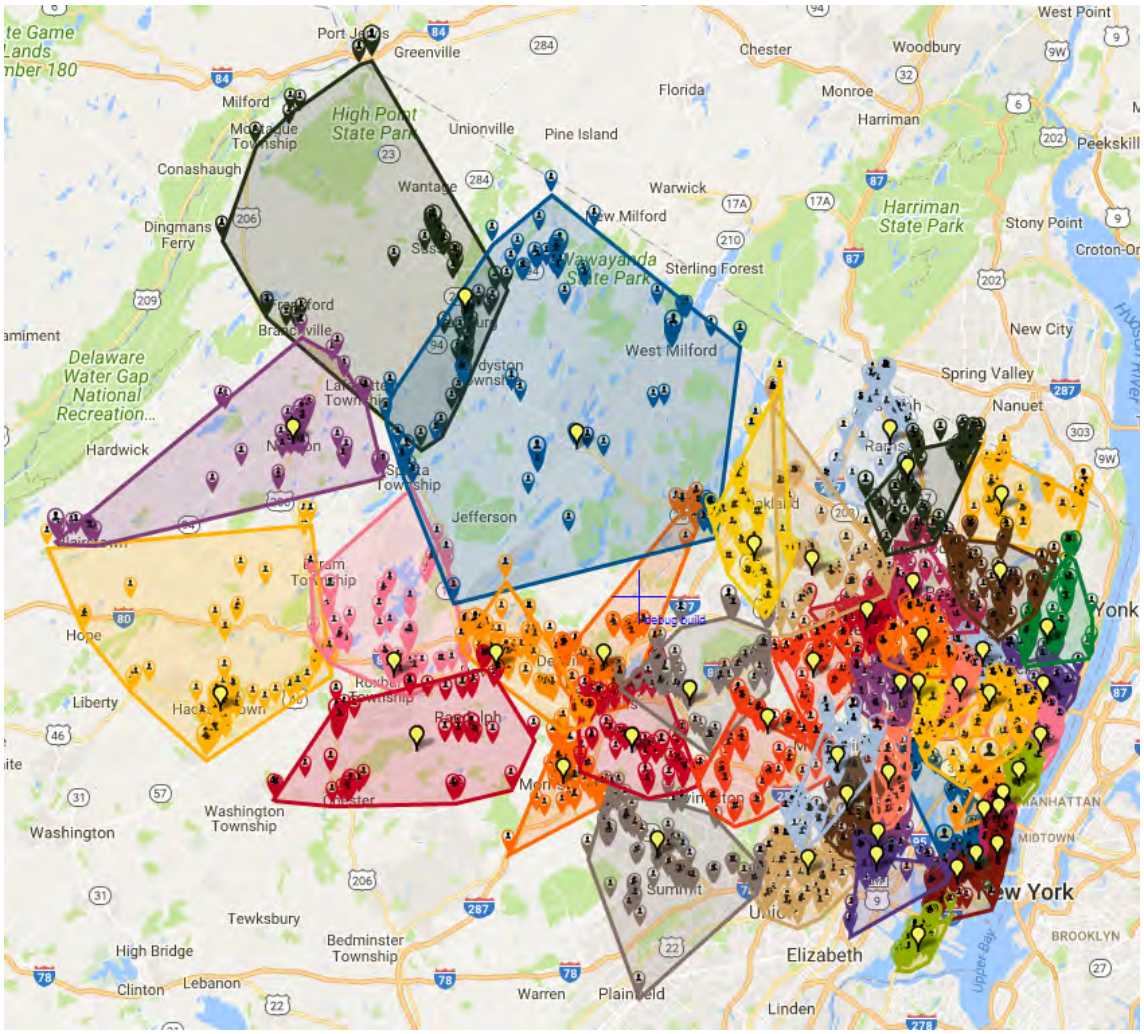


Figure A.47: New York City Results Map

A.17 Taiwan I

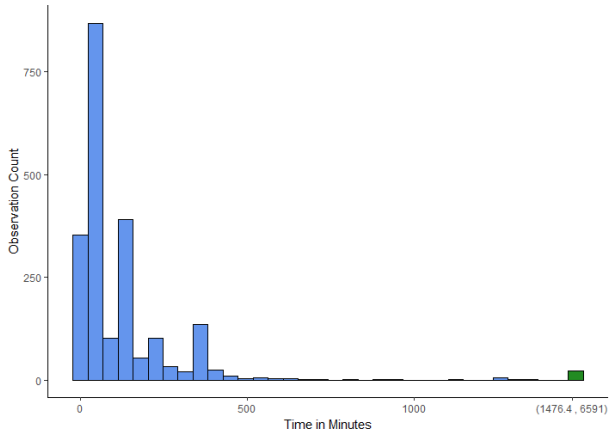


Figure A.48: Taiwan I Histogram

Table A.32: Taiwan I Statistics

Minimum Value	9
Maximum Value	6591
Mean (Standard Deviation)	140.9 (364.92)
Observation Count	2143
Unique Observation Count	145
Mode (count) 2nd Mode (count) 3rd Mode (count)	42 (653) 114 (296) 9 (206)

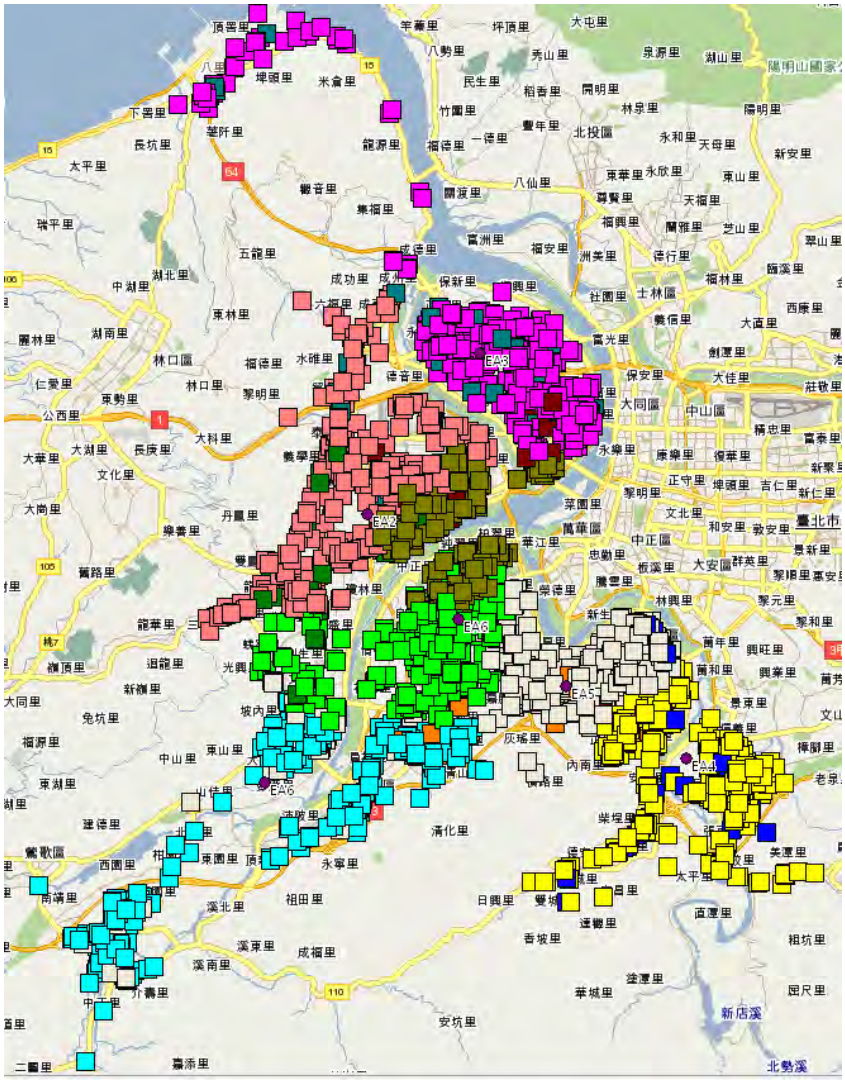


Figure A.49: Taiwan I OTR

Table A.33: Taiwan I Results KPIs

Mean Size (Working Time)	21561,29
Standard Deviation	539,33
Ratio Maximum to Expected	1,037
Ratio Minimum to Expected	0,971
Average Neighbourhood Ratio	0,8476
Average Silhouette Coefficient	0,2913
Davies-Bouldin Index	1,0267
Scaled Compactness	0,254
Scaled Centerpoint Distance	0,184
Running Time Total	00:00:23
Running Time Algorithm	00:00:18

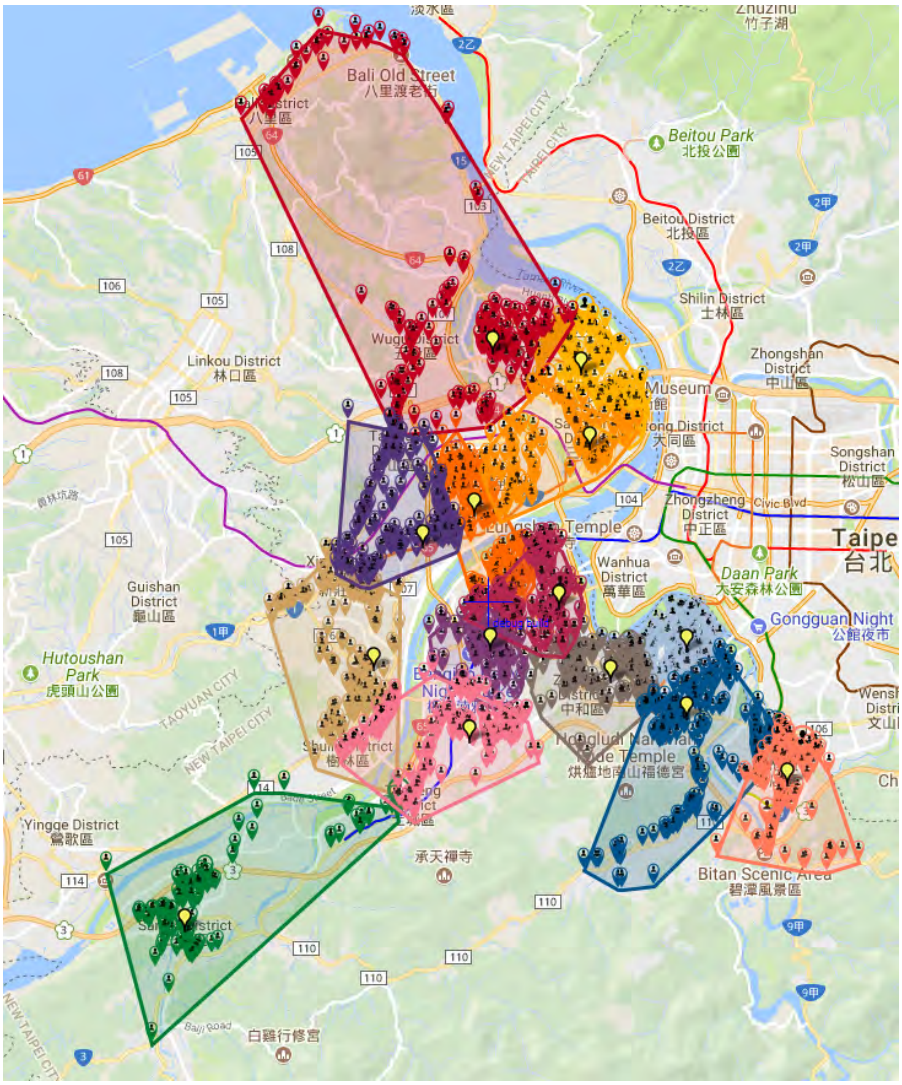


Figure A.50: Taiwan I Results Map

A.18 Taiwan II

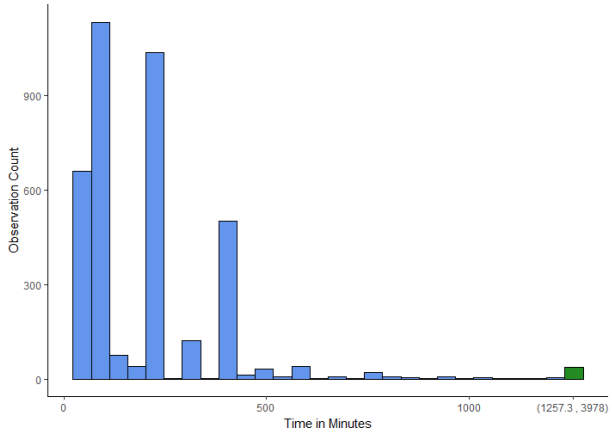


Figure A.51: Taiwan II Histogram

Table A.34: Taiwan II Statistics

Minimum Value	34
Maximum Value	3987
Mean (Standard Deviation)	210.4 (252.11)
Observation Count	3770
Unique Observation Count	119
Mode (count) 2nd Mode (count) 3rd Mode (count)	102 (1115) 204 (981) 34 (644)

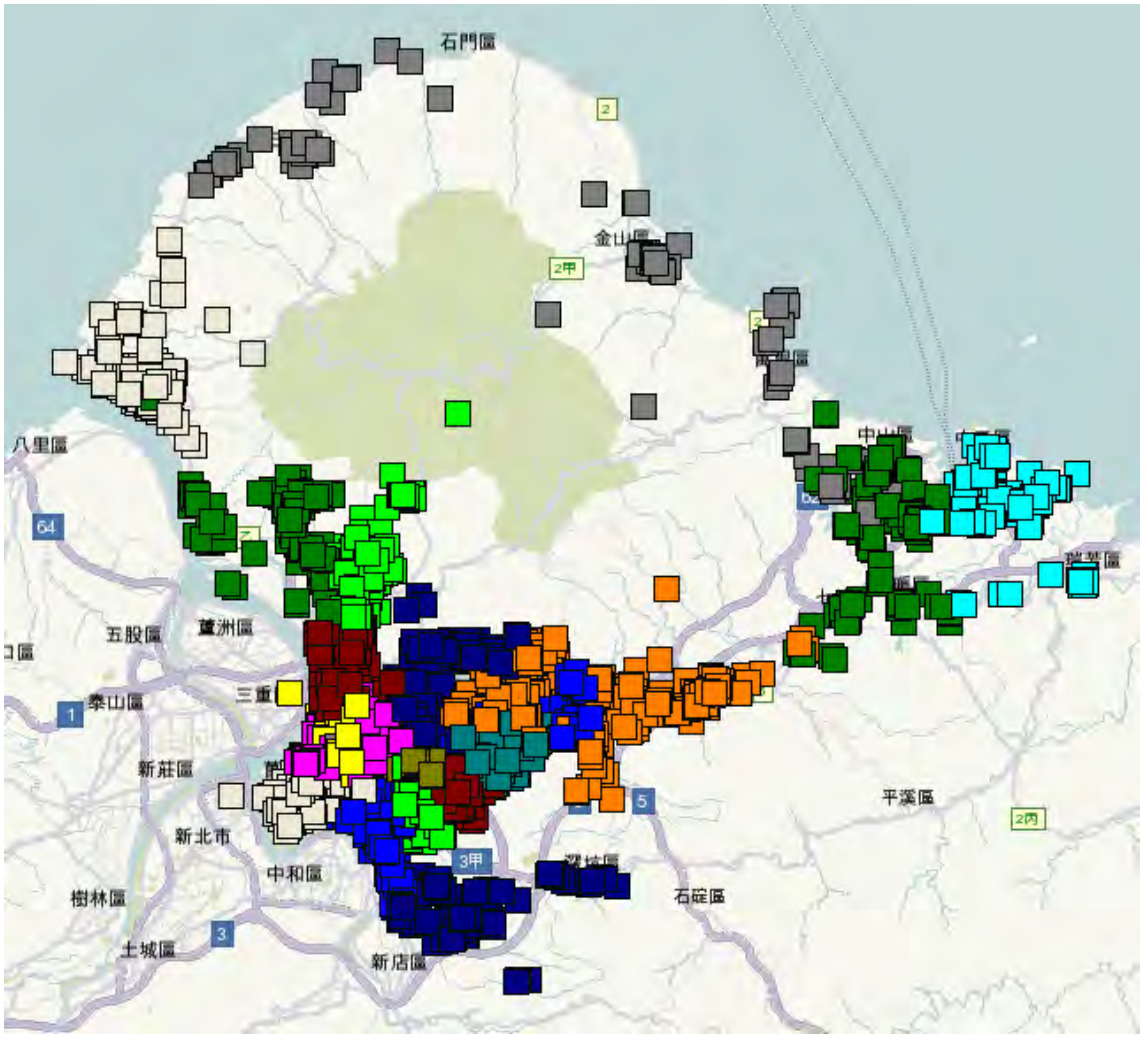


Figure A.52: Taiwan II OTR

Table A.35: Taiwan II Results KPIs

Mean Size (Working Time)	36054,64
Standard Deviation	421,98
Ratio Maximum to Expected	1,029
Ratio Minimum to Expected	0,973
Average Neighbourhood Ratio	0,8017
Average Silhouette Coefficient	0,1811
Davies-Bouldin Index	1,6042
Scaled Compactness	0,267
Scaled Centerpoint Distance	0,200
Running Time Total	00:01:16
Running Time Algorithm	00:01:00

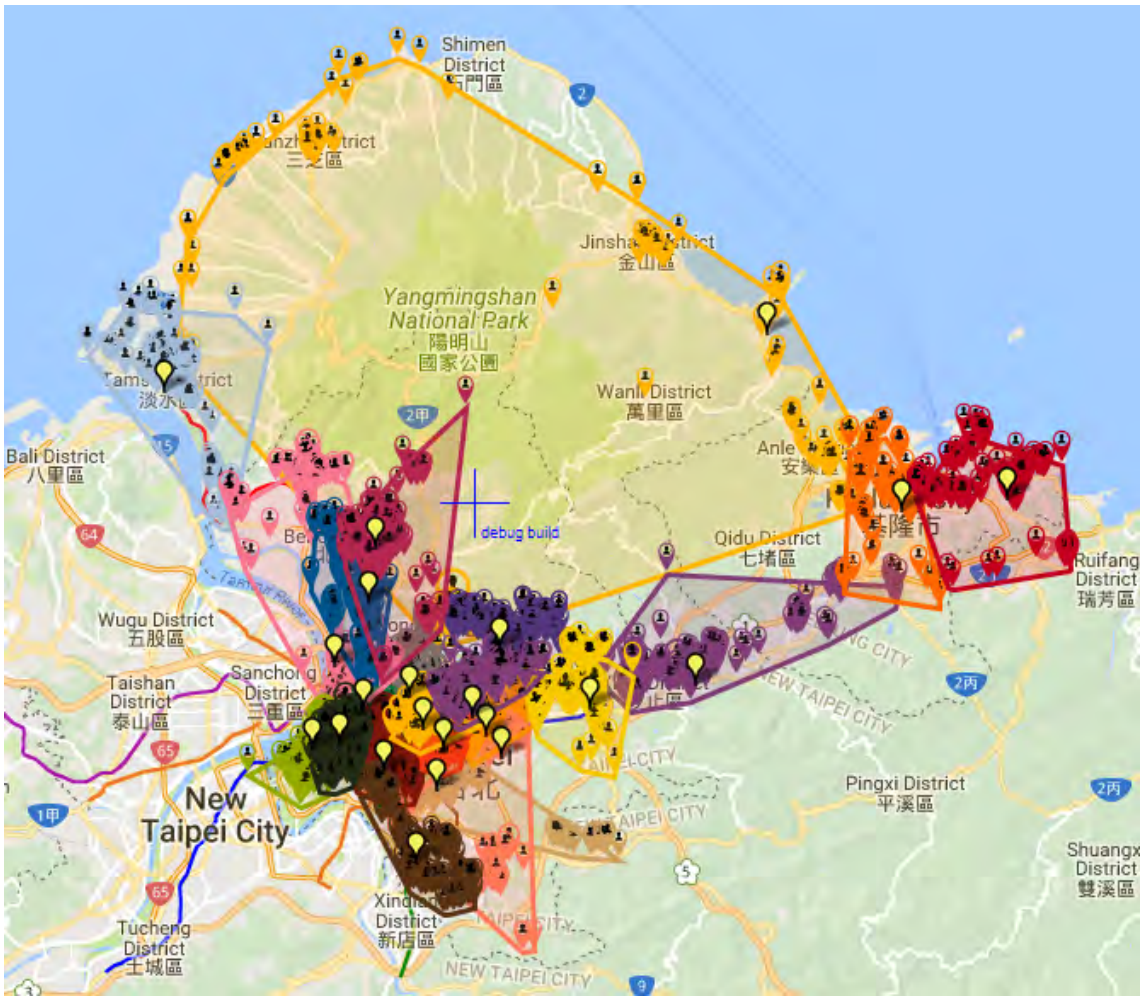


Figure A.53: Taiwan II Results Map

A.19 Brazil

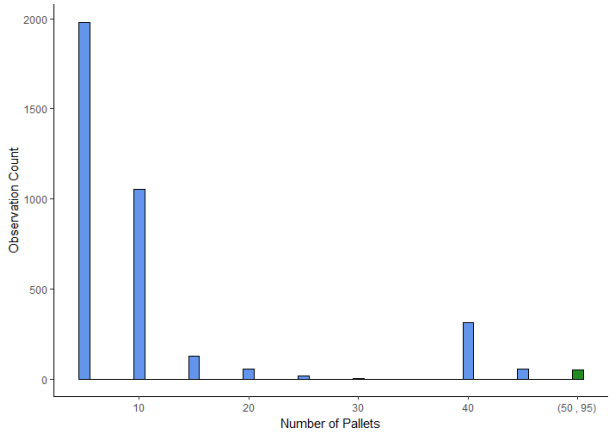


Figure A.54: Brazil Histogram

Table A.36: Brazil Statistics

Minimum Value	5
Maximum Value	95
Mean (Standard Deviation)	11.48 (12.06)
Observation Count	3659
Unique Observation Count	17
Mode (count) 2nd Mode (count) 3rd Mode (count)	5 (1981) 10 (1055) 40 (314)

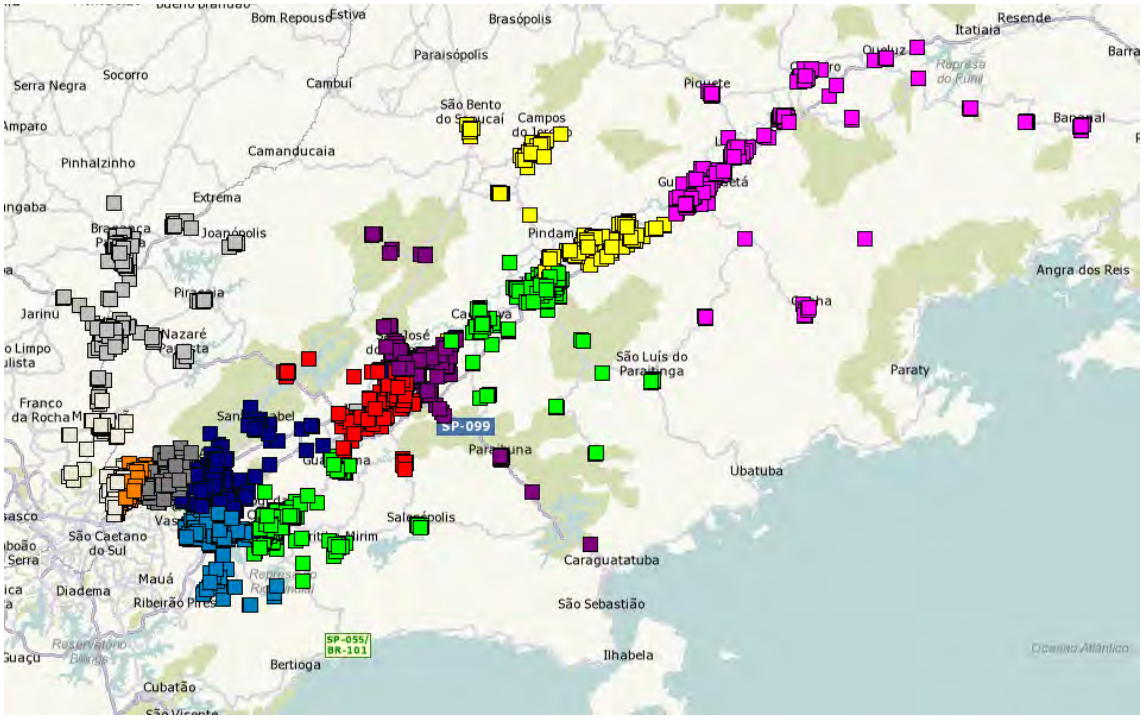


Figure A.55: Brazil OTR

Table A.37: Brazil Results KPIs

Mean Size (Pallets)	3499,17
Standard Deviation	220,98
Ratio Maximum to Expected	1,075
Ratio Minimum to Expected	0,922
Average Neighbourhood Ratio	0,8751
Average Silhouette Coefficient	0,2407
Davies-Bouldin Index	0,9740
Scaled Compactness	0,255
Scaled Centerpoint Distance	0,179
Running Time Total	00:00:43
Running Time Algorithm	00:00:28

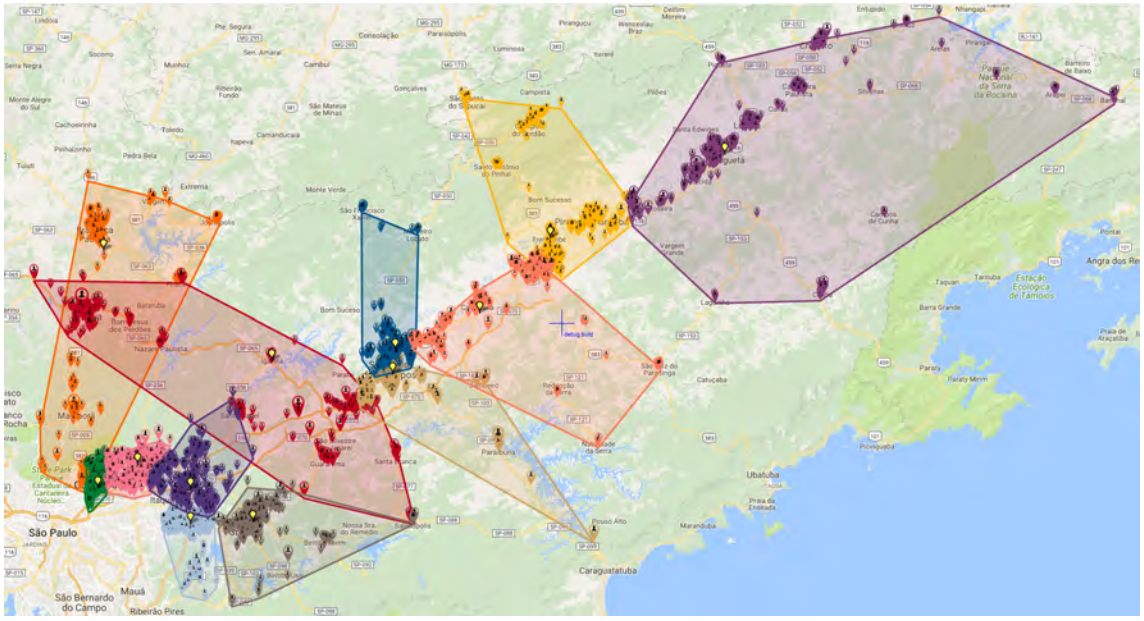


Figure A.56: Brazil Results Map

A.20 Belgium

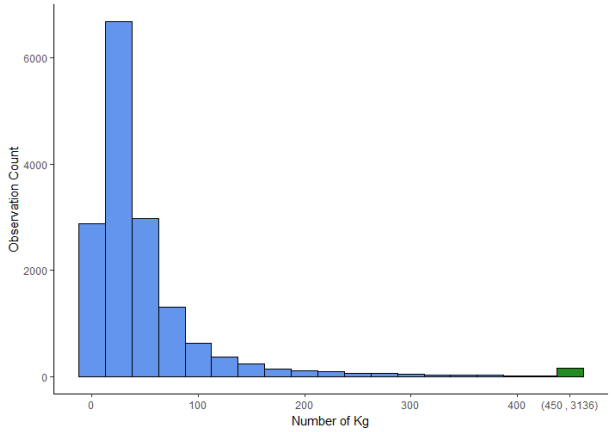


Figure A.57: Belgium Histogram

Table A.38: Belgium Statistics

Minimum Value	0
Maximum Value	3136
Mean (Standard Deviation)	55.3 (117.0)
Observation Count	15831
Unique Observation Count	454
Mode (count)	30 (1229)
2nd Mode (count)	24 (1088)
3rd Mode (count)	36 (997)

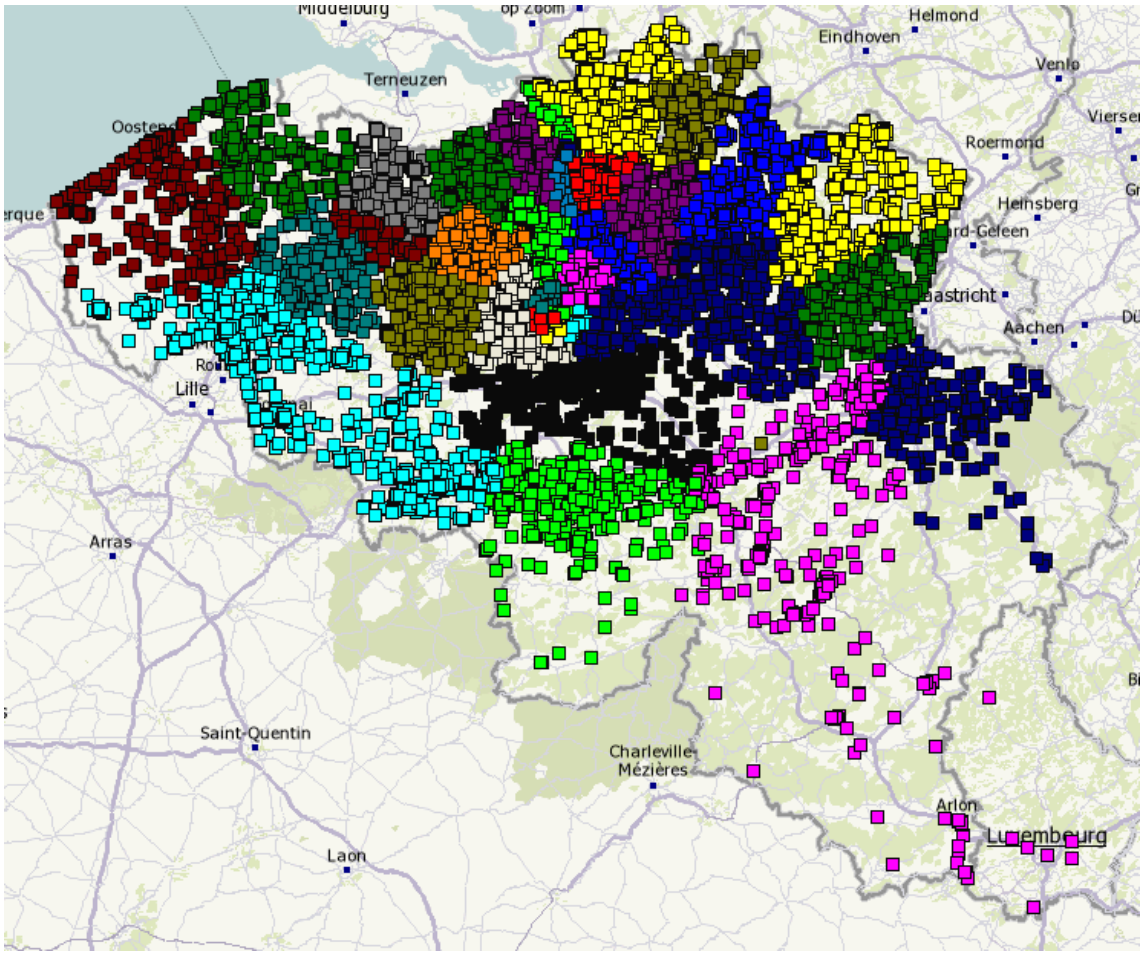


Figure A.58: Belgium OTR

Table A.39: Belgium Results KPIs

Mean Size (Kg)	22445,90
Standard Deviation	1270,69
Ratio Maximum to Expected	1,079
Ratio Minimum to Expected	0,928
Average Neighbourhood Ratio	0,7755
Average Silhouette Coefficient	0,1649
Davies-Bouldin Index	0,6719
Scaled Compactness	0,166
Scaled Centerpoint Distance	0,121
Running Time Total	00:13:03
Running Time Algorithm	00:10:40

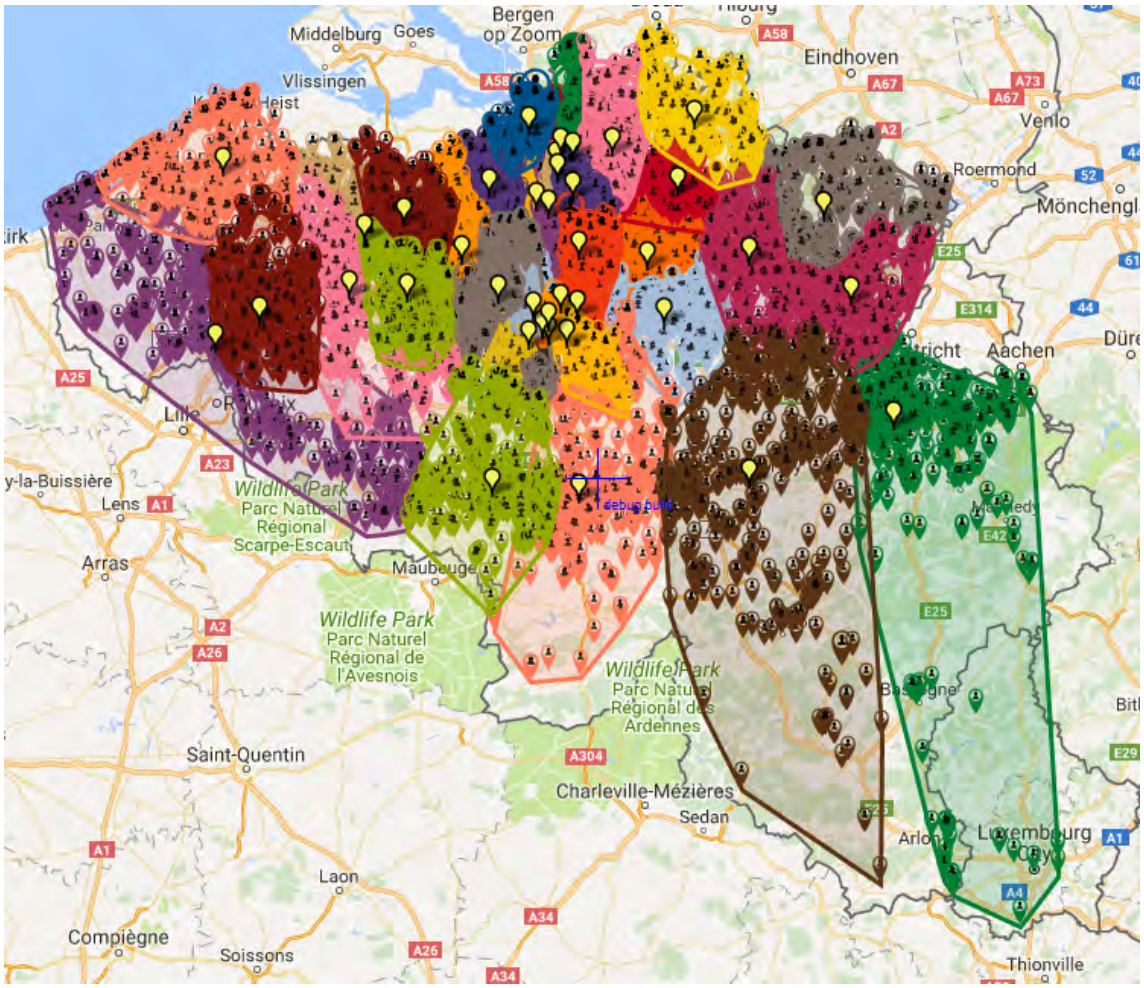


Figure A.59: Belgium Results Map

A.21 USA I

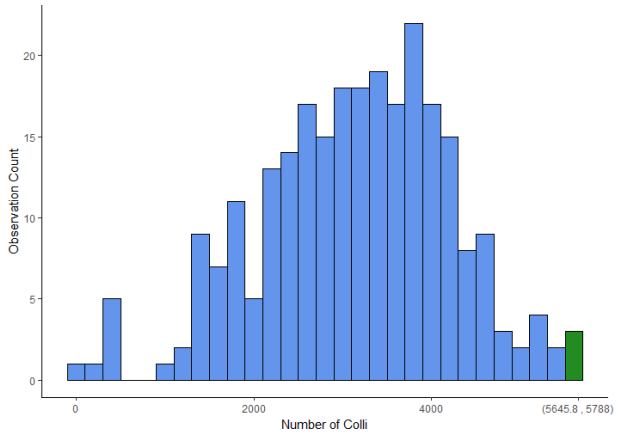


Figure A.60: USA I Histogram

Table A.40: USA I Statistics

Minimum Value	97
Maximum Value	5788
Mean (Standard Deviation)	3155 (1104.70)
Observation Count	260
Unique Observation Count	252
Mode (count)	2437 (2)
2nd Mode (count)	2991 (2)
3rd Mode (count)	3022 (2)

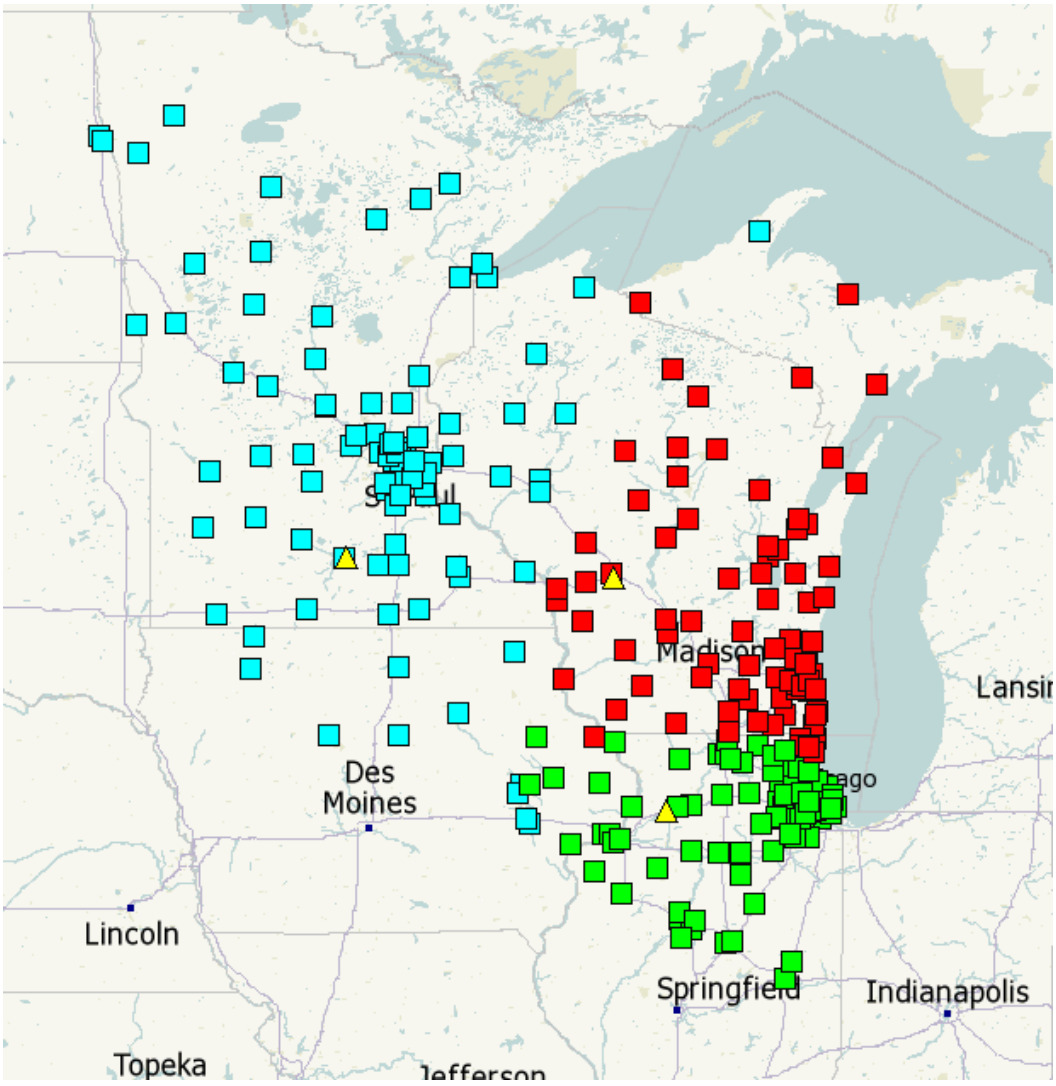


Figure A.61: USA I OTR

Table A.41: USA I Results KPIs

Mean Size (Colli)	273456,67
Standard Deviation	1252,22
Ratio Maximum to Expected	1,004
Ratio Minimum to Expected	0,995
Average Neighbourhood Ratio	0,9091
Average Silhouette Coefficient	0,3813
Davies-Bouldin Index	1,0354
Scaled Compactness	0,515
Scaled Centerpoint Distance	0,349
Running Time Total	00:00:00
Running Time Algorithm	00:00:00

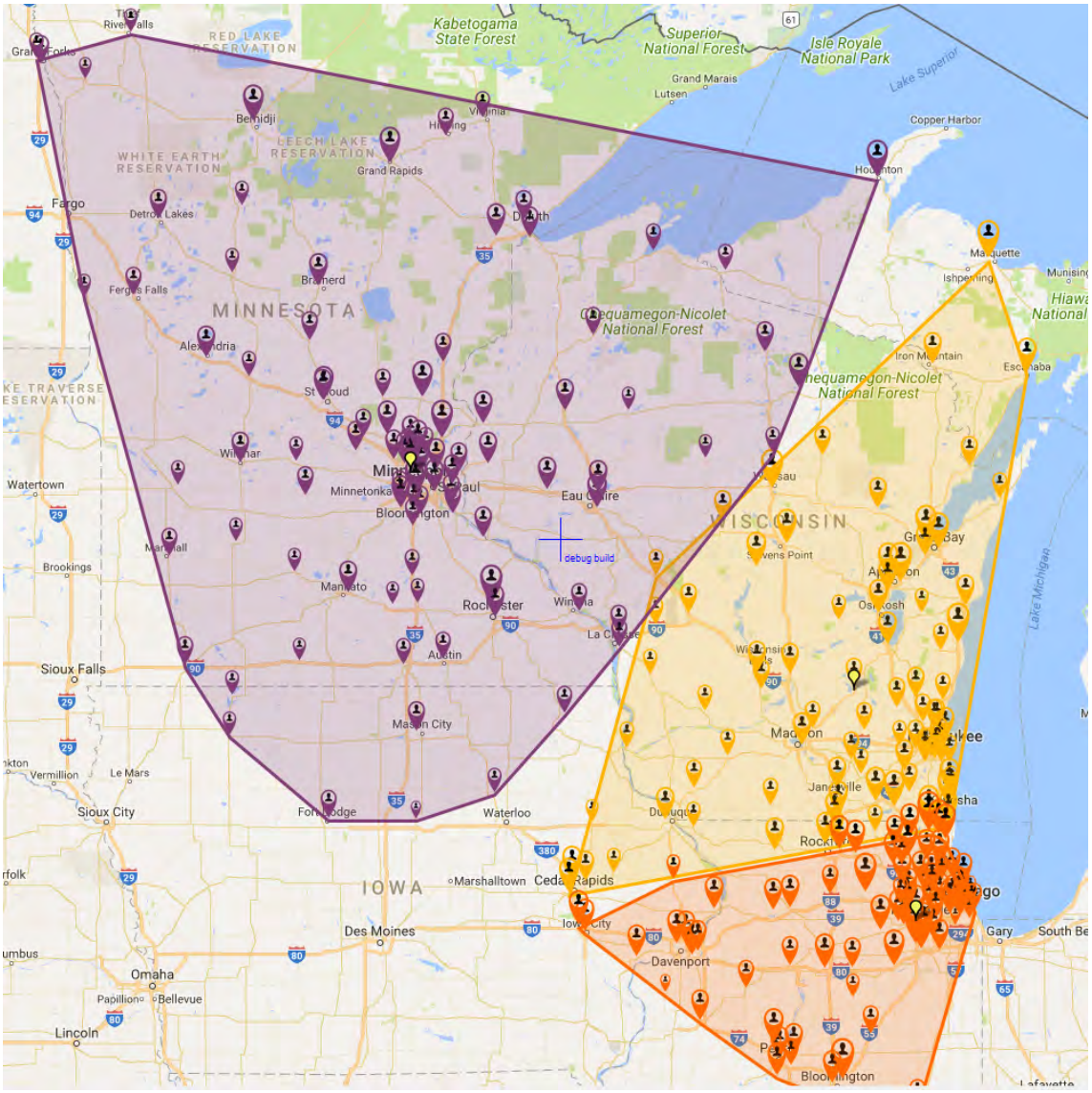


Figure A.62: USA I Results Map

A.22 USA II

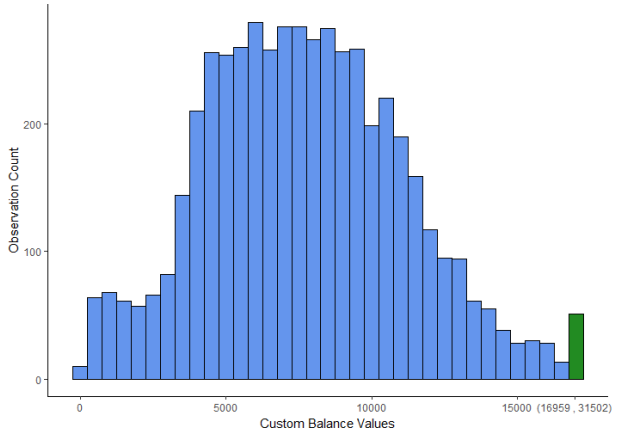


Figure A.63: USA II Histogram

Table A.42: USA II Statistics

Minimum Value	6
Maximum Value	31502
Mean (Standard Deviation)	7591 (3510.70)
Observation Count	5060
Unique Observation Count	4198
Mode (count)	9334 (6)
2nd most common (count)	4699 (5)
3rd most common (count)	4922 (4)

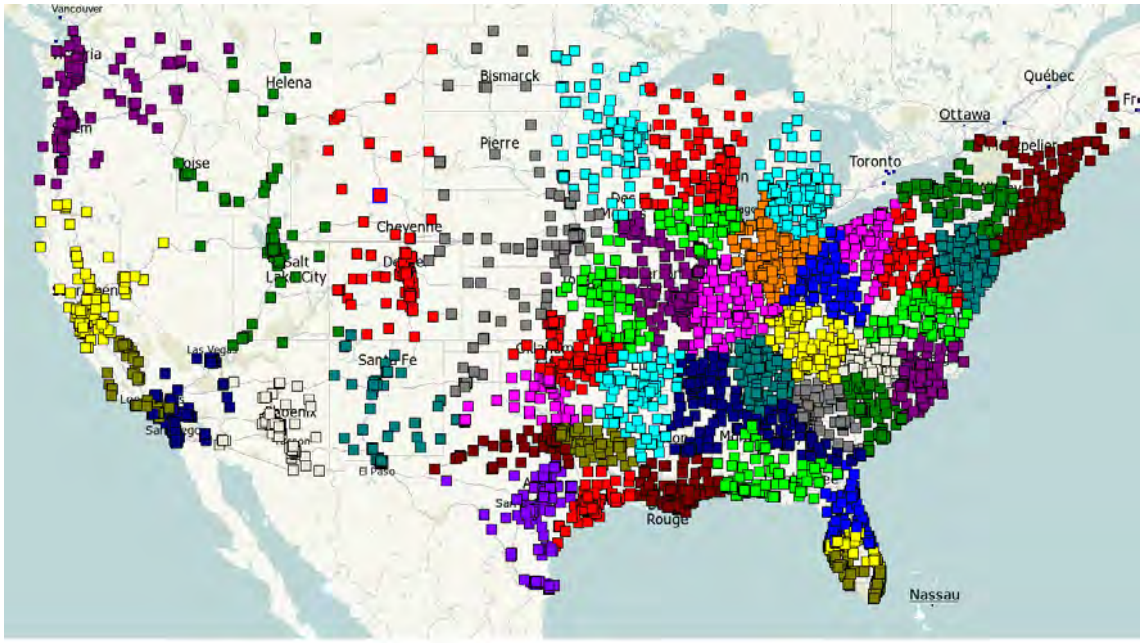


Figure A.64: USA II OTR

Table A.43: USA II Results KPIs

Mean Size (Custom Input)	892336,36
Standard Deviation	8383,55
Ratio Maximum to Expected	1,015
Ratio Minimum to Expected	0,984
Average Neighbourhood Ratio	0,8159
Average Silhouette Coefficient	0,2649
Davies-Bouldin Index	0,6767
Scaled Compactness	0,135
Scaled Centerpoint Distance	0,097
Running Time Total	00:03:12
Running Time Algorithm	00:02:39

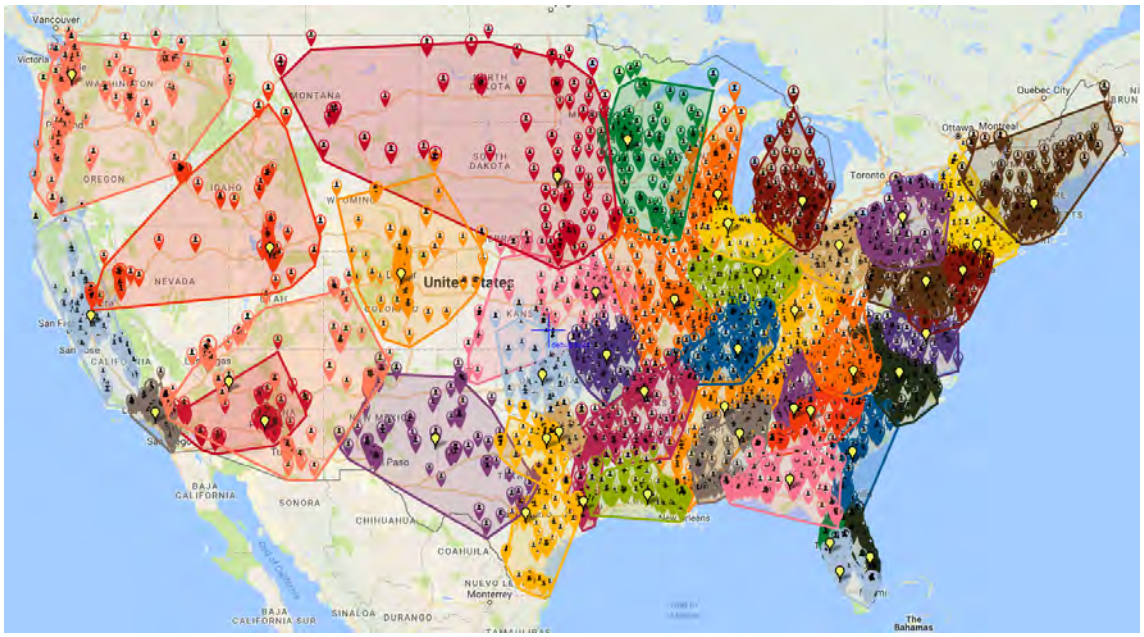


Figure A.65: USA II Results Map

Bibliography

Agrawal, R., J. Gehrke, D. Gunopulos, and P. Raghavan (1998). Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD '98 Proceedings of the 1998 ACM SIGMOD international conference on Management of data 27*(2), 94 – 105.

Andrew, A. (1979). Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters 9*(5), 216 – 219.

Ankerst, M., M. Breunig, H. Kriegel, and J. Sander (1999). Optics: Ordering points to identify the clustering structure. *SIGMOD '99 Proceedings of the 1999 ACM SIGMOD international conference on Management of data 28*(2), 49 – 60.

Arthur, D. and S. Vassilvitskii (2006). k-means++: The advantages of careful seeding. *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*.

Baeza-Yates, R. and B. Ribeiro-Neto (1999). *Modern Information Retrieval*. Addison Wesley, New York.

Banerjee, A. and J. Ghosh (2004). Frequency sensitive competitive learning for balanced clustering on high-dimensional hyperspheres. *IEEE Transactions on Neural Networks 15*(3), 702 – 719.

Berkhin, P. (2006). A survey of clustering data mining techniques. In J. Kogan, C. Nicholas, and M. Teboulle (Eds.), *Grouping Multidimensional Data.*, pp. 25–71. Springer Berlin Heidelberg.

Bowerman, R., P. Calamai, and G. Hall (1994). The spacefilling curve with optimal partitioning heuristic for the vehicle routing problem. *European Journal of Operational Research 76*, 128 – 142.

Choo, J., R. Jiamthapthaksin, C. Chen, O. Celepcikay, C. Giusti, and C. Eick (2011). Mosaic: A proximity graph approach for agglomerative clustering. *Data Warehousing and Knowledge Discovery*.

Chou, C., W. Chaovalitwongse, T. Berger-Wolf, B. DasGupta, and M. Ashley (2011). Capacitated clustering problem in computational biology: Combinatorial and statistical approach for sibling reconstruction. *Computers & Operations Research 39*(3), 609 – 619.

Cormen, T., C. Leiserson, R. Rivest, and C. Stein (2014). *Introduction to Algorithms*. Cambridge, Massachusetts: The MIT Press.

Dantzig, G. and J. Ramser (1959). The truck dispatching problem. *Management Science 6*(1), 80 – 91.

Davies, D. and D. Bouldin (1979). A cluster separation measure. *IEEE transactions on Pattern Analysis and Machine Intelligence 1*(2), 224 – 227.

Day, W. and H. Edelsbrunner (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification 1*, 7 – 24.

Defays, D. (1977). An efficient algorithm for a complete link method. *The Computer Journal 20*(4), 364–366.

Dempster, A., N. Laird, and D. Rubin (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society 39*(1), 1 – 38.

Desgraupes, B. (2013). *Clustering Indices*. R package version 1.2.7.

Deza, M. and E. Deza (2016). *Encyclopedia of Distances*. Springer-Verlag Berlin Heidelberg.

Drineas, P., A. Frieze, R. Kannan, S. Vempala, and V. Vinay (1999). Clustering in large graphs and matrices. *In SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*.

Dunn, J. (1974). Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics 4*, 95 – 104.

Elkan, C. and G. Hamerly (2002). Alternatives to the k-means algorithm that find better clusterings. *Proceedings of the eleventh international conference on Information and knowledge management*.

Ester, M., H. Kriegel, J. Sander, and X. Xu (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 226 – 231.

Fernandez, C., V. Moreno, B. Curto, and J. Vicente (2010). Clustering and line detection in laser range measurements. *Robotics and Autonomous Systems 58*, 720–726.

Filipovych, R., S. Resnick, and C. Davatzikos (2011). Semi-supervised cluster analysis of imaging data. *NeuroImage 54*(3), 2185–2197.

Ford, L. and D. Fulkerson (1956). Solving the transportation problem. *Management Science 3*(1), 24–32.

Gabriel, K. and R. Sokal (1969). A new statistical approach to geographic variation analysis. *Systematic Zoology 18*(3), 259–278.

Ghiasi, S., A. Srivastava, X. Yang, and M. Sarrafzadeh (2002). Optimal energy aware clustering in sensor networks. *Sensors 2*, 258–269.

Guha, S., R. Rastogi, and S. K (2000). Rock: A robust clustering algorithm for categorical attributes. *Information Systems 25*(5), 345–366.

Guha, S., R. Rastogi, and S. K (2001). Cure: An efficient clustering algorithm for large databases. *Information Systems 26*(1), 35–58.

Gupta, G. and J. Ghosh (2001a). Detecting seasonal trends and cluster motion visualization for very high dimensional transactional data. *First SIAM Conference on Data Mining*, 115–129.

Gupta, G. and J. Ghosh (2001b). Value balanced agglomerative connectivity clustering. *SPIE conference on Data Mining and Knowledge Discovery* (3), 115–129.

Gupta, G. and M. Younis (2003). Load-balanced clustering in wireless sensor networks. *IEEE International Conference on Communications 3*, 1848–1852.

Jain, A. and R. Dubes (1988). *Algorithms for Clustering Data.* Prentice-Hall, Englewood Cliffs, N.J.

Kaufman, L. and P. Rousseeuw (1990). *Finding Groups in Data: an Introduction to Cluster Analysis.* John Wiley & Sons, 1990.

Kernighan, B. and S. Lin (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Tech 49*, 291 – 307.

Kriegel, H. and E. Schubert (2016). The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowledge and Information Systems 52*, 341–378.

Kulkarni, R. and P. Bhave (1985). Integer programming formulations of vehicle routing problems. *European Journal of Operational Research 20*, 58 – 67.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Trans Inf Theory 28*(2), 129 – 136.

Lynch, P. and S. Horton (2016). *Web Style Guide, 4th Edition*. Yale University Press.

McLachlan, G. and K. Basford (1988). *Mixture models : inference and applications to clustering*. Marcel Dekker, New York.

Murtagh, F. and P. Contreras (2011). Methods of hierarchical clustering. arxiv: Cs 1105.0121.

Openshaw, S. (1977). A geographical solution to scale and aggregation problems in region-building, partitioning and spatial modelling. *Transactions of the Institute of British Geographers 2*(4), 459–472.

Ordoñez, A., H. Ordonez, J. Corrales, C. Cobos, L. Wives, and L. Thom (2017). Grouping of business processes models based on an incremental clustering algorithm using fuzzy similarity and multimodal search. *67*, 163–177.

Raymond, T. and J. Han (1994). Efficient and effective clustering methods for spatial data mining. *In Proceedings of the 20th VLDB Conference*, 144–155.

Rosenkrantz, D., R. Stearns, and P. Lewis (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing 6*(3), 563–581.

Rousseeuw, P. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics 20*, 53 – 65.

Sander, J., M. Ester, H. Kriegel, and X. Xu (1998). Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery 2*(2), 169–194.

Sheikholeslami, G., S. Chatterjee, and A. Zhang (2000). Wavecluster: a wavelet-based clustering approach for spatial data in very large databases. *The International Journal on Very Large Data Bases 8*, 289–304.

Sibson, R. (1972). Slink: An optimally efficient algorithm for the single link cluster method. *The Computer Journal 16*(1), 30–34.

Skillicorn, D. and C. Leuprecht (2015). Clustering heterogeneous semi-structured social science datasets. *Procedia Computer Science 51*, 2908–2912.

Strehl, A. and J. Ghosh (2003). Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing 15*(2), 208–230.

Varghese, B., A. Unnikrishnan, and K. P. Jacob (2013). Spatial clustering algorithms - an overview. *Asian Journal of Computer Science And Information Technology 3*(1), 1–8.

Veness, C. (2017). Calculate distance, bearing and more between latitude/longitude points. `http://www.movable-type.co.uk/scripts/latlong.html`.

Villalba, L. and A. Corripio (2015). Smartphone image clustering. *Expert Systems with Applications 42*(4), 1927–1940.

Voorhees, E. (1986). Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management 22*(6), 465–476.

Wang, W., J. Yang, and R. Muntz (1997). Sting : A statistical information grid approach to spatial data mining. *Proceedings of the 23rd International Conference on Very Large Data Bases*, 186–195.

Ward, J. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association 58*(301), 236–244.

Zhang, T., R. Ramakrishnan, and M. Livny (1996). Birch: An efficient data clustering method for very large databases. *SIGMOD '96 Proceedings of the 1996 ACM SIGMOD international conference on Management of data 25*(2), 103–114.