Stochastic User Equilibrium with Full Route Set in Dynamic Traffic Assignment

Jeroen Verstraete[1]
Dr. Ir. Willem Himpe[1]
Prof. Dr. Ir. Chris M.J. Tampère[1]

Abstract

In current practice, there are two main concepts to handle the route set in an equilibrium route choice model. The first one is to maintain a fixed route set throughout the iterations for each OD pair. The second one is to determine the route set during the execution (every iteration or less), depending on the current congestion levels. They both have some problems. With a fixed route set, there is a change that an important route is not considered. While having a flexible route set over the iterations may negatively affect the convergence, Recursive Logit could resolve these issues as it is an implicit route choice model that considers all routes. It 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 paper first illustrates how recursive logit converges very stably to static equilibrium. The main contribution of this paper is the finding of a method to implement a full route choice with recursive logit in a dynamic assignment. Another contribution is suggestion a few important parameters that determine the utility for a traveller. Other parameters can easily be added. Finally, the relation between these parameters and finding a solution is formulated.

Keywords: Dynamic Traffic Assignment, Stochastic User Equilibrium, Recursive Logit, full Route Set

1 Problem

In every traffic assignment model, a route set needs to be chosen. A route set is a collection of all plausible routes between an origin destination pair. Ideally, the route set should contain every possible 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

[1]KU Leuven, L-Mob (Leuven Mobility research Center)

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.
Let us consider a stochastic user equilibrium (including the extreme case of approaching zero variance for a deterministic solution), current algorithms work with an explicit route set, this means that every route considered is explicitly stated. 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. With the change of topological order, the route set changes. 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 of the quality of the assignment.
There are two concepts of how to deal with an explicit 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 criterion 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 and Ltd TTRV (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 fixed route set is used. Consider the network in figure 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 2. How the gap is calculated, will be explained later. Important now is that a gap is a way to quantify the solution. The lower the gap, the closer to the equilibrium.

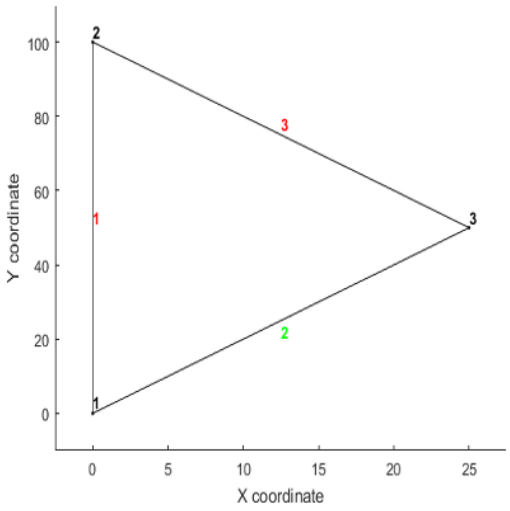


Figure 1: A simple network. (Black numbers represent the node numbers, coloured numbers the link number.)

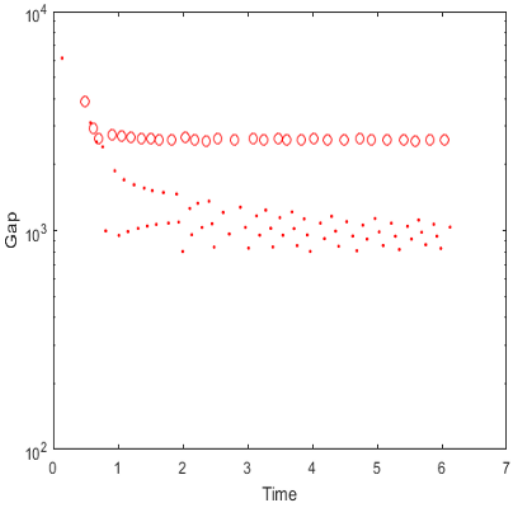


Figure 2: Convergence of the Dial algorithm, a 'o' indicates only one route is in the route set, while a '.' indicates both routes are in the set.

A solution to this problem 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: $w_{n,m} = exp(-\alpha * Cost_{n,m})$. Then W expresses the weights between each pair of two links directly (with a weight of 0 if the path doesn't exists). 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 $W + W^2 + W^3 + \ldots = (I - W)^{-1} - I$. 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 $P_{ijk} = w_{ir} * exp\left(-\alpha * Cost_k * w_{sj}/w_{ij}\right)$. 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 will be handled in more detail in the next section.
After recursive logit is introduced (section 2), it is implemented in a static assignment (section 3). When implementing it, focus is given to parameters and network characteristics that determine the utility a traveller experience. This paper 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 section 4, where a method is developed to show how recursive logit can be of use in a dynamic assignment. This method is then illustrated on a medium sized network in section 6.

2 Recursive Logit

This section 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.
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 along the path. For example the probability that traveller n uses path $1 \rightarrow 3 \rightarrow 4$ is calculated as: $P_{1,3} * P_{3,4}$.
The questions now remains on how to calculate such matrix P. For convenience, the same notation is used as in the paper of Fosgerau et al. (2013). See figure 3 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.

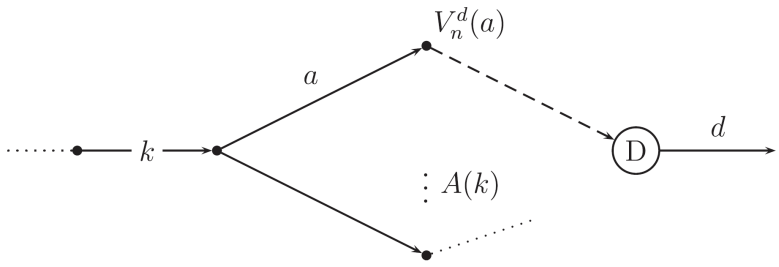


Figure 3: Illustration of notation (Fosgerau et al., 2013)

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 1). 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 consist of the travel time on link *k* and possible turn impedances. To use the properties of logit, this random term $\varepsilon_n(a)$ is assumed independent and identically distributed extreme value type 1 with zero mean. The expected utility from the traveller on link *k* is the maximum sum of these utilities, namely:

$$V_n^d(k) = E\left[\max_{a\in A(k)}\left(v_n(a|k) + \mu\varepsilon_n(a) + V_n^d(a)\right)\right] \tag{1}$$

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

$$P_n^d(a|k) = \frac{\exp\left(\frac{1}{\mu}\left(v_n(a|k) + V_n^d(a)\right)\right)}{\sum_{a'\in A}\left(\exp\left(\frac{1}{\mu}\left(v_n(a'|k) + V_n^d(a')\right)\right)\right)} \tag{2}$$

The Expected Maximum Perceived Utility function in a logit 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:

$$V_n^d(k) = \begin{cases} \mu \ln \sum_{a\in A}\left(\delta(a|k)\exp\left(\frac{1}{\mu}\left(v_n(a|k) + V_n^d(a)\right)\right)\right) & \forall k \in A \setminus d \\ 0 & k = d \end{cases} \tag{3}$$

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. If the network contains a loop, the value for the utility on

a link will depend on others and its own value. If equation 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:

$$\exp\left(\frac{1}{\mu}V_n^d(k)\right) = \begin{cases} \sum_{a\in A}\left(\delta(a|k)\exp\left(\frac{1}{\mu}\left(v_n(a|k)+V_n^d(a)\right)\right)\right) & \forall k \in A \setminus d \\ 1 & k = d \end{cases} \tag{4}$$

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.

$$M_{ka} = \begin{cases} \delta(a|k)\exp\left(\frac{1}{\mu}v_n(a|k)\right) & k \in A(k) \\ 0 & k \notin A(k) \end{cases} \tag{5}$$

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:

$$\mathbf{z} = \mathbf{M}\mathbf{z} + \mathbf{b} \Longleftrightarrow (\mathbf{I} - \mathbf{M})\,\mathbf{z} = \mathbf{b} \tag{6}$$

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.
Probabilities can now be calculated:

$$P_k = \frac{M_k \bullet z^T}{M_k z} \tag{7}$$

With • 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).

$$\left(I - P^T\right) F = G \tag{8}$$

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 *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.

3 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.
The recursive logit part in a static assignment takes place in the inner loop part of the algorithm. Outside of this loop, another loop tries to achieve equilibrium conditions. Figure 4 gives the steps of the algorithm. The algorithm is implemented in Matlab R2016b.
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 utility for link *k* to destination *d*. One column of Z is a previously calculated vector z. Equation 6 is then formulated as:

$$\mathbf{Z} = \mathbf{MZ} + \mathbf{B} \Longleftrightarrow (\mathbf{I} - \mathbf{M})\,\mathbf{Z} = \mathbf{B} \tag{9}$$

The matrix Z has the dimensions $(|A| \times |D|)$, 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:

$$\mathbf{B} = \begin{array}{l} \\ RealLinks \\ OriginConnectors \\ DestinationConnectors \end{array} \begin{array}{c} DestinationConnectors \\ \left[\begin{array}{c} \mathbf{0} \\ \mathbf{0} \\ \mathbf{I} \end{array}\right. \end{array}$$

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

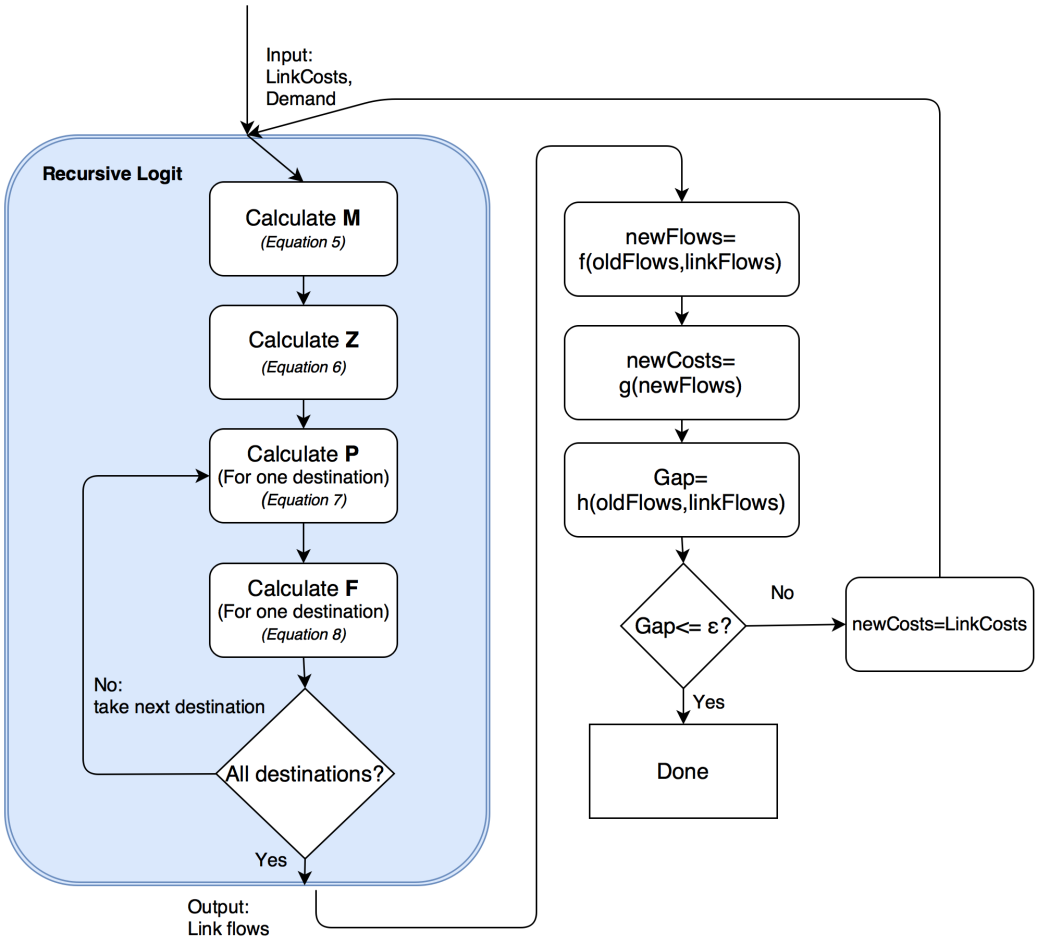


Figure 4: Overview of the algorithm used in the static assignment case

total link flows (independent of the destination), this is the output of the Recursive Logit part.
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 defines what the new link flows will be, will it be the direct outcome from the recursive logit or will a 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 route decreases. The fact that recursive logit works with a (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 a MSA step, a proportional step size is used. A proportional step size gives a fixed weight to the new 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 new calculated flows. An example of a smooth convergence of recursive logit is found in figure 5.
In this static assignment a simple BPR function is used to determine the costs on each link.
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.
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

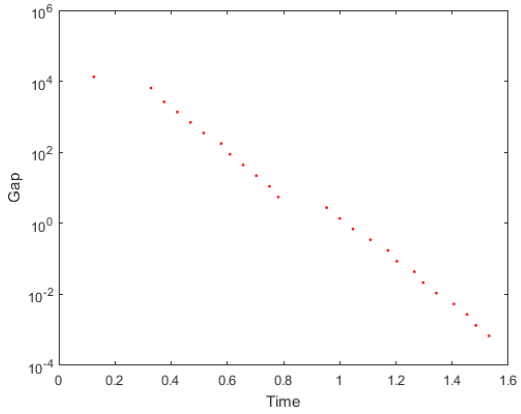


Figure 5: Convergence of recursive logit with proportional update

can be. The utility from a turn is:

$$v_n(a|k) = \beta_{TT} * TT + \beta_{UTurn} * Uturn + \beta_3 * Characteristic3 + \dots \quad (10)$$

Each characteristic has it own parameter. When calibrating, only the proportion between different beta parameters will have to be calibrated. Which means that β_{TT} can equal one for simplicity.
An example of a characteristic that can be used is a penalty for an 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 (10) 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 6). Figures 7 shows an assignment without any u-turn penalties (7a) and one with a very high u-turn penalty (7b).
The corresponding P-matrices are given in table 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 shows also that every possible path is always considered as a possibility. In table 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 al the other traffic uses a path with a u-turn. The probability lowers with every extra loop the traffic takes.

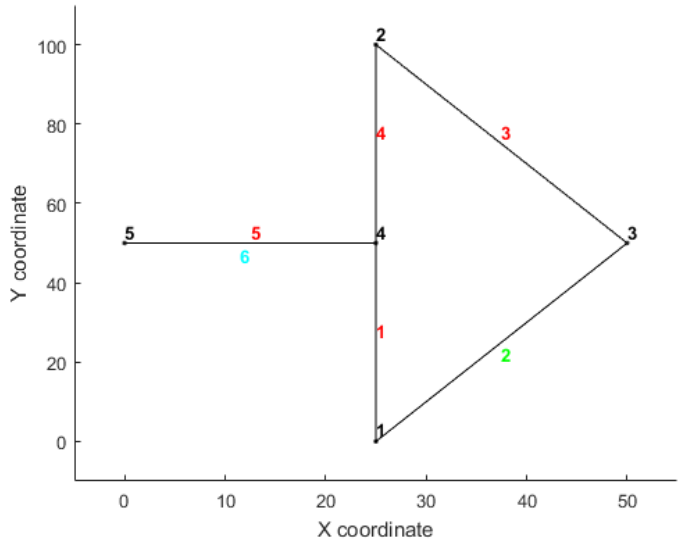
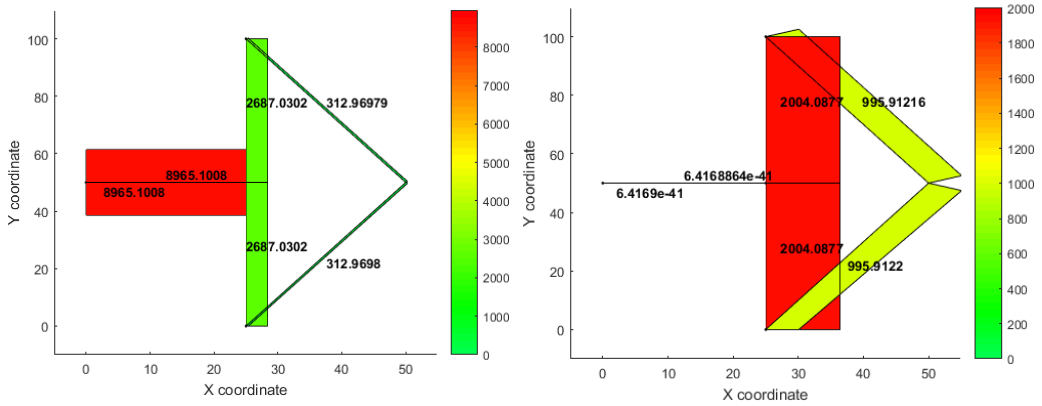


Figure 6: A simple network. Black numbers represents node numbers, while coloured ones gives the link number.



(a) Without a penalty for u-turns

(b) With a penalty for u-turns

Figure 7: Link flows on a simple network, demand is 3000 from bottom node (1) to top node (2)

Table 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 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).

4 Dynamic 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 (http://www.mech.kuleuven.be/en/cib/traffic/downloads). 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.
An overview of the algorithm is displayed in figure 8. The outer loop consist 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.
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 experienced downstream utilities. These are linked through the travel time 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 8 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.
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. An example on how to do this, assume a small port of a network, plotted in figure 9. 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. *t1*, *t2*, *t3* and *t4* 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* ($tt_a(t1)$) is exact a time step, while the travel time on link *b* ($tt_b(t1)$) is between one and two time steps at time $t1$. To calculate the value of $V_n^d(k,t1)$. Node *1* has two different outgoing links, *a* and *b*. The end of link *a* is reached at time $t1 + tt_a(t1) = t2$ when leaving link *k* at time $t1$(the same holds for b with time $t1 + tt_b(t1) = t^*$). As stated before, $V_n^d(k,t1)$ is calculated as:

$$V_n^d(k,t1) = \frac{1}{\mu} * ln\left[\left(v_n^d(k,a,t1) + V_n^d(a,t2)\right) + \left(v_n^d(k,b,t1) + V_n^d(b,t^*)\right)\right] \tag{11}$$

With $v_n^d(k,a,t1)$ being the utility for going from link *k* to link a at time $t1$. Of course not every time is represented in time steps. For time t^* that is in between $t2$ and $t3$, a linear interpolation is used. Take for example that $t^* = t2 + 0.4 * \Delta t$. This means that $V_n^d(b,t2)$ will be calculated as: $0.6 * V_n^d(b,t2) + 0.4 * V_n^d(b,t3)$. To do this, $V_n^d(b,t2)$ and $V_n^d(b,t3)$ needs to be known at the time of calculation of time step t1, which is why we proceed through time from the latest time totT in upwind order.
Once all maximum perceived utilities are calculated, the last thing remaining is

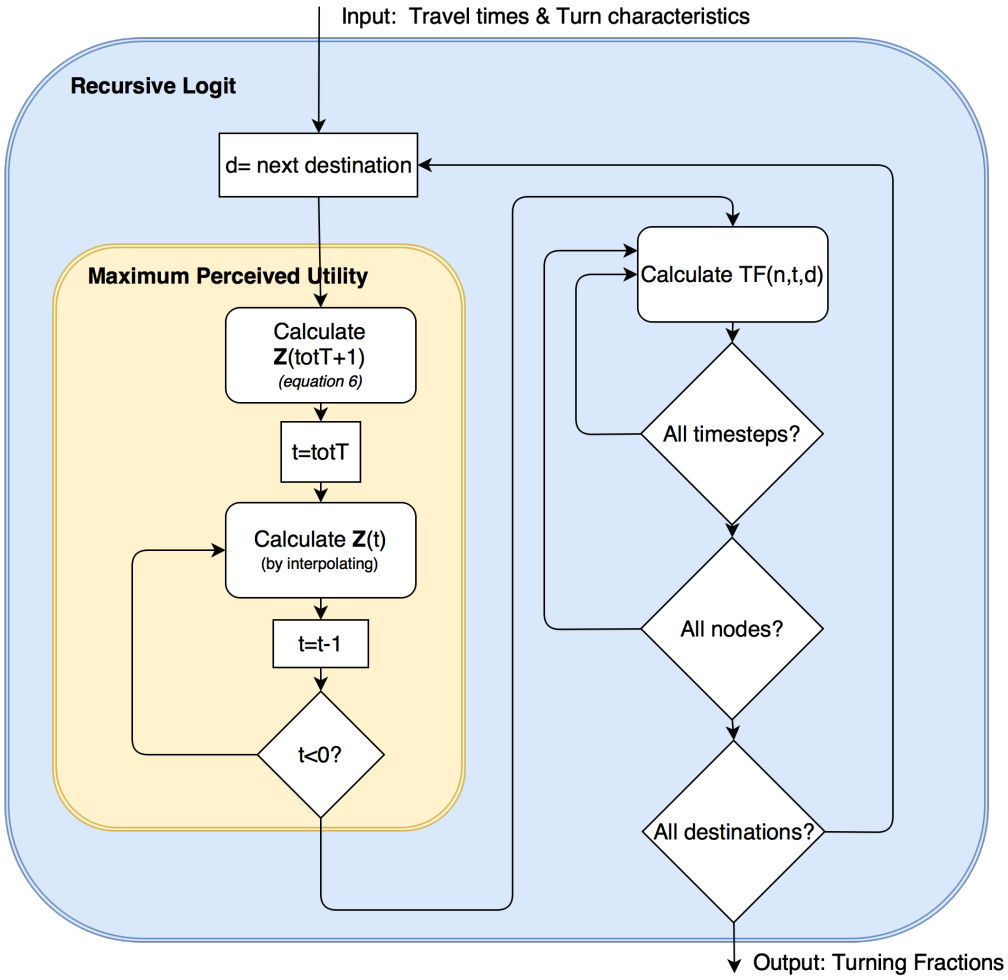


Figure 8: The algorithm used in the dynamic assignment

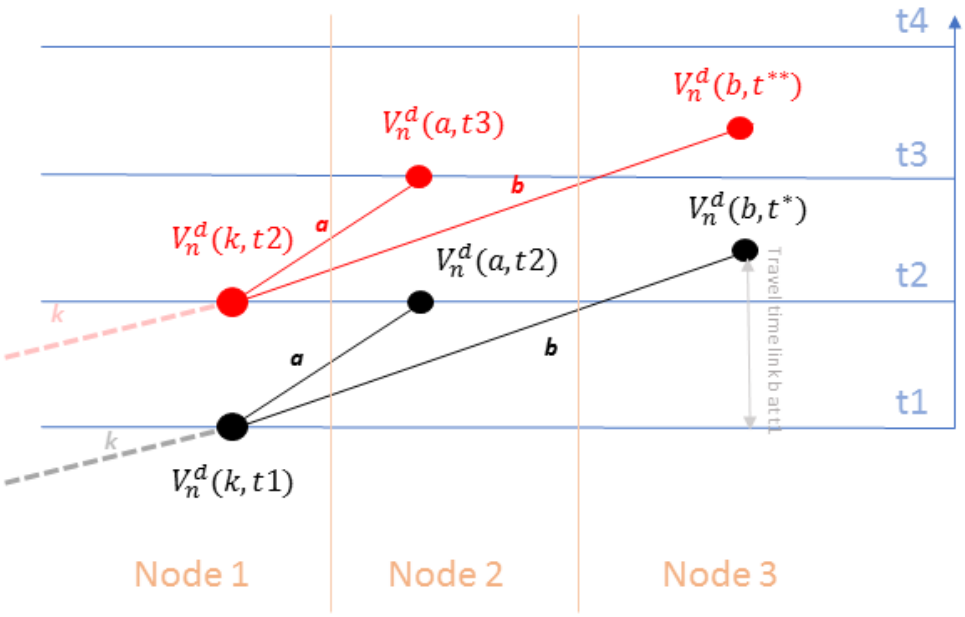


Figure 9: A simple network plotted with its time dimension. The time (z-axes) indicates the time the end of the link will be reached.

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. In this algorithm all nodes are iterated from the start time to the end time. Returning to the example in figure 9, the probability of going to link *a* (at time $t1$) is then:

$$P(k,a,t1) = \frac{exp\left[\mu\left(v_n^d\left(k,a,t1\right) + V_n^d\left(a,t2\right)\right)\right]}{exp\left[\mu\left(v_n^d\left(k,a,t1\right) + V_n^d\left(a,t2\right)\right)\right] + exp\left[\mu\left(v_n^d\left(k,b,t1\right) + V_n^d\left(b,t^*\right)\right)\right]} \tag{12}$$

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

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.

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)).

Another thing to notice is that the algorithm can be part of a warm started assignment. A warm start means, that results from a previous calculation are used as the initial condition. This way less calculations are needed if only a small part in 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.

One last thing to notice is the fact that nothing this 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.

5 Guaranteed solution

As said before, there is no guarantee that 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. An 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 consist only of links with a positive utility, then travellers will never leave the loop because they keep gaining utility.
If such singularities occurs, 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. In this research all penalties are fixed, but nothing prevents the penalty to be dynamic. If with this worst case matrix M no singularities occurs, then neither will they occur during computations where utility can only decrease.

6 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. The 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 destination. The network is plotted in figure 10.

Figure 11 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 a second iteration with (quite severe) congestion is large.

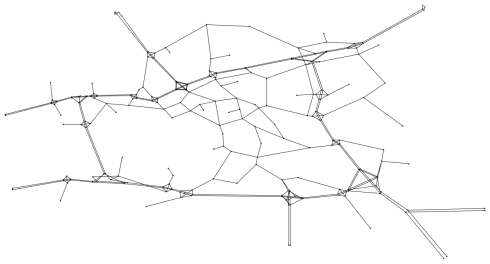


Figure 10: The network of Rotterdam

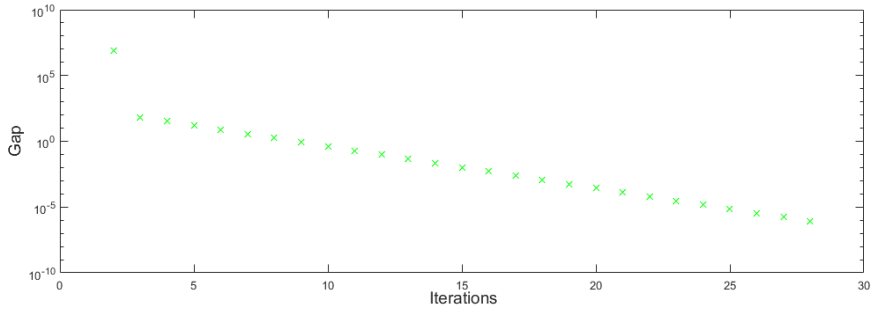


Figure 11: Convergence of the Rotterdam network

The matlab traffic toolbox of LTM makes it easy to animate the flows over the network. This way results can easily been interpreted. To show what the algorithm can do, the turning fractions for one link towards one destination are visualised over time in figure 12. 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.

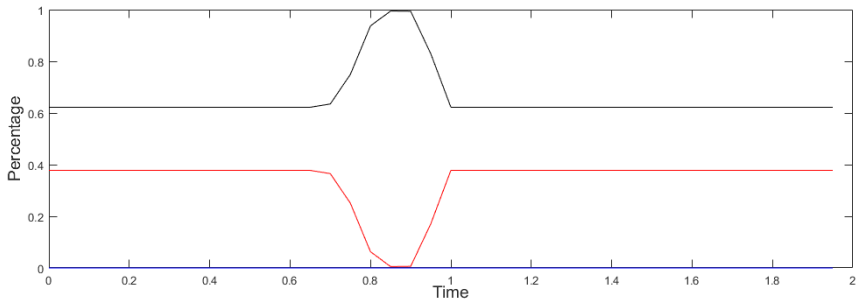


Figure 12: Turning fractions from one link towards one destination over time

7 Further research: 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 a 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.

8 Further research: larger time steps

Suppose the time steps are twice as big as in figure 9, then $V_n^d(a, t')$ will need to be interpolated between $V_n^d(a, t1)$ and $V_n^d(a, t2)$. (This is visualized in figure 13) 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 (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.

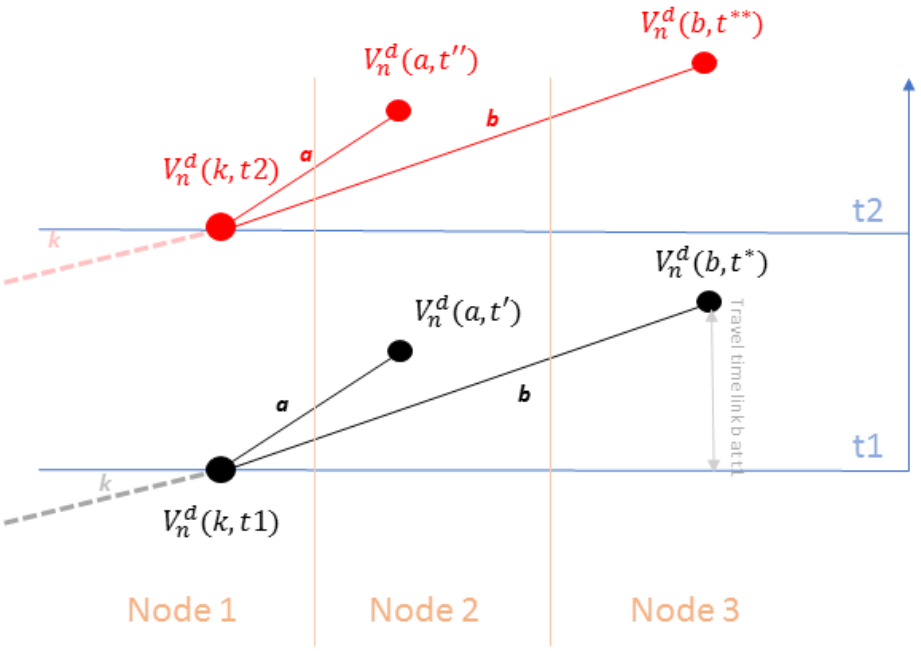


Figure 13: Same as figure 9 but with twice as big time steps.

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 the total assignments means 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 LTM Himpe et al. (2016).

9 Conclusion

Explicit route sets have some down sides, 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 infinity number) always. In this research it 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 easily 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 lower impact the more iterations are done, like an MSA step.

This paper has shown that the static procedure described by (Fosgerau et al., 2013) can be generalized 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 represents. 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.

Another obvious topic for future research is embedding the stochastic DTA with recursive logit full route set of this paper in a framework that considers departure time adjustments as well.

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 works on its own to quickly determine the turning fractions if something small changes (for example an accident on the highway witch makes the capacity drop). In this case the travel times could be handled as instantaneous.

References

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 and F. Ltd TTRV. 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.