Using Column Generation for the Bus Line Planning Problem

Master Thesis

Econometrics and Management Science

Kai Huiskamp*

Erasmus School of Economics
Erasmus University Rotterdam

December 2016

Abstract

The *Line-planning problem* is one of the most important tasks in the planning of a bus line network. It consists of finding an optimal set of bus lines and passenger routes. Therefore, we use two objectives: we minimize the transport operating costs, and we minimize the traveling times of the passengers. In this thesis, we present a new model for the bus line planning, making use of column generation. The main contribution of our model is, in comparison to the existing models, the way we deal with generating new potential bus lines by using practical insights. Furthermore, we discuss the impact of the start set of bus lines, we investigate the complexity of the problem and we consider multiple scenarios, where each scenario is defined by using different input parameters. Computational results are shown for theoretical data as well as for data for the region of Breda, The Netherlands.

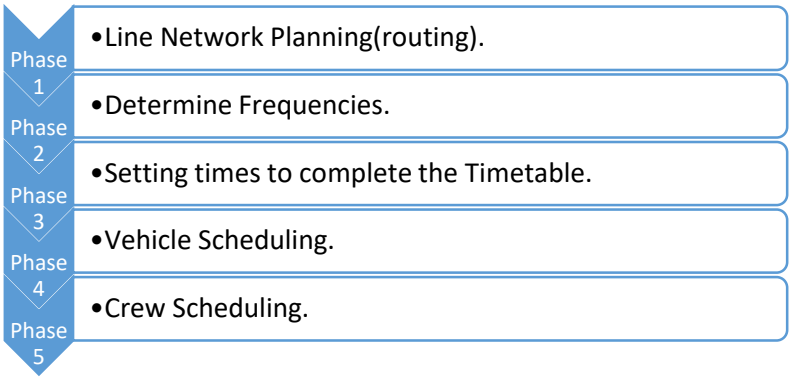
Key words: public transportation; line planning; column generation; longest path

* Student Number: 348882kh, supervised by Twan Dollevoet & Paul Bouman

Content

1. Introduction

Bus planning, the process of going from a blank map to a complete bus line network including a timetable, equipment schedule and personnel schedule, is the most important task of the public transport companies. A good timetable with corresponding bus lines can lead to huge savings in costs and can give a proper service level to the customers. Within the problem of bus planning, there are (in general) 5 different phases to determine the final timetable and resource schedule. These phases are as follows:



The first phase, routing, is one of the most important steps. But, the current bus lines of Arriva (and a lot of other companies) are based on historical decisions and are updated by local analysis instead of (econometric) investigations. Therefore, in this thesis, we develop a method for creating the bus lines based on mathematical formulas and optimization processes. The data provided by the 'OV-Chipkaart' will be the input of the method. This data gives the best known information about the demand for bus transport by giving us a great insight in the origin-destination data (OD-matrix).

So, the problem in this thesis is to create a new bus line network, which ensures that each passenger can reach its destination without violating certain service level requirements. The data of the passenger demand is given by the OD-matrix. We have to take care of two objectives: we want to minimize the operating costs of lines, but we also want to minimize the customer discomfort. In many articles, the total passenger traveling time and/or the number of transfers is used as a measurement of the customer discomfort. When modeling this problem, we made use of the commodity flow model given in Borndörfer, Grötschel & Pfetsch (2007), which indeed takes into account the total passenger traveling time and the transport operating costs. It generates the bus lines by making use of column generation, and the passengers can make use of different passenger paths. The new factor in this article is the way we deal with the creation of new bus lines. The pricing problem for the bus lines is a longest path problem, which is NP-Hard, and therefore we need a heuristic.

We identify multiple contributions of this project. First of all because, within the company of Arriva, a research like this is never done. The 'OV-Chipkaart' data is not yet used in combination with mathematics to create a bus line network. Furthermore, there is a lot of literature about bus planning, but the existing literature is mostly describing one part of the problem: vehicle and crew scheduling, instead of line planning.

The paper is organized as follows. In chapter 2 we will summarize the existing literature about the line planning problem so far. In section 3 we will relate the line planning problem to the practical problems within Arriva and Zight. After that, our model approaches are discussed in chapter 4. In chapter 5, the two different data cases, a fictional problem as well as a practical problem for the region of Breda, Netherlands, are described. The results of these two cases are presented in chapter 6. We end with the conclusions in chapter 7 and further research in chapter 8.

2. Literature

The line planning problem is discussed in many articles. In this chapter we will give a short overview of these different articles. The many given models can be roughly classified into four types. First, you have to choose between working with or without a line pool. Then, you also have to determine whether using a cost-approach or a customer-approach.

In a cost-oriented approach for the public transportation company, companies wants to minimize cost while still maintaining a certain service level. Working without a line pool while using the cost-oriented approach is the base of the first methods that are used. The idea is to create smaller line parts, and combine these parts into main bus lines, which is often an interactive process. The 'Skeleton' method (Silman, Barzily & Passy, 1974) is an early example of this method. They combined the endpoints and the intermediate nodes of a route by defining shortest paths. The weights of these paths are determined by characteristics as their length or traveling time. In a similar way, but making use of the customer-approach, Quak (2003) constructed lines by combining multiple small lines with a high demand: the so called k-lines. Another example of working without a line pool is the article of Szeto & Wu (2011). They solved not only a bus line planning problem, but also a frequency setting problem. The main disadvantage of these methods is that they are not considering an exact solution.

Working with a line pool can be considered as a two-step approach: in the first step, the user creates a set of bus lines, making use of the given data. In the second step, bus lines are chosen form this set to create a line plan. Ceder & Wilson (1986) worked with a line pool by generating lines in an iterative method. For each generated line, it holds that the length of this line is within a certain factor from the length of the shortest path. Optimizing such a set by using a local search strategy is introduced by Mandl (1980) and Israeli & Ceder (1995) introduced a quadratic set covering approach. The main disadvantage of these model is that, when not generating all the possible lines, the model will never solve the problem to optimality.

The main articles that we use are more based on advanced integer programming techniques. For example, a branch-and-cut approach is used by Goossens, van Hoesel, & Kroon (2006). They solved a practical railway problem within a reasonable calculation time. Schöbel & Scholl (2005) and Scholl (2005) described a Dantzig-Wolfe decomposition approach. Their objective is to minimize the number of transfers and the total transport time. The best related article using integer programming techniques is the one from Borndörfer et al. (2007). They minimize both the total bus line distance and the total passenger path distance (without taking into account transfers). In their formulation they created a decision variable which selects the optimal passenger paths and they created decision variable which selects the optimal bus lines. The authors show that the resulting pricing problem is NP-hard (Borndörfer et al., 2007)**.** They generate the line pool by using a column generation approach. A special case in this article is the scenario without capacities.

Besides solving the line planning problem for the bus system, there is one development we want to point out: system split. System split means that the customers can make use of different types of transport. These passengers are then distributed over different 'systems', assuming that they are always choosing for the fastest system. Bouma & Oltrogge (1994) took the system split into account, when developing a branch-and-bound based software system, which was used by the Dutch railway network.

During our research we also made use of more general articles. First of all, we applied the article of Lahaie (2008), who clearly explained the process from a primal formulation to a dual formulation. A great overview of all the methods and models is described by Schöbel (2011) while some really useful tips about network design are given in the article of Nielsen & Lange (2007).

3. Problem definition

At the end of 2014, Arriva has won the concessions for performing the public transport in Noord-Brabant for the next two years. These two concessions consist of the eastern and western part of North-Brabant. Due to the small margins in the public transport sector, there is a need to find ways to intelligently increase the financial efficiency, while maintaining quality and ridership. Therefore, the main (practical) question of this research will be:

> ***How can Arriva, using data from the 'OV-Chipkaart' system, increase the profitability in Public Transport in North-Brabant, while maintaining a basic service level and ridership?***

During the internship, we investigated several methods which can give an answer to this question. These methods vary from line planning to vehicle scheduling (see figure 1). Because Arriva and Zight are much more interested in the first three stages, and because especially the line planning problem has not been studied that much, the focus of our research will be on this line planning problem.

The line planning problem (LPP) we want to solve consist of 2 objectives: minimize the total distance traveled by the passengers and minimize the total costs of the bus lines. The distanced traveled by passengers is defined by making use of passenger paths. A passenger path is a sequence of bus stop connections (arcs) that a passenger can use to arrive at its destination. For each group of passengers with the same origin and destination (OD-Pair), a lot of passenger paths are possible. The model will determine the optimal path for each OD-Pair.

The total costs of the bus lines are determined by the length of the bus lines plus a fixed cost for each bus line used. Furthermore, a bus line must satisfy the following requirements:

- A bus line is a sequence of directed arcs. This means that a bus line can contain the arc (u, v) without containing the arc (v, u).
- Furthermore, the arc (u, v) is never followed by the arc(v, u).
- A bus line will never use the same arc twice (or more) during the same ride.
- A bus line will also not serve a bus stop more than two times during the same ride.

For now, a bus line is not restricted on the number of arcs. The line planning problem can now be mathematically formulated as follows:

$$Min\left[\lambda * \sum_{p\in P} y_p * \tau_p\right] + [(1-\lambda) * \sum_{l\in L} x_l * (c_l + F)] \quad (1)$$

$$S.t.$$

$$\sum_{p\in P_{st}} y_p = d_{st} \qquad \forall\, st \in D \quad (2)$$

$$\sum_{p\in P_a} y_p - K * \sum_{l\in L|a\in l} x_l \leq 0 \qquad \forall\, a \in A \quad (3)$$

$$y_p \in \mathbb{N}$$

$$X_l \in \{0,1\}$$

The variables, parameters and sets that are used in the LP formulation are as follows:

Sets:

A: $set\ of\ arcs, an\ arc\ is\ a\ link\ between\ 2\ bus\ stops.$

D: $set\ of\ OD\ pairs, an\ OD\ pair\ means\ that\ there\ is\ demand\ to\ go\ from\ bus\ stop\ s\ to\ bus\ stop\ t.$

P: $set\ of\ all\ passenger\ paths, every\ path\ is\ a\ sequence\ of\ arcs.$

P_{st}: $set\ of\ passenger\ paths\ corresponding\ to\ OD\ pair\ (s,t).$

P_a: $set\ of\ passenger\ paths\ which\ are\ 'using'\ arc\ a.$

L: $set\ of\ bus\ lines, each\ busline\ is\ also\ a\ sequence\ of\ arcs.$

Decision Variables:

y_p: $number\ of\ passengers\ which\ are\ making\ use\ of\ path\ p.$

x_l: $binary\ variable\ which\ determines\ if\ line\ l\ is\ selected\ or\ not.$

Parameters:

λ: $scaling\ factor\ for\ the\ bus\ line\ costs\ versus\ the\ passenger\ path\ costs$

τ_p: $length\ of\ passenger\ path\ p.$

c_l: $length\ of\ busline\ l.$

d_{st}: $demand\ of\ passengers\ travelling\ from\ bus\ stop\ s\ to\ bus\ stop\ t.$

F: $Fixed\ Cost\ of\ a\ busline$

K: $Capacity\ of\ a\ busline, because\ no\ capacity\ is\ used, K\ is\ equal\ to\ the\ total\ demand$

The objective (1) consists of two parts: the total length plus the fixed costs of the chosen bus lines and the total length of the chosen passenger paths. Because the length of the passenger paths are multiplied by the number of passengers that are using the paths, we introduced a scalar λ. This scalar makes sure that both summations are in the same range, and is calculated as follows:

$$\lambda = \frac{\sum_{l \in L} c_l}{\sum_{st \in D} D_{st} * \tau_{p(st)} + \sum_{l \in L} c_l}$$

In this equation, $p(st)$ is the shortest path going from bust stop s to bus stop t . Furthermore, the two constraints are straight forward: (2) takes care that there is a passenger path for each origin-destination pair and (3) ensures that there is at least one bus line covering an arc that is used by one or more passenger paths.

4. Methods

As said before, there are two options to solve the line planning problem. We can start with an empty network and create the lines during the optimization process, or we can work with a line pool. The methods we used are based on a line pool: given a set of possible bus lines, choose those lines that are optimizing our objective.

Given that we have chosen for the line pool option, we can solve the line planning problem with two different approaches. The first approach is the exact one: it solves the problem to optimality. The second approach is making use of a heuristic. We will describe the exact approach in section 4.1 and the heuristic approach in section 4.2

4.1 Exact approach

The exact method will solve the problem to optimality. But, due to the exponentially rising calculation time, a lot of problems cannot be solved exactly. Most researchers believe our line planning problem can also not be solved in polynomial time. Because they believe that the solving time for the line planning problem rises exponentially, we need to use a heuristic. This heuristic is based on column generation: we will start with a small set of lines and will add those lines that can possibly optimize the objective function. This heuristic (algorithm) is described in the next section.

4.2 Bus line Network Algorithm: Column Generation

Because the number of possible bus lines rises exponentially with the number of bus stops, we will use column generation for solving the line planning problem (see Barnhart, Johnson, Nemhauser, Savelsbergh, & Vance (1998) for an introduction). By making use of column generation, we solve the LP relaxation of the (integer) line planning problem. In the LP relaxation the variables x_l and y_p are no longer integer variables. By making use of column generation, we will only add those lines to our line pool which can give us a lower objective value.

4.2.1 Dual Problem

During the column generation method we will solve two pricing problems. These two pricing problems are determined by using the dual problem of the LP relaxation. The variables of the dual problem are as follows: $\pi_{st}, st \in D$ for the demand constraints (2) and $\mu_a, a \in A$ for the capacity constraints (3). One of the most important properties of a dual problem is:

Proposition 4.1. *If the primal problem has a global optimum, then the dual problem will also have a global optimum and, even stronger, the objective values will be the same.*

So, solving the dual problem gives us the same objective value as the primal problem. The formulation of the dual problem is as follows:

$$Max \ \sum_{s,t \in D} \pi_{st} * d_{st} \qquad (1)$$

$$S.t.$$

$$\pi_{st} - \sum_{a \in p} \mu_a \leq \lambda * \tau_p \qquad \forall\, p \in P_{st}, st \in D \qquad (2)$$

$$K * \sum_{a \in l} \mu_a \leq (1 - \lambda) * (c_l + F) \qquad \forall\, l \in L \qquad (3)$$

In this formulation, π and μ are the decision variables. The dual problem turns out to be a maximization problem, therefore, the goal is now restrict π and μ in such a way that the optimal (maximum) solution of the dual problem is as low as possible. This can be done by generating those paths and lines that are making the restrictions tighter. When generating these paths and lines we make use of two sub-problems, the so called pricing problems.

4.2.2 Pricing Problems

The heuristic needs as input a (initial) set of bus lines and passenger paths. These two sets must give a feasible solution, otherwise the input variables of the pricing problems do not contain any values and the pricing problems cannot be solved. A feasible solution means that there must be at least one passenger path for each OD-Pair and these paths must be covered by the bus lines. After a certain set of lines and paths gives us a solution (which is optimal given that set), we want to check if this optimum is also a global optimum. To check the optimality of an LP solution a sub-problem called 'pricing problem' is solved. The pricing problems tries to identify columns to enter the line pool and/or path pool. If such columns are found, they are added to the set of lines and paths and the LP relaxation and dual problem is re-optimized. The two pricing problems of our problem are described in the next two sections.

4.2.2.1 Pricing Problem Passenger Paths

First, we discuss the pricing problem of the passenger paths. The reduced cost are given by $\overline{\tau_p}$, and are calculated as follows:

$$\overline{\tau_p} = \lambda * \tau_p - \pi_{st} + \sum_{a \in p} \mu_a = -\pi_{st} + \sum_{a \in p} (\mu_a + \lambda * \tau_a)$$

Thus, the corresponding pricing problem consists of finding a path p from OD pair(s, t) such that

$$-\pi_{st} + \sum_{a \in p} (\mu_a + \lambda * \tau_a)$$

is minimized and smaller than zero. This problem is a minimum weighted path problem with positive weights $\mu_a + \lambda * \tau_a$ and is therefore solvable in polynomial time.

4.2.2.2 Pricing Problem Bus lines

Secondly, we discuss the pricing problem for the line variables y_l. The reduced cost $\overline{c_l}$ for a variable y_l is:

$$\overline{c_l} = (1 - \lambda) * (c_l + \mathrm{F}) - K \sum_{a \in l} \mu_a = (1 - \lambda) * \mathrm{F} + \sum_{a \in l} (1 - \lambda) c_a - K \mu_a$$

Thus, the corresponding pricing problem consists of finding a path l such that

$$(1 - \lambda) * \mathrm{F} + \sum_{a \in l} ((1 - \lambda) c_a - K \mu_a)$$

is minimized and smaller than zero. This problem is also a minimum weighted path problem, but now, because $c_a - K\mu_a$ can be negative, the final weights can also be negative. Therefore, the pricing problem for the line variables can also be written as a maximum weighted path problem and is therefore NP-hard (Garey & Johnson, 1979). Another problem when solving the pricing problem is the possible presence of negative cycles. If so, the problem becomes intractable.

Furthermore, Borndörfer et al. (2007) show that the LPP is NP-hard and therefore we need to develop a heuristic which solves the line pricing problem in polynomial time. For this algorithm, we make use of two basic shortest path algorithms. The most famous algorithm to solve the shortest path problem is the algorithm of Dijkstra (Dijkstra, 1959). But, the algorithm of Dijkstra is not able to handle negative weighs. Therefore, we choose to use the algorithm of Bellman-Ford, see Bellman (1958) and Ford & Fulkerson (1956). They proposed an algorithm that is able to solve a shortest path in polynomial time, even if the graph contains negative arc weights. If a network consist of one or more negative cycles, the algorithm of Bellman-Ford will find these cycles but is of course not able to solve the shortest path problem.

When looking at the weights of the bus line pricing problem, there is a probability for negative cycles in the graph. Therefore, we need to adjust the algorithm of Bellman-Ford, so that it can handle negative cycles. In Borndörfer et al. (2007), a maximum of k arcs is used to prevent the algorithm to get stuck. In the algorithm we propose, we make us of the logic of a bus line, as described in chapter 3. Because the possible new bus lines are restricted on the number of duplicate nodes and arcs, the algorithm can never get stuck in a (negative) cycle. Now, if there is a path with negative length, the algorithm will find it and will return a bus line with an overall length smaller than zero. Notice that this path will only be chosen if the reduced cost are also smaller than zero, in other words, if the length is smaller than $(1 - \lambda) * \mathrm{F}$.

Below, the adjusted Bellman-Ford algorithm based on this idea is shown. In the first five lines the heuristic is initialized: only the distance to the source vertex is known and for none of the vertices a path to these vertices is known. Then, from line 7 the shortest paths are determined. Each iteration, the algorithm determines if the length of the path from the source to vertex u, plus the length of arc (u, v), is shorter than the length of the current path from the source to vertex v. Normally, if this is true the path of v will be updated. But, in the adjusted algorithm the two extra restrictions are implemented. In line 12 the amount of times vertex v is in $path\ u$ is calculated and in line 13 the amount of times arc (u, v) is in $path\ u$ is calculated. Now, the path to v is only adjusted if:

- The new distance to node v is shorter
- Adding arc (u, v) to path u does not result in a bus line that violates the characteristics of a bus line

Input:

A graph, represented as lists of *vertices* and *arcs*

A vertex $source$, the starting node.

Variables:

Boolean *optimal*, which is true until it is shown that the current solution is not optimal.

Output:

A list with paths *predecessor*, containing the current shortest paths for each vertex
A list *distance*, containing the current shortest distances for each vertex.

Procedure:

```
1   for each vertex v in vertices do
2       distance[v] := inf              // At the beginning , all vertices have a weight of infinity
3       predecessor[v] := null          // And a null predecessor
4   end for
5
6   distance[source] := 0               // Except for the Source, where the weight is zero
7
8   for i from 1 to size(vertices)-1 do
9       optimal := true
10      for each edge (u, v) with weight w in edges do
11
12              if distance[u] + w < distance[v] & adding (u, v) does not violate the bus line then
13                      Update predecessor[v], distance[v]
14
15                      optimal := false
16              end if
17      end for
18  end for
19
20  return distance, predecessor
```

4.2.3 Main Algorithm

Making use of the pricing problems, we determined the following algorithm to find the best possible path pool and line pool:

Input:

A list of bus stops, each bus stop is a *node*
A list of bus stop connections, each connection is an *arc*
A start set of lines, each line is represented as a list of *arcs*
Pricing problems for the *passenger paths* ($PB_{PassengerPaths}$) and the *bus lines* ($PB_{BusLines}$)

Variables:

Parameter *reduced_costs*, which is determined by the pricing problems

Output:

A set of lines, containing the current lines and the new lines
A set of passenger paths

Procedure:

Create start set of passenger paths

Solve $PB_{PassengerPaths}$
Solve $PB_{Buslines}$

$reduced\_costs := \min(\mathrm{PB}_{\mathrm{PassengerPaths}}, \mathrm{PB}_{\mathrm{Buslines}})$

While $reduced\_costs$ < 0 **do**

    if $PB_{PassengerPaths}$ < 0 **then**
        Update Passenger Paths
    end if

    if $PB_{Buslines}$ < 0 **then**
        Update Bus Lines
    end if

    Solve Dual Problem
    Solve $PB_{PassengerPaths}$
    Solve $PB_{Buslines}$

    $reduced\_costs := \min(\mathrm{PB}_{\mathrm{PassengerPaths}}, \mathrm{PB}_{\mathrm{Buslines}})$

End while

In line 1 the start set of passenger paths is created. For this set of paths we used the following logic:

- Use the network that is determined by the current set of bus lines. Create for each OD pair the shortest path in this network.
- If the current bus lines do not cover all arcs: use the network with all arcs in it. Now, create again for each OD pair the shortest path, using the whole network. If a path from an OD pair is not already in the set of passenger paths, add it to the set of passenger paths.

Now that we have an initial set for both the bus lines and the passenger paths, we can solve the passenger path and bus line pricing problems. If the reduced costs of these pricing problems is negative, we have found a bus line and/or passenger path that improves the solution of the original problem. In this case, we enter the while loop and update the sets by adding the columns with the most negative reduced costs before solving the pricing problems again. We continue with these iterations until we do not find any improving column anymore.

Proposition 4.1. *The used column generation algorithm does not solve the relaxed LP to optimality*

First of all, the adjusted Bellman-Ford Algorithm will not always find the optimal bus line, i.e. the bus line with the shortest path that is not violating the characteristics. But, if there is at least one arc with negative weight, the fixed costs are equal to zero and all start nodes are allowed, the algorithm will always find a negative path. Therefore, the following proposition holds:

Proposition 4.2. *The used column generation algorithm does solve the relaxed LP to optimality, if the fixed cost of a bus line is equal to zero and all start nodes are available.*

Namely, if the fixed cost are equal to zero and all start nodes are available, the algorithm does find the optimal solution making use of more iterations and shorter bus lines. This is the case because by using a lot of short bus lines, even the smallest variable cost reductions are possible, without resulting in higher fixed costs.

However, shorter bus lines are not preferred and they can be avoided by using a fixed cost per bus line. Furthermore, if the fixed cost of a bus line are not equal to zero, it is possible that the Bellman-Ford algorithm finds a potential bus line with a negative distance that is lower than $F * (1 - \lambda)$. If this is the case, the bus line will not be added to the line pool. When using the algorithm, there are examples where there still exists a bus line with negative reduced costs which cannot be found by the adjusted Bellman-Ford algorithm. The algorithm terminates while there is still a better solution possible and therefore the algorithm does not always find the optimal solution.

4.3 Getting the integer solution

Going from solving the relaxed LP to solving the integer line planning problem takes a few more steps, because the solution of the relaxed LP does not have to be the same as the solution of the integer LP. To get to a proper integer solution, we use two approaches. The first one is described in section 4.3.1 and makes the solution less dependent on the initial set of bus lines. The second approach is a heuristic to get the final integer solution, based on fixing bus lines and performing the column generation algorithm with these fixed bus lines.

4.3.1 Making the solution less dependent on the start set of bus lines

As mentioned in section 4.2, the adjusted Bellman-Ford Algorithm will not always find the bus line with the highest reduced cost. Therefore, the bus lines which are added during the column generation phase are different for each initial set of bus lines and the final integer solution depends on the initial set of bus lines. Therefore, we choose to run multiple repetitions to get different line pools (and path pools). Each repetition is based on a different initial set of bus lines. The first repetition uses the current bus line network as input, from the second repetition we start with a random initial set of bus lines. Because we can only start with a set of bus lines which ensures that all passengers can reach their destination, the start sets are determined as follows:

```
1   StartBuslineSet := empty{}
2   BuslineSetComplete := 0
3
4   While (BuslineSetComplete == 0) do
5
6       Get random OD-Pair going from s to t
7       Create (shortest) line between source s and sink t
8       Add line to set of bus lines StartBuslineSet
9
10      If ( All OD-Pairs can reach their destination ) then
11              BuslineSetComplete := 1
12      End if
13
14  End while
```

The method randomly picks *OD-Pair*(s, t) in line 6, which is connected by the shortest path between both bus stops (line 7). We pick another OD-Pair until each passenger can reach his destination. The start set of passengers is determined in the same way as explained before: for each OD Pair the set is extended with his shortest path in the whole network plus his shortest path in the start bus line network. By using this method we hope that the final results do not depend on the initial set of bus lines anymore.

4.3.2 The Fixing Heuristic: Solving the column generating algorithm with fixed bus lines

After the model solved several iterations we have our set of bus lines and passenger paths. With these two sets, we solve the relaxed LP for the first time and fix the bus line with the highest fractional value in the relaxed LP. With this bus line fixed, we solve the dual problem and solve the column generation algorithm again. After solving the column generation algorithm, we solve the relaxed LP and fix the second line with the highest fractional value. We will continue with this method until there is no new bus line with a fractional value in the relaxed LP solution. It is important to note that when a bus line is fixed during an iteration of the fixing heuristic, the bus line is also fixed in all the following iterations. After the 'fixing heuristic' procedure more lines are fixed then necessary, therefore we solve the integer line planning problem for the last time, with only those bus lines that are fixed in the 'fixing heuristic' procedure

Because we fix lines during this procedure, we need to add a restriction to the model. The used (new) parameter for this restriction is as follows:

δ_l: $Binary\ parameter, equals\ 1\ if\ bus\ line\ l\ is\ fixed\ in\ the\ solution$

And the corresponding restrictions is:

$$x_l \geq \delta_l\ , \qquad \forall\ l \in L$$

An extra restriction in the primal formulation means an extra variable in the dual formulation. We denote this variable as β_l. The new dual formulation is as follows:

$$Max\ \sum_{s,t \in D} \pi_{st} * d_{st} + \sum_{\boldsymbol{l \in L}} \boldsymbol{\beta_l} * \boldsymbol{\delta_l} \qquad (1)$$

$$S.t.$$

$$\pi_{st} - \sum_{a \in p} \mu_a \leq \lambda * \tau_p \qquad \forall\ p \in P_{st}, st \in D \qquad (2)$$

$$K * \sum_{a \in l} \mu_a + \boldsymbol{\beta_l} \leq (1 - \lambda) * (c_l + F) \qquad \forall\ l \in L \qquad (3)$$

Because for a new potential line δ_l will always be equal to zero, β_l does not have any influence on the pricing problem for bus lines. Therefore, the pricing problem for bus lines will remain the same. If the generated bus line is exactly the same to a line that is already in the line pool, we continue with the algorithm without adding a new line to the line pool. The algorithm of this heuristic looks as follows:

```
1  While (There is a bus line with a fractional value) do
2      Solve LP Relaxation to get the fractional values
3      If( At least one line with fractional value > 0) then
4              Fix line l with the highest fractional value of x_l
5
6              Solve Column Generation Method to generate a new bus line
7      End if
8  End while
```

5. The Data Instances

We will use two data sets to evaluate the model. The first data set will be a fictional one with 15 bus stops. The second data set is from the company of Arriva and is covering the region of Breda. In the next two sections we will describe both networks using three different characteristics: the bus stops and the arcs, the set of current bus lines in the network and the $Origin - Destination$(OD) matrix.

5.1 Fictional Data Instance

This data set is taken from Jaramillo-Álvarez, Gonzalez-Calderon, C., & Gonzalez-Calderon, G. (2013). It consist of 15 bus stops.

5.1.1 The bus stops (nodes) and the arcs

Below, the network is shown. Because it is important that we use a directed graph, both directions of the arcs are shown.

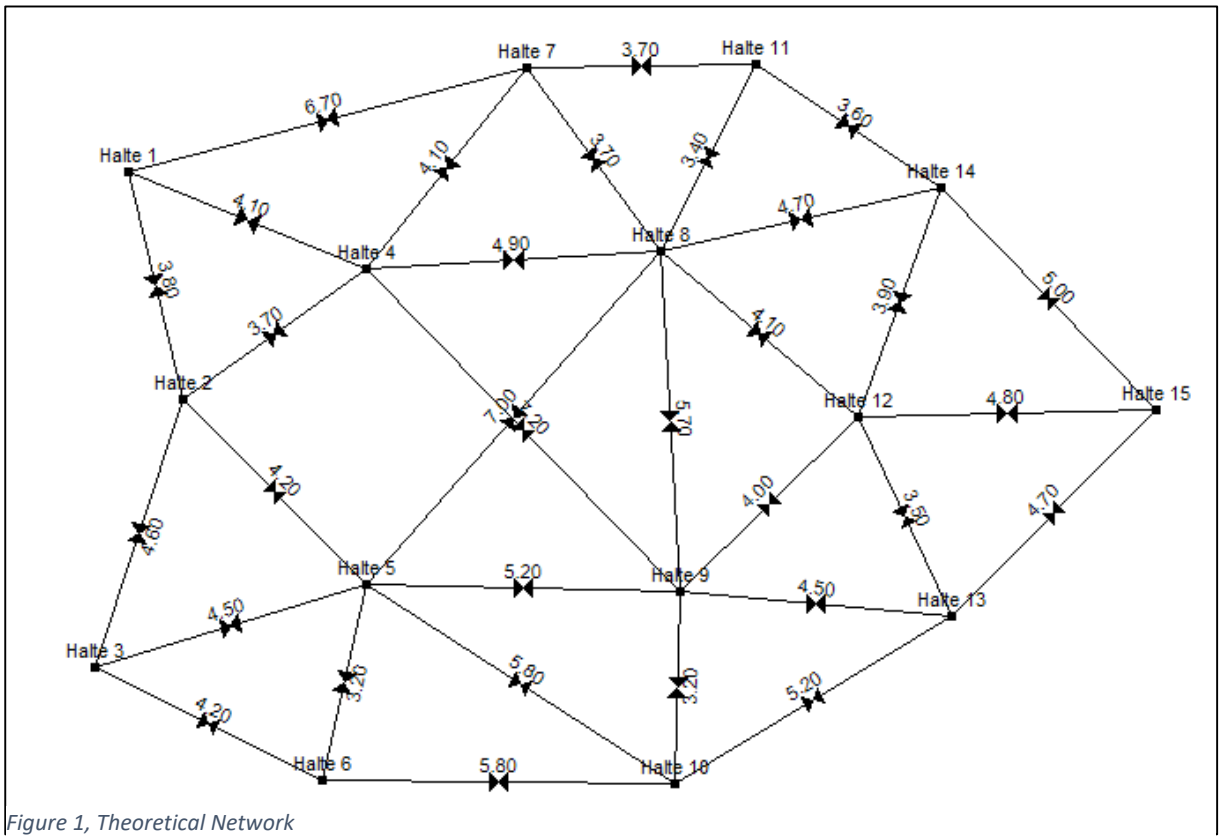


Figure 1, Theoretical Network

The 15 bus stops are connected by 64 arcs. The arcs are directed and symmetric: the distance/costs from a to b is the same as the distance from b to a.

5.1.2 Current Bus Lines

As described in the chapter 4, the model needs a set of 'current' bus lines. For this fictional network, we generated the bus lines as below. These bus lines are all starting in node 8. Important is that each node is covered at least once by the bus lines.

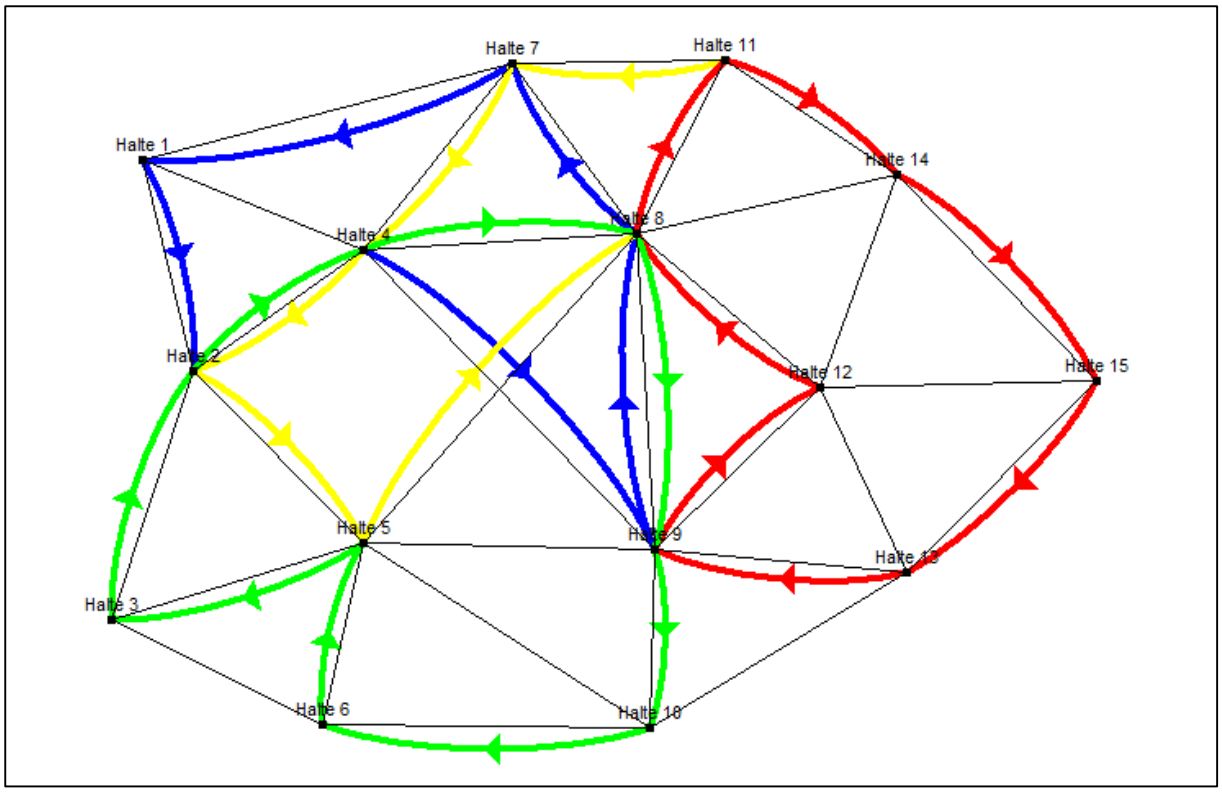


Figure 2, Current Bus Lines in Theoretical Network

5.1.3 Origin Destination matrix

We kept the origin destination matrix the same as in the article of Jaramillo-Álvarez et al. (2013). As you can see, each pair of bus stops has a positive number of customers. Also important is the fact that this matric is not symmetric. The origin destination (OD) matrix, which shows the number of passengers going from bus stop s to bus stop t, looks as follows:

From/To	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	50	20	35	47	15	18	26	21	5	12	10	35	34	30
1	17	0	3	20	24	26	13	5	68	12	2	34	16	25	31
2	1	8	0	29	54	55	1	14	25	4	8	8	21	2	50
3	26	5	17	0	35	29	30	2	31	54	85	17	62	17	33
4	51	53	87	41	0	68	95	64	51	19	9	10	14	25	36
5	12	20	18	65	11	0	20	35	24	31	15	15	32	30	65
6	69	17	21	24	19	24	0	37	21	25	26	65	13	15	14
7	19	15	28	35	51	32	17	0	81	19	12	7	10	20	18
8	16	24	27	35	39	47	54	44	0	27	10	15	20	29	7
9	20	34	35	10	60	20	22	20	10	0	15	50	60	15	15
10	15	14	25	30	35	30	50	70	31	30	0	44	16	13	24
11	10	20	35	11	60	50	27	4	10	7	10	0	10	21	30
12	26	30	20	91	10	90	40	60	19	8	31	30	0	19	7
13	14	15	27	10	22	2	20	28	7	60	13	31	6	0	11
14	50	15	10	20	10	7	18	33	10	35	20	8	60	70	0

Table 1, Origin Destination Matrix Theoretical Network

5.2 Practical data set: region of Breda

For the internship at Arriva/Zight, we were asked to investigate the region of Noord-Brabant. After some quick research, we found out that there were 3000 bus stops in this province. Because this amount of bus stops will lead to a huge computation time, we decided to focus on the region of Breda.

5.2.1 The bus stops (nodes) and the arcs

Within the region of Breda, there are two different types of bus stops: the bus stops in Breda, which are currently covered by the city bus lines, and the stops outside of Breda, served by the regional and faster bus network. To handle the two types of bus stops we made the following decisions, resulting in 222 bus stops:

- Each bus stop belonging to the city network of Breda is added to the network.
- The bus stops which are covered by the regional network are combined in several groups. Each group of bus stops is called a superstop. For example: all bus stops in Etten-Leur are combined to one big bus stop.
- The idea behind the superstops is as follows: when modeling the bus lines through the network of Breda, there will be lines that are going to other cities. For these lines and this problem, it does not matter how the bus lines are running through the other cities: it will only give us more calculation time.

The second question is how to connect the 222 stops. We use the current bus line network to answer this question. Each arc that is part of a bus line is added to the network. These arcs are directed, so, in the extreme situation, it can be that a bus can drive from A to B, but not the other way around. The resulting number of arcs is 451.

5.2.2 Current Bus Lines

As mentioned in the previous section we use the current bus lines, which are given by the company of Arriva. The current bus line network of the region of Breda is as follows:

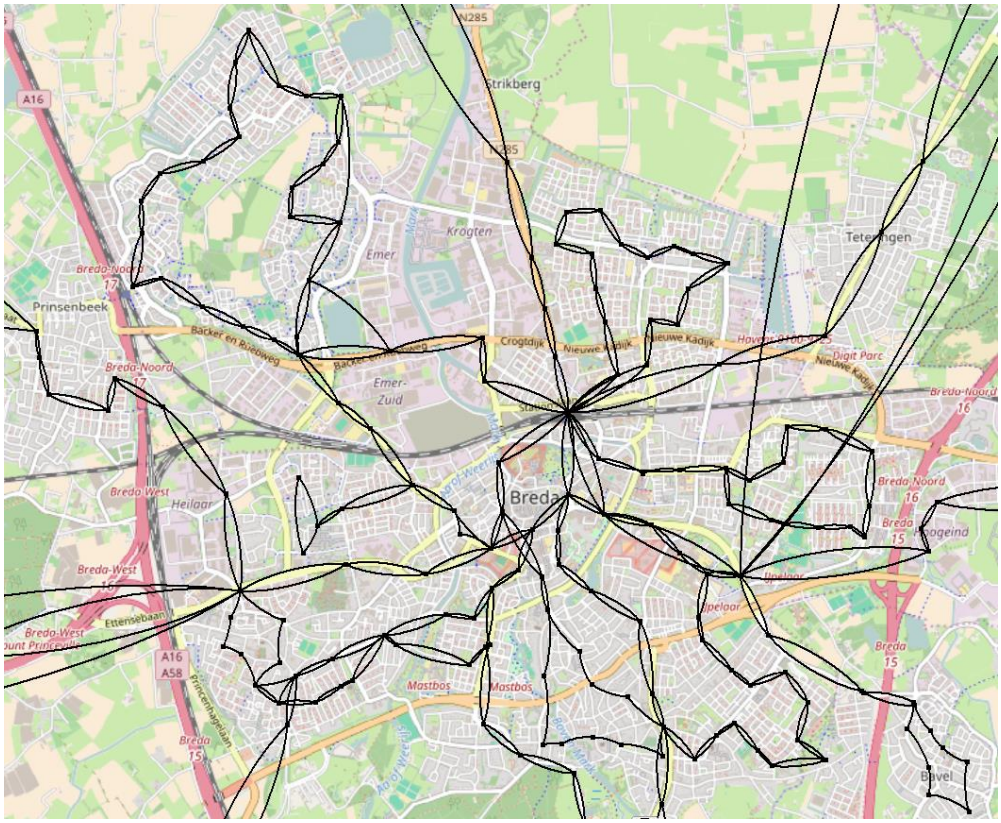


Figure 3, Current Bus Lines in Network of Region of Breda

5.2.3 Origin Destination matrix

From Arriva and Zight we got the passenger demand for the period January-August. We adjusted the demand from and to a superstop by taking the summation of the demand of the underlying bus stops. Because it can happen that passengers are already on their destination within a superstop, the demand of OD pair s, s can be larger than zero. After finishing the OD Matrix we came to the conclusion that there were a lot of s, t pairs with demand less than 200. This means that such a trip is made less than one time a day. To ensure a faster solution, we will only take those pairs in account, which have got a higher demand than 200 passengers. This resulted in 2396 OD Pairs. To get a better view on which OD Pairs have the highest demand, we use the figure below. In this figure, only OD Pairs with a demand higher than 5000 are connected and shown. It is clear that the weighted geographical midpoint is around Breda Central Station. The other crowded bus stop is Breda Centrum.

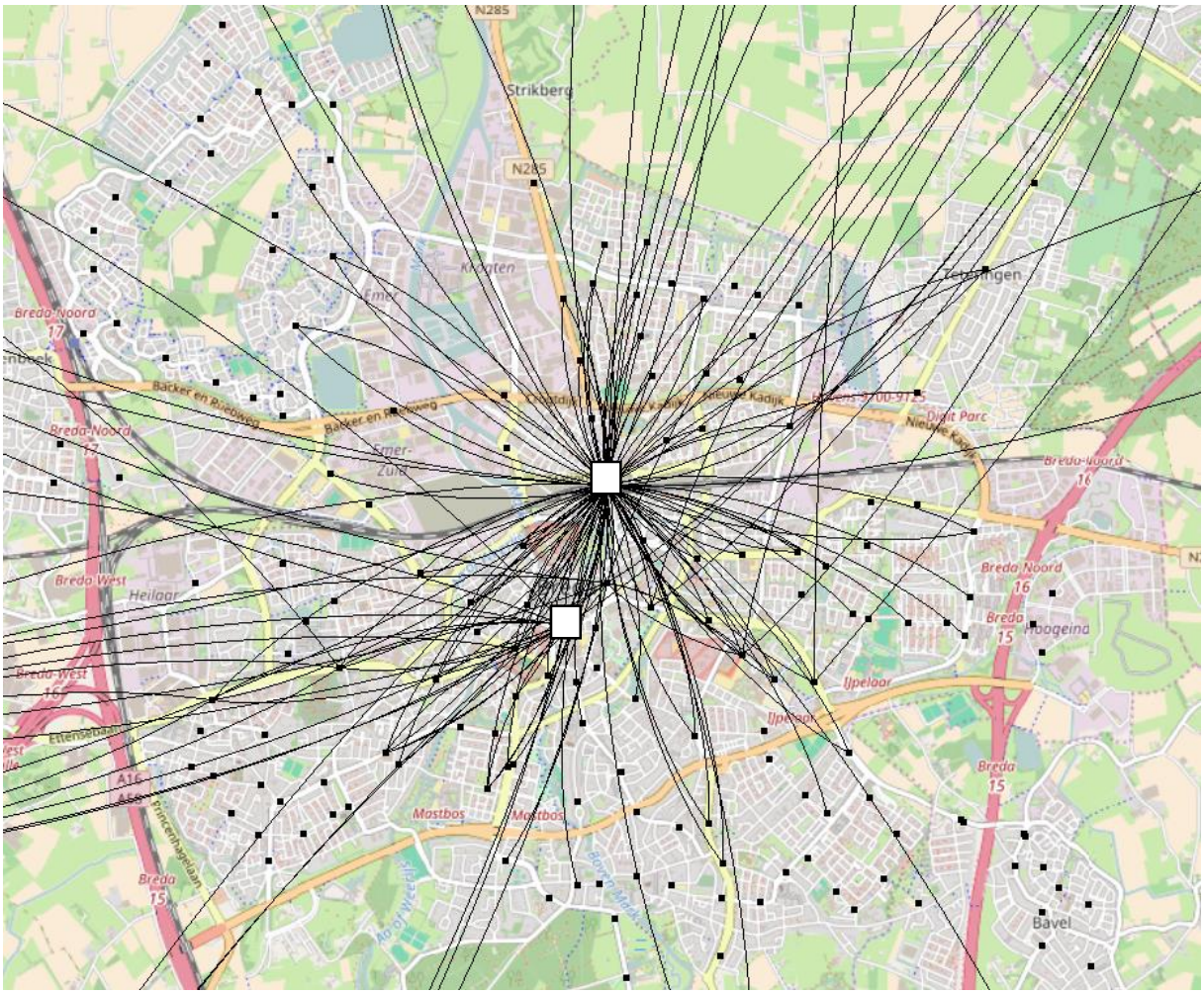


Figure 4, Origin Destination Connections with Demand higher than 5000

6. Results

To do a sensitivity analysis on restrictions of the bus lines we create multiple scenarios. In each scenario, we either adjust the fixed cost of a bus line or the possible start/end nodes of a bus line. In section 6.1 we discuss the results of the fictional data set. In section 6.2 we discuss the results of the practical dataset. We show in each section two objective values: the objective value of the dual problem of the LP relaxation (defined as 'Dual Objective Values'), to show the lower bound of the integer LP, and the objective value of the integer LP (defined as 'Primal Objective Values'). By doing this, we can analyze the differences between these two problems.

6.1 Fictional Data Set

For the fictional data set, the scenarios are determined as follows:

- The fixed cost F of a bus line is in the set $\{0, 50, 100\}$.
- If the start and end node of a bus line is restricted, these two nodes are in the set $\{4, 5, 8, 9\}$

For the fictional data set we show three different kinds of analysis. In section 6.1.1 we show the results per scenario per repetition. In section 6.1.2 the 'moving window' results are shown. Finally, in section 6.1.3, we show the results of the integer approach, and compare the results with the current situation. Note that we use 100 repetitions per scenario, i.e. 100 different initial sets of bus lines. Also, the initial sets of bus lines of, for example, repetition 1, is not the same in each scenario.

6.1.1 Results per Repetition

By showing the results per repetition we want to analyse the dependency on the start set of bus lines. If the final objective value differs a lot among the different repetitions, the dependency is high. In figure 5, the dual objectives are shown per repetition, per scenario. We clearly see that the higher the fixed cost, the higher the objective. It is also nice to see that scenario 'Option 6: All lines possible, fixed cost = 0' always reaches the same objective. So, if the fixed cost are zero and all bus stops can be chosen as starting points, the model will always solve the problem to the same (optimal) solution, which supports our statement in section 4.2.3 that the LP relaxation can be solved to optimality if the fixed cost of a bus line is zero.

When analysing the dependency, the first thing that attracts our attention is that the higher the fixed cost per bus line, the higher the variance between repetitions. This can be explained by the fact that a

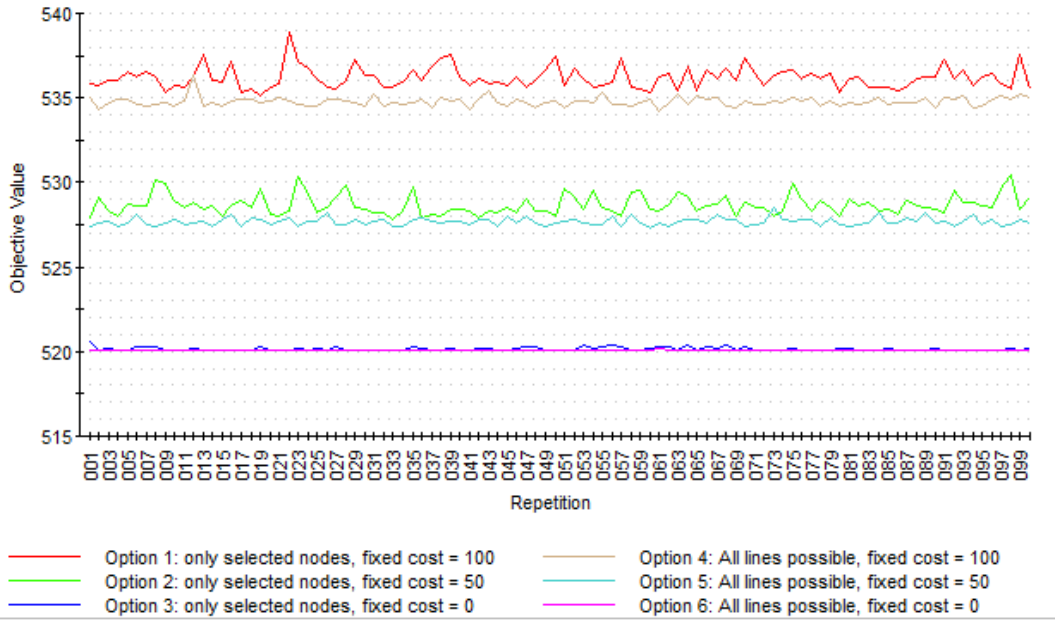


Figure 5, Dual Objective Values per Repetition, Theoretical Network, Computation time ~ 5 minutes

bus line with potential reduced cost is not added to the set of bus lines, because of the fixed cost part in the pricing problem. Therefore, the probability to get stuck in a local optimum is higher. The second interesting thing we see is that the lines with restricted starting points almost always give a higher objective function. This can be explained; some interesting potential bus lines cannot be chosen because of the wrong start bus stop.

If we take a look at figure 6, we see the objective values of the primal problem by solving the integer LP, per repetition, per scenario. Again, the objective values of the scenarios with higher fixed costs are higher. But, the objective values of option 6 are not equal to each other in the primal solution. Regarding the dependency on the initial sets, we again see that the variance is bigger in the scenarios with higher fixed costs. But, due to the high variance of the objective values, it is no longer the case that the scenarios without restricted start bus stops are always resulting in a lower objective value. For example, for only 62 repetitions it applies that the objective value of option 4 is smaller than the objective value of option 1.

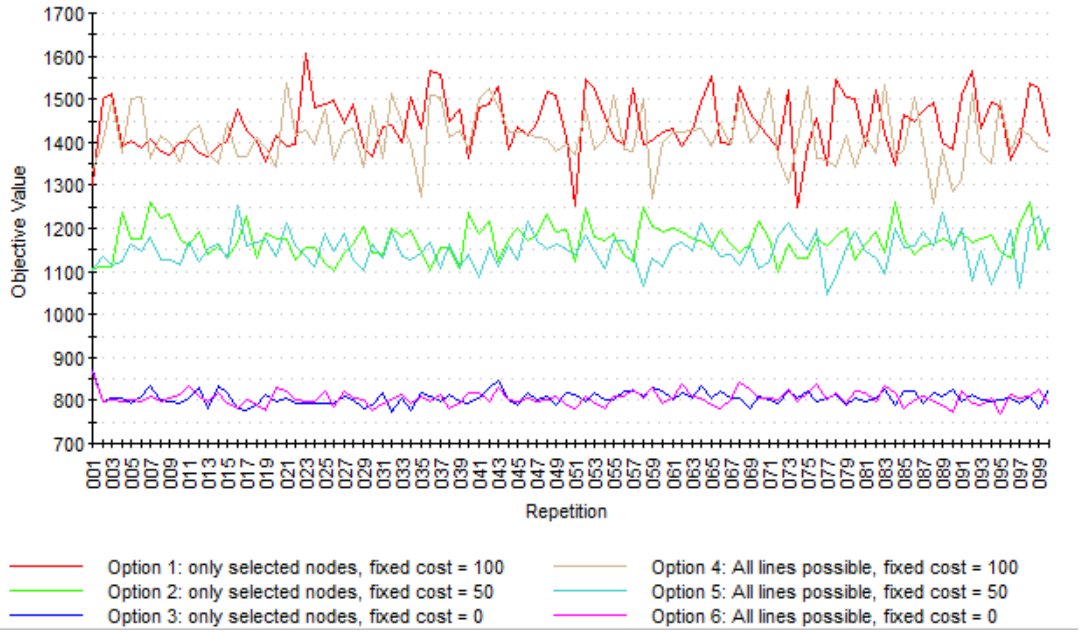


Figure 6, Primal Objective Values per Repetition, Theoretical Network. Computation time ~ 1 hours

In the table below we show some final figures of the repetitions results. With average we mean the average over all the iterations. Fixed costs of 50 or 100 gives similar results, whereas fixed cost equal to zero clearly shows more and shorter bus lines. Also interesting is the fact that the total length of the bus lines is a lot shorter in the scenarios with zero fixed costs. This is because we can get rid of some arc connections when using shorter paths

Scenario	Avg of Number Of Bus lines	Avg of Length per Bus line	Avg of Total Length of Bus lines
Option 1: only selected nodes, fixed cost = 100	5.34	72.03	384.62
Option 4: All lines possible, fixed cost = 100	5.12	72.44	370.91
Option 2: only selected nodes, fixed cost = 50	5.58	65.59	365.99
Option 5: All lines possible, fixed cost = 50	5.06	71.66	362.59
Option 3: only selected nodes, fixed cost = 0	18.83	13.76	259.03
Option 6: All lines possible, fixed cost = 0	20.54	12.33	253.29

Table 2, Overall Statistics of Integer Results per Repetition, Theoretical Network

6.1.2 Moving Window Results

In this section we will analyze the convergence of the objective value. This is done by making use of a 'moving window' approach: in each iteration, the new unique bus lines and paths are added to the existing corresponding sets. We then solve the line planning problem with the extended sets of bus lines and passenger paths. Because the lines and paths in iteration x are still possible options in iteration $x + 1$, the objective value of iteration $x + 1$ will never be higher than the objective value of iteration x.

In figure 7 we see the moving window results of the dual problem. These results are obtained within an hour. Despite the fluctuations in the objective value per repetition, see section 6.1.1, the convergence to a stable solution is really quick. After 10 iterations, the decrease of the objective value per extra iteration is almost equal to 0%. Furthermore, the difference between the first iteration and the last iteration is small and the convergence to a stable solution takes longer for scenarios with higher fixed costs per bus line.

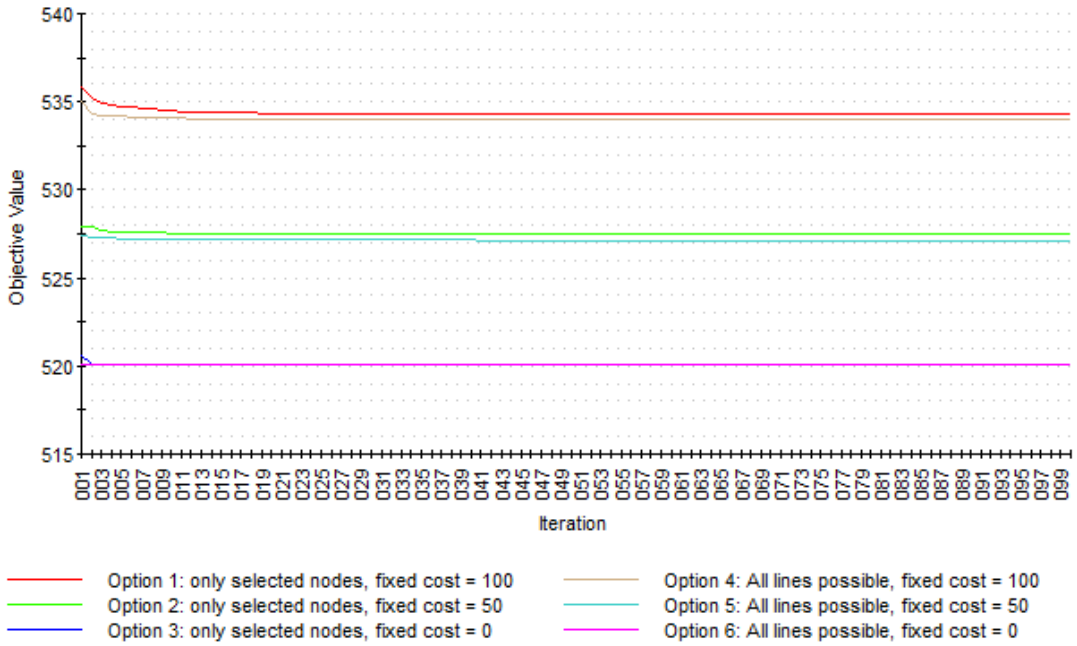


Figure 7, Dual Objective Values per Moving Window, Theoretical Network. Computation time ~ 1 hour.

When we look at moving window results of the primal (integer) problem, figure 8, we see a much higher difference between the objective value of the first iteration and the objective value of the converged situation. Depending on the fixed cost per bus line, the objective value reduces up to 30%. Figure 8 is a nice example of why we use heuristics to solve the line planning problem. The solving time rises exponentially when adding new lines, for example, solving the mixed integer line planning problem with around 4000 integer variables takes already more than one hour. This is the reason that we cannot show the objective values for all iterations and scenarios.

Really interesting is the fact that the calculation time is not that long for the scenarios without fixed bus line costs. Apparently, because of the shorter bus lines in these scenarios, the problem is easier to solve.

Scenario	Avg of Number Of Bus lines	Avg of Length per Bus line	Avg of Total Length of Bus lines
Option 1: only selected nodes, fixed cost = 100	3.12	74.63	232.66
Option 4: All lines possible, fixed cost = 100	3.08	74.99	231.23
Option 2: only selected nodes, fixed cost = 50	3.28	70.28	230.37
Option 5: All lines possible, fixed cost = 50	3.06	73.70	225.58
Option 3: only selected nodes, fixed cost = 0	31.96	5.35	171.10
Option 6: All lines possible, fixed cost = 0	28.28	6.06	171.40

Table 3, Overall Statistics of Integer Results per Moving Window, Theoretical Network

In table 3 the statistics of the moving window results are given. Interesting to see is that compared to the repetition results, the average number of bus lines per iteration drops significantly for the scenarios with fixed bus line costs, while it increases for option 3 and 6. Furthermore, the average length of each bus line stays the same for the fixed costs scenarios, but halves in option 3 and 6. The average total length of the optimal bus lines decreases for all scenarios.

For the scenarios with fixed costs it seems that, because of a bigger set of possible bus lines, the model has the option to choose fewer lines, while still covering the necessary bus stop (arc) connections. This results in fewer unnecessary lines, and therefore, lower fixed costs and a smaller network. The average number of bus lines is high in option 3 and 6, because there are no fixed costs involved. The model chooses those lines that cover all necessary bus stop connections, but no bus stop connection is covered by more than one bus line.

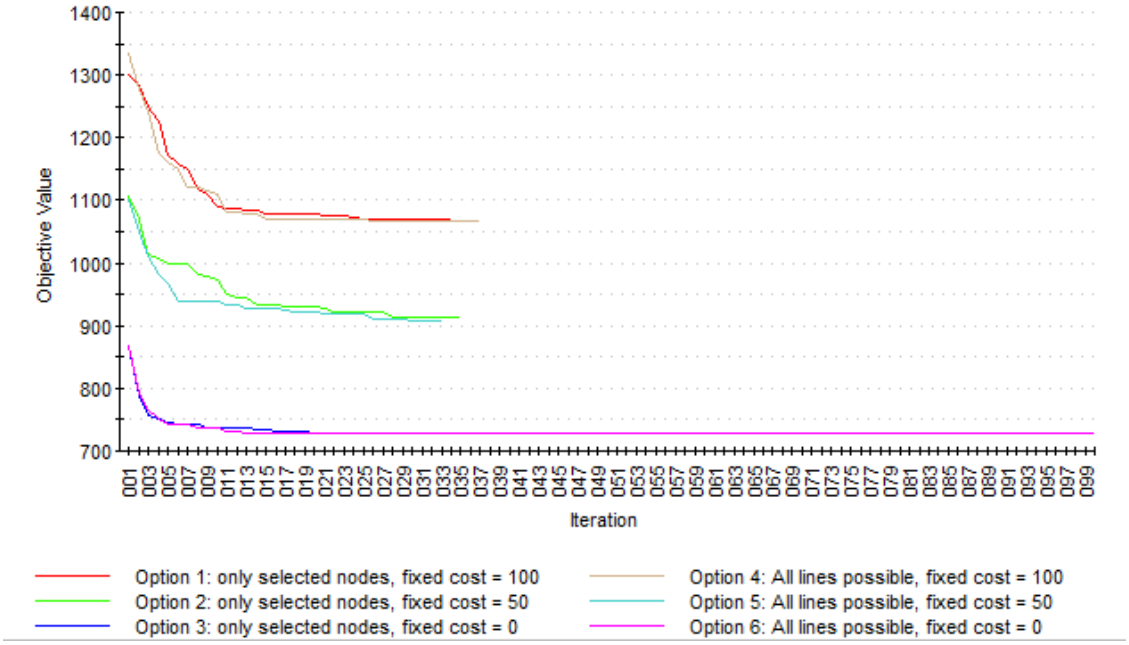


Figure 8, Primal Objective Values per Moving Window, Theoretical Network. Computation time ~ 24 hours.

6.1.3 Fixing Heuristic Results

In table 4 the final results of the line planning problem are shown. We can immediately see that the objective values significantly improve compared to the current situation. If we dive deeper in the results we notice that this improvement is because of the shorter passenger distance. It seems that, despite the use of scalar λ, the length of the passenger paths is still more important for the model. To decrease the length of the passenger paths the model chooses to create longer lines which cover significantly more arcs than in the current situation.

Scenario	# Bus Lines	Avg Line Length	Total Line Length	Total Path Length	Objective Value	# Arcs Covered
Option 1: only selected nodes, fixed cost = 100	3	79.57	238.70	2,118	1,091	51
Option 4: All lines possible, fixed cost = 100	3	71.23	213.70	2,173	1,087	47
Current Situation, fixed cost = 100	4	30.45	121.80	3,162	1,367	25
Option 2: only selected nodes, fixed cost = 50	4	61.10	244.40	2,000	970	53
Option 5: All lines possible, fixed cost = 50	3	66.00	198.00	2,273	932	41
Current Situation, fixed cost = 50	4	30.45	121.80	3,162	1,169	25
Option 3: only selected nodes, fixed cost = 0	39	4.25	165.90	2,123	728	39
Option 6: All lines possible, fixed cost = 0	39	4.25	165.90	2,123	728	39
Current Situation, fixed cost = 0	4	30.45	121.80	3,162	971	25

Table 4, Final Results Line Planning Problem, Theoretical Network

In figure 9 the new network, which is obtained by scenario 'Option 1: only selected nodes, fixed cost = 100', is shown. In this figure, we see again that almost all arcs are covered. In future work, we have to investigate the ideal value of λ such that covering less arcs in the network will be more desirable. In figure 9 we also see another disadvantage of our model: despite the restrictions on the bus lines, the bus lines in the solution are still not applicable in real life because of the multiple cycles in each bus line. The figures of the final network of the other scenarios are shown in the appendix.

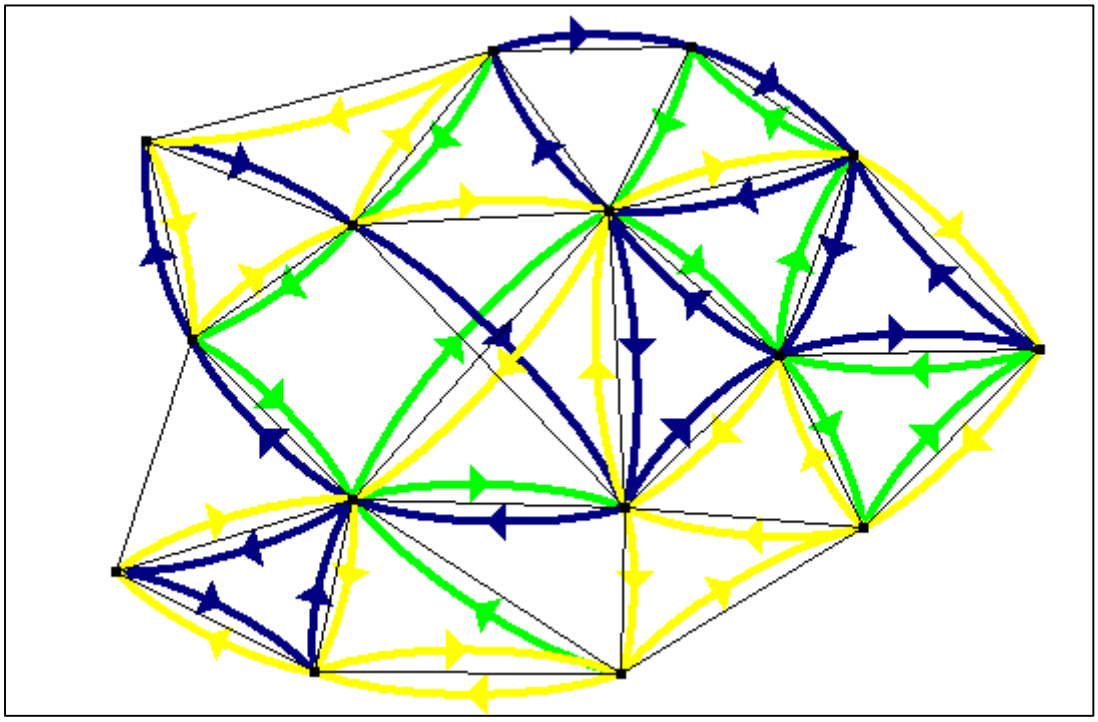


Figure 9, New Bus Line Network, Scenario: Option 1, Theoretical Network, it can be that some are are covered by multiple lines, only one color is shown then

6.2 Practical Data Set

For the practical data set, the scenarios are determined as follows:

- The fixed cost F of a bus line is in the set $\{0, 5000, 10000\}$.
- If the start and end node of a bus line is restricted, these two nodes are in the set of the following bus stops:

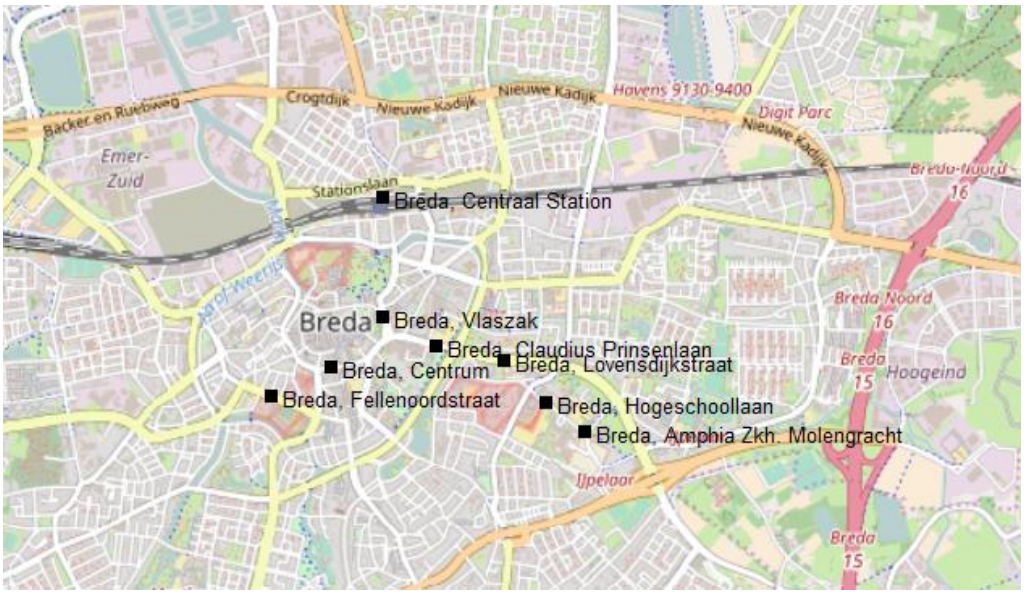


Figure 10, Start bus stops for restricted scenarios

For the practical data set we will also show the repetition results (section 6.2.1), the 'moving window' results (section 6.2.2) and the final integer results (section 6.2.3). For this data set we used 30 iterations, and, in contrast to the theoretical set, the initial set of bus lines in iteration i of scenario j is the same as the initial set of bus lines in iteration i of scenario k.

6.2.1 Results per Repetition

The dual objectives per repetition, per scenario are as follows:

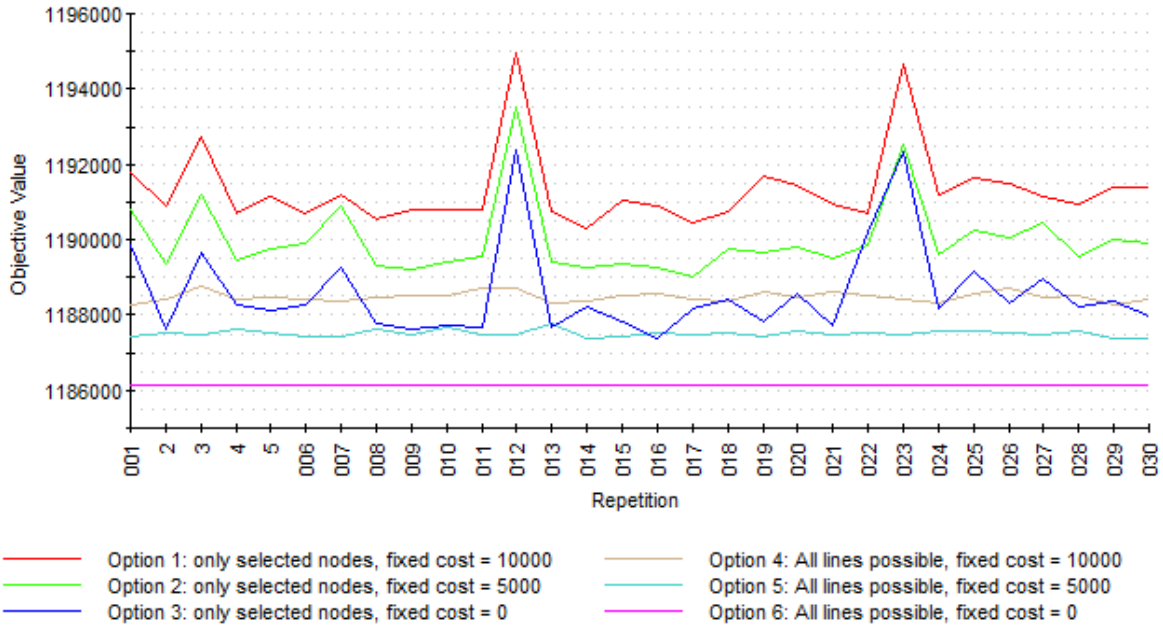


Figure 11, Dual Objective Values per Repetition, Practical Network. Computation time ~ 1 hour

In this graph we again see that the objective value depends on the fixed costs, but it also depends a lot on the possible start nodes, certainly compared with the results of the theoretical data set. This can be explained by the much smaller (relative) set of possible start nodes that we used for this practical data set. The initial set of bus lines is even more important and the variance between the objective functions is large.

If we look at the scenarios where all bus stops are allowed as starting point, we see more stable objective functions. Again, scenario 'Option 6' always gives the same objective, which again supports our statement in section 4.2.3 that the LP relaxation can be solved to optimality if the fixed cost of a bus line is zero.

If we take a look at figure 12, we see the objective values of the primal problem by solving the integer LP, per repetition, per scenario. This time the variance of the scenarios with a restriction on the set of possible start bus stops is smaller, but again, the objective values of these scenarios are significant higher than the scenarios without restrictions.

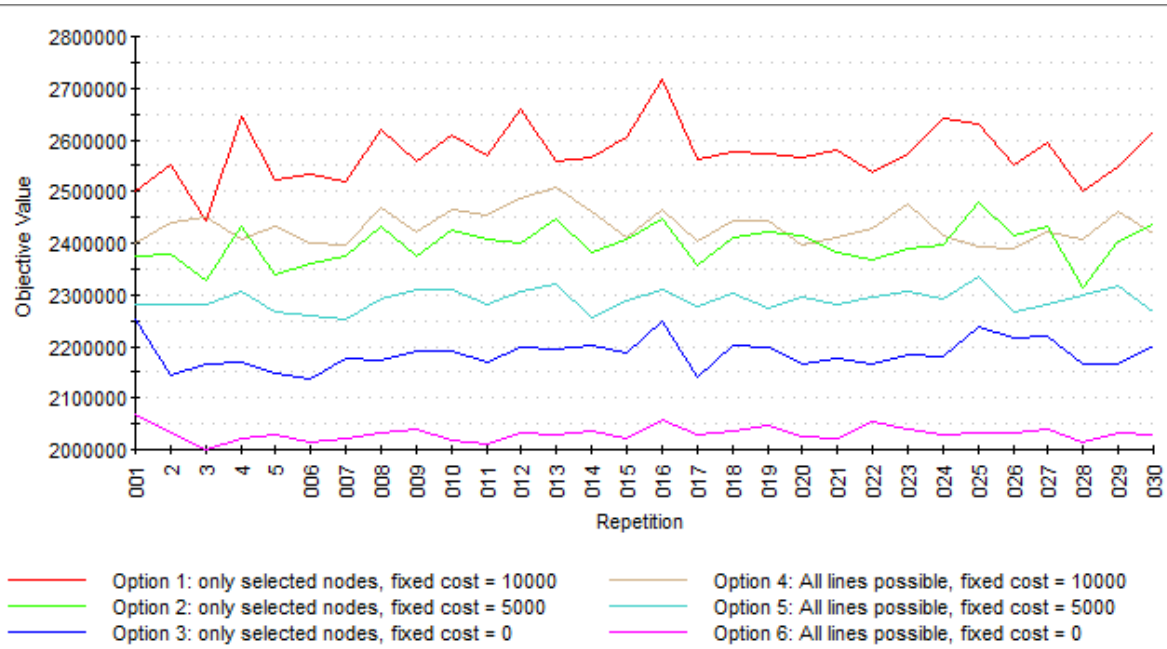


Figure 12, Primal Objective Values per repetition, Practical Network Computation time ~ 2 hours

In table 5 below we show some final figures of the repetition results. This time the height of the fixed cost does have influence on the number of bus lines: the higher the fixed cost, the smaller the average number of bus lines in the optimal solution. Also interesting to see is the influence of the restriction on the start set of bus stops. Because we cannot choose all possible bus lines, the average total length of all bus lines is significantly higher.

Scenario	Avg of Number Of Bus lines	Avg of Length per Bus line	Avg of Total Length of Bus lines
Option 1: only selected nodes, fixed cost = 10000	35.37	29,393	1,039,521
Option 4: All lines possible, fixed cost = 10000	27.40	35,599	975,421
Option 2: only selected nodes, fixed cost = 5000	40.37	25,056	1,011,439
Option 5: All lines possible, fixed cost = 5000	32.63	28,842	941,226
Option 3: only selected nodes, fixed cost = 0	48.47	20,670	1,001,783
Option 6: All lines possible, fixed cost = 0	64.80	13,063	846,484

Table 5, Overall Statistics of Integer Results per repetition, Practical Network

6.2.2 Moving Window Results

In figure 13 we see the moving window results of the dual problem. The convergences to a stable solution takes slightly longer for the restricted scenarios, but it converges again really quickly. After 10 iterations, the decrease of the objective value per extra iteration is almost equal to 0%. Also interesting is the high objective function in the first iteration for the scenarios with restrictions on the start bus stops. Clearly, the bus lines with low costs are starting and/or ending outside the set of restricted bus stops.

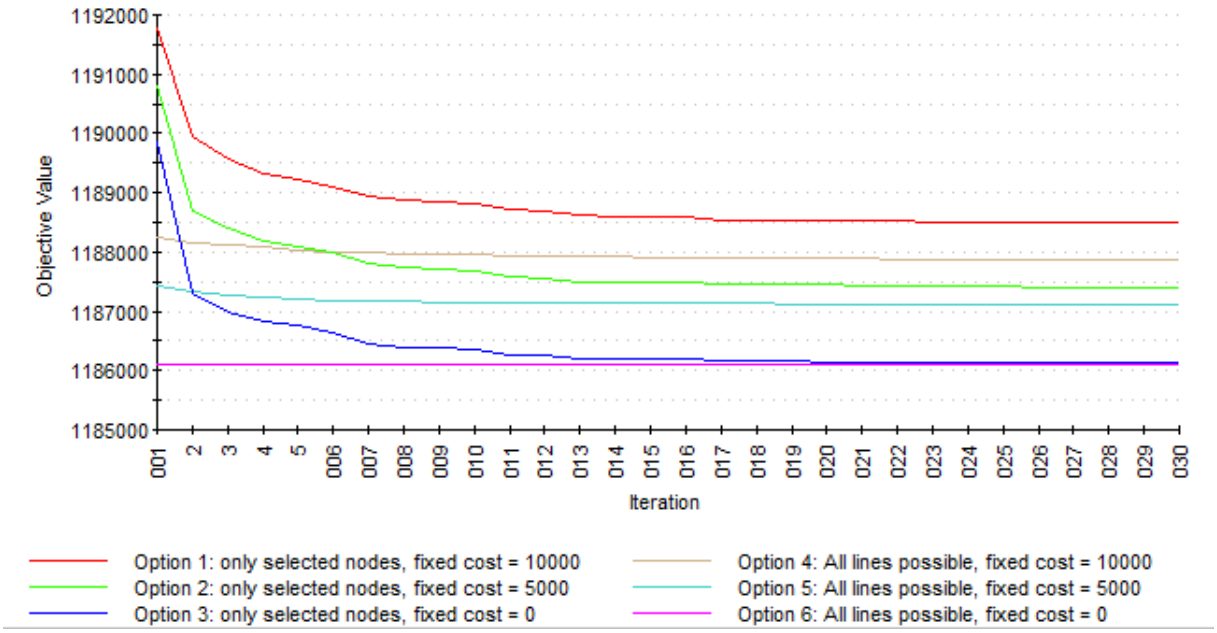


Figure 13, Dual Objective Values per Moving Window, Practical Network. Computation time ~ 24 hours

If we look at the moving window results of the primal model, figure 14, we see the same results as for the theoretical data set. First, the objective values reduce drastically, but after 15 iterations, the improvement per iteration is very small. And again, with each extra iteration, and thus more possible bus lines, the calculation time increases exponentially. Solving the line planning problem exactly for a large instance of data is therefore not possible within reasonable time and a heuristic like the fixing heuristic proposed in section 4.3.2 is needed.

In table 6 the statistics of the moving window results are given. If we compare these results with the repetition results, we see that the average number of bus lines per iteration decreases a lot for all scenarios. Opposed to this is a larger average bus line length, but the average total length of the optimal bus lines decreases for all scenarios.

Scenario	Avg of Number Of Bus lines	Avg of Length per Bus line	Avg of Total Length of Bus lines
Option 1: only selected nodes, fixed cost = 10000	21.23	39,723	843,448
Option 4: All lines possible, fixed cost = 10000	16.87	49,447	834,005
Option 2: only selected nodes, fixed cost = 5000	23.80	34,792	828,048
Option 5: All lines possible, fixed cost = 5000	20.70	39,108	809,540
Option 3: only selected nodes, fixed cost = 0	61.20	12,759	780,875
Option 6: All lines possible, fixed cost = 0	145.30	5,122	744,253

Table 6, Overall Statistics of Integer Results per Moving Window, Practical Network

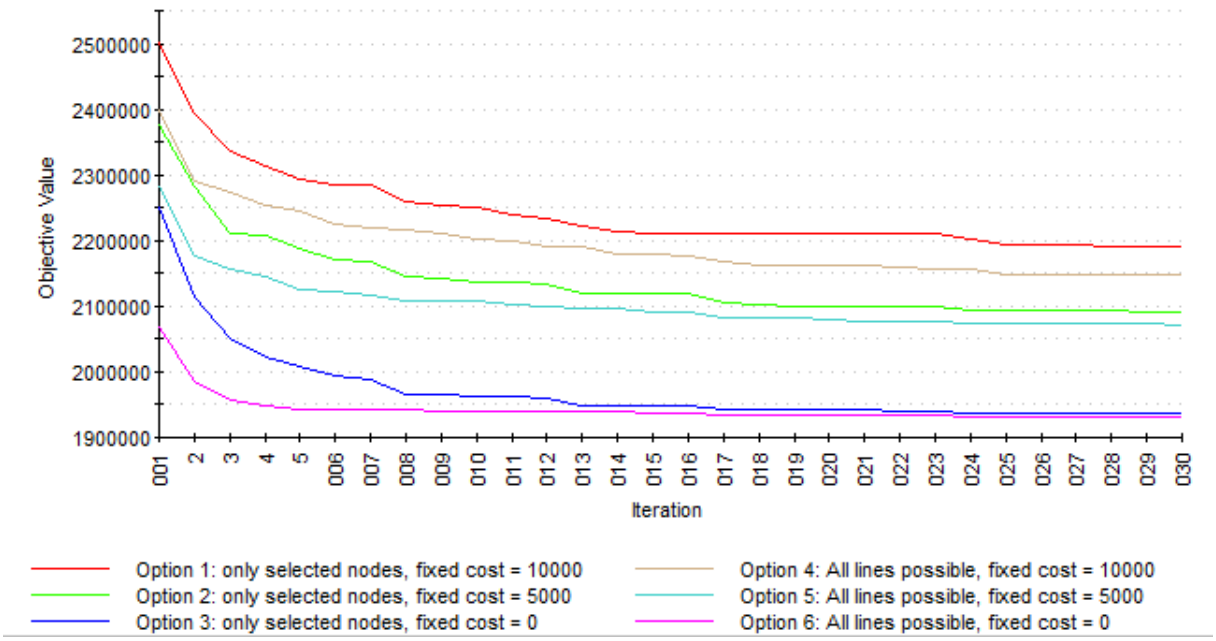


Figure 14, Primal Objective Values per Moving Window, Practical Network

6.2.3 Fixing Heuristic Results

In table 7, the final results of the line planning problem are shown for the practical data set. We again see significantly improved objective values compared to the current situation. But this time, the improvement is mainly because of the shorter total bus line length. The total bus line length is a lot shorter because the model decides to cover less arcs. This is the opposite result compared to the theoretical data and is probably due to the more centralized demand. The length of the passenger paths is almost the same for each scenario. This is due to no new arcs being available for the creation of bus lines, and that the model determines to cover less arcs.

Figure 15 shows the new bus line network for scenario 'Option 4, All lines possible, fixed cost = 10000'. In this figure we see that, compared to the current network, more connections are only covered one-way. The final bus line networks from the other scenarios are shown in the appendix.

Scenario	# Bus Lines	Avg Line Length	Total Line Length	Total Path Length	Objective Value	# Arcs Covered
Option 1: only selected nodes, fixed cost = 10000	42	20,435	858,270	26,778,825	2,469,597	435
Option 4: All lines possible, fixed cost = 10000	34	24,621	837,120	26,854,383	2,368,616	424
Current Situation, fixed cost = 10000	29	40,686	1,179,886	26,279,687	2,649,713	451
Option 2: only selected nodes, fixed cost = 5000	45	18,459	830,649	27,207,398	2,254,670	421
Option 5: All lines possible, fixed cost = 5000	43	17,936	771,269	27,767,736	2,216,870	412
Current Situation, fixed cost = 5000	29	40,686	1,179,886	26,279,687	2,504,716	451
Option 3: only selected nodes, fixed cost = 0	104	7,557	785,951	27,041,128	1,982,276	410
Option 6: All lines possible, fixed cost = 0	336	2,180	732,448	27,475,332	1,935,688	399
Current Situation, fixed cost = 0	29	40,686	1,179,886	26,279,687	2,359,719	451

Table 7, Final Results Line Planning Problem, Practical Network

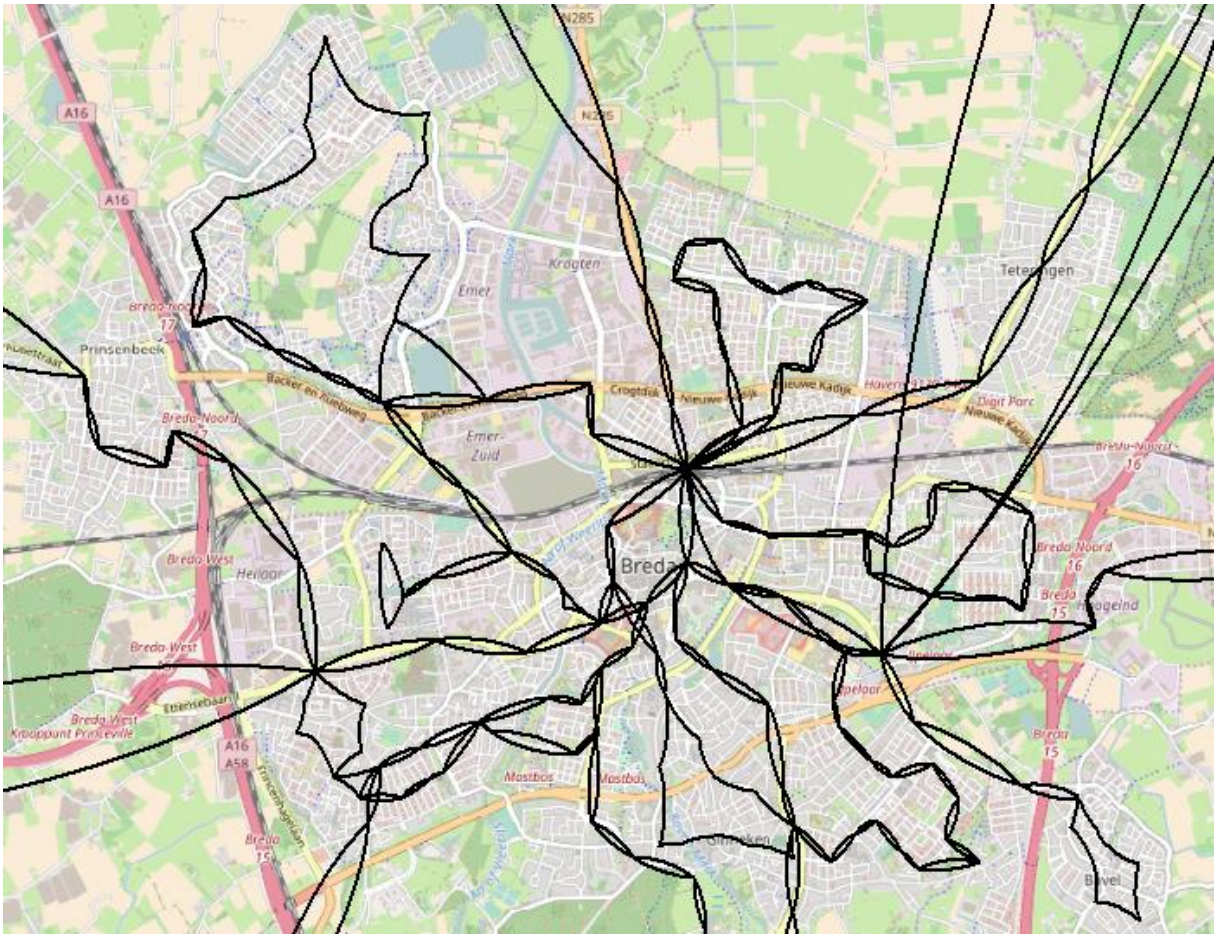


Figure 15, New Bus Line Network, Scenario: Option 4, Practical Network

7. Conclusion

In this article we developed a new method for solving the line planning in public transport. This model is based on the earlier work of Borndörfer et al. (2007) who presented the mathematical formulation of the line planning problem. In the model, the set of bus lines and the set of passenger paths are created dynamically making use of column generation and the corresponding pricing problems. We followed Borndörfer et al. (2007) by using a shortest path algorithm to solve the passenger pricing problems, but we created a new algorithm to solve the bus line pricing problems.

By using the characteristics of a bus line we were able to create a proper heuristic for solving the bus line pricing problem. Because we use the practical characteristics of a bus line, the model is more related to real-life line planning problems. A disadvantage of our heuristic is that it does not solve the relaxed line planning problem to optimum if the fixed cost of a bus line is higher than zero or if not all nodes can be start or end nodes of the bus lines.

We used our model to solve two different data instances: one theoretical set of 15 bus stops and one practical set consisting of the bus network of Breda (Arriva). For these two instances we created different scenarios, where each scenario is a unique combination of a fixed cost per bus line and a set of allowed start bus stops per bus line. To check the divergence to the lower bound and to check the dependence on the initial set of bus lines we solved multiple repetitions of each scenario, each repetition starting with a new (unique) set of bus lines.

We showed that both data instances can be solved within reasonable computation time for all scenarios. Also, the objective functions of the different scenarios are close (within 10%) to their corresponding moving average bound. Furthermore, we conclude that working with multiple iterations is a good method to eliminate possible dependency issues.

The main research question, ‘How can Arriva, using data from the ‘OV-Chipkaart’ system, increase the profitability in Public Transport in Noord-Brabant, while maintaining a basic service level and ridership?’ has been partially answered. The results show that the objective function can be significantly reduced, but, despite these encouraging results of our model, the gap to the practical implementation is still too large. Therefore, further research is needed on how to connect the mathematical formulas with practical implementations. Some quick examples of this further research are described in the next chapter.

8. Further Research

As mentioned in the conclusion, there are still a lot of improvements possible. There especially is room for improvement in the connection between the theoretical model and the practical usage. At this moment, the cost corresponding to a bus line and a passenger path are still very theoretical. For example, we do not take into account capacities, frequencies and customer transfers. In this chapter we will briefly describe the mathematical formulation of such an extension.

First of all, we need to take into account the capacity of each bus. Instead of using the Big K, we introduce parameter k_l, the capacity of each bus on bus line l. Furthermore, we introduce decision variable f_l, which determines the frequency of line l during a given period. Using these two new parameters, the original restriction (3) in the LPP formulation will change to:

$$\sum_{p \in P_a} y_p - \sum_{l \in L | a \in l} f_l * k_l \leq 0 \qquad \forall\, a \in A$$

And to determine if a line is used we need the following restriction:

$$f_l \leq K * x_l \qquad \forall\, l \in L$$

To deal with customer transfers the network graph will change drastically. For each bus line and its corresponding bus stops a node is needed. In this way, a passenger can travel from node A (Stop 1, Line 1) to node B (Stop 2, Line 1), where he will make a transfer to node C (Stop 2, Line 2). See figure 16 for a graphical representation of this graph. As you can already see, the graph explodes when adding the new nodes and arcs. The cost of a passenger path is then split into two parts: the total travel distance and the total transfer costs. These transfer costs are calculated by assigning a 'Transfer Penalty' for each transfer arc, i.e., an arc between two nodes at the same bus stop.

As described in the literature section, Schöbel & Scholl (2005) developed multiple models which take into account the total travel times of the passengers and the number of transfers of the passengers. They also presented a model that takes frequencies into account. In their article they showed that many real world cases can be solved by a solution approach based on Dantzig-Wolfe decomposition, especially if a solution is not found because of the calculation time.

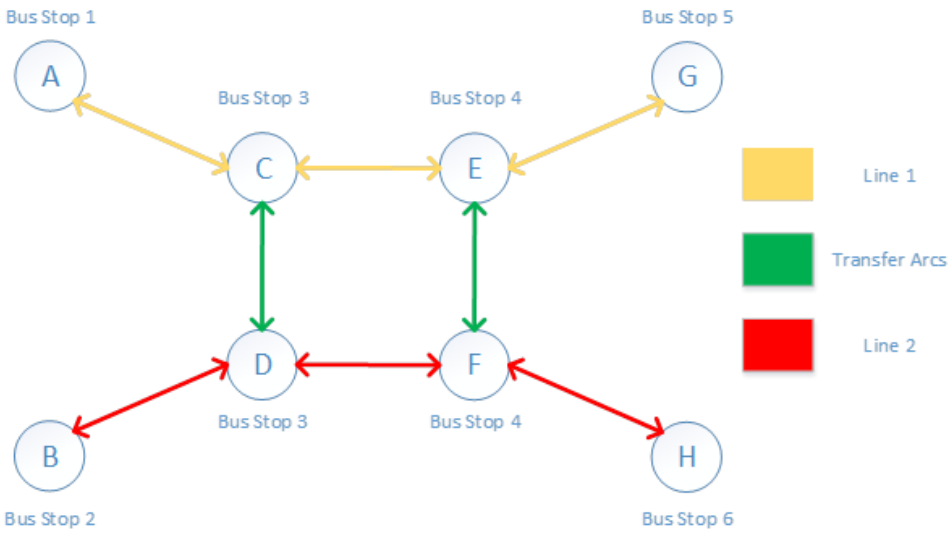


Figure 16, Graphical Representation of a Transfer Network

9. References

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., & Vance, P. (1998). Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, *46*(3), 316-329. http://dx.doi.org/10.1287/opre.46.3.316

Bellman, R. (1958). On a routing problem. *Quarterly of Applied Mathematics*, *16*(1), 87-90. http://dx.doi.org/10.1090/qam/102435

Borndörfer, R., Grötschel, M., & Pfetsch, M. (2007). A Column-Generation Approach to Line Planning in Public Transport. *Transportation Science*, *41*(1), 123-132. http://dx.doi.org/10.1287/trsc.1060.0161

Bouma, A. & Oltrogge, C. (1994). Linienplanung und Simulation für öffentliche Verkehrswege in Praxis und Theorie. *Eisenbahntechnische Rundschau 43,* 369–378.

Ceder, A. & Wilson, N. (1986). Bus Network Design. *Transportation Research Part B: Methodological*, *20*(4), 331-344. http://dx.doi.org/10.1016/0191-2615(86)90047-0

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269-271. http://dx.doi.org/10.1007/bf01386390

Ford, L. & Fulkerson, D. (1956). Maximal flow through a network. *Journal Canadien De Mathématiques*, *8*(0), 399-404. http://dx.doi.org/10.4153/cjm-1956-045-5

Garey, M. & Johnson, D. (1979). *Computers and intractability* (1st Ed.). San Francisco: W.H. Freeman.

Goossens, J., van Hoesel, S., & Kroon, L. (2006). On solving multi-type railway line planning problems. *European Journal of Operational Research*, *168*(2), 403-424. http://dx.doi.org/10.1016/j.ejor.2004.04.036

Israeli, Y. & Ceder, A. (1995). Transit Route Design Using Scheduling and Multi Objective Programming Techniques. *Lecture Notes in Economics and Mathematical Systems*, 56-75. http://dx.doi.org/10.1007/978-3-642-57762-8_5

Jaramillo-Álvarez, P., Gonzalez-Calderon, C., & Gonzalez-Calderon, G. (2013). Route Optimization of Urban Public Transportation. *DYNA*, *80*(180), 41-49. Retrieved December 29, 2016, from http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0012-73532013000400006&lng=en&tlng=en

Lahaie, S. (2008). How to take the Dual of a Linear Program, Retrieved from http://www.cs.columbia.edu/coms6998-3/lpprimer.pdf

Mandl, C. (1980). Evaluation and optimization of urban public transportation networks. *European Journal of Operational Research*, *5*(6), 396-404. http://dx.doi.org/10.1016/0377-2217(80)90126-5

Quak CB (2003) Bus line planning. Master's thesis, TU Delft

Schöbel, A. (2011). Line planning in public transportation: models and methods. *OR Spectrum*, *34*(3), 491-510. http://dx.doi.org/10.1007/s00291-011-0251-6

Schöbel, A. & Scholl, S. (2005). Line planning with minimal travelling time. Preprint 1-2005, Universität Göttingen, Germany.

Scholl, S. (2005). Customer-oriented line planning. PhD. Thesis, Universität Göttingen, Germany.

Silman, L., Barzily, Z., & Passy, U. (1974). Planning the route system for urban buses. *Computers & Operations Research*, *1*(2), 201-211. http://dx.doi.org/10.1016/0305-0548(74)90046-x

Szeto, W. & Wu, Y. (2011). A simultaneous bus route design and frequency setting problem for Tin Shui Wai, Hong Kong. *European Journal of Operational Research*, *209*(2), 141-155. http://dx.doi.org/10.1016/j.ejor.2010.08.020

Nielsen, G. & Lange, T. (2007). Network design for public transport success: theory and examples. *Norwegian Ministry of Transport and Communications, Oslo.*

Appendix A. New Networks Theoretical Data

Notice that some arcs are covered by multiple bus lines, if this is the case, not all colors are shown.

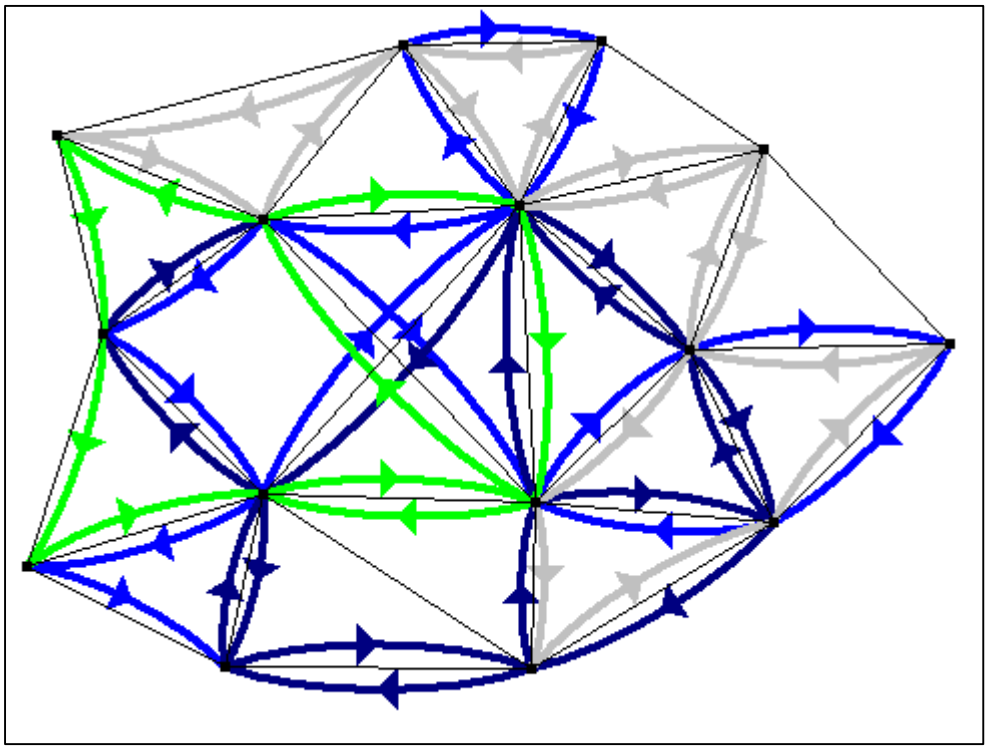


Figure 17, New Bus Line Network, Scenario: Option 2, Theoretical Network

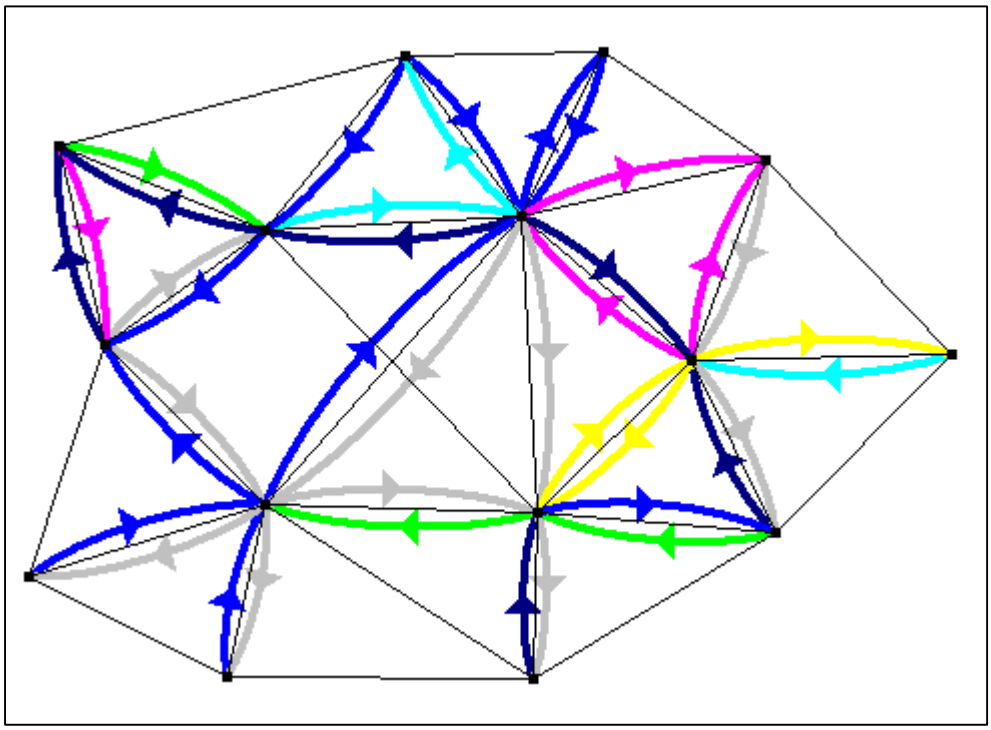


Figure 18, New Bus Line Network, Scenario: Option 3, Theoretical Network

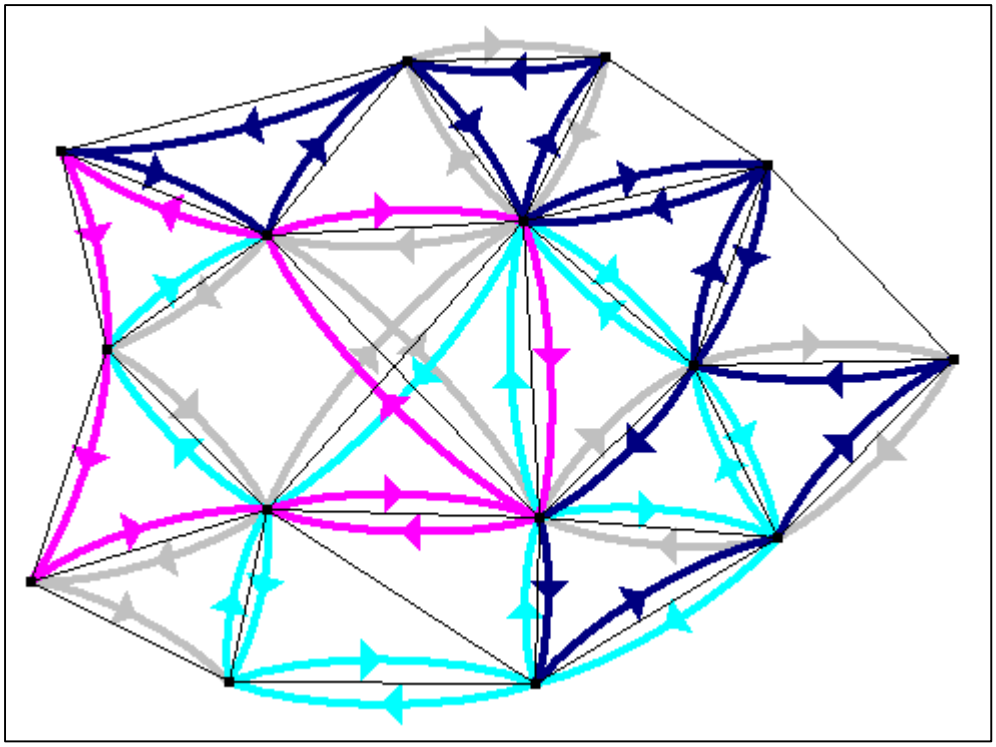


Figure 19, New Bus Line Network, Scenario: Option 4, Theoretical Network

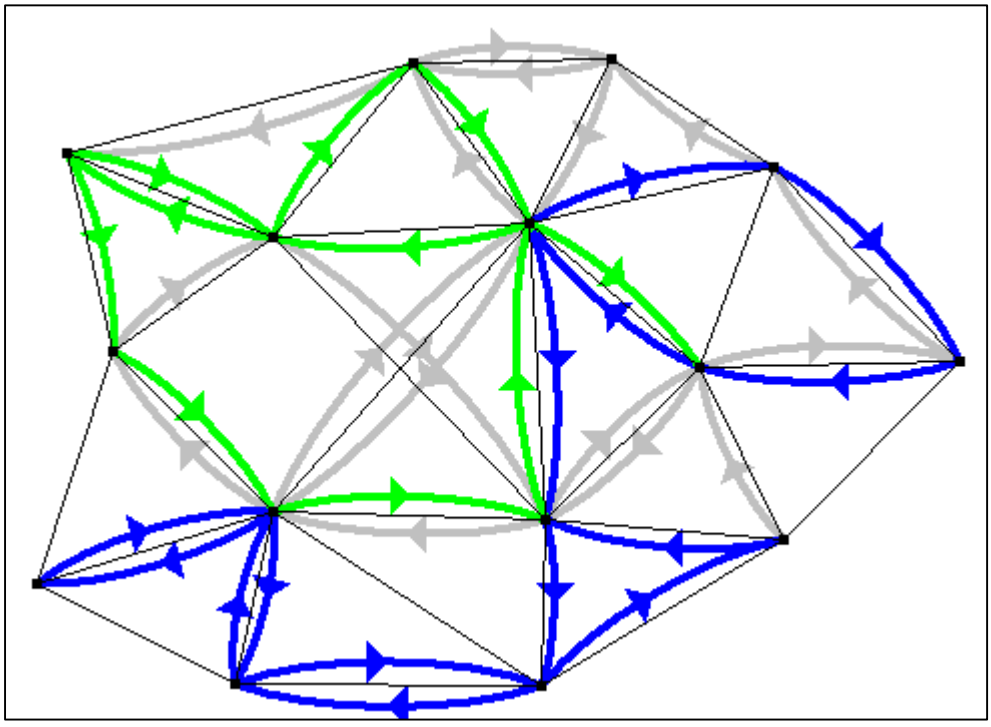


Figure 20, New Bus Line Network, Scenario: Option 5, Theoretical Network

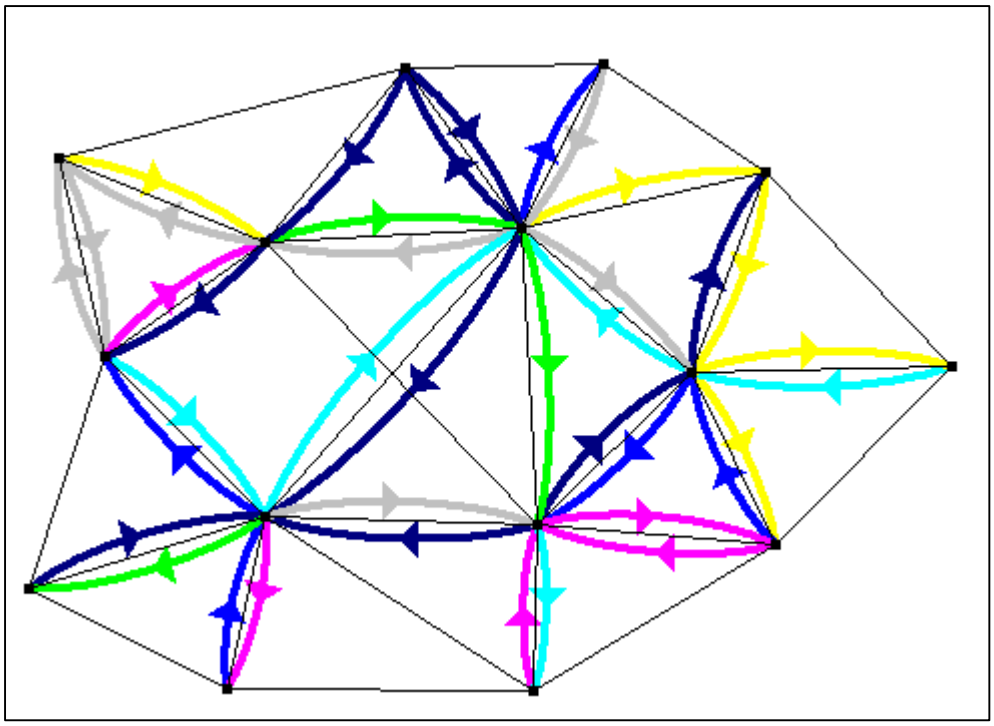


Figure 21, New Bus Line Network, Scenario: Option 6, Theoretical Network

Appendix B. New Networks Practical Data

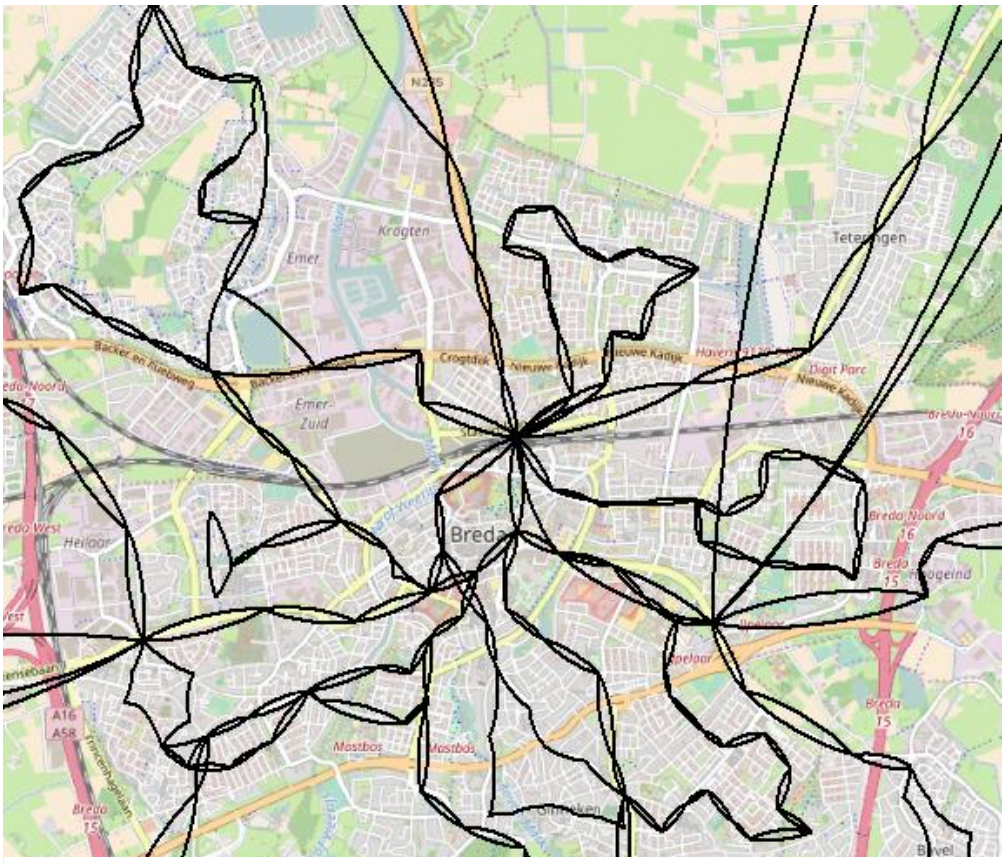


Figure 22, New Bus Line Network, Scenario: Option 1, Practical Network

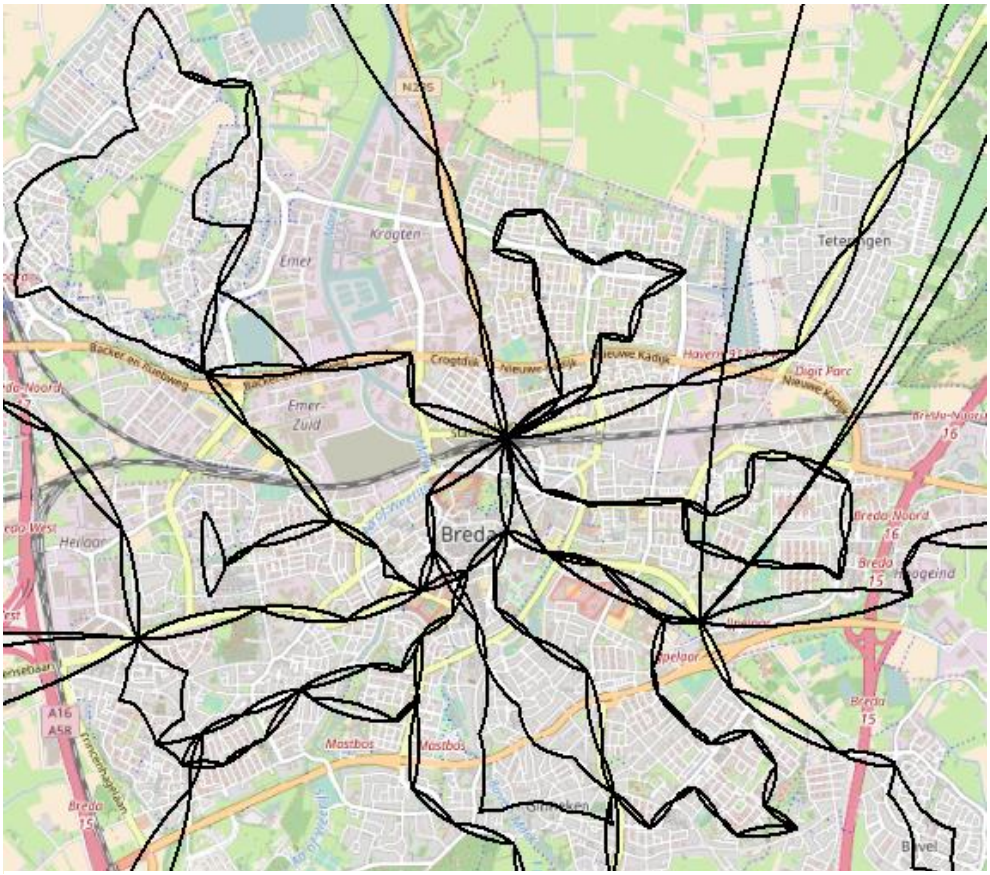


Figure 23, New Bus Line Network, Scenario: Option 2, Practical Network

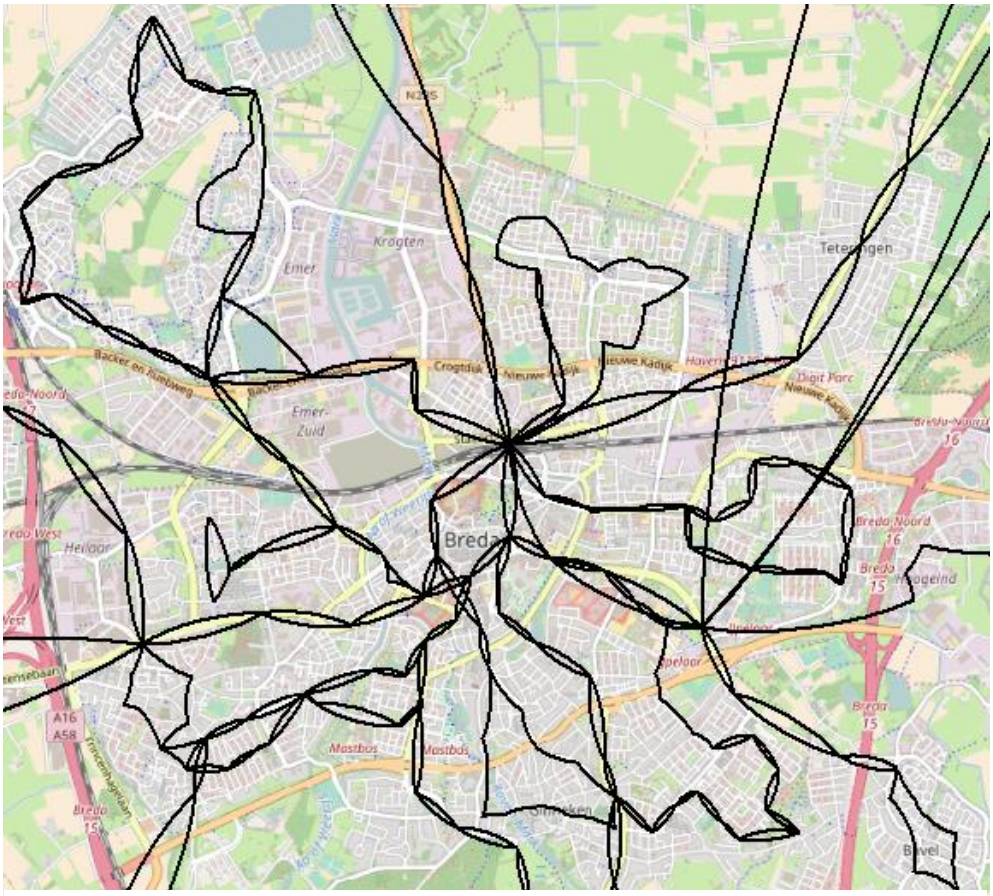


Figure 24, New Bus Line Network, Scenario: Option 3, Practical Network

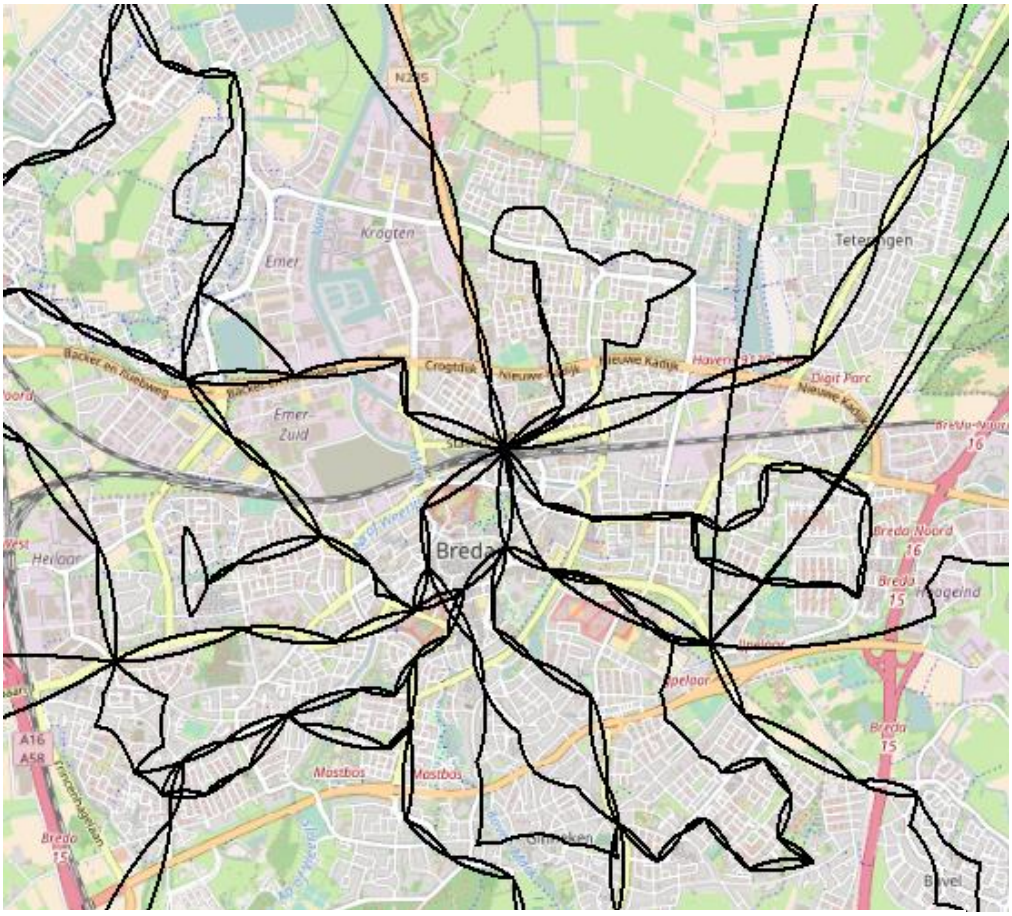


Figure 25, New Bus Line Network, Scenario: Option 5, Practical Network

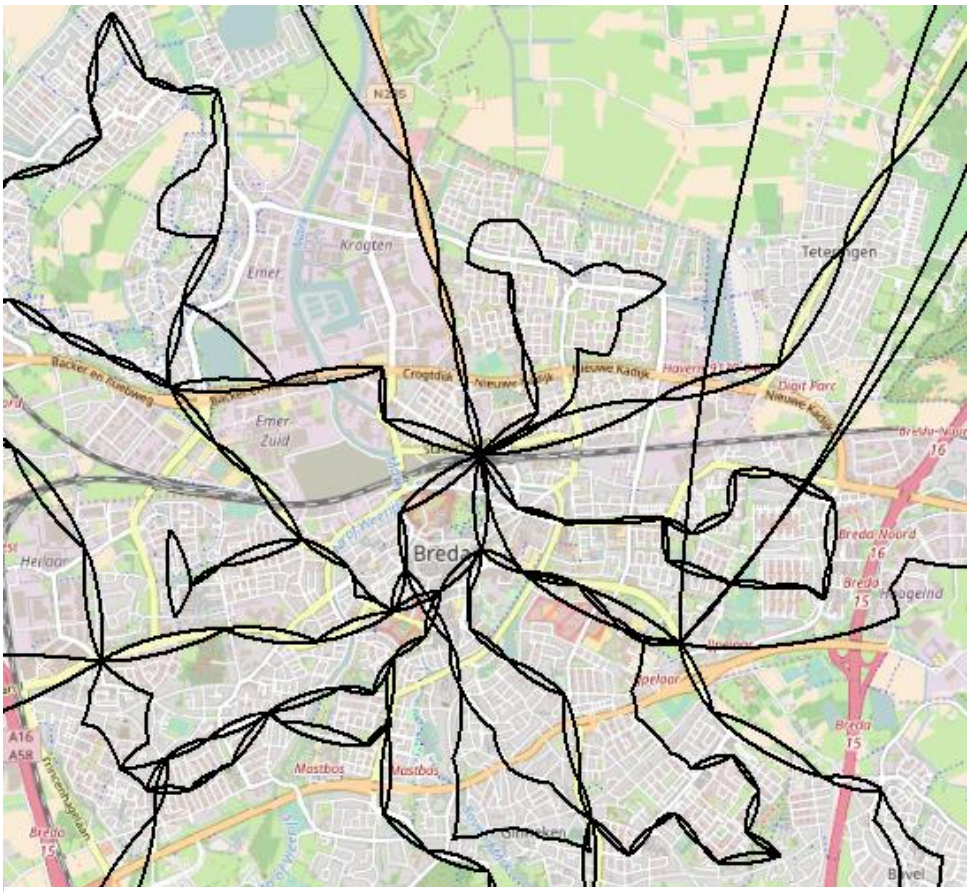


Figure 26, New Bus Line Network, Scenario: Option 6, Practical Network