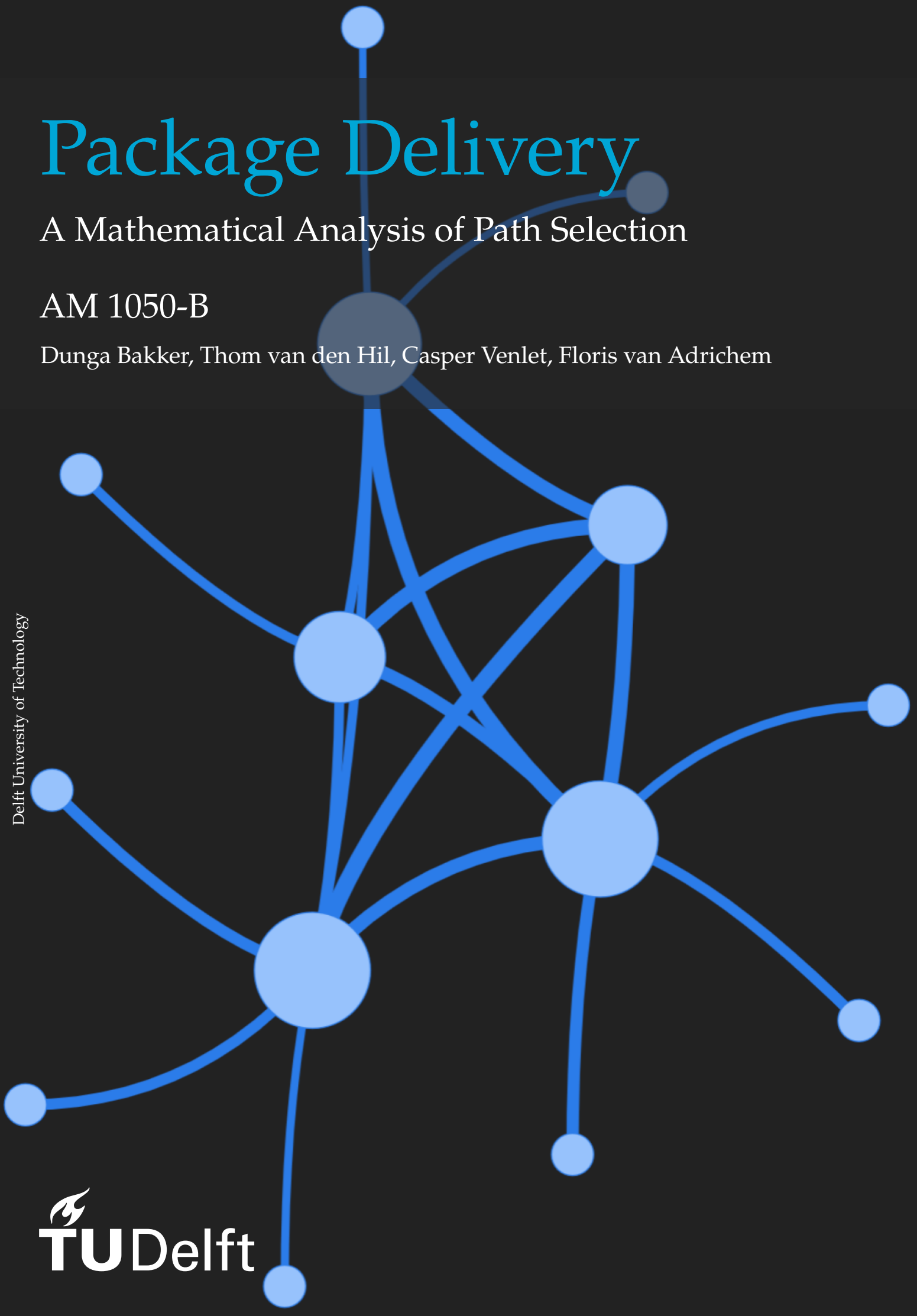
Package Delivery

A Mathematical Analysis of Path Selection

AM 1050-B

Dunga Bakker, Thom van den Hil, Casper Venlet, Floris van Adrichem



Delft University of Technology

TU Delft

Abstract

In this report, a mathematical optimisation of distributing packages between 10 or 15 nodes is found. For every node, the amount of packages leaving the node and going to the node have been given, as well as the the cost of this transportation. The goal is to transform some nodes into hubs so that packages can be sorted in those places. Indirect transportation is cheaper than direct delivery. First, the problem is written as an Integer Linear Problem to get a guaranteed optimal solution. This approach however takes a significant amount of time for large data sets and therefore heuristic algorithms are constructed, which achieve a slightly less optimal answer in a considerably faster time. The ILP is around two orders of magnitude slower than the second heuristic algorithm, but the second heuristic algorithm is on average 5.13% more expensive. For the data sets used in this report, using three hubs is optimal for data sets with both 10 and 15 nodes.

The last part of the article is about an extension for the original problem and the resulting changes in the way the hubs get placed. For a discount of 15%, the resulting change is surprising. For high discount rates, only one hub is placed. For a lower rate, nothing in the hub placement changes. If the discount is equal to 15%, only two hubs are built.

Contents

Nomenclature

List of Abbreviations

Abbreviation	Description
NTM	Node Transport Modules
HTM	Hub Transport Modules
CNC	Cheapest Node Connection
ILP	Integer Linear Problem

List of Variables

Variable	Description	Unit
N	set of nodes	-
n	amount of nodes / length of data set	-
n_{max}	the amount of samples used from the runtimes of PuLP on the large data set	-
S	the hub selection, so the set of hubs	-
C	the current hubs selection, so the current set of hubs	-
R	the set of remaining hubs, defined as $N \setminus S$	-
z	total costs / ILP objective function	-
z_{min}	minimal costs	-
$f(n)$	objective function as an estimate for the runtime complexity of PuLP, where n denotes the length of the data set	-
c_1	parameter in objective function f	-
c_2	parameter in objective function f	-
f_i	the fixed cost of building a hub at location i, for $i \in N$	-
w_{ij}	the amount of packages or flow that goes from i to j, for $i, j \in N$	-
c_{ij}	the cost to transfer 1 package from i to j, for $i, j \in N$	-
χ	multiplier of collection costs for a package going from a node to a hub	-
α	multiplier of transfer costs for a parcel going from a hub to hub	-
δ	multiplier of distribution costs for a parcel going from a hub to a node	-
ξ	multiplier of the cost if the node is an NTM or an HTM	-
h_i	1 if i is a hub, 0 otherwise	-
e_{ij}	1 if there is an edge from i to j, 0 otherwise	-
a_{ij}	1 if j is a hub and i and j are connected, 0 otherwise	-
v_i	1 if i is an NTM, 0 otherwise	-
m_{ij}	1 if the edge from i to j is an HTM, 0 otherwise	-
t_{ij}	is equal to $h_i h_j$	-
t_{ijk}	is equal to $h_i \cdot (1 - h_j) \cdot e_{kj}$	-
t_{ijkl}	is equal to $(1 - h_i)(1 - h_j)e_{ik}e_{lj}$	-
p_{ijkl}	is equal to $a_{ik}a_{jl}$	-
q_{1ijkl}	is equal to $n_k p_{ijkl}$	-
q_{2ijkl}	is equal to $n_l p_{ijkl}$	-
q_{3ijkl}	is equal to $m_{kl} p_{ijkl}$	-

1

Introduction

In today's society, transportation makes up one of the most vital industries in the modern world. Many companies specialize in transporting personal and commercial cargo over large distances, and these enterprises have a large role in the current market. They employ many people all over the world and offer services to many consumers.

These businesses often receive a lot of demand for their services. Many industrial and agricultural companies provide door-to-door delivery on a national level, which means that the transportation business tries to deliver to as many locations as possible. More often than not, direct delivery is neither efficient nor cost-effective. It is for this reason that all distribution businesses utilize sorting centers: locations where packages are first transported to, sorted, and then sent along their way in bulk.

Choosing positions for the sorting centers proves to be a difficult task. Even though it seems trivial to first place sorting centers in or close to major cities, this might not actually be the cheapest or fastest option in reality. It is thus proposed: what methodology may be applicable to efficiently select distribution center locations and connect the remaining cities to these distribution centers?

In Chapter 2, the direct definition of the problem is outlined and in Chapter 3, an ILP model is given as a solution to this problem. This strategy gives an optimal solution in a very slow manner, thus in Chapter 4, two different heuristic algorithms are given. These both provide possibly sub-optimal solutions. Finally, in Chapter 5, the results of the three strategies will be shown and their effectiveness will be compared.

In Chapter 6, an addition to the existing problem is made: the so-called Node Transport Modules (NTMs) and Hub Transport Modules (HTMs) are added into the equation. These cost-lowering modules build on the situation by giving more methods of transportation. Furthermore, in Chapter 7 the ILP shown in Chapter 3 is simplified and an ILP-strategy for this additional problem is detailed.

2

Transportation Problem

In the problem, a given number of cities have to be connected in such a way that a package can be transported from any city to any other city. This could be interpreted as packages being transported from consumer to consumer, or from producer to producer. The transport through cities themselves can be neglected in this problem.

Packages cannot be directly delivered from one city to another by default. There are three steps:

1. Bring the package to a sorting center.
2. Bring the package to a distribution center.
3. Deliver the package to its destination.

By default, any sorting center can function as a distribution center and vice versa. All three of these steps have certain costs to perform. These are given by the cost to travel between two cities and a multiplier that is different for each step. Not every city may function as a distribution center: a hub has to be built in the city before it can act as a distribution facility. The construction of hubs also has costs and these are different per city.

A hub may function as both a sorting center and a distribution center. However, it is usually inefficient for a hub to fulfill both of these purposes for one specific path: two hubs are utilized in most but not all circumstances to deliver a package from one location to another.

If a hub is built in a specific city, this city is automatically connected to every other city with a hub, and it may also be connected to every city that is not a hub. Cities without a hub must only be connected to one city with a hub. A result of this requirement is that, by taking all three of the steps mentioned earlier into account, the package only has to travel through a minimum of 2 and a maximum of 4 cities.

To further calculate the cost, the amount of packages that have to be transported from each city to every city is given in the problem, and this will be multiplied with the cost and multipliers mentioned earlier, to which the costs to build the hubs can be added to give the total cost.

The goal is to be able to transport all the packages for as little money as possible. To do this, the right number of hubs needs to be determined and the ideal locations of the hubs need to be found, which can be done in any order. When this is done, the cities that are not hubs need to be connected to the correct hubs and then the system is complete.

Two trivial solutions to the problem that do not give the optimal solution are:

- Build a single hub and every package goes through this hub. This way the cost to build every hub (only one) is low, but the transportation costs will be very high.
- Build a hub in every city. This would make the transportation costs minimal, but the costs to build the hubs are too high for this to be the optimal solution.

Thus the optimal solution has more than one, but fewer hubs than the number of cities there are. This could be different if the costs to build hubs and the costs to transport between cities were different, but in this situation, that is not the case.

3

The ILP Model for Transportation Problem

Integer linear programming (ILP) is a mathematical method for solving optimization problems in which the variables are restricted to integers. This Chapter aims to provide an ILP model as a solution to the problem in Chapter 2. To this end, the input and variable definitions are first presented in Section 3.1 after which the complete ILP model is given in Section 3.7. Since the length of the objective function and list of constraints are relatively substantial, in Sections 3.2 through 3.6 the details for the complete derivation are provided.

3.1. The input and variables definitions

Input. Let

$$
\begin{aligned}
N &= \text{set of nodes,}\\
f_i &= \text{the fixed costs to build a hub at location } i, \text{ for } i \in N,\\
w_{ij} &= \text{the amount of packages or flow that goes from } i \text{ to } j, \text{ for } i, j \in N,\\
c_{ij} &= \text{the cost to transfer 1 package from } i \text{ to } j, \text{ for } i, j \in N,\\
\chi &= \text{multiplier of collection costs for a package going from a node to a hub,}\\
\alpha &= \text{multiplier of transfer costs for a parcel going from a hub to another hub,}\\
\delta &= \text{multiplier of distribution costs for a parcel going from a hub to a node.}
\end{aligned}
$$

c_{ii} is equal to 0 for all $i \in N$ in all the data sets used in this paper.

Variable definition. Let

$$
h_i = \begin{cases} 1, & \text{if } i \text{ is a hub} \\ 0, & \text{if } i \text{ is not a hub,} \end{cases}
$$

$$
e_{ij} = \begin{cases} 1, & \text{if there is an edge from } i \text{ to } j \\ 0, & \text{if there is no edge from } i \text{ to } j. \end{cases}
$$

The edges are undirected, so $e_{ij} = e_{ji}$ for all $i, j \in N$.

3.2. Dividing the objective function into sub-functions

To formulate the objective function (the function to min- or maximize), it is more effective to split the objective function up into a few more compact sub-functions. The objective function can be divided into four sub-functions that give expressions to calculate the following:

- cost of building the hubs,
- cost of packages that go from a hub to another hub,
- cost of packages that go from a hub to a non-hub or from a non-hub to a hub,
- cost of packages that go from a non-hub to another non-hub.

3.3. Cost of building the hubs

The total cost of building all the hubs can be calculated by going over all the nodes and only summing up the build cost of the hubs. This can be done with the following formula:

$$\sum_{i \in N} f_i h_i \tag{3.1}$$

This is because $h_i = 1$ if and only if i is a hub, so only the fixed costs of the nodes that are hubs are summed up.

3.4. Cost from hub to hub

The cost of all packages that go from a hub i to the same or another hub j is equal to the flow multiplied by the cost and the transfer multiplier, or $w_{ij}\alpha c_{ij}$. The cost c_{ij} needs to be added to the total cost if and only if h_i and h_j are both equal to 1, which can be modelled by multiplying it with $h_i h_j$. It is however necessary to linearize $h_i h_j$. This can be achieved by replacing $h_i h_j$ with t_{ij} and adding the following constraints:

$$t_{ij} \leq h_i, \qquad i, j \in N \tag{3.2}$$
$$t_{ij} \leq h_j, \qquad i, j \in N \tag{3.3}$$
$$t_{ij} \geq h_i + h_j - 1, \quad i, j \in N \tag{3.4}$$
$$t_{ij} \in \{0, 1\}, \qquad i, j \in N. \tag{3.5}$$

The total cost from hub to hub is now given by

$$\sum_{i \in N} \sum_{j \in N} t_{ij} w_{ij} \alpha c_{ij}. \tag{3.6}$$

3.5. Cost from hub to non-hub and vice-versa

The path of a package that goes from a hub i to a non-hub j has to go through a hub k that is connected to j. Hub k can be any hub including i. This means that the cost per unit flow along this route ($i \to k \to j$) can be calculated with $\alpha c_{ik} + \delta c_{kj}$. Reversing this route then gives rise to the path of a package from non-hub j to i through k. Similarly, the cost per unit flow along this route ($j \to k \to i$) is given by $\chi c_{jk} + \alpha c_{ki}$. To prevent double-counting when summing over all $i, j, k \in N$, it is necessary to pick the reference point of view of either the first or the second path. One can for instance choose for the first path, which signifies that i is the (origin) non-hub and j is the (destination) non-hub that is reached through hub k. This is the case if and only if h_i, $(1 - h_j)$ and e_{kj} are all equal to 1, since e_{ik} is already equal to one by assumption as i and k are hubs (see Chapter 2). This can be modelled by multiplying the costs with $h_i \cdot (1 - h_j) \cdot e_{kj}$, which can again be linearized by replacing it with t_{ijk} and adding the following constraints:

$$t_{ijk} \leq 1 - h_i, \qquad i, j \in N \tag{3.7}$$
$$t_{ijk} \leq h_j, \qquad i, j \in N \tag{3.8}$$
$$t_{ijk} \leq e_{ik}, \qquad i, j \in N \tag{3.9}$$
$$t_{ijk} \geq 1 - h_i + h_j + e_{ik} - 2, \quad i, j \in N \tag{3.10}$$
$$t_{ijk} \in \{0, 1\}, \qquad i, j \in N. \tag{3.11}$$

However, problems can arise if $e_{yy} = 1$ for some non-hub y. Namely, the route $x \to y$ (where x is a hub) can be achieved by setting either $i = x$, $k = x$ and $j = y$ or $i = x$, $k = y$ and $j = y$, which results

in double-counting costs when summing over all $i, j, k \in N$. Therefore, it is necessary to restrain the undesired second path ($x \to y \to y$) by disallowing a connection from a non-hub to itself, which can be achieved by using the following constraint:

$$e_{ii} = h_i, \quad i \in N. \tag{3.12}$$

Note that connections to itself are only disallowed for non-hubs, but since the value of e_{ii} for a hub i does not play a role in the cost calculations, it is justified to set the value for e_{ii} equal to 1. Additionally, this constraint gives a motivation to disregard the check whether or not k is a hub, because if $e_{kj} = 1$, then k must be a hub. Finally, multiplying the costs with the proper flow gives the total cost from hub to non-hub and vice-versa:

$$\sum_{i \in N} \sum_{j \in N} \sum_{k \in N} t_{ijk} \Big(w_{ij}(\alpha c_{ik} + \delta c_{kj}) + w_{ji}(\chi c_{jk} + \alpha c_{ki}) \Big) \tag{3.13}$$

3.6. Cost from non-hub to non-hub

The path of a package that goes from a non-hub i to a non-hub j has to go through hubs k and l, where k is connected with i and l with j. This means that the cost per unit flow along this route ($i \to k \to l \to j$) is given by $\chi c_{ik} + \alpha c_{kl} + \delta c_{lj}$. Note that this includes the costs from i to i ($i \to k \to k \to i$) since the cost $c_{ii} = 0$ for all $i \in N$ (see the problem explanation in Chapter 2), so the cost from k to k is zero. When summing over $i, j, k, l \in N$, it is necessary to ensure that i is a non-hub, j is a non-hub, i is connected with k and j is connected with l (it does not need to be checked that k and l are hubs, since i and j have a single connection). This is the case if and only if $(1 - h_i)$, $(1 - h_j)$, e_{ik} and e_{lj} are equal to 1, since if $e_{ik} = e_{lj} = 1$, then k and l are automatically hubs by constraint 3.12 and $e_{kl} = 1$ by assumption (see Chapter 2). This can be modelled by multiplying the costs with $(1 - h_i) \cdot (1 - h_j) \cdot e_{ik} \cdot e_{lj}$, which can once again be linearized by substituting it with t_{ijkl} and adding the following constraints:

$$t_{ijkl} \leq 1 - h_i, \qquad i, j \in N \tag{3.14}$$
$$t_{ijkl} \leq 1 - h_j, \qquad i, j \in N \tag{3.15}$$
$$t_{ijkl} \leq e_{ik}, \qquad i, j \in N \tag{3.16}$$
$$t_{ijkl} \leq e_{lj}, \qquad i, j \in N \tag{3.17}$$
$$t_{ijkl} \geq 1 - h_i + 1 - h_j + e_{ik} + e_{lj} - 3, \qquad i, j \in N \tag{3.18}$$
$$t_{ijkl} \in \{0, 1\}, \qquad i, j \in N. \tag{3.19}$$

Note that $e_{ii} = 1$ allows for undesired paths like $i \to i \to k \to i$, where i is a non-hub and k a hub. Constraint 3.12 is once again applied to solve this issue. Multiplying the costs with the proper flow now gives the total cost from non-hub to non-hub:

$$\sum_{i \in N} \sum_{j \in N} \sum_{k \in N} \sum_{l \in N} t_{ijkl} w_{ij} (\chi c_{ik} + \alpha c_{kl} + \delta c_{lj}) \tag{3.20}$$

3.7. The complete ILP model

There are a few extra constraints that are necessary for the ILP. These are:

1. h_i and e_{ij} are binary variables, so $h_i, e_{ij} \in \{0, 1\}$ for all $i, j \in N$.
2. There is a least one hub, so $\sum_{i \in N} h_i \geq 1$.
3. A non-hub only has a single connection, but a hub can be connected to everything, so $1 \leq \sum_{j \in N} e_{ij} \leq 1 + (|N| - 1) \cdot h_i$ for all $i \in N$.

By combining the sub-functions 3.1, 3.6, 3.13 and 3.20 to obtain the objective function and combining the constraints 3.2 through 3.5, 3.7 through 3.11, 3.12, 3.14 through 3.19 and item 1 through 3 from above,

the complete model then becomes

$$\begin{aligned}
\min z,\ \text{where } z = & \sum_{i\in N} f_i h_i + \\
& \sum_{i\in N}\sum_{j\in N} t_{ij} w_{ij} \alpha c_{ij} + \\
& \sum_{i\in N}\sum_{j\in N}\sum_{k\in N} t_{ijk}\Big(w_{ij}(\alpha c_{ik} + \delta c_{kj}) + w_{ji}(\chi c_{jk} + \alpha c_{ki})\Big) + \\
& \sum_{i\in N}\sum_{j\in N}\sum_{k\in N}\sum_{l\in N} t_{ijkl} w_{ij}(\chi c_{ik} + \alpha c_{kl} + \delta c_{lj})
\end{aligned}$$

$$\begin{aligned}
\text{s.t.}\quad t_{ij} &\le h_i, && i,j \in N \\
t_{ij} &\le h_j, && i,j \in N \\
t_{ij} &\ge h_i + h_j - 1, && i,j \in N \\
t_{ij} &\in \{0,1\}, && i,j \in N \\
t_{ijk} &\le 1 - h_i, && i,j \in N \\
t_{ijk} &\le h_j, && i,j \in N \\
t_{ijk} &\le e_{ik}, && i,j \in N \\
t_{ijk} &\ge 1 - h_i + h_j + e_{ik} - 2, && i,j \in N \\
t_{ijk} &\in \{0,1\}, && i,j \in N \\
t_{ijkl} &\le 1 - h_i, && i,j \in N \\
t_{ijkl} &\le 1 - h_j, && i,j \in N \\
t_{ijkl} &\le e_{ik}, && i,j \in N \\
t_{ijkl} &\le e_{lj}, && i,j \in N \\
t_{ijkl} &\ge 1 - h_i + 1 - h_j + e_{ik} + e_{lj} - 3, && i,j \in N \\
t_{ijkl} &\in \{0,1\}, && i,j \in N \\
e_{ii} &= h_i, && i \in N \\
h_i, e_{ij} &\in \{0,1\}, && i,j \in N \\
\sum_{i\in N} h_i &\ge 1, \\
1 \le \sum_{j\in N} e_{ij} &\le 1 + (|N| - 1)\cdot h_i, && i \in N.
\end{aligned}$$

The Simplex Algorithm is a popular algorithm for linear programming and Harvey J. Greenberg proved that worst case, the runtime complexity is exponential**[1]**. It is therefore assumed that the PuLP, Gurobi and CPLEX solvers also have an exponential runtime complexity.

4

Heuristic Algorithms

Heuristic algorithms sacrifice optimality for speed with as goal providing a good approximation in a timely fashion. This is different to the ILP solution in Chapter 3, which does guarantee to give the optimal solution, but is computationally expensive. This Chapter aims to provide some heuristic algorithms as solutions to the problem in Chapter 2. First, in Section 4.1 the heuristic that is used in both algorithms is introduced. After that, the heuristic algorithms are presented in Section 4.2 and Section 4.3. Finally, a summary is given in Section 4.4.

4.1. The CNC Heuristic

A heuristic is a function that ranks the next possible branches at each step of the algorithm by using the currently available information and therefore reducing the number of considerations. A heuristic algorithm uses heuristics to quickly solve an optimization problem in a possibly sub-optimal manner.

In both algorithms below, the cheapest node connection (CNC) heuristic is used. The CNC heuristic says that every non-hub i must be connected to hub j such that c_{ij} is minimal. This means that the problem in Chapter 2 is now reduced to finding the optimal hub selection since the problem of connecting the non-hubs is solved using the CNC heuristic.

4.2. Using an Exhaustive Search

The first heuristic algorithm is an exhaustive search. It will consider all possible hub selections and select the one with the minimal z-value (see Section 3.7). The algorithm can be described as follows:

1. Repeat steps 2 through 4 for every current hub selection $C \in \mathcal{P}(N)$ and keep track of the minimum costs $z_{min} := \infty$ with respect to (minimal) hub selection S.
2. Determine h_i for all $i \in N$ using C by setting $h_c = 1$ for all $c \in C$ and $h_r = 0$ for all $r \in N \setminus C$.
3. Determine the connections e_{ij} for all $i, j \in N$ by using the CNC heuristic, connecting every hub to every other hub (including itself) and setting $e_{ij} = \max\{e_{ij}, e_{ji}\}$ for every $i, j \in N$.
4. Calculate the costs z using h_i and e_{ij} and if $z \leq z_{min}$, update z_{min} and S accordingly.
5. The solution is now determined by S (h_i and e_{ij} can be computed by using S in a similar fashion to steps 2 and 3).

The main disadvantage of this algorithm is that it is very time-inefficient with a time complexity of $O(2^n n^4)$, where n denotes the amount of nodes. This is because it tests a significant number of hub selections, which will likely not produce a minimal z-value. When for instance $|C| > \left\lfloor \frac{|N|}{2} \right\rfloor$, there will be isolated hubs, namely hubs that are not connected to any non-hub. This is only cost-efficient if the flow from an isolated hub to itself is large, as the cost from a hub to itself is zero. However, the relatively

high fixed costs of building a hub often negate this cost improvement. The multipliers χ, α and δ do however form a motivation to build isolated hubs if the ratio is sufficiently skewed. Namely, when the transfer multiplier is significantly lower than the other multipliers, it is cheap to maximize the amount of packages sent between hubs. As a result, building isolated hubs as opposed to leaving them as nodes might be interesting.

4.3. Using a Greedy Algorithm

The second heuristic algorithm is a greedy algorithm. At each step, it will opt for the optimal hub selection at that stage. The algorithm can be described as follows:

1. Start with an empty hub solution selection S, so $S = \emptyset$, and define the remaining hubs as $R = N \setminus S$ (note that R is dependent on S). Keep track of the minimum costs with z_{min}, which is initially equal to ∞.
2. Repeat steps 3 through 5 for the current hub selection C for $C = S \cup \{r\}$ for every $r \in R$ and keep track of the hub in R that adds the fewest to the costs z with h_{min}.
3. Determine h_i for all $i \in N$ using C by setting $h_c = 1$ for all $c \in C$ and $h_r = 0$ for all $r \in N \setminus C$.
4. Determine the connections e_{ij} for all $i, j \in N$ by using the CNC heuristic, connecting every hub to every other hub (including itself) and setting $e_{ij} = \max\{e_{ij}, e_{ji}\}$ for every $i, j \in N$.
5. Calculate the costs z using h_i and e_{ij} and update z_{min} and h_{min} accordingly if $z \leq z_{min}$.
6. Go to step 8 if z_{min} has not been updated. Add h_{min} to S otherwise.
7. Repeat step 2 until R is empty.
8. The solution is now determined by S (h_i and e_{ij} can be computed by using S in a similar fashion to steps 3 and 4).

The main disadvantage of this algorithm is that it might hone into a local minimum instead of the global minimum with respect to the CNC-heuristic, which is guaranteed to be obtained by the exhaustive heuristic algorithm. So by using a sufficiently skewed data set, the hub selection found by the exhaustive heuristic algorithm might be more optimal than the one found by the greedy heuristic algorithm. The advantage over the exhaustive heuristic algorithm however, is its polynomial time complexity $O(n^6)$ (as opposed to an exponential one).

4.4. Summary

The heuristic used in both algorithms is the CNC heuristic, which states that every non-hub must be connected to the hub with the cheapest edge. The first algorithm is exhaustive and considers all possible hub selections, while the second algorithm is greedy and iteratively chooses the locally optimal hub selection. The exhaustive search algorithm has the main disadvantage of being computationally expensive and the main disadvantage of the greedy algorithm is that it might produce a local optimum. In Section 5.2, the advantages and disadvantages of the heuristic, the heuristic algorithms and the ILP will be elaborated on further and compared.

5

Results

In Chapter 3 and Chapter 4, solutions to the problem in Chapter 2 were provided. In this Chapter, their results of applying them on the small and large data set are first presented in Section 5.1, after which the algorithms are compared and discussed in Section 5.2 and finally a summary is given in Section 5.3.

5.1. Results

The results of applying the algorithms in Chapter 3 and Chapter 4 on the small and big data set (see Section A.1) can be found in Table 5.2 and Figures 5.2 through 5.5. The differences between the ILP and the heuristics have been highlighted in red and the similarities in green. The thickness of the edge represents the edge's cost. Interestingly, for both the small and large data sets, the exhaustive heuristic algorithm produces the same results as the greedy one. Furthermore, the amount of hubs in all cases is three. The solution graph for the small data set from the ILP is identical to the one from the heuristics except for node 5. For the large data set, almost the entire graph produced by the heuristics is different from the ILP, but oddly enough, the costs are relatively similar, with an increase in costs of 1.72%.

Since running PuLP on the large data set takes an unreasonable amount of time, an estimate is made by curve fitting with the Python Library SciPy. First, the runtimes are measured for data sets of lengths one through ten. These data sets can be obtained by starting with the large data set and for example only using the data for nodes 1 through 5 in the case of a data set with length five. Since the time complexity is exponential (see the end of Section 3.7), the objective function used for curve fitting the runtime is $f(n) = c_1 \cdot c_2^{n-1}$ [1] Then the constants c_1 and c_2 are estimated for an amount of different samples n_{max}. So, for example, if $n_{max} = 4$, a curve fit is made using the runtimes for data sets of lengths one, two, three and four.

In Figure 5.1, the thick, blue line represents the runtime in seconds and the other lines represent the curve fits for different n_{max}. The estimate for $n_{max} = 1$ has been left out, since SciPy does not have enough data in this case (the objective function f has two parameters). From this Figure, it can be deduced that the estimated runtime with respect to n is generally underestimated if $n_{max} < n$, which implies that the error between the estimated runtime and actual runtime in seconds grows relatively rapid as n increases. For example, for $n = 10$, $n_{max} = 2$ through $n_{max} = 9$ give estimated runtimes of 243, 457843, 5071, 1770, 4671, 2821 and 4749 seconds respectively, while the actual runtime for $n = 10$ is 12473 seconds. In Table 5.1, the values for parameters c_1 and c_2 of the objective function f are given for different values for n_{max}. Using the values for $n_{max} = 10$, $f(15)$ then gives an estimated runtime of 54 years for running PuLP on the complete large data set.

[1] In theory, replacing $n-1$ with n in f does not make a difference, since the difference is just a constant, which can be accounted for with c_1. In practice however, this gives an error when using SciPy, which is why $n-1$ is used instead.

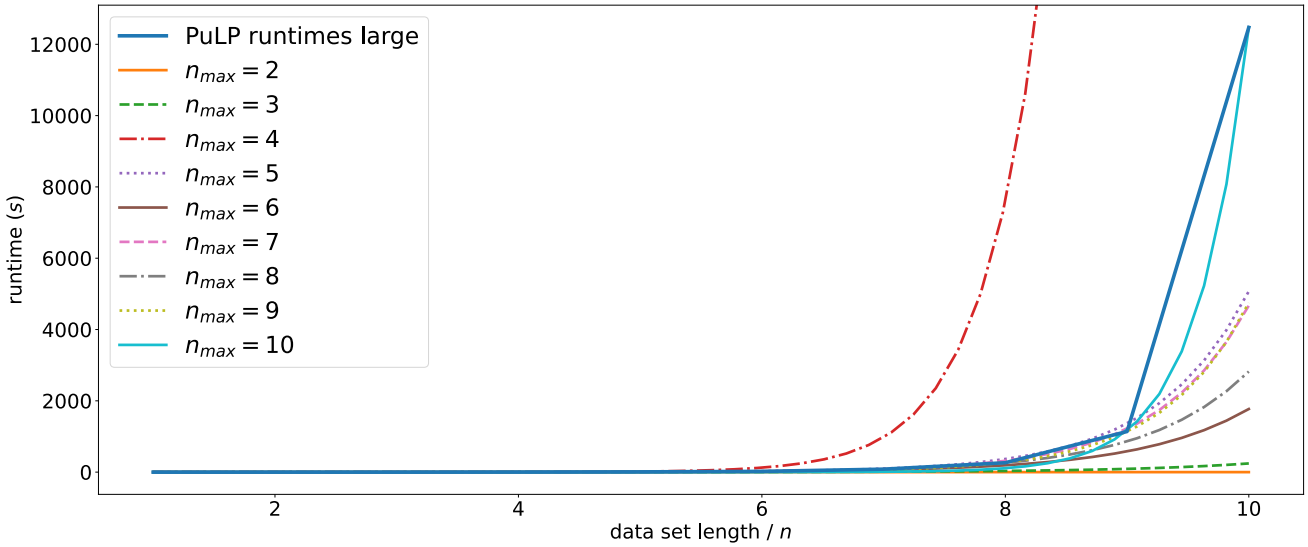


Figure 5.1: The data set of length n is obtained by using the data from the large data set for the nodes up to and including n. The thick, blue line represents the runtime of running PuLP on the large data set in seconds and all the other lines represent the curve fits (estimates) for different n_{max}, where n_{max} is the amount of samples used from the runtime, so that only the nodes up to and including n_{max} are used for the estimate.

n_{max}	2	3	4	5	6	7	8	9	10
c_1	0.065050	0.035405	0.004463	0.037794	0.080833	0.026272	0.065947	0.012437	0.000007
c_2	1.111178	2.669028	7.764601	3.713175	3.035805	3.831145	3.270264	4.170650	10.655219

Table 5.1: Values for parameters c_1 and c_2 of the objective function f (which estimates the runtime for the scaled, large data set of length n) for different values of n_{max}.

Small Data Set		
	Objective z-value	Execution time (HH:MM:SS) [2]
ILP Model (PuLP)	234.443	06:52:36
ILP Model (Gurobi)	234.443	00:00:21
ILP Model (CPLEX)	234.443	00:02:21
Exhaustive Heuristic Algorithm	234.953	00:00:18
Greedy Heuristic Algorithm	234.953	00:00:01
Large Data Set		
	Objective z-value	Execution time (HH:MM:SS)
ILP Model (PuLP)	136.832	54 years*
ILP Model (Gurobi)	136.832	00:44:33
ILP Model (CPLEX)	136.832	17:37:35
Exhaustive Heuristic Algorithm	139.184	00:39:03
Greedy Heuristic Algorithm	139.184	00:00:04

Table 5.2: The results of applying the algorithms in Chapter 3 and Chapter 4 on the small and big data set (see Section A.1). * is estimated using curve fitting, see Section 5.1 and Table 5.1.

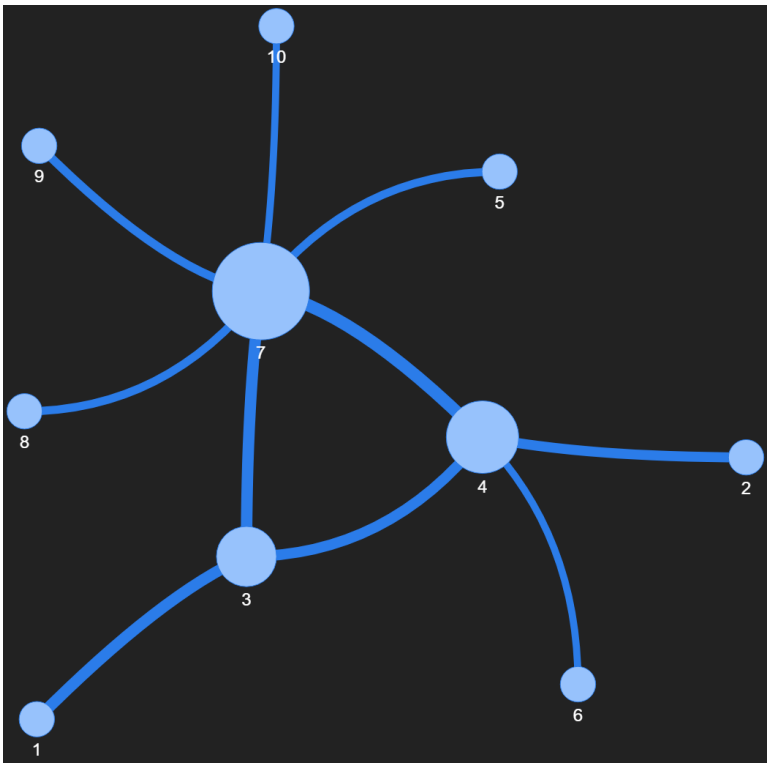


Figure 5.2: Results of applying the ILP solver with a PuLP / Gurobi / CPLEX implementation on the small data set. The thickness of the edge represents the cost of the edge.

[2]To provide some context for the runtime, the Python scripts in Chapter B have been run on a Windows 10 Pro machine with 64-bits processor and an Intel(R) Core(TM) i7 CPU 860 at 2.80 GHz, which has 4 cores.

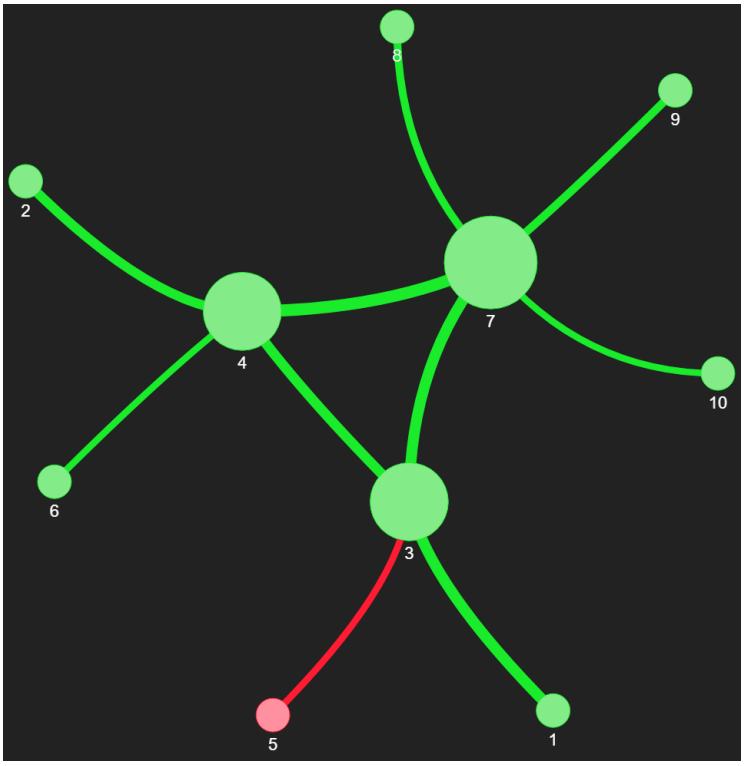


Figure 5.3: Results of applying the exhaustive / greedy heuristic algorithm on the small data set. The thickness of the edge represents the cost of the edge. In green the nodes that are the same as in the optimal solution and in red the nodes that are different in the optimal solution.

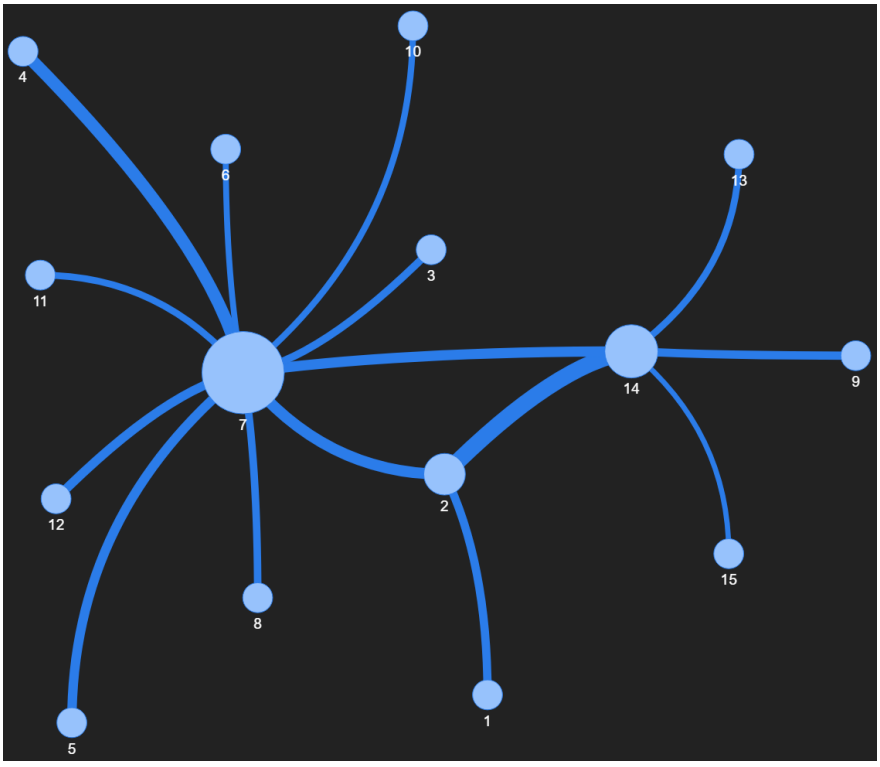


Figure 5.4: Results of applying the ILP solver with a Gurobi / CPLEX implementation on the large data set. The thickness of the edge represents the cost of the edge.

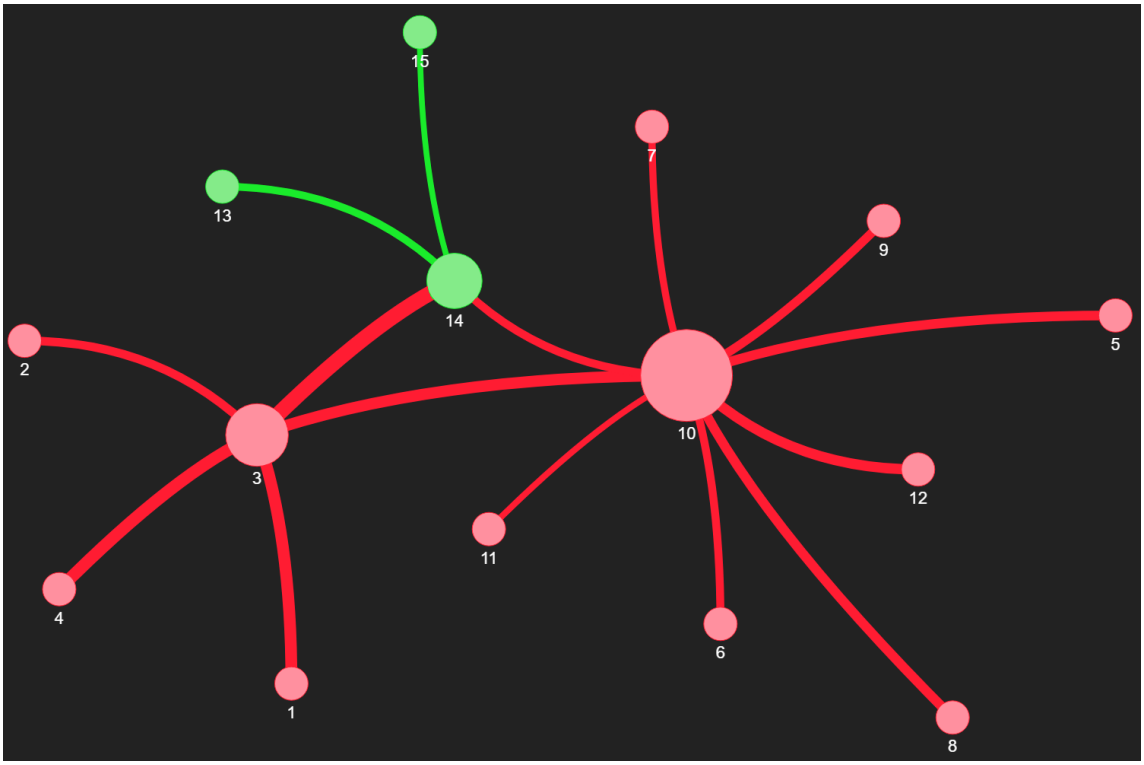


Figure 5.5: Results of applying the exhaustive / greedy heuristic algorithm on the large data set. The thickness of the edge represents the cost of the edge. In green the nodes that are the same as in the optimal solution and in red the nodes that are different in the optimal solution.

5.2. Discussion

To compare the various algorithms, it is useful to analyse the differences when applying the algorithms on random, realistic data sets. To this end, the following strategy is used to generate these data sets. The generated data sets will be either of size 10 or 15, since the data sets are based on the small and large data sets, which are assumed to be realistic. For the generation of the data, a normal distribution is made from the mean and standard deviation of the fixed hub costs, flow and costs (the multipliers are kept the same as the small and large data sets). These normal distributions are then used to generate new data sets, where specifically for the costs, c_{ii} is set to 0 for all $i \in N$.

First of all, it is clear that Gurobi is the preferred ILP algorithm since it has the lowest runtime. Furthermore, applying the exhaustive and greedy heuristic algorithms on 50 random data sets of length 10 shows that for some data sets the greedy heuristic algorithm indeed hones into a local minimum as on average the total costs (z-value) are 2.14% higher. However, since the runtimes of the exhaustive heuristic algorithm and Gurobi are of the same order, the heuristic algorithm serves no advantage over using Gurobi and hence, the greedy heuristic algorithm is the preferred algorithm of the two. Running Gurobi and the greedy heuristic algorithm on another 50 random data sets of length 10 gives an average increase in costs of 3.80% and with a length of 15, the average increase is 5.13%. Thus, if there is a lack of time or the length of the data set is relatively large, resulting in unfeasible runtimes, it might be best to opt for the greedy heuristic algorithm as opposed to Gurobi. Simultaneously, the average increase in costs obtained by using the greedy heuristic algorithm instead of Gurobi does seem to grow as n grows, which is a another downside to consider.

5.3. Conclusion

For both data sets, the costs from the ILP and the heuristics are similar (less than 1% and 2% for the small and large data set respectively), the results of the exhaustive and greedy heuristic algorithm are identical and in all cases, the amount of hubs is three. From the runtimes it is clear that Gurobi

is the preferred algorithm for the ILP and that the exhaustive heuristic algorithm does not provide any additional advantage over the greedy heuristic algorithm. By applying the algorithms on random, realistic data sets, an average difference in price of 3.80% between Gurobi and the greedy heuristic algorithm is found for the small data set and 5.13% for the large data set. From this, it can be concluded that for significantly large enough runtimes, it is best to use the greedy heuristic algorithm and Gurobi otherwise.

6

Problem Expansion

In the base problem, the costs of transportation between two directly connected nodes are described with one non-variable value. This value possibly represents the currently assigned costs of that connection using one specific strategy. However, in real life, there exist multiple ways to transport packages, and different methods usually come with different costs. Not every mode of transport may be used in every circumstance. Larger, more fuel-consuming vehicles are often more expensive and may have difficulty navigating tighter areas, while smaller vehicles may be less costly, but they aren't capable of transporting packages over large distances.

Currently, it is pretty common to see 3 different modes of transport in the field of product delivery: trucks exchanging packages between larger cities, semi-trucks delivering packages between smaller cities and smaller vehicles like vans and cargo bikes delivering packages from door to door.

Suppose we want to implement this kind of distinction in terms of transportation in the given model. We define the following additional modes of transport:

- Hub Transport Modules (HTM):

  HTMs, representing larger trucks, may only transport inbetween two assigned hubs. This lowers the cost of transportation between these two hubs by 50%, but has a fixed cost of 4000 allocated to its implementation on a connection. Furthermore, HTMs will be limited to 2 in total in a given situation.

- Node Transport Modules (NTM):

  NTMs, representing smaller vehicles like cargo bikes, may only transport from an assigned hub to all its directly linked non-hub nodes and back. This lowers the cost of transportation by 50% between these locations, but, like the HTMs, they have a fixed cost of 3000 on their implementation on a hub. NTMs are also restricted to only 2 implementations on a given situation.

By adding these two different factors into the equation at the beginning, the set of nodes will likely be arranged differently. Some non-hubs might be turned into hubs, and vice versa. Analysing this situation may give more insight into the efficiency of different forms of delivery, as well as providing more understanding of certain business-related decisions in both the model and real life.

7

The ILP Model for the Extended Problem

Because the problem outlined in Chapter 6 is more complicated than the problem described in Chapter 2, the original ILP in Section 3.7 is too slow to solve the expanded problem in a reasonable amount of time. By changing the original ILP, it can be made quicker so that the problem in Chapter 6 can be solved more efficiently.

Firstly, the variables are defined in Section 7.1. Secondly, in Section 7.2 the ILP from Chapter 3 will be updated to utilize the new variables. In Section 7.3, new constraints corresponding to the implementation of HTMs and NTMs are added into the equation, and these additions are put together into a new ILP in Section 7.4

7.1. Variables for the new ILP

N = set of nodes,
f_i = the fixed costs to build a hub at location i, for $i \in N$,
w_{ij} = the amount of packages or flow that goes from i to j, for $i, j \in N$,
c_{ij} = the cost to transfer 1 package from i to j, for $i, j \in N$,
χ = multiplier of collection costs for a package going from a node to a hub,
α = multiplier of transfer costs for a parcel going from a hub to another hub,
δ = multiplier of distribution costs for a parcel going from a hub to a node,
ξ = multiplier of the cost if the node is an NTM or an HTM.

Variable definition. Let

$$a_{ij} = \begin{cases} 1, & \text{if j is a hub and i and j are connected for } i, j \in n \\ 0, & \text{in all other situations,} \end{cases}$$

$$m_{ij} = \begin{cases} 1, & \text{if the edge from i to j is an HTM for } i,\ j\ \in n \\ 0, & \text{if the edge from i to j is not an HTM for } i,\ j\ \in n, \end{cases}$$

$$v_i = \begin{cases} 1, & \text{if the hub i is an NTM for } i \in n \\ 0, & \text{if the hub i is not an NTM for } i \in n. \end{cases}$$

7.2. Updated Equation

By using a_{ij}, the ILP for the problem in Section 3.7 can be simplified to:

$$\min z, \text{ where } z = \sum_{i \in N} f_i a_{ii} + \sum_{i \in N} \sum_{j \in N} \sum_{k \in N} \sum_{l \in N} p_{ijkl} w_{ij} (\chi c_{ik} + \alpha c_{kl} + \delta c_{lj}).$$

$$\begin{aligned}
\text{s.t.} \quad & p_{ijkl} \leq a_{ik}, && i,j,k,l \in N \\
& p_{ijkl} \leq a_{jl}, && i,j,k,l \in N \\
& p_{ijkl} \geq a_{ik} + a_{jl}, && i,j,k,l \in N \\
& p_{ijkl} \in \{0,1\}, && i,j,k,l \in N \\
& a_{ii} \in \{0,1\}, && i \in N \\
& \sum_{j \in N} a_{ij} \geq 1 && i \in N.
\end{aligned}$$

The ILP functions because p_{ijkl} is only equal to one if the path from i to k and the path from l to j exist. Because k and l are both hubs, k and l must also be connected, so there is a path from i to j. All that is left to do is to calculate the cost of the path. This is done in the same way as in Section 3.7.

7.3. Explanation of the new Constraints

There are three new situations to add to the ILP:

- The hub k is an NTM so the collection costs get multiplied by ξ.
- The hub l is an NTM so the distribution costs get multiplied by ξ.
- The edge between hub k and hub l is an HTM so the transfer costs get multiplied by ξ.

This can be implemented by subtracting $1 - \xi$ times the cost of the path, if there is an NTM or an HTM.

7.3.1. Hub k or hub l is an NTM

Finding out if hub k is an NTM is easy because $v_k = 1$ if k is an NTM. The next step is multiplying v_k with p_{ijkl}. All that is left to do is multiplying this with $(1 - \xi)\chi c_{ik} w_{ij}$ and subtracting this from the ILP from Section 7.2. But $v_k p_{ijkl}$ is not linear. It can be linearized by substituting it with q_{1ijkl} and adding the following constraints:

$$q_{1ijkl} \leq p_{ijkl}, \qquad i,j,k,l \in N \tag{7.1}$$
$$q_{1ijkl} \leq v_k, \qquad i,j,k,l \in N \tag{7.2}$$
$$q_{1ijkl} \geq p_{ijkl} + v_k - 1, \qquad i,j,k,l \in N \tag{7.3}$$
$$q_{1ijkl} \in \{0,1\}, \qquad i,j,k,l \in N. \tag{7.4}$$

In the situation where l is an NTM the steps are mostly the same. First multiplying v_l with $pijkl$, then all that is left to do is multiplying it with $(1 - \xi)\delta c_{lj} w_{ij}$. Subtracting all of this from the ILP from Section 7.2. But $v_l p_{ijkl}$ is not linear. It can be linearized by substituting it with q_{2ijkl} and adding the following constraints:

$$q_{2ijkl} \leq p_{ijkl}, \qquad i,j,k,l \in N \tag{7.5}$$
$$q_{2ijkl} \leq v_l, \qquad i,j,k,l \in N \tag{7.6}$$
$$q_{2ijkl} \geq p_{ijkl} + v_l - 1, \qquad i,j,k,l \in N \tag{7.7}$$
$$q_{2ijkl} \in \{0,1\}, \qquad i,j,k,l \in N. \tag{7.8}$$

7.3.2. There is an HTM

For the HTM the steps are once again mostly the same. The edge between k and l is an HTM if $m_{kl} = 1$. So $mklp_{ijkl}$ is only equal to 1 if the path exists and $k \leftrightarrow l$ is an HTM. All that is left to do is multiplying this with $(1 - \xi)\alpha c_{kl} w_{ij}$ and subtracting all of this from the ILP from Section 7.2. But $m_{kl} p_{ijkl}$ is not linear. It can be linearized by substituting it with q_{3ijkl} and adding the following constraints:

$$q_{3ijkl} \leq p_{ijkl}, \qquad i, j, k, l \in N \tag{7.9}$$
$$q_{3ijkl} \leq m_{kl}, \qquad i, j, k, l \in N \tag{7.10}$$
$$q_{3ijkl} \geq p_{ijkl} + m_{kl} - 1, \qquad i, j, k, l \in N \tag{7.11}$$
$$q_{3ijkl} \in \{0, 1\}, \qquad i, j, k, l \in N. \tag{7.12}$$

7.4. A new ILP

There are a few extra constraints that are necessary for the ILP. These are:

1. There are at most two NTMs, so $\sum_{i \in N} v_i \leq 2$. .
2. There are at most two HTMs, so $\sum_{i \in N} \sum_{j \in N} m_{ij} \leq 4$.
3. The paths of HTMs are symmetrical, so $m_{ij} = m_{ji}$

By combining the sub-functions from above to obtain the objective function and combining the constraints 7.1 through 7.8, 7.9 through 7.12 and item 1 through 3 from above, the complete model then becomes:

$$\begin{aligned} \min z, \text{ where } z = & \sum_{i \in N} f_i a_{ii} + \sum_{i \in N} \sum_{j \in N} \sum_{k \in N} \sum_{l \in N} p_{ijkl} w_{ij} (\chi c_{ik} + \alpha c_{kl} + \delta c_{lj}). \\ & - q_{1ijkl}(1 - \xi)\chi c_{ik} w_{ij} \\ & - q_{2ijkl}(1 - \xi)\delta c_{lj} w_{ij} \\ & - q_{3ijkl}(1 - \xi)\alpha c_{kl} w_{ij} \end{aligned}$$

$$
\begin{aligned}
\text{s.t.} \quad & p_{ijkl} \leq a_{ik}, && i,j,k,l \in N \\
& p_{ijkl} \leq a_{jl}, && i,j,k,l \in N \\
& p_{ijkl} \geq a_{ik} + a_{jl}, && i,j,k,l \in N \\
& p_{ijkl} \in \{0,1\}, && i,j \in N \\
& a_{ii} \in \{0,1\}, && i \in N \\
& \sum_{j \in N} a_{ij} \geq 1 && i \in N \\
& q_{1ijkl} \leq p_{ijkl}, && i,j,k,l \in N \\
& q_{1ijkl} \leq v_k, && i,j,k,l \in N \\
& q_{1ijkl} \geq p_{ijkl} + n_k - 1, && i,j,k,l \in N \\
& q_{1ijkl} \in \{0,1\}, && i,j,k,l \in N \\
& q_{2ijkl} \leq p_{ijkl}, && i,j,k,l \in N \\
& q_{2ijkl} \leq v_l, && i,j,k,l \in N \\
& q_{2ijkl} \geq p_{ijkl} + n_l - 1, && i,j,k,l \in N \\
& q_{2ijkl} \in \{0,1\}, && i,j,k,l \in N \\
& q_{3ijkl} \leq p_{ijkl}, && i,j,k,l \in N \\
& q_{3ijkl} \leq m_{kl}, && i,j,k,l \in N \\
& q_{3ijkl} \geq p_{ijkl} + m_{kl} - 1, && i,j,k,l \in N \\
& q_{3ijkl} \in \{0,1\}, && i,j,k,l \in N \\
& \sum_{i \in N} v_i \leq 2 \\
& \sum_{i \in N} \sum_{j \in N} m_{ij} \geq 4 \\
& m_{ij} = m_{ji}, && i,j \in N.
\end{aligned}
$$

7.5. Results

This ILP can give different results to the original ILP, not only by placing NTMs on and HTMs between existing hubs, but with certain values for discount, build cost and build limit the hubs may be built in entirely different places. Without any cost nor limit set to the NTMs and HTMs, the ILP would just give the same exact solution as before, but with NTMs and HTMs in every possible position, which would mean that the cost of transporting packages would be lowered by the set discount variable, while the cost to build hubs stays the same. This did not seem like an interesting result, so either a limit on the amount of NTMs and HTMs that could be built had to be set or there needed to be costs associated with building these.

With either of these limits installed, the discount still seemed to high initially. With the initial discount set at 50%, the ILP only built one hub and then sent all the packages through this one hub as this was ridiculously cheap, but this did again not seem like an interesting result. This meant that lowering the discount given by NTMs and HTMs would be necessary.

With a lower discount and build costs associated with the NTMs and HTMs, it was observed that one of two things would happen:

- The build costs were too high to make building NTMs and HTMs efficient, meaning that the solution would be the exact same as before the expansion.

- The ILP would still send all the packages through NTMs and HTMs, sometimes only building one hub again, giving the same result as it would give with a higher discount again.

As these results have already been deemed uninteresting, it can be concluded that there needs to be a limit on the number of available NTMs and HTMs, which will logically all be used up, as they only give a cost decrease.

Eventually, it was decided to use a single NTM and a single HTM, as this would make it so that the ILP could create a regular network without placing NTMs on every hub and HTMs between them. Then the discount needed to be determined. This was initially set to 10%, which did unfortunately not give an interesting result, as it was the same solution as the original ILP with an added NTM and HTM, but of course it did lower the cost a bit.

Then increasing the discount to 15% finally gave an interesting enough result, as one of the hubs got removed entirely and all the non-hubs this hub had connected were now connected to the NTM that was built. This is especially interesting because it could be interpreted as merging two hubs, which makes a lot of sense given the cost decrease between a certain pair of hubs. In the end, this solution gives a total cost of 221358 compared to 234443 in the original ILP, which means that 13085, or around 5.6%, was saved.

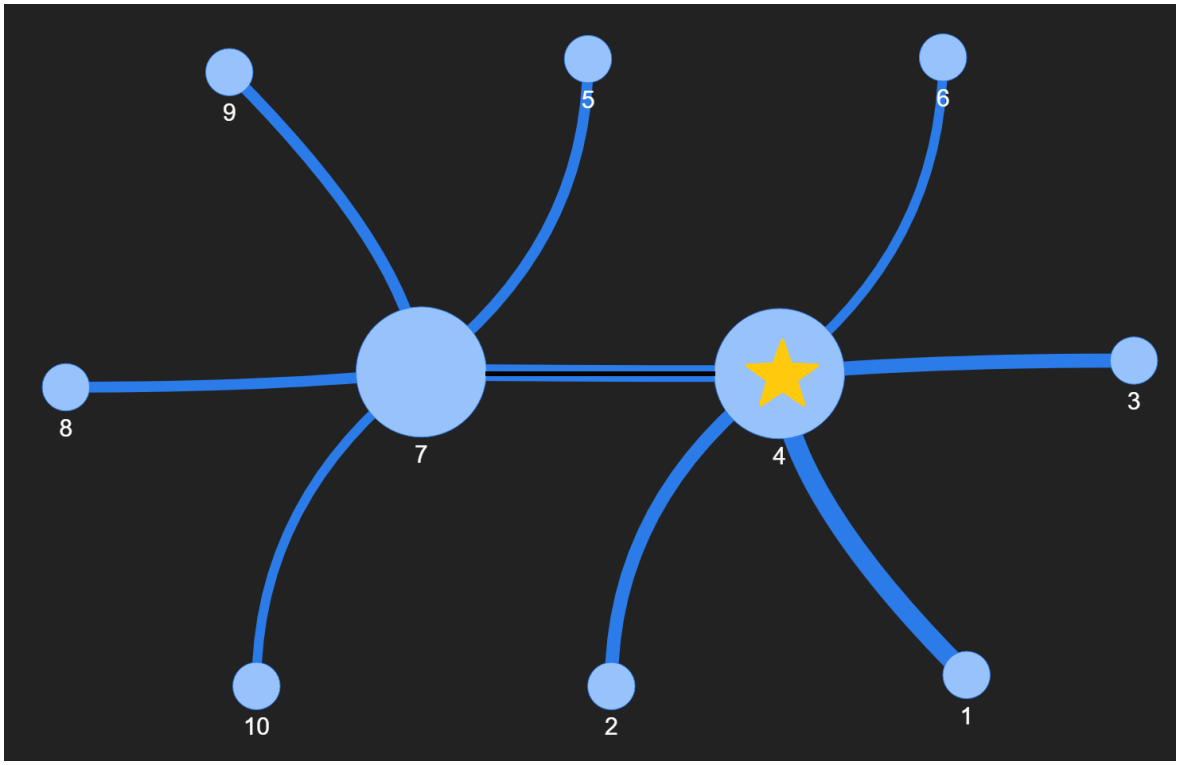


Figure 7.1: A star is placed on the NTM and a double line is used to show the HTM. The thickness of the edge represents the cost of the edge.

This is the solution for the small dataset, which already took the ILP around ten minutes to solve, so trying to solve the large dataset would be practically impossible considering the increase in solving time that was seen in the original ILP when going from the small dataset to the large one.

8

Conclusion

There are multiple methods to find a solution to the question asked in Chapter 2. The optimal solution can be found with the ILP from Chapter 3. This is not the quickest way to get a solution, thus in Chapter 4, heuristic algorithms are given that give a less optimal solution in a faster time. These solutions had the same amount of hubs but not the same nodes were hubs. The cost difference for these algorithms is less than 2% for the given data sets. For random data sets, the solution of the greedy heuristic algorithm is expected to be around 5.13% more expensive than the solution of ILP. However, the heuristic algorithm is over 600 times quicker. So depending on the use case, one algorithm may be more suitable than the other.

The optimal way to place the hubs, NTMs and HTMs in the problem from Chapter 6 depends on the discount given by the HTMs and the NTMs. If the discount is too high, only one hub is placed. When the discount is low, there will not be a difference between the solution for the old problem and the new problem. With a discount of 15%, one fewer hub is placed and the cost gets lowered by 5.6% for the small data set.

Bibliography

[1] Greenberg, Harvey J., (1997) Klee–Minty Polytope Shows Exponential Time Complexity of Simplex Method the University of Colorado at Denver

A

Appendix: Datasets

A.1. Datasets

These datasets were used for the calculations made in the report, however, the calculations made are identical independent of the dataset. The datasets listed are only examples.

A.1.1. Small Datasets

	1	2	3	4	5	6	7	8	9	10
1	75	37	55	19	20	18	57	17	19	16
2	26	38	25	27	18	23	38	20	12	17
3	67	39	51	22	24	21	71	20	23	19
4	17	33	19	25	16	23	40	23	11	19
5	17	25	21	19	23	18	73	20	24	19
6	12	18	12	16	11	17	37	22	10	18
7	82	78	90	68	95	73	312	99	110	110
8	35	42	36	48	36	58	173	90	41	92
9	12	12	15	10	19	11	68	15	24	20
10	22	25	23	28	23	33	109	51	30	63

Table A.1: Package flows for the small data set. The columns represent the origin and the rows represent the destination.

	1	2	3	4	5	6	7	8	9	10
1	0	20	16	23	23	30	32	33	36	36
2	20	0	20	13	25	19	30	25	38	31
3	16	20	0	14	7	19	16	18	21	21
4	23	13	14	0	14	7	18	13	27	18
5	23	25	7	14	0	16	9	13	14	14
6	30	19	19	7	16	0	16	8	25	13
7	32	30	16	18	9	16	0	9	10	7
8	33	25	18	13	13	8	9	0	18	5
9	36	38	21	27	14	25	10	18	0	14
10	36	31	21	18	14	13	7	5	14	0

Table A.2: Cost of transporting one package for the small data set. The columns represent the origin and the rows represent the destination.

1	28767
2	28377
3	29774
4	24301
5	25853
6	20763
7	34166
8	33859
9	24718
10	33686

Table A.3: Hub costs for small data set. The left column represents that specific node and the right column represents the fixed cost of turning that node into a hub.

collection	3
transfer	1
distribution	2

Table A.4: Transportation multipliers for the small data set.

A.1.2. Large dataset

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	6	10	5	3	6	6	2	2	2	2	2	2	3	9	2
2	21	39	18	10	22	21	8	7	8	8	8	6	10	35	5
3	6	10	9	9	6	8	7	5	3	5	6	4	4	15	4
4	4	6	10	11	4	7	8	6	3	6	8	5	4	15	4
5	7	12	6	4	8	7	3	3	5	4	3	3	6	20	3
6	17	31	17	12	18	18	9	7	8	8	9	6	10	35	5
7	4	7	11	12	5	7	9	7	3	7	9	5	4	17	5
8	2	4	5	5	3	4	4	5	2	4	4	5	3	16	4
9	3	5	4	4	6	4	3	4	9	5	3	4	11	37	4
10	4	6	9	9	5	6	7	6	4	6	7	5	5	20	4
11	2	3	5	6	2	4	4	3	2	3	4	3	2	9	3
12	2	4	4	4	3	3	3	6	3	3	3	7	3	23	6
13	4	8	6	5	9	7	4	5	13	7	4	6	18	57	5
14	25	45	36	31	40	35	24	34	43	32	24	38	56	181	32
15	10	17	17	16	14	14	12	25	14	15	14	31	18	121	26

Table A.5: Package flows for the large data set. The columns represent the origin and the rows represent the destination.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	12	18	37	16	18	23	32	21	25	30	37	27	32	33
2	12	0	10	26	20	15	19	26	25	24	27	32	30	33	32
3	18	10	0	19	18	9	10	16	22	17	18	22	25	26	25
4	37	26	19	0	35	25	22	17	37	28	25	21	39	36	33
5	16	20	18	35	0	10	15	24	5	13	18	27	11	17	18
6	18	15	9	25	10	0	6	15	13	9	12	19	16	18	17
7	23	19	10	22	15	6	0	10	16	7	8	14	17	17	15
8	32	26	16	17	24	15	10	0	24	13	9	6	24	20	16
9	21	25	22	37	5	13	16	24	0	11	16	25	6	12	15
10	25	24	17	28	13	9	7	13	11	0	5	14	11	9	8
11	30	27	18	25	18	12	8	9	16	5	0	9	15	11	8
12	37	32	22	21	27	19	14	6	25	14	9	0	24	18	14
13	27	30	25	39	11	16	17	24	6	11	15	24	0	8	11
14	32	33	26	36	17	18	17	20	12	9	11	18	8	0	4
15	33	32	25	33	18	17	15	16	15	8	8	14	11	4	0

Table A.6: Cost of transporting one package for the large data set. The columns represent the origin and the rows represent the destination.

1	15237
2	13250
3	12428
4	15111
5	16651
6	16312
7	11673
8	14259
9	14846
10	11062
11	11834
12	12995
13	14883
14	11348
15	10013

Table A.7: Hub costs for large data set. The left column represents that specific node and the right column represents the fixed cost of turning that node into a hub.

collection	3
transfer	1
distribution	2

Table A.8: Transportation multipliers for the large data set.

B

Appendix: Python Implementation of Algorithms

The algorithms from Chapter 3, Chapter 4 and Chapter 7 have been implemented in Python. The link to the GitHub repository is https://github.com/oThommy/Modelling-B. For the initial problem, the exhaustive heuristic algorithm can be found in src/intuitive_algo_1.py and the greedy one in src/intuitive_algo_2.py. The ILP solver for the original problem has been implemented in three ways, namely using PuLP, Gurobi and CPLEX, which can be found in src/ilp_solver.py. The ILP solver for the expanded problem has been implemented in Gurobi, which can be found in src/problemexpansion.py.