Master Thesis

University of Twente.

Solving the trip based transport model using iterative optimization algorithms

Author:

Tim van Genderen

Company supervisor:
Luuk Brederode (DAT.Mobility)

University supervisors:
Alexander Skopalik (University of Twente.)
Matthias Walter (University of Twente.)

Graduation supervisor:
Marc Uetz (University of Twente.)

Chair:
Discrete Mathematics and Mathematical Programming (DMMP)
Applied Mathematics

November 2020

UNIVERSITY OF TWENTE.

Abstract

This thesis proposes a more robust method for the estimation of lognormal cost function parameters within the trip-based gravity model for transport models. The parameters are currently calibrated using empirical trip length distribution, but the proposed method determines the parameters by using the mathematical relation between the parameters of the trip-based gravity model and the dual variables of the original optimization problem of finding the trip distribution with maximal entropy. First, the current trip-based gravity model together with its derivation from the entropy optimization problem, solving procedure and the currently implemented calibration method is described. Afterwards, the solving procedures for solving NLP of the entropy optimization problem are discussed, together with an extension that creates a hybrid between the lognormal cost function and a discrete cost function. These solving procedures are validated and tested on a realistic transport model for the Dutch city of Almere and its results are compared to those of the trip-based gravity model.

Contents

1 Introduction

Transport planning models are made to forecast road-traffic in a simplified representation of the real-world. Usually they are developed to provide answers like: how does restructuring this junction effect congestion, what infrastructure has to be built in the next 20 years to minimize congestion and whether a certain measure will reduce pollution. Transport planning models play an important role to support governments in their decision making for questions as listed above.

In order to have an accurate transport model, one needs to predict people's movements correctly. The current situation in the world, a global pandemic due to the coronavirus, emphasizes this even more. The travelling behaviour before and after this pandemic is very different: people travel way less in general and due to possible restrictions on public transit, they are more inclined to use the car or bike. These are just a few examples to stress the importance of accurately representing people's behaviour when modelling their movements.

1.1 Four Steps of Traffic Modelling

The traditional four-step model is one of the most used models for forecasting and modelling transportation and traffic [10]. In this modelling approach a certain area is partitioned into different zones, together with an underlying road network containing information such as distance, speed and capacity of a road. It is important to note that movements are estimated only in the chosen area. Therefore, models usually consist of larger areas such as provinces or a country. Would only a city be modelled, then people are restricted to only travel within the city, which is unrealistic. Furthermore, socioeconomic data, e.g. population, employment, education provisions and number of households, for each zone is required to determine parameters for applying the model. Together with survey data it is used for parameter calibration, enabling the model to represent the current situation correctly. The steps of the traditional four-step model are:

1. **Trip generation:** In the first step the socioeconomic data is used in order to determine the number of trips leaving and entering a zone, defined as the production and attraction of the zone respectively.
2. **Trip distribution:** In this step the production and attraction values of each zone are matched, i.e. modelling the destination selection of the travelers, resulting in a origin-destination (OD) matrix representing the trip distribution. An entry in this matrix represents the number of trips from one zone to another and is usually made more specific by adding information such as the used mode of travel. The matrix is estimated using a cost function relating the willingness to travel with the cost of the trip as a guideline. The model is calibrated based on survey data and nowadays also on mobile phone data.
3. **Mode choice:** After obtaining the origin-destination matrix, the trip distribution is split up by determining which mode is used for each trip. This mode choice, also called modal split, is estimated using observed data of the modelled area.
4. **Route assignment:** The final step distributes the obtained trip distribution over the given road network, i.e. a route is assigned to each trip. Combining all routes, the traffic load can be computed and places of congestion in the network can be identified.

In the current implementation at DAT.Mobility, the trip distribution and mode choice are simultaneously integrated in the trip-based gravity model. Moreover, a feedback loop exists between these simultaneous steps and the route assignment, allowing the model input to be corrected according to the computed route assignment. After the route assignment step, congestion places are identified and travelling via these places takes longer than previously expected. By adjusting the costs, the new situation with congestion is represented and its trip distribution and mode choice can be computed. By iteratively doing this, the equilibrium between network supply and travel demand can be represented. A visualisation of the process is shown in Figure 2.

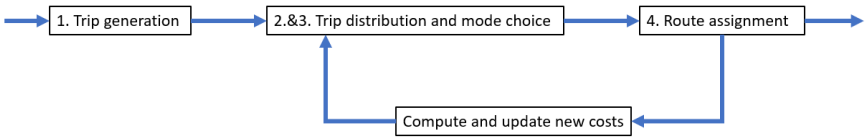


Figure 2: Overview of the four-step model

The original problem is to find the solution that maximizes the entropy, i.e. trip distribution that has the highest probability of occurrence. The trip-based gravity model does not use this formulation directly, but solves the feasibility problem that results from solving the entropy problem with Lagrange multipliers. This also introduces the so called gravity equation that describes the willingness to travel with respect to the cost. One main difference between the two problems is that the feasibility problem includes a gravity equation, whereas this behaviour was encapsulated in constraints in the entropy model. The most important values to estimate are the beta variables, and its corresponding constraints are the so-called budget constraints which limits the sum of the generalized costs of the trips in the model.

The parameters that are used in this gravity equation are estimated by using a calibration method. The current calibration procedure makes use of empirical trip length distributions and modal splits and finds the parameters that minimizes the difference between the modelled and observed trip length distribution.
However, rather than solving the feasibility problem with the gravity equation, one could also solve the general problem by solving the original entropy problem. It turns out that the parameters of the gravity equation have a one-to-one relationship with the corresponding constraints of the entropy problem by being their dual multipliers. Due to the mathematical relation between the parameters and the entropy problem constraints, solving the entropy problem directly should yield more robust method for estimating parameters than the calibration method. Of course, all of this is under the assumption that the correct input is used for both methods. However, this is could be a possible obstacle for the entropy problem. Replacing the gravity equation by its corresponding constraints also introduces new values that need to be estimated, namely the right-hand side of these constraints. These values are currently unknown since they were not used in the trip-based gravity model and not kept track of in the empirical data.
The entropy maximization problem is an example of a nonlinear program (NLP) with equality constraints. In this thesis we look two of the most used techniques for solving these problems, the Frank-Wolfe algorithm (FW) and Sequential Linear Programming (SQP).
One important note is that this thesis is about an alternative way to determine the parameters of the gravity equation and not an alternative to the trip-based gravity model. For application purposes, one is better off using the trip-based gravity model for computing the optimal trip distribution given the parameters due to its efficient solving procedure.

1.2 Goal and Outline of Report

The main goal of this thesis is to research to what extent an iterative optimization algorithm approach (FW and SQP) can be used to translate a given travel-time budget constraint into deterrence function parameters in the multimodal trip-based gravity model. Besides that, the deterrence function parameters somehow have to be retrieved from the solution of the entropy problem, so we need to establish the mathematical relation between the two formulations. One improvement upon the deterrence function can be made by introducing distance bins to the constraints, since this would result in parameters per distance bin. This was not possible when using the current calibration method, but could lend itself for this extension and the question is whether that is correct. Lastly, the new procedures have to made into a working prototype. However, since the the entropy problem is an NLP which is mathematically seen harder to solve than the feasibility problem with the gravity equation, the question remains whether the working prototype is scalable enough to be used as an alternative to the current calibration method. For this prototype the goal is to compute the parameters for the full Almere model, including all purposes, over night.

In Section 2 the currently implemented version of the trip-based model is discussed, including the entropy formulation in Section 2.2 and the calibration method in Section 2.6. Afterwards, the mathematical side of the solving procedures FW and SQP together with a proof of their relationship to the deterrence function parameters are discussed in Section 3. Section 4 covers the practical side of applying FW and SQP in the context of solving the entropy optimization problem, together with how the deterrence function can be extended to support distance bins. An implementation of these methods is tested in Section 5 on the Almere model and their performance is analyzed by analyzing the results of synthetic data. Lastly, Section 6 discusses these results together with the conclusions and recommendations for future work.

2 Trip-based Gravity Model

This section describes the trip-based gravity model together with the calibration method to obtain the corresponding required parameters, as currently used at DAT.Mobility. Firstly, we give the formulation of the doubly constrained gravity model in Section 2.1 together with some model specific choices that have to be made. Section 2.2 gives the derivation of this model, which originates from the question of finding the trip distribution with maximal entropy. Later on, this model has evolved into the currently implemented triply constrained gravity model at DAT.Mobility, as stated in Section 2.4, with its solving procedure in Section 2.5. Finally, Section 2.6 describes the currently used calibration method to obtain the model specific parameters.

2.1 Doubly Constrained Gravity Model

2.1.1 Trip-End Constraints

First of all, in order to have a correct origin-destination matrix (OD-matrix) $\mathbf{T} = \sum_m T^m$, we must have that for each zone of origin i, the number of trips leaving that zone must be equal to the observed production P_i of that zone. Similarly, for each zone of destination j, the number of trips entering that zone must be equal to the observed attraction A_j of that zone. These constraints are called the trip-end constraints:

$$\sum_{j\in\mathcal{J},m\in\mathcal{M}} t_{ijm} = P_i \qquad \forall i \in \mathcal{I}, \tag{2.1}$$

$$\sum_{i\in\mathcal{I},m\in\mathcal{M}} t_{ijm} = A_j \qquad \forall j \in \mathcal{J}, \tag{2.2}$$

where t_{ijm} is the (i,j)-th entry of the OD-matrix T^m.

From the trip-end constraints (2.1) and (2.2) it follows that:

$$\sum_{i\in\mathcal{I}} P_i = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J},m\in\mathcal{M}} t_{ijm} = T = \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I},m\in\mathcal{M}} t_{ijm} = \sum_{j\in\mathcal{J}} A_j, \tag{2.3}$$

where T is the total number of modelled trips.
However, the productions and attractions from the trip generation (the first step of the four-step model) can be inconsistent. This can be resolved to satisfy (2.3) by balancing them. This can be done in two ways: scale the sum of the productions such that it matches the sum of the attractions, or the other way around. If one assumes that the attractions are more accurate than the productions, the new productions are:

$$P_i' = \left(\frac{\sum_j A_j}{\sum_i P_i}\right) \cdot P_i \qquad \forall i \in \mathcal{I}. \tag{2.4}$$

On the other hand, if the productions are assumud to be more accurate, then the new attractions are:

$$A_j' = \left(\frac{\sum_i P_i}{\sum_j A_j}\right) \cdot A_j \qquad \forall j \in \mathcal{J}. \tag{2.5}$$

2.1.2 Generalized Costs

Travelling from zone i to zone j with mode m is not free and has some costs attached to it, captured by the generalized costs c_{ijm}. These generalized costs can depend on a lot of factors, such as distance, travel time and possible fuel costs [10].

2.1.3 Gravity Equation

The gravity equation is used to determine the values in the OD matrix and is derived in Section 2.2. It is given by:

$$t_{ijm} = p_i P_i a_j A_j F^m(c_{ijm}) \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, m \in \mathcal{M}, \tag{2.6}$$

where $p_i \geq 0$ and $a_j \geq 0$ are the balancing factors for the production an attraction respectively, and $F^m(c_{ijm})$ the chosen cost/deterrence function. In the solution procedure (see Section 2.5) both the rows and columns are scaled, the balancing factors p_i and a_j keep track of these respectively. The reasoning behind the naming of the gravity equation is given in Section 2.1.4.
Intuitively, if two zones are close to each other one would expect that there are more people travelling between them than between two zones that are further apart. The gravity equation can do exactly that if the deterrence function is chosen in a certain way. Multiple used deterrence functions are discussed in the next section.

The gravity equation can be written, as given below, in a shorter way by setting $O_i = p_i \cdot P_i$ and $D_j = a_j \cdot A_j$ and is used in the remainder of the report.

$$t_{ijm} = O_i D_j F^m(c_{ijm}) \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, m \in \mathcal{M}. \tag{2.7}$$

2.1.4 Deterrence Function

The deterrence function in the gravity equation is crucial and its parameters are different for each mode in order to determine the modal split. Usually, one of the following deterrence functions is chosen. An example of all functions is given in Figure 3:

- **Exponential**: $F^m(c_{ijm}) = e^{\beta_m c_{ijm}}, \quad \beta_m < 0$
  The most used one, involving one parameter that takes care of the steepness of the distribution.
- **Lognormal**: $F^m(c_{ijm}) = \alpha_m e^{\beta_m ln^2(c_{ijm}+1)}, \quad \beta_m < 0,\ \alpha_m > 0$
  The distribution used for the gravity model at DAT.Mobility. Besides the steepness parameter β_m, it also involves a parameter α_m, influencing the modal split.
- **Top-lognormal**: $F^m(c_{ijm}) = \alpha_m e^{\beta_m ln^2(\frac{c_{ijm}}{\gamma_m})}, \quad \beta_m < 0,\ \alpha_m, \gamma_m > 0$
  An extension of the lognormal distribution with a third parameter γ_m, causing the function to first increase before decreasing like the lognormal function.
- **Discrete**: $F^m(c_{ijm}) = F^m_k \quad \text{if } c_{ijm} \in I^m_k$
  A complete different one than the distributions above, it gives the user a lot of freedom. The user can divide the domain of c_{ijm} values by defining breakpoints c^m_0 up to $c^m_{n_m}$. This results in the intervals $I^m_1 = (c^m_0, c^m_1]$ up to $I^m_{n_m} = (c^m_{n_m-1}, c^m_{n_m}]$ in which the user can specify the corresponding deterrence functions values F^m_1 up to $F^m_{n_m}$.

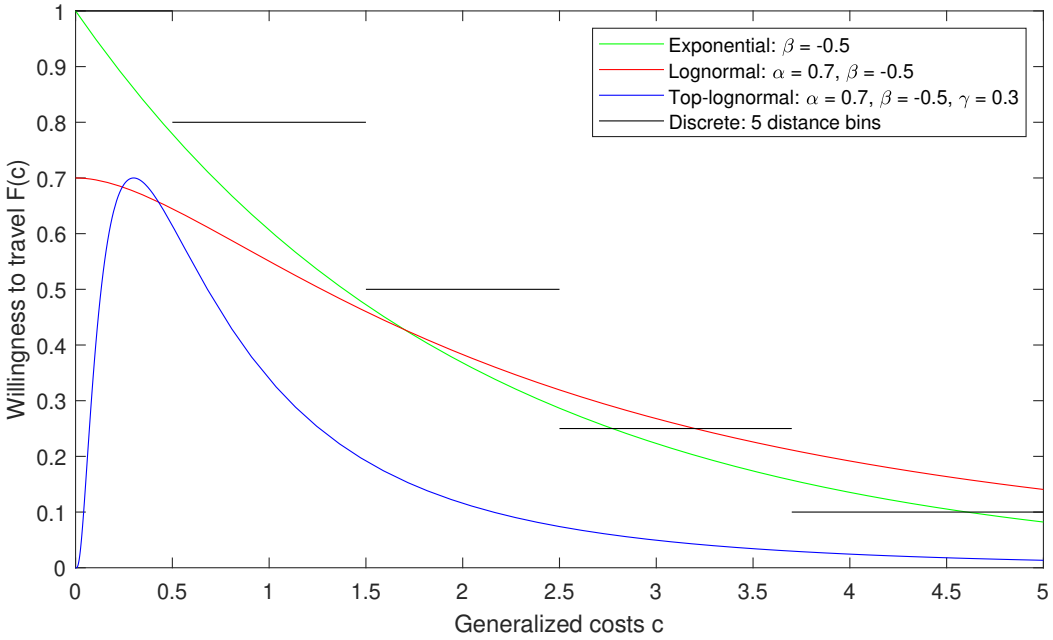


Figure 3: Examples of the different deterrence functions

One other deterrence function that is not used in the context of travel demand models, but does explain why this model is called the gravity model, is $F^m(c_{ijm}) = \frac{1}{c_{ijm}^2}$.
When substituted in the gravity equation (2.6), this yields:

$$\begin{aligned} t_{ijm} &= p_i P_i a_j A_j \frac{1}{c_{ijm}^2}, \\ &= (p_i a_j) \frac{P_i \cdot A_j}{c_{ijm}^2}, \end{aligned}$$

for all $i \in \mathcal{I}, j \in \mathcal{J}$ and $m \in \mathcal{M}$. This is very similar to Newton's Law:

$$F_1 = F_2 = G \frac{m_1 \cdot m_2}{r^2}.$$

One of the reasons why this distribution function is not used, is due to the fact that Newton's Law states $F_1 = F_2$ which would translate to $t_{ijm} = t_{jim}$ in the trip model. However, this does not have to be correct since we can have that the costs are not bidirectional, i.e. c_{ijm} does not have to be equal to c_{jim}.

2.1.5 Solving the doubly Constrained Gravity Model

The doubly constrained gravity model discussed so far can be formulated as below. The equations respectively follow from (2.7), (2.1), (2.2) and the definition of balancing factors.

$$t_{ijmu} = O_i D_j F^m(c_{ijm}) \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, m \in \mathcal{M}, \tag{2.8}$$

$$\sum_{j \in \mathcal{J}, m \in \mathcal{M}} t_{ijm} = P_i \qquad \forall i \in \mathcal{I}, \tag{2.9}$$

$$\sum_{i \in \mathcal{I}, m \in \mathcal{M}} t_{ijm} = A_j \qquad \forall j \in \mathcal{J}, \tag{2.10}$$

$$O_i, D_j \geq 0 \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}. \tag{2.11}$$

It can be solved by using the Furness method, also known as iterative proportional fitting (IPF) [2]. Note that this method is for solving the gravity equation only and FW or SQP would not replace it. The Furness method iteratively scales the modelled productions and attractions by scaling the rows and columns respectively. This is done until the difference between the observed and modelled productions and attractions are negligible. It does not matter whether the columns or rows are scaled first, but it is convenient to know that the last scaled ones match the constraint exactly. So if the columns are scaled last, we have that the modelled attractions are the same as the observed attractions. In (2.4) and (2.5) either the productions or attractions are balanced, since one of those was believed to be the most accurate. Therefore, it can be chosen in the Furness method to scale these at last, thus modelling these values exactly. The Furness method for solving the doubly constrained gravity model, is listed in Algorithm 1.

Algorithm 1 Solution algorithm for the doubly constrained trip-based gravity model

Initialization of O, D, T
forall $i \in \mathcal{I}$ **do**
  $O_i \leftarrow P_i$
end
forall $j \in \mathcal{J}$ **do**
  $D_j \leftarrow A_j$
end
forall $m \in \mathcal{M}$ **do**
  forall $(i, j) \in \mathcal{I} \times \mathcal{J}$ **do**
    $t_{ijm} \leftarrow O_i \cdot D_j \cdot F^m(c_{ijm})$
  end
end
while *not converged* **do**
  Row Scaling:
  forall $i \in \mathcal{I}$ **do**
    if $\sum_{j \in \mathcal{J}, m \in \mathcal{M}} t_{ijm} > 0$ **then** $f \leftarrow \left(\frac{P_i}{\sum_{j \in \mathcal{J}, m \in \mathcal{M}} t_{ijm}} \right)$ **else** $f \leftarrow 0$
    forall $j \in \mathcal{J}$ **do**
      forall $m \in \mathcal{M}$ **do**
        $t_{ijm} \leftarrow f \cdot t_{ijm}$
      end
    end
  end
  Column Scaling:
  forall $j \in \mathcal{J}$ **do**
    if $\sum_{i \in \mathcal{I}, m \in \mathcal{M}} t_{ijm} > 0$ **then** $f \leftarrow \left(\frac{A_j}{\sum_{i \in \mathcal{I}, m \in \mathcal{M}} t_{ijm}} \right)$ **else** $f \leftarrow 0$
    forall $i \in \mathcal{I}$ **do**
      forall $m \in \mathcal{M}$ **do**
        $t_{ijm} \leftarrow f \cdot t_{ijm}$
      end
    end
  end
end

2.2 Derivation of Trip Model

The described model so far has yet no mathematical foundation why it outputs the optimal trip distribution. This section proves that the doubly constrained gravity model maximizes the entropy, as first shown by Wilson [16]. The entropy, i.e. the probability of occurrence of the trip distribution, is defined as:

$$\text{entropy}(\mathbf{T}) = \frac{T!}{\prod\limits_{i\in\mathcal{I},j\in\mathcal{J},m\in\mathcal{M}} t_{ijm}!}. \tag{2.12}$$

Since the natural logarithm as a monotone increasing function, this is equivalent to maximizing:

$$\begin{aligned} ln(e(\mathbf{T})) &= ln\left(\frac{T!}{\prod\limits_{i\in\mathcal{I},j\in\mathcal{J},m\in\mathcal{M}} t_{ijm}!}\right) = ln(T!) - ln\left(\prod_{i\in\mathcal{I},j\in\mathcal{J},m\in\mathcal{M}} t_{ijm}!\right) = ln(T!) - \sum_{i\in\mathcal{I},j\in\mathcal{J},m\in\mathcal{M}} ln(t_{ijm}!) \\ &= ln(T!) - \sum_{i\in\mathcal{I},j\in\mathcal{J},m\in\mathcal{M}} (t_{ijm}\cdot ln(t_{ijm}) - t_{ijm}) \end{aligned}$$

where we use the Stirling's approximation $ln(n!) = n\cdot ln(n) - n$ in the last step. Since $ln(T!)$ is a constant, this can be simplified even more to maximizing

$$-\sum_{i\in\mathcal{I},j\in\mathcal{J},m\in\mathcal{M}} t_{ijm}\cdot ln(t_{ijm}) - t_{ijm}. \tag{2.13}$$

The proof given in the remainder of this section uses the derivation of Willekens [15] with the additional constraints:

$$\sum_{i\in\mathcal{I},j\in\mathcal{J}} t_{ijm}\cdot c_{ijm} = C_m \qquad \forall m\in\mathcal{M}, \tag{2.14}$$

where C_m is the total amount of generalized costs spent on trips made by mode m. Estimating the values of C_m is important, since they immediately relate to the values of beta (see Section 3.3 or 3.4 for more details). When applying the model these values should follow from the survey data that is being used, but they are currently unknown due to the fact that these values are not required for the trip-based gravity model as discussed in the Section 2.1. How this problem is solved in our case when applying FW or SQP is discussed in Section 5.

The optimization problem for maximizing the entropy is now formulated as:

$$\max_{\mathbf{T}} -\sum_{i\in\mathcal{I},j\in\mathcal{J},m\in\mathcal{M}} t_{ijm}\cdot ln(t_{ijm}) - t_{ijm} \tag{2.15}$$

$$\text{subject to:} \sum_{j\in\mathcal{J},m\in\mathcal{M}} t_{ijm} = P_i \qquad \forall i\in\mathcal{I} \tag{2.16}$$

$$\sum_{i\in\mathcal{I},m\in\mathcal{M}} t_{ijm} = A_j \qquad \forall j\in\mathcal{J} \tag{2.17}$$

$$\sum_{i\in\mathcal{I},j\in\mathcal{J}} t_{ijm}c_{ijm} = C_m \qquad \forall m\in\mathcal{M} \tag{2.18}$$

$$t_{ijm} \geq 0 \qquad \forall i\in\mathcal{I}, j\in\mathcal{J}, m\in\mathcal{M} \tag{2.19}$$

This optimization problem can be solved by using Lagrange multipliers. By using λ_i, μ_u and β_m as the Lagrange multipliers for the constraints, the Lagrangian becomes:

$$\mathcal{L}(\mathbf{T}, \lambda, \mu, \beta) = -\sum_{i\in\mathcal{I}, j\in\mathcal{J}, m\in\mathcal{M}} (t_{ijm} \cdot ln(t_{ijm}) - t_{ijm}) + \sum_{i\in\mathcal{I}} \lambda_i \left(P_i - \sum_{j\in\mathcal{J}, m\in\mathcal{M}} t_{ijm} \right) + \sum_{j\in\mathcal{J}} \mu_j \left(A_j - \sum_{i\in\mathcal{I}, m\in\mathcal{M}} t_{ijm} \right) + \sum_{m\in\mathcal{M}} \beta_m \left(C_m - \sum_{i\in\mathcal{I}, j\in\mathcal{J}} t_{ijm} c_{ijm} \right).$$

Computing the partial derivatives of the Lagrangian with respect to its variables gives:

$$\frac{\partial \mathcal{L}}{\partial t_{ijm}} = -ln(t_{ijm}) - \lambda_i - \mu_j - \beta_m c_{ijm} \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, m \in \mathcal{M}, \tag{2.20}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = P_i - \sum_{j\in\mathcal{J}, m\in\mathcal{M}} t_{ijm} \qquad \forall i \in \mathcal{I}, \tag{2.21}$$

$$\frac{\partial \mathcal{L}}{\partial \mu_j} = A_j - \sum_{i\in\mathcal{I}, m\in\mathcal{M}} t_{ijm} \qquad \forall j \in \mathcal{J}, \tag{2.22}$$

$$\frac{\partial \mathcal{L}}{\partial \beta_m} = C_m - \sum_{i\in\mathcal{I}, j\in\mathcal{J}} t_{ijm} c_{ijm} \qquad \forall m \in \mathcal{M}. \tag{2.23}$$

Note that (2.21), (2.22) and (2.23) must be equal to zero due to the equality constraints (2.16), (2.17) and (2.18), respectively. Moreover, in order to find a local optimum, we also require a tangency condition leading to (2.20) being equal to zero as well. This yields:

$$t_{ijm} = e^{-\lambda_i - \mu_j - \beta_m c_{ijm}} = e^{-\lambda_i} \cdot e^{-\mu_j} \cdot e^{-\beta_m c_{ijm}} \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, m \in \mathcal{M}. \tag{2.24}$$

By substituting $O_i = e^{-\lambda_i}$ and $D_j = e^{-\mu_j}$ in (2.24) we obtain the gravity equation with an exponential deterrence function:

$$t_{ijm} = O_i D_j e^{-\beta_m c_{ijm}} \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, m \in \mathcal{M}. \tag{2.25}$$

Thus we have that for (2.25) we obtain a local optimum. Actually, since the objective function (2.15) is a concave function, we have that the solution is also a global optimum.

In other words, the gravity equation with an exponential deterrence function obtains a global maximum, thus maximizing the entropy.
Furthermore, if instead of the additional constraint in (2.14) other constraints were chosen, this would result in a different deterrence function [10]. For example, when adding the constraints:

$$\sum_{i\in\mathcal{I}, j\in\mathcal{J}} t_{ijm} = M_m, \quad \forall m \in \mathcal{M} \qquad \text{and}$$

$$\sum_{i\in\mathcal{I}, j\in\mathcal{J}} t_{ijm} ln^2(c_{ijm} + 1) = C_m, \quad \forall m \in \mathcal{M},$$

this would result in the lognormal deterrence function, as used in the trip-based gravity models at DAT.Mobility.

An important side-note is that, as stated earlier in this section, the values of C_m are hard to come by. However, the values of M_m are known, and in fact used in the triply constrained gravity model in Section 2.4 in the form of the modal split parameters α.

2.2.1 Solution properties of biproportional fitting procedure

2.2.2 Uniqueness

First of all, feasible balancing factors are not unique. Suppose that (O, D) is a solution to the doubly constrained gravity model (2.8) - (2.11), then we clearly have for any $\lambda > 0$ that $(\lambda \cdot O, \frac{1}{\lambda} \cdot D$ is a valid solution as well. However, this is the only way to obtain other feasible balancing factors. In other words, a feasible set of balancing factors is unique up to this constant λ, which is proven in Theorem 1.

Theorem 1. *Let $M \in \mathbb{R}^{m \times n}_{\geq 0}$ be a matrix without zero rows or columns. Suppose that we have production and attraction values $P \in \mathbb{R}^{m}_{>0}$ and $A \in \mathbb{R}^{n}_{>0}$ and the corresponding set of balancing factors $O, \tilde{O} \in \mathbb{R}^{m \times m}_{>0}$ and $D, \tilde{D} \in \mathbb{R}^{n \times n}_{>0}$, i.e. both balanced matrices $T = OMD$ and $\tilde{T} = \tilde{O}M\tilde{D}$ satisfy the productions P and attractions A constraints. Then the following two statements are true:*

1. $T = \tilde{T}$
2. $\exists \lambda > 0$ *s.t.* $\tilde{O} = \frac{1}{\lambda} \cdot O$ *and* $\tilde{D} = \lambda \cdot D$

Proof. Statement 1 directly follows from Theorem 4 as stated in Rothblum [13].
Now we can use statement 1 to prove statement 2. Since $T = \tilde{T}$ holds, or equivalently $OMD = \tilde{O}M\tilde{D}$, we must have that $diag(O)^T \cdot diag(D) = diag(\tilde{O})^T \cdot diag(\tilde{D})$. Considering row i, we must have $O_i \cdot D = \tilde{O}_i \cdot \tilde{D}$ and thus we can choose $\lambda = \frac{O_i}{\tilde{O}_i} > 0$ to have $\lambda \cdot D = \tilde{D}$. Similarly, for column j we have $D_j \cdot O = \tilde{D}_j \cdot \tilde{O} = \lambda \cdot D_j \cdot \tilde{O}$ and thus $O = \lambda \cdot \tilde{O}$, or equivalently $\tilde{O} = \frac{1}{\lambda} \cdot O$. □

2.2.3 Existence of Solution and Convergence

Before applying Theorem 1, one first requires a solution. However, Pukelsheim [12] provides an example of a non-feasible solution in which the biproportional fitting procedure results in an oscillating behaviour between two matrices. That is, after column scaling we obtain matrix A and performing row scaling on A results in matrix B, performing column scaling on this matrix B results in A again, and so on.

Pukelsheim also states the requirement for convergence of the biproportional fitting procedure and is given in Theorem 2. It uses the L_1-error of an matrix, which is can be calculated for an OD-matrix $\mathbf{T}^k$ in iteration k by:

$$f(k) = \frac{1}{2} \sum_{i \in \mathcal{I}} \left| \sum_{j \in \mathcal{J}} t^k_{ij} - P_i \right| + \frac{1}{2} \sum_{j \in \mathcal{J}} \left| \sum_{i \in \mathcal{I}} t^k_{ij} - A_j \right|.$$

Moreover, for matrix $\mathbf{T}$, row i and column j are said to connected if $t_{ij} >$. Consequently $\mathcal{J}(\mathcal{I})$ denotes the subset of columns $\mathcal{J}$ that are connected to the rows $\mathcal{I}$.

Theorem 2. *Given an initial solution $\boldsymbol{T}^0 \in \mathbb{R}^{m \times n}_{\geq 0}$ with no zero rows or columns, production values $P \in \mathbb{R}^{m}_{>0}$ and attraction values $A \in \mathbb{R}^{n}_{>0}$, the limit of the L_1-error during the biproportional fitting procedure is given by*

$$\lim_{k \to \infty} f(k) = \max_{I \subset \{1, \ldots, m\}} \left(\sum_{i \in \mathcal{I}} P_i - \sum_{j \in \mathcal{J}(\mathcal{I})} A_j \right)$$

and the biproportional fitting procedure converges if and only if this limit is zero.

2.3 Purposes and User Classes

In practice, there is more data available of a trip besides the origin zone i, destination zone j, used mode m and its cost c_{ijm}. The most important one is the purpose of the trip. For example, on a normal day most employed people will travel to their work in the morning and returning in the afternoon. Therefore, we can label these trips with purpose "Home → Work" and "Work → Home" respectively. Usually, a wide variety of

purposes are in the model. For example, an average model at DAT.Mobility uses the home related purposes of Work, Business, Education, Stores, Other and home unrelated purposes of "Business → Business".
Since the purpose is available per trip, a gravity model can be run per purpose. This is meaningful to do due to the fact that the parameters of the deterrence functions are usually different per purpose. For example, people are more willing to travel an hour to work than they would for going to the grocery store. Running the gravity model per purpose means an extra index for the OD matrix. For now, it is not introduced in the formulation of the gravity model, but keep in mind that these purposes exist and that the OD matrices are also made per purpose.

Besides purposes, there is one other significant aspect of a trip: the user class. At DAT.Mobility the user class is either "car owner (co)" or "non-car owner (nco)". One of the reasons to distinct between user classes is the same as for the purposes: the parameters of the deterrence functions are usually different per user class. For example, if one has to travel 150km a car owner would just take the car, but a non-car owner does not have this luxury and is more likely to go by e.g. public transit. Note that a non-car owner still can travel by car, but in that case they would be a passenger.
Another reason for introducing user classes is that the model can be made more accurate since at the home side of the trip, there is data available concerning the number of car owners. This means that for a model with purpose "Home → ..." we can split the productions P_i to P_{iu}, similarly a model with purpose "... → Home" the attractions A_j can be split up to A_{ju}.

2.4 Triply Constrained Gravity Model

An extension of the doubly constrained gravity model can be made by adding a modal split constraint. From the trip-end data one can extract the total number of trips that are made by user class u and mode m, denoted as the modal split $\widehat{MS}_{mu}$. Usually the modal split is used for determining the parameters of the deterrence functions, but Brethouwer [3] showed that the modal split constraints can be incorporated directly into the gravity model.
Together with Section 2.3 this results in the following model for "Home → ...":

$$t_{ijmu} = O_{iu} D_j F^{mu}(c_{ijmu}) \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, m \in \mathcal{M}, u \in \mathcal{U}, \tag{2.26}$$

$$\sum_{j \in \mathcal{J}, m \in \mathcal{M}} t_{ijmu} = P_{iu} \qquad \forall i \in \mathcal{I}, u \in \mathcal{U}, \tag{2.27}$$

$$\sum_{i \in \mathcal{I}, m \in \mathcal{M}, u \in \mathcal{U}} t_{ijmu} = A_j \qquad \forall j \in \mathcal{J}, \tag{2.28}$$

$$\sum_{i \in \mathcal{I}, j \in \mathcal{J}} t_{ijmu} = \widehat{MS}_{mu} \qquad \forall m \in \mathcal{M}, u \in \mathcal{U}, \tag{2.29}$$

$$t_{ijmu} \geq 0 \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, m \in \mathcal{M}, u \in \mathcal{U} \tag{2.30}$$

Similarly, the formulation for the model "... → Home" is:

$$t_{ijmu} = O_i D_{ju} F^{mu}(c_{ijmu}) \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, m \in \mathcal{M}, u \in \mathcal{U}, \tag{2.31}$$

$$\sum_{j \in \mathcal{J}, m \in \mathcal{M}, u \in \mathcal{U}} t_{ijmu} = P_i \qquad \forall i \in \mathcal{I}, \tag{2.32}$$

$$\sum_{i \in \mathcal{I}, m \in \mathcal{M}} t_{ijmu} = A_{ju} \qquad \forall j \in \mathcal{J}, u \in \mathcal{U}, \tag{2.33}$$

$$\sum_{i \in \mathcal{I}, j \in \mathcal{J}} t_{ijmu} = \widehat{MS}_{mu} \qquad \forall m \in \mathcal{M}, u \in \mathcal{U}, \tag{2.34}$$

$$t_{ijmu} \geq 0 \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, m \in \mathcal{M}, u \in \mathcal{U} \tag{2.35}$$

Note that a purpose such as "Business → Business" has user class data at both trip ends, meaning that this can be modelled by making separate models per user class, not having to split up either the productions or attractions.

2.5 Solving the triply Constrained Gravity Model

The triply constrained model as stated in Section 2.4 can be solved by running the algorithm 2. This algorithm can be used for "Home → ..." purposes, the other way around is very similar and is not be listed.

Algorithm 2 Solution algorithm for the triply constrained trip-based gravity model

Initialization of O, D, T
forall $i \in \mathcal{I}$ **do**
  $O_{iu} \leftarrow P_{iu}$
end
forall $j \in \mathcal{J}$ **do**
  $D_j \leftarrow A_j$
end
forall $(m,u) \in \mathcal{M} \times \mathcal{U}$ **do**
  $\alpha_{mu} \leftarrow \widehat{MS}_{mu}$
  forall $(i,j) \in \mathcal{I} \times \mathcal{J}$ **do**
    $t_{ijmu} \leftarrow O_{iu} \cdot D_j \cdot F^{mu}(c_{ijm})$
  end
end
while *not converged* **do**
  Column Scaling:
  forall $j \in \mathcal{J}$ **do**
    if $\sum_{i\in\mathcal{I},m\in\mathcal{M},u\in\mathcal{U}} t_{ijmu} > 0$ **then** $f \leftarrow \left(\frac{A_j}{\sum_{i\in\mathcal{I},m\in\mathcal{M},u\in\mathcal{U}} t_{ijmu}} \right)$ **else** $f \leftarrow 0$
    forall $i \in \mathcal{I}$ **do**
      forall $m \in \mathcal{M}$ **do**
        $t_{ijmu} \leftarrow f \cdot t_{ijmu}$
      end
    end
  end
  Row Scaling:
  forall $u \in \mathcal{U}$ **do**
    forall $i \in \mathcal{I}$ **do**
      if $\sum_{j\in\mathcal{J},m\in\mathcal{M}} t_{ijmu} > 0$ **then** $f \leftarrow \left(\frac{P_{iu}}{\sum_{j\in\mathcal{J},m\in\mathcal{M}} t_{ijmu}} \right)$ **else** $f \leftarrow 0$
      forall $j \in \mathcal{J}$ **do**
        forall $m \in \mathcal{M}$ **do**
          $t_{ijmu} \leftarrow f \cdot t_{ijmu}$
        end
      end
    end
  end
  Modal Split Scaling:
  forall $(m,u) \in \mathcal{M} \times \mathcal{U}$ **do**
    if $\sum_{i\in\mathcal{I},j\in\mathcal{J}} t_{ijmu} > 0$ **then** $f \leftarrow \left(\frac{\widehat{MS}_{mu}}{\sum_{i\in\mathcal{I},j\in\mathcal{J}} t_{ijmu}} \right)$ **else** $f \leftarrow 0$
    $\alpha_{mu} \leftarrow f \cdot \alpha_{mu}$
    forall $(i,j) \in \mathcal{I} \times \mathcal{J}$ **do**
      $t_{ijmu} \leftarrow f \cdot t_{ijmu}$
    end
  end
end

2.6 Calibration Method

The current calibration method that is being used at DAT.Mobility is developed by Pots [11] and formulates the triply constrained gravity model for a certain purpose as a bi-level optimization problem. Note that since we use the triply constrained gravity model there is only the need to find the beta parameters for the model.
In order to find the beta parameters that describe the movements of people as best as possible, we want to find the ones that represent the empirical trip length distributions. Let us denote an empirical trip length distribution for a mode-userclass pair (m, u) by $\hat{d}_{muk}$ and its corresponding relative trip length distribution $\hat{d}^{rel}_{muk}$ in percentages, computed by:

$$\hat{d}^{rel}_{muk} = \left(\frac{\hat{d}_{muk}}{\sum_{k' \in \mathcal{K}} \hat{d}_{muk'}} \right) \cdot 100(\%).$$

Similarly, we denote the modelled trip length distribution for a mode-userclass pair by d_{muk} and the modelled relative trip length distribution d^{rel}_{muk} by:

$$d^{rel}_{muk} = \left(\frac{d_{muk}}{\sum_{k' \in \mathcal{K}} d_{muk'}} \right) \cdot 100(\%).$$

Since the values of the empirical and modelled trip length distributions may vary depending on the number of trips in the model, we are looking to match the relative version of them as closely as possible. Therefore, the objective function for the calibration method is the sum of the squared differences:

$$F(\boldsymbol{\beta}) = \sum_{m \in \mathcal{M}, u \in \mathcal{U}, k \in \mathcal{K}} \left(d^{rel}_{muk} - \hat{d}^{rel}_{muk} \right)^2,$$

where $\boldsymbol{\beta}$ denotes the vector of all parameters β_{mu}.

The bi-level optimization problem consists of solving the gravity model as the inner problem and selecting the optimal $\boldsymbol{\beta}$ as the outer optimization problem. Mathematically speaking, the bi-level optimization problem for a certain purpose is formulated as:

$$\begin{aligned} \min_{\boldsymbol{\beta}<0, (\boldsymbol{O},\boldsymbol{D},\boldsymbol{\alpha}) \geq \mathbf{0}} \quad & F(\boldsymbol{\beta},\boldsymbol{O},\boldsymbol{D},\boldsymbol{\alpha}) \\ \text{s.t.} \quad & (\boldsymbol{O},\boldsymbol{D},\boldsymbol{\alpha}) \in \underset{(\tilde{\boldsymbol{O}},\tilde{\boldsymbol{D}},\tilde{\boldsymbol{\alpha}}) \geq \mathbf{0}}{\arg\max} \{ e(\mathbf{T}) \mid \mathbf{T} = \mathbf{T}(\boldsymbol{\beta},\tilde{\boldsymbol{O}},\tilde{\boldsymbol{D}},\tilde{\boldsymbol{\alpha}}) \text{ satisfies constraints relevant to purpose} \} \end{aligned} \tag{2.36}$$

where $(\boldsymbol{O},\boldsymbol{D},\boldsymbol{\alpha})$ denotes the set of balancing factors that relate to the purpose.

If we assume that the IPF procedure converges, the balancing factors are essentially only dependent on of $\boldsymbol{\beta}$, i.e. we can write $\boldsymbol{O} = \boldsymbol{O}(\boldsymbol{\beta})$, $\boldsymbol{D} = \boldsymbol{D}(\boldsymbol{\beta})$ and $\boldsymbol{\alpha} = \boldsymbol{\alpha}(\boldsymbol{\beta})$ and therefore $F(\boldsymbol{\beta}(\boldsymbol{\beta}),\boldsymbol{O}(\boldsymbol{\beta}),\boldsymbol{D}(\boldsymbol{\beta}),\boldsymbol{\alpha}(\boldsymbol{\beta})) = F(\boldsymbol{\beta})$. The following flowchart illustrates what the bi-level optimization problem looks like and how the objective value is only dependent on $\boldsymbol{\beta}$.

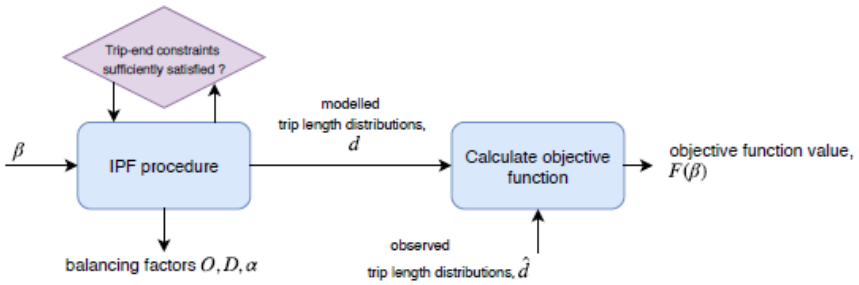


Figure 4: Flowchart of the bi-level optimization problem for a certain purpose [11].

Regarding the solving procedure of the bi-level optimization problem, the inner problem consists op solving the triply constrained gravity model and can be done by using Algorithm 2. For the solving procedure of the outer problem we refer to Pots [11] in which different approaches are discussed that are out of scope for this thesis.

3 Mathematical Background

This section covers the mathematics behind solving the entropy formulation and proving the added value of these procedures for the parameter estimation of the gravity model. First of all, Section 3.1 gives an introduction to nonlinear programming and classifies the entropy optimization problem. Afterwards, Section 3.1.1 and 3.1.2 state two possible solving procedures for the entropy optimization problem, the Frank-Wolfe and Sequential Quadratic Programming algorithm respectively. The next sections explain why these methods are so useful in context of the entropy problem. First, Section 3.2 introduces the mathematical concepts that are used to prove the mathematical relation between the parameters of the gravity model and the entropy formulation.

3.1 Nonlinear Programming

The entropy formulation (2.15)-(2.18) is an example of a nonlinear program (NLP). An NLP is an optimization problem where either, or both, the objective function or at least one constraint is nonlinear. Generally speaking, an NLP is of the form:

$$\begin{aligned} \max_{x \in \mathbf{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & h(x) = 0. \end{aligned} \tag{3.1}$$

The solving procedure of an NLP depends on the type of objective function and its constraints. For example, the objective function can be convex, concave or neither. Moreover, it can also a specific kind of equation, such as linear or quadratic. Besides that, one can also distinguish between different feasible sets, for example where all the constraints are linear, or convex in general. In the remainder of this section, we only focus on the NLP classification of the entropy model: a concave or convex nonlinear objective function with linear constraints.

Definition 1. *A function $f(x)$ is called convex if, for all x and y and $0 \leq \lambda \leq 1$:*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y).$$

Definition 2. *A function $f(x)$ is called concave if and only if $-f(x)$ is convex.*

Let us first verify that our objective function (2.13) is concave. Since this function is a real-valued, twice differentiable function on the open interval $\mathbb{R}_{>0}$, we can use the following theorem [1]:

Theorem 3. *Let f be a real-valued, twice differentiable function on the open interval $(a, ..., b)$. Then f is convex on $(a, ..., b)$ if and only if $f'' \geq 0$ on $(a, ..., b)$.*

For our objective function we have:

$$\begin{aligned} f(\mathbf{T}) &= -\sum t_{ijm} ln(t_{ijm}) - t_{ijm}, \\ f'(\mathbf{T}) &= -\sum ln(t_{ijm}), \\ f''(\mathbf{T}) &= -\sum \frac{1}{t_{ijm}}. \end{aligned}$$

Since $-f(\mathbf{T}) = \sum \frac{1}{t_{ijm}} > 0$, we have that $f(T)$ is convex by Definition 1, consequently our objective function (2.13) is concave by Definition 2.

One of the most used procedures to solve these problems are interior-point methods. It is a class of algorithms that are used to solve both linear and nonlinear convex optimization problems. [ref] Two of such methods for the nonlinear case are discussed, the Frank-Wolfe (FW) and Sequential Quadratic Programming (SQP) algorithms, which approximate the objective function in the first order and second order in each iteration respectively.

3.1.1 Frank-Wolfe

The Frank-Wolfe method, also known as the conditional gradient method, was originally proposed by Frank and Wolfe. [6] It is applicable to the general NLP (3.1) when f is a convex differentiable function and the constraints g are linear. It iteratively solves a linear approximation of the objective function in the current iterate, i.e. the current value and an additional linear correction term. For example, in iteration $k+1$ the approximation is given by

$$f(x^k) + \nabla f(x^k)^T (x - x^k).$$

Since x^k, $f(x^k)$ and $\nabla f(x^k)$ are fixed values, maximizing this approximation of the objective function is equivalent to maximizing:

$$\nabla f(x^k)^T x.$$

The optimization problem for this iteration $k+1$ maximizes this function with respect to x, where x has to be feasible, resulting in:

$$\begin{aligned} \min_{x \in \mathbf{R}^m} \quad & \nabla f(x^k)^T (x - x^k) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & h(x) = 0. \end{aligned} \tag{3.2}$$

Even though the linear approximation of the objective function steadily improves from the current iterate x^k to the solution x of 3.5, this might not be the case for the general nonlinear objective $f(x)$. Therefore, the Frank-Wolfe algorithm includes a procedure to find the point along the line between the current iterate and the solution of 3.5 with the maximum value for $f(x)$. The resulting point will then be the next iterate for the Frank-Wolfe algorithm. Different procedures are discussed in Section 4.2.

3.1.2 Sequential Quadratic Programming

The Sequential Quadratic Programming (SQP) procedure is another commonly used algorithm for solving NLP's, in fact, it is considered to be one of the most effective methods for solving constrained nonlinear optimization problems. In contrast to Frank-Wolfe, SQP models the NLP by considering a quadratic programming (QP) subproblem for a given iterate. The QP-subproblem is a quadratic approximation of the objective function for the current iterate. The solution is then used as the next iterate for the algorithm. For iteration $k+1$ the quadratic approximation of the objective function $f(x)$ is given by

$$f(x^k) + \nabla f(x^k)^T (x - x^k) + \frac{1}{2}(x - x^k)^T Hf(x^k)(x - x^k),$$

where Hf has to be positive semi-definite and symmetric, and is defined as:

$$(Hf(x))_{ij} := \frac{\partial^2 f(x)}{\partial x_i \partial x_j}.$$

The optimization problem for this iteration $k+1$ maximizes this function with respect to x, where x has to be feasible, resulting in:

$$\begin{aligned} \min_{x \in \mathbf{R}^m} \quad & \nabla f(x_k)^T (x - x_k) + \tfrac{1}{2}(x - x_k)^T Hf(x_k)(x - x_k) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned} \tag{3.3}$$

3.2 Lagrange Multipliers and KKT conditions

In the next two sections, we will show the relation between the two methods and the parameters of the gravity model. In order to do so, this section provides an explanation of the KKT conditions, the conditions for a solution of an NLP to be optimal. These conditions are a generalization the method of Lagrange Multipliers, a method used for finding the local minima/maxima of a function with only equality constraints. In the context of solving the general NLP 3.1, this means finding the saddle points, i.e. points where all derivatives are zero, of the following Lagrangian, where μ and λ are the KKT-multipliers corresponding to the inequalities and equality constraints respectively.

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \mu^T g(x) + \lambda^T h(x),$$

In order for a point x^* to be a saddle point it has to satisfy the stationary condition $\nabla\mathcal{L}(x^*, \lambda^*, \mu^*)_{x^*} = 0$, or more specifically:

$$\nabla_{x^*}\mathcal{L}(x, \lambda, \mu) = \nabla f(x^*) + \mu^{*T}\nabla g(x) + \lambda^{*T}\nabla h(x) = 0$$

and therefore it is required that the functions f, g and h should be continuously differentiable.

Besides that, a solution x^* to the general NLP 3.1 of course has to be feasible, i.e. satisfy the constraints. This leads to the primal feasibility conditions $g(x^*) \leq 0$ and $h(x^*) = 0$.

The last conditions originate from the equality and inequality constraints and their KKT multipliers. For the inequality constraints $g(x^*) \leq 0$, it is required that for the corresponding KKT multipliers should hold $\mu^* \geq 0$ in order for the dual of the NLP to be feasible. Moreover, it is required that $\mu^{*T} g(x^*) = 0$ due to complementary slackness, which states that if some dual variable $\mu_j^* \geq 0$ then the corresponding constraint must be an equality, $g_j(x) = 0$.

To summarize this section, the KKT conditions for the general NLP 3.1 are stated in the following theorem [7], [5].

Definition 3. *If x^* is an optimal solution to the general NLP 3.1, then there exist KKT multipliers λ^* and μ^* such that the following statements hold true:*

1. *Stationary:* $\nabla f(x^*) - (\nabla g(x^*))^T \mu^* - (\nabla h(x^*))^T \gamma^* = 0$
2. *Primal Feasibility:* $g(x^*) \geq 0$, $h(x^*) = 0$
3. *Complementary Slackness:* $\mu^{*T} g(x^*) = 0$
4. *Dual Feasibility:* $\mu^* \geq 0$

3.3 Validation of Frank-Wolfe

In order to validate the usage of Frank-Wolfe (FW for short) for the entropy model, it needs to be shown that the dual multipliers from the last Frank-Wolfe approximation LP correspond to those from the KKT multipliers of the original problem.
Therefore, let us consider a general convex nonlinear problem with a convex objective function $f(x)$, as below. Note that the entropy model falls under this category, since an equality constraint can be rewritten as two inequality constraints.

$$\begin{aligned} \min_{x \in \mathbf{R}^m} \quad & f(x) \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{3.4}$$

Assume that FW terminates after k iterations, meaning that the solution of iteration k cannot be improved upon. Therefore for the optimal solution of FW we have $x^* = x_k = x_{k+1}$. Recall that the optimization problem for iteration $k+1$ of FW is as stated below.

$$\begin{aligned} \min_{x \in \mathbf{R}^m} \quad & \nabla f(x_k)^T (x - x_k) \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{3.5}$$

Since the theorem stating the relation between the entropy model the and solution of the final iteration of FW involves the dual variables of the latter, let us first derive the dual. Since the constant part of the objective function does not change the optimization problem or the optimal solution, the optimization problem (3.5) is equivalent to:

$$\begin{aligned} \min_{x \in \mathbf{R}^m} \quad & \nabla f(x_k)^T x \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{3.6}$$

Using this formulation we can state the dual of the LP from the last iteration of FW [14]:

$$\begin{aligned} \max_{\mu \in \mathbf{R}^n; \gamma \in \mathbf{R}^m} \quad & b^T \mu \\ \text{s.t.} \quad & A^T \mu + \gamma = \nabla f(x_k) \\ & \mu, \gamma \geq 0 \end{aligned} \tag{3.7}$$

Theorem 4. *Assume that x^* is a optimal solution obtained from the final iteration of FW* (3.6) *and (μ^*, γ^*) is the solution to its dual* (3.7)*. Then these solutions also satisfy the KKT conditions for* (3.4) *[9]:*

1. *(Stationary)* $\nabla f(x^*) + (\nabla(b - Ax^*))^T \mu^* + (\nabla(-x^*))^T \gamma^* = 0$
2. *(Feasibility)* $Ax^* \geq b$, $x^* \geq 0$
3. *(Complementary Slackness)* ${\mu^*}^T (b - Ax^*) = 0$, ${\gamma^*}^T (-x^*) = 0$
4. *(Dual Feasibility)* $\mu^*, \gamma^* \geq 0$

Proof. In order to show KKT condition 1 we need to verify that the following holds.

$$\begin{aligned} & \nabla f(x^*) + (\nabla(b - Ax^*))^T \mu^* + (\nabla(-x^*))^T \gamma^* = 0 \\ \Leftrightarrow \quad & \nabla f(x^*) - A^T \mu^* - \gamma^* = 0 \\ \Leftrightarrow \quad & \nabla f(x^*) = A^T \mu^* + \gamma^* \end{aligned}$$

However, since x^* is an optimal solution to the primal (thus $x^* = x_k$), this equation is also a constraint in the dual (3.7) and therefore holds.

KKT condition 2 immediately follows from the fact that x^* is a solution to (3.6), thus satisfying the constraints $Ax^* \geq b$ and $x^* \geq 0$.

In order to show KKT condition 3, we take a look at the objective values of the primal and the dual. For every primal feasible x and dual feasible (μ, γ), we have that the primal objective value after substituting the dual constraint becomes:

$$\begin{aligned} \nabla f(x_k)^T x &= \left(A^T \mu + \gamma\right)^T x \\ &= \mu^T A x + \gamma^T x \end{aligned}$$

Since both $\mu, \gamma \geq 0$ from the dual constraint and $Ax \geq b$ and $x \geq 0$ from the primal constraint hold, we have found a lower bound to the primal objective value:

$$\nabla f(x_k)^T x \geq \mu^T b$$

Observe that the right-hand side of this inequality is the dual objective value, and the left-hand side was the primal objective value we started with. Therefore, this inequality is the weak duality property of the FW model.
Due to the model also having the strong duality property, for an optimal primal x^* and optimal dual (μ^*, γ^*) this inequality is actually an equality. The only way to obtain this is when:

$$\begin{aligned} \mu^{*T} A x^* + \gamma^{*T} x^* &= \mu^{*T} b \\ \mu^{*T}(Ax^* - b) &= -\gamma^{*T} x^*. \end{aligned} \tag{3.8}$$

Since $\mu^* \geq 0$ and $Ax^* \geq b \Leftrightarrow Ax^* - b \geq 0$, we must have $\mu^{*T}(Ax^* - b) \geq 0$. Furthermore, since $\gamma^* \geq 0$ and $x \geq 0$, we also have $\gamma^{*T} x^* \geq 0$.
Therefore, (3.8) only holds when both $\mu^{*T}(Ax^* - b) = 0$ and $\gamma^{*T} x^* = 0$, which proves KKT condition 3.

KKT condition 4 immediately follows from the non-negativity constraints on μ and γ in the dual.
This concludes the proof. □

In order to show how this theorem helps in our context of the entropy model (2.15)-(2.18), assume that we are solving the model by FW, resulting in a converged run with solution x^*. Applying Theorem 4 gives us that x^* is indeed a local optimum, and due to the convexity of the objective $f(x)$ also a global optimum.

3.4 Validation of SQP

In order to validate the usage of SQP for the entropy model, it needs to be shown that the dual multipliers from the last Frank-Wolfe approximation LP correspond to those from the KKT multipliers of the original problem.
Therefore, let us consider a general convex nonlinear problem with a convex objective function $f(x)$, as below. Note that the entropy model falls under this category, since an equality constraint can be rewritten as two inequality constraints.

$$\begin{aligned} \min_{x \in \mathbf{R}^m} \quad & f(x) \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{3.9}$$

Assume that SQP terminates after k iterations, meaning that the solution of iteration k cannot be improved upon. Therefore for the optimal solution of SQP we have $x^* = x_k = x_{k+1}$. Recall that the optimization problem for iteration $k+1$ of SQP is as stated below, with Hf positive semi-definite and symmetric.

$$\begin{aligned} \min_{x \in \mathbf{R}^m} \quad & \nabla f(x_k)^T (x - x_k) + \tfrac{1}{2}(x - x_k)^T Hf(x_k)(x - x_k) \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{3.10}$$

Since theorem stating the relation between the entropy model the and solution of the final iteration of SQP involves the dual variables of the latter, let us first derive the dual. By separating the objective function of (3.10) into its quadratic, linear and constant part, it is rewritten in a general quadratic objective function, from which we can derive the dual easily.

$$\begin{aligned} &\nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T Hf(x_k)(x - x_k) \\ &= \nabla f(x_k)^T x - \nabla f(x_k)^T x_k + \frac{1}{2} x^T Hf(x_k) x - \frac{1}{2} x^T Hf(x_k) x_k - \frac{1}{2} x_k^T Hf(x_k) x + \frac{1}{2} x_k^T Hf(x_k) x_k \\ &= \frac{1}{2} x^T Hf(x_k) x + \left(\nabla f(x_k)^T - x_k^T Hf(x_k)\right) x + \left(\frac{1}{2} x_k^T Hf(x_k) x_k - \nabla f(x_k)^T x_k\right) \end{aligned}$$

Since the constant part does not change the optimization problem or the optimal solution, the optimization problem (3.10) is equivalent to:

$$\begin{aligned} \min_{x \in \mathbf{R}^m} \quad & \tfrac{1}{2} x^T H f(x_k) x + \left(\nabla f(x_k)^T - x_k^T H f(x_k) \right) x \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{3.11}$$

Using this formulation we can state the dual of the QP from the last iteration of SQP [DORN]:

$$\begin{aligned} \max_{y \in \mathbf{R}^m; \mu \in \mathbf{R}^n; \gamma \in \mathbf{R}^m} \quad & -\tfrac{1}{2} y^T H f(x_k) y + b^T \mu \\ \text{s.t.} \quad & A^T \mu + \gamma - H f(x_k) y = \nabla f(x_k) - H f(x_k) x_k \\ & \mu, \gamma \geq 0 \end{aligned} \tag{3.12}$$

According to Theorem in section 5 of [DORN], if x^* is a solution to the primal, then there exists a solution $(y, \mu, \gamma) = (x^*, \mu^*, \gamma^*)$ that is dual feasible. Furthermore we have strong duality, meaning that the optimal objective values of (3.11) and (3.12) coincide.

Theorem 5. *Assume that x^* is a optimal solution obtained from the final iteration of SQP* (3.11) *and (x^*, μ^*, γ^*) is the solution to its dual* (3.12)*. Then these solutions also satisfy the KKT conditions for* (3.9) *[8]:*

1. *(Stationary)* $\nabla f(x^*) + (\nabla(b - Ax^*))^T \mu^* + (\nabla(-x^*))^T \gamma^* = 0$
2. *(Feasibility)* $Ax^* \geq b$, $x^* \geq 0$
3. *(Complementary Slackness)* ${\mu^*}^T (b - Ax^*) = 0$, ${\gamma^*}^T (-x^*) = 0$
4. *(Dual Feasibility)* $\mu^*, \gamma^* \geq 0$

Proof. In order to show KKT condition 1 we need to verify that the following holds.

$$\begin{aligned} & \nabla f(x^*) + (\nabla(b - Ax^*))^T \mu^* + (\nabla(-x^*))^T \gamma^* = 0 \\ \Leftrightarrow \quad & \nabla f(x^*) - A^T \mu^* - \gamma^* = 0 \\ \Leftrightarrow \quad & \nabla f(x^*) = A^T \mu^* + \gamma^* \end{aligned} \tag{3.13}$$

This condition is shown by taking a look at the dual constraint for an optimal dual solution:

$$\begin{aligned} A^T \mu^* + \gamma - H f(x_k) x^* &= \nabla f(x_k) - H f(x_k) x_k \\ A^T \mu^* + \gamma &= \nabla f(x_k) + H f(x_k)(x^* - x_k) \end{aligned}$$

Due to x^* being an optimal solution to the primal, we have that $x^* = x_k$. Therefore it can be further simplified to:

$$\begin{aligned} A^T \mu^* + \gamma &= \nabla f(x^*) + H f(x^*)(x^* - x^*) \\ A^T \mu^* + \gamma &= \nabla f(x^*) \end{aligned}$$

which equals the first KKT condition (3.13) that we needed to show.

KKT condition 2 immediately follows from the fact that x^* is a solution to (3.11), thus satisfying the constraints $Ax^* \geq b$ and $x^* \geq 0$.

In order to show KKT condition 3, we take a look at the objective values of the primal and the dual. For every primal feasible x and dual feasible (x, μ, γ), we have that the primal objective value after substituting the dual constraint becomes:

$$\begin{aligned} \frac{1}{2} x^T H f(x_k) x + \left(\nabla f(x_k)^T - x_k^T H f(x_k) \right) x &= \frac{1}{2} x^T H f(x_k) x + \left(A^T \mu + \gamma - H f(x_k) x \right)^T x \\ &= -\frac{1}{2} x^T H f(x_k) x + \mu^T A x + \gamma^T x \end{aligned}$$

Since both $\mu, \gamma \geq 0$ from the dual constraint and $Ax \geq b$ and $x \geq 0$ from the primal constraint hold, we have found a lower bound to the primal objective value:

$$\frac{1}{2}x^T Hf(x_k)x + \left(\nabla f(x_k)^T - x_k^T Hf(x_k)\right)x \geq -\frac{1}{2}x^T Hf(x_k)x + \mu^T b$$

Observe that the right-hand side of this inequality is the dual objective value, and the left-hand side was the primal objective value we started with. Therefore, this inequality is the weak duality property of the SQP model.

Due to the model also having the strong duality property, for an optimal primal x^* and optimal dual (x^*, μ^*, γ^*) this inequality is actually an equality. The only way to obtain this is when:

$$\begin{aligned} \mu^{*T}Ax^* + \gamma^{*T}x^* &= \mu^{*T}b \\ \mu^{*T}(Ax^* - b) &= -\gamma^{*T}x^*. \end{aligned} \tag{3.14}$$

Since $\mu^* \geq 0$ and $Ax^* \geq b \Leftrightarrow Ax^* - b \geq 0$, we must have $\mu^{*T}(Ax^* - b) \geq 0$. Furthermore, since $\gamma^* \geq 0$ and $x \geq 0$, we also have $\gamma^{*T}x^* \geq 0$.
Therefore, (3.14) only holds when both $\mu^{*T}(Ax^* - b) = 0$ and $\gamma^{*T}x^* = 0$, which proves KKT condition 3.

KKT condition 4 immediately follows from the non-negativity constraints on μ and γ in the dual.
This concludes the proof. □

In order to show how this theorem helps in our context of the entropy model (2.15)-(2.18), assume that we're solving the model by SQP, resulting in a converged run with solution x^*. Applying Theorem 5 gives us that x^* is indeed a local optimum, and due to the convexity of the objective $f(x)$ also a global optimum.

4 FW/SQP Applied to the Trip Model

This section covers the process of converting the mathematical based model in section 3 into a working implementation. Firstly, Section 4.1 states what the trip model looks like when we apply FW or SQP to the entropy formulation (2.15)-(2.18). However, there are certain specifications of the model have to be chosen. Section 4.2 explains how the step size can be chosen when using FW and Section 4.3 discusses various convergence criteria that one could use to determine whether the resulting trip distribution has converged. Lastly, Section 4.4 states how the entropy formulation can be extended by introducing distance bins in order to obtain beta values per distance bin.

4.1 Mathematical Formulation and Pseudocode

First of all, the mathematical models for FW (3.10) and SQP (3.5) should be applied to the entropy formulation of the problem in (2.15)-(2.18). Therefore, let us derive the gradient and hessian first.

$$
\begin{aligned}
f(\mathbf{T}) &= - \sum_{i\in\mathcal{I},j\in\mathcal{J},m\in\mathcal{M},u\in\mathcal{U}} t_{ijmu}\cdot ln^2(t_{ijmu}) - t_{ijmu} \\
\frac{\partial f(\mathbf{T})}{\partial t_{ijmu}} &= -ln(t_{ijmu}) \\
\frac{\partial^2 f(\mathbf{T})}{\partial t_{i_1j_1m_1u_1}\partial t_{i_2j_2m_2u_2}} &= \begin{cases} -\frac{1}{t_{i_1j_1m_1u_1}} & \text{if } i_1=i_2, j_1=j_2, m_1=m_2, u_1=u_2 \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

Substituting these into the FW formulation results in the following model for iteration l. Here, $\left(s^l_{ijmu} - t^{l-1}_{ijmu}\right)$ is shortened to $d(s_{ijmu})$.

$$
\begin{aligned}
\max_{\mathbf{S}^l} \quad & \sum_{i,j,m,u} \left(-ln\left(t^{l-1}_{ijmu}\right)\right)\cdot d(s_{ijmu}) & & \\
\text{s.t.} \quad & \sum_{j,m} s^l_{ijmu} = P_{iu} & & i\in\mathcal{I}, u\in\mathcal{U} \\
& \sum_{i,m,u} s^l_{ijmu} = A_j & & j\in\mathcal{J} \\
& \sum_{i,j} s^l_{ijmu} = M_{mu} & & m\in\mathcal{M}, u\in\mathcal{U} \\
& \sum_{i,j} s^l_{ijmu}\cdot ln^2(c_{ijm}+1) = C_{mu} & & m\in\mathcal{M}, u\in\mathcal{U} \\
& s^l_{ijmu} \geq 0 & & i\in\mathcal{I}, j\in\mathcal{J}, m\in\mathcal{M}, u\in\mathcal{U}
\end{aligned}
\tag{4.1}
$$

Similarly, the formulation of the SQP applied model in iteration l becomes:

$$
\begin{aligned}
\max_{\mathbf{S}^l} \quad & \sum_{i,j,m,u} \left(-ln\left(t^{l-1}_{ijmu}\right)\right)\cdot d(s_{ijmu}) - \tfrac{1}{2} d(s_{ijmu})\cdot \tfrac{1}{t^{l-1}_{ijmu}}\cdot d(s_{ijmu}) & & \\
\text{s.t.} \quad & \sum_{j,m} s^l_{ijmu} = P_{iu} & & i\in\mathcal{I}, u\in\mathcal{U} \\
& \sum_{i,m,u} s^l_{ijmu} = A_j & & j\in\mathcal{J} \\
& \sum_{i,j} s^l_{ijmu} = M_{mu} & & m\in\mathcal{M}, u\in\mathcal{U} \\
& \sum_{i,j} s^l_{ijmu}\cdot ln^2(c_{ijm}+1) = C_{mu} & & m\in\mathcal{M}, u\in\mathcal{U} \\
& s^l_{ijmu} \geq 0 & & i\in\mathcal{I}, j\in\mathcal{J}, m\in\mathcal{M}, u\in\mathcal{U}
\end{aligned}
\tag{4.2}
$$

Note that both formulations are identical besides the fact that there is an additional second order term in the objective function for SQP that is non-existent for FW. Therefore, in the implementation we will consider them both at the same time and specify the few differences where needed. Again, we stress the issue of the unknown values for C_m. How these are solved when applying the model are discussed in Section 5 at the relevant subsections.

Algorithm 3 Pseudocode of the entropy model when FW or SQP is applied.

load data: P, A, c, M, C

find *feasible solution* t^0_{ijmu} *subject to the constraints:* ▷ *Initial solution*

$\sum_{j,m} t^0_{ijmu} = P_{iu} \qquad \forall i, u$

$\sum_{i,m,u} t^0_{ijmu} = A_j \qquad \forall j$

$\sum_{i,j} t^0_{ijmu} = M_{mu} \qquad \forall m, u$

$\sum_{i,j} t^0_{ijmu} ln^2(c_{ijm} + 1) = C_{mu} \qquad \forall m, u$

$t^0_{ijmu} \geq 0 \qquad \forall i, j, m, u$

while *not converged* **do**

$d(s_{ijmu}) = \left(s^l_{ijmu} - t^{l-1}_{ijmu}\right)$

find *solution* s^l_{ijmu} *of the optimization problem:*

if *FW* **then** objective $= \sum_{i,j,m,u} \left(-ln\left(t^{l-1}_{ijmu}\right)\right) \cdot d(s_{ijmu})$ ▷ Different objective functions

if *SQP* **then** objective $= \sum_{i,j,m,u} \left(-ln\left(t^{l-1}_{ijmu}\right)\right) \cdot d(s_{ijmu}) - \frac{1}{2} d(s_{ijmu}) \cdot \frac{1}{t^{l-1}_{ijmu}} \cdot d(s_{ijmu})$

maximize objective **subject to:** ▷ Solve subproblem

$\sum_{j,m} s^l_{ijmu} = P_{iu} \qquad \forall i, u$

$\sum_{i,m,u} s^l_{ijmu} = A_j \qquad \forall j$

$\sum_{i,j} s^l_{ijmu} = M_{mu} \qquad \forall m, u$

$\sum_{i,j} s^l_{ijmu} ln^2(c_{ijm} + 1) = C_{mu} \qquad \forall m, u \; s^l_{ijmu} \geq 0 \qquad \forall i, j, m, u$

if *FW* **then**

determine step size γ ▷ Determine next iterate

$t^l_{ijmu} = (1 - \gamma) \cdot t^{l-1}_{ijmu} + \gamma \cdot s^l_{ijmu}$

else if *SQP* **then**

$t^l_{ijmu} = s^l_{ijmu}$

end

As one can see, there are two differences between FW and SQP: the additional term in the objective function for SQP as discussed before, and the computation of the new iterate. The SQP procedure uses the solution of the QP subproblem as the next iterate, but in the case of FW, it takes the current iterate into account as well. For FW, the next iterate is a convex combination of the current iterate and the solution of the subproblem, $\mathbf{T}^l = (1 - \gamma_l) \cdot \mathbf{T}^{l-1} + \gamma_l \cdot \mathbf{S}^l$, where γ_l denotes the fraction of the latter and is denoted as the step size for iteration l.

4.2 Step Size for FW

Generally speaking, there are two ways to determine the value of γ_l, inexact and exact. In the inexact method, γ is a predetermined value that decreases in the number of iterations done (denoted by l), usually this value is similar to $\frac{1}{2l}$. The exact method determines γ by using line search. Since the optimal value of γ_l is somewhere along a line created by a convex combination of two known trip distributions, we can apply this method. By looking at the corresponding entropy values, we can find the γ_l that results in the highest entropy value with respect to the two known trip distributions.

Note that the resulting new iterate $(1 - \gamma_l) \cdot \mathbf{T}^{l-1} + \gamma_l \cdot \mathbf{S}^l$ is still a valid trip distribution, i.e. it satisfies the constraints in (4.1):

$$\sum_{j,m}(1-\gamma_l)t^{l-1}_{ijmu}+\gamma_l s^l_{ijmu}=(1-\gamma_l)\sum_{j,m}t^{l-1}_{ijmu}+\gamma_l\sum_{j,m}s^l_{ijmu}=(1-\gamma_l)P_{iu}+\gamma_l P_{iu}=P_{iu},$$

$$\sum_{i,u,m}(1-\gamma_l)t^{l-1}_{ijmu}+\gamma_l s^l_{ijmu}=(1-\gamma_l)\sum_{i,u,m}t^{l-1}_{ijmu}+\gamma_l\sum_{i,u,m}s^l_{ijmu}=(1-\gamma_l)A_j+\gamma_l A_j=A_j,$$

$$\sum_{i,j}(1-\gamma_l)t^{l-1}_{ijmu}+\gamma_l s^l_{ijmu}=(1-\gamma_l)\sum_{i,j}t^{l-1}_{ijmu}+\gamma_l\sum_{i,j}s^l_{ijmu}=(1-\gamma_l)M_{mu}+\gamma_l M_{mu}=M_{mu},$$

$$\begin{aligned}\sum_{i,j}(1-\gamma_l)t^{l-1}_{ijmu}ln^2(c_{ijm})+\gamma_l s^l_{ijmu}ln^2(c_{ijm}) &= (1-\gamma_l)\sum_{i,j}t^{l-1}_{ijmu}ln^2(c_{ijm})+\gamma_l\sum_{i,j}s^l_{ijmu}ln^2(c_{ijm})\\ &= (1-\gamma_l)C_{mu}+\gamma_l C_{mu}=C_{mu},\end{aligned}$$

which follows from the fact that both $\mathbf{T}^{l-1}$ and $\mathbf{S}^l$ are trip distributions that satisfy the constraints in (4.1).

Even though the exact line search procedure results in a more accurate γ_l, it is obviously more computationally expensive than a set value (per iteration). On the other hand, due to its accuracy, it requires fewer iterations before the solution of FW has converged. Therefore, a decision between speed and accuracy has to be made.

4.3 Convergence Criteria

For both FW and SQP a suitable convergence criterion is required to determine whether the solution has converged. Below are the possible convergence criteria listed together with an argumentation of why they are applicable.

1. **Largest (absolute) change in trip distribution**. The model has converged when the resulting trip distribution does not change within an iteration. Therefore, for iteration l, we consider the model to be converged when the largest element of $\mid \mathbf{T}^{l-1}-\mathbf{T}^{l} \mid$ is below a certain threshold.
2. **Increase in objective value**. The model is an optimization problem that maximizes the entropy, thus it would be reasonable to consider its behaviour over the iterations as a possible convergence criterion. Since the optimal value is different for each model, especially when looking at the total amount of trips, we only consider the relative change in objective value over an iteration. Therefore, for iteration l, we consider the model to be converged when $\frac{obj(\mathbf{T}^l)-obj(\mathbf{T}^{l-1})}{obj(\mathbf{T}^{l-1})}$ is below a certain threshold.
3. **Absolute change in beta values**. The main goal of the model is to determine accurate beta values, so that they can be used as an input to faster computable trip models they currently use at DAT.Mobility. Therefore, a convergence criterion based on the change of the beta values over an iteration would be plausible as well. For iteration l, we consider the model to be converged when $\mid \beta^{l-1}_{mu}-\beta^{l}_{mu} \mid$ is below a certain threshold $\forall m\in\mathcal{M}, u\in\mathcal{U}$.
4. **Relative change in beta values**. The corresponding relative version of the method described above. For iteration l we consider the model to be converged when $\mid \frac{\beta^{l-1}_{mu}-\beta^{l}_{mu}}{\beta^{l-1}_{mu}} \mid$ is below a certain threshold $\forall m\in\mathcal{M}, u\in\mathcal{U}$.

In the remainder of this thesis, either the first or the fourth convergence criteria will be applied to the model. Since the main goal is to find accurate beta values, in most cases the convergence criteria will be the absolute change in beta values. One important remark is that it is possible for the beta values to not change much in the first few iterations. Therefore, the model should do at least a couple of iterations before it the convergence criteria is applied. More details on this behaviour can be found in Section 5.2.
The only situation in which the first criteria is applied, is when dealing with randomly generated data as

is done for analyzing the performance of the model in Section 5.3. In this case it is unknown beforehand how large or how small these values will be, since the randomly generated data does not represent a realistic situation. Therefore, the most strict convergence criteria is selected in order to guarantee that the model has converged. This convergence criteria could be used in all instances, but due to the last couple of iterations only changing the beta values slightly when using realistic data (see Section 5.2 for more details), the decision is made to select the fourth criteria in order to save time in these cases.

4.4 Parameters per Distance Bin

An extension that can be made to the entropy optimization problem that wasn't possible previously, is to have a constraint and thus beta parameter per distance bin. This idea originated from having a discrete deterrence function and this extension would be a hybrid between a discrete and a lognormal function. The extension is possible due to the fact that the calibration method for the gravity model calibrates with respect to the trip length distribution, whereas it can be specified as constraints in the entropy optimization problem. Remember that the entropy formulation has the constraints:

$$\sum_{i\in\mathcal{I},j\in\mathcal{J}} t_{ijmu} = M_{mu} \qquad \forall m \in \mathcal{M} \tag{4.3}$$

$$\sum_{i\in\mathcal{I},j\in\mathcal{J}} t_{ijmu}c_{ijm} = C_{mu} \qquad \forall m \in \mathcal{M} \tag{4.4}$$

Let us introduce K distance bins with increasing breakpoints $0 = \delta_0, \delta_1, ..., \delta_K$ for certain costs. For the k-th distance bin this gives $D_k = c_{ijm} : \delta_{k-1} \leq c < \delta_k$ for $k \in \mathcal{K} = \{1, ..., K\}$. Substituting these distance bins in the constraints (4.3) and (4.4) results in the new constraints:

$$\sum_{i\in\mathcal{I},j\in\mathcal{J}} \sum_{c_{ijm}\in D_k} t_{ijmu} = M_{muk} \qquad \forall m \in \mathcal{M}, k \in \mathcal{K} \tag{4.5}$$

$$\sum_{i\in\mathcal{I},j\in\mathcal{J}} \sum_{c_{ijm}\in D_k} t_{ijmu}c_{ijm} = C_{muk} \qquad \forall m \in \mathcal{M}, k \in \mathcal{K} \tag{4.6}$$

5 Results

In this section the implemented version of both SQP and FW is tested on various instances. Unfortunately, due to unknown reasons the FW model did not work properly for realistic instances. It would either iterate infinitely long with the smallest step size possible, or take an unreasonably long time per iteration with the corresponding intermediate beta values being always zero. However, when using randomly generated instances it did result into the expected outcome and the above issues never occurred. Therefore, in the remainder of this section, we only look at the results for the SQP method, except for Section 5.3 where synthetic data is used in order to analyze the performance of the different methods.

In Section 5.1 and 5.2 the SQP implementation is validated for the small model of Delft and the more realistic model of Almere respectively. For these runs the model input is determined in such a way that it should result in the same beta values as used for the gravity model. If we have that the beta values of the SQP model (i.e. dual multipliers) and the trip-based gravity model (i.e. KKT multipliers) coincide, then this follows the mathematical proof in Section 3.4 and the implementation is validated. In Section 5.3 randomly generated data is used to compare the performance of the the different available methods: the SQP model, the FW model with a set step size per iteration and the FW using a line search procedure to determine the step size. Section 5.4 compares the results for the purpose 'Work' of the SQP model and the trip-based gravity model when the same exogenous data is used. In this case, the used exogenous data source is OViN ("Onderzoek Verplaatsingen in Nederland") and is managed by Statistics Netherlands (CBS). OViN yearly produces a dataset storing reports of movements by Dutch people. Respondents receive a questionnaire and are asked to fill out their made trips together with its specifications. This results in sample of the Dutch population (about 0.25% filled in a questionnaire), after which the samples are scaled to approximate the trips of the whole population [4]. All models made by DAT.Mobility are based on OViN, i.e., the model input (productions, attractions, deterrence function parameters, etc.) are calculated using OViN. Finally, in Section 5.5 we take a look at the new extension as discussed in 4.4 and its resulting deterrence functions.

5.1 Validation: Delft, 25 zones

Specifications of the Delft model:

- Number of zones: 25, the city of Delft and its adjacent regions
- Number of modes: 3, {Car, Public Transit (PT), Bike}
- No userclasses

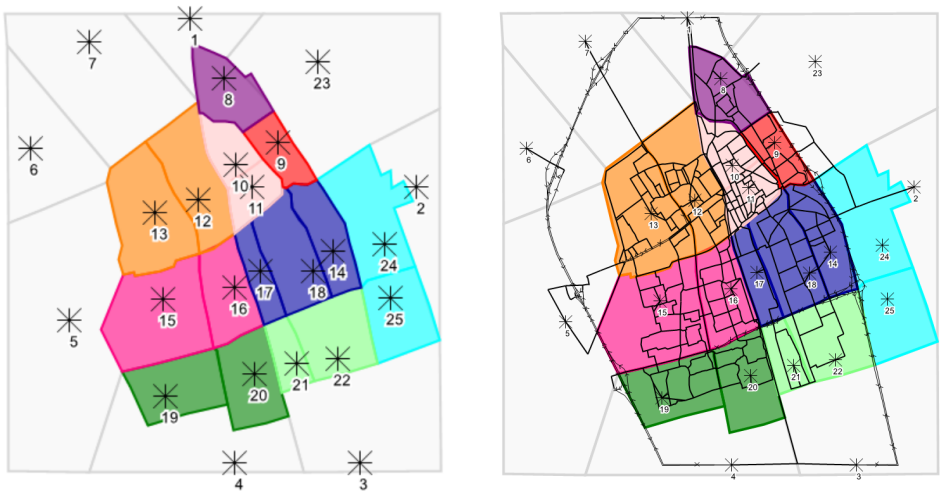


Figure 5: The left picture shows an overview of all zones in the Delft model and the detailed version showing the roads is depicted on the right, emphasizing the simplicity of this model. The $(*)$ sign denotes a centroid of a zone.

The first test to verify that the model works correctly was done on a small example for the city of Delft, also used in the tutorial for Omnitrans, the transport planning software used at DAT.Mobility. For this particular example, only the trips that are made with the purpose 'Work' in a day are modelled.
The SQP model uses the same input as the trip model, but there is one problem: the trip model uses the parameters α_m and β_m. These are not of use for the SQP model, in fact, β_m is the main result of the model, and somehow have to be converted to the constraint values M_m and C_m:

$$\sum_{i\in\mathcal{I},j\in\mathcal{J}} t_{ijm} = M_m, \quad \forall m \in \mathcal{M} \qquad \text{and}$$

$$\sum_{i\in\mathcal{I},j\in\mathcal{J}} t_{ijm} ln^2(c_{ijm}+1) = C_m, \quad \forall m \in \mathcal{M},$$

Since the t_{ijm} values (i.e. the trip distribution) are unknown beforehand, they have to be estimated in a certain way. When comparing both models one should use the same exogenous data used for deriving the model input for the trip model. However, since we only want to validate the model in this section, it is sufficient to use for example the resulting trip distribution from the trip model. This ensures that the resulting values of C_m and M_m are consistent with the β_m and α_m used as input respectively. To give more intuition on why this is reasonable to do, think of it this way: the α_m and $beta_m$ are used to describe the willingness of a person to travel using mode m at various costs (see Section 2.1.4), this behaviour is encapsulated in the trip distribution, thus making it appropriate to calculate M_m and C_m with.

After obtaining the correct input, the SQP results in the beta parameters given in Table 1. The important difference between these values is that the second column are the parameters used by the trip-based gravity model in order to estimate the trip distribution, whereas the third column are the estimated parameters by the SQP method that satisfy the modal split and budget constraints derived from the solution of the trip-based gravity model.

	Trip-based gravity model	SQP model
β_{car}	-0.5	-0.4999971
β_{pt}	-0.5	-0.4999955
β_{bike}	-0.6	-0.5999961

Table 1: Beta values after the validation run for the SQP model on Delft.

5.2 Validation: Almere, 1400 zones

Specifications of the Almere model:

- Number of zones: 1400, the Netherlands but mainly focused on the city of Almere
- Number of modes: 3, {Car, Public Transit (PT), Bike}
- Number of userclasses: 2, {Car owners (co), Non car owners (nco)}

Again, only the trips that are made with the purpose 'Work' in a day are modelled.

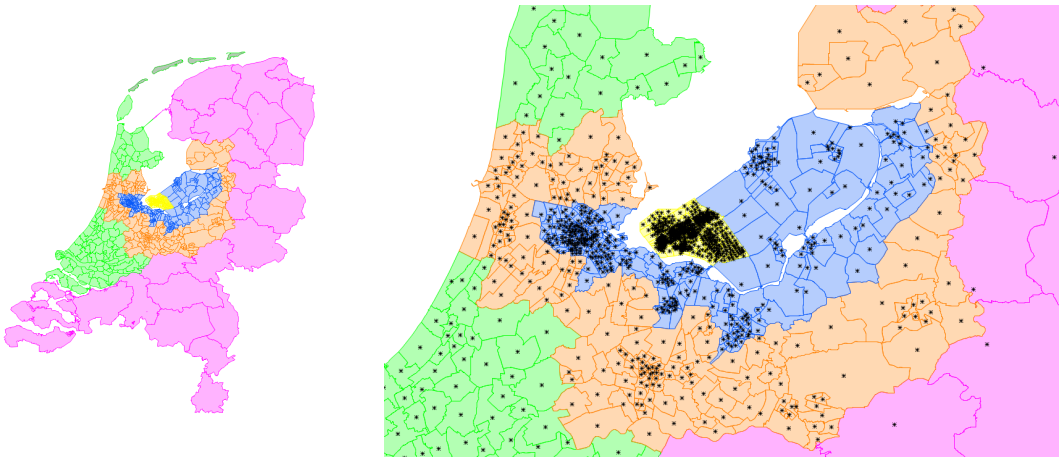


Figure 6: An overview of the zones in the Almere model. The left picture shows all zones, whereas the right one is zoomed in on the city of Almere (in yellow). The $(*)$ sign in the right picture denotes a centroid of a zone and visualizes the density of the zones around Almere.

We have the exact same problem with the model input as for the Delft model (Section 5.1). In the same way as described there the missing values of M_{mu} and C_{mu} are computed from the resulting trip distribution of the trip model.

After obtaining the correct input, the SQP model gives the following resulting β_{mu} values. Again, note that there is a huge difference in how one obtains these values, they are an input to the gravity model and a result of the SQP model.

	Trip model	SQP model
$\beta_{car,co}$	-0.662	-0.66192
$\beta_{pt,co}$	-0.447	-0.44701
$\beta_{bike,co}$	-1.131	-1.13259
$\beta_{car,nco}$	-0.712	-0.71192
$\beta_{pt,nco}$	-0.463	-0.46296
$\beta_{bike,nco}$	-1.182	-1.18277

Table 2: Beta values after the validation run for the SQP model on Almere.

5.3 Performance

This section analyzes how scalable both SQP and FW are by using synthetic data. We have to use randomly generated data here, since most realistic data instances that are used at DAT.Mobility have thousands of zones. Almere with its 1400 zones is one of the smallest models they work with.

For these runs we looked at 5 different number of zones values, namely 300, 600, 900, 1200 and 1500. Each one has 2 userclasses and 3 modes, as is the case for almost all models at DAT.Mobility and 5 instances are randomly generated. Each of the 3 models is tested against every instance and in the remainder of this section the shown results are the mean of those 5 instances.

There are mainly two interesting results to look at when analyzing these runs: running time and memory usage. In order for the methods to be applicable to larger instances than Almere, it has to both compute within a reasonable about of time and not be restricted by the amount of RAM in a computer. For the remainder of this section, if we are talking about memory usage, note that this is about the amount of used RAM (in GB).

In order to strengthen these results, let us first state something about the level of convergence. The final

objective values of the 25 instances are compared between the three different methods and for each one of the instances, the objective values are equal for all three methods. On top of that, due to the convergence of the iterative optimization algorithms and choosing the most strict convergence criteria (see Section 4.3), the number of iterations does not vary that much either. All 25 instances for all three methods were solved in 21 to 23 iterations. Therefore, we can safely compare the results that are discussed below.

5.3.1 Running Times

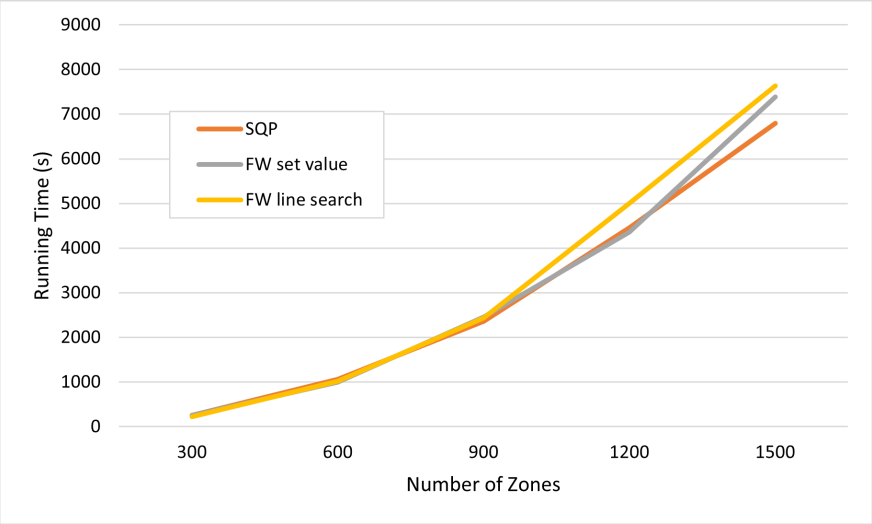


Figure 7: Running times of the three methods with synthetic data

The running times of the three methods are really close to one another. Obviously, the line search method takes longer to compute the step size for FW than a set value, which one can see in Figure 7 as well. Even though SQP solves a QP in each iteration and FW an LP, the running times of SQP seem to be slightly lower. In order to determine the behaviour of the running times, let us look at the corresponding log-log plot:

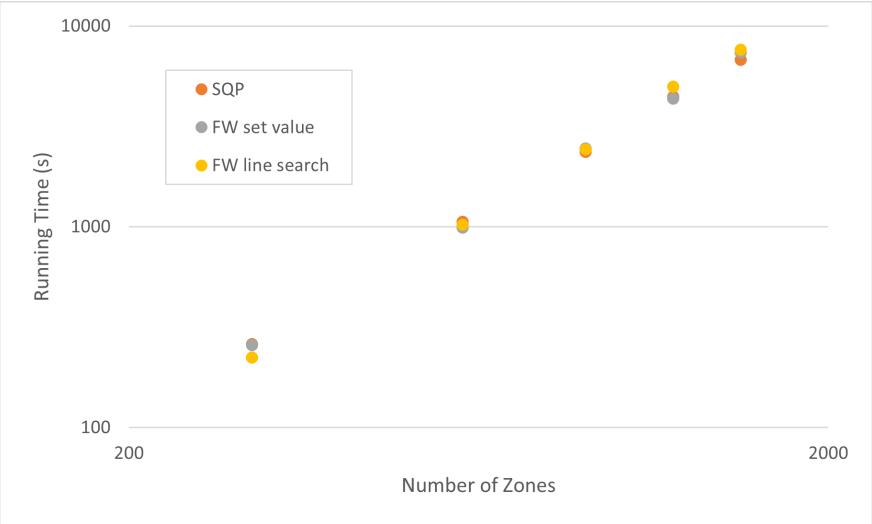


Figure 8: Log-log plot of the running times of the three methods with synthetic data

As one can see, the points lie on a straight line in the log-log plot of Figure 8, which indicates that the running times follow a polynomial behaviour. When looking at the power trendline between those points for each method, we obtain the following table, where y denotes the running time in seconds and n the number of zones.

Method	Formula power trendline
SQP	$y = 0,0024n^{2.033}$
FW set value	$y = 0,0017n^{2.082}$
FW line search	$y = 0,0007n^{2,211}$

Table 3: Power trendlines for the log-log plot of Figure 8

For both SQP and FW with a set step size value, the power of the trendline is very close to 2 and the FW method using a line search procedure is a bit behind them, probably due to the inclusion of the line search procedure. Since those values are close to 2, we can conclude that the running times increase according to a quadratic function.

If we assume that this behaviour is the same even for instances with more zones, we can extrapolate the obtained functions to other models that are used at DAT.Mobility in order to have an idea of what their running time would be. Most urban and regional models they use have around 4.000 to 6.000 zones, with the most ambitious model having 13.000 zones. Assuming that there are no reasons (e.g. memory issues) why the models would not work, this results in the following predicted running times:

Number of zones	SQP	FW set value	FW line search
4.000	14,0 hours	14,9 hours	17,9 hours
6.000	32,0 hours	34,6 hours	43,9 hours
13.000	154,0 hours	173,5 hours	242,5 hours

Table 4: Predicted (extrapolated) running times of other models used at DAT.Mobility

5.3.2 Memory Usage

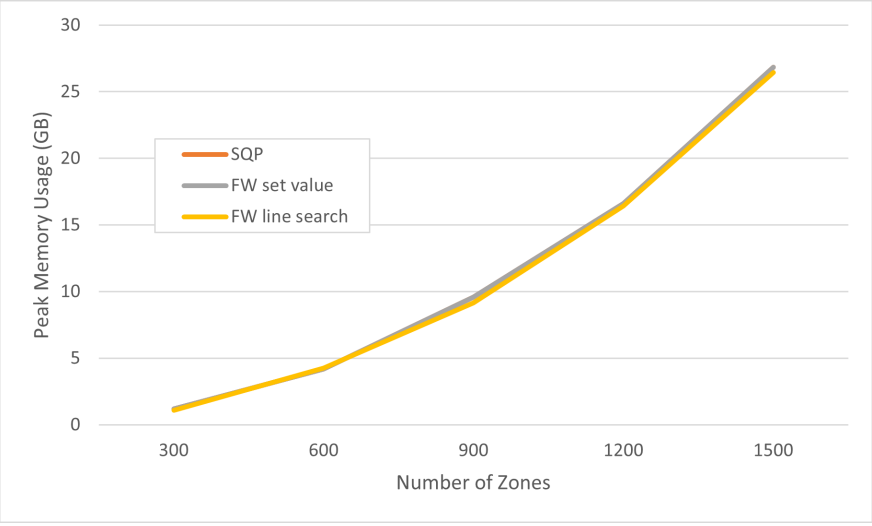


Figure 9: Peak memory usage of the SQP model with synthetic data

When looking at the peak memory usage throughout a run for the models, the results are even closer to each other compared to Figure 7. Since the methods are very similar in terms of their algorithms, they have

to keep track of the same variables and constants and use roughly the same functions. This also results in their peak memory usage being close to the same.

Similarly as for the running times, let us depict Figure 14 on a log-log scale in order to determine their behaviour over the number of zones.

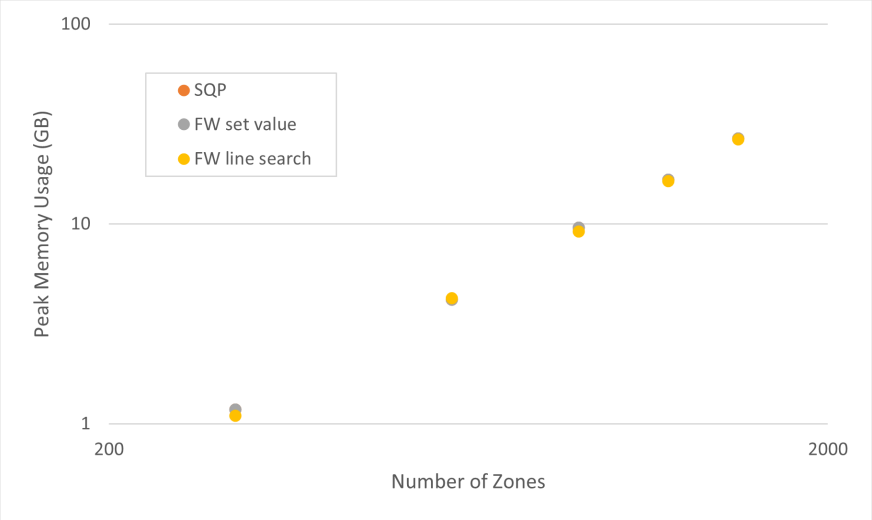


Figure 10: Log-log plot of the peak memory usage for the three models with synthetic data

Again, this results in the points being on a straight line, indicating a polynomial behaviour. Since the three methods are so close to one another, let us only consider the trendline for the SQP method, which is $z = 0,00005n^{1,939}$ where z is the memory usage in GB and n the number of zones.

If we assume the same behaviour throughout every given number of zones, we can extrapolate the results and look at the same instances as in Table 5:

Number of zones	Predicted peak memory usage
4.000	193 GB
6.000	424 GB
13.000	1900 GB

Table 5: Predicted (extrapolated) peak memory usage of other models used at DAT.Mobility

5.4 Comparison Trip-Based Gravity Model and SQP Model

In this section we compare the results of the trip-based gravity model and the SQP model for the trips made with purpose 'Work'. In order to do so, we use the same exogenous data source OViN (see Section 5 for more information). Since this data was already used previously to calibrate the beta parameters for the trip-based gravity model, the values are unchanged from the ones we have been using before. Again, we have to deal with the problem of not having the constraint values M_m and C_m. However, the M_m values follow immediately from OViN, which actually gives us these value per distance bin: M_{mk}. The values of M_m are obtained by aggregating over the distance bind.

For this run, the values of C_m are estimated by looking at the average value of $ln^2(c_{ijm} + 1)$ in the solution of the trip-based gravity model for each distance bin. This results in an average budget value per trip per distance bin k and is multiplied by the number of trips in that distance bin M_{mk} (from OViN) in order to obtain an estimation of the value C_{mk}. The values of C_m are then obtained by summing over the distance bins.

The resulting beta parameters are listed in Table 6 and the trip length distributions of the modes car, public transit and bike are shown in Figure 11, 11 and 11 respectively. Regarding the running time, it took the SQP model around 75 minutes to compute the resulting trip distribution.

	Trip model	Validation Run	Exogenous data
$\beta_{car,co}$	-0.662	-0.66192	-0.235
$\beta_{pt,co}$	-0.447	-0.44701	-0.207
$\beta_{bike,co}$	-1.131	-1.13259	-0.708
$\beta_{car,nco}$	-0.712	-0.71192	-0.245
$\beta_{pt,nco}$	-0.463	-0.46296	-0.201
$\beta_{bike,nco}$	-1.182	-1.18277	-0.717

Table 6: Comparison of beta values after the exogenous run for Almere

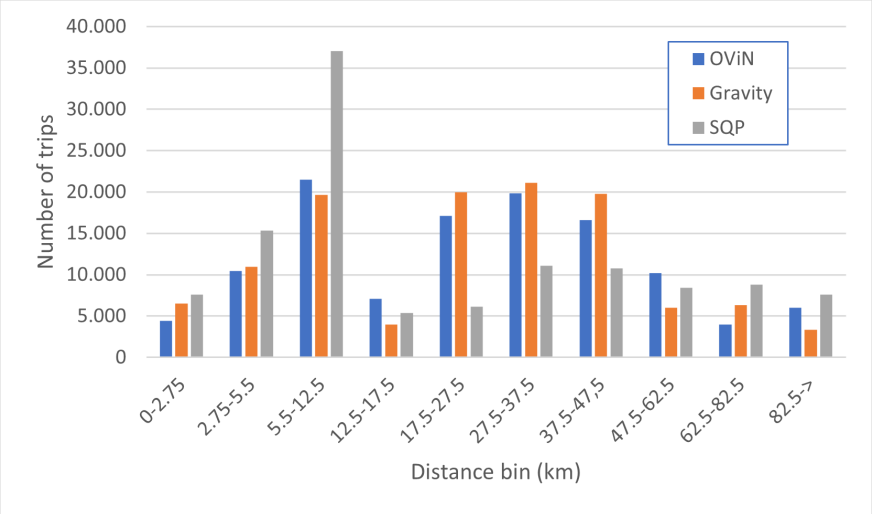


Figure 11: Trip length distribution for the mode Car

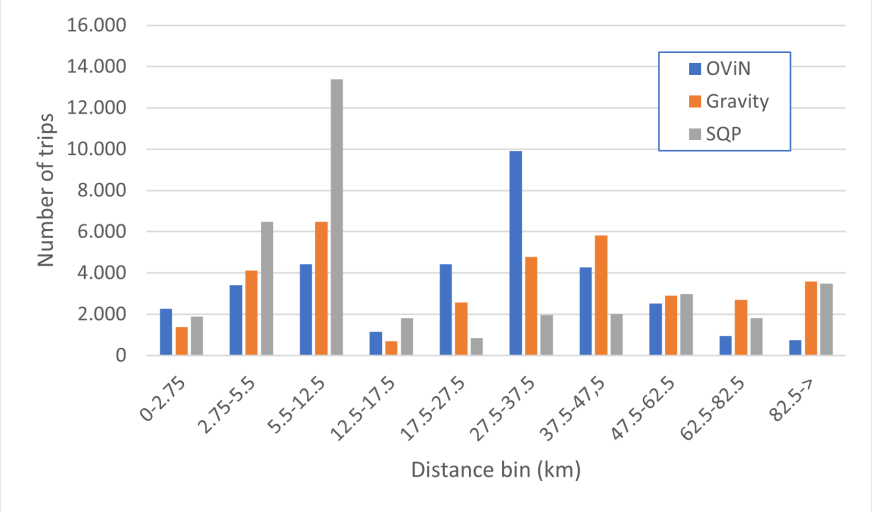


Figure 12: Trip length distribution for the mode Public Transit

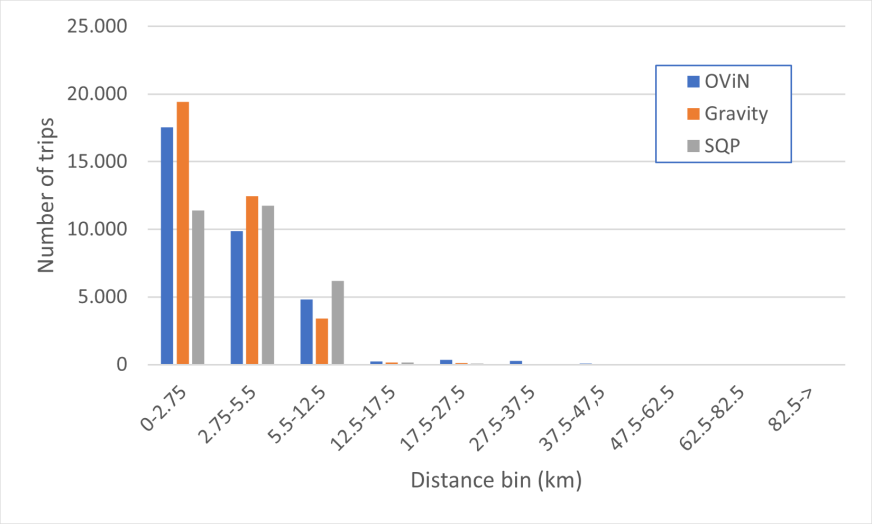


Figure 13: Trip length distribution for the mode Bike

5.5 Distance Bins Extension

This section covers the results of the new opportunity that the SQP model gives, namely a beta parameter and therefore a deterrence function per distance bin. For this run, the same exogenous data is used as in Section 5.4, where also the explanation is given on how to obtain the constraint values M_{mk} and C_{mk}.

The results of the different deterrence functions are found in Figure 15. Note that the deterrence function has the generalized costs as variables, whereas we obtain a beta parameter per distance bin. In order to still visualize this function, the distance bins are approximated by generalized cost bins by comparing the distance and generalized costs of all trips.

Since the trip length distribution is encapsulated in the constraints by M_m in the SQP model with distance bins, it would be useless to show them compared to OViN and the trip-based gravity model in contrast to Section 5.4. In terms of running time, it is a bit slower than the SQP model without distance bins, but not by that much. The regular SQP model terminates in around 75 minutes, whereas the distance bin version runs in 100 minutes.

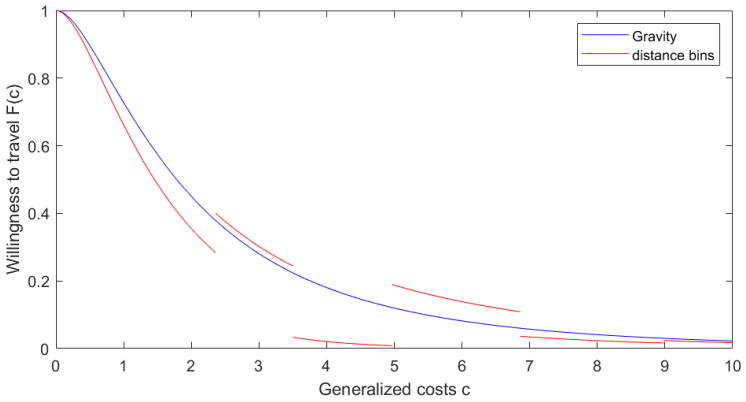
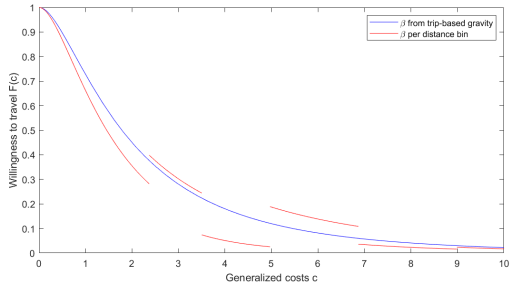
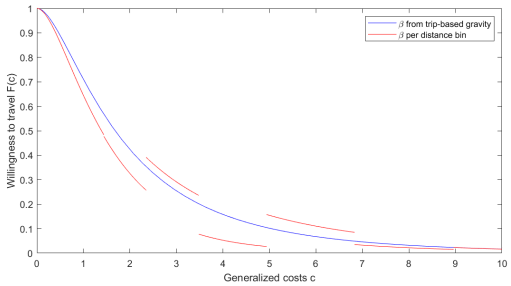


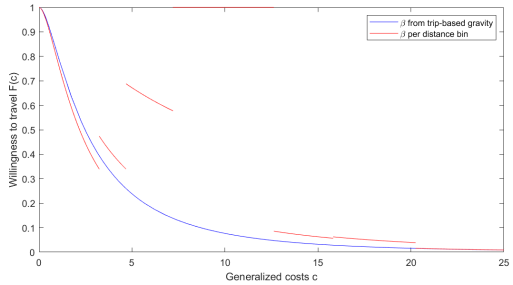
Figure 14: Plot of a beta per distance bin and a general beta for the mode-user pair car, co



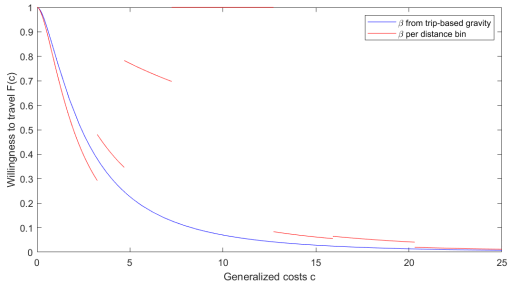
(a) Car, car owners



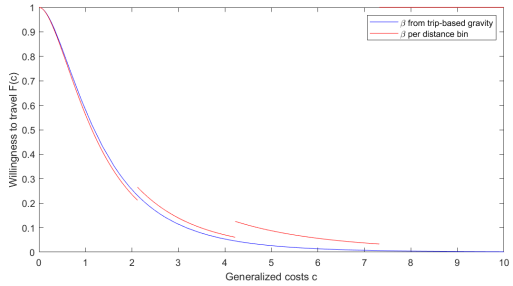
(b) Car, non-car owners



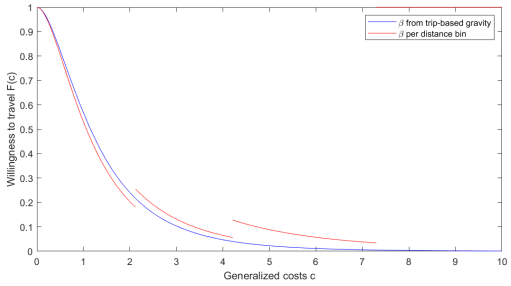
(c) Public Transit, car owners



(d) Public Transit, non-car owners



(e) Bike, car owners



(f) Bike, non-car owners

Figure 15: Deterrence function per distance bin for the run with exogenous data for various mode-userclass pairs

6 Discussion, Recommendations and Conclusions

In this thesis a more robust alternative to the calibration method for the trip-based gravity model was developed. The proposed method considers the optimization problem of maximizing the entropy and solves it directly rather than applying the method of Lagrange multipliers which results in the feasibility problem that the trip-based gravity model solves. The relationship between the two formulations regarding the beta parameters of the deterrence functions was proven and afterwards used to validate the implementation of the SQP procedure on the models of Delft and Almere. Afterwards, the SQP method was compared to the trip-based gravity model by using the same data source to obtain the input for the model. Finally, the results for a deterrence function per distance bin was established, an extension to the deterrence function that is not possible when using the trip-based gravity model. This would make it possible to tailor the beta parameters even more and represent the behaviour of people's movements more accurately. The SQP model solved the Almere model for the purpose 'Work' in around 75 minutes and 100 minutes without and with distance bins respectively. If one were to solve the complete Almere model, i.e. all 7 purposes, this would take roughly 9 for the SQP model without distance bins and 12 hours for the SQP model with distance bins. Since the goal was to do so over night, the prototype meets this requirement for both cases under the assumption that the SQP model behaves similarly, or at least not worse, for other purposes. However, when the current implementation of the SQP model would be applied to more complicated models at DAT.Mobility, like most urban and regional regional models sitting around 4000 to 6000 zones, it will most likely run into memory issues. At 4000 zones the predicted required amount of RAM would be 193 GB, whereas current computers usually have at most 64 and possibly 128 GB of RAM.

Throughout this thesis, the two subjects that are mainly worth discussing are whether the budget constraint values are estimated correctly and why the runs with exogenous data, for both with and without distance bins, result in beta parameters that are different than one would expect.
First of all, let us discuss the differences between the runs using exogenous data. As one can tell from Table 6, the resulting beta parameters from the SQP model are not close to the values that are used in the trip-based gravity model. It is yet unclear why this is the case, but the model does work properly as can be seen from the validation runs and therefore the difference in parameters has its origin in the data that is fed into the SQP model. In this thesis we have discussed multiple times the issue of the new budget constraint values C_m being unknown and what procedure is used to estimate those values. However, even though we have tried to estimate these values as accurate as possible, it is very well possible that these values are not good enough. To give some intuition behind this, consider the values estimated by the SQP model in Table 6. For all mode-userclass pairs, the estimated beta value is way higher than those used for the trip based gravity model. Recall that a deterrence function with a high beta value results in a function that is less steep, i.e. the willingness to make trips with high generalized costs is still quite high. This might indicate that the value of the budget constraints is way to high and that a lot of people are allowed to make trips with high costs. Besides the new estimated budget constraint values, it could also be the case that other aspects of the data are unreliable. To name some examples, due to historical reasons, quite a lot of people in Almere have their city of work being Amsterdam because they lived there before and moved to Almere afterwards. Or the generalized costs are calculated too simplistically by by not taking into account the initial costs when travelling with the public transport for example. The beta parameters try to compensate for such flaws in the data input, resulting in these weird values.

Another possibility is that it could be caused by unreliable data in the data source. When we for example take a look at the deterrence functions per distance bin in Figure 15 we see that for certain distance bins, especially the bins 12.5-17.5km and 17.5-27.5km for car and public transit, have beta values that are nowhere close to the surrounding ones. Taking a deeper look into the OViN data, which is a database of survey data, learns us that at those distance bins there were way less observations than the distance bins before and after. Given the geographical situation of Almere and the fact that we are looking at the purpose 'Work', this does make sense since it lies around 30km away from Amsterdam, where a lot of people go to work. Since there is a lower number of observations, this might indicate that these values are unreliable and therefore could influence the results (in a negative way) by functioning as noise. Note that this problem is not caused by the lower number of expected attractions for trips in these distance bins, since the attractions do not depend on

the distance bin, but are is a general constraint for each zone in the model. One recommendation to research whether unreliability in this data is the cause of these weird beta values is to translate the unreliability in number of trips per distance bin to unreliability in generalized costs.

As stated in the introduction of Section 5, the FW method did not result in anything reasonable when using realistic data instances, whereas it did work as intended for synthetic data. Since the method did work for randomly generated instances, this indicates that it somehow can not handle the specific values in realistic data. This could for example be due to the first order and second order approximation of the objective function when compared to SQP. Further research should clarify whether this assumption is correct or if there is another difference between FW and SQP that causes FW to not work for the tested realistic instances.

A nice advantage of using FW or SQP is that since we require an initial solution, we need to solve the feasibility problem before iterating. This feasibility problem has the same constraints as the optimization problem and therefore we know whether a feasible solution exist even before the first iteration is done. This is completely the opposite to IPF in the trip-based gravity model where it is computationally impossible to determine beforehand whether IPF converges and therefore only knows whether it has converged after running the model.
One could possibly improve upon the feasibility problem that is used to determine the initial solution. Since the easiest initial solution to come up with includes a couple of large values and mostly zeros, this solution is nowhere near being realistic. Instead of only looking at the feasibility problem, one could add an objective value in order to start with a better initial solution and therefore decreasing the number of iterations and running time.
When the feasibility problem shows that there is no feasible problem, one should give the model more freedom. This can be done in multiple ways, for example by loosening some constraints, especially the ones that are the most unreliable. This was actually done for the exogenous runs of the model, rather than having the budget constraints be an equality, one could add some tolerance to it by stating that it may deviate 5% for example. Another option to give the model more freedom could for example be to aggregate zones, making the attraction and production constraints less tight.

References

[1] A.A. Ahmadi. Lecture 7 [lecture notes]. https://www.princeton.edu/~aaa/Public/Teaching/ORF523/S16/ORF523_S16_Lec7_gh.pdf, 2015. Accessed 7 October 2020.

[2] N.F. Stewart B. Lamond. Bregman's balancing method. *Transportation Research Part B: Methodological*, 15(4):239–248, 1981.

[3] J. Brethouwer. *The multi-constrained gravity model And how to solve it using multi-proportional fitting.* Report of internship at DAT.Mobility, 2018.

[4] CBS. Onderzoek verplaatsingen in nederland (ovin). https://www.cbs.nl/nl-nl/onze-diensten/methoden/onderzoeksomschrijvingen/korte-onderzoeksbeschrijvingen/onderzoek-verplaatsingen-in-nederland--ovin--, 2017.

[5] Joydeep Dutta and CS Lalitha. Optimality conditions in convex optimization revisited. *Optimization Letters*, 7(2):221–229, 2013.

[6] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[7] Geoff Gordon and Ryan Tibshirani. Karush-kuhn-tucker conditions. *Optimization*, 10(725/36):725, 2012.

[8] H.W. Hoppe and Christopher Linsenmann. Optimization theory chapter 4: Sequential quadratic programming. https://www.math.uh.edu/~rohop/fall_06/index.html, 2006. Accessed: 13 July 2020.

[9] Angelia Nedich. Constrained nonlinear problems necessary kkt optimality conditions [lecture slides]. http://www.ifp.illinois.edu/~angelia/ge330fall09_nlpkkt_l26.pdf, 2009. Accessed 8 July 2020.

[10] J. D. Ortúzar and L.G. Willumsen. *Modelling Transport.* John Wiley & Sons, Ltd, third edition edition, 2001.

[11] M. Pots. *Gravity model parameter calibration for large scale strategic transport models.* Report of internship at DAT.Mobility, 2018.

[12] Friedrich Pukelsheim and Bruno Simeone. On the iterative proportional fitting procedure : Structure of accumulation points and l 1-error analysis. 2009.

[13] U. Rothblum and H. Schneider. Scalings of matrices which have prespecified row sums and column sums via optimization. *Linear Algebra and its Applications*, pages 737–764, 1989.

[14] Arnoldo C. Hax Stephen P. Bradley and Thomas L. Magnanti. *Applied Mathematical Programming.* Addison-Wesley Publishing Company, 1977.

[15] F.J. Willekens. Entropy, multiproportional adjustment and the analysis of contingency tables. *Systemi Urbani*, 2:171–201, 1980.

[16] A. G. Wilson. The use of entropy maximising models, in the theory of trip distribution, mode split and route split. *Journal of Transport Economics and Policy*, 3(1):108–126, 1969.