ERASMUS UNIVERSITY ROTTERDAM

ERASMUS SCHOOL OF ECONOMICS

MASTER THESIS QUANTITATIVE LOGISTICS AND OPERATIONS RESEARCH

Clustering for Vehicle Routing in Kenya

Author
Florine GROENEN - 369570

Supervisor
Dr. T. DOLLEVOET
Second assessor
Dr. R. SPLIET

April 14, 2018

Abstract

When attempting to solve the extensively researched vehicle routing problem (VRP) in a non-western context, literature may still be limited. The problem in this paper is presented by a Kenyan company, which copes with four issues regarding their daily VRP: they prefer fixed drivers to visit each demand location, demand is very infrequent and unknown until one day in advance and the set of demand locations is highly variable. As a solution, a two-phase approach to solve the problem was introduced, consisting of a clustering phase and a daily route optimization phase. Clusters should be formed for a longer period of time and such that daily routes are feasible. The K&P algorithm and the Districting algorithm are proposed for clustering, which have arisen from respectively the Capacitated Cluster Problem and the Districting Problem. Both are evaluated by means of transportation costs and the cluster stability, the percentage of demand locations served by the same driver after clustering, for the purpose of evaluating how and whether the two can be balanced. As a result, it was shown that especially the K&P algorithm leads to good results, with an average cluster stability of 70%. The cluster stability could be increased further by implementing a simple extension. In addition, a proper method for constraining the clusters was introduced. Finally, it is recommended to the company to neglect the distance to the depot in the clustering objective, rather than incorporating it, to enhance the compactness of the clusters. This was found to result in lower costs and an increased cluster stability. In addition, it is advised to adopt growth strategies such that new customers are found both within and outside of the current territory. This will lead to a decrease in costs by searching within the territory and an increase in cluster stability by searching outside the territory.

Contents

1 Introduction **3**

2 Literature **4**

3 Problem description **7**

4 Data **9**

4.1 Data cleaning 11

4.2 Modeling agent properties and behavior 12

4.3 Overview of assumptions 13

5 Methodology **14**
5.1 General approach . . . 14
5.2 K&P algorithm . . . 15
5.3 Districting algorithm . . . 18
5.4 Differences between the two methods . . . 23
5.5 Constraining the cluster capacity . . . 23
5.6 Sensitivity analysis on growth strategies . . . 25

6 Program implementation **25**
6.1 Program development . . . 25

7 Tuning the K&P algorithm **28**

8 Tuning the Districting algorithm **30**

9 Results **31**
9.1 Algorithm performance . . . 31
9.2 Evaluation of the capacity constraints . . . 34
9.3 Increasing cluster stability . . . 37
9.4 Impact evaluation of possible growth strategies . . . 38

10 Conclusion **40**

A Data cleaning **43**

B Outlier detection **44**

C Distributions **44**
C.1 Distributions drop-frequency . . . 44
C.2 Distributions arriving and leaving rate . . . 46

D Best Driver Problem formulation **48**

E Programming details **49**

1 Introduction

The vehicle routing problem (VRP) is a well-known and thoroughly researched topic in logistics. Many solutions to solve the problem have been given over the years, which have often proven to deliver good results. However, these solutions are usually applied to problems within the Western world. That means they have found to be applicable to problems within the context of such countries. When a vehicle routing problem arises in the rural areas of an African country such as Kenya, the situation can be different to such an extent that a revision of common VRP methods is necessary.

The problem presented in this research is from a Kenyan company that sells products via an agent structure to the rural and peri-urban areas of the country. Customers buy and pick up products at these agents and the company delivers the products six days a week to these agents. These agents form a fine-mesh distribution network in which currently around 1500 agents are involved. The problem the company is facing with respect to their distribution network consists of four aspects: agents should be visited preferably by the same (fixed) driver, demand on a certain day is unknown until one day in advance, many agents have very infrequent demand and lastly, the set of agents varies greatly (often, new agents arrive and current agents leave). Though the issue of fixed drivers is the main problem in this paper, all of these three latter issues add an extra challenge to solving it.

In the above described situation, a solution method to solving the issue of having fixed drivers could be to cluster the agents together, prior to finding daily optimal routes within these clusters when demand becomes known. This will be referred to as the *two-phase approach* to solving the VRP. Clusters should be formed such that daily routes through the cluster are feasible within a fixed time frame and are cost efficient. At first sight, this approach seems to solve the problem. However, because of the highly varying set of agents, the clusters should be updated frequently. This may lead to variations in the clusters and consequently, driver-agent relations may not be fixed for long. Clustering is thus in this case about finding a balance between cluster stability and cost efficiency of the clusters over time, given a set of constraints. To the best of our knowledge, no research has been done into this specific problem and this paper will therefore make a first attempt at it.

The research will be split up in two research questions. First of all, it is questioned what clustering methods one should use to balance cluster stability and transportation costs. This question will be split in different subquestions. As a start, what will be the effects of various clustering methods on the cluster stability and transportation costs? Secondly, how can one constrain the clusters in terms of size, such that daily routes are feasible? And

lastly, what type of cluster shape, either compact and round or long and narrow, would be preferred in this context? In addition, the company is interested in suggestions about possible growth strategies in terms of where to find new agents. Therefore, the second research question is whether it matters to both the cluster stability and the transportation costs, where in the network new agents are found.

The remaining part of this paper will be ordered as follows: First, the literature on this topic will be summarized, followed by a more detailed description of the problem and the corresponding available data. Hereafter, the methodology will be discussed. Finally, the details of the implementation and the results will be discussed, followed by a conclusion.

Note: the words vehicle, driver and cluster and the words agent and customer will be used interchangeable throughout this paper, as they refer to the same. This also implies that a customer no longer refers to the person that buys commodities at the agents. This person is referred to as a buyer instead.

2 Literature

Over the years, issues regarding consistency within VRP methods have been thoroughly researched. The VRP problem posted in this paper contains also an issue of consistency, namely the requirement of fixed drivers. In literature, this type of consistency is labeled person-oriented consistency. Person-oriented consistency can be seen from both the customer (driver-consistency) and the driver perspective (customer-consistency). That is, a customer could appreciate being visited by the same driver as well as that the driver could appreciate having to visit the same customers (Kovacs et al., 2014). Both are relevant to the problem, though customer-consistency being more important than driver-consistency. In any case, as the problem is strictly person-oriented consistent, this consistency should hold from both perspectives. Problems requiring person-oriented consistency arise, for example, because it can be used to fasten the learning curve of a driver (Zhong et al., 2007). Kovacs et al. (2014) present an overview of available research on VRP methods that account for person-oriented consistency. However, these methods are usually not designed to assure fixed drivers, but only to provide some level of person-oriented consistency. An exception to this is the method of Spliet and Dekker (2016), which allows the level of consistency to vary. They introduce the Driver Assignment Vehicle Routing Problem (DAVRP) and attempt to solve it. The idea behind it is to assign drivers to customers before demand is known and requires a driver to visit at least a fraction α of its assigned customers after demand becomes known. Note that when $\alpha = 1$, the problem is

equal to requiring fixed drivers. This requirement should hold for a given set of demand realizations. For values of α lower than 1, they allow backup drivers to visit unplanned orders. The clustering problem, called driver assignment, is formulated as an integer linear programming problem and is solved by means of a branch-and-bound algorithm. The impact of driver consistency on transportation costs is evaluated based on multiple problem instances and their findings show on average an increase in costs of 12.7% when $\alpha = 1$, implying fixed drivers, as opposed to $\alpha = 0$. The maximum found increase over all problem instances was 20.9% in this case.

The possibility to split the VRP into two problems, as is done in the two-phase approach, was also recognized by Fisher and Jaikumar in 1981. They noticed that the problem could be split into a Generalized Assignment Problem (GAP) and a Traveling Salesman Problem (TSP). Here, the GAP deals with the clustering of customers into vehicles under a capacity constraint, while the TSP is solved for the actual routing within the clusters. Their formulation assumes demand to be known. Note that this split is not done in order to create long term clusters, but is simply a different approach to solving a VRP formulation. However, if demand was to be constant over time, the GAP would provide a clustering solution. Solving the GAP in their paper aims to minimize the costs of adding a customer to a cluster, in terms of the extra distance a vehicle needs to travel to visit the customer when traveling from the depot to the center of the cluster and back. The center of the cluster is represented by the seed customer, which is considered to be the median customer of a cluster. Similarly, one could also solve a Capacitated Clustering Problem (CCP), instead of solving the GAP. The two problems are very similar, with the main difference that the GAP requires preselected seed customers for the clusters, whereas the CCP does not. Instead, the CCP finds these seed customers as part of the optimization. Though the CCP can be used for all kinds of clustering problems, it has also been used for vehicle routing before. Various heuristics for CCP problems are known, such as the one proposed by Mulvey and Beck (1984), which has been improved by Koskosidis and Powell (1992). The objective of their CCP problem is to choose a median customer as the seed of each cluster that minimizes the distance of all customers in the cluster to its corresponding median. Their heuristic contains three phases. The first phase assigns customers to an initial set of seed customers, by the regret function. The regret function is defined by the difference in distance between the first and second closest seed. In short, it is a penalty for assigning a customer to the second closest seed instead of the closest seed. Customers are assigned to a cluster in decreasing order of regret. The second phase of the heuristic recomputes the median of each cluster and updates the seed of the cluster in case this is a different customer than the current seed customer. The third phase of the

heuristic improves the solution by making local switches (Mulvey and Beck, 1984). The heuristic of Koskosidis and Powell (1992) extends this algorithm by avoiding using randomly selected seeds in the initial phase of the heuristic and by making use of previously obtained information through iterating between the three phases. Negreiros and Palhano (2006) solve a problem very similar to CCP, except that the seed of a cluster is defined to be the center of the coordinates in a cluster, rather than the median customer. This specific problem they refer to as the Capacitated Centred Clustering Problem (CCCP). They propose a heuristic consisting of two phases: a constructive phase and an improvement phase. This improvement phase makes use of a Variable Neighbourhood Search.

Note that the GAP, CCP and CCCP are not necessarily meant to create long term solutions for clustering. Districting methods on the other hand aim to divide a set of customers or an area for a longer period of time. Within districting problems the districts (clusters) are defined before demand is known and the specific routing decisions are made after demand becomes known (Kovacs et al., 2014). Districting can be required in all kind of situations, such as political districting and health care districting. In this case the focus will be specifically on districting methods for vehicle routing with uncertain demand. To our knowledge, the earliest research on this topic was done by Christofides in 1971. He suggested an algorithm that is based on the intuitive proposition that "the best design of fixed routes is such that the sum of the frequencies of the links (i , j) where i is a customer on one route and j a customer on a different route, is as small as possible". That means, the more often the two customers were on the same route in a sequence of daily route optimizations, the higher the need to put them in the same district. His algorithm simply lists this number for each pair of customers in a descending order and creates the districts based on this sequence. In the case where the number and location of customers also vary from day to day, which is a situation that could account for both infrequent customers and a variable set of agents (to a certain extent), these fixed routes are turned into fixed areas. This is done by drawing squares around all the customers and adjusting the algorithm slightly such that customers can only be in the same fixed area if their squares have a common boundary. Wong and Beasley (1984) use the same idea, but aim to minimize total costs corresponding to all combinations of having two customers in the same subarea. As soon as two customers occur relatively often on different routes instead of on the same routes over a specified period of time, these costs are higher. As such, in addition to the number of times two customers occur on the same route, which is used in the algorithm of Christofides (1971), they also take into account the number of times the customers are on a different route. When creating the division, a workload constraint is taken into account, which restricts the number of customers in a district. It is noted that it is easier to con-

strain these areas in terms of capacity than in terms of total distance traveled. Similarly, Lei et al. (2012) present an algorithm for districting with uncertain demand, whereby a distinction is made between customers with and without uncertain demand. Contrary to the previous algorithms, it minimizes the fixed and variable vehicle costs as well as the compactness of the areas. To measure compactness it assumes an equal distribution of customers over a compact area. Their approach uses the division of the total area into squared basic units, which are merged to the nearest unit in case it does not contain a customer. An initial solution is created by selecting a basic unit as seed and expanding its district by adding the neighboring unit leading to the lowest increase in district workload. Thereafter, this initial solution is improved by making use of a large neighborhood search heuristic.

3 Problem description

For the sake of understanding all four aspects of the posted problem (requirement of fixed drivers, unknown and infrequent demand and a highly variable set of agents), the Kenyan situation with which the company deals will be discussed in more detail. The most important problem for the company is that whenever a new agent is added to the network, it takes a long time for a driver to find this new agent. This is because of various reasons, such as the absence of actual addresses and proper navigation solutions. In order for drivers to deliver the orders it is therefore important that the driver is familiar with the environment of the agents. At the same time, it increases the service level of the company by increasing trust between the agent and the driver. As such, ideally, each agent should have a fixed driver. In addition to this issue, the agent demand only becomes known one day in advance. This implies that this demand is unknown when forming clusters and as such, these fixed drivers should be able to deal with any daily demand realization. Thirdly, many agents do not have orders every day but rather for example every week or two times a week. These *infrequent agents* thus have a high probability of zero-demand. In such a situation, it would be cost efficient to include many agents in one cluster, because the number of agents that needs to be visited on an average day will be much lower. However, at the same time, daily routes should remain feasible for the drivers. Lastly, the company operates in a highly changing environment, meaning that frequently, new agents pop up and current agents leave. The variability is defined by the daily rate with which agents arrive and leave expressed in number of agents, referred to as the *daily arriving rate* and the *daily leaving rate*. This leads to an extra challenge when forming clusters, because these clusters will have to be updated regularly.

In the two-phase approach to the VRP that will be considered in this paper, clustering will be done each *cluster period* and the actual route planning on a daily basis. That is, new clusters are formed at the start of each cluster period and new agents arriving between these clustering runs are simply added to an existing cluster by means of some criteria. As such, clusters are formed before demand is known and the daily routes are formed only after demand becomes known. Creating the daily routes is done at the start of each day and can be done by simply solving the TSP for all agents with demand in a cluster. The cluster period is set equal to two weeks, which is equal to 12 working days and will be evaluated based on half a year, consisting of 144 working days. Because of the fixed driver requirement it is important for the company to evaluate the clusters not only in terms of *transportation costs*, but also in terms of cluster stability. Transportation costs are expressed in Kenyan Shillings (KES) and consist of a variable cost per kilometer (13.23 KES) and per vehicle in use (6400 KES). The *cluster stability* of a certain clustering solution is defined as the percentage of agents that will be served by the same vehicle as in the previous clustering solution. As such, the first research question, as formulated before, is what clustering methods are able to balance *both* performance measures. Although both measurements are thus important, the objective of the clustering problem is to minimize the transportation costs. Consequently, the effect on cluster stability can be analyzed and a proper balance can be found. However, in this paper the focus will be on minimizing only one aspect of the costs, namely the total distance. Deciding on the number of vehicles will be done separately, for the sake of simplification.

Logically, clusters can be only of limited size in order to have feasible daily routes. Because of the badly working navigation devices in Kenya, leading to differences between estimated and actual driving times, the company prefers putting constraints on the number of agents visited, rather than on time. The company considers 30 to be the maximum number of agents that can be visited by one driver per day and 22 the corresponding average. The number of agents that require a visit on a day will from now on be referred to as the number of *drops*. Additionally, the frequency of an agent with which a drop occurs is referred to as the *drop-frequency* and is expressed per day. By constraining the clusters in terms of drop-frequency, the average number of stops can be restrained. Doing research into whether the cluster size can be constrained in this way will help to answer this first research question. Note that the company has no other capacity constraints in terms of volume or weight on the vehicles.

Another idea with respect to different clustering approaches, is that different cluster shapes could be obtained by including versus neglecting the costs of traveling from and to the depot for each of the clusters. This approach is best visualized, respectively, as

clustering like a daisy (Figure 1) or like a bouquet of roses (Figure 2). That is, when using the daisy-method, the distance from and to the depot is incorporated in the optimization and as such, the clusters are expected to look like loops from and to the depot (D). In the roses-method, each cluster is a rose: compact, though without link to the depot (D). This method thus neglects the distance to and from the depot. It is suspected that clustering like roses would be a better method for this specific problem, because the cluster centers of such compact clusters are expected to vary less in such a dynamic environment, likely leading to a higher cluster stability. However, it is questioned if the incorporation or exclusion of the distance to and from the depot indeed impacts the shape of the clusters. This and the question whether there is a preference for either method when valuing both cluster stability and transportation costs are subquestions of the first research question.

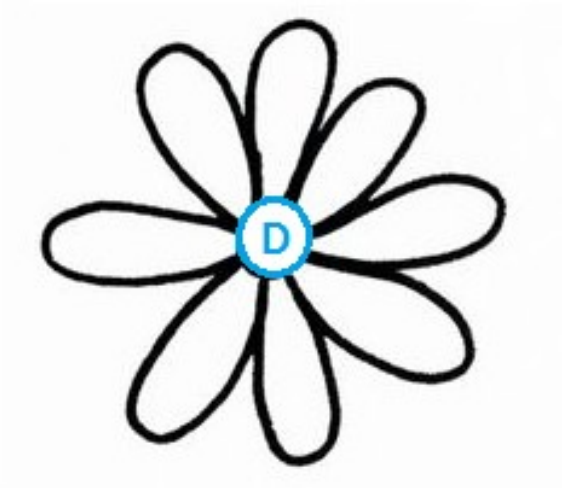

Figure 1: Clustering according to a daisy

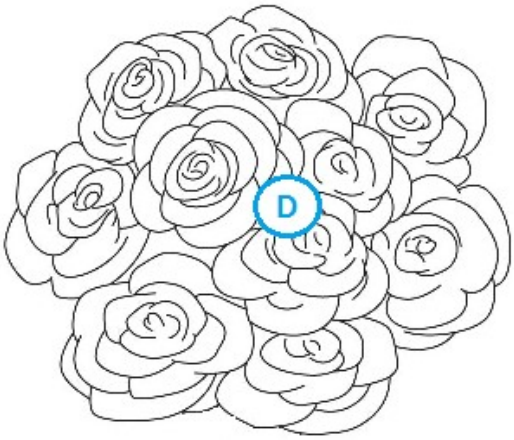

Figure 2: Clustering according to a bouquet of roses

Finally, the company is expecting to grow further and as such, is wondering if it matters where its sales team looks for new agents. Two common, though opposing, strategies are to look for agents outside the current area and to find more agents within the current area. The effects of altering the location of arriving agents onto cluster stability and transportation costs should be evaluated as to answer the second research question concerning possible growth strategies for the company.

4 Data

For the sake of evaluating different clustering methods, these methods will be applied to simulated and real data. The real data is used to evaluate the actual performance of the methods and the simulation is used mainly to analyze the sensitivity of the performance

measures to certain characteristic of the set of agents. For this purpose, data about the agents over a longer period of time is necessary. The dataset that is made available by the company includes data from July 2016 to and including June 2017 and thus consist of data of one year. The clustering methods will be applied onto the first 24 weeks (144 days) of 2017, which corresponds to approximately the second half of the dataset. This is for the reason that we would like this same period to be predicted in the simulation, which needs the first half year as a learning set for modeling the data characteristics. This implies that the learning set corresponds to all the data of 2016. The details of the simulation will be discussed in Section 4.2.

The obtained dataset contains all orders of all agents over the year. Note that an order corresponds to the demand of a buyer and as such, multiple orders can be placed at an agent per day. By aggregating the orders of each agent per day, the total daily demand per agent can be retrieved. The dataset includes the following information per data point (per order):

- *Order Number*: unique six digit number corresponding to the order
- *Order Delivery Date*: delivery date of the order
- *Order Quantity*: number of products that are ordered
- *Agent ID*: unique four to six digit number corresponding to an agent
- *Agent Name*: name of the agent
- *Agent Lat* and *Agent Lon*: latitude and longitude coordinates of the agent
- *Agent Join Date*: date that the agent joined the company
- *Agent Status*: status of the agent on the 1st of July 2017 (active or inactive)
- *Agent Last Order Date*: date of most recent order at the agent on the 1st of July 2017
- *Agent Cluster*: the one cluster to which the agent is currently connected, indicated by a number

Note that some characteristics are defined per agent, while others are defined per order. The dataset of one year contains 271,493 data points (orders). When aggregating the orders of each agent per day, there are 100,454 entries left. In total, the dataset contains 2,591 unique agents. In the last month of the dataset, 1,599 agents had an order, and are therefore considered active agents on the 1st of July. The data is provided in Excel.

In addition, certain information was not directly part of the original data, but required for this research. Fortunately, they could be derived from the dataset. This considers the following information:

- *Agent Leaving Date*: this date is defined to be non-existing in case the Agent Status is active and to be equal to the Agent Last Order Date in case the Agent Status is inactive
- *Agent Daily Demand Occurrence*: binary variable indicating if the agent had a demand (1) or no demand (0) on a certain day
- *Daily Arriving Rate*: number of arriving agents per day based on the Agent Join Date of all agents
- *Daily Leaving Rate*: number of leaving agents per day based on the Agent Leaving Date of all agents

Note that the latter three characteristics are defined in different units (per day or per agent per day) than the rest of the data. Note that when using the Agent Daily Demand Occurrence, we no longer need to bother with the order quantity. This is preferred for this problem, because capacity constraints are put solely on the number of agents in a cluster.

4.1 Data cleaning

In order to be able to use the dataset, the set is cleaned by means of error and outlier detection. Errors are detected by performing multiple checks. For example, it should be checked if all order dates of an agent are between its join data and last order date and whether there are any irregularities in the data (specifically in the Agent Lat and Lon, the Agent Join Date and the Agent Cluster, as these are expected to have irregularities due to administrative difficulties according to the company). An overview of all checks and the corresponding adjustments made to the data can be found in Appendix A.

Hereafter, the dataset is checked for outliers. First of all, it was observed that the data contained many agents that had only one or two drops, while already being active for a long period of time. In consultation with the company, it was decided to remove such agents from the dataset. The remaining dataset of agents consists of 2,296 agents and 100,013 orders.

Lastly, a specific outlier detection is done for the data of the Daily Arriving Rate and Daily Leaving Rate. Boxplots were used to find outliers in these data. When using boxplots for outlier detection, one usually specifies an observation to be an outlier, when it

lays outside a certain distance from the interquartile range. This distance is defined by a constant k, which represents k times the interquartile range. This leads to a range for accepting observations equal to $[Q1 - k(Q3 - Q1), Q3 + k(Q3 - Q1)]$, where Q1 and Q3 are the first and third quartile respectively. Like McGill et al. (1978) have suggested, in this research we set $k = 3$ in order to find the outliers. No outliers were found in the data. A more detailed description of the outlier detection can be found in Appendix B.

4.2 Modeling agent properties and behavior

Recall that the simulation of agents is necessary to evaluate the sensitivity of the performance measures of certain clustering methods to certain characteristics of the set of agents. For the simulation, predictions of the characteristics of the set of agents for the first 144 days of 2017 are necessary. The necessary characteristics in this research are limited to the daily arriving and leaving rate of agents, the coordinates of agents, the drop-frequency of agents and the demand occurrences at agents. Sensitivity analysis can then be applied to certain input parameters of the simulation, which will be discussed in detail in Section 5. All required information will be simulated based on the information obtained from the learning data, which was considered all available data of 2016. An overview of the found assumptions for simulating these parameters are found in the next subsection (Section 4.3). The remaining part of this section will only discuss how these assumptions were obtained.

For the arriving and leaving rates and the drop-frequency, distributions were estimated based on the learning data. In order to find theoretical distributions that fit the data best, QQ plots are used to visually find a good fit, the Jarque-Bera statistic is used to check for normality and the Kolmogorov-Smirnov statistic to verify the level of fit. The details of this analysis can be found in Appendix C. In summary, for all parameters, finding a proper distribution to fit the data was rather complicated. Also the Kolmogorov-Smirnov statistics usually did not show a good fit for the chosen distributions. However, as the purpose of the simulation is a sensitivity analysis, the accuracy of prediction is not our main concern. In addition, the clustering models will be evaluated with measurements based on the dataset as a whole. As such, the details of the simulation are of less importance.

Similarly, it suffices to use the coordinates of all active agents on January 1st 2017 as a starting point of the coordinates of the set of agents at the start of the simulation. Obviously, arriving agents do not just pop up everywhere, but some areas may be more or less dense. Nevertheless, coordinates of newly arriving agents will be generated uniformly within a specified squared area, that is equal to the area used for data cleaning of the coordinates (see Section A). This area accounts for growth both within and outside the starting

territory and is approximately 20% larger than the starting territory. The area corresponds to the same area as in which new agents appear in the real data during that period. This is acceptable, because the distance between two places is calculated by the great circle distance anyway and not by making use of any navigation device. However, it should be remembered throughout the research that the real distribution of the coordinates is not uniform. Unfortunately, no common benchmarks can be used for the purpose of simulating coordinates, because they do not correspond to the the problem presented in this paper.

To obtain a model for simulating the demand occurrences, remember that one simply distinguishes between zero and non-zero demand. Because of the high probability of non-zero demand, it works well to simply accept a demand occurrence with a certain probability. The drop-frequency in this case is assumed to be the acceptance probability. In case demand is rejected, the demand occurrence variable is set equal to 0, else it is set equal to 1.

4.3 Overview of assumptions

Below one can find an overview of all the assumptions that will be used in the simulation in this paper.

- The coordinates of a new agents are uniformly distributed over the range [-1.33, -0.35] for the lat-coordinate and [36.17, 37.60] for the lon-coordinate.
- The drop-frequency is discretely exponentially distributed over the 24 values from the set $\{\frac{1}{24}, \frac{2}{24}, ..., \frac{24}{24}\}$, with $\lambda = \frac{1}{0.35}$. In case of a simulated value above 1, this value will be discarded and a new value will be simulated.
- The daily arriving rate of new agents is drawn from a geometric distribution with $p = \frac{1}{5.40+1}$.
- The daily leaving rate of agents is drawn from a geometric distribution with $p = \frac{1}{3.12+1}$.
- A demand occurrence is assumed to be accepted with a probability equal to the drop-frequency. This means that a demand occurrence on a certain day is accepted if a draw from the uniform distribution on the interval of [0,1] is below the value of the drop-frequency.

5 Methodology

5.1 General approach

The discussed two phase approach, consisting of a clustering phase and a route planning phase, will be applied in this research. The focus hereby lays on finding a proper clustering method to be used in the first phase. Evaluation of various clustering methods will take place based on two performance measures, namely the cluster stability and the transportation costs. For the clustering phase, two different methods based on the literature review will be applied:

1. Algorithm of Koskosidis and Powell
2. Algorithm of Wong and Beasley

Note that each of these methods have risen from a different problem, namely the CCP and the Districting Problem. This is for the purpose of comparison. The main difference between the two problems is that these have risen from a, respectively, short term versus long term perspective for finding clusters. Later it will become clear that another important difference is that the CCP does not take into account a depot, while the Districting Problem does (see Section 5.4). For the sake of clarity, the first algorithm will be referred to as the *K&P algorithm* and the second algorithm as the *Districting algorithm*. The choice for the above two methods is made because they seem to be most suitable to the problem. Other methods were not chosen because, amongst other reasons, the above presented algorithms were improvements of these methods, had large computation times or were rather complicated to implement considering the fact that the purpose of this research is finding the impact on cluster stability and costs. In addition, these two algorithms are attractive because they are very intuitive. Finally, it is important to note that both problems do not explicitly minimize total costs nor distance. However, their objectives are sufficiently related to the total distance and thus costs, to be considered proper methods for solving the clustering problem.

The implemented program will run the two-phase approach over 144 days, by solving the clustering run every cluster period (two weeks) by means of one of the two algorithms. The cluster stability can be calculated after every clustering run, while the transportation costs can be taken from the solution to the daily route planning problem. The cluster stability which is measured after each clustering run, will be found by linking the new found clusters to the previous clusters in such a way that the cluster stability is at its optimum. From a driver perspective, this means that clusters should be linked such that the highest

number of agents are served by the same driver. To find this optimum the Best Driver Problem, an integer programming problem, is introduced, of which the formulation is explained in Appendix D. The daily route planning problem can be simply described as the TSP, which finds the minimum route through all agents with demand, starting and ending at the depot.

Because the methods do not optimize the number of vehicles, as a start, this number will be fixed to 25 vehicles, which represents the current fleet of the company. This will be compared to using the minimum number of vehicles necessary to satisfy the vehicle capacity constraints. In this case the number of vehicles will increase over time whenever growth in the set of agents leads to an infeasible clustering solution. Assuming 25 vehicles will give a feasible solution in all cases, the latter case should always lead to a number of trucks that is less than or equal to 25. Moreover, the distance between agents will be calculated using the great circle distance multiplied by a distance correction factor. This distance correction factor will be based on routes in Google maps for a random sample of agent combinations. Furthermore, the company currently makes use of two depots to serve all agents. Because it would be extremely complicated to make the choice of which depot serves which agent a variable in the optimization, the depot from which an agent is served will be fixed. Linking agents to the depots is done in consultation with the company.

Unfortunately, both algorithms as presented do not account explicitly for infrequent agents. However, both include constraints on the cluster size that are unnecessary and that can be easily adapted to fit the problem, such that they account for the order-frequencies instead. This will be explained in Section 5.2 and 5.3. Because the maximum number of agents per cluster on a certain day should hereby be restricted, the instances for which the TSP will be solved, will be relatively small. As such, the corresponding mixed integer programming problem can be solved in the second phase of the two-phase approach.

5.2 K&P algorithm

The first algorithm that will be discussed, is the algorithm as presented by Koskosidis and Powell (1992), which aims to solve the CCP. In summary, the CCP presents the problem of assigning each customer in a set of customer to a seed customer under a set of constraints (recall that customers and agents are the same). The seed customer can be seen as the representative of a cluster and is usually located centrally in the cluster. Based on the formulation presented by Koskosidis and Powell, the following sets, variables and parameters will be defined:

sets:

C : set of customers $i = 1, ..., I$

S : set of potential seed customers $j = 1, ..., J$

variables:

$$x_{ij} = \begin{cases} 1 & \text{if customer } i \text{ belongs to cluster with seed customer } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{if customer } j \text{ is a seed} \\ 0 & \text{otherwise} \end{cases}$$

parameters:

c_{ij} : distance between customer i and j

q_i : drop-frequency of customer i

Q : the capacity of clusters in terms of drop-frequency

K : number of vehicles

Accordingly, the CCP can be described by the following integer programming formulation:

$$min \sum_i \sum_j c_{ij} x_{ij}$$

$$s.t. \sum_i q_i x_{ij} \leqslant Q \qquad \forall j \in S \tag{1}$$

$$\sum_j x_{ij} = 1 \qquad \forall i \in C \tag{2}$$

$$x_{ij} \leqslant y_j \qquad \forall i \in C, \forall j \in S \tag{3}$$

$$\sum_j y_j = K \tag{4}$$

$$x_{ij}, y_j \in \{0, 1\} \qquad \forall i \in C, \forall j \in S \tag{5}$$

Note that for the problem presented in this paper, each customer can be a seed customer and therefore, the set C equals set S. Moreover, because of the different context, two small adjustments have been made in the above formulation to the problem that was presented by Koskosidis and Powell. The first change is in the objective, which previ-

ously minimized total costs, which consisted of the variable transportation costs. Since it is assumed that costs are proportional to distance, in this formulation the total distance is minimized instead. The second change is made to account for both time constraints and the infrequent customers and because the volume constraint imposed in the original problem is unnecessary. Where the parameters q_i and Q were defined by the authors as the shipment size of customer i and the vehicle volume capacity respectively, these are now defined in terms of drop-frequency. Consequently, for each driver the maximum average number of stops can be restricted. As such, a different purpose of the constraint is implied, namely constraining workload instead of volume. The capacity of the clusters will be constrained to Q for all clusters. This constraining method will be evaluated as will be explained in Section 5.5. Furthermore, the second set of constraints ensures that each customer is assigned to only one seed customer and the third set of constraints ensures that no customer can be assigned to a customer that is not selected as a seed. Finally, the fourth constraint ensures that the total number of seed customers is equal to the total number of vehicles. The CCP is NP-complete and it has been shown that exact algorithms are inefficient for large problems (Koskosidis and Powell, 1992).

Therefore, the K&P algorithm was presented to solve the CCP. This algorithm solves the problem in three iterative steps, namely creating an initial solution based on the regret function for a given set of seed customers, then recalculating the seed customers of each cluster and finally making local exchanges in order to improve the solution. The problem starts with an initial set of seed customers and returns to the previous step whenever a change to the solution is made. That is, if one or more new seeds have been found in the second step, it returns to the first step with the new set of seed customers. Also, whenever improvements are made in the third step, it returns to step 2 and if necessary to step 1. The pseudocode for this, can be found in Algorithm 1. Note that this pseudocode is based on the one described in the paper in which the algorithm is introduced (Koskosidis and Powell, 1992). However, the initialization phase and the first step are slightly altered. In Koskosidis and Powell (1992), the initial set of seeds is found by means of the Pseudo-Knapsack Assignment algorithm. However, this is one of the extensions made to the algorithm of Mulvey and Beck (1984), though this algorithm already gave good results. Because the focus of this research is on balancing stability and costs rather than finding a cost-efficient solution and because this extension is rather time-consuming, the initial seeds will be selected randomly as was done by Mulvey and Beck. In addition, contrary to the proposed algorithm of Koskosidis and Powell, the number of clusters does not increase in step 1 of the algorithm, whenever the current number of clusters leads to an infeasible solution. Instead, the algorithm should label the problem as infeasible.

Moreover, the most important methodological aspects of the algorithm are the following:

- The regret function (see line 9) of a customer i is defined as the difference between the distance to the second and first closest seed. In the first step the customers are sorted based on a decreasing regret value and assigned to a cluster in that same order. However, it will also be tried to solve the assignment problem (step 1) by solving the corresponding integer programming problem exactly. That is, because the seed customers are now fixed, one can solve the previously specified formulation of the CCP without variable y_j, and as such, without constraint 3 and 4. In the case of solving the first step exactly, because the corresponding solution will be optimal, step 3 can be skipped. As such, the outerloop does not exist. Both approaches will be tried and compared.

- In step 2 new clusters seeds are found by simply choosing the customer in the cluster that minimizes the total distance to all other customers in the cluster.

- The local exchange method in step 3 only considers pairwise exchanges. An exchange is made when this leads to an improvement in the objective value. Each pair is only inspected once.

Moreover, the following parameters are defined within the algorithm. A proper value should be chosen for these.

- Epsilon (ϵ, see line 24) defines the minimum difference in objective value in step 2 necessary to repeat step 1.

- The *maximum iterations* is the maximum number of times the outerloop (see line 6) and innerloop (line 7) should be repeated.

In addition, as an extension to this algorithm for the purpose of enhancing the possibility to obtain stable clusters, a slightly different implementation is proposed. The main idea of this extension is to ignore the initialization phase of creating clusters (step 0) and instead, using the previous cluster seeds as a starting point. This extension can be compared to the original implementation based on the two performance measures.

5.3 Districting algorithm

Secondly, the Districting algorithm will be discussed. The goal of this algorithm is to divide an area belonging to one depot into K subareas, whereby all customers in one subarea are served by one driver and as such represent a cluster. The objective is to achieve

input : List of all customers to be clustered (*Customers*)
output: *K* clusters, with each customer assigned to one of these clusters

```
/* Step 0:  Initalization                                                    */
Seeds ← randomly select K customers from Customers to be cluster seeds ;
iterationsOuterloop ← 0 ;
iterationsInnerloop ← 0 ;
step1done ← false ;
step2done ← false ;
outerloop: while (step2done = false) and (iterationsOuterloop < max iterations) do
    innerloop: while (step1done = false) and (iterationsInnerloop < max iterations) do
        /* Step 1:  Greedy Assignment                                        */
        foreach customer (c ∈ Customers and c ∉ Seeds) do
            calculate regret ;
        end
        sortedCustomers ← sort Customers based on decreasing value of regret ;
        foreach customer ∈ sortedCustomers do
            add customer to feasible cluster with nearest cluster seed ;
        end
        step1done ← true ;
        /* Step 2:  Median Relocation                                        */
        foreach cluster do
            new seed ← find customer in cluster which would, if it was to be the
              cluster seed, result in the minimum objective ;
            if new seed ≠ current seed then
                replace cluster seed ;
            end
        end
        step2done ← true ;
        iterationsInnerloop ← iterationsInnerloop + 1 ;
        if (objective_old − objective_new > ϵ) then
            step1done ← false ;
        end
    end
    /* Step 3:  Local Exchanges                                              */
    foreach pair of clusters (k_1, k_2) do
        foreach pair of customers (r ∈ k_1, s ∈ k_2) do
            ΔC ← costs in terms of objective value of exchanging customer r and s ;
            if (ΔC < 0) and (exchanging r and s is feasible) then
                exchange r and s ;
            end
        end
    end
    iterationsOuterloop ← iterationsOuterloop + 1 ;
    if any local exchanges are made then
        step2done ← false ;
    end
end
```

Algorithm 1: Pseudocode for K&P algorithm

this at minimum "costs", whereby Wong and Beasley (1984) base these costs on the number of times two customers were on the same route relative to the number of times they were not, calculated over a certain time period of length T. To clarify this, let f_i be the number of times a delivery occurs for customer i in the time period and a_{ij} the number of times customer i and j where on the same route within this time period. The associated costs b_{ij} of having customer i and j in the same district are then defined as follow:

$$b_{ij} = \min(f_i, f_j) - a_{ij} \tag{6}$$

Note that b_{ij} is always non-negative. Now the objective function of the algorithm is the following minimization function:

$$\min \sum_{k}^{K} \sum_{i \in A_k} \sum_{j \in A_k} b_{ij} \tag{7}$$

Where K is the total number of subareas (clusters) and A_k is the set of all customers that are in cluster k. Because the Districting Problem focuses on finding clusters for a longer period of time, it puts a constraint on the workload of a cluster at every day in a period of length T. This workload is expressed as the sum over the order quantities of all customers in a cluster on a day t. That is, if s_{it} presents the order quantity of customer i on day t and S is the workload capacity of a vehicle in terms of the quantity ordered, then the following constraints should hold:

$$\sum_{i \in A_k} s_{it} \leqslant S \qquad \forall k, t \tag{8}$$

In the context of the posted problem, it is preferred to express this constraint in terms of drop-frequency q_i. Note that the two are closely related, as the drop-frequency can be calculated from information about the order quantity over a period of time. As such the corresponding constraint will look as follows:

$$\sum_{i \in A_k} q_i \leqslant Q \qquad \forall k \tag{9}$$

where

$$q_i = \frac{\sum_{t=1}^{T} \mathbb{1}_{s_{it}>0}}{T} \tag{10}$$

and Q the workload capacity of a vehicle in terms of number of drops. Note that the clusters are less constrained than previously, as this no longer needs to hold on a daily basis. However, the drop-frequency will be calculated over 4 weeks, regardless of the value of T. This constraining method will be further explained in Section 5.5.

The general idea of the algorithm as proposed is to generate an initial solution by means of the values b_{ij} and improving this by making interchanges. The corresponding pseudocode can be found in Algorithm 2, which is based for a large part on the one of Wong and Beasley. However, they handle the customers with zero demand in the period T differently. Because the value of b_{ij} for these agents will be zero for all j, this would give a misleading result. Whereas they solve this issue by simply adding these customers manually, we have incorporated this in the algorithm. At the end of the second step zero-customers are then added to the same cluster as their nearest customer, regardless of capacity constraint. The third step can then still turn a possibly infeasible solution into a feasible one. In addition, similar to the K&P algorithm, an infeasible solution is handled differently. Wong and Beasley suggested that if an agent cannot be added to any cluster in the assignment phase, the workload constraint can be violated. However, in this research, violated constraints are not allowed in the final solution and therefore, the program should just indicate the problem to be infeasible. The most important aspects of the algorithm will be highlighted:

- As a start, b_{ij} should be calculated by solving the VRP problem over a time frame of length T. The VRP aims to find routes to serve all demand points starting and ending at the depot. The objective of this problem is to minimize the total distance, given some capacity constraints. Note that the capacity constraints in this daily problem should no longer be in terms of drop-frequency, but can be set on the number of agents on the route (which is at maximum 30). A commonly used method for solving the VRP, is the method of Clarke and Wright (1964). Their method distinguishes between a sequential and a parallel method that could be used. The difference between the two is that in the sequential method, contrary to the parallel method, it is only allowed to build one route at the time. The parallel method is found to often provide better results (Lysgaard, 1997). Moreover, the parallel method generally tends to have larger and thus less routes (Serrels, 2015). This also turned out to be the case for this problem when testing both methods on multiple days. For

these reasons, the parallel method, instead of the sequential method was used in the algorithm. The parallel method of Clarke and Wright is based on the principle of combining two routes by means of taking out the ending depot of the first route and the starting depot of the second route and connecting the routes here instead. Combinations are made only if this leads to overall (positive) savings. Savings are defined to be the difference in total distance between serving both routes separately, starting and ending at the depot and connecting the two routes, starting and ending at the depot. Adding two routes is done in order of decreasing saving, which is constantly updated (Serrels, 2015).

- In the initialization phase the K seed customers are chosen by selecting customers that have as few routes as possible in common with the already chosen seed customers balanced with being as far away from the depot as possible. Though the authors do not mention specifically the reason for the latter requirement, the reason for this could be to get more distinct clusters, as customers around the depot can more easily be on the same route. The *initalization criteria* (see line 10) is therefore defined as selecting customer i that produces the maximum to the following:

$$\max \left[M \min(b_{ij} | j \in I) + d_{0i} \prod_{j \in S} d_{ij} | i \notin I, i = 1, ..., n \right] \tag{11}$$

  Here, M is a very large number, d_{0i} is the distance from customer i to the depot and d_{ij} the distance between customer i and j. Note that set I is the set of all customers.

- Customers are added to subareas in a decreasing order of importance, which is based on some sort of criteria. This paper will consider the drop-frequency to be this criteria: the higher the drop-frequency, the more important the agent. A customer i is added to the subarea k that generates the lowest increase in costs, which is $\sum_{j \in A_k} b_{ij}$, given that the previously explained workload constraint still holds.

- Interchanges are made after allocating all customers to a cluster, by both moving a customer to a different subarea as well as swapping two customers of two different subareas. These changes are only made when an improvement is made to the total set of subareas in terms of a decrease in objective value or a decrease in total violation of capacity constraints (see line 18 to line 35). The latter one refers to the total value above Q for all clusters, in Equation 9. Each customer and each pair of customers is only inspected once.

Note that the length of the period over which the algorithm learns (T) is the only param-

eter to be set in the algorithm.

5.4 Differences between the two methods

To summarize, there are three important differences between the K&P algorithm and the Districting algorithm. First of all, as mentioned before, the problem they are meant to solve are fundamentally different in the timeframe for which they cluster the agents. The CCP has risen from the problem of clustering customers on a certain day with known demand. The Districting Problem on the other hand is the problem of dividing a set of customers in districts for a longer period of time, implying that demand is not yet known. Consequently, the objectives of the problems are very different. The second difference has to do with the previously mentioned idea that clustering like roses would be more stable than clustering like a daisy. The difference between the two was the exclusion and incorporation of the distance from and to the depot respectively. As mentioned before, this is also an important difference between the K&P algorithm and the Districting algorithm. As has been seen in the detailed description of the algorithm, the K&P algorithm just focuses on minimizing the distance from agents to their cluster seed, completely ignoring the distance to the depot. The Districting algorithm on the other hand, minimizes the sum over all b_{ij} per cluster, which value is based on the solution to the VRP, which does consider a depot. The comparison between the two algorithms should thus result in an answer to the subquestion of the first research question about possible cluster shapes. Lastly, the fact that the depot is incorporated in the Districting algorithm, leads to the requirement that all agents should be assigned to a depot *before* clustering. Likewise, the number of vehicles to be assigned to each depot needs to be fixed in advance. The K&P algorithm is more flexible in this sense.

5.5 Constraining the cluster capacity

As has previously been seen, both methods will be implemented such that they restrain the cluster size in terms of drop-frequency rather than time, volume or weight. This means the number of drops per day and thus indirectly the workload are restricted. This constraint is implemented such that daily routes are feasible, by taking into account the limited working time and the infrequent customers. The maximum workload in terms of drop-frequency is set to 24 as a starting point. This value is set higher than the average number of stops a day, because it is an upper bound and it is set lower than the maximum number of stops a day, because it considers an average and not a maximum. However, as demand is unknown at the moment of clustering, no guarantee can be given that daily

input : List of all customers to be clustered (*Customers*)
output: k clusters, with each customer assigned to one of these clusters

```
zeroCustomers ← put all customers with zero demand in the past T days in a separate list ;
/* Step 0: find b_ij                                                    */
for past T days do
    solve VRP problem by means of the algorithm of Clarke and Wright;
end
foreach customer i do
    foreach customer j do
        determine b_ij ;
    end
end
/* Step 1: Initialization                                               */
choose K customers as starting points for the clusters according to initialization criteria ;
/* Step 2: Customer assignment                                          */
sortedCustomers ← sort customers based on importance criteria ;
foreach customer ∈ sortedCustomers do
    add customer to feasible cluster resulting in lowest objective value ;
end
foreach customer ∈ zeroCustomers do
    add customer to the same cluster as its nearest customer ;
end
/* Step 3: Local exchanges                                              */
foreach customer i ∈ sortedCustomers do
    foreach cluster k with i ∉ A_k do
        if moving customer i to cluster k improves objective and cluster k remains feasible then
            move customer i to cluster k ;
        end
        if moving customer i to cluster k decreases total violation of capacity constraints then
            move customer i to cluster k ;
        end
    end
    foreach customer j ∈ sortedCustomers, with j and i not in the same cluster do
        if Switching customer i and customer j improves objective and both clusters remain feasible then
            switch customer i and customer j ;
        end
        if Switching customer i and j decreases total violation of capacity constraints then
            switch customer i and customer j ;
        end
    end
end
```

Algorithm 2: Pseudocode of Districting algorithm

routes are also feasible in reality. As such, this constraining method should be evaluated. This can be done by finding the actual number of stops on a daily basis after applying the clustering algorithms. As was previously mentioned, according to available information from the company, a maximum of 30 agents can be visited daily by one driver. As such it is of interest how many days a driver has to visit more than 30 stops, how many "extra" stops are made on average in such a case and what the average number of stops is in a cluster. A sensitivity analysis would be helpful to evaluate the effect of the chosen value for this maximum cluster capacity on cluster stability and transportation costs. For this purpose, this value will be varied between 20 and 28. In addition, this sensitivity analysis can be used to make a suggestion about the most proper value to constrain the workload for the company.

5.6 Sensitivity analysis on growth strategies

Another sensitivity analysis can be used to evaluate the effects of where new agents are located on the algorithm performance, in terms of cluster stability and transportations costs. For this reason, the generation of coordinates for new agents should be altered. Recall that as a default, they are generated randomly within a squared area that is 20% bigger than the starting area. This corresponds to the real expansion of the set of agents. Generating these new agents should be done in two other ways:

- Strictly generating coordinates within the starting area
- Strictly generating coordinates outside the starting area, but within the larger area that was specified before.

This last analysis will reveal the effects of different strategies to find new agents for the company and as such lead to answers for the second research question.

6 Program implementation

6.1 Program development

The full program of running the first 144 days starting on January 1st 2017, updating the set of agents, clustering by means of some clustering algorithm and running a daily TSP has been implemented in Java. Note that the following clustering algorithms (and their extensions), all discussed in Section 5, have been implemented:

1. K&P algorithm with regret method to assign customers to clusters
2. K&P algorithm with exact assignment of customers to clusters
3. K&P algorithm without initialization phase, but instead keeping previous cluster seeds
4. Districting algorithm

The input-files and output-file of the program are Excel-files. In case the program is ran with the real data, the program takes in three files, namely a file listing all the agents at the beginning of the period, a file listing all the *future agents* that arrive throughout the period and a file of the drops of all these agents in this period. For the agents that are there from the start of the period, the drops from the four weeks before this start are also uploaded to use for calculating the drop-frequency of these agents at the start of the period. The starting list of agents consists of 912 agents, which are all the agents with a join date before January 1st 2017 and a leaving date after January 1st 2017. The list of future agents contains all other agents that had an order in the first 24 weeks of 2017, which were 1011 agents. In total, 67,533 drops correspond to these 1923 agents in this period of time. In case the program is ran with simulated data, only the first file listing the agents at the start of the period is required as input. For calculating the drop-frequency at this start, the demand over the four weeks before the start are simulated. Furthermore, the program makes use of CPLEX to solve the TSP, the Best Driver Problem and the exact optimal assignment of agents to clusters in step 1 of the K&P algorithm (if the regret method is not used). A very large part of the individual functions within the program are tested with the help of unit tests. In addition, the full program has been tested on a test set of three clearly separated clusters of 10 agents. Running the program takes between 2.5 and 5.5 hours depending on the settings and the clustering method that is used. The most important aspects of the program are listed below:

- As the first working day of 2017 is a Monday, the program starts on a Monday. The first clustering problem is solved on day 1 and from then onwards repeated every two weeks, implying that it is also solved on day 13, 25, 37 and so on. In addition, it is solved once more after passing all 144 days. Clustering always takes place before the start of the day. In total, 13 clustering runs are performed within the period.
- When simulating the data, agents are added and removed on a daily basis. Agents to be removed are picked at random. When using the real data, at the start of every week, just before the clustering run, new agents with their first demand in the

upcoming week are added (assuming new agents become known more or less one week in advance). Agents are removed at the start of every four weeks if they did not have demand in the past four weeks and if they will not have any demand in the future. This corresponds to the actual policy of the company which states agents to be inactive when not having had orders for the past month.

- Whenever a new agent is added, this agent is assigned to the nearest cluster, regardless of capacity constraints. For the K&P algorithm this implies it is added to the cluster with the nearest seed. For the Districting algorithm this implies it is added to the same cluster as its nearest agent.

- The traveling salesman problem solves the problem of finding a tour through all agents with a demand on a specific day, while starting and ending at the depot. The traveling salesman problem is implemented according to the MTZ formulation (Laporte, 1992). Though other formulations are also possible, the MTZ formulation was easiest to implement and therefore chosen. As it turns out, the model sometimes has problems proving that the found optimum really is an optimum to the problem and can therefore take a very long time. To reduce computation time, it was tested if (approximately) the same result could be obtained when restraining computation time. This was tested by trying to solve 10 possible TSP problems, within 10, 20, 30, 40, 50, 70 and 100 seconds. As turned out, in 7 out of 10 cases, the same solution was found for all time limits. In 2 cases, the same solution was found when running at least 20 seconds, which lead only to a decrease of 0.6% and 0.2% compared to the solution found after 10 seconds. The last case, found the same solution for running 70 seconds or more, which lead to a decrease of 3.5% compared to a run of 10 seconds. As time reduction is important, it was concluded that it sufficed to put the time limit for solving the TSP to 10 seconds.

- The depot division is created resulting in a more or less equal division in terms of number of agents between the two depots. However, the average drop-frequency turns out to be higher in one of the areas and as such that area may require more vehicles. In the K&P algorithm it is possible that agents are assigned to different depots within one cluster. In order to calculate the total costs per day of a cluster, a depot is assigned to the cluster by the majority rule, based on all agents in the cluster (not just the ones with demand on that day). In the Districting algorithm, the clustering problem is solved per depot, as it takes into account the depot in the algorithm. Consequently, each cluster consists of only agents linked to the same depot. In this case, each depot receives an equal amount of vehicles and in case of

an uneven amount, the depot with the higher average drop-frequency receives one more. Note that it is assumed that all drivers can work from both depots.

- The program is able to solve the clustering problem with a fixed number of vehicles or with an increasing number of vehicles. In the latter one, the program adds a vehicle to account for growth that would otherwise result in an infeasible solution. As a starting point, the number of vehicles is set equal to the minimum number of vehicles necessary to satisfy capacity constraints. This minimum is the ceil of the entire workload (that is the total sum of drop-frequencies over all agents), divided by the maximum workload of a truck (in terms of drop-frequency). The latter value corresponds to the value of the cluster capacity as was specified in Section 5.5.

- A distance correction factor of 1.9 is used to be multiplied with the great circle distance. This means that it is assumed that on average the actual distance of a route between two points based on Google Maps is 1.9 times the great circle distance between the two points. This result was based upon the distances between 6 different points (15 distances in total), which were picked at random and well spread over the dataset. The great circle distances from these pairs ranged from 3 to 90 km.

The details of the implementation can be found in Appendix E. For the Districting algorithm it is important to mention that to implement the algorithm of Clarke and Wright which solves the daily VRP problem for the past T days, a code taken from a report from the Edinburgh Napier University was used (Serrels, 2015). In the report, this code was extensively described and explained. The method was tested in order to verify if it indeed worked. Recall that the parallel method, not the sequential method, was implemented (see Section 5.3).

Moreover, various parameters had to be defined within the two algorithms. For the K&P algorithm, this involved setting the number of iterations of the algorithm and the stopping criterion epsilon for step 2. For the Districting algorithm, the length of the time frame (T), over which the algorithm calculates the optimal VRP, had to be fixed. The following sections will describe the decision making for these parameters in detail.

7 Tuning the K&P algorithm

For choosing the value of epsilon in the K&P algorithm, simple reasoning is used. This epsilon is used in step 2, the recalculation of the cluster's seed (see line 24 in Algorithm 1). It defines the minimum difference between the previous objective and the new objective

after recalculation, such that step 1 should be repeated. As the objective is defined as the sum of distances from each agents to its assigned cluster seed, it seems reasonable that one does no longer care about differences which are smaller than one kilometer. As such, epsilon is set equal to 1.0.

The setting of the maximum number of iterations directly influences the running time of the algorithm. Therefore, it was checked what the influence is on the objective of the K&P algorithm when varying the maximum number of iterations between 10, 20, 30, 40, 50 and 100 iterations. The corresponding graph with the average objective of each clustering run (13 clustering runs for each cluster period, which is 2 weeks, in total) for each of these values of the parameter is shown below (Figure 3). Note that these are obtained using the K&P algorithm with the regret method applied to real data. The value of the maximum number of iterations is the same for both the innerloop (step 1 and 2, see line 7) and the outerloop (step 3, see line 6) in Algorithm 1. Evidently, there is no clear relationship between this parameter and the objective value. Note that the difference between the maximum and the minimum objective value is small, with a maximum difference of 1.71%. This seems to imply that similar results could be obtained with a small number of iterations and that the difference in objective value simply has to do with randomness in the simulation. Indeed, looking closer into the objective value at each iteration, it can be seen that the inner loop usually only requires about 5 iterations before the stopping criteria is reached (see line 24). For the outer loop, by looking at the objective value at every iteration for 100 iterations, it was found that the algorithm usually terminates before 20 iterations because no improvement could be made and otherwise continues until the maximum number of iterations has been reached (see first row of Table 1). However, in the latter case, the objective often just alternates between two values or just remains constant. This is probably due to the fact that the regret function does not necessarily find the optimal assignment and because of this, local switches will always be made. In addition, it is possible that the same switches between seeds or between agents are made at each iteration. In such cases, usually the final solution has been found after less than 20 iterations (see second row of Table 1). Therefore, as these extra iterations do not seem to lead to any improvement and a reduction in computation time is favored, the maximum number of iterations is set equal to 20.

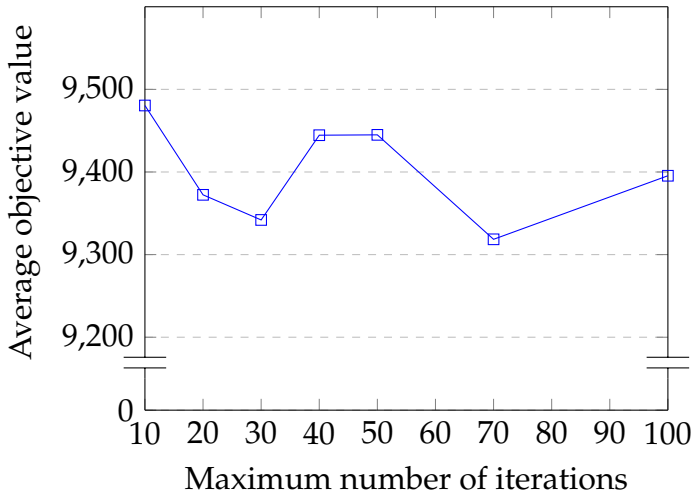

Figure 3: Evaluation of the influence of the maximum number of iterations in the K&P algorithm (with regret method) on the average objective value (applied on real data, with increasing number of clusters).

Day	1	13	25	37	49	61	73	85	97	109	121	133	after day 144
Number of iterations performed	100	13	5	100	23	100	100	13	100	100	19	100	100
Number of iterations to solution	100	12	4	3	23	1	6	12	7	100	18	2	15

Table 1: Number of iterations of the outerloop before terminating versus the number of iterations of the outerloop before finding the best solution when performing at maximum 100 iterations of the K&P algorithm (with regret method, performed on real data, with increasing number of clusters).

In the case of using an exact assignment of agents to clusters, the outer loop does not exist because assignment is already optimal (see Section 5.2). In this case, the inner loop needs at most 20 iterations to find the minimum objective value. Since this was possible in terms of time constraints, the maximum number of iterations for the K&P algorithm with exact assignment was also set to 20.

8 Tuning the Districting algorithm

For the Districting algorithm, a decision for the value of T, the length of the time period over which the algorithm learns, needs to be made. Obviously, the value of T directly impacts the computation time. Because the drop-frequency is calculated over 24 days, it makes sense to set T to at most 24. As such, the values 6, 12, 18 and 24 are tried for T, corresponding to 1, 2, 3 and 4 weeks respectively. For each value of T the average objective value over all 13 clustering runs is found. Recall that in the Districting algorithm, this objective value is expressed as the sum of b_{ij} over all agent pairs in a cluster (see Equation 7). Because b_{ij} is defined as the (absolute) difference between the number of times agent i and j were in the same cluster and the number of times they were not (see Equation 6), on

average, this number is expected to increase proportionally to T. As a consequence, for the sake of comparison the objective values should be divided by $\frac{T}{6}$, such that the objective value is shown for the same length of time, namely a week. The results can be found in Figure 4. Note that these are obtained using the Districting algorithm applied to real data. From the graph you can see a clear increase in objective value when increasing T from 6 to 12, which is most probably due to the fact that agents which have no demand in period T are left out of the objective value. For $T = 6$ this means considerably less agents are included in this value. Considering time constraints and because the objective value does not differ much between the values of T larger than 12, it should suffice to use $T = 12$ for the Districting algorithm.

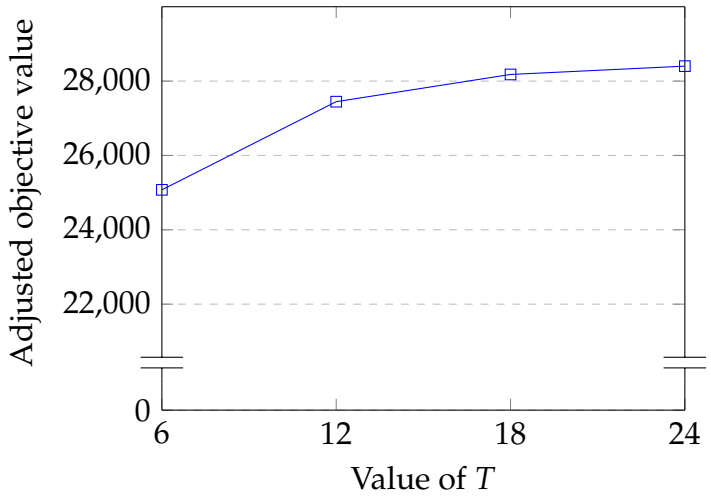

Figure 4: Evaluation of the influence of the value for T in the Districting algorithm on the adjusted objective value (applied to real data, with increasing number of clusters). This is the objective value as defined by Equation 7 and adjusted by dividing it by $\frac{T}{6}$.

9 Results

9.1 Algorithm performance

Finally, both algorithms are tuned and can be applied on real and simulated data, such that they can be evaluated in terms of the performance measures of transportation costs and cluster stability. Note that these performance measures are different from the objective functions that are attempted to be minimized by the algorithms. The results of running all 144 days for both the K&P algorithm (denoted with KP) and the Districting algorithm (denoted with DA) on these data are shown in Table 2. Both the results of fixing the number of clusters to 25 (KP25 and DA25) and letting the number of clusters increase whenever necessary (KPincr and DAincr) are displayed in this table. The results contain

the average cluster stability over all 13 cluster runs and the average daily distance, average number of trucks and average daily (transportation) costs in Kenyan Shillings over all 144 days. To give an idea on the feasibility of the routes, the average percentage of daily routes with more than 30 agents (the potential maximum) is also shown. Note that the set of agents increased much more in reality than was predicted in this paper. Hence a decrease is visible in the number of trucks (when the number is not fixed) in the results of the simulation. However, since the daily distance increased regardless in the results of the simulation, it seems that the simulated locations of the agents are more spread out than is the case in reality. This has especially an impact on the overall costs in the case of a fixed number of trucks, where lower daily costs were obtained, compared to the daily costs when the methods were applied on real data.

	Real data				Simulation			
Averages	KP25	DA25	KPincr	DAincr	KP25	DA25	KPincr	DAincr
Cluster stability (%)	69.4	38.8	71.3	40.4	65.8	41.0	72.9	46.9
Daily distance (km)	5443	7805	4732	6971	6390	8121	4887	7015
Number trucks	25.0	25.0	18.6	23.0	25.0	25.0	14.7	16.3
Daily costs (KES)	232,011	263,266	181,539	239,432	244,541	267,437	158,525	197,346
Routes with >30 customers (%)	2.6	0.9	7.4	5.5	0.4	0.0	2.8	1.5

Table 2: Algorithm performance in terms of average cluster stability in percentages, average daily distance in kilometers, average number of trucks per day and average daily (transportation) costs in Kenyan Shillings. Results are obtained from running the program on 144 days including 13 clustering runs.

Clearly, the K&P algorithm gives a much better cluster stability and also much lower costs in all cases. Only the feasibility in terms of the percentage of routes with more than 30 customers is better for the Districting Algorithm. Averaging over all four cases, the average daily costs resulting from the Districting algorithm are approximately 20% higher and the cluster stability approximately 40% lower. There are multiple reasons possible for this difference. First of all, the difference in costs is partially due to fact that agents are strictly assigned to one of the depots in the Districting algorithm, contrary to the flexible assignment in the K&P algorithm. This leads for instance to more clusters and thus more vehicles, in the case of the Districting Algorithm. The same reasoning can be applied to explain the lower percentage of routes with more than 30 customers when using this method. When forcing a depot division in the K&P algorithm, the costs also rise with 5.1% on average. Secondly, the obtained clusters from the Districting algorithm have a lot of overlap. This can be seen clearly when comparing the visualization of the clustering solution obtained at day 1, which is displayed in Figure 5 and 6. Note that these solutions are obtained based on real data. Apparently, the Districting algorithm does not work so well on this dataset. This may have to do with the relatively low drop-frequencies of the

agents. Remember that the Districting algorithm attempts to minimize $\sum_k^K \sum_{i \in A_k} \sum_{j \in A_k} b_{ij}$, with $b_{ij} = \min(f_i, f_j) - a_{ij}$, where $\min(f_i, f_j)$ is the minimum of the number of times agent i and j had a demand and a_{ij} is the number of times agent i and j were on the same route. As a direct consequence of the low drop-frequencies, $\min(f_i, f_j)$ will be low and therefore b_{ij}, which makes it harder to distinguish between pairs that should be in one cluster and pairs where one agent just had very few drops. In addition to this, the low drop-frequencies lead to high variability in the daily VRP problems, which leads to low values of a_{ij} if agents often have drops on different days, even if it makes sense to have them in the same cluster. Altogether, this leads to the conclusion that the Districting algorithm is not an appropriate algorithm to use in the case of infrequent demand. Lastly, the worse performance of the Districting algorithm may have to do with the fact that the clusters obtained by the algorithm are not as compact, as can also be seen from Figure 5 and 6. Instead, its clusters are very stretched. This confirms the idea that incorporating the distance to and from the depot in the algorithm, indeed leads to more spread-out clusters. Also, the clusters in Figure 6 generally have long stretched shapes; similar to the shape of the leaves of the daisy. At the same time, the rose shape is visible in the solution of the K&P algorithm, where clusters are quite compact. This finding thus seems to confirm the idea that including or excluding the depot from the calculations influences the shape of the clusters. Consequently, as far as these empirical results can prove the point, the results in Table 2 show that roses-shaped clusters perform much better, in terms of both cluster stability and costs, in the situation of the company.

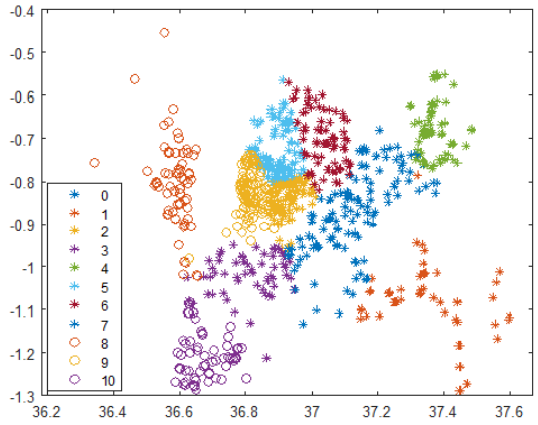

Figure 5: Clustering solution at day 1 of K&P algorithm (with regret method) with increasing number of clusters on real data. This figure shows 923 agents assigned to 11 different clusters. The horizontal axis shows the longitude coordinates, whereas the vertical axis shows the latitude coordinates.

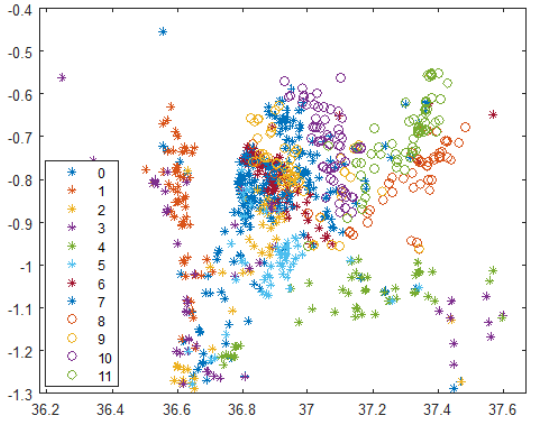

Figure 6: Clustering solution at day 1 of Districting algorithm with increasing number of clusters on real data. This figure shows 923 agents assigned to 12 different clusters. The horizontal axis shows the longitude coordinates, whereas the vertical axis shows the latitude coordinates.

Another important remark to be made on Table 2 is that it is perhaps surprising that the cluster stability is lower when keeping the number of clusters constant over time instead of starting at the minimum and increasing this number whenever necessary. This has most probably to do with the fact that fixing the number of vehicles, requires more vehicles (clusters) overall to account for growth. Therefore, it is less likely that agents are assigned to the same driver. Moreover, when having more clusters than necessary by fixing the number, the clusters may become less 'natural'. That is, some clusters may be forced to split when this is not necessary. Especially because cluster seeds are picked at random in the K&P algorithm, this could lead to more instable clusters. Note that, clearly, increasing the number of clusters is also preferred over fixing it in terms of costs.

Finally, it should be mentioned that when using the exact assignment in the K&P algorithm, rather than assigning agents to clusters based on the regret function, on average over the four different cases, the cluster stability went down with 4.79% and the costs went down with 1.04%. This implies, that apparently finding the optimal solution does not foster the cluster stability, while it only leads to a small cost reduction. Based on the above results it can be concluded that the K&P algorithm with the regret method seems to be the most appropriate method for this specific problem. Therefore, for obtaining the remaining results, we will continue using this method.

9.2 Evaluation of the capacity constraints

According to the problem description, daily routes should not contain more than 30 agents. As such, in the proposed solution method the clusters are constrained in terms of drop-frequency. It is of interest how well this works and what a proper maximum for the cluster capacity should be. As a default, the sum of the drop-frequencies of all agents in a cluster is constrained to 24. Based on real data, this led to 7.4% of all daily routes to have more than 30 agents. On average, in case of exceeding this number, this led to 3.2 extra agents on those daily routes. In this section the effects of changing the maximum cluster capacity is evaluated. For this purpose, a maximum cluster capacity of 20 to 28 is tried for the case of real data.

The effects of varying the cluster capacity on the percentage of daily routes with more than 30 agents and on the average number of extra agents is shown in Figure 7 and 8 respectively. Note that *extra* agents refers to the number of agents above 30 on the daily routes, only in the case that a route has more than 30 agents. Figure 7 shows a increasing growth in the percentage of routes with more than 30 agents when increasing the cluster capacity. Everything below and including 23 results in a percentage that is less than or around 5%. For a maximum cluster capacity of 24, this percentage is still only 7.4%. In

addition it can be seen from Figure 8 that for maximum cluster capacities below 25 the average number of extra agents on the daily routes will at most be around 3. From this it can be concluded that in case one would really like to be on the safe side, the cluster capacity should be constrained to no higher than 23, since this leads to approximately 5% of the daily routes having more than 30 agents and in such a case, on average, no more than 3 extra agents on top of the maximum of 30.

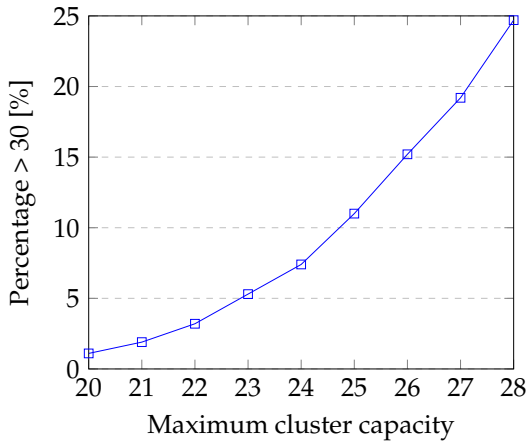

Figure 7: Effect of alternating the maximum cluster capacity on the percentage of daily routes that has more than 30 agents.

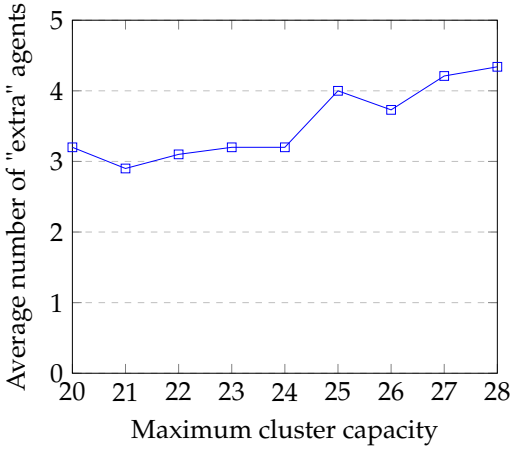

Figure 8: Effect of alternating the maximum cluster capacity on the average number of extra agents, where extra agents are the number of agents above 30 on the daily routes (only counting the cases that there are more than 30 agents).

Moreover, the effects of this constraint on the evaluation measures in this research are of importance. Figure 9 and 10 show the effects of varying the cluster capacity on the average cluster stability and the average daily costs respectively. A decrease in costs is visible when increasing the cluster capacity, which is logical as it is rather expensive to insert an extra truck instead of letting one truck serve more agents. The total decrease in costs from the lowest to the largest cluster capacity is as much as 16%. In terms of costs, you would thus like your clusters to be as large as possible. However, it is interesting that Figure 9 also shows that it is not necessarily to the advantage of the cluster stability to have too large clusters, nor to have too small clusters. Though the larger cluster have a slightly higher cluster stability, there is no clear indication that there is a relation between the cluster stability and the maximum cluster capacity. The steep decrease in cluster stability for a maximum cluster capacity of 23 may have to do with one or two very bad observed cluster stabilities. This is possible because the initial seeds in the K&P algorithm are still generated at random.

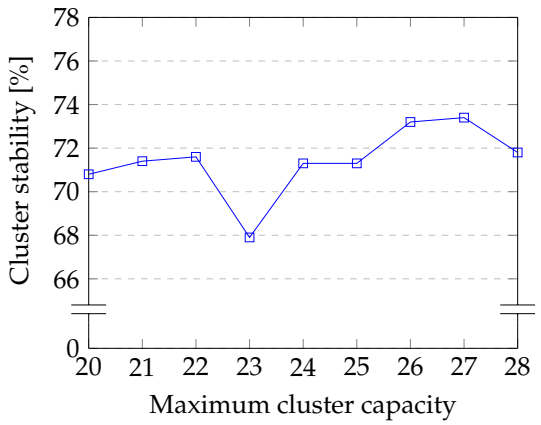

Figure 9: Effect of alternating the maximum cluster capacity on the average cluster stability.

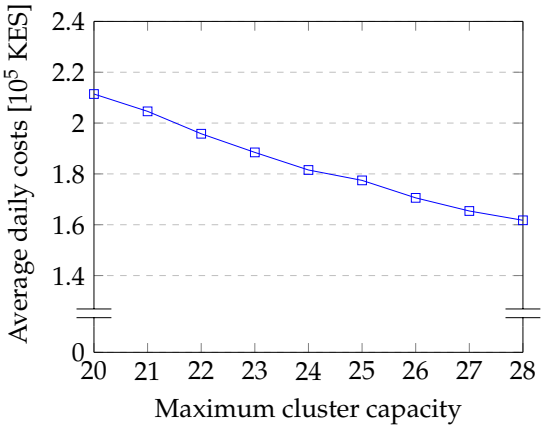

Figure 10: Effect of alternating the maximum cluster capacity on the average daily costs in Kenyan Shillings.

Obviously, the drop-frequency plays an important role in this specific analysis. For this reason, it is of interest what the effect is on the results of having less frequent or more frequent agents. Therefore, the exact same analysis is done for different average drop-frequencies, which is input for the simulation. This average drop-frequency is set to 0.35 by default, and is decreased and increased with 50% leading to simulating with a drop-frequency of 0.175 and 0.525. Whereas 0.35 corresponded to ordering more or less twice a week, these new values correspond to ordering more or less once a week versus three times a week. This is compared to the previously discussed results obtained from the real data. The most important findings are summarized in Table 3. The table shows the difference between the minimum and maximum of both costs and cluster stability. In addition, it shows the maximum cluster capacity corresponding to at most 5% of the daily routes having more than 30 customers. Lastly, information about a cluster capacity of 24 was shown, as this cluster capacity was used in the results in Section 9.1.

average drop-frequency:	0.175	0.35	0.525	real data
min-max difference costs (%)	25	36	33	31
min-max difference cluster stability (%)	15	16	8	8
maximum CC with percentage above 30 <5%	24	25	25	22
For CC=24: percentage above 30 (%)	3.9	2.8	1.7	7.4
For CC=24: average extra customers	2	2.2	1.8	3.2

Table 3: Results of sensitivity analysis of varying the maximum cluster capacity applied to simulated data with a different average drop-frequency and applied to the real data. The min-max difference is the percentage difference between the minimum and maximum costs/cluster stability obtained from varying the cluster capacity from 20 to 28. The maximum CC (cluster capacity) with percentage above 30 is the highest cluster capacity which results in having more than 30 customers in at most 5% of the cases. The last two lines show the percentage above 30 and the average extra customers for a cluster capacity of 24.

From this table it can be seen that the results do not differ much when varying the drop-frequency. Relatively large cost differences are obtained in all four cases (between 25 and 36 %), while the differences in cluster stability is smaller (between 8 and 16 %). As before, a clear decrease in costs was visible for all four cases when increasing the cluster capacity. For the cluster stability, there was again no clear pattern to be found. This suggests to not focus too much on the stability, but rather on the percentage of routes that have more than 30 agents. With respect to this, in all cases, an increasing growth in this percentage was found when increasing the cluster capacity (just like in Figure 7). The table shows that a cluster capacity of at most 22 consistently leads to less than 5% of the routes having more than 30 agents. The results for the real data were worse than those of the simulated data, leading to believe that the drop-frequency of agents is more variable in reality than is simulated. However, even for the real data, a maximum cluster capacity of 24, led only to 7.4% of the routes having more than 30 agents.

Finally, it should be noted to keep in mind that results on the number of stops are also dependent on the performance of the clustering algorithms. If clusters turn out to have shorter daily routes, perhaps a driver could be able to handle more than 30 stops a day. Further research into this or evaluation of practical results could be useful.

9.3 Increasing cluster stability

In this extension of the K&P algorithm, the seeds initialization phase (Step 0, see line 1 in Algorithm 1) is partially ignored with the aim to enhance cluster stability. Instead, the seeds resulting from the previous cluster run are taken as starting points. As such, this first step is only needed whenever there is an increase in the number of clusters. Only an amount of seeds equal to the difference in the number of clusters then needs to be generated. Note that this does not imply, that the seed customers are constant over time, because the second step of median relocation (Step 2, see line 19) can still replace the seeds. In addition, to avoid that an inactive agent becomes a seed, we do allow seeds to change in between cluster runs. That is, whenever a leaving agent is the seed of the cluster, the seed is replaced by the nearest agent that is in the same cluster.

This extension was applied to both real data and simulated data. Their results can be found in Table 4 and 5 respectively. As it turns out, keeping the seeds leads to a significant improvement in cluster stability. In the case of simulated data, this improvement even went up to 16.5%. At the same time, it also leads to a small decrease in costs. Though this is surprising, it may have to do with simple luck in the initial random generation of cluster seeds. Finding proper starting points for the seeds in the first clustering run may as such be an interesting field of further research. Moreover, it cannot be excluded

that the decrease in costs may also be due to some problem specifics. However, it would be extremely hard to determine which aspects of the posted problem are causing this. Additional research into this topic would be necessary to gain more insight.

Averages	Normal	Keep seeds	Difference (%)
Cluster stability (%)	71.3	80.3	+12.6
Daily distance (km)	4732	4689	-0.9
Number trucks	18.6	18.6	0
Daily costs (KES)	181,539	180,968	-0.3

Table 4: Applied to real data: differences in results between the normal K&P algorithm and its extension where the previous seeds are kept at every cluster run (with regret method, with increasing number of clusters).

Averages	Normal	Keep seeds	Difference (%)
Cluster stability (%)	72.9	84.9	+16.5
Daily distance (km)	4887	4697	-3.9
Number trucks	14.7	13.8	-6.2
Daily costs (KES)	158,525	150,144	-5.3

Table 5: Applied to simulated data: differences in results between the normal K&P algorithm and its extension where the previous seeds are kept at every cluster run (with regret method, with increasing number of clusters).

9.4 Impact evaluation of possible growth strategies

For the company it is of interest to evaluate what the effect is of different growth strategies on the performance of the clustering. In the current simulation, on average 5.40 agents arrive and 3.12 agents leave on a daily basis. This results in an absolute growth of 2.28 agents (see Section 4.3). As was previously mentioned, the original area of the coordinates of agents at the start of 2017 was 20% smaller than the expansion area that was used in the simulation. As it turns out, new agents were mostly found in northern and western direction. A visualization of these areas can be found in Figure 11, where the orange square indicates the starting territory and the blue area *together* with the orange area the expansion area, which is also used in the simulation. Consequently, whereas the standard boundaries were [-1.33, -0.35] for the latitude and [36.17,37.60] for the longitude, it is assumed that the starting territory corresponds to the boundaries [-1.33, -0.45] for the latitude and [36.32, 37.60] for the longitude. This is approximately the case in reality. As such, the current simulation (expanding both within and outside the starting territory) will be compared with expanding strictly in the following two areas:

- Within the starting territory: [-1.33, -0.45] for latitude and [36.32, 37.60] for longitude. This corresponds to the orange area in Figure 11.
- Strictly outside the starting territory: so either [-0.45, -0.35] for latitude in combination with the longitude of the starting area, [36.17, 36.32] for longitude in combination with the latitude of the starting area or a combination of the two new intervals. This corresponds to the blue area in Figure 11.

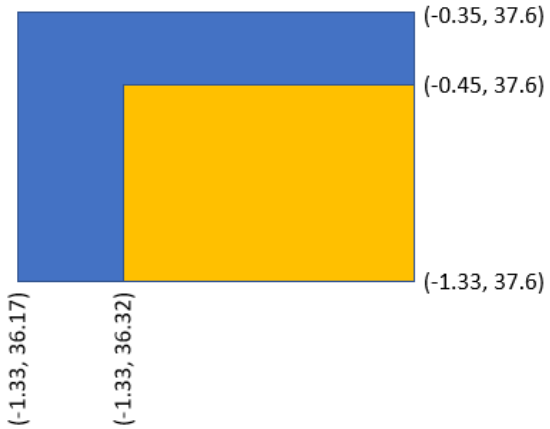

Figure 11: Visualization of starting territory (orange) and expansion area (blue)

Consequently, the sensitivity of the K&P algorithm for the boundaries of this area is evaluated and the difference in performance measures is displayed in percentages in Table 6. Note that this difference is compared to the standard expansion. This table shows that, as expected, the total costs decrease when new agents solely appear within the starting area. Also, perhaps unexpectedly, total costs decrease when new agents are located strictly outside of this area. This is solely related to the decrease in the number of trucks, as the daily distance actually increased. This decrease in number of trucks can be explained by randomness in simulating the drop-frequencies of new agents. Therefore, it would be better to measure costs by the distance and in that case an increase can actually be found. Most interesting, however, is that the cluster stability decreases considerably when expanding inside the starting territory. This may have to do with the high compactness of the set of agents when solely expanding inside, which results in more indifference of the algorithm between certain cluster solutions. That is, in a very dense area, more possible cluster solutions may give good results. When expanding into new areas, the overall compactness of the set of agents is less, what could result in a more natural cluster division. As such, it can be concluded that it would certainly be more cost efficient to find agents only within the boundaries of the current territory of the company. However, growth outside

this area or in both areas at the same time, may result in a higher cluster stability, which possibly weights out the larger distances (and thus costs).

	Difference (%)	
Averages	inside territory	outside territory
Cluster stability (%)	-3.2	-1.1
Daily distance (km)	-5.2	+1.2
Number trucks	-1.1	-1.7
Daily costs (KES)	-2.8	-0.5

Table 6: Performance differences of K&P algorithm for growth strictly within the starting territory versus strictly outside the starting territory, compared to the standard area (with regret method, applied to simulated data and an increasing number of clusters).

10 Conclusion

The Kenyan company discussed in this paper has four issues regarding finding a solution to their daily vehicle routing problem: they prefer fixed drivers to visit their agents, demand at agents is very infrequent and unknown until one day in advance and the set of agents is highly variable. This problem seemed yet unknown in VRP literature, as these issues are not often encountered in the Western world, let alone in this specific combination. The solution to this problem as presented in this paper is a two-phase approach to solve the VRP. That is, on a regular basis, the set of agents is divided into clusters and on a daily basis, the TSP is solved per cluster, only involving the agents with a demand occurrence. Two clustering algorithms are proposed, namely the K&P algorithm and the Districting algorithm. The main differences between the two, were the long term perspective of the Districting algorithm versus the short term perspective of the K&P algorithm and the incorporation (exclusion) of the distance to the depot into the Districting problem (K&P algorithm). Both were evaluated based on total transportation costs and the newly introduced measurement of cluster stability, which was defined as the percentage of agents that is served by the same driver as in the previous clustering solution. As it turned out, the Districting algorithm was found not to work well in a situation with such infrequent demand. The K&P algorithm on the other hand, led to better results both visually and numerically based on the two performance measurements. It gave an average cluster stability of about 70%. This led us to believe that ignoring the distance to the depot (clustering like roses), which was done in the K&P algorithm, is better than inclusion of this distance (clustering like a daisy) in terms of cluster stability. However, the impact of cluster shapes on transportation costs is hard to analyze, as other factors play a role.

Moreover, it has been shown that clusters can properly be constrained in terms of drop-frequency, while still requiring a maximum of 30 agents to be visited per day. Constraining the sum of drop-frequencies over all agents in a cluster to a maximum of 24 seems to be a suitable value, leading to only 7.4% of the routes having more than 30 agents, based on the real data. To be even more on the safe side, a maximum of 22 should be adopted.

In order to enhance the cluster stability further in the K&P algorithm, it is suggested to keep the encountered seed customers of the previous clustering solution as input for the next clustering run. When applied to the real data, this led to a cluster stability of more than 80%, while costs even decreased. As such, it is highly recommended to the company to implement this extension. In addition, we would suggest the company, in case of using clustering algorithms, to ensure growth in agents both within and outside the current territory. Whereas the growth within could improve costs, the growth outside the territory is necessary to ensure a higher cluster stability.

In conclusion, a proper two-phase method for solving the VRP in the context of the rural and peri-urban area of Kenya has been introduced. This method leads to both a reasonable cluster stability as well as cost efficiency. Tricks to enhance this cluster stability even further have been given. However, to gain more insight, it is recommended to look into the effect of the K&P algorithm or a different clustering method onto the clustering capacity in practice. In case of a large increase in efficiency due to using such algorithms, this may result in the possibility to widen the capacity constraints. Moreover, it may be helpful to investigate the reasons for costs to decrease when keeping the cluster seeds as a starting point for the next cluster run. Also, further research could be necessary to analyze the effect of randomly initializing the cluster seeds in the K&P algorithm, instead of picking them based on some criteria or logic.

References

Christofides, N. (1971). Fixed routes and areas for delivery operations. *International Journal of Physical Distribution*, 1(2):87–92.

Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Fisher, M. L. and Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124.

Koskosidis, Y. A. and Powell, W. B. (1992). Clustering algorithms for consolidation of

customer orders into vehicle shipments. *Transportation Research Part B: Methodological*, 26(5):365 – 379.

Kovacs, A. A., Golden, B. L., Hartl, R. F., and Parragh, S. N. (2014). Vehicle routing problems in which consistency considerations are important: A survey. *Networks*, 64(3):192–213.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operations Research*, 59:321–247.

Lei, H., Laporte, G., and Guo, B. (2012). Districting for routing with stochastic customers. *EURO Journal on Transportation and Logistics*, 1(1):67–85.

Lysgaard, J. (1997). Clarke & wright's savings algorithm. *Department of Management Science and Logistics, The Aarhus School of Business*, 44.

McGill, R., Tukey, J. W., and Larsen, W. A. (1978). Variations of box plots. *The American Statistician*, 32(1):12–16.

Mulvey, J. M. and Beck, M. P. (1984). Solving capacitated clustering problems. *European Journal of Operational Research*, 18(3):339 – 348.

Negreiros, M. and Palhano, A. (2006). The capacitated centred clustering problem. *Computers & Operations Research*, 33(6):1639 – 1663.

Serrels, S. (2015). Clarke-wright vehicle routing algorithm implementation report. Edinburgh Napier University, https://github.com/dooglz/Java-Clarke-Wright.

Spliet, R. and Dekker, R. (2016). The driver assignment vehicle routing problem. *Networks*, 68(3):212–223.

Wong, K. and Beasley, J. (1984). Vehicle routing using fixed delivery areas. *Omega*, 12(6):591 – 600.

Zhong, H., Hall, R. W., and Dessouky, M. (2007). Territory planning and vehicle dispatching with driver learning. *Transportation Science*, 41:74–89.

A Data cleaning

- Coordinates (Agent Lat and Agent Lon): The company has defined a range within which agents should be found. This range is defined by the interval [-1.33, -0.35] for the lat-coordinates and [36.17, 37.60] for the lon-coordinates. This implies that all coordinates outside this range are incorrect. These coordinates will be replaced when loading the agents into the simulation by a replacing coordinate that is drawn from a uniform distribution over the specified ranges. This is the case for 79 agents.

- Agent Status: If the status of an agent was set to "active", but the Agent Last Order Date was before the 1st of June 2017, the status of this agent was changed to "inactive". This is for the reason that according to the policy of the company, an agent that did not have any orders in a month becomes "inactive", but this change has not been implemented yet for these agents. This involved 125 agents.

- Agent Join Date: if the earliest order date of an agent is before the Agent Join Date (114 agents) or if the Agent Join Date fell in July 2017 (2 agents), this is regarded as a mistake. To solve the error, the Agent Join Date is set equal to the earliest order date for all such agents.

- Agent Last Order Date: In theory, the Agent Last Order Date should be equal to the latest order date of an agent. However, because of administrative reasons, it is possible that the latest order date is before the Agent Last Order Date. In these cases, it was suggested by the company to replace the Agent Last Order Date by the latest order date of the agent, which involved 54 agents. The opposite, that is, the Agent Last Order Date is before the latest order date did not occur.

- Agent Cluster: No action is taken in case an agent did not have a cluster. This was the case for 21 agents.

- Agent Leaving Date: This date did not yet exists, but is necessary for further evaluation of the data. It is assumed to exist only if the status of an agent is "inactive" and to be equal to the Agent Last Order Date.

- Order date: the orders of the cleaned set of agents was checked for order dates that fell on a Sunday, because the company does not deliver on this day. 20 orders took place on Sundays. If this was the case, the order was moved to the Monday after, which meant a change in order date (for 11 observations) or a higher order quantity if

this Monday already had an order (9 observations). This therefore let to a shrinkage of 9 in the total set of orders.

B Outlier detection

Outlier detection on full dataset:

- Agents with 1 or 2 drops and whose Last Order Date is not in June are removed from the dataset. The Agent Last Order Date is checked, because if the agent did order in the last month, it could suggest that the agent just started ordering products. In total, 295 agents and their corresponding 432 drops were removed from the dataset.

Outlier detection on data of one variable, with the help of boxplots:

- Number of agents leaving or arriving on a day is also cleaned with the help of boxplots. This dataset first had to be created by summing over the number of agents with the same join date or leaving date per date. If such a date was found to be a Sunday, the number of leaving or arriving agents of that specific day were added to the Monday afterwards. The upper bound was found to be 30 and 13 for respectively the arriving and leaving rate. However, the boxplots did not show any outliers.

C Distributions

C.1 Distributions drop-frequency

In order to find the distribution of the drop-frequency, the drop-frequency first had to be calculated for the set of agents over a certain time frame. As the drop-frequency is defined to be the frequency of ordering per day over a time frame of four weeks, the drop-frequency of the agents in the last three such time intervals of 2016 have been calculated. These time intervals more or less correspond to October, November and December.

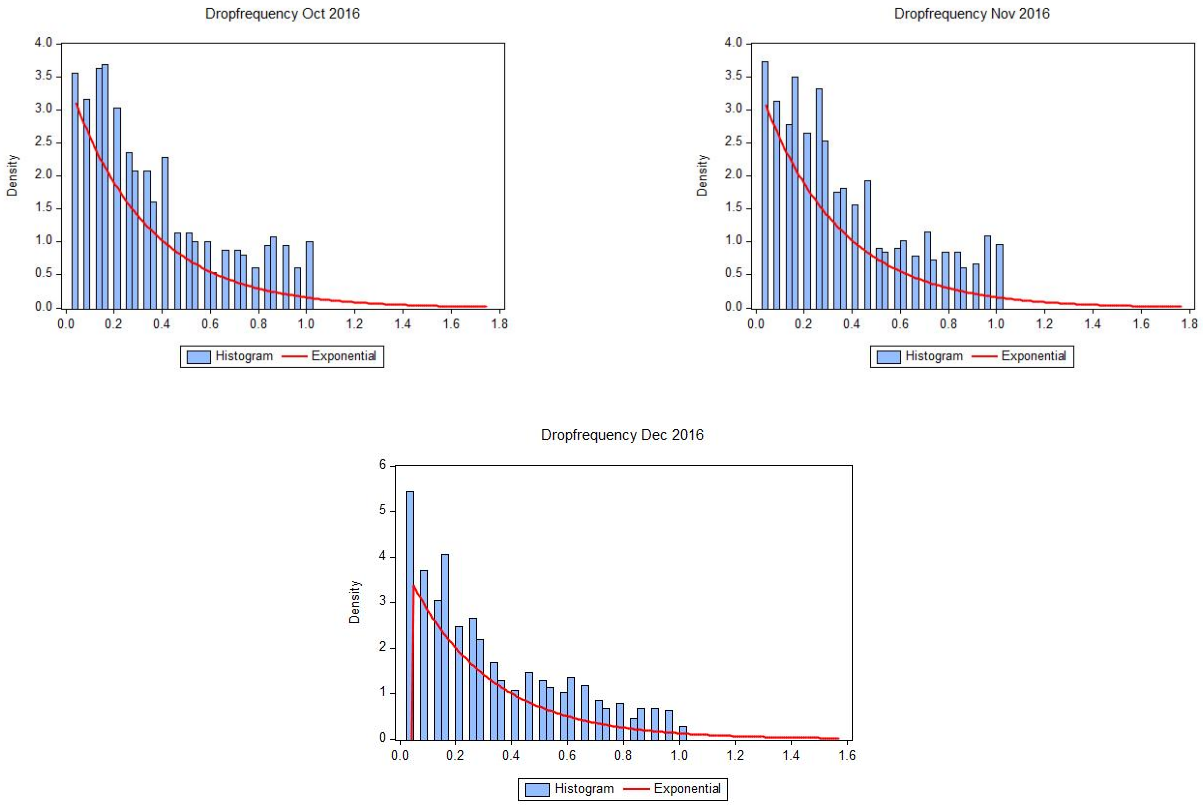

Figure 12: Histogram and exponential distribution for drop-frequencies of October, November and December 2016

From the histograms (Figure 12) created from these data it can clearly be observed that there are many more agents with a low drop-frequency than with a high drop-frequency. Even though the exponential distribution does not show a very good fit (with a probability 0.000 of coming from the exponential distribution for all three time intervals), it represents the just mentioned observation best. In addition, note that it can be observed from Figure 12 that the exponential distribution usually underestimates the drop-frequency. However, in order to ensure that the simulated drop-frequency is between 0 and 1, all observations above 1 will be discarded and a new number will be generated. Consequently and because underestimation is the case for all observations, the resulting simulation may more or less fit the reality.

Finally, to account for the discrete value of the drop-frequency (the value can only be a multiple of $\frac{1}{24}$, because it is based on observations over 24 days), the CDF will be split in 24 parts, with each part representing a value from the set $\{\frac{1}{24}, \frac{2}{24}, ..., \frac{24}{24}\}$. Note that 0 is excluded, as this would mean that the agent is inactive. Note that the parameter λ of the exponential distribution is set equal to the inverse of the average drop-frequency of the three intervals, which is 0.35.

C.2 Distributions arriving and leaving rate

First of all, note that the arrival rate had many observations still from before July 2016, all the way back until in 2014. However, these observations are useless as our dataset lacks information about this period and possibly many other (currently inactive) agents had entered these days. For this reason, only the data of the arriving rates from the second half of 2016 is used in this analysis.

For both the arriving rate and leaving rate it was checked if seasonality, a breakpoint or a trend was visible. This was done by inspecting the graph and by making use of moving averages. None of these indicated that seasonality or a breakpoint was present in the data. However, a positive trend was visible for the leaving rate. Indeed, when trying to model the data by means of an OLS regression with a constant and a trend coefficient, a significant trend of 0.017277 per day was found. On the other hand, via the same way, no significant trend was found for the arriving rate, confirming our observations. However, it seems illogical to have an increasing leaving rate, while the arriving rate remains constant as this would imply that at some point in time, the total set of agents will be shrinking, while the company is actually expected to grow. When following the estimated OLS model based on the data of 2016, it turns out that the average leaving rate rises above the arriving rate in July 2017 already, which means this would happen during the simulation. Because it is undesirable to have a larger leaving rate than arriving rate in our simulation, it is assumed that the growth of the leaving rate stops at the end of 2016. This means that it is expected that the average leaving rate in 2017 is constant and equal to the arriving rate at the last day of 2016, which is 3.12 according to the result of the OLS regression.

Next, the normal, exponential, logarithmic and uniform distributions were tested to fit the data. Note that for the leaving data those QQ plots were applied to the data after subtraction of the trend. The corresponding QQ plots can be found in Figure 13 and 14. For the arriving rate, the exponential distribution seemed to fit best, conforming also with the high value (127.44) of the Jarque-Bera statistic (which should be below 5.99 to indicate a normal distribution with a probability of 95%). For the leaving rate, both the exponential, normal and logistic distribution seem to fit relatively well. However, because the leaving rate can only have non-negative values, the exponential distribution would be the better choice. This means that, because the values should also be discrete, they are best modeled by a geometric distribution, as this is equal to the floor of an exponential distribution. Note that the mean values of the rates were found to be 5.40 and 3.12 for the arrival and leaving rate respectively and the mean of the geometric distribution is

equal to $\frac{1-p}{p}$. Consequently, the arriving and leaving rate will be simulated by geometric distributions with respectively parameters p_A and p_L, where $p_A = \frac{1}{5.40+1}$ and $p_L = \frac{1}{3.12+1}$.

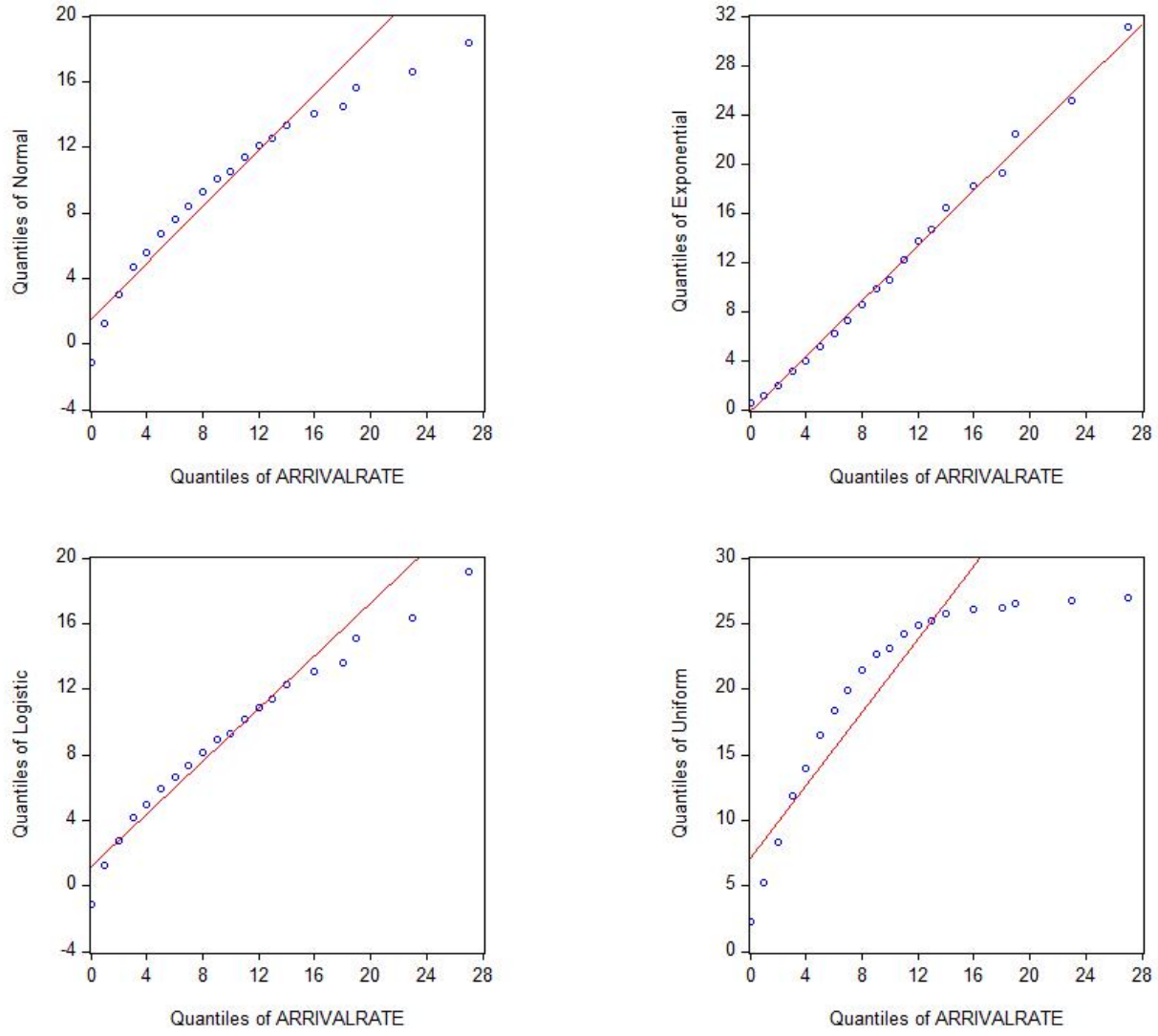

Figure 13: QQ-plots of arriving rate compared to normal, exponential, logistic and uniform distribution

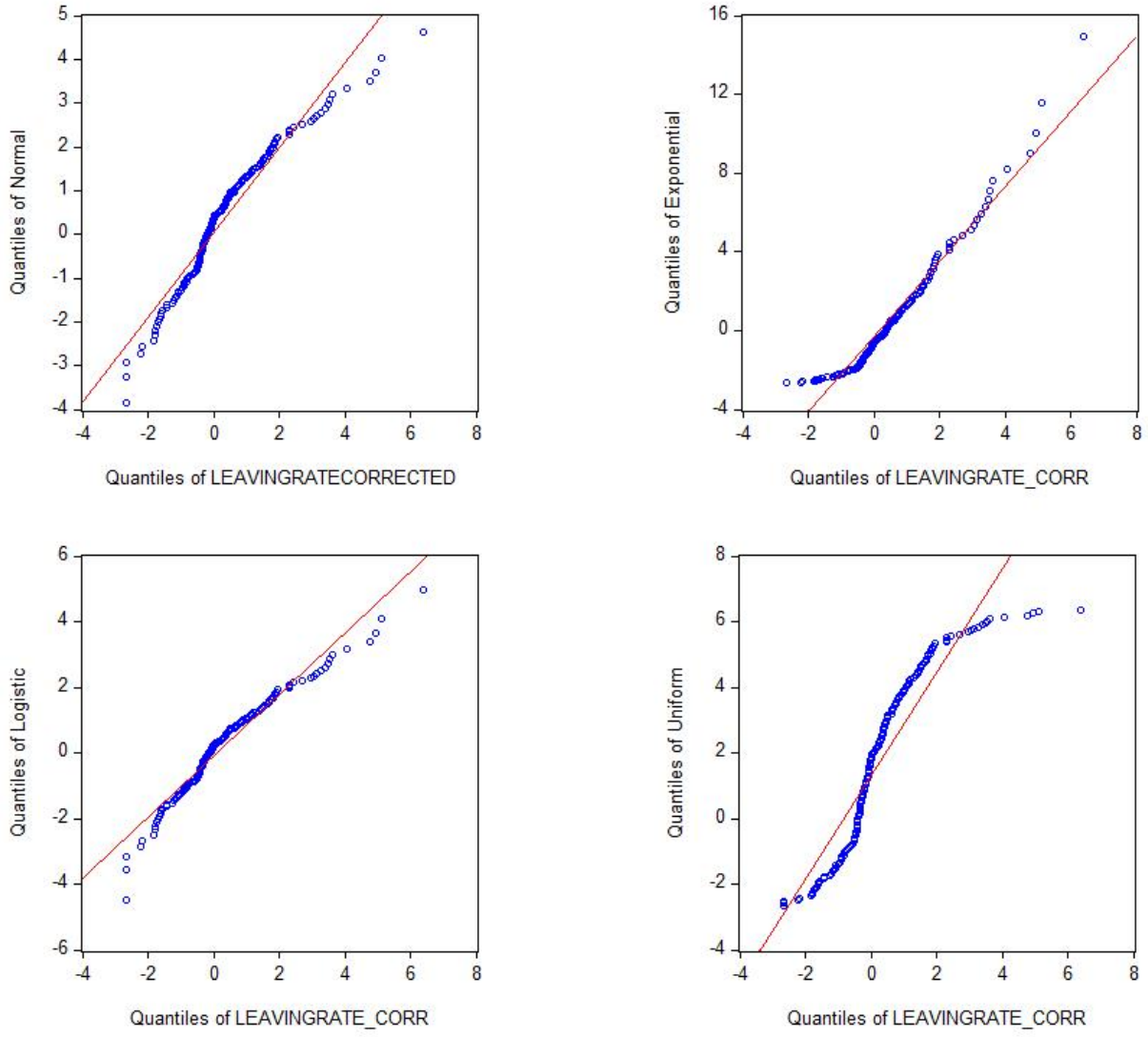

Figure 14: QQ-plots of leaving rate compared to normal, exponential, logistic and uniform distribution

D Best Driver Problem formulation

The Best Driver Problem (BDP) is equal to the Assignment Problem or Bipartite Matching Problem. Its formulation is as follows:

sets:

C : set of clusters $i = 1, ..., I$

S : set of potential drivers $j = 1, ..., J$

variables:

$$x_{ij} = \begin{cases} 1 & \text{if cluster } i \text{ is served by driver } j \\ 0 & \text{otherwise} \end{cases}$$

parameters:

$$c_{ij} : \text{number of agents in cluster } i \text{ that were previously served by driver } j \tag{12}$$

Accordingly, the BDP can be described by the following integer programming formulation:

$$min \sum_i \sum_j c_{ij} x_{ij}$$

$$s.t. \sum_j x_{ij} = 1 \qquad \forall i \in C \tag{13}$$

$$\sum_i x_{ij} = 1 \qquad \forall j \in C \tag{14}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in C, \forall j \in S \tag{15}$$

E Programming details

The drop-frequency is calculated at the start of every clustering run for all agents. The drop-frequency is based on the drops of an agent in the past 4 weeks. The following rules hold:

- If an agent is a new agent (arrived in the past 4 weeks), its drop-frequency is set equal to the average drop-frequency (0.35, see C.1). This was decided, because for a new agent the drop-frequency is still unknown and as such, the average should be a decent forecast.
- If an agent is older than 4 weeks, but has had no orders in the past 4 weeks, its drop-frequency is set equal to the minimum drop-frequency ($\frac{1}{24}$, see C.1). This was decided because otherwise such an agents would not be accounted for in the capacity constraints.
- In all other cases, the drop-frequency is calculated according to the definition, which means it is calculated by summing over the number of drops in the past 4 week and divide this number by the number of days (24).

Regarding the K&P algorithm:

- Step 3 (local exchange) is implemented such that it takes the first potential exchange found. For convenience, seed agents are excluded in this step and can thus not be switched.