KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Stochastic User Equilibrium with Full Route Set in Dynamic Traffic Assignment

Jeroen Verstraete

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: verkeer, logistiek en intelligente transportsystemen

Promotor:
Prof. Dr. Ir. C. M.J. Tampère

Assessor:
Prof. Dr. S. Proost

Begeleider:
Dr. Ir. W. Himpe

Academiejaar 2016 – 2017

Preface

This thesis is my final work of my studies at KU Leuven. While struggling the first year to find my way, the result and the whole course of my studying career is something I feel proud of. With its ups and downs, I would not have been here if I was all by myself.

First of all, I thank my parents who gave me the opportunity to complete these studies. Even after a rough start, they kept believing in me, which was certainly an extra motivator.

In the last year of the bachelor, I met prof. Vansteenwegen who made me even more interested in the master VLITS (Logistics and Traffic). But I really found my interest in the topics prof. Tampère lectured. My interest combined with his enthusiasm made me a studious, motivated student. I would like to thank him for taking the time to guide me in this thesis, but also to share his enthusiasm with the students in the lectures and the opportunity he gave me to do a student job at the research center.

At this student job, I met Dr. Willem Himpe, who is the assistant of this thesis. He helped me with every problem and putted new research topics on the table. I would like to thank him for guiding me in this thesis and accompanying me on my first conference.

After a day of hard work/study, I could always count on my friends (at university or at home) to release steam. I also want to thank Elisa, my girlfriend, to be on my side this whole journey and to come with me to the distant city of Leuven.

I hope you, the reader, enjoy reading this thesis as much as I had while making it.

Jeroen Verstraete

Contents

Abstract

Equilibrium algorithms distribute traffic in a network taking into account the congestion it induces. Typically this leads to two stages in the methods for finding an equilibrium solution. The first stage handles the route set generation and the second distributes the traffic over route alternatives. In stochastic equilibrium route choice models the most important problem is defining a consistent route set.

In current practice, there are two main concepts to handle route sets. Either to maintain a fixed route set in the distribution phase or to update route set during the distribution phase. Both approaches have shown their limitations. In case of a fixed route set, there is a chance that an important route is not considered. This might happen if unforeseen congestion is present in the network. To solve this problem, the second approach develops methods with a flexible route set during the equilibrium procedure. However, this may negatively affect the convergence. In this thesis a method is developed that solves these issues.

It is proposed to implicitly consider all possible routes such that the route set does not require updating during the distribution phase and no routes are missing from the solution. Recursive Logit (RL) is an implicit route choice model that considers all routes, hence the route set does not depend on congestion levels, neither does it change over iterations. RL calculates the turning percentages from each node to each destination separately. From these percentages, destination-based flows can easily be calculated (in a static assignment).

This thesis first illustrates how recursive logit converges very stable in a static stochastic user equilibrium. The main contribution of this thesis is a method to implement a full route choice with recursive logit in a dynamic assignment. Another contribution is the identification of a few important parameters that determine the costs for a traveller. Other parameters can easily be added. Finally, the relation between these parameters and finding a solution is formulated.

Samenvatting

Een verkeersmodel verdeelt het verkeer over een netwerk. Om te weten welke routes het verkeer zal nemen richting hun bestemming, is het belangrijk om een goede routeset te beschouwen in een model. Een routeset is de set van alle routes (tussen alle HB-paren) die het model beschouwt. Momenteel zijn er twee gebruikelijke methoden om om te gaan met de routeset in een evenwicht routekeuzemodel. Als eerste kan de routeset vast zijn gedurende de iteraties voor elke HB paar. Als tweede kan de routeset tijdens het uitvoeren berekend worden (iedere iteratie of minder frequent, afhankelijk van de huidige congestie in het netwerk) en dus flexibel zijn. Beide methodes hebben hun problemen. Zo is er bij een vaste routeset kans dat een relevante route niet in de routeset zit, dit verslechterd uiteraard de kwaliteit van de toedeling. Terwijl een flexibele routeset de convergentie verslecht door de toevoeging van extra compliciteit. Deze extra compliciteit komt dat buiten de stroom ook de routeset moet convergeren. Er is dus nood aan een andere aanpak.

Recursive Logit (RL) kan hier een oplossing voor bieden. RL is namelijk een impliciet routekeuzemodel dat altijd alle routes in overweging neemt. Deze routeset verandert dus niet tijdens de iteraties. RL berekent de kansen van iedere afslag voor iedere knooppunt voor iedere bestemming. De kansen van een afslag zullen afhangen van de reistijd op de link, als van eigenschappen van de afslag (bv. links afslaan) en van de verwachte utiliteit (met onder andere de reistijd) tot de bestemming op het einde van de link. RL vertrekt van de definities gegeven door Dial over utiliteit. Een groot verschil is dat door alle mogelijke routes mee te nemen, de verwachte utiliteit op het einde van een link tot de bestemming kan afhangen van zich zelf door de mogelijkheden van lussen. Van de kansen van deze afslagen kan eenvoudig een bestemming gebaseerde hoeveelheid verkeer berekend worden (in een statische toedeling). Door het gebruiken van alle routes, komen soms routes naar voor die minder relevant zijn, bijvoorbeeld routes met een lus of zelfs tien keer dezelfde lus.

Deze thesis zal eerst illustreren hoe stabiel RL convergeert in een statische toedeling. De grootste bijdrage van deze thesis is het ontwikkelen van een methode dat een volledige route keuzeset van RL toepast in een dynamische toedeling. Ook hier zal de stabiliteit besproken worden. Een andere bijdrage is het identificeren van belangrijke eigenschappen van een afslag dat de utiliteit van een reiziger kan bepalen. Andere eigenschappen kunnen makkelijk toegevoegd worden. Als laatste is een verband tussen de parameters van deze eigenschappen en het vinden van een oplossing geformuleerd. De ontwikkelde methode kan ook gebruikt worden in andere contexten, bijvoorbeeld bij het genereren of beoordelen van routesets.

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations

DNL	Dynamic Network Loading
DTA	Dynamic Traffic Assignment
I-LTM	Intelligent Link Transmission Model
LTM	Link Transmission Model
MSA	Method of Successive Averages
OD	Origin-Destination
RL	Recursive Logit

Introduction

"Belgium is one of the leaders in traffic jams."

Every once in a while this quote is printed on every journal. Besides the weather, traffic is for some people the most favourite topic to complain about. Almost everyone has to deal with traffic on a daily basis and some therefore claim to have *'the solution'* to *'the traffic problem'*. This *'solution'* is not only heard in bars after some drinks, but appear also in newspapers by traffic experts, or so they introduce themselves.

The government needs a way to known the impact of these *'solutions'*. This is done by analysing all sorts of data, but unfortunately not all data is known. Luckily for the government, researchers have developed a way to predict (interpolate or extrapolate) missing data. Traffic can be modelled by using mathematical formulations. The term traffic model is very broad. It can go from short term (tactical level) predictions to more long term predictions (strategic level). Some assume each person to act the same way, while others simulate each person differently. The model gives some choices to the simulated users. Examples of these choices are for example what transport mode a user will use, which route he will take to his destination or even when he will leave his house.

This thesis will focus on defining the possible answers of one of these choices given to the users, namely to the question which route the users take. Which route a user can take is limited to the routes a model considers. Most models consider only a subset of all the possible routes. This seams logical because who would want to take a route that makes a serious detour. However, with congestion on the network, analysing the travel time of each route can reveal that a route with a serious detour in free flow conditions (no congestion) can be the shortest with congestion on the network. This unknown delay (due to congestion) before the model runs, makes it hard to conclude which route set should be used. When the route set contains all possible routes, the route a certain traveller prefers is per definition in the route set. A problem is that really every route is in the route set, also the routes with one loop or the routes with ten times the same loop. This thesis will show a way of how to cope with this property.

While answering the question which route the users will take, the method takes into account that users can have their own often unknown preferences or individual perception of the network conditions. This means that a stochastic user equilibrium needs to be solved to take into account the uncertainty of the choice process. In the extreme case that all users have the same (known) preferences, the method can also solve the deterministic user equilibrium.

A deterministic user equilibrium is a special case of a stochastic user equilibrium with a variance of zero, this means that all travellers experience the same benefits/costs. A stochastic user equilibrium is much more realistic then a deterministic one and will behave more predictable to a change in the network, it is also perfect observable.

This thesis will use a theoretical described method (based on Recursive Logit) to take a full route set into account and will be implemented in a static and dynamic assignment. It will show that working with a fixed (full) route set is much more stable, because the a full route set reduces the complexity. A less complex problem is very useful, this way the method can be implemented in any real time system. The method can also be used to help other algorithms decide which routes should be included in the route set or to evaluate a given route set.

An implicit full route choice model has been theoretical described for a static assignment, this thesis will first implement it. The real research will be finding a method that does the same in a dynamic assignment. This will be done in such way it is described as general as possible, which means it can be easily adopted and changed if needed.

After recursive logit is introduced (chapter 2), it is implemented in a static assignment (chapter 3). When implementing it, focus is given to parameters and network characteristics that determine the utility a traveller experience. This thesis shows how all destinations can be handled together in a static assignment, rather than one by one. The positive features of the static assignment gives enough reasons to research if it can also be implemented in a dynamic assignment. This is done in chapter 4, where a method is developed to show how recursive logit can be of use in a dynamic assignment. Chapter 5 gives more examples on how to use the algorithm and in which cases it can help. Chapter 6 then concludes with some topics for further research.

Chapter 1

Route sets

In every traffic assignment model, a route set needs to be chosen. A traffic assignment model distributes traffic over a network taking the effects of congestion on certain roads into account. A route set is a collection of all plausible routes between an origin-destination (OD) pair that the algorithm considers. Ideally, the route set should contain every plausible route. Under a fixed route set, a converged equilibrium is the solution of a convex combination of route fractions, a problem easily solved by generalized techniques. The problem of finding a consistent route set is, on the other hand, more difficult and could potentially deteriorate convergence to an equilibrium solution. Considering all routes in the network for every iteration towards convergence might solve the latter problem.

There are different classifications of route sets. Two of these classifications are mentioned below. First, the difference between an explicit and an implicit route set is handled. The other classification is between fixed and flexible route sets. In the last section of this chapter is explained what an implicit fixed route set is and why a full fixed implicit route set is used in this thesis.

1.1 Classification of route sets

Let us consider a stochastic user equilibrium (including the extreme case of approaching zero variance for a deterministic solution), some algorithms work with an explicit route set, this means that every route considered is explicitly stated. Enumerating all routes in a network with loops is impossible, an explicit route set can not contain all routes in any real network. Some subset of all possible routes should be chosen. Not including a relevant route in the subset has a negative impact on the quality of the assignment. The opposite of explicit is implicit, an implicit route set does not enumerate paths but defines the turns that are plausible. This can be all turns (as will be done in this research) or only a subset of possible turns (like Dial's algorithm proposed in Dial and Voorhees (1971)). In this latter case, the topological order determines which turns are plausible. Only turns that connect a lower to a higher node in the topological order are considered. The route set is different if the topological order is changed. The topological order being determined by the distance of the shortest path between a node and the destination.

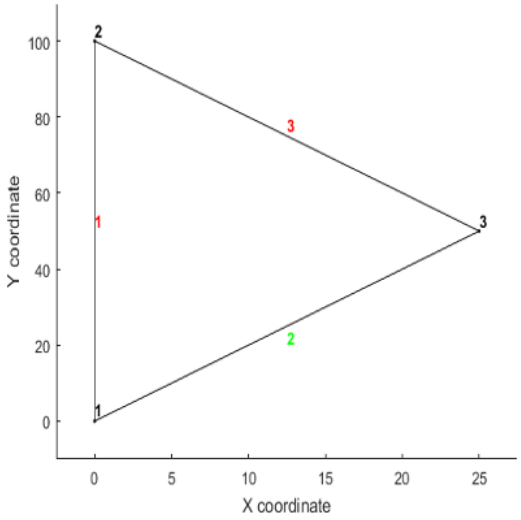


Figure 1.1: A simple network. (Black numbers represent the node numbers, coloured numbers the link number.)

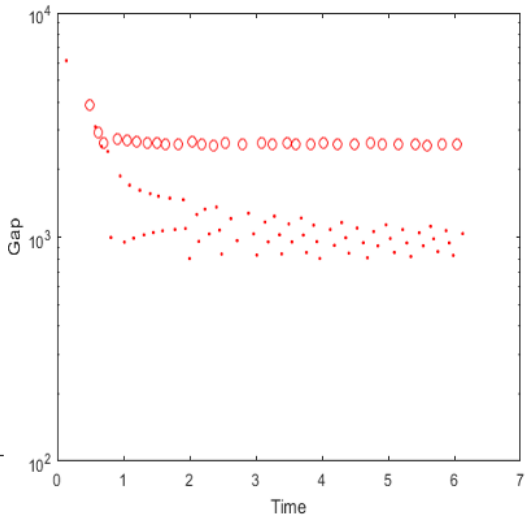


Figure 1.2: Convergence of the Dial algorithm, a 'o' indicates only one route is in the route set, while a '.' indicates both routes are in the set.

There are two concepts of how to deal with a route set during the iterations of an equilibration algorithm. First, a fixed route set can be used. A fixed route set is a route set that is determined at the beginning of the algorithm. Working with a fixed route set gives advantages for computing time and smoothness of convergence, but increases the chance of missing a relevant route. Second, the route set can also be flexible. During the execution of the algorithm, the flexible route set changes, the route set can change every iteration or only when the algorithm thinks it is necessary, similar to column generation techniques for combinatorial problems. A fixed size can be used for the route set, for example only 5 paths, or the size can be variable due to a variety of criteria. An example of such criteria is the topological order. Frejinger et al. (2009) has an average choice set size of 9.66, which is constructed by a bias random walk. L Bovy (2009) uses a branch and bound method to determine the route set.

Let us illustrate by a small example the convergence problems that can arise when a flexible route set is used. Consider the network in figure 1.1, the only demand is from node *1* to node *2*. There are two possible routes (note that the links are unidirectional), namely directly from node *1* to node *2* by using link *1* or by using link *2* and *3*. By using high demand and non-constant cost functions on the links, the topological order of the nodes changes while Dial's algorithm executes. The resulting poor convergence is plotted in figure 1.2. How the gap is calculated, will be explained later. Important now is that a gap is a way to quantify the quality of the solution. The lower the gap, the closer to the equilibrium.

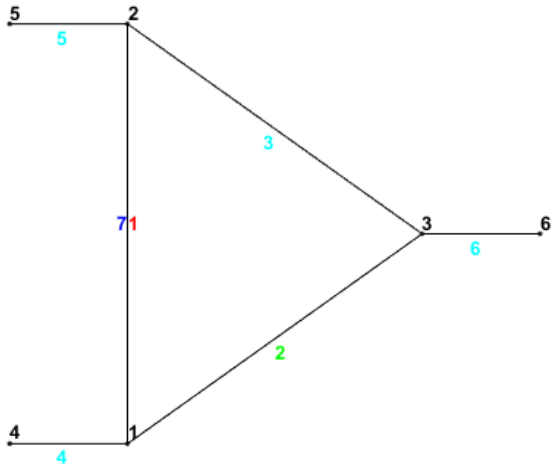


Figure 1.3: A symmetric triangle network

Besides the less smooth convergence of the algorithm, a flexible route set defined by the topological order misses some relevant routes. The next example shows an extreme case where two routes, who have the same cost (no congestion on the network), can not be both in the route set. Figure 1.3 illustrates this network. Both origins (nodes *4* and *5*) have two possible routes to the destination (node *6*). While having the same costs, the paths (defined by the links) *5,7,2,6* and *4,1,3,6* can not be both in the route set that is constructed by a topological order. The problem here is that or node *2* is higher than node *3* in the topological order or the other way around. This means that only one of the links *1* or *7* can be an effective link. Note that in a deterministic user equilibrium this problem does not arise, because only the shortest paths would be used. With a full route set, all four routes would always be considered.

1.2 A Full Fixed Implicit route set

A solution to these problems is working with a full route set, which considers all possible routes in each iteration. Due to the fact that in many networks, an infinite number of routes exists, the route set will be implicit. This means that not every route in the route set is explicitly written. Bell (1995) formulated alternatives to Dial's logit assignment algorithm, this inspired Fosgerau et al. (2013) to formulate the recursive logit.

The idea of Bell is simple, let W be a matrix of weights constructed as follows:

$$w_{n,m} = exp(-\alpha * Cost_{n,m}) \tag{1.1}$$

With n and m being links and α the cost parameter.

Then W expresses the weights between each pair of two links directly (with a weight of 0 if the path doesn't exist). Bell shows then that W^2 expresses the combined weights of all the routes between each pair of links consisting of exactly 2 links. The combined weights of all routes between each pair of links consisting of any number of links is

$$W + W^2 + W^3 + \ldots = (I - W)^{-1} - I = Y \tag{1.2}$$

This can only be calculated if (I-W) can be inverted, thus not every network (with its specific parameters) can have such combined matrix of weights. The probability that a link *k* (connecting node *r* to node *s*) is used for an OD-pair i to j can be calculated by:

$$P_{ijk} = y_{ir} * exp\left(-\alpha * C_k * \frac{y_{sj}}{y_{ij}}\right) \tag{1.3}$$

With C_k the cost on link *k*. The path choice is thus reduced to a sequential link choice model.

Fosgerau more explicitly states the conditional probability for a traveller *n* going to a destination *d* will use link *k* given the traveller is on link *a*. (For formulas, see later.) This probability depends on the utility for going from link *a* to *k* (this can include a random utility component ϵ_n) and the expected downstream utility. This leads to a system of equations that under certain conditions can be solved. This recursive logit (RL) will be handled in more detail in the next chapter.

It is with this full fixed implicit route set, that always considers all alternatives to a route, that this research hopes to show a way to resolve rerouteing problems. From now on if a route set is mentioned, a full fixed implicit route set is intended if not specified further.

Chapter 2

Recursive Logit

This chapter is strongly based on the work of Fosgerau et al. (2013). Some details are explained with another view, to better understand the steps the research took afterwards.

For convenience, the same notation is used as in the paper of Fosgerau et al. (2013). See figure 2.1 for the visualisation of the notation. While *a* and *k* are links, *d* is the destination link (a connector). Connectors are links without a sink node (destination) or source node (origin). Connectors are the only place traffic can appear or disappear. A(k) is the set of outgoing links from the sink node of link *k*. Another way to look at A(k) is a set of possible turns one can make at the end of link *k*. Turns are fully described with the two links connected by that turn.

The recursive logit states the conditional probability that a certain turn will be taken to a destination, given the traveller is at the end of a link. A matrix P is formed which collects all the probabilities towards a given destination. This matrix will have dimensions [(l+2c) * (l+2c)], with *l* the number of links and *c* the number of connectors. P_{ij} gives then the probability of going to link (or connector) *j* given travellers location on link (or connector) *i*. Note that links are defined in only one way, making a two-way road modelled as two separate links. Given these probabilities, a path probability can be calculated by multiplying every turn probability along the path. For example the probability that traveller *n* uses path $1 \rightarrow 3 \rightarrow 4$ is calculated as: $P_{1,3} * P_{3,4}$.

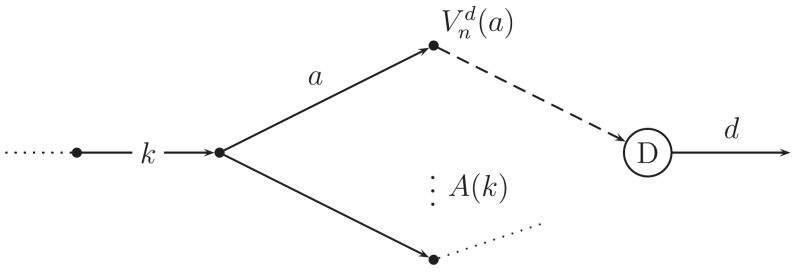


Figure 2.1: Illustration of notation (Fosgerau et al., 2013)

The question now remains on how to calculate such matrix P. If a traveller *n*, on link *k* needs to chose which turn to take to destination *d*, he compares the (instantaneous) utilities for each turn with their expected downstream utility ($V_n^d(a)$, the last term in equation 2.1). The utility is the benefit the users experience when travelling the link, with a negative utility being a cost. The (instantaneous) utility consists of two parts, a part that for every traveller is the same $v_n(a|k)$ and a part that is unknown and different for every traveller $\mu\varepsilon_n(a)$. The first part consists of the travel time on link *k* and possible turn impedances. To use the properties of logit, the random term $\varepsilon_n(a)$ is assumed independent and identically distributed extreme value type 1 with a mean of zero. The expected utility from the traveller on link *k* is the maximum sum of these utilities, namely:

$$V_n^d(k) = E\left[\max_{a\in A(k)}\left(v_n(a|k) + \mu\varepsilon_n(a) + V_n^d(a)\right)\right] \tag{2.1}$$

By using a multinomial logit model, the probability that a link will be used for a route to destination *d*, given link *k*, is:

$$P_n^d(a|k) = \frac{\exp\left(\frac{1}{\mu}\left(v_n(a|k) + V_n^d(a)\right)\right)}{\sum_{a'\in A}\left(\exp\left(\frac{1}{\mu}\left(v_n(a'|k) + V_n^d(a')\right)\right)\right)} \tag{2.2}$$

The Expected Maximum Perceived Utility function in a logit model is the logsum over all possible choices that can be made. The expected downstream utility ($V_n^d(k)$) can then be written as follows:

$$V_n^d(k) = \begin{cases} \mu \ln \sum_{a\in A}\left(\delta(a|k)\exp\left(\frac{1}{\mu}\left(v_n(a|k) + V_n^d(a)\right)\right)\right) & \forall k \in A \setminus d \\ 0 & k = d \end{cases} \tag{2.3}$$

With $\delta(a|k)$ equal to one if $a \in A(k)$ and zero otherwise. If link *k* is the destination, the utility is zero. This way of writing the utilities is the same as in the paper of Dial and Voorhees (1971).

There is a huge different between RL and the definition of Dial. Dial and Voorhees (1971) introduced a topological order, in this way $V_n^d(k)$ depends on the value of $V_n^d(a)$ (as expressed in equation 2.3) but $V_n^d(a)$ does not depend on the value of $V_n^d(k)$. When this is the case, computing the downstream utilities is just using the equations in the right order. When a topological order is used, the node order is determined by the shortest path and the node the closest to the destination is calculated first. The other nodes can be calculated by going over all nodes according to the topological order.

Things change if the network contains a loop, then the value for the utility on a link will depend on others and its own value. If equation 2.3 is written for every link, then a system of non-linear equations is obtained. To make a linear system of equations, the variables needs to be transformed.

If from both sides of the equation the exponential is taken and raised to the power of $\frac{1}{\mu}$, then this is the result:

$$\exp\left(\frac{1}{\mu}V_n^d(k)\right) = \begin{cases} \sum_{a\in A}\left(\delta(a|k)\exp\left(\frac{1}{\mu}\left(v_n(a|k)+V_n^d(a)\right)\right)\right) & \forall k \in A \setminus d \\ 1 & k = d \end{cases} \tag{2.4}$$

To write the system of equations in matrix notation, three matrices are introduced:
b ($|A| \times 1$), **z** ($|A| \times 1$) and **M** ($|A| \times |A|$). With **b** being a vector with $b_k = 0$ for $k \neq d$ and $b_d = 1$, **z** a vector that gives the destination dependent transformed utility of the end node of a link, namely $z_k = \exp\left(\frac{1}{\mu}V_n^d(k)\right)$. Note that z_k is the same for each link sharing the same end node. M is a matrix of utilities for each turn, this is destination independent.

$$M_{ka} = \delta(a|k)\exp\left(\frac{1}{\mu}v_n(a|k)\right) \tag{2.5}$$

The component $v_n(a|k)$ consists of turn dependent utilities but also the travel time on link a. The system of equations can now be formulated as followed:

$$\mathbf{z} = \mathbf{Mz} + \mathbf{b} \Longleftrightarrow (\mathbf{I} - \mathbf{M})\,\mathbf{z} = \mathbf{b} \tag{2.6}$$

Where **I** is the identity matrix. It is clear that the system only has a solution if **I**-**M** is invertible. This will be handled later. (See section5.1)

Probabilities can now be calculated:

$$P_k = \frac{M_k \bullet z^T}{M_k z} \tag{2.7}$$

With $\bullet$ the element by element multiplier. The P matrix is thus calculated row per row, where a row represents the given state (link). M_k is the corresponding row of matrix M. With these probabilities, Fosgerau et al. (2013) gives another system of equations to solve the link flows (towards one destination).

$$\left(I - P^T\right)F = G \tag{2.8}$$

With F a vector ($|A| \times 1$) the link flows (towards one destination), G a vector ($|A| \times 1$) with G_k = demand from link (connector) *k* to destination *d*.

There is a matrix P for every destination, as recursive logit is destination based. There are no differences in probabilities when the traveller had begun his journey from another origin.

Chapter 3

Full route set in a Static assignment

To fully understand the concept of the Recursive Logit, it was implemented in a static assignment first. Because the dynamic assignment's first step will be a static assignment, it is useful to elaborate this a little.

A static traffic assignment, is an assignment that has no time dimension. Given the demand, it calculates how many people will use each link.

3.1 The algorithm

The recursive logit part in a static assignment takes place in the inner loop part of the algorithm. Given the costs of the links (travel times), the inner loop calculates the corresponding flows. Outside of this loop, another loop tries to achieve equilibrium conditions. The costs that are the input of the inner loop are here updated and smoothed given the flows. Figure 3.1 gives the steps of the algorithm. The algorithm is implemented in Matlab R2016b. Section 3.1.3 gives an overview of how the algorithm is coded in Matlab.

3.1.1 Recursive Logit

The inner loop consists of the recursive logit part. As explained above, all calculations are destination based and for a whole OD-matrix, a loop is introduced. To reduce calculations, the transformed utilities for each link towards a destination (the Z_k elements) are now calculated in a matrix Z, where Z_{kd} is the transformed downstream utility for link k to destination d. One column of Z is a previously calculated vector z. Equation 2.6 is then formulated as:

$$\mathbf{Z} = \mathbf{MZ} + \mathbf{B} \Longleftrightarrow (\mathbf{I} - \mathbf{M})\,\mathbf{Z} = \mathbf{B} \tag{3.1}$$

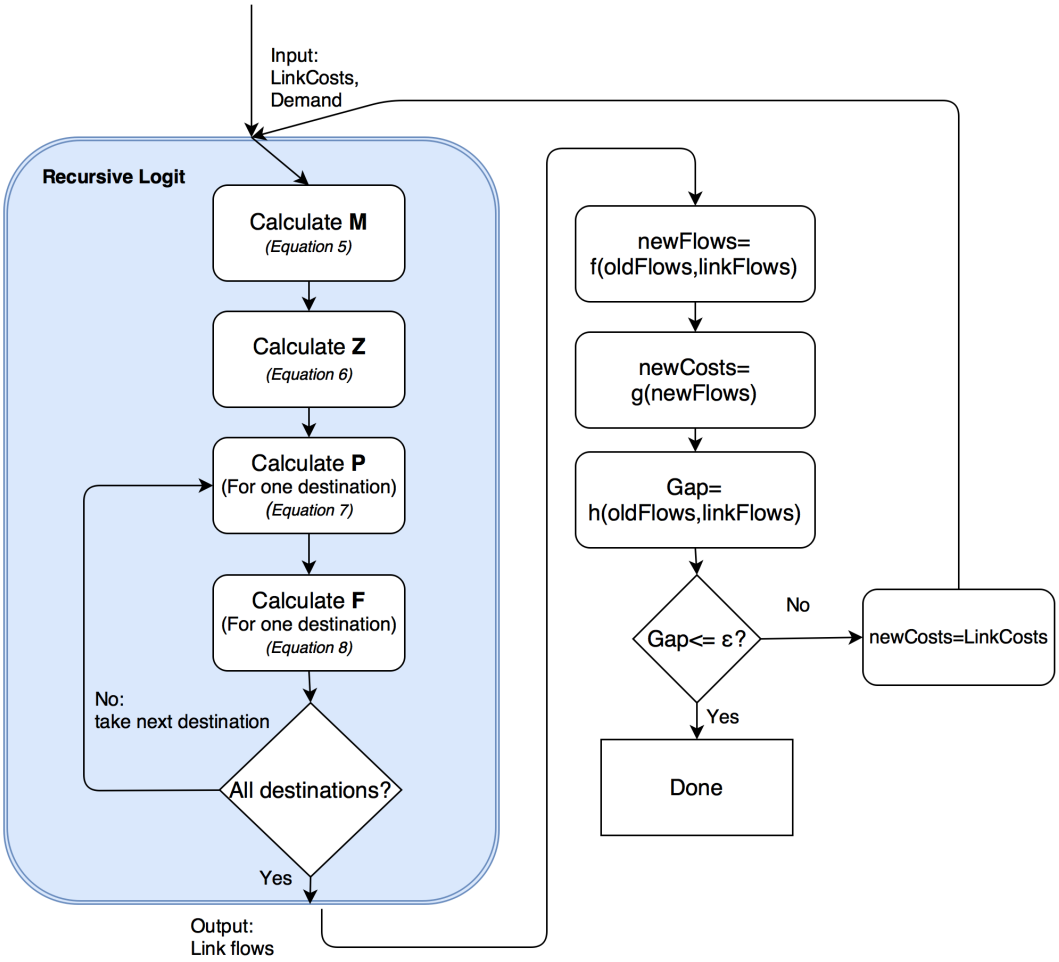


Figure 3.1: Overview of the algorithm used in the static assignment

The matrix Z has as dimension $(|A| \times |D|)$, where $|D|$ is the total number of destinations. M and I remain the same. B $(|A| \times |D|)$ is the destination matrix, with in every column one element that equals 1. If all the links are split in: real links, origin connectors and destination connectors, then B typically has the following layout:

$$\mathbf{B} = \begin{array}{c} \\ RealLinks \\ OriginConnectors \\ DestinationConnectors \end{array} \begin{array}{c} DestinationConnectors \\ \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{I} \end{bmatrix} \end{array}$$

With these adjustments, all transformed utilities for each link to each destination are calculated in one system of equations. The probabilities and link flows can only be calculated for each destination. Link flows are added together to get the total link flows (independent of the destination), this is the output of the Recursive Logit part.

3.1.2 Equilibrium

To find the solution of the static assignment, the equilibrium needs to be calculated. To do this, three functions needs to be further defined. The first functions define what the new link flows will be, will it be the direct outcome from the recursive logit or will an MSA step (Robbins and Monro, 1951) or any other smoothing of the step size be introduced. The second function needs to calculate the new link costs with the new link flows. The last function needs to calculate a 'gap' or improvement with the last iteration. This gap will then be used as a stopping criterion. Besides a gap, a maximum number of iterations can also be used to be certain the loop will end. These three functions will be handled in the next three paragraphs.

Once a new iteration is calculated, the overall solution takes a step in the new calculated direction. How big this step is, depends on the algorithm used. A much-used algorithm to determine the step size is MSA, where the step size is the inverse of the iteration number. This means that the influence of the new calculated iteration decreases. In other words, the amount of traffic that can change routes decreases. The fact that recursive logit works with an implicit full route set, makes the algorithm more stable. This is logical because the route set can no longer change over iterations (as is possible in Dial's algorithm). Instead of an MSA step, a proportional step size is used. A proportional step size gives a fixed weight to the newly calculated average between the previous (averaged) and new calculated results. This way new calculations have a larger impact on the new averaged result. In this research, a proportional step size of 0.5 is used. This means that the new flows are the average of the previous flows and newly calculated flows. An example of a smooth convergence of recursive logit is found in figure 3.2. The value of 0.5 is randomly selected. Section 3.2 goes into more detail about this and tries to define the impact of different step sizes.

In this static assignment, a simple BPR function is used to determine the costs on each link given the flows. A BPR function is of the following form:

$$TT_a = TT_a^{FF}\left(1 + \alpha\left(\frac{v_a}{c_a}\right)^{\beta}\right) \tag{3.2}$$

With TT_a the travel time on link a, TT_a^{FF} the free flow travel time on link a, v_a the volume on link a and c_a the capacity of link a. α and β are function parameters. In this thesis the values 0.15 for α and 4 for β is used, as suggested by the Bureau of Public Roads (BPR).

For the gap function, the sum of the absolute values of the difference between the old link flows and the link flows from the recursive logit is taken. ε is set at 10^{-3} (with a maximum number of iterations of 2000). This means that when the new outcome of flows only make a maximum of 10^{-3} vehicles reroute, the algorithm takes the new link flows as the equilibrium.

3.1.3 Pseudo code

As said before, the algorithm is split into two functions, the equilibrium function and the recursive logit function (algorithm 3.1). The latter being the blue part of figure 3.1, while the first being the other parts around the blue part.

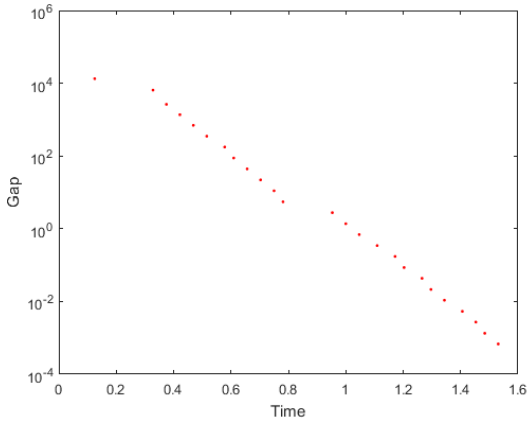


Figure 3.2: Convergence of recursive logit with proportional update

To reduce calculations, the M matrix is divided in different parts.

$$\mathbf{M} = \begin{array}{c} \\ Links \\ Origins \\ Destinations \end{array} \begin{array}{c} \begin{array}{ccc} Links & Origins & Destinations \end{array} \\ \begin{bmatrix} LL & LO & LD \\ OL & OO & OD \\ DL & DO & DD \end{bmatrix} \end{array}$$

Only the parts LL and OL change during the iterations due to the travel time. All other parts are either zero or one.

Algorithm 3.1 Recursive Logit Equilibrium in Static Assignment

function STATIC EQUILIBRIUM(Network,Demand,betas)
    Calculate all characteristics ▷ Including free flow travel times
    Calculate M ▷ With free flow travel times
    Calculate B
    while it<maxIt and gap<gapTreshold **do**
        $it \leftarrow it + 1$
        $newFlows \leftarrow RecLog(M, Demand, betas, travelTimes)$
        $gap \leftarrow Max(abs(Flows - newFlows))$ ▷ Max difference on a link
        $Flows \leftarrow update(Flows, newFlows)$ ▷ Depends on step size
        $travelTimes \leftarrow BPR(Flows)$
    end while
end function

Algorithm 3.2 Recursive Logit

function RECLOG(M,Demand,betas,travelTimes)
    Update M with travelTimes ▷ Only LL and OL
    $Z \leftarrow (I - M) \backslash B$
    $flows \leftarrow zero$
    for all destinations **do**
        for all links and origins **do**
            $P_k = \frac{M_k \bullet z_d^T}{M_k z_d}$
        end for
        $F \leftarrow \left(I - P^T\right) \backslash Demand_d$
        $flows \leftarrow flows + F$
    end for
end function

3.2 Influence of the step size

To see the impact of the different step sizes, a static assignment, with everything else in the network the same, is calculated for different step sizes. The same network as provided in chapter 1 is used. This network is plotted again in figure 3.3. The demand is 3000 people from node *1* to node *2*. The algorithm continues till the gap is smaller than 10^{-10}, this is smaller then before to better show the different convergence between the different step sizes.

Figure 3.4 show the four different convergence plots. First, a traditional MSA-step is used. The convergence criterion is not reached after 2500 iterations. The update is limited to the step size of one divided by the iteration number, which is very small after a lot of iterations.

Second, different proportional step sizes are tested. The values used are 0.1 and 0.5 and even a full step size of one. This latter means that the last calculated flows are directly, without any smoothing, used for the calculation of the costs. The proportional step size of 0.1 needs 240 iterations, while the proportional step size of 0.5 only needed 38 iterations. The full step size was in this example still stable and needed only 10 iterations. Note that because everything else is the same in the static assignment, each iteration needs more of less the same amount of time. This means that the full step size is 24 times faster than the proportional step size of 0.1 if the initialisation time is ignored.

Figure 3.5 gives the different convergence plots in one figure. The proportional update produces a linear convergence (on a plot with the y-axis logarithm). The different between each proportional update can easily be seen, how larger the proportional step size, how steeper the convergence.

The fact that there is only one destination makes finding the equilibrium much easier. The test network gives a first impression of what the maximum gain in changing the step size can be.

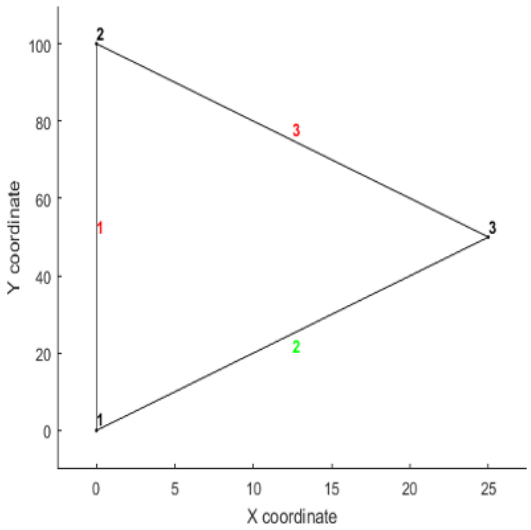
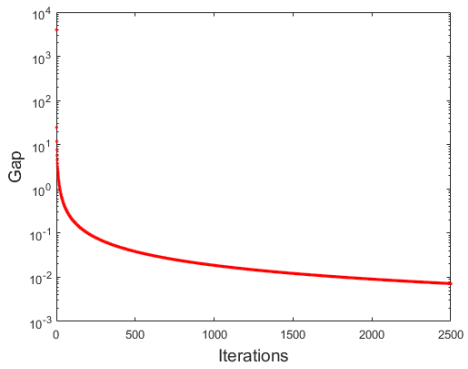
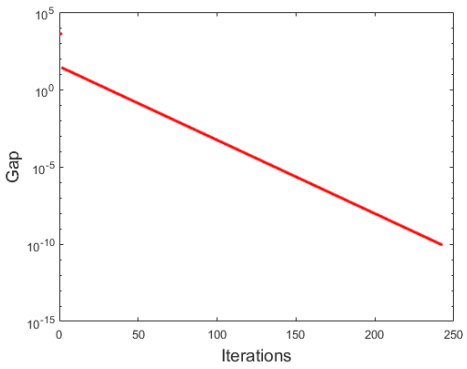


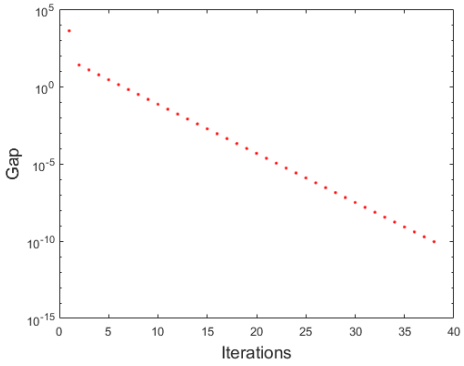
Figure 3.3: Test network



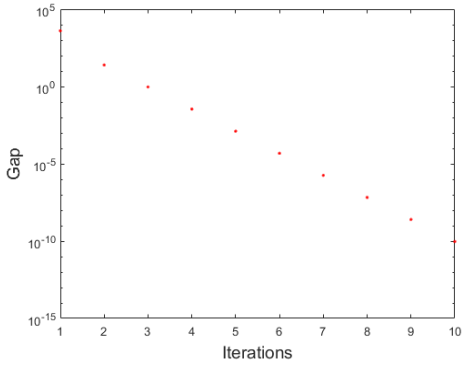
(a) MSA-step



(b) Proportional update of 0.1



(c) Proportional update of 0.5



(d) Proportional update of 1

Figure 3.4: Convergence of different step sizes

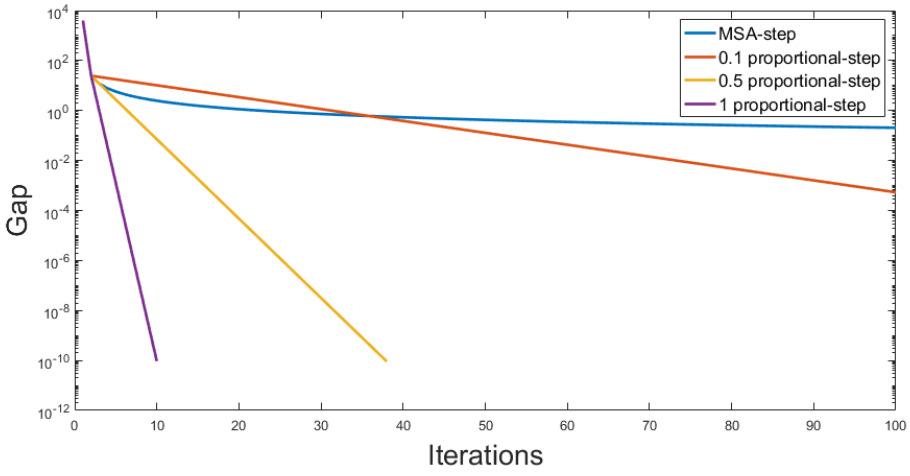


Figure 3.5: Network 1 convergences with different step sizes

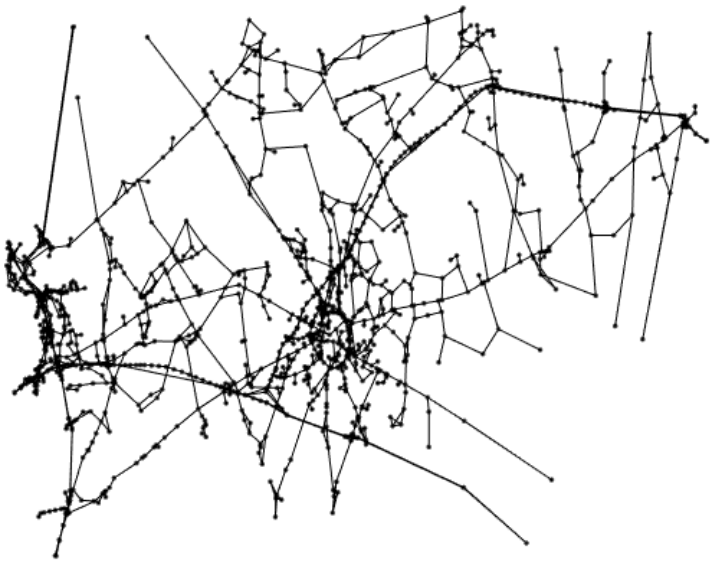
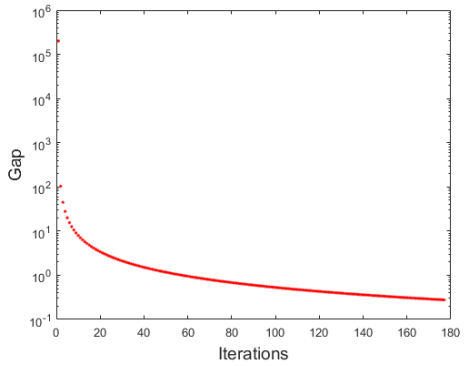


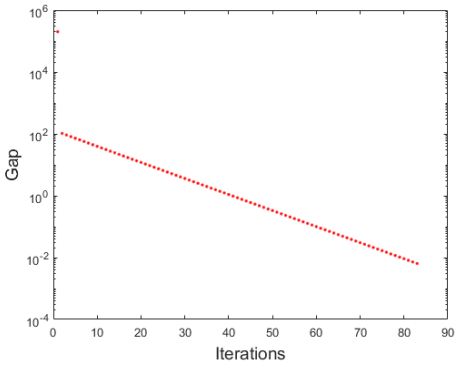
Figure 3.6: Network of Leuven

To test this further, a network of Leuven is taken (see figure 3.6). This network consists of 1493 nodes and 3203 links, with 210 different destinations from 211 origins.

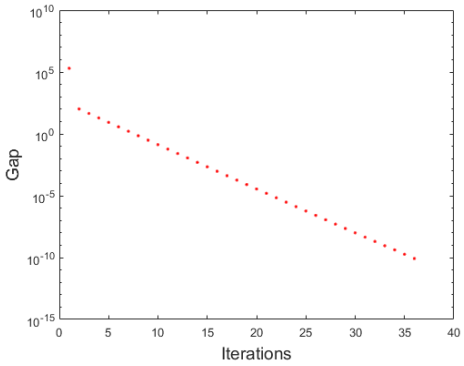
Figure 3.7 gives the convergence of the different step sizes separately, while figure 3.8 shows them in one figure. The same conclusions can be made as before. It does not appear that the algorithm can flip-flop between two states when a full step size is used. An other view on this is the following: during the calculation of the equilibrium, the input (travel times) and output (link flows) are never the same as in a previous iteration. The output at iteration k is different from all *(k-1)* previous iterations. Further tests are needed to see if this is always the case.



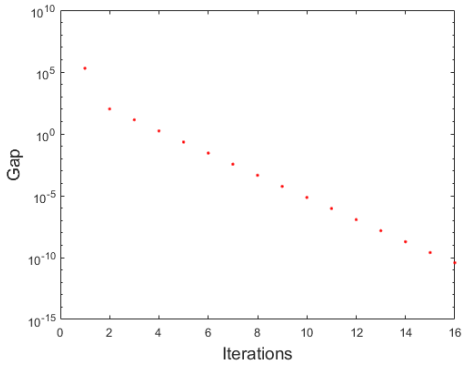
(a) MSA-step



(b) Proportional update of 0.1



(c) Proportional update of 0.5



(d) Proportional update of 1

Figure 3.7: Convergence of different step sizes on the network of Leuven

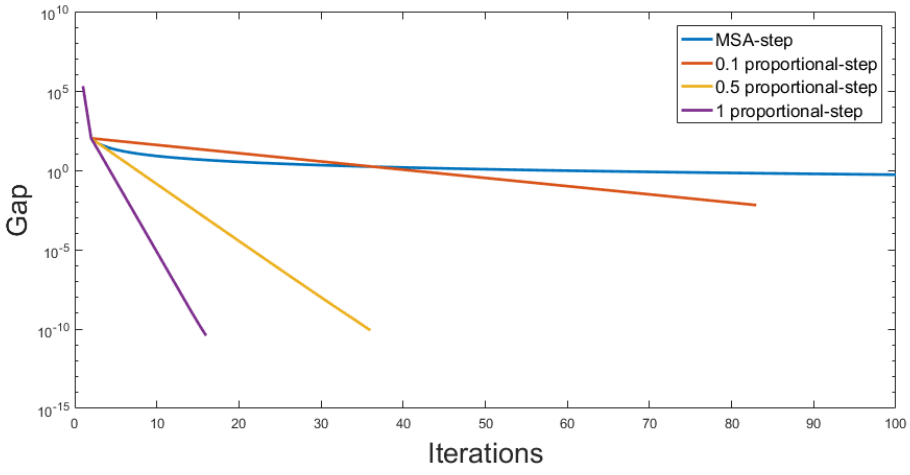


Figure 3.8: Convergences with different step sizes on the Leuven network

3.3 Turn characteristics

As stated above, the utility to make the turn from link *k* to link *a* ($v_n(a|k)$) contains the travel time on link *a* and other characteristics of the turn. A few examples will show how these characteristics influence the results and what these characteristics can be. The utility from a turn is:

$$v_n(a|k) = \beta_{TT} * TT + \beta_{UTurn} * Uturn + \beta_3 * Characteristic3 + \dots \tag{3.3}$$

Each characteristic has its own parameter. When calibrating, only the proportion between different β parameters will have to be calibrated. Which means that β_{TT} can equal one for simplicity.

3.3.1 U-Turn

An example of a characteristic that can be used is a penalty for a U-turn. In most assignments 'searching traffic' (traffic that for example is looking for a parking spot), is not considered hence U-turns are not an expected behaviour. An extra penalty for U-turns in the utility will then reduce the probability for routes with U-turns (hence also for routes with 2 successive U-turns which would form unrealistic cycles). To demonstrate this, an assignment has been done to a small network (figure 3.9). Figures 3.10 shows an assignment without any U-turn penalties (3.10a) and one with a very high U-turn penalty (3.10b).

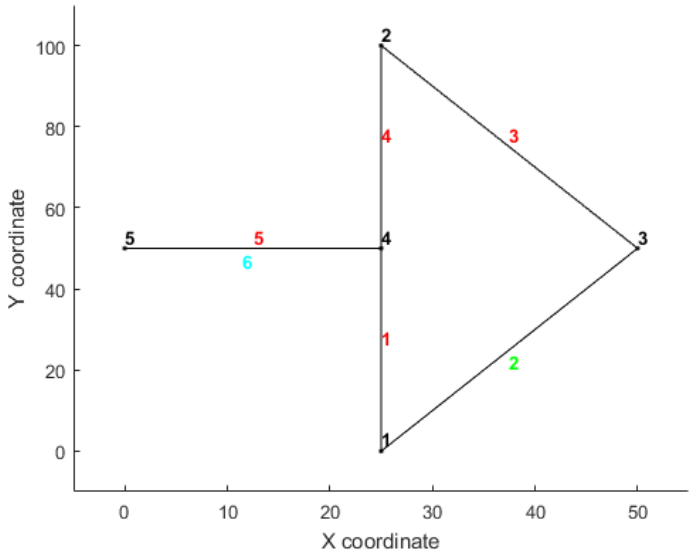
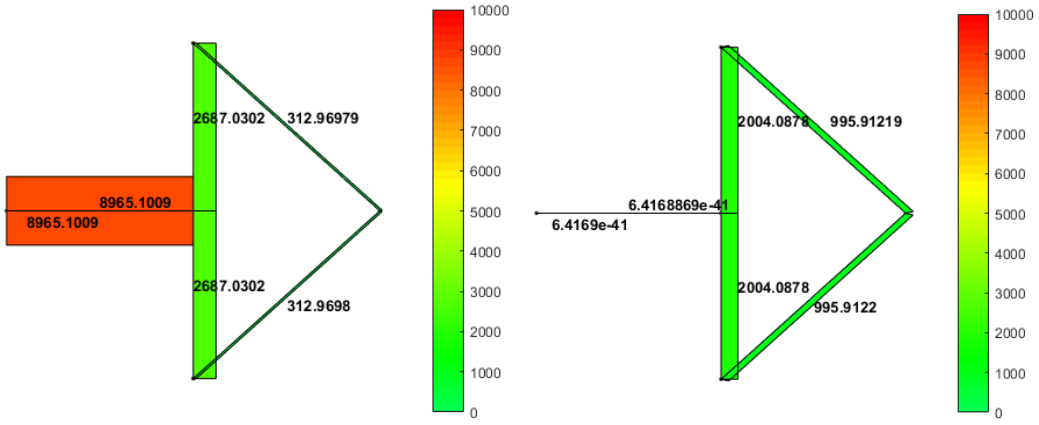


Figure 3.9: A simple network. Black numbers represents node numbers, while coloured ones gives the link number.



(a) Without a penalty for U-turns

(b) With a penalty for U-turns

Figure 3.10: Link flows on a simple network with U-Turns, demand is 3000 from bottom node (1) to top node (2)

The corresponding P-matrices are given in table 3.1. With the U-turn penalty, the probability has lowered but is still larger than zero. With these matrices, the probability of a path can be calculated. These matrices show also that every possible path is always considered as a possibility. In table 3.2 a few of these paths probabilities are calculated. Without any penalty, the probability of using a path without any U-turn is only $0.207 + 0.10 = 0.307$ or 30.7%. This means that the other traffic uses a path with a U-turn. The probability lowers with every extra loop the traffic takes.

Because RL works with a multinomial model without correlations between the different paths, the ratio of the probability of the two routes not effected by the U-turn penalty stays the same. This is because, in a logit model, only the different in utilities defines this ratio. In this example, this gives $0.20/0.10 = 0.67/0.33$. The ratio of probabilities between the two paths without U-turns stays the same.

Table 3.1: P-matrix of the assignment

Without U-turn penalties

P	1	2	3	4	5	6	7	8
1				0,23	0,77			
2			1					
3								1
4								1
5						1		
6				0,23	0,77			
7	0,90	0,10						
8								

With U-turn penalties

P	1	2	3	4	5	6	7	8
1				1	∼0			
2			1					
3								1
4								1
5						1		
6				1	∼0			
7	0,67	0,33						
8								

Blank entries represents zero, while ∼0 represents a value close to zero but slightly larger. Link *7* is the connector to node *1* (origin), while link *8* is the connector to node *2* (destination).

Table 3.2: Probability of paths

Path	**Probability without penalty**	**Probability with penalty**
7→1→4→8	0.90 * 0.23 * 1 = 0.207	0.67 * 1 * 1 = 0.67
7→2→3→8	0.10 * 1 * 1 = 0.10	0.33 * 1 * 1 = 0.33
7→1→5→6→4→8	0.90 * 0.77 * 1 * 0.23 *1 = 0.159	∼0
7→1→5→6→5→6→4→8	0.90 * 0.77 * 1 * 0.77 * 1 * 0.23 * 1 = 0.123	∼0
...	...	...

Link 7 is the connector to node 1 (origin), while link 8 is the connector to node 2 (destination).

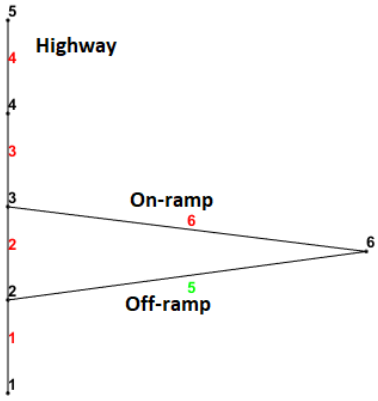
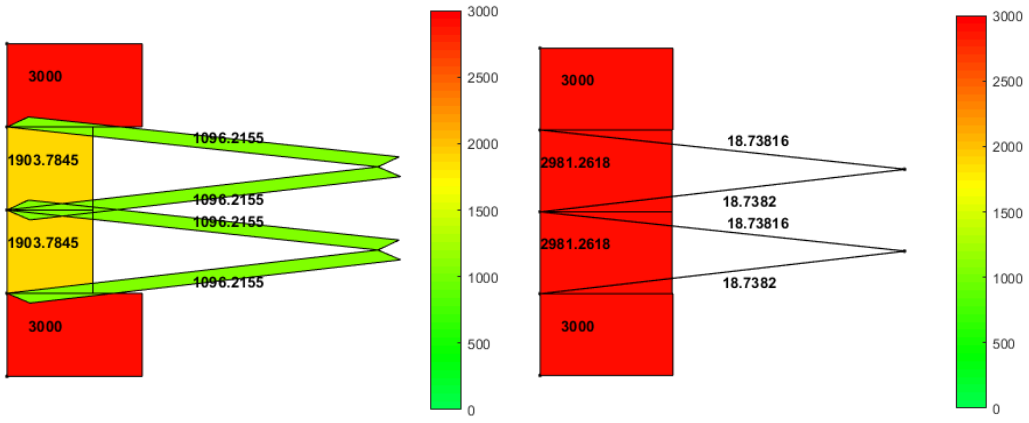


Figure 3.11: A network with different hierarchical links

3.3.2 Hierarchy

Most network layouts are made of different hierarchical layers, with the highway being the highest layer in the hierarchy. Imagine now a highway with an off- and on-ramp (as illustrated in figure 3.11). Taking the off- and on-ramp instead of just the highway can have only a little less utility. Therefore, in a stochastic assignment, it will be used by people in the model. In real, people will only consider this option if they can avoid very severe congestion as using the off- and on-ramp is not really an alternative. A penalty for lowering in a hierarchical layer can help this problem. Figure 3.12 shows the link flows before and after the introduction of a penalty.

The same analysis of the P-matrix like with the U-turn penalty could be made here. The figures show clearly that with the hierarchical penalty, fewer people use the off- and on-ramp.



(a) Without a penalty for hierarchy

(b) With a penalty for hierarchy

Figure 3.12: Link flows on a simple network with hierarchy, demand is 3000 from bottom node (1) to top node (5)

3.4 Conclusion

The formulas proposed by Fosgerau et al. (2013) (equations 2.6,2.7 and 2.8) can be easily implemented for a static assignment. By using a full route set, the algorithm is more stable, which leads to the use of proportional step sizes (to even a full step size). This then leads to a much smoother convergence.

Unfortunately, not all characteristics of RL are desirable. Without taking the correlation between different paths into account, a path with a small (in terms of utility) loop will only have a little less probability than the route without the loop. By introducing penalties, some of these unrealistic characteristics can be eliminated, for example, the U-turns.

RL is based on a logit-model, which does not handle correlations in the error terms. This typically occurs with overlapping routes. This is discussed in more detail in section 6.1.

Nonetheless, the overall conclusion is positive and makes further research to make RL work in a dynamic traffic assignment desirable.

Chapter 4

Full route set in Dynamic assignment

A dynamic traffic assignment has a time dimension. Time has to be made a discrete variable. This is done my introducing time steps. Each time step represents a specific time. Given the demand of every time step, a dynamic traffic assignment calculates the flows on each link at each time step. While it seems not complex of adding just a time dimension, the complexity is much larger in a dynamic than a static traffic assignment.

The algorithm based on Recursive logit made for this research is only a part of a total dynamic traffic assignment. The dynamic network loading (DNL) of the link transmission model (LTM)(Himpe et al., 2016) will be used to test and evaluate the developed algorithm. This software is freely available on the internet[1]. The algorithm described below can easily be implemented in other algorithms. We consider no departure time choice, hence demand is fixed for every time interval.

4.1 The Algorithm

An overview of the algorithm is displayed in figure 4.1. The outer loop consists of iterating over all destinations. First, the maximum perceived utility per time step is determined (the yellow block in the figure). After that, the turning fractions are calculated (per node per time step for each destination). These turning fractions are slightly different defined then the P matrix in the static assignment. Turning fractions are defined per node instead of one P matrix for the whole network. This is done because the DNL of LTM works with turning fractions. The algorithm can easily be adjusted to a 3d P matrix, with the 3rd dimension being the time. With these turning fractions, DNL will calculate the link flows which leads to new travel times.

[1] http://mech.kuleuven.be/en/cib/traffic/downloads

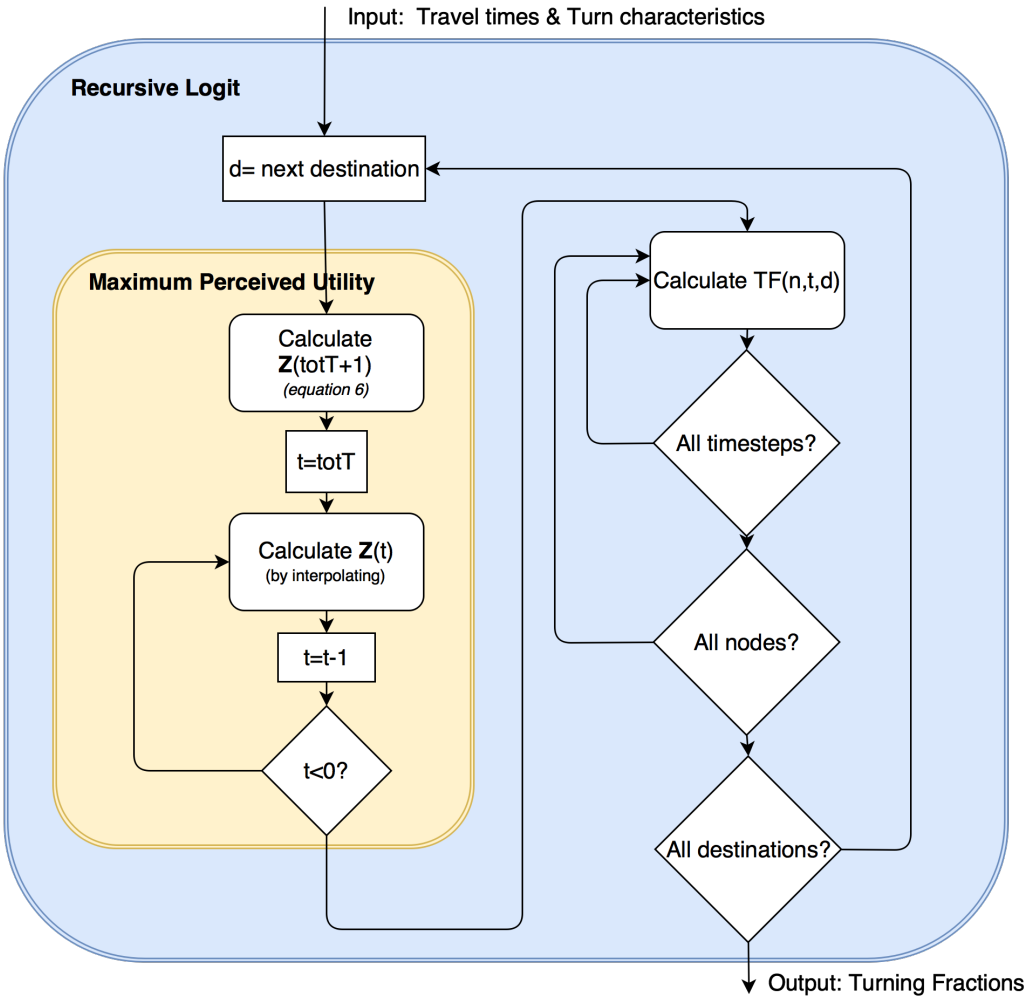


Figure 4.1: The algorithm used in the dynamic assignment

4.1.1 Maximum Perceived Utility

To determine the (transformed) maximum perceived utility, a static assignment is used in the last (*totT+1*) time step. Next, the other time steps are calculated in an upwind order. This is done because $Z(t)$ can depend only on time steps $t^* \geq t$. To calculate the other time steps, the algorithm will interpolate between two (already given) values of $Z(t')$. This is true because the utility at a node depends on the expected downstream utilities. These are linked through the travel time (and other turn characteristics) between two nodes. To make the algorithm simpler, only time steps that are smaller than the minimum travel time are used. In this case $Z(t)$ will only depend on time steps $t^* > t$. Section 6.2 will go into more detail about this and also propose a suggestion to make the algorithm handle the extra complexity. This extra complexity comes from the fact that $V_n^d(a,t)$ requires (for example) $V_n^d(k,t)$ which itself can be related to $V_n^d(a,t)$, all in the same time step.

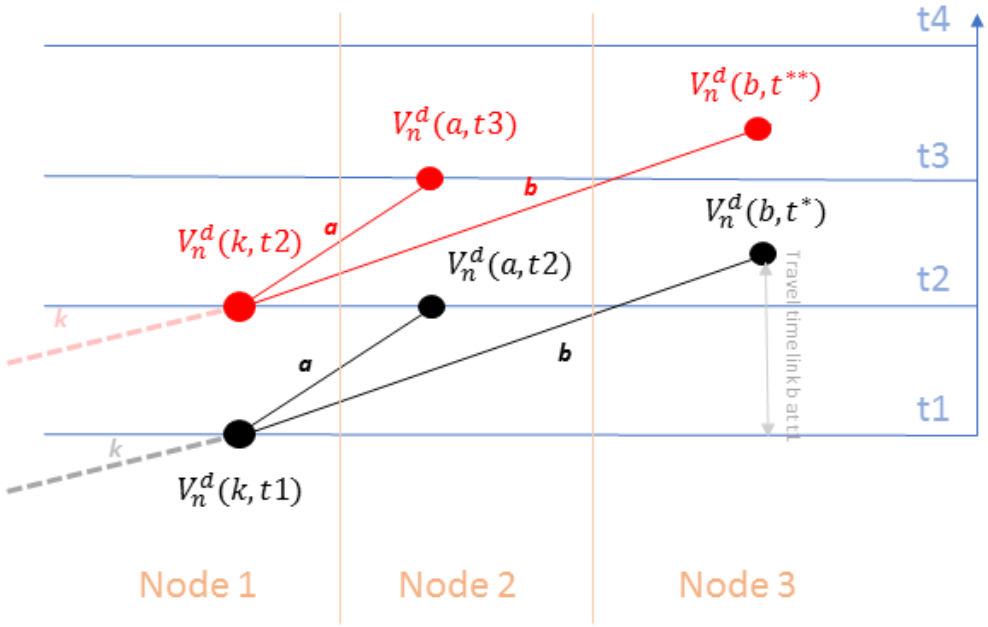


Figure 4.2: A simple network plotted with its time dimension. The time (z-axes) indicates the time the end of the link will be reached.

To calculated the maximum perceived utilities of the next time step, the logsum is taken. The logsum is the natural logarithm of the sum of utilities of all possibilities. Here is an example on how to do this, assume a small part of a network, plotted in figure 4.2. In the figure, three nodes are plotted on the x-axes. Node *1* is connected to the two other nodes by links *a* and *b*. The y-axes defines the time. t_1, t_2, t_3 and t_4 are different time steps used in the algorithm. The vertical distance of a link represents the travel time of the link at time of the beginning of the link. The travel time on link *a* ($tt_a(t_1)$) is exact a time step, while the travel time on link *b* ($tt_b(t_1)$) is between one and two time steps at time t_1. The problem now is to calculate the value of $V_n^d(k,t_1)$. Node *1* has two different outgoing links, *a* and *b*. The end of link *a* is reached at time $t1 + tt_a(t_1) = t_2$ when leaving link *k* at time t_1(the same holds for *b* with time $t_1 + tt_b(t_1) = t^*$). As stated before, $V_n^d(k,t_1)$ is calculated as:

$$V_n^d(k,t_1) = \frac{1}{\mu} * ln\left[\left(v_n^d\left(k,a,t_1\right) + V_n^d\left(a,t_2\right)\right) + \left(v_n^d\left(k,b,t_1\right) + V_n^d\left(b,t^*\right)\right)\right] \quad (4.1)$$

With $v_n^d(k,a,t_1)$ being the utility for going from link *k* to link *a* at time t_1. Of course not every time is represented in time steps. For time t^* that is in between t_2 and t_3, a linear interpolation is used. Take for example that $t^* = t2 + 0.4 * \Delta t$. This means that $V_n^d\left(b,t_2\right)$ will be calculated as: $0.6 * V_n^d\left(b,t_2\right) + 0.4 * V_n^d\left(b,t_3\right)$. To do this, $V_n^d\left(b,t_2\right)$ and $V_n^d\left(b,t_3\right)$ needs to be known at the time of calculation of time step t_1, which is why we proceed through time from the latest time *totT* in an upwind order.

4.1.2 Turning Fractions

Once all maximum perceived utilities are calculated, the last thing remaining is to calculate the turning fractions. To calculate these turning fractions (or probabilities) the utilities for each possibility is needed. These depend on turning characteristics, the travel times (per time interval) and maximum perceived utility of the end node. All these ingredients are already known, this means that every calculation towards probabilities is independent of other results. The order in which the turning fractions are calculated, is thus unimportant. It can even be done in parallel. In this algorithm, all nodes are iterated from the start time to the end time. Returning to the example in figure 4.2, the probability of going to link *a* (at time t_1) is then:

$$P(k,a,t1) = \frac{exp\left[\mu\left(v_n^d\left(k,a,t_1\right) + V_n^d\left(a,t_2\right)\right)\right]}{exp\left[\mu\left(v_n^d\left(k,a,t_1\right) + V_n^d\left(a,t_2\right)\right)\right] + exp\left[\mu\left(v_n^d\left(k,b,t_1\right) + V_n^d\left(b,t^*\right)\right)\right]} \tag{4.2}$$

As before, the maximum downstream utility of a node at a time between time intervals may need to be known. Linear interpolation between the two known time steps is again the solution to this problem.

Note that because once the maximum downstream utility is known, the calculation of the turning fraction can be done. There can therefore be more efficient ways to implement this method. If the turning fraction is calculated as soon as the maximum downstream utility is known, there would be no need to interpolate twice.

The output of this algorithm are the turning fractions, which are stored in a data structure of size $|N| \times |T| \times |D|$, with $|N|, |T|$and$|D|$ the total number of nodes, time steps and destinations respectively. Each $TF(n,t,d)$ is a matrix $(|IL| * |OL|)$, with $|IL|$&$|OL|$ being the number of incoming and outgoing links on the node *n* respectively. $TF(n,t,d)_{a,b}$ is the percentage of flow going to link *b* that was on link *a* towards destination *d* at node *n* at time *t*. Each row of a $TF(n,t,d)$ matrix is summed to one, because destinations are defined as links. Over iterations towards convergence, these TF are proportionally averaged.

4.1.3 Equilibrium

In this research, the time steps used in the algorithm determining the turning fractions and the one used in the DNL are the same. This does not necessary have to be the only option. When different time steps are used, the question remains on how to interpret these turning fractions. One possibility is to assume the turning fractions fixed between the time steps of the algorithm described. Another possibility is to (linearly) interpolate between the turning fractions. For simplicity of this research, both time steps are taken the same size (which can be done here without constraint on the DNL because the time step in the iterative version of LTM does not need to comply with CFL conditions (Himpe et al., 2016)). In the used DNL, turning fractions are fixed within a time interval and based on the utility and maximum utility of the last vehicle in that time interval.

The gap is calculated over the link flows as followed: the maximum total change of flows in a time step. While TF are averaged over the iterations.

4.1.4 Pseudo code

The algorithm is split into three separate functions, which call each other. The first part (algorithm 4.1) is the outer loop and tries to reach equilibrium. The second part is the blue part of figure 4.1, it calculates the turning fractions once the maximum perceived utility is given (algorithm 4.2). The maximum perceived utility is calculated in the last part (algorithm 4.3) and is the yellow part of the figure.

As said before, in the dynamic case the output of the algorithm are turning fractions (with [#nodes,#timesteps,#destinations] as the dimension). This is done because DNL of LTM works with such turning fractions, the algorithm can easily be adapted to output a 3rd P-matrix with time as it thirds dimension. However this matrix is very sparse and therefore more efficient data structures are required.

Note that in the algorithms below, θ is used instead of $\frac{1}{\mu}$.

Algorithm 4.1 Recursive Logit Equilibrium in Dynamic Assignment

function DYNAMIC EQUILIBRIUM(Network,Demand,betas,θ,dt,totT)
  Calculate all characteristics
  $TT \leftarrow cvn2tt(cvn_{up}, cvn_{down}, dt)$
  $Flows \leftarrow cvn2flows(cvn_{up}, dt)$
  $TF \leftarrow TurningFractions(Network, Characteristics, TT, betas, dt, totT)$
  while it<maxIt and gap<gapTreshold **do**
    $[cvn_{up}, cvn_{down}] \leftarrow LTM_{MC}(Network, dt, totT, TF)$
    $TT \leftarrow cvn2tt(cvn_{up}, cvn_{down}, dt)$
    $newFlows \leftarrow cvn2flows(cvn_{up}, dt)$
    $TF_{new} \leftarrow$
      $TurningFractions(Network, Characteristics, TT, betas, dt, totT)$
    $TF \leftarrow update(TF, TF_{new})$ ▷ Depends on step size
    $gap \leftarrow max(abs(Flows - newFlows))$ ▷ Max sum over all time steps
    $Flows \leftarrow newFlows$
    $it \leftarrow it + 1$
  end while
end function

Algorithm 4.2 Turning Fractions

function TURNING FRACTIONS(Network,betas,θ,dt,totT)
  for every destination d **do**
    $maxUtilities \leftarrow maxUtility(d)$
    for all nodes n **do**
      $links_{in} \leftarrow ingoing(n)$
      $links_{out} \leftarrow outgoing(n)$
      for all time steps t, from 0 to totT **do**
        $P \leftarrow zeros(links_{in}, links_{out})$
        for all ingoing links l_{in} **do**
          for all outgoing links l_{out} **do**
            $t_{arrival} \leftarrow t + TT_{l_{out}}$
            $[t_1, t_2] \leftarrow timeToTimeSteps(t_{arrival})$
            $utility_{downStream} \leftarrow$
              $interpolate\left[utilmap\left(l_{out}, t_1\right), utilmap\left(l_{out}, t_2\right)\right]$
            $penalty \leftarrow penalty(Network, Characteristics, betas)$
            $utility \leftarrow \beta_{TT} * TT + penalty + utility_{downStream}$
            $P(l_{in}, l_{out}) \leftarrow exp(\theta * utility)$
          end for
        end for
        $TF(n, t, d) \leftarrow normalize(P)$
      end for
    end for
  end for
end function

Algorithm 4.3 Maximum Perceived Utility

function MAXUTILITY(Network,betas,θ,dt,totT)
  Calculate M
  Calculate b
  $z \leftarrow (I - M) \backslash b$
  $utilmap(:, totT + 1) \leftarrow \frac{1}{\theta} * ln(z)$
  for all time steps t, from totT to 0 **do**
    for all links l_{in} **do**
      $links_{out} \leftarrow outgoing(l_{in})$
      for all outgoing links l_{out} **do**
        $t_{arrival} \leftarrow t + TT_{l_{out}}$
        $[t_1, t_2] \leftarrow timeToTimeSteps(t_{arrival})$
        $utility_{downStream} \leftarrow$
          $interpolate\left[utilmap\left(l_{out}, t_1\right), utilmap\left(l_{out}, t_2\right)\right]$
        $penalty \leftarrow penalty(Network, Characteristics, betas)$
        $utility \leftarrow \beta_{TT} * TT + penalty + utility_{downStream}$
        $utilmap(l_{in}, t) \leftarrow utilmap(l_{in}, t) + exp(\theta * utility)$
      end for
      $utilmap(l_{in}, t) \leftarrow \frac{1}{\theta} * ln(utilmap(l_{in}, t))$
    end for
  end for
end function

4.2 Influence of the step size

Like in the static assignment, the step size is important for the convergence in the dynamic assignment. Can the same conclusions be made about the stability of the algorithm? To test this, a small network is used, which can be seen in figure 4.3. There are two routes for going west to east, after one hour, the demand is increased to a value larger than the capacity of a link (link 5) on the shortest path. This way a congestion is simulated. After some time, the demand decreases again and the congestion should disappear.

As before, four different step sizes are considered, namely the MSA-step, and three different proportional step sizes (0.1, 0.5 and 1). The different convergence can be seen in figure 4.4 separately and in figure 4.5 together.

There is a difference with the static assignment, only one of the step sizes reached convergence (gap lower than 10^{-10}) in less than 2500 iterations. How larger the proportional step size, how steeper the convergence but how larger the minimum gap. Taking a full step does not go into the direction of convergence. Only the proportion update of 0.1 has reached a gap lower than 10^{-10}, but it seems just lucky at the end as the convergences went up and down.

If we take a closer look at what happens when using a proportional update of 1, we see that the algorithm flip-flops constantly between two situations (after 150 iterations). To visualise this, figure 4.6 shows the two split fractions at node 2 of these two states. The flip-flop only occurs for certain time steps after the demand decreases again.

Analysing the plots, reveal that there can be a better step size, for example, a dynamic one that starts big and get lower during the iterations. Something analogue as the MSA-step but that decreases slower. A simple test shows that improvement can indeed be found. Figure 4.7 shows different step sizes analogue as MSA. Instead of inverting the iteration number (1/it) as the step size, different roots of the iteration number is taken.

Further research to see what the best step size would be, is needed.

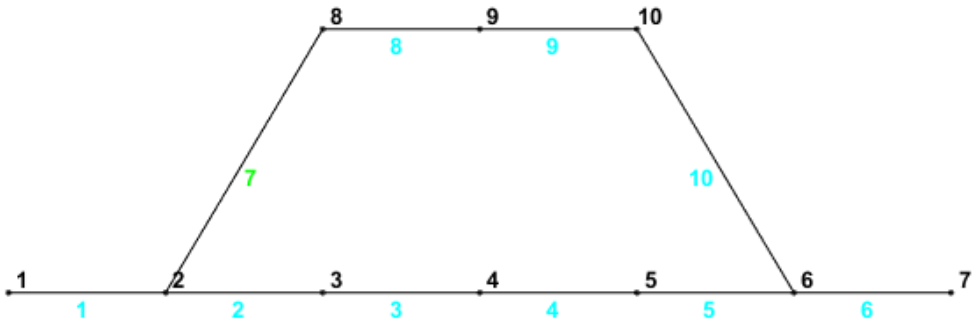
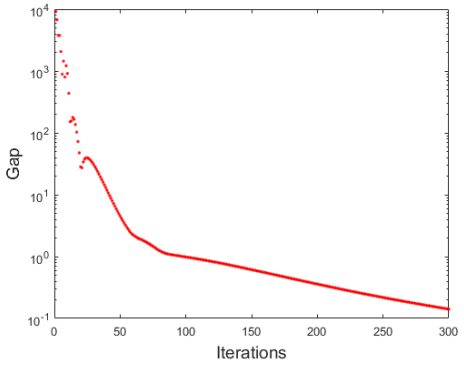
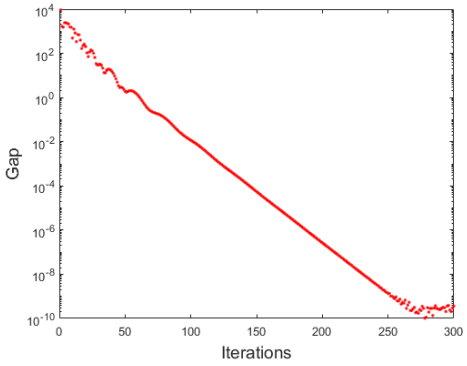


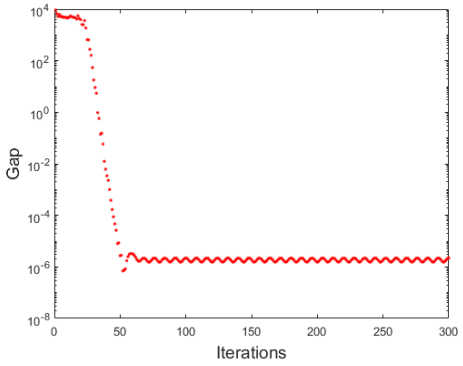
Figure 4.3: Test network of the dynamic assignment



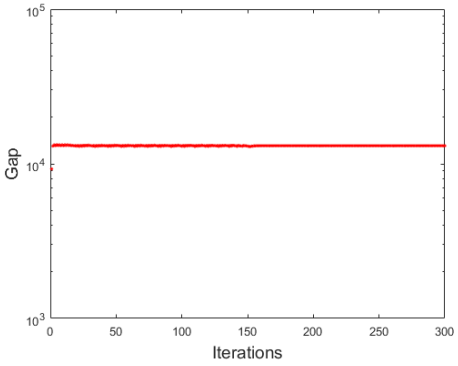
(a) MSA-step



(b) Proportional update of 0.1



(c) Proportional update of 0.5



(d) Proportional update of 1

Figure 4.4: Separate convergence of different step sizes in a dynamic assignment

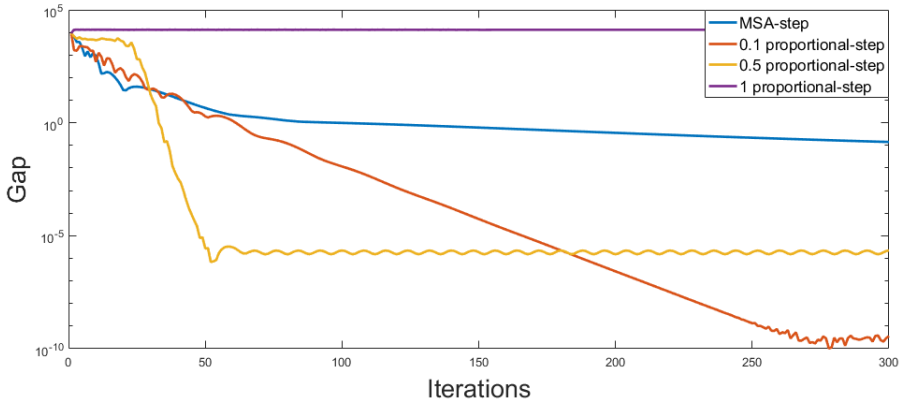
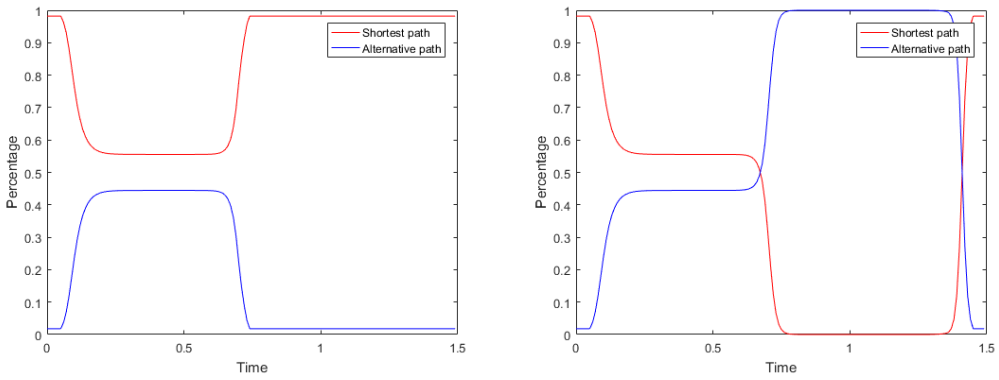


Figure 4.5: Convergences with different step sizes in a dynamic assignment



(a) Situation 1 (b) Situation 2

Figure 4.6: Split fractions for proportional step size of 1

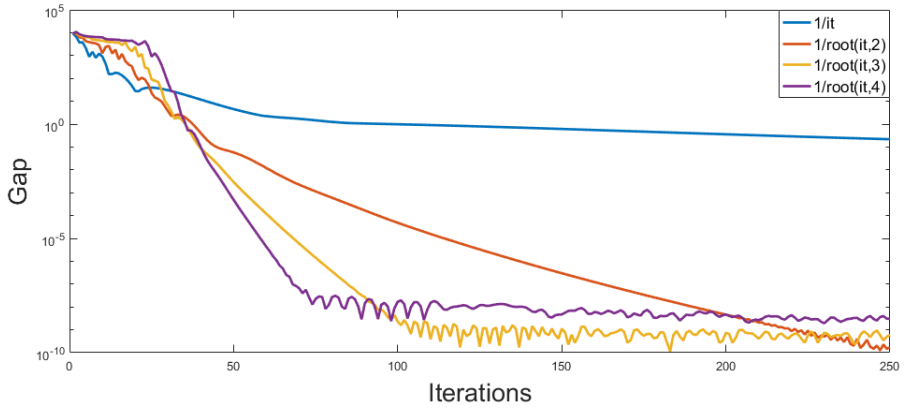


Figure 4.7: Convergences with different step sizes in a dynamic assignment

4.3 Turn characteristics

The same turn characteristics can be implemented in the dynamic assignment. With the extra time dimension, turn characteristics can also change with time. An example of such characteristic is, for example, light intersections, it can be that during peak hours the green time for one turn is higher than in normal hours. While developing the recursive logit in the dynamic assignment, other turn characteristics have come to the attention. These can also be implemented in the static case as long as they are not time dependent.

4.3.1 Left and right turns

An extra turn characteristic that has been added for the dynamic assignment, is a penalty for left and right turns. These penalties can be dependent on physical aspects of the turn like for example the turning degree or the number of outgoing links.

Besides a flat penalty for turning left or right, turning delays can be implemented. These delays can depend on the flow of the crossing turn that has higher priority, for example, the straight turn of the opposite link.

4.3.2 Light intersection

When there is a light at an intersection, people will experience some delay depending on the green time of the turn and the people going to the intersection. It is possible to take these delays into account for the route choice model. The subcritical delays are not captured by LTM. If demand is however above intersection capacity it will be taken into account. People will here choose taking the delay into account, but will not experience it on the network. Some other DNL's with other node models do take this into account, but this requires added complexity to the DNL.

In the literature, there are many kinds of delay functions, like the formulation of Webster (1958). This formulation is not very useful in this context as the formulation of Webster only holds to a saturation degree of one. The saturation degree is the ratio between the flow and the adjusted capacity. The adjusted capacity is the capacity of the link multiplied by the percentage of green time (=seconds of green/seconds of a cycle). For this small addition to the model, a simple linear delay function is used. In further research, other functions can easily be implemented.

A small example can show how this can work. Consider the network in figure 4.8. There is only demand from node 1 to node 8. There is a traffic light at the intersection that leads to the three possible paths (no loops are possible) that are equally long (only different in utility is the delay on the intersection). The green times for each route is plotted in the figure.

Figure 4.9 shows the split at each time. At time $t = 0.5$ the demand increases. This increases the flow on the first link, which results in higher saturation degrees. The route with the lowest impact (going right) becomes relatively more popular, while the route with the highest impact (going left) becomes relatively less popular.

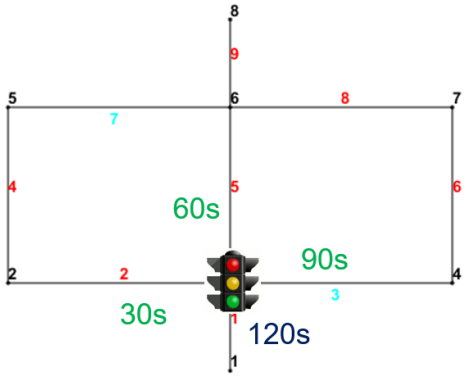


Figure 4.8: Example network with traffic lights
A total cycle of 120 seconds with 30, 60 and 90 seconds for going left, straight and right respectively.

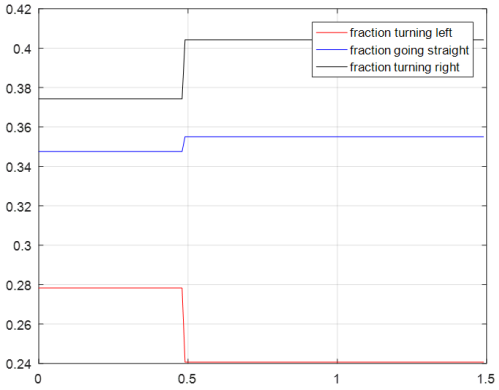


Figure 4.9: Split fractions on the network with traffic lights

4.3.3 Toll

Toll is another import characteristic that is implemented. Toll can be constant over time and place and traffic intensity but can also be dependent on all three dimensions. If toll is dependent on the traffic intensity, it adds extra complexity to the problem (like light intersections).

Adding toll can be helpful to calculate the system optimum, the toll is then equal to the congestion cost on a link (WARDROP, 1952). But it can also be helpful to evaluate certain toll policies. Traffic pricing is a hot topic right now and every model should, therefore, be able to work with it. Traffic pricing finds its origin at the fact that people only take (and experience) personal utility into account. They do not consider the price for society because they travel. This cost for society is the fact that because of traveller travelling, other travellers will experience more delay. There are also environmental costs that travellers do not always consider. By traffic pricing, this shortage of the market can be solved.

An example of a dynamic toll is shown here. In the network on figure 4.10 there is toll on link *5* and link *6*. The toll on link *6* is constant while the toll on link *5* is raised between *t=0.5* and *t=1*. The different paths have the same cost (same length and same speed) besides the toll. Figure 4.11 shows the different split fractions. As expected, the percentages change when the toll changes.

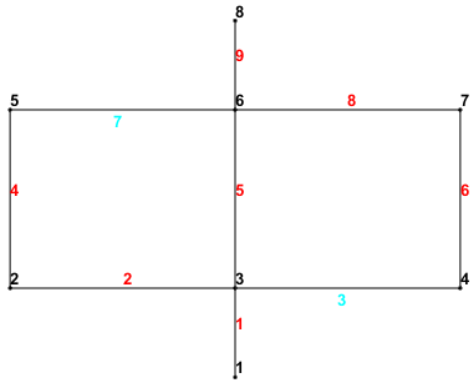


Figure 4.10: Example network with dynamic toll

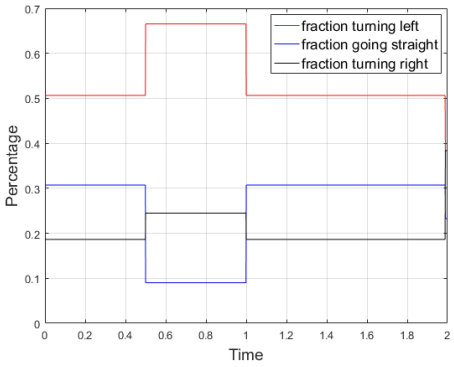


Figure 4.11: Split fractions on the network with dynamic toll

4.4 Conclusion

As the model shows, it is possible to use recursive logit in the dynamic assignment to include a full route set choice model. By implementing the method as described above, some gain of using recursive logit seems to be lost. A proportional step size of 1 can flip-flop between two states (in the static assignment no such case have been found). But as showed by a small example, better dynamic step sizes can still be used to reach convergence relatively fast.

Turn characteristics can also in the dynamic assignment easily be added, but with the extra time dimension, dynamic penalties can be added. A good example is toll. Good pricing policies include a time depending toll, it is, therefore, a plus that this model can easily handle it.

The next chapter goes into more detail on how and when the algorithm can be used.

Chapter 5

Use of the algorithm

In this chapter, some (alternative) use of the algorithm is given. The algorithm as described in the previous chapter is used in a larger network. Later another use of the algorithm is described. But first the answer to the question "Is there always a meaningful solution in the assignment?" is given.

5.1 Guaranteed solution

There is no guarantee that the recursive logit calculation on a network will have a (feasible, meaningful) solution. To get a meaningful solution, **I**-**M** needs to be invertible. The physics of the network (which links connect to which links) and also the used turn characteristics with their beta parameters will determine the characteristics of **M**. When **I**-**M** is almost singular, z will have values larger than 1. This means that the maximum perceived utilities will be positive.

Another more intuitive way to look at this is the following: if there are many links with low costs and the variance of the random utility ε is large, then it is possible that the utility will be larger than zero. A utility larger than zero is bad because if travellers take this link, they gain utility. This of course results in the fact that travellers want to use this link, if possible even multiple times. Imagine that now a small loop consists only of links with a positive utility, then travellers will never leave the loop because they keep gaining utility. This is demonstrated in figure 5.1.

If such singularities occur, introducing extra penalties (negative utility) is a way to avoid it. To check if such singularities can occur on a network, it is sufficient to construct the matrix M with the highest utility possible. The highest utility possible is reached when the travel time is the free flow travel time. The travel time can only grow, herewith reducing utility. The same holds for penalties, M needs to be tested with the lowest possible penalty to the utility. If with this worst case matrix M no singularities occurs, then neither will they occur during computations where the utility can only decrease.

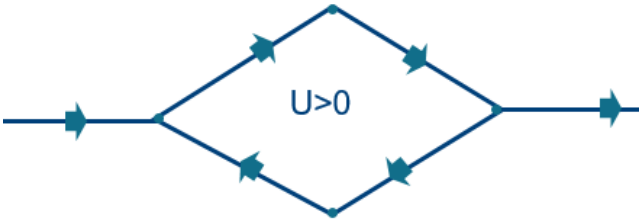


Figure 5.1: Small network with a positive utility loop

5.2 Results of a larger network

To show how the algorithm performers on a more real network, the algorithm is executed on the network of Rotterdam. This network is a rudimentary representation of the ring road and main arterial roads in the Rotterdam area. This network consists of 560 links and 331 nodes. There are in total 44 nodes that serve as origin and 44 different nodes as a destination. The network is plotted in figure 5.2.

Figure 5.3 shows that the convergence is linear (with the y-axes of the gap in logarithmic scale), this is because a proportional step size is used. Between the first and second iteration, a large gap is noticed. This is normal considering that in the first iteration all travel times are free-flow travel times, so the improvement towards the second iteration with non free-flow travel times is large.

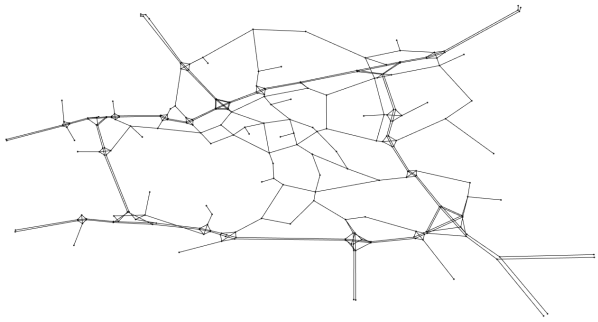


Figure 5.2: The network of Rotterdam

The Matlab traffic toolbox of LTM makes it easy to animate the flows over the network. This way results can easily be interpreted. To show what the algorithm can do, the turning fractions for one link towards one destination are visualised over time in figure 5.4. Two possibilities remain at zero probability, while the other two are taken both by the traffic. Due to circumstances, like more demand or more congestion, the ratio of probabilities between the two changes.

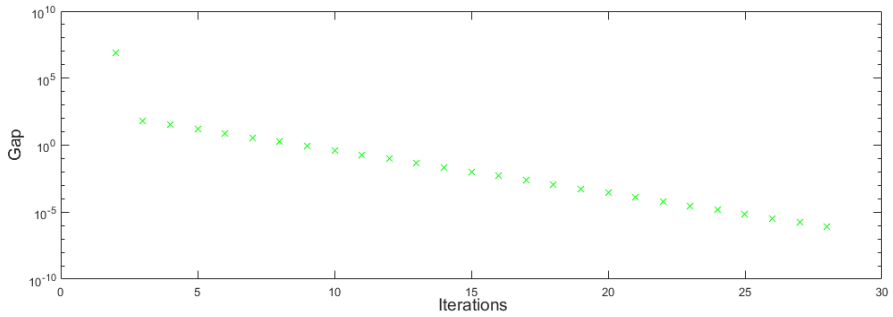


Figure 5.3: Convergence of the Rotterdam network

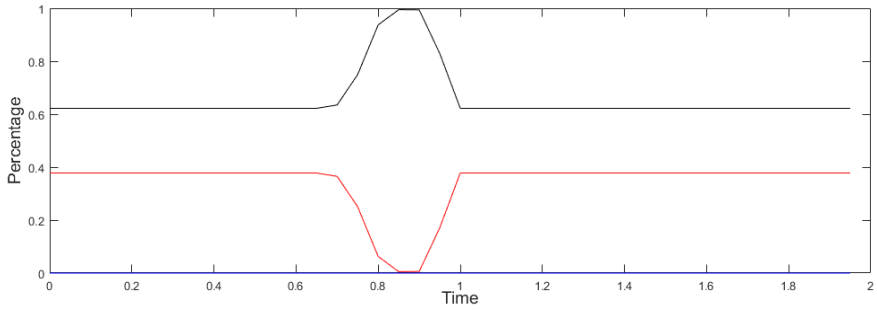


Figure 5.4: Turning fractions from one link towards one destination over time

5.3 Warm start

Another use of the algorithm is that it can be part of a warmly started assignment. A warm start means that results from a previous calculation are used as the initial condition. This way fewer calculations are needed if only a small part of the network has changed. As this algorithm takes travel times as input, free flow travel times (in the case of cold start) or travel times from a previous calculation on the same network (warm start) can both be used.

5.4 Departure time choice model

As said before, the DNL of LTM does not work with a departure time choice model. The route choice model of the dynamic assignment procedure can easily be combined with a departure time choice model. The Recursive Logit algorithm is constructed in such way that only the travel times and network characteristics are needed as input for calculation of the turning fractions, regardless of the magnitude of the demand or flow on the links.

The fact that people will change their departure time will have an influence on the travel times, but so does congestion. It is therefore unlikely that it would cause a problem in the route choice model. Of course, the equilibrium problem is more complex.

5.5 Route set generator, evaluator

Besides being used directly in a traffic assignment, the route choice model can be changed to generate a route set. A fixed route set gives more stability to the algorithm, but if the route set is not full, it is hard to know which routes to include.

Another use of the algorithm can, therefore, be to generate the route set. All routes with a probability higher than a threshold can then be used as plausible routes. The input of the algorithm can then be the travel times as measured on the streets or from other data.

Besides generating the route set, it can also be implemented to evaluate a route set or a route set generator. With a given route set, the algorithm can see if it contains all plausible routes or if it misses some.

The algorithm can also be used to quickly search for alternative routes, for example, what route will users take if the highway is blocked completely. By adjusting only one probability, another algorithm can determine which path has the highest probability.

Chapter 6

Further Research

The research in this thesis is rather new and ungrounded territory. It is therefore logical that not everything is researched yet. In this chapter, a few suggestions are proposed, based on the results of this research.

6.1 Path correlations

It is a well-known deficiency of the logit-model that biases occur in the choice probabilities if the error terms of the options are correlated. This typically occurs with overlapping routes. Recursive logit also suffers this deficiency.

This gives the first topic to research further. *"In real networks, paths connecting a given origin destination pair share links. Due to this physical overlap, it is generally thought that paths share unobserved attributes meaning that the path utilities are correlated. Ignoring this correlation may result in erroneous path probabilities and substitution patterns."* argues Fosgerau et al. (2013). He gives an example how to handle path size logit (Ben-akiva et al., 2012) in a full route set model for a static assignment. Further research needs to be done to see if the same approach can be made in a dynamic assignment.

6.2 Larger time steps

In the dynamic assignment, interpolation needs to be done between two already known downstream utilities from other time steps.

If we retake the example as given before but with larger time steps, see figure 6.1. Suppose the shortest travel time steps is smaller then a time step, then $V_n^d(a, t')$ will need to be interpolated between $V_n^d(a, t1)$ and $V_n^d(a, t2)$. The problem that now occurs is that $V_n^d(a, t1)$ is not yet (fully) calculated, making the interpolation impossible. A possible solution to this problem is introducing an extra loop around each time step. At first, the maximum perceived utility of a node is set equal to the values of the maximum perceived utilities at one time step later. After all transformed utilities from that time step are calculated, the calculations

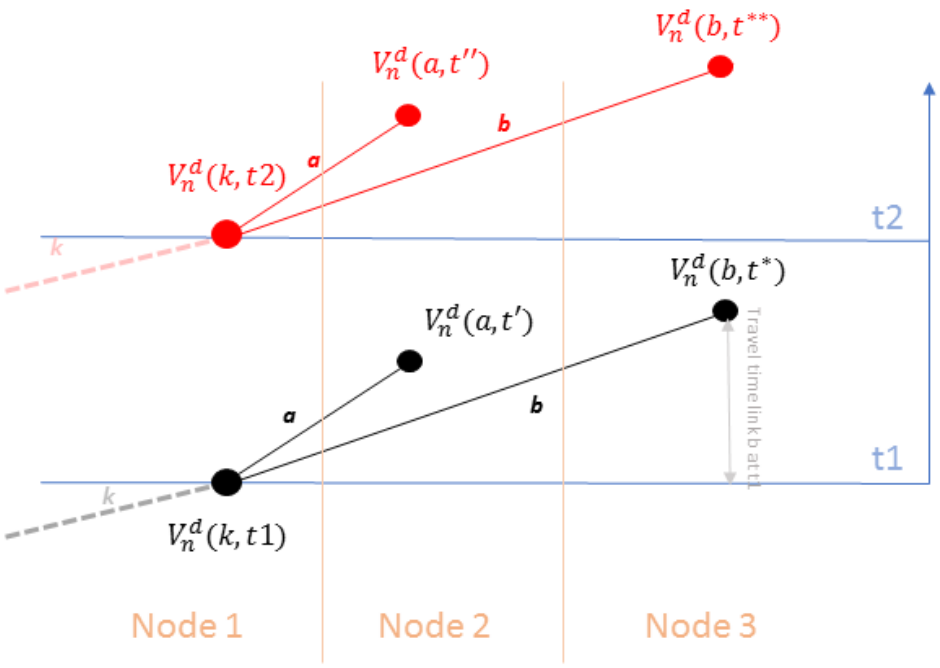


Figure 6.1: A simple network plotted with its time dimension

(interpolations) are repeated. This will now result in different results, this process is repeated until a fixed point is reached. Bigger time steps will reduce the general computation cost (for a given time domain), while introducing new iteration costs. A consideration is needed. But first, further research is needed to find out if it is possible at all. Note that larger time steps in the total assignments mean that also the DNL part of the algorithm should be able to handle these larger time steps. Most DNL algorithms are however constrained in their time step size due to so-called CFL conditions, with only a few exceptions like I-LTM Himpe et al. (2016).

Conclusion

Explicit route sets have some downsides, they do not contain every possible route and with a flexible route set the convergence problem is harder to solve. A fixed route set makes the algorithm more stable. An implicit full route set guarantees that no possible route is missed because it considers all possible routes (can be an infinite number) always. In this research that is done by using recursive logit, which calculates the probability of each turn given the traveller's destination. The utility of a turn depends on the turn characteristics. As always, the travel time is part of the utility. The utility of a turn is constructed in such a way, it is easy to add new turn characteristics. A characteristic can be a toll or a boolean for left-turns to a boolean indicating an increase (or decrease) in link hierarchy. The latter could, for instance, be used to avoid unrealistic routes that leave a higher road category over a very short distance (e.g. off-on-ramp combinations; or unrealistic rat-running through residential streets).

This makes the researched method promising to replace current algorithms used. Of course, further research needs to be done. A first issue is that not all networks-parameter sets will have a solution. Without a dedicated check for singularities, the algorithm would produce unrealistic flows. However, if the calibrated parameters are in the feasible solution space, a solution is guaranteed. By using a fixed full route set, the algorithm appears to be more stable. This results in fewer iterations needed for convergence, as now a fixed proportional update can be used instead of an update that has a lower impact the more iterations are done, like an MSA step. Further research towards dynamic step sizes needs to be done because the proportional update appears to have a lower bound (what can be larger than the convergence criteria).

This thesis has shown that the static procedure described by (Fosgerau et al., 2013) can be generalised to a dynamic traffic assignment by interpolating the maximum perceived utilities in an upwind order. After the maximum perceived utilities are determined, the probabilities of each turn can be calculated. These are then the input of a DNL.

By implementing penalties, turn characteristics can be implemented in the algorithm. There is no limit on how many penalties are allowed, neither on what they represent. This makes it easy to adjust utilities.

With these promising results, further research is needed. One topic can be finding a smarter way to determine if the network is guaranteed to have a feasible solution or how to change the network the smartest way possible in order to render a singular network feasible.

Faster computation times and smoother convergence are desirable properties of an assignment. With a warm start, which gives even faster computation times, the algorithm can be used in a real-time setting. The algorithm can also work on its own to quickly determine the turning fractions if something small changes (for example an accident on the highway which makes the capacity drop). In this case, the travel times could be handled as instantaneous.

Besides using the method only in a traffic assignment, it can also help to generate or to evaluate complete route sets or one alternative route. Another obvious topic for future research is embedding the stochastic DTA with recursive logit full route set of this thesis in a framework that considers departure time adjustments as well.

This thesis has shown some good and bad characteristics for using a full route set in a dynamic assignment. The research done is new and showed some good insight in the problem. The results show potential and does not close the door for further research.

Bibliography

M. G. H. Bell. Alternatives to Dial's logit assignment algorithm. *Transportation Research Part B*, 29(4):287–295, 1995. ISSN 01912615. doi: 10.1016/0191-2615(95)00005-X.

M. E. Ben-akiva, S. Gao, Z. Wei, and Y. Wen. A dynamic traffic assignment model for highly congested urban networks. *Transportation Research Part C: Emerging Technologies*, 24: 62–82, 2012. ISSN 0968090X. doi: 10.1016/j.trc.2012.02.006.

R. B. Dial and A. M. Voorhees. PROBABILISTIC MULTIPATH TRAFFIC ASSIGNMENT WHICH OBVIATES PATH ENUMERATION The problem. *Transpn Res*, 5:83–111, 1971.

M. Fosgerau, E. Frejinger, and A. Karlstrom. A link based network route choice model with unrestricted choice set. *Transportation Research Part B: Methodological*, 56:70–80, 2013. ISSN 01912615. doi: 10.1016/j.trb.2013.07.012.

E. Frejinger, M. Bierlaire, and M. Ben-Akiva. Sampling of alternatives for route choice modeling. *Transportation Research Part B: Methodological*, 43(10):984–994, 2009. ISSN 01912615. doi: 10.1016/j.trb.2009.03.001.

W. Himpe, R. Corthout, and M. J. C. Tampère. An efficient iterative link transmission model. *Transportation Research Part B: Methodological*, 92:170–190, oct 2016. ISSN 01912615. doi: 10.1016/j.trb.2015.12.013.

P. H. L Bovy. On Modelling Route Choice Sets in Transportation Networks: A Synthesis. *Transport Reviews*, 29(1):43–68, 2009. ISSN 0144-1647. doi: 10.1080/01441640802078673.

H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. ISSN 0003-4851. doi: 10.1214/aoms/1177729586.

J. G. WARDROP. ROAD PAPER. SOME THEORETICAL ASPECTS OF ROAD TRAFFIC RESEARCH. *Proceedings of the Institution of Civil Engineers*, 1(3):325–362, may 1952. ISSN 1753-7789. doi: 10.1680/ipeds.1952.11259.

F. V. Webster. TRAFFIC SIGNAL SETTINGS. *Road Research Lab Tech Papers /UK/*, 1958.