Recursive Logit Route Choice Modelling for Traffic Assignment

Jeroen Verstraete (KU Leuven) (FWO SB PhD Fellowship)
Chris M.J. Tampère (KU Leuven)

Abstract

Recursive Logit gives not only possibilities to develop new route choice models for calibration or evaluation. The model can also be used in traffic assignment. Using the formulations as mentioned in (Fosgerau et al, 2013) to calculate the downstream utilities, however, is not efficient. First, taking the inverse of a large matrix is slow. And second, it is not able to be warm started. This paper will show how much faster an iterative procedure that can be warm started is. Another contribution of this paper is the overcoming of numeric problems that occur when working on a large network or with a small variance parameter.

1. Introduction

1.1 Problem with Route Sets

A route choice model predicts which routes people take to their destination. The model computes probabilities over a subset of all possible routes of a network that the traveller is likely to consider.

In equilibrium traffic assignment models, two ways exist to handle the route set during equilibration. Either, the route set is determined before the first iteration and remains fixed throughout iterations. Or, alternatively, the route set changes flexibly during equilibration depending on the congestion levels. Both flexible and fixed route sets have their advantages and disadvantages. A flexible route set complicates convergence to equilibrium, as not only the route choice probabilities, but also the route set need to converge. A fixed route set does not have this problem. However, it is difficult to define a priori (i.e. before congestion levels at equilibrium are known) which fixed subset of routes is attractive to the traveller and whether all relevant route alternatives are included.

A fixed *full* route set containing *all* existing route alternatives avoids both problems. For example, Recursive Logit (Bell, 1995; Fosgerau, Frejinger and Karlstrom, 2013) considers an implicit full route set for the estimation of route choice models. (Verstraete, Himpe and Tampère, 2017) integrated Recursive Logit as a route choice model in a static or dynamic traffic assignment. The model calculates for each turn the probability of being chosen while traveling towards a destination.

1.2 Assignment with Full Route Set

In (Verstraete, Himpe and Tampère, 2018), we showed how the Recursive Logit can be adapted to be used in a Dynamic Traffic Assignment. There are two main parts in this model. With the link costs as an input, the first part is to calculate the downstream utilities of each link (at each time). Once these are known, the second part is to calculate the turn probabilities of each link (at each time).

This model has some downsides concerning numerical errors, scalability and efficiency. Using the formulations as mentioned in (Fosgerau, Frejinger and Karlstrom, 2013) for calculating the downstream utilities is not efficient. First, it involves inversion of a large matrix, which is slow (for formulas, see below). And second, it is not able to be warm started. Warm starting the method, which means it is initialized the state with a previous result, has two advantages. First, during the iterations of an assignment, each iteration can be warm started with the result of the latest iteration. This makes especially the last iterations much cheaper as the solution point is close to the previous one. Second, when a traffic assignment is used in practice, multiple similar variants (e.g. infrastructure, demand, or control scenarios) of a network are evaluated. It is logical to expect the final solution point of a variant to be not far from the base equilibrium point. If each variant is initialized with the base equilibrium point, fewer iterations will be needed.

This paper will show how much faster an iterative procedure is when it is warm-started. Another contribution of this paper is the overcoming of numeric problems that occur when working on a large network or with a small variance parameter.

Section 2 will give an overview of the normal formulations, while section 3 gives the modifications. Section 4 will compare the performance of the different methods. Section 5 gives an insight into further work and Section 6 gives some conclusion.

2. Formulas

2.1 Recursive Logit

The expected downstream utility at the end of a link is calculated recursively in the same way as in Dial's algorithm (Dial and Voorhees, 1971), namely:

$$V_k^d = E[\max_{a \in A_k}(v_{ka} + \mu\varepsilon + V_a^d)] \quad (1)$$

With v_{ka} the utility for going from link k to a; ε a random utility component; μ a scale parameter; V_a^d the expected downstream utility at link a, and A_k the set of links that are accessible from link k. The Expected Maximum Perceived Utility function in a logit model can be analytically solved as a logsum. Equation 1 then becomes:

$$V_k^d = \mu \ln \sum_{a \in A_k} \exp(\frac{1}{\mu}(v_{ka} + V_a^d)) \quad (2)$$

To write the system of equations in matrix notation, three matrices are introduced: $\boldsymbol{b}$ ($|A| \times 1$), $\boldsymbol{z}$ ($|A| \times 1$) and $\boldsymbol{M}$ ($|A| \times |A|$). With $\boldsymbol{b}$ being a vector with $\boldsymbol{b}_k = 0$ for $k \neq d$ and $\boldsymbol{b}_d = 1$, $\boldsymbol{z}$ a vector that gives the destination-dependent transformed utility of the end node of a link: $\boldsymbol{z}_k = \exp(\frac{1}{\mu}V_n^d(a))$. $\boldsymbol{M}$ is a matrix of transformed utilities for each turn. $\boldsymbol{M}_{ka} = \exp(\frac{1}{\mu}v_{ka})$ Then, equation 2 becomes:

$$\boldsymbol{z} = \boldsymbol{M} \cdot \boldsymbol{z} + \boldsymbol{b} \quad (3)$$

This is solved by:

$$(\boldsymbol{I} - \boldsymbol{M})\, \boldsymbol{z} = \boldsymbol{b} \tag{4}$$

With $\boldsymbol{I}$ the identity matrix. This equation shows that in principle, $\boldsymbol{z}$ can be directly solved by inversion of the matrix $(\boldsymbol{I} - \boldsymbol{M})$. Alternatively, the system of equations can easily be rewritten and solved in the iterative form:

$$\boldsymbol{z}_i = \boldsymbol{M} \cdot \boldsymbol{z}_{i-1} + \boldsymbol{b} \tag{5}$$

2.2 Probability Calculation

Calculating the probability once the downstream utilities are known is straightforward:

$$\boldsymbol{P}_{ka} = \frac{e^{\frac{1}{\mu}(v_{ka}+V(a))}}{\sum e^{\frac{1}{\mu}(v_{ki}+V(i))}} \tag{6}$$

Or according to (Fosgerau, Frejinger and Karlstrom, 2013) in the transformed form as:

$$\boldsymbol{P}_k = \frac{\boldsymbol{M}_k \circ \boldsymbol{z}^T}{\boldsymbol{M}_k \boldsymbol{z}} \tag{7}$$

With $\boldsymbol{M}_k$ row *k* of matrix $\boldsymbol{M}$.

3. Modifications

In this section, the modifications made are discussed. It will be shown that for large networks, the theoretical solution of the previous section is inefficient and can become numerically unstable. The main contribution of this paper is selecting the right set of modifications that resolve these issues such that the overall problem can be used efficiently in larger networks.

3.1 Downstream Utilities: Warm starting and flag mechanism

To start the procedure of Recursive Logit in the iterative form, there should be an initial $\boldsymbol{z}_0$. This can come from a previously calculated scenario, hence enabling warm start.

The iterative structure of the problem also allows smarter updating schemes. Equation 5 still shows *z* as a vector that is updated as a whole in every iteration. When one part of the network is close to equilibrium, but another part is still far off, it makes no sense to repeat already converged calculations for the first part. A flag mechanism is proposed to only update the links where one of their dependencies have significantly changed. This means when the problem is warm-started, also a vector indicating which parts of the network have been changed is given as an input. From there on, only significant changes to the values will propagate through the network until convergence. The idea of a flag mechanism is simple, but the outcome is very important as small scenario changes can be calculated fast (Himpe, Corthout and Tampère, 2016). Equation 5 then becomes:

$$\boldsymbol{z}(flag)_i = \boldsymbol{M}_{flag} \cdot \boldsymbol{z}_{i-1} + \boldsymbol{b} \tag{8}$$

With $\boldsymbol{z}(flag)_i$ only the $\boldsymbol{z}$ values that are flagged and $\boldsymbol{M}_{flag}$ the corresponding rows of matrix $\boldsymbol{M}$ for the flagged links.

When there are no initial values for $\boldsymbol{z}_0$, which means the algorithm is 'cold'-started, we found that the method performance best when it was initialized with every downstream utility as negative infinity except from the destination, which gets a downstream utility of zero. (Resulting in respectively $\boldsymbol{z}$ values of 0 and 1.) This results in a 'wave' of flag activations going over the network, starting from the destination. The overhead of the flag mechanism is in this case larger than the benefit as each link will be flagged at least once and possibly multiple times. There may exists better initializations.

3.2 Numerical errors

In both subparts, downstream utility and probability calculation, there is a problem with the current formulations in section 2 regarding to numerical errors. It depends on how small a number can be represented in the software. E.g. in Matlab 2019 on a normal computer, the smallest positive number one can represent is 10^{-324}. Representing this number as an exponential becomes: e^{-745}. Calculating the exponential of any number smaller than -745 becomes zero. This is a problem in the calculations as the matrix $\boldsymbol{M}$ or vector $\boldsymbol{z}$ can no longer be constructed or the probabilities (equation 7) can no longer be calculated.

Even if matrix $\boldsymbol{M}$ and vector $\boldsymbol{z}$ are calculable, the result of equation 4 can still be so small it cannot store the correct value and rounds it to a zero. A $\boldsymbol{z}$ value of zero represents an infinite cost to reach the destination.

Instead of making matrix vector multiplications and summations of the transformed utilities, the formulas can be adjusted to work directly with the utilities. There are two operations that need to be adjusted. Note that for the result of an operation, we are only interested in the exponent value.

3.2.1 Exponential product

$$e^{product} = e^{a} * e^{b} \tag{9}$$

This is straightforward as:

$$product = a + b \tag{10}$$

The same holds for division:

$$e^{division} = \frac{e^{a}}{e^{b}} \tag{11}$$

$$division = a - b$$

3.2.2 Exponential sum

$$e^{sum} = e^{a} + e^{b} \tag{12}$$

To rewrite this, an intermediate step is needed. Note that in our model a and b represent utilities which are always negative (or zero for the destination).

$$max = \max(a, b) \tag{13}$$

$$sum = max + \log(e^{a-max} + e^{b-max}) \quad (14)$$

By isolating the maximum exponent, the formula is always calculable. If e^{a-max} is not calculable, it means that the difference between the exponents is so big, it has no influence on the sum of the exponentials. Note that the log of the sum of the difference in utilities (second term in equation 14) is very small. One could even argue to only compute e^{a-max} if the result has influence on the significant numbers of the result ($|a - \max| < 33$), but checking this appeared to require more time than calculating it.

3.2.3 Probability

As equation 6 only consists of sums and one division, it can be reformulated such that it can be calculated. But there is a catch. When the normal formulations do not work anymore, it means that we have large exponents. When doing the division, there will be a subtraction between two large numbers where one is the maximum utility with a small addition (the log of the sum of the difference in utilities) and the other is potentially the maximum utility. This is very unstable for numerical errors. Therefore, the formulations to calculate the probability have a twist. Instead of doing the operations separately in sequence, one can make a new operation:

$$P_a = \frac{e^a}{e^a + e^b} \quad (15)$$

$$max = \max(a, b) \quad (16)$$

$$P_a = e^{a-max+\log(e^{a-max}+e^{b-max})} \quad (17)$$

In this way, the small difference in utility due to the log of the sum of the difference of utilities is not lost in the calculation. It only becomes irrelevant if $a \ll max$, but for $a \cong max$ it is crucial.

4. Downstream Utility Performance

There are five different methods described in this paper to calculate the downstream utility.

1. Normal Recursive Logit (RL)
2. Iterative Recursive Logit (IRL)
3. Iterative Recursive Logit with flags (IRLF)
4. Iterative Scaled Recursive Logit (ISRL)
5. Iterative Scaled Recursive Logit with flags (ISRLF)

RL solves the system of equations by taking the inverse of a large matrix. IRL solves it in an iterative procedure. ISRL also solves it iteratively but reformulates the formulas such it can be calculated with big negative values. IRL and ISRL both update all values in each iteration. IRLF and ISRLF use a flag mechanism to only update those values that have a dependency that changed.

There are four different kinds of scenarios that are of interest. First, one assignment iteration on a cold start on an empty network. Second, a full assignment which starts from an empty network, but each iteration is initialized with the previous iteration. Third, a full assignment which starts from the second scenario. In this scenario the network is slightly changed. Fourth, and last, a full

assignment where the variation parameter is chosen so small, the exponents become too large to be calculated in the normal formulations.

Note that it does not matter which method is used for the stability or convergence in the assignment itself as each method gives the same downstream utilities/probabilities as long as they can be calculated.

The network used in the scenarios is the network of Leuven, visualised in Figure 1. It consists of 2128 links and 154 origins/destinations.

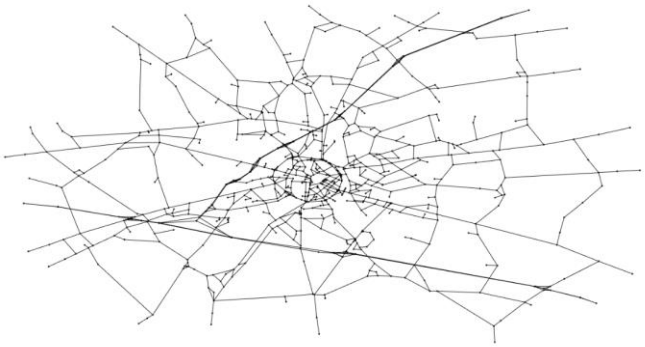

Figure 1 Network of Leuven

4.1 Cold Start

For this scenario, the first iteration of an assignment is calculated starting with an empty network and thus free-flow travel times. The downstream utilities are calculated for each destination.

Table 1 Cold Start Scenario: Computation Times

Method	*Total Time (seconds)*	*One call (seconds)*
RL	23.3166	0.1514
IRL	0.4250	0.0028
IRLF	5.9213	0.0384
ISRL	22.7668	0.1478
ISRLF	16.5628	0.1076

Table 1 shows that the normal iterative methods (IRL,IRLF) perform better than the non-iterative method (RL). The fastest method, which is ~50 times faster than RL, is the iterative formulation without any scaling or flag mechanism (IRL). As the methods are cold-started, every downstream utility of a link needs to be adjusted at least once (in the iterative formulation). This results in the overhead of the flag mechanism to be greater than the benefit of only updating a part of the links. Because the formulation that allows big exponent is not written in matrix vector multiplications, but goes over every link, the flag mechanism makes it slightly faster. Therefore, there is no further interest in ISRL as warm-starting the problem will ISRLF make even more faster.

4.2 Full Assignment Cold start

For this case, we did a full static assignment on the Leuven network. Link costs are calculated by the well-known BPR-function (States, 1964). Due to the stability of the route set and characteristics of the Leuven network, a proportional step size of 0.5 is used. (Note that this is not generally the case, but the greater stability allows for bigger step sizes and fewer iterations, see (Verstraete, Himpe and Tampère, 2017) for more information.)

As stated above, the different methods will all produce the same downstream utilities resulting in the same number of iterations. The convergence of the assignments is visualized in Figure 2. What this case study tries to show is the different computation time needed for calculating the downstream utilities as some can be warm started and have a flag mechanism. Table 2 shows the computation times for each method, excluding computation time for calculating the probabilities and resulting link flows as these calculations are the same. Each method is called 3080 (=20*154) times during the whole assignment.

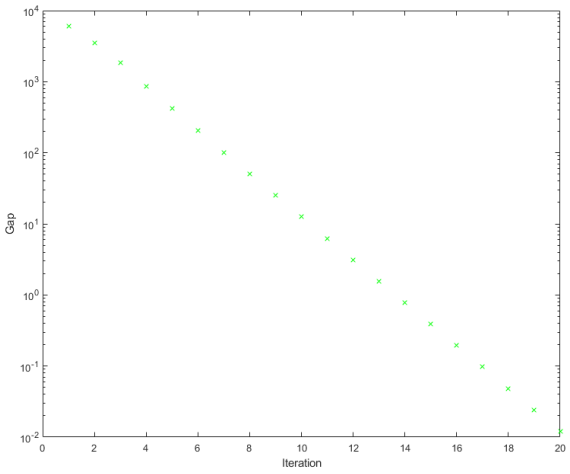

Figure 2 Convergence plot

Table 2 Full Assignment Case: Computation Times

Method	*Total Time (seconds)*	*One Call (seconds)*
RL	431.593	0.14012
IRL	2.024	0.00066
IRLF	1.426	0.00046
ISRLF	266.635	0.08657

Table 2 shows the corresponding computation times for calculating the downstream utilities. Two interesting facts are shown here. First, the average computation time for one call has gone down for all iterative methods. This is due to the fact that each iteration is now warm-started. Second, the methods with a flag mechanism are now performing better than their counterpart without flag mechanism.

IRLF is now performing ~300 times faster than RL, increased by a factor of 6 by using a flag mechanism. A total best performance will be achieved when for the first iterations no flag mechanism is used while for the later iterations a flag mechanism is used, when the total assignment is cold-started.

4.3 Full Assignment Warm start

Starting from the results of a previous assignment, the network is now slightly changed. For this example, we decrease the maximum speed on the highway to 100km/h instead of 120km/h. Only the normal formulation (RL) is not able to be warm-started. As the other methods start from a point in the solution space that is close to the final equilibrium, it takes them less iterations. This leads to less method calls and thus faster total computation time.

The flag mechanism shows it true potential when the network size is large and the variation on the network is small compared to the used starting equilibrium.

4.4 Small Variance Case or Large Network

When we set the variance parameter (μ) to be very small, the values of the z vector and M matrix becomes incalculable. This means that the first three methods (RL, IRL and IRLF) do not produce a useful result as they assume the destination not reachable or only by very close origins.

ISRL and ISRLF are not only useful when the variance parameter is small. When the network is very large, the downstream utilities can become so negatively large, the calculations round it to negative infinity which means that the destination is not reachable. Even with these special formulations, the methods are faster than calculating the inverse of a matrix.

5. Future work

The original recursive logit does not handle correlation between different alternatives (i.e. it implements a multinomial logit route choice model). However, (Mai, Fosgerau and Frejinger, 2015) developed the Nested Recursive Logit. This model uses different variance parameters for each link, representing the correlation among them. To be able to do this, they use an iterative solution scheme. This model could be easily implemented in the four different methods to calculate the downstream utility. This will be done in future work.

Another point for future work is the fact that while calculating the downstream utilities iteratively, there is information gained by the method to identify cycles that attract too much probability (see (Verstraete and Tampère, 2019). Or even in the worst case, make the matrix singular which would also mean that the iterative procedure would not converge. By calculating the downstream utilities iteratively, one can calculate the influence a value has on its own. We want to use this in the first iteration of an assignment to close certain turns on cycles, such that this influence is no longer existing and the singularity is removed.

6. Conclusions

By changing the formulations of the problem, the method to calculate the downstream utilities becomes much more efficient, about 300 times faster for cold starting an assignment and even more for a warm-started assignment. The new formulations make it also possible to work on larger

networks. It also increases the feasible solution space of the variance parameter due to overcoming numerical errors.

These modifications make the Recursive Logit idea a real possible route choice model in a static or dynamic traffic assignment. The stable route choice made convergence smooth and limited the iterations needed. With these modifications, each iteration is even faster.

Further research will eliminate cyclic routes from the implicit, complete route set that distort the solution of the Recursive Logit model. After this, a fair comparison between existing route choice models and the recursive logit idea can be made.

7. References

Bell, M. G. H. (1995) 'Alternatives to Dial's logit assignment algorithm', *Transportation Research Part B.*

Dial, R. B. and Voorhees, A. M. (1971) 'PROBABILISTIC MULTIPATH TRAFFIC ASSIGNMENT WHICH OBVIATES PATH ENUMERATION The problem', *Transpn Res*. Pergamon Press, 5, pp. 83–111.

Fosgerau, M., Frejinger, E. and Karlstrom, A. (2013) 'A link based network route choice model with unrestricted choice set', *Transportation Research Part B: Methodological*, 56, pp. 70–80.

Himpe, W., Corthout, R. and Tampère, M. J. C. (2016) 'An efficient iterative link transmission model', *Transportation Research Part B: Methodological*, 92, pp. 170–190.

Mai, T., Fosgerau, M. and Frejinger, E. (2015) 'A nested recursive logit model for route choice analysis', *Transportation Research Part B: Methodological*, 75, pp. 100–112.

States., U. (1964) 'Traffic assignment manual for application with a large, high speed computer.' Washington: U.S. Dept. of Commerce, Bureau of Public Roads, Office of Planning, Urban Planning Division; for sale by the Superintendent of Documents, U.S. Govt. Print. Off. Available at: file://catalog.hathitrust.org/Record/000968330.

Verstraete, J., Himpe, W. and Tampère, C. M. J. (2018) 'Route Choice Equilibration Algorithm for Stochastic Dynamic Traffic Assignment with Full Route Set'.

Verstraete, J., Himpe, W. and Tampère, M. J. C. (2017) 'Stochastic User Equilibrium with Full Route Set in Dynamic Traffic Assignment', in Cools, M. and Limbourg, S. (eds) *Proceedings of the BIVEC-GIBET Transport Research Days 2017: Towards an Autonomous and Interconnected Transport Future*, pp. 373–387.

Verstraete, J. and Tampère, C. M. J. (2019) 'Stochastic Route Choice Modelling with an Implicit Acyclic Full Route Set'