Taxis Dispatch

Minimizing waiting times by maximizing coverage

M.J.K. van Wijngaarden

TU Delft

Taxis Dispatch

Minimizing waiting times by maximizing coverage

by

M.J.K. van Wijngaarden

Student number: 4357418
Project duration: March 8, 2019 – October 18, 2019
Thesis committee:

Dr. ir. J.T. van Essen,	TU Delft, supervisor
J. Pierotti, Msc.	TU Delft, supervisor
Prof. dr. ir. M.C. Veraar,	TU Delft
Dr. ir. J.G. Spandaw,	TU Delft

TU Delft

Abstract

In this thesis, we investigate whether maximizing the coverage of taxis can be beneficial when the goal is to minimize the waiting times of the clients. When dispatching taxis, often only current requests are taken into account and not future ones. We examined how beneficial it can be to take coverage into account. For taxi companies it is important to keep their customers satisfied, by serving them as quickly as possible. Often companies assign the taxis to the nearest requests. We investigate whether there are better dispatch methods. We developed three dispatch policies which use ILP models, and discussed what the best option is. The first policy does not consider the coverage and minimizes the waiting times of the clients among the requests that are taking place at that moment. The second one aims to maintain a good coverage and short waiting times when assigning taxis to requests. The third one also aims to maintain a good coverage and short waiting times when assigning taxis to requests and also relocates taxis to gain a better covered area. Those policies can be a contribution for taxi companies because they help to serve clients more quickly. To determine what the best way is of dispatching taxis, we run a simulation on the different policies on a real-data map and compared the results. The literature over taxi dispatch mostly conducts research on a small area like a city or a village. In this thesis, taxis dispatch takes place on a larger area where taxis commute between cities.

Contents

1 Introduction **1**

2 Literature Review **3**

3 Problem Formulation **5**
3.1 Determining q . . . 5
3.2 Penalty . . . 7

4 Policies **9**
4.1 Iterative Process. . . . 9
4.2 First Come First Serve . . . 10
4.3 Cover . . . 11
4.4 Cover and Relocate . . . 12

5 Computational Studies **15**
5.1 Input Data . . . 15
5.1.1 Generating Requests . . . 15
5.1.2 Travel Time Data . . . 15
5.1.3 Penalty Values . . . 16
5.2 Example . . . 17
5.3 Results . . . 18
5.3.1 Initial Conditions . . . 19
5.3.2 Results Initial Conditions . . . 19
5.3.3 Results Larger Number of Taxis and Higher Frequency . . . 21
5.3.4 Results Higher Frequency . . . 22

6 Conclusion **23**

Bibliography **25**

1 Introduction

In a world where time has great value, it is very important for taxi companies to know how to distribute their vehicles in order to best satisfy their customers. Companies can lose clients if their taxis take too long to reach them. In this thesis, we discuss different ways to decide which taxi should serve a client and how to distribute the vehicles. Nowadays, the most common policy is to serve the clients based on the closest taxi available.

A policy for taxi dispatching is an algorithm determining which taxi should serve a new request and how taxis should be distributed over an area. There are exponentially many combinations of how to dispatch taxis. In fact, every taxi can stay at the place where it is or it can drive to one of its adjacent cities where it may have a bigger probability for new requests to appear. When there are new requests, is assigning the nearest taxi the best option or is it better to send another taxi such that the coverage remains higher?

For instance, let us assume a situation where there are two taxis in an area. Taxi 1 is in the middle of the area and taxi 2 is in the south of the same area. When there is a request, at the east side of the area and taxi 1 is a bit closer that taxi 2, it might be smarter to send taxi 2. In fact, for the next request, the chances are that taxi 1 will be closer than taxi 2. In Figure 1.1 there is an example. On the left, we take the coverage into account when assigning a taxi to request 1, on the right, we assign the closest taxi to request 1. In both situations, we assign the taxi that remains to request 2. In this example, the total waiting time for the requests is less when we take (future) coverage into account. So, considering the coverage can have a big advantage when minimizing the total waiting time of clients.

Figure 1.1: Example of different ways to dispatch taxis

Different policies on how to dispatch vehicles can make a huge difference for a taxi company. With a good working policy, it is possible to serve more clients with the same number of taxis, thereby increasing the income of the companies. In addition, reducing the average waiting time makes clients more satisfied which leads to more future costumers for the company. Thus, it is really worth the effort to research a better dispatching policy.

2

Literature Review

In this thesis, we study different policies for dispatching taxis in order to minimize the total waiting time. In this section, we review some meaningful work which analysed similar problems.

To have a good taxi customer service, it is essential to have an efficient and fast dispatch system. Most dispatch systems only focus on reaching the customer as fast as possible, which means that the policy is to send the closest taxi to the customer (Lee et al., 2004). This policy is only concerned with the requests that take place at that moment and does not consider future requests. When a taxi is assigned to a request, then, it becomes unavailable for a certain time. This policy is mostly used in practice, although it was shown to be sub-optimal (Carter et al., 1972). To improve the service, more than one request should be considered. Since this policy is far from optimal, new algorithms were developed that outperformed this approach (Jagtenberg et al., 2015). The main difficulty of this problem is that the origin and destination of each request are unknown. This is a common problem for all the existing taxi companies.

The dispatch problem can be viewed from two sides. From the perspective of the passenger: the waiting time should be as short as possible. From the perspective of the taxi: the time it takes to find a customer has to be minimized. A taxi driver wants to minimize the time to find a passenger, because driving around looking for clients involves extra costs. If the taxi can find more passengers in a certain amount of time, it will be able to transport more passengers and it makes more profit. Research has been done with the aim of finding passengers for taxis. In particular, the question whether it is better for a taxi to wait or to search for passengers in a certain situation (Li et al., 2011). These authors analyze the average numbers of how often passengers are picked up by taxis in a certain period at a certain place. This allows them to find the best method for each situation, where they know the time and location of the taxi. The big difference with their work and this thesis is that they mainly try to minimize the time it takes for a taxi to find a passenger, while in this thesis, we try to minimize the waiting time of the passengers.

Nowadays, almost all vehicles are equipped with GPS devices; gathering and analyzing this data can provide a better understanding of the dynamic of an area. At first, the GPS devices where only used for navigation and localization. GPS data sets can also be used to create a routing algorithm to improve taxi dispatch systems by sending vehicles to pathways where there is a higher expectation for customers (Bi et al., 2003). Data are also used to predict the demand for taxis in certain areas (Chang et al., 2010) and to predict the vacant taxis in a city (Phithakkitnukoon et al., 2010). In Singapore, a satellite-based taxi dispatch is used to send available taxis to their customers (Liao, 2001). In Hong Kong, the government has been gathering the waiting times of the customers and taxis, and the availability and utilization of taxis since 1986. With this information, systems were developed that describe the movements of available and unavailable taxis. Those systems helped the government with decision making and taxi-regulation (Wong et al., 2001) because they gave insight of the impact into taxi fleet size and the degree of demand for taxi drives. In 1998, Yang and Wong introduced a model to describe the movement of the occupied and vacant taxis with the objective to minimize the individual search time required to meet the next costumer. In this thesis, we try to minimize the total waiting time

of the customers.

A lot of algorithms have been developed to get a vehicle to multiple places as quickly as possible (Laporte, 1992). One of the most common problems is that a delivery driver has to go to different cities and the travelled route has to be chosen in such a way that it is as short as possible and that the driver arrives at the same point as where he started from. This is known as the travelling salesman problem (TSP). Another similar problem is the vehicle routing problem (VRP) which is an m-TSP. While the TSP has one cycle (tour), the VRP makes use of multiple vehicles hence creating multiple tours. In the VRP, demand can be associated with the city that the deliverer has to visit and the vehicles have a maximum capacity. A time window and extra conditions can be applied to the problem hence forcing a route to reach a city within a certain period of time (El-Sherbeny, 2010). VRP can be divided into 3 categories. The first one aims to minimize the number of vehicles needed, such that all conditions of the problem are met with as few vehicles as possible. The second one minimizes the length of the routes, so choosing the routes in such a way that the vehicles have to travel the shortest possible distance. Finally, the third one focuses on minimizing the time, such that all cities are reached as quickly as possible.

A problem related to taxi dispatch is the dial-a-ride problem, which deals with requests for transportation between origin and destination points (Cordeau and Laporte, 2007). The difference is that the dial-a-ride problem is very flexible with its execution time, while with taxis it is very important that the client does not have to wait too long. Another similar problem is the k-server problem (Manasse et al., 1990). In the k-server problem, there is a metric space with a fixed number of servers, where a new request can appear in a given time step. The objective is to minimize the travelling time deciding which server has to respond to a request. The main distinction between the k-server problem and the taxi dispatch problem is the following; in the k-server problem, requests for service do not overlap in time, while with the taxi dispatch problem, taxis can be unavailable when there is a new request.

Taxis are not the only vehicles with a dispatch problem; ambulances and police cars face the same issues. References about models for ambulances that determine the best locations and the required number of ambulance to be placed at each base can be found in (Jagtenberg et al., 2015) and (van den Berg and Aardal, 2015). There are some crucial differences between those dispatch problems and ours. The final destination of an ambulance with its patient is known in advance (in most cases the nearest hospital) while taxis have to bring the clients to their chosen destinations. In addition, the presence of police cars reduces the crime rates and thereby the number of requests while, typically taxis generate more requests by driving through locations.

It is essential for taxis that they provide a good coverage for the region such they can reach new requests as soon as possible. There are many models which determine the locations of vehicles like taxis, ambulances, police cars, etc. Those models try to dispatch the vehicles in such a way that every part of an area is quickly accessible; hence, they take into account possible future requests. Among those models, we point out the Maximal Coverage Location Model (MCLP) and Location Set Covering model (LSCM). The MCLP is a model that maximizes the coverage given limited resources, which is the fixed number of vehicles (Church and Revelle, 1974). Instead, the LSCM model reviews the area and determines the minimum number of vehicles needed to cover that area (Toregas et al., 1971). In those models, they assume conditions which are not always possible for taxis. The first assumption is that the vehicles are always available. In the taxi scenario, a vehicle might be busy with a client, making it unavailable. The second assumption is that the coverage of a certain area assumes value 1 or 0. In our case, we prefer to have a fractional coverage, to appreciate more the variations between totally uncovered and perfect coverage. Daskin introduced a variant of the MCLP where it is not assumed that a vehicle is always available. The model was labeled as MEXCLP (Maximum Expected Covering Location Problem) and it takes into account that a vehicle may be unable to respond to demands. Despite the introduction of this model which could improve vehicles dispatch systems, only limited work was published where it is applied on taxi dispatch systems.

Our main contribution is to analyze taxi dispatch policies that minimize the average waiting time of the clients (instead of minimizing the waiting time for taxis who are searching for potential clients). To test the effectiveness of our policies, we propose a case study using real data. Almost all the articles that investigated taxis dispatch, did that for an intraurban situation, in this thesis we examine an interurban situation.

3

Problem Formulation

In this section, we introduce the notations that we use. In this problem, passengers request a taxi to bring them to their destination. Let K be the set of taxis, which can be dispatched over the set of cities I. The set of requests is defined as R. The number of inhabitants of city $i \in I$ is defined by d_i. Accordingly, the total number of inhabitants of the whole area is defined by $\sum_{i \in I} d_i = D$. The initial time for our model is S while the final time is F. We use a time step of Δt and T indicates the set of all time instants . The driving time between two cities $i, j \in I$ at time t is given by τ_{ijt}. There is a probability that a taxi is busy with a client and that it cannot serve the next request, which we denote by q. The set of requests that take place at time t is defined as R_t, where $\bigcup_{t \in T} R_t = R$. We consider a city covered by a taxi $k \in K$, when that taxi takes no longer than the threshold time γ to reach that city. The set of cities laying within the threshold time γ from city $i \in I$ at time $t \in T$ is defined as $I_{it} \subset I$ and the set of cities that are adjacent to city i is defined as $A_i \subset I$. It can be the case that a taxi takes too long to reach a requests and therefore loose that request. We consider a request lost if a taxi need more than time β to reach it. Before a taxi could serve a new client, it must first complete its requests. We define tc_k the time that it takes for taxi $k \in K$ to complete all its assigned requests. The time that a taxi needs to reach a request is given by t_{rk} with $k \in K$ and $r \in R$. This takes into account that the taxi first needs to finish all its assigned requests. The last scheduled destination of a taxi $k \in K$ is defined as $d_k \in I$, while $o_r \in I$ and $d_r \in I$ are the origin and destination point of a request, respectively. We make the following assumptions. Firstly, there is a fixed number of taxis. Secondly, the travel times τ_{ijt} are known in advance. Also, we assume that whenever a taxi is serving a client, it is not available. So, while a taxi is approaching a client or driving to its destination, it cannot serve other clients. Finally, the last assumption is that the probability distribution of the demand for each time instant is known in advance. The average number of requests that take place in a time step Δt is given by λ. Table 3.1 summarizes the notations described above.

3.1. Determining q

In this section, we determine the probability q that a taxi is assigned to a request, which means that it cannot serve another request for some time. We do this by computing the probability that a taxi is not available, by dividing the total workload by the number of taxis times the number of time steps. The total workload is equal to the average number of requests during the day times the average travel time of a taxi when it must serve a client.

$$q = \frac{\text{Average number of requests} \cdot \text{Average travel time}}{\text{Number of taxis} \cdot \text{Number of times steps}}$$

First, we estimate what is the total workload of a day. We know that λ is the average number of requests during time step Δt. There is a total of $|T|$ time steps, so we expect that there is a total of $|T| \cdot \lambda$ requests. Now we determine what the average duration of a request. A request consists of two parts, driving to the starting point of the request and driving to the destination point of the request.

Table 3.1: Notations

K	The set of taxis
R	The set of requests
I	The set of cities
d_i	The number of inhabitants of a city $i \in I$
D	The total number of inhabitants, $\sum_{i \in I} d_i = D$
S	Release date of the taxis
F	Due date of the taxis
Δt	The time step of the model
T	The set of all considered time instants
τ_{ijt}	The driving time between $i \in I$ and $j \in I$ at time $t \in T$
R_t	The set of requests that take place at time $t \in T$
q	The probability that a taxi is not available
γ	The threshold time for coverage
λ	The average number of requests in a time step Δt
β	The threshold time of a request
I_{it}	The set of cities that can reach city $i \in I$ within threshold time γ at time $t \in T$
A_i	The set of cities that are adjacent to city $i \in I$
tc_k	The time taxi $k \in K$ needs to complete all its assigned requests
t_{rk}	The time a taxi $k \in K$ needs to reach request $r \in R$
o_r	Origin point request $r \in R, s_r \in I$
d_r	Destination request $r \in R, d_r \in I$
d_k	The final destination of taxi $k \in K$ when he completes all its assigned requests, $d_k \in I$

Determining the average driving time to the starting point of a request is hard, because there are $|K|$ taxis dispatched over an area and there are $|I|$ locations where the new request can start from, which gives almost endless possibilities of how fast a taxi can reach a starting point of a request. In our model, we expect that every request has to be reached within the time β. Hence, we assume the average driving time to the starting point of a request to $\frac{1}{2}\beta$.

Secondly, we determine the driving time from the starting to the destination point of a request. We define the average travel time over all time steps T as:

$$\frac{\sum_{t \in T} \tau_{ijt}}{|T|} = \overline{\tau}_{ij}.$$

Now we can compute the average driving time from city $i \in I$ to a random destination point $j \in I$. We calculate this by, multiplying the probability that the taxi has to drive to destination point $j \in I$ times the driving time from city i to j. The probability of a city to be chosen depends on its number of inhabitants.

$$\sum_{\substack{j \in I \\ j \neq i}} \mathbb{P}(\text{destination point is } j) \cdot \text{Driving time from } i \text{ to } j =$$

$$\sum_{\substack{j \in I \\ j \neq i}} \frac{d_j}{D - d_i} \cdot \overline{\tau}_{ij} = \frac{\sum_{\substack{j \in I \\ j \neq i}} d_j \overline{\tau}_{ij}}{D - d_i}$$

The reason why $j \neq i$ is because the starting point can not be the same as the destination point. That is also why we divide by $D - d_i$.

Now we have obtained the average driving time from a random city $i \in I$ to the destination point of a request. To determine the average driving time of a request from its starting to its destination point, we have to consider all cities as possible starting point. We calculate this by multiplying the probability that a city $i \in I$ is

the starting point times the average driving time from city $i \in I$ to a random destination point $j \in I$:

$$\sum_{i \in I} \frac{d_i}{D} \frac{\sum_{\substack{j \in I \\ j \neq i}} d_j \overline{\tau}_{ij}}{D - d_i} = \frac{\sum_{i \in I} d_i \frac{\sum_{\substack{j \in I \\ j \neq i}} d_j \overline{\tau}_{ij}}{D - d_i}}{D}.$$

So the average time spend on a request is:

$$\overline{\tau}_t = \frac{\sum_{i \in I} d_i \frac{\sum_{\substack{j \in I \\ j \neq i}} d_j \overline{\tau}_{ij}}{D - d_i}}{D} + \frac{1}{2}\beta.$$

Then, the probability that a taxi is assigned to a request is given by:

$$q = \left(\frac{\sum_{i \in I} d_i \frac{\sum_{\substack{j \in I \\ j \neq i}} d_j \overline{\tau}_{ij}}{D - d_i}}{D} + \frac{1}{2}\beta \right) \cdot \lambda \Big/ |K| \cdot \Delta t.$$

3.2. Penalty

To determine which policy works best, we define the penalty s_{rk}, which is a penalty that is paid for each taxi that is assigned to a request. We name this penalty s_{rk} and it depends on the time that it takes for taxi $k \in K$ to reach request $r \in R$ i.e., the value t_{rk}. The longer it takes for a taxi to reach a request, the higher the penalty. The penalty is piece wise linear monotonic, which means that after every time interval it increases. The increasing steps get bigger after every interval. We do this because we consider that it is worse to have a client waiting a long time instead of having two clients waiting a short time. There is also a maximum for the penalty because, after a certain time, we consider that requests are no longer willing to wait for their taxi; hence, the client is considered lost. This loss is associated with the maximum penalty which is $s_{\max}$.

4

Policies

In this section, we introduce our three policies. The first policy is called First Come First Serve and it focuses only on the requests that take place at a certain time instant, so it does not take future requests into consideration. The second policy is called Cover; with this policy, when assigning a taxi to a request, it takes into account the coverage of each taxi, but does not relocate them. The third policy is called Cover and Relocate; this policy also takes into account the coverage when assigning a taxi to a request and relocates the taxis that are not dispatched to increase the total coverage.

4.1. Iterative Process

When we dispatch taxis, we use an iterative process. Each of the three different policies uses its own ILP which assigns taxis to requests and (eventually) relocates the vehicles. The process repeats itself after every time step Δt. The flowchart in Figure 4.1 displays how the iterative process works.

0. Time = S. All taxis are stationed at a city, so $tc_k = 0$, $\forall k \in K$ and d_k is the city where taxi $k \in K$ is stationed.

1. All taxis need Δt time less to complete their requests. So, $tc_k = \max\{0, tc_k - \Delta t\}$.

2. Determine if there are requests. Go to Step 4 if there are no requests. If there are requests $r \in R_t$, determine o_r and d_r.

3. Solve the ILP corresponding to the used policy. The ILP of the policy determines which taxis $k \in K$ should serve which requests $r \in R_t$ and where taxi $k \in K$ should be relocated to. If a taxi k is assigned to a request r, then the new time that taxi k needs to complete all its assigned requests is its old time (tc_k) plus the driving time from its final destination to the starting point of the new request plus the travel time from the starting point of the new request to the destination point of the new request. Let $x = d_k$, $y = o_r$ and $z = d_r$. Then, we update t_{ck} with $tc_k + \tau_{xyt_1} + \tau_{yzt_2}$, where, $t_1 = tc_k + t$ and $t_2 = t_1 + \tau_{xyt_1}$. The new final destination of taxi k is the destination point of request r. So $d_k = d_r$. In case of policy Cover and relocate, when taxi $k \in K$ is relocated from city $i \in I$ to city $j \in I$. Then the new final destination of taxi k is city j, so $d_k = j$ and tc_k becomes the travel time from city i to j at time t, so $tc_k = \tau_{ijt}$.

4. Time = Time + Δt. Go to Step 1 if Time $\neq$ F.

5. Stop the iterative process

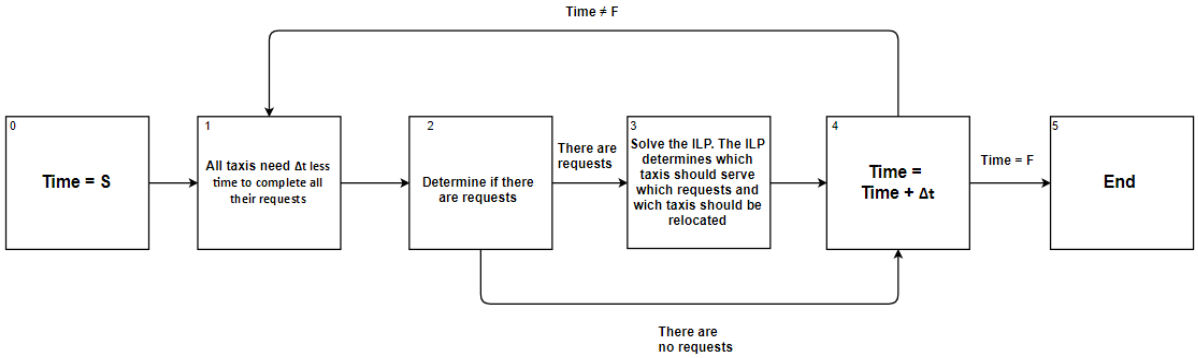

Figure 4.1: Flowchart iterative process

4.2. First Come First Serve

This policy is exactly what most taxi companies apply (Lee et al., 2004). When there are requests at a time instant, it assigns taxis to requests so that the total waiting time is minimal for that particular time instant. In this policy, when the taxi has dropped off its passenger, it stays at its destination.

In this model, we minimize the penalty (which is equal to s_{rk}) that is paid when a taxi $k \in K$ is assigned to a request $r \in R_t$. We precompute parameters t_{rk} by determining how long it takes for taxi k to complete all its requests and add the time from its final destination to the starting point of request r. With t_{rk}, we determine the associated s_{rk}. When a request is too far away, there is no point in sending a taxi there, because the client is already lost in the case the maximum penalty is paid. In order to model this as an ILP problem, we define variable x_{rk} which indicates whether taxi $k \in K$ is assigned to request $r \in R_t$.

$$x_{rk} = \begin{cases} 1, & \text{If taxi } k \in K \text{ is assigned to request } r \in R_t. \\ 0, & \text{Otherwise.} \end{cases} \qquad \forall r \in R_t, \quad \forall k \in K$$

We write the ILP-model for this problem as follows:

$$\text{Minimize} \sum_{r \in R_t} \left[\sum_{k \in K} s_{rk} x_{rk} + s_{\max} \left(1 - \sum_{k \in K} x_{rk} \right) \right] \tag{4.1}$$

Subject to

$$\sum_{k \in K} x_{rk} \leq 1 \qquad \forall r \in R_t \tag{4.2}$$

$$\sum_{r \in R_t} x_{rk} \leq 1 \qquad \forall k \in K \tag{4.3}$$

$$\sum_{k \in K} t_{rk} x_{rk} \leq \beta \qquad \forall r \in R_t \tag{4.4}$$

$$x_{rk} \in \{0, 1\} \qquad \forall r \in R_t, \quad \forall k \in K$$

The objective function (4.1) consists of two terms: the first term indicates the penalty to be paid when a taxi is assigned to a request and the second term indicates the maximum penalty to be paid when a request is not assigned to a taxi. When taxi k is linked to request r, the second term of (4.1) is 0. If there is no taxi assigned to request r, then $\sum_{k \in K} x_{rk}$ is 0. Because of this, the first term of (4.1) equals 0 and the second term equals $s_{\max}$ (the maximum penalty). Because a maximum penalty is paid if no taxi is assigned to a request, the ILP solver tries to ensure that a taxi is assigned to each request but this is not always possible. Constraint (4.2) enforces that a request cannot be assigned to multiple taxis. Since $\sum_{k \in K} x_{rk}$ equals the number of taxis that are assigned to request r and we have set that sum smaller than or equal to 1, there can never be more than 1 taxi assigned to a request. Constraint (4.3) enforces that a taxi cannot be assigned to two or more requests at the same time. We know that $\sum_{r \in R_t} x_{rk}$ is equal to the number of requests from R_t to which taxi k is assigned and we set this sum smaller or equal to 1. So, taxi k can never serve more than 1 request from R_t. Constraint

(4.4) makes sure that the travel time from a taxi to a request does not take longer than β time instant. t_{rk} is equal to the time it takes for taxi k to reach request r and x_{rk} is 1 when taxi k is assigned to request r. So, forcing $\sum_{k\in K} t_{rk}x_{rk}$ to be smaller or equal to β, it can never happen that a taxi assigned to any request takes longer than β time to reach that request.

4.3. Cover

This policy takes into account the total coverage when assigning a taxi to a client, but it does not relocate the taxis after serving a passenger. When there are new requests, it determines how long it takes for each taxi to reach the requests and what the total coverage becomes considering that some taxis are unavailable because they are assigned to some requests. Then, it assigns the taxis to requests such that a good trade off between maximizing coverage and minimizing the total waiting time is achieved. Consider city $i \in I$ that is within the range of the threshold time of m taxis, then, we define the expected covered demand of that city as $E_m = d_i(1-q^m)$. The marginal contribution of the m^{th} taxi to the coverage is $E_m - E_{m-1} = d_i(1-q)q^{m-1}$. Then, it holds that $\sum_{m=1}^{N} d_i(1-q)q^{m-1} = d_i(1-q^N)$, so we aim to have an N as high as possible for every $i \in I$.

This model aims to maintain a good coverage and short waiting times when assigning taxis to requests. As explained in Section 4.2, t_{rk} is the time it takes for taxi k to reach request r. The expected covered demand of a city $i \in I$ is defined by $E_m = d_i(1-q^m)$, where m is the number of taxis that can reach the city within the threshold time γ. To maximize the total coverage of an area, we maximize $\sum_{i\in I}\sum_{m=1}^{|K|} d_i(1-q)q^{m-1}y_{im}$, where y_{im} is a variable equal to 1 if city i can be reached within the threshold time γ by at least m taxis. We want to maximize the coverage without considering the taxis that are assigned to requests. We define parameter z_{ikt}, which is equal to 1 if taxi k can reach city i at time t within the threshold time, otherwise is 0. When we compute the driving time of taxi k to a city i, we determine how long it takes for taxi k to complete all its requests (tc_k) and add the driving time from its final destination to city i. So, if $tc_k + \tau_{d_k i t_1} \leq \gamma$, where $t_1 = tc_k + t$, then $z_{ik} = 1$. In order to model this as an ILP problem, we define variables y_{im}, $i \in I$, $m \in \{0,1,...,|k|\}$ and x_{rk} $r \in R_t$, $k \in K$, as follows:

$$x_{rk} = \begin{cases} 1, & \text{If taxi } k \in K \text{ is assigned to request } r \in R_t. \\ 0, & \text{Otherwise.} \end{cases} \qquad \forall r \in R_t, \quad \forall k \in K$$

$$y_{im} = \begin{cases} 1, & \text{if city } i \text{ is within range of at least } m \text{ taxis.} \\ 0, & \text{Otherwise.} \end{cases} \qquad \forall i \in I \quad , \forall m \in \{0,1,2,...,|K|\}$$

The following is the ILP model for this policy.

$$\text{Minimize} \quad D\sum_{r\in R_t}\left[\sum_{k\in K} s_{rk}x_{rk} + s_{\max}\left(1-\sum_{k\in K}x_{rk}\right)\right] - \sum_{i\in I}\sum_{m=1}^{|K|} d_i(1-q)q^{m-1}y_{im} \tag{4.5}$$

Subject to

$$\sum_{k\in K} x_{rk} \leq 1 \qquad \forall r \in R_t \tag{4.6}$$

$$\sum_{r\in R_t} x_{rk} \leq 1 \qquad \forall k \in K \tag{4.7}$$

$$\sum_{k\in K} t_{rk}x_{rk} \leq \beta \qquad \forall r \in R_t \tag{4.8}$$

$$\sum_{m=1}^{|K|} y_{im} \leq \sum_{k\in K}\left[z_{ikt}\left(1-\sum_{r\in R_t}x_{rk}\right)\right] \qquad \forall i \in I \tag{4.9}$$

$$x_{rk} \in \{0,1\} \qquad \forall r \in R_t, \quad \forall k \in K$$

$$y_{im} \in \{0,1\} \qquad \forall i \in I, \quad \forall m \in \{0,1,2,...,|K|\}$$

In objective function (4.5), we minimize the waiting times while maximizing the coverage. The coverage term

is negative because it is a minimisation problem. In (4.5), we multiply the first term with D to have a trade off between the coverage and the waiting time. Constraint (4.6) enforces that a request cannot be assigned to multiple taxis. Constraint (4.7) forces a taxi to not be assigned to two or more requests at the same time. Constraint (4.8) ensures that the travel time from a taxi to a request does not take longer than β. Constraint (4.9) enforces that a city is covered by m taxis if at least m taxis can reach city i within the threshold time without considering the taxis that are being assigned to requests. $\sum_{k\in K} z_{ikt}$ is equal to the number of taxis that can reach city i within the threshold time. If any of these taxis is assigned to a request then $1-\sum_{r\in R_t} x_{rk}$ equals 0, and therefore, it will not be counted. This ensures that y_{im} is never higher than the number of taxis that can reach city i within the threshold time minus the taxis that are assigned to a request. There is no need to add the extra constraint $y_{ih} \leq y_{im}$ for $h \leq m$. This will always hold for an optimal solution, since $E_m - E_{m-1}$ is decreasing in m.

4.4. Cover and Relocate

This policy takes the coverage into account when assigning a taxi to a client and relocates the taxis to increase the total coverage. We consider only adjacent cities for the relocation. This way taxis do not drive too far to their new location. This has two advantages, the first advantage is that the new situation where the coverage is increased occurs a lot faster. The second advantage is that a taxi does not spend too much time relocating between cities.

This model aims to maintain a good coverage and short waiting times when assigning taxis to requests and also to gain a better covered area by relocating taxis. In this model, we the set A_i of cities adjacent to city i and parameter the set I_{it} of cities that can reach city i within the threshold time γ at time t. The set K_{0t} denotes the set of taxis where $tc_k = 0$, so all taxis that are not assigned to requests or that are not being relocated. To model this policy, we introduce binary variable c_{ijk}, is equal to 1 if taxi k is relocated from city $i \in I$ to city $j \in I$. Since we only relocate taxis that are stationed at a city and those taxis can only relocate from the city where they are stationed to an adjacent city, we consider the variable c_{ijk} only for $k \in K_{0t}$, $i = d_k$ and $j \in A_{d_k}$. In order to model this as an ILP problem, we define variables x_{rk} $r \in R_t$, $k \in K$,y_{im} $i \in I$, $m \in \{0,1,...,|k|\}$ and c_{d_kjk} $k \in K_{0t}$, $i \in I_k$, $j \in A_{d_k}$ as follows:

$$x_{rk} = \begin{cases} 1, & \text{If taxi } k \in K \text{ is assigned to request } r \in R_t. \\ 0, & \text{Otherwise.} \end{cases} \qquad \forall r \in R_t, \quad \forall k \in K$$

$$y_{im} = \begin{cases} 1, & \text{if city } i \text{ is within range of at least } m \text{ taxis.} \\ 0, & \text{Otherwise.} \end{cases} \qquad \forall i \in I \qquad , \forall m \in \{0,1,2,...,|K|\}$$

$$c_{d_kjk} = \begin{cases} 1, & \text{If taxi } k \in K_{0t} \text{ relocates from city } d_k \text{ to city } j \in A_{d_k} \\ 0, & \text{Otherwise.} \end{cases} \qquad \forall k \in K_{0t} \qquad , \forall j \in A_{d_k}$$

We write the ILP-model for this problem as follows:

$$\text{Minimize} \quad D \sum_{r\in R_t} \left[\sum_{k\in K} s_{rk} x_{rk} + s_{\max} \left(1 - \sum_{k\in K} x_{rk} \right) \right] - \sum_{i\in I} \sum_{m=1}^{|K|} d_i (1-q) q^{m-1} y_{im} \tag{4.10}$$

Subject to

$$\sum_{k\in K} x_{rk} \leq 1 \qquad \forall r \in R_t \tag{4.11}$$

$$\sum_{r\in R_t} x_{rk} \leq 1 \qquad \forall k \in K\backslash K_{0t} \tag{4.12}$$

$$\sum_{r\in R_t} x_{rk} + \sum_{j\in A_{d_k}} c_{d_k jk} \leq 1 \qquad \forall k \in K_{0t} \tag{4.13}$$

$$\sum_{k\in K} t_{rk} x_{rk} \leq \beta \qquad \forall r \in R_t \tag{4.14}$$

$$\sum_{m=1}^{|K|} y_{im} \leq \sum_{k\in K}\left[z_{ikt}\left(1-\sum_{r\in R_t} x_{rk}\right)\right] + \sum_{\{k\in K_{0t}|d_k\notin I_{it}\}}\sum_{j\in(I_{it}\cap A_{d_k})} c_{d_k jk} - \sum_{\{k\in K_{0t}|d_k\in I_{it}\}}\sum_{j\in((I\backslash I_{it})\cap A_{d_k})} c_{d_k jk} \qquad \forall i \in I \tag{4.15}$$

$$x_{rk} \in \{0,1\} \qquad \forall r \in R_t, \quad \forall k \in K$$

$$y_{im} \in \{0,1\} \qquad \forall i \in I, \quad \forall m \in \{0,1,2,...,|K|\}$$

$$c_{ijk} \in \{0,1\} \qquad \forall i,j \in I, \quad \forall k \in K$$

In objective function (4.10), we multiply the first part with D to have a trade off between the coverage and the waiting time. Constraint (4.11) enforces that a request cannot be assigned to multiple taxis. Constraint (4.12) enforces that a taxi, in the set of taxis that are being relocated or are assigned to requests, cannot be assigned to two or more requests at the same time. Constraint (4.13) ensures that a taxi, in the set of taxis that are not assigned to requests or are not being relocated, can only be assigned to one request or can be relocated to one city or neither. Since $\sum_{r\in R_t} x_{rk}$ is equal to the number of requests to which taxi k is assigned and $\sum_{j\in A_{d_k}} c_{d_k jk}$ is equal to the number of cities where taxi k is relocated to, if their sum is less or equal to 1, then, a taxi can never, in the same time instant, be relocated more than once or be assigned to multiple requests. Constraint (4.14) makes sure that the travel time from a taxi to a request does not take longer than β. Constraint (4.15) enforces that a city is covered by m taxis if at least m taxis can reach city i within the threshold time without taking into account the taxis that are being assigned to requests. We have already discussed that $\sum_{k\in K}\left(z_{ikt}\left(1-\sum_{r\in R_t} x_{rk}\right)\right)$ is equal to the number of taxis that can reach city i within the threshold time minus the taxis that are busy serving clients. Then, $\sum_{\{k\in K_{0t}|d_k\notin I_{it}\}}\sum_{j\in(I_{it}\cap A_{d_k})} c_{d_k jk}$ is equal to the number of taxis that are relocated from city that is outside the threshold time to a city that lies within the threshold time of city i. Instead, $\sum_{\{k\in K_{0t}|d_k\in I_{it}\}}\sum_{j\in((I\backslash I_{it})\cap A_{d_k})} c_{d_k jk}$ is equal to the number of taxis that are relocated from a city within the threshold time of city i to a city which is outside the threshold time of city i.

5

Computational Studies

In this Section, we set out the initial conditions for the iterative process and apply them to the three policies we developed. Those results are reviewed in this Section. This simulation we evaluated on a real-data map. A small example of the iterative process is also elaborated so that it provides clarity of how it exactly works.

5.1. Input Data

For the simulation, we created a real-data map, where requests can be generated on and be taxis dispatched and relocated. In the remain of this Section, we describe how we generated our data and how we turned our parameters.

5.1.1. Generating Requests

Requests can take place at any time step t. To generate the requests for a given time step t, we use the Poisson distribution where the probability of events can be defined as:

$$\mathbb{P}(m \text{ events in interval}) = \frac{\lambda^m}{m!} e^{-\lambda}. \tag{5.1}$$

In this equation, λ is the average number of events per interval. In our model, we used an interval of Δt, so if we expect n request every Δt time steps, then $\lambda = n$. When a request is generated at a given time step, the origin point and destination point must also be generated. Origin and destination points are randomly selected, considering probabilities directly proportional to the number of inhabitants of the cities. The selection of the destination point is constrained to be different from its origin point. This way, we can generate requests at random times and random places which are influenced by the number of inhabitants of a city.

5.1.2. Travel Time Data

To use realistic data, we retrieve travel times from Google Maps. Google Maps measures data from local road services such as highway cameras, as well as speed and location information from other Android devices accessing Google Maps. The departure time is important because the driving time depends on it. For example, the driving time is longer during rush hour. Using Google Maps, we collected data of the driving time for every quarter of an hour for one day. Travel time for instants in between are computed by interpolating the extremes. We use data from May 9th 2019 which is a normal weekday with normal traffic conditions. We determine the driving time between cities that are not adjacent by using the Dijkstra algorithm. Figure 5.1 shows all the considered cities and arcs of the province South Holland.

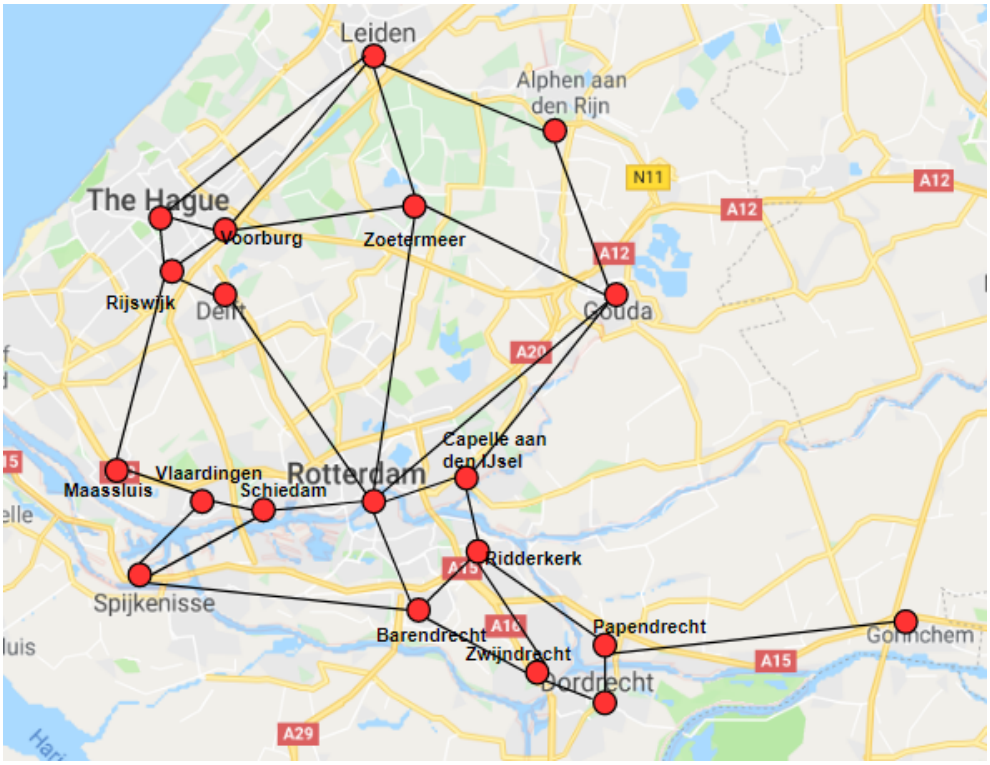

Figure 5.1: Cities and arcs used in computational study

5.1.3. Penalty Values

The penalty s_{rk} returns a value depending on how much time taxi $k \in K$ takes to reach request $r \in R_t$. After the first 10 minutes this penalty value increases every 5 minutes. The penalty value is the same between 0-10, minutes, 11-15 minutes, 16-20 minutes, etc. We do not expect that passengers are willing to wait more than 40 minutes for their taxi, so after 40 minutes the penalty value of s_{rk} achieves its maximum and assigns a penalty of 45. The penalty values are given in Figure 5.2.

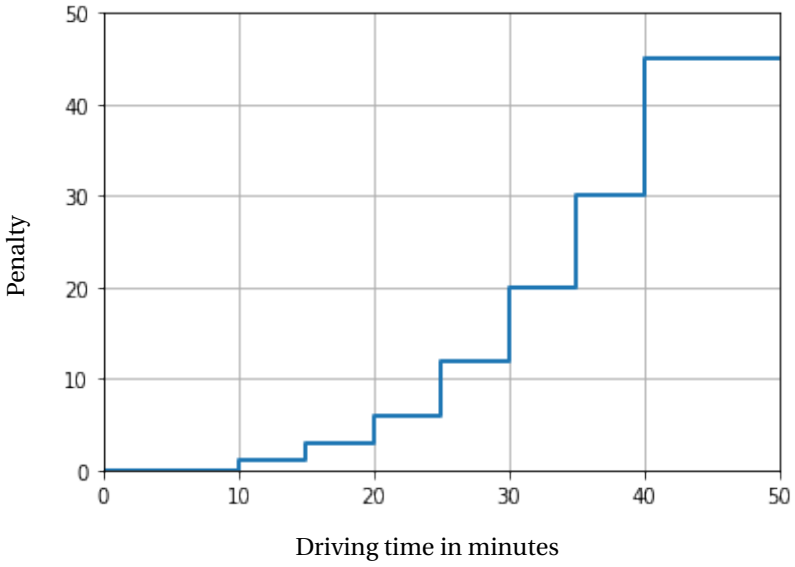

Figure 5.2: Penalty values s_{rk}

5.2. Example

To show how the policies work, a small example is given below. First, the initial conditions of the taxis are given, so the number of taxis that are being used and where they are positioned. Then, we generate the set R_1, which contains requests with randomly selected origin and destination points. With the information of the set R_1 and the initial situation of the taxis, the three different policies will determine which taxis will serve the requests and which taxis will be relocated. With the results of the policies, the penalty can be determined for each policy, by measuring how long the requests have to wait for their taxi. After this, a new situation will occur for each policy which describes where the taxis will be placed as indicated by the policy. Therefore, each policy has its own situation of the position of the taxis. After this, a new set R_2 is created of randomly generated requests. Then, the policies will determine which taxis should serve the requests of R_2 and whether taxis should be relocated. Afterwards, the penalty for each policy can be determined. At the end, we can compare the three policies by examining the total penalty. In Table 5.1, the initial positions of the three taxis are given.

Table 5.1: Initial positions of the three taxis

Taxi	Final destination taxi (d_k)	Time to finish tasks (t_{ck})
k_1	Leiden	0 minutes
k_2	Rotterdam	0 minutes
k_3	Dordrecht	0 minutes

Table 5.2 describes the requests that are generated in R_1 with their origin and destination points.

Table 5.2: Requests R_1

Request	Origin point (o_r)	Destination point (d_r)	Time (t)
r_1	Delft	Alphen aan den Rijn	0
r_2	Zwijndrecht	Gouda	0

Table 5.3 describes the results of the policies when they have as input R_1 and the initial positions of the taxis. These results indicate when a taxi has to serve a request and when it should be relocated. After that, the corresponding penalty is calculated using the penalty function.

Table 5.3: Results of policies for requests in R_1

Results First Come First Serve	Results Cover	Results Cover and Relocate
$x_{r_1k_2} = 1$	$x_{r_1k_1} = 1$	$x_{r_1k_2} = 1$
$x_{r_2k_3} = 1$	$x_{r_2k_3} = 1$	$x_{r_2k_3} = 1$
		$c_{9,6,k_1} = 1$
k_2 assigned to r_1	k_1 assigned to r_1	k_2 assigned to r_1
k_3 assigned to r_2	k_3 assigned to r_2	k_3 assigned to r_2
		k_1 relocates to The Hague
Penalty = 7	Penalty = 13	Penalty = 7

Tables 5.4, 5.5 and 5.6 describe the new situation of the taxis, so their new destination and the travel time to reach it before they get there. Table 5.4 shows the displacement of policy First Come First Serve, Table 5.5 displays the new setting of policy Cover and Table 5.6 pictures the new situation of policy Cover and Relocate.

Table 5.4: Displacement of policy First Come First Serve

Taxi	Final destination taxi (d_k)	Time to finish tasks (t_{ck})
k_1	Leiden	0 minutes
k_2	Alphen aan den Rijn	69 minutes
k_3	Gouda	44 minutes

Table 5.5: Displacement of policy Cover

Taxi	Final destination taxi (d_k)	Time to finish tasks (t_{ck})
k_1	Alphen aan den Rijn	73 minutes
k_2	Rotterdam	0 minutes
k_3	Gouda	44 minutes

Table 5.6: Displacement policy of Cover and Relocate

Taxi	Final destination taxi (d_k)	Time to finish tasks (t_{ck})
k_1	The Hague	30 minutes
k_2	Alphen aan den Rijn	69 minutes
k_3	Gouda	44 minutes

Table 5.7 shows the request that is generated in R_2 with its origin and destination point.

Table 5.7: Requests R_2

Request	Origin point (o_r)	Destination point (d_r)	Time (t)
r_3	Voorburg	Rijswijk	25

Furthermore, Table 5.8 describes the results of the policies when serving R_2 from their previous displacements. These results indicate when a taxi has to serve a request and if a taxi should be relocated.

Table 5.8: Results of policies for requests in R_2

Results First Come First Serve	Results Cover	Results Cover and Relocate
	$x_{r_3k_2} = 1$	$x_{r_3k_1} = 1$
r_3 is not served	k_2 assigned to r_3	k_1 assigned to r_3
Penalty = 45	Penalty = 6	Penalty = 6

Table 5.9: Results of the example

	Penalty First Come First Serve	Penalty Cover	Penalty Cover and Relocate
R_1	7	13	7
R_2	45	6	6
Total	52	19	13

The results of this example are given in Table 5.9, which shows that policy Cover and Relocate achieved the best solution, after that policy Cover and policy First Come First Serbe has the worst result. This simple example proves that taking the coverage into account can have major consequences on the total penalty to be paid.

5.3. Results

The policies First Come First Serve, Cover and Cover and Relocate, described in Sections 4.2, 4.3 and 4.4, have been applied to the iterative process described in Section 4.1 with the initial conditions given in Section 5.3.1. For sensitivity analysis, the simulation also tested for different values of D while keeping the d_i fixed. During the simulation, it was recorded how often a request was lost because it was not reachable by a taxi on time. The total penalty for each day was recorded, Figure 5.3 shows the outcome of the 5 days. The results are obtained on DAS-5 (The Distributed ASCI Supercomputer 5) which is a six-cluster wide-area distributed system.

5.3.1. Initial Conditions

To simulate, we use a map of the 20 most populated cities in South Holland, as displayed in Figure 5.1. We divide 20 taxis over 4 cities, 5 taxis in Rotterdam, 5 taxis in The Hague, 5 Taxis in Leiden and 5 taxis in Zoetermeer. These 4 cities have been chosen because they have the most inhabitants and are well spread over the area. The iterative process uses time steps of 1 minute and we set a test for 5 days, because that is usually equal to one workweek. So we have 7200 time instants. The start of the iterative process starts on Monday at 00:00 and ends on Friday at 23:59. A city is considered to be covered if a taxi can reach it within 30 minutes, so, $\gamma = 30$ minutes. For the simulation we want to pick the frequency in such a way that the probability is around 0.5 that a taxi is unavailable. Then there is a request every 4 minutes, because we get a q value of 0.497 when λ is equal to $\frac{1}{4}$ and q is the probability that a taxi is unavailable. Table 5.10 summarizes the initial conditions mentioned above. As described in Section 5.1.3, we expect passengers not to wait more than 40 minutes, so $\beta = 40$ minutes.

Table 5.10: Initial conditions of computational study

Parameter	Settings
$\|K\|$	20
$\|I\|$	20
Δt	1 minute
$\|T\|$	7200
S	Monday at 00:00
F	Friday at 23:59
λ	1/4
q	0.497
γ	30 minutes
β	40 minutes
$tc_k \quad \forall k \in K$	0
$d_k \quad \forall k \in \{k_1, k_2, k_3, k_4, k_5\}$	Rotterdam
$d_k \quad \forall k \in \{k_6, k_7, k_8, k_9, k_{10}\}$	Leiden
$d_k \quad \forall k \in \{k_{11}, k_{12}, k_{13}, k_{14}, k_{15}\}$	The Hague
$d_k \quad \forall k \in \{k_{16}, k_{17}, k_{18}, k_{19}, k_{20}\}$	Zoetermeer

In order to determine how much the coverage should be considered when assigning taxis to requests, we carried out several measurements with different weighting factors. In the objective functions (4.5) and (4.10) the first term is multiplied by D, the greater the value of D, the less coverage is taken into account. For the value D we examined the values 40000, 1000000 and the number of inhabitants of the 20 cities in South Holland which is equal to 2463281.

5.3.2. Results Initial Conditions

Examining Figure 5.3, we observe that the policy Cover and Relocate (policy 3) has the lowest penalty, after that the policy Cover (policy 2) and the policy First Come First Serve has the highest penalty (policy 1). There is a significant difference between the different policies. The policy Cover and Relocate performs better than the other two policies. The policy Cover and Relocate experiences a constant increase in the penalty over time, while with policy Cover and First Come First Serve it is more variable. These results are robust even for different values of D (i.e. the trade off between waiting times and coverage). Table 5.11 and 5.12 show the penalty values of policy Cover and policy Cover and Relocate, for the different values of D.

Table 5.11: Results policy Cover for different values of D

	Penalty with $D = 40000$	Penalty with $D = 1000000$	Penalty with $D = 2463281$
Day 1	1315	1277	1283
Day 2	2288	2413	2518
Day 3	1358	1288	1332
Day 4	2045	1860	1817
Day 5	1958	1691	1718
Total	8964	8529	8668

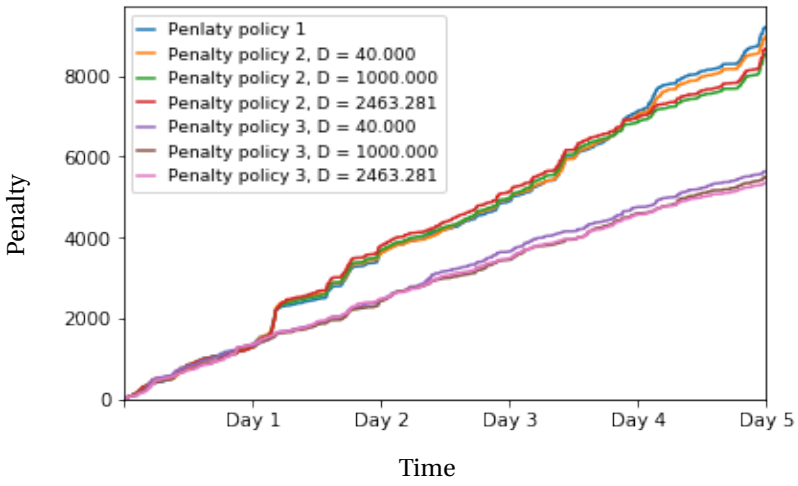

Figure 5.3: The course of the total penalty

If we examine Table 5.11 we observe that policy Cover with a value of 1000000 for D has the lowest total penalty and therefore has the best result. The Table shows that there is quite some difference between the total penalty for the different values of D (more than 400), so finding a good value for D for the objective function can certainly make a difference. With a value of 40000 for D, it achieves the best results on day 2 (while on the other days it got the worst results). This may be due to the fact that there were more requests that day and that therefore the coverage should be taken less into account. That would also explain the high penalties for that day.

Table 5.12: Results policy Cover and Relocate for different values of D

	Penalty with D = 40000	Penalty with D = 1000000	Penalty with D = 2463281
Day 1	1326	1346	1346
Day 2	1139	1086	1086
Day 3	1189	1017	971
Day 4	1093	1143	1110
Day 5	888	903	933
Total	5635	5495	5348

Table 5.12 shows that the value of all inhabitants of all cities (2463281) for D has the best result. With the policy Cover and Relocate there is less difference between the results for the different values of D in comparison with the policy Cover. It also varies a lot for which value of D has the best result on a day. To compare the three different policies, we put the results of policy First Come First Serve, Cover and Cover and Relocate with their best results for D, in Table 5.13.

Table 5.13: Results of the three policies

	Penalty First Come First Serve	Penalty Cover	Penalty Cover and Relocate
Day 1	1354	1277	1346
Day 2	2271	2413	1086
Day 3	1277	1288	971
Day 4	2198	1860	1110
Day 5	2104	1691	933
Total	9204	8529	5348

Examining the results in Table 5.13, we observe that the policy Cover and Relocate has the lowest penalty. After that, the policy Cover while the policy First Come First Serve has the highest penalty. It is interesting to note, that after the first day, there is still little difference between the results of the policies. During the simulation, the differences becomes larger. This implies that it takes some time before considering the coverage

becomes useful. This could be because it takes a while before the taxis are re-positioned in such a way that a good coverage of the taxis is achieved. Obtaining good coverage is also at the expense of minimizing waiting times when you assign taxis to requests, which is why policy First Come First Serve does not get worse results in the beginning than the other policies. It is also interesting to notice that policy Cover and Relocate gets much sooner better results than policy First Come First Serve, in comparison with policy Cover. The policy Cover is only getting after 3 days better results than First Come First Serve, while Cover and Relocate is already performing much better than First Come First Serve after day 1. This might be because policy Cover and relocate has obtained good coverage more quickly than policy Cover and therefore it will get sooner better results. To compare the policies with each other it is also useful to examine the number of unserved requests and the computation time of the policies, that are shown in table 5.14.

Table 5.14: Number of unserved requests and computational time of the policies

	First Come First Serve	Cover	Cover and Relocate
Number of unserved requests	31	25	4
Average time to compute in seconds	0.158	1.678	25.820

Table 5.14 shows that Cover and Relocate also has the least unserved requests, after that Cover and First Come First Serve has the most unserved requests. So Cover and Relocate performs best, which is understandable because for every unserved request, the highest penalty is paid. Table 5.14 also shows that there is a huge difference between the computation time of the different policies. The difference between the policy Cover and Relocate and First Come First Serve is more than a factor 100. The difference between the policy Cover and First Come First Serve is more than 10. This can be problematic if one is working with larger areas and more taxis, because the computing time will become even larger, especially for Cover and Relocate, it could take too much time to find the optimal solution.

5.3.3. Results Larger Number of Taxis and Higher Frequency

We examined if we get the same results if we increase the number of taxis and the frequency in such a way that the probability that a taxi is not available remains the same. The number of taxis that are dispatched over the cities is now equal to 35. The last 15 are distributed over the next largest cities, so 5 in Dordrecht, 5 in Alphen aan den Rijn and 5 in Schiedam. Then we set λ equal to $\frac{153}{350}$, because with this value for λ, q is equal to 0.497. Figure 5.4 and Table 5.15 show the results.

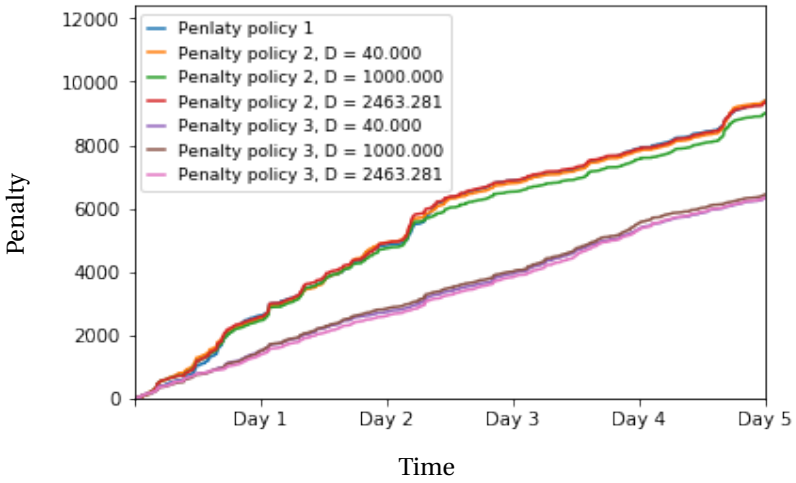

Figure 5.4: The course of the total penalty with a higher frequency and a larger number of taxis

If we observe Figure 5.4, it is noticeable that the results are similar as the simulation with the initial con-

ditions described in Section 5.3.1. In this case the policy Cover and Relocate has also the lowest total penalty and therefore the best result, after that Cover has the lowest total penalty and First Come First Serve has the highest total penalty. With these results, Cover and Relocate also experiences a constant increase. So it does not affect the results much, if the number of taxis and the frequency of the requests is increased, but only if that happens in such a proportion that the probability that a taxi is unavailable remains the same. The policies Cover and Relocate and Cover got almost the same results for different values for D. It also holds that policy Cover and Relocate achieved the best result for D = 2463281 and Cover also achieved the best result for D = 1000000. To compare the three different policies, we put the results of policy First Come First Serve, Cover and Cover and Relocate with their best results for D, in Table 5.15.

Table 5.15: Number of unserved requests and computational time of the policies

	First Come First Serve	Cover	Cover and Relocate
Number of unserved requests	6	7	1
Average time to compute in seconds	0.488	5.082	92.305

What noticeable is when we compare Table 5.15 and 5.14, is that the computational time is now much longer than in the previous simulation, while only 15 extra taxis are dispatched. This could mean that the computational time can increase exponentially if the number of taxis and the frequency are increased. The number of unserved requests is less, this may be due the fact that there are more taxis and that the probability that there is a taxi nearby that could reach it in time, is higher.

5.3.4. Results Higher Frequency

It is also interesting to examine a situation in which the taxis are constantly busy, so the frequency is so high that the taxis can no longer be stationed at a city. We did a simulation with the same initial conditions as explained in Section 5.3.1 but we set the frequency so high that there is a request every 2 minutes, so λ is equal to 1/2. Figure 5.5 shows the result.

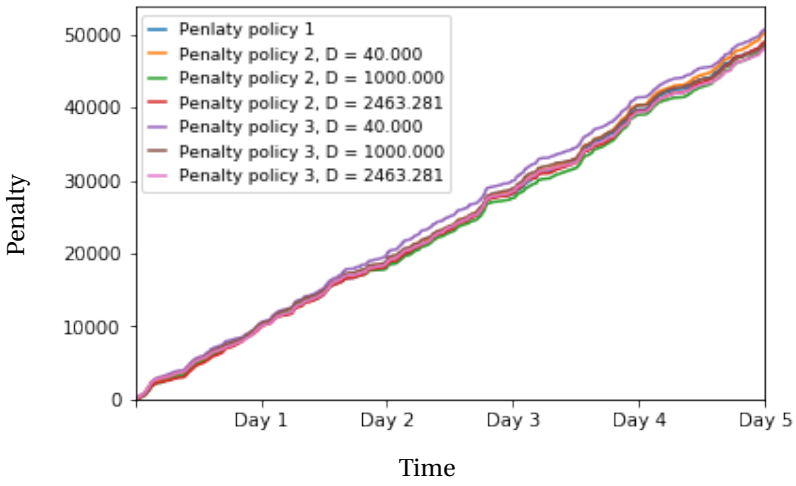

Figure 5.5: The course of the total penalty with a high frequency of requests

All policies have almost the same total penalty. Also, there is not much difference in the course of the total penalty over time. This could be due to the fact that when a taxi is almost finished with a request, that now the probability is much higher that there will be a new request close to him due to the high frequency. This is not the case when there are less requests. This makes the coverage of the taxi less important. Finally, policy Cover and Relocate does not really have the chance to relocate the taxis to get a good coverage because, to relocate a taxi, it has to be stationed in a city while now taxis are so busy that this hardly happens.

6

Conclusion

In this thesis we have studied the impact of coverage when assigning taxis to requests, by comparing different policies that regulate the dispatch of taxis to requests. One of these policies did not consider coverage while the others did. To see to what extent taking coverage into account can benefit the waiting time of clients, we have created two policies. The first policy only takes the coverage into account when assigning a taxi to a request while, with the other one, taxis can also be relocated when not linked to a request. These policies are then compared in a simulation to examine what the best option is.

In Section 5.3 clarify that considering the coverage in a policy for dispatching taxis, certainly has a positive effect on the outcome. Especially if taxis are relocating for maximizing the coverage. We have also observed that it takes some time before considering the coverage can be beneficial and that it does not immediately deliver better results. Taking into account the coverage also works best when the frequency is not too high. However, Table 5.14 and 5.15 show that the computation time of the policies considering the coverage is higher than the computational time of the policy that does not consider the coverage. This can cause problems if these decisions have to be taken quickly.

In order to expand this research for the optimal dispatch of taxis, we advise to further the research what a good trade off would be and what the relation would be between a good trade off and the frequency of the requests. It is also worth to further study on how to relocate taxis for total coverage. In this thesis, we relocate taxis only among adjacent cities. For example, consider the set of cities that are located at a certain distance from the current city. In my thesis, it is the case that if a taxi relocates, it must first drive to the city where it relocates to, before it can drive back, so it is interesting to find out which better results van be achieved if that was not the case.

Bibliography

Jinbo Bi, Kristin Bennett, Mark Embrechts, Curt Breneman, and Minghu Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:1229–1243, 2003.

Grace M. Carter, Jan M. Chaiken, and Edward Ignall. Response areas for two emergency units. *Operations Research*, 20(3):571–594, 1972.

Han-Wen Chang, Yu-Chin Tai, and Jane Yung-Jen Hsu. Context-aware taxi demand hotspots prediction. *International Journal of Business Intelligence and Data Mining*, 5(1):25, 2010.

Richard Church and Charles Revelle. The maximal covering location problem. *Papers in Regional Science*, 32 (1):101–118, 1974.

Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

Mark S. Daskin. A maximum expected covering location model: formulation, properties and heuristic solution. *Transportation Science*, 17(1):48–70, 1983.

Nasser A. El-Sherbeny. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University-Science*, 22(3):123–131, 2010.

C.J. Jagtenberg, Sandjai Bhulai, and R.D. Van der Mei. An efficient heuristic for real-time ambulance redeployment. *Operations Research for Health Care*, 4:27–35, 2015.

Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

Der-Horng Lee, Hao Wang, Ruey Long Cheu, and Siew Hoon Teo. Taxi dispatch system based on current demands and real-time traffic conditions. *Transportation Research Record*, 1882(1):193–200, 2004.

Bin Li, Daqing Zhang, Lin Sun, Chao Chen, Shijian Li, Guande Qi, and Qiang Yang. Hunting or waiting? discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 63–68. IEEE, 2011.

Ziqi Liao. Taxi dispatching via global positioning systems. *IEEE Transactions on Engineering Management*, 48 (3):342–347, 2001.

Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.

Santi Phithakkitnukoon, Marco Veloso, Carlos Bento, Assaf Biderman, and Carlo Ratti. Taxi-aware map: Identifying and predicting vacant taxis in the city. In *International Joint Conference on Ambient Intelligence*, pages 86–95. Springer, 2010.

Constantine Toregas, Ralph Swain, Charles Revelle, and Lawrence Bergman. The location of emergency service facilities. *Operations Research*, 19(6):1363–1373, 1971.

Pieter L. van den Berg and Karen Aardal. Time-dependent $MEXCLP$ with start-up and relocation cost. *European Journal of Operational Research*, 242(2):383–389, 2015.

K.I. Wong, S.C. Wong, and Hai Yang. Modeling urban taxi services in congested road networks with elastic demand. *Transportation Research Part B: Methodological*, 35(9):819–842, 2001.

Hai Yang and S.C. Wong. A network model of urban taxi services. *Transportation Research Part B: Methodological*, 32(4):235–246, 1998.