EFFECTS OF JUNCTION DELAY FUNCTIONS ON TRAFFIC MODELLING USING LARGE-SCALE STRATEGIC TRAFFIC MODELS

Martijn van Arem

GOUDAPPEL COFFENG

15-5-2019 until 24-7-2019

UNIVERSITY OF TWENTE.

mobility consultants Goudappel Coffeng

Preface

This report was written with the goal to get an indication of the possibilities of junction delay functions for large scale strategic traffic models. The focus of the report would thus be to see if the convergence of the model would increase when these functions would be implemented.

The research for this report was done during the bachelor assignment of Martijn van Arem.

I would like to like to thank Lissy La Paix Puello and Bastiaan Possel for the guidance and feedback they have given during and around the 10 weeks in which this report came together.

Table of contents

Summary of the report

This report will investigate the effects of different methods of junction modelling on the convergence of large-scale strategic traffic models. The main hypotheses would be a more simplistic approach, which would have less variables, could be able to increase the rate of convergence of strategic traffic models. A promising option could be Junction delay functions, as they only have the rough layout of the junction and traffic volumes as variables.

To see what the effects of junction delay functions would be on the convergence. The main question is formulated to be: *To what extent could junction delay functions increase the convergence of large-scale strategic traffic models, while reducing the amount of input and retaining a high accuracy in comparison to the old method?*

While answering this question, the first part will investigate different methods that could be used to describe the delays or time penalties that vehicles get when using the junction. As a result, three methods will be compared to one another (two focussed on junction delay functions and one that is compared to as the original method). These methods all have a different approach to junction modelling.

The first method would be the method of Vasvári (Vasvári, 2015). This method works with a single equation and a set of parameters to describe each of the junctions. The delay curve of this equation set looks flat with low increase in delay until the capacity of the junction is reached. At this point the function gets an exponential growth. This gives a large delay when the junctions capacity is reached.

The second method has a different approach. This method has different equations for signalized and unsignalized junctions (Aashtiani & Iravani, 1999). This method has a maximum delay value of 18 seconds for an unsignalized junction, which is a low value. This implies that the unsignalized junction always has a low delay time for these types of junctions. While for the signalized junctions, this method gives the roughly equal delays to the other two methods. This method gives all types of junctions a slowly increasing delay, where the delays would also increase if the traffic loads relatively low.

The third method is already implemented in OmniTRANS (DAT.Mobility, 2016), which would thus be the one that is compared to. This method is different in the fact that the other two have volume factor as variable, whereas this method also has a capacity reduction due to the other traffic directions. This means that there are more complex calculations involved.

After these three methods were described, they were tested to see which of the methods would have the best convergence. This was done in the modelling program OmniTRANS, where the tutorial network of Delft was used, as for this purpose a not too detailed network would satisfy.

When the testing of the model was done, the results of 50 iterations of the network for each method were compared to each other. This showed that, unfortunately, the currently implemented version of junction modelling was the method that had the highest rate of convergence. Not all was lost, as the second method was not too far behind regarding the convergence and the eventual difference in load between iterations.

This means that the expectations at the start of the research were not met. While this would not imply that there is no opportunity that a similar method could not achieve a higher convergence. As the methods that were tested were developed for Hungary and Iran systems respectively. While the test that were done were done on a Dutch traffic system. This means that further research could

very well give adjustments to these new methods, by calibration and validation, to give them the higher convergence rate that was expected in this report.

Introduction

With traffic modelling becoming a common tool to use for mobility consultants, the used models are getting more precise. While increasing in accuracy, the models are also getting more complex. This would mean that within the strategic (static) large-scale models used for the prediction of future traffic flows, the convergence of the model is decreasing. This loss of convergence is due to the increasing complexity of the methods that are used for predicting travel time delays. To increase the convergence again, junction delay functions might give a possibility to simplify some of the calculations that are needed to reach an equilibrium in the model.

For this the research will investigate the effect of junction modelling on the convergence of the model. This could be described as the amount of iterations needed to get the traffic assignment to an equilibrium. This could make for faster results when working with the large-scale traffic models.

Research questions

To make sure that the goal of the research would be completed, research questions were formulated to give a guide. These questions will therefore give a rough structure for the report and will also provide goals to accomplish, while working towards an answer to the main question:

To what extent could junction delay functions increase the convergence of large-scale strategic traffic models, while reducing the amount of input and retaining a high accuracy in comparison to the old method?

To reach the answer to this research question, the sub-questions were defined which were divided into three categories. The first of the questions is aimed towards the literature side of the research. Which means, looking into the definition of junction delay functions and possible versions that already exist. The second part is aimed towards implementing these methods into software and analysing the results. Thirdly, the traffic convergence and traffic assignment of the different methods will be compared.

SQ 1: What equations are in use for the modelling of junction delays when using the method of node delay functions?
SQ 2: What are the effects on the model when the junction delay functions are being used?
SQ 3: In what way do the analysed methods compare to the current method of junction modelling in OmniTRANS, looking at the convergence and traffic assignment?

The answers to these questions will be given in het main part of the report. SQ 1 will be answered in the chapter literature research. SQ 2 will be spread out over the chapters of the methodology and results. The last sub-question will be covered in the results section, as this is more of a discussion of the outcomes of the new methods.

Literature review

This chapter will be divided into two parts. The first will mainly focus on giving some background information about the way traffic modelling is done and at which level the junction delay functions would come into the model. The second part has the focus more on the different types of junction modelling practises there are and will also give some possible advantages and disadvantages of the different types of modelling.

Background information

Modelling of traffic is a complex undertaking, which has loads of factors to determine routing of vehicles. To gain some more information about the main factors in the routing behaviour in strategic traffic models, some background information about the influence of time and cost functions in the static traffic models was needed. For this, background research was done towards volume delay functions and the choices that are made due to costs of certain routes.

The different generations of choice models almost all focus on the time that is needed to travel over a certain route (Prasker & Bekhor, 2004). The different types of calculations operate from the same basic principle: that individual routes should have the lowest cost possible. As for a large network, there is a vast variety in choices possible. This would take a long time to calculate, were it not for stochastics. A mathematical method to reduce the amount of calculations to be done so the calculation time would not be too long.

In the beginning, the different methods that were used had quite some errors associated with them, meaning that the routes in the model would not per se represent the actual situation on the roads. For this to be solved, new methods were formulated. These methods were increasingly more accurate to the state where they are now. The method in use in the main steam of traffic modelling and route choices would be the stochastic user equilibrium (Prasker & Bekhor, 2004). But due to the calculation time of this, it would not be the preferred way of modelling. Within OmniTRANS the choice is most of the time made to use the method of Volume Averaging, which saves in calculation time. Therefore, this will become the method against which the junction delay functions are compared.

Next to this, there are more variables which would influence the routing decisions, as everyone has their personal preferences for the route to make their daily travels. These preferences have less flexibility than the travel time of the routes. This means that time would be the major factor in route choice decisions. Therefore, the focus of the delay functions should be the increase of travel time at a higher volume of traffic. Luckily, this is also the way that route costs are formulated in the software and thus will it be possible to implement the new methods directly (DAT.Mobility, 2016).

So, it could be said that the travel time of journeys is the main factor which influences the route choices, which will underline the importance of the delays of junctions, meaning that they must represent the real-world situation in a good manner.

Review of junction modelling methods

For modelling of junction delays there are a couple of options. In this paragraph the (dis)advantages of three junction modelling methods will be described. These three modelling methods are the methods that are used in the research. The choice for these three was made because they all have a different approach to junction modelling.

For example, Vasvári (2015) uses for his formulas a set of parameters to alter the results of his delay function to increase or decrease the delay value of the various junction types, while having the same

main function. This gives an almost non-increasing function until the capacity of the junction is reached. Here the delay will increase exponentially, giving large costs to the junction if the capacity is exceeded.

This method was created and verified for a Hungarian traffic system, so it is to be expected that it will not have a one on one fit with the Dutch model. Still there is an expectation that the resemblance will be close enough to get results which are close to the real-world situation.

The second method that will be investigated, is the method of Aashtiani and Iravani (1999), who have created a difererent method of implementing junction delays functions into a traffic model. These functions have quite a different way to calculate the delay than the one mentioned above, as there are no parameters to adjust the range of the function.

For this method this is not nessesary, as the method has a more detailed way of describing the different types of junctions (Aashtiani & Iravani, 1999). The method is spilt into two different parts, the first of these parts would describe unsignalized junctions. The unsignalised junctions would include both priority and equal junctions. This is due to the fact that this method is created for Iran, where the driving style is different than in the Netherlands.

The second set of equations is designed to calculate the delays of the signalized junctions. This set has multiple functions to estimate a cycle time, red time of a junction leg and the total delay due to the signalized juctions (Aashtiani & Iravani, 1999).

The shape of this delay function is a more uniform increase. A drawback is that the maximum delay for unsignalized junction is just 18 seconds, which is a low value compared to the maximum 300 seconds that is currently implemented in OmniTRANS (DAT.Mobility, 2016). This might give problems for the traffic assignement, but could also pose a opportunity to increase the rate of convergence to the final equilibirum.

The third type of Junction delay modelling which is looked into, is the method that is already in use in the modelling software OmniTRANS (DAT.Mobility, 2016). This method will be the most extensive of the three. It is chosen to be compared to as this is the method that is implemented in OmniTRANS as the default junction modelling method.

This method uses a large set of equations, which will also take the cross traffic into account, as well as the cycle time of the signalized junctions and queue delays (DAT.Mobility, 2016). The cross traffic will give a new capacity for the junction. This would be altered a coulple of times to compensate for the traffic that is going over the junction so it would not exceed the maximum capacity. Next to this, queues are also formed to compensate for this use of less traffic at the junction. These queues will also give delays for the different junction types.

Added value

While looking for possible implementations of the different types of junction delay functions, it was not found that this was done in a certain type of traffic modelling program. Due to this, the focus of this research would be to implement the found methods and see what the effect would be on the convergence of the model. This could decrease the amount of iterations needed to reach an equilibrium within the traffic model in comparison with the current method. This is needed to make sure that when using the model for traffic predictions, the differences between variants are due to the changes made in the model and not because the model had not fully reached an equilibrium. The main gain here would thus be the lowered amount of iterations needed which would most likely reduce the time needed to get to the stable situation.

When using junction delays, they will change during each iteration. This is due to the differences in load that occur until the equilibrium is reached. Therefore, the junction modelling is one of the major influences on the convergence of the traffic model and with the lowered number of parameters that could influence the delay output from the junctions. The expectation is that the model would converge faster compared to the current method of junction modelling in OmniTRANS. This is expected as to the differences in delay costs between iterations might become lower when there are fewer changing parameters involve. This gives a better indication of what would be the most optimal routing choice.

With this possibility of faster convergence, the modelling of traffic could be done in less time and when the accuracy of the model is still high, the junction delay functions could give a method that would increase the value of the large-scale strategic traffic models. As the time needed could be less compared to the current method where a large amount of inputs is required for junction modelling. This might thus provide a possibility to decrease the amount of time needed to get an indication of the traffic loads in the system, due to the faster convergence and therefore less calculation time needed.

Methodology

Description of junction modelling methods.

As described different options were investigated, the two literature methods have different equations for the modelling of junction delays. The equation used for the first of the possible methods would therefore be the equation that are created by Vasvári (2015). This method has the advantage that one equation describes all the different possibilities as it makes use of a changing set of parameters. This would increase the adjustability of the equation, thus if one would like to change the delay values to fit their traffic situation, the formula could be altered easily after for example, traffic observations.

The used equation will look like the following: (1)

$$t(v) = \varphi_1 \cdot d \cdot \left(\varphi_2 \cdot \sqrt{\alpha^2 \cdot \left(\varphi_3 - \frac{v}{n \cdot c}\right)^2 + \beta^2} - \alpha \cdot \left(\varphi_3 - \frac{v}{n \cdot c}\right) - \beta \right) \quad (1)$$

Where:

t(v) = the delay time due to the traffic volume (sec)
n = the number of lanes in the link (-)
d = the length of the link (m)
v = the traffic volume (passenger car equivalent (PCE))
c = the maximum capacity of a lane (PCE)
ϕ_1, ϕ_2, ϕ_3, α, β= parameters for a better adjustability (-)

The different parameters that were determined by Vasvári's research will be stated in Appendix A. These are not altered for the testing of the junctions or the network, making the assignment less accurate, as they were determined for the Hungarian traffic system.

The second literature method that is tested has as set of equations to estimate junction delays. These equations describe the signalized junctions and unsignalized junctions in different ways. This gives a different approach on modelling of junctions and could thus give a different rate of convergence while using the same model.

The equations for the signalized junctions of the second method are based on the Webster formula (Webster, 1958). This formula is tested and would give a good representation of the delay that is induced by traffic lights. Thus, the first equation of the set (Formula 2) gives an estimation on the cycle time of the traffic signals, this would therefore give that the timing of the signals is always a fixed cycle. This cycle is therefore determined by the types of roads leading to the junction. The second formula (Formula 3) gives the red time of the link leading to the junction. These two formulas will be given next:

$$C_j = 1 + \left(\frac{\sum_{i \in s_j} W_{ij}}{8} \cdot \frac{|s_j|}{4} \right) \quad (2)$$

Where:

C_j = the cycle time on the junction j (minutes)
s_j = the set of links entering the junction (-)
W_{ij} = the weights of the links towards the junction (-)
With the following values:
2, if the link is a local road
3, if the link is a minor trough road
4, if the link is a major trough road
5, if the link is a express way

Which is in turn used by Formula 3:

$$r_{ij} = 1.2 \cdot C_j \cdot \left(\frac{|s_j| \cdot W_{ij}}{2 \cdot \sum_{s \in s_j} W_{ij}} \right) \quad (3)$$

These timing formulas for the traffic signals are needed to give an estimation of the average delay that could be found for the different sizes of signalized junctions. The equation that estimates the delay (Formula 4), will give the delay in minutes, but as the junction delay in the software was required in seconds, the formula (4) is multiplied by 60 seconds.

$$d(x) = \frac{r^2}{2 \cdot C \cdot \left(1 - \frac{x}{\mu \cdot w}\right)} \quad (4)$$

In which:

μ = the exiting capacity of the lanes (PCE)
r = the red time of the cycle (minutes)
C = cycle time of the traffic lights (minutes)

The second set of equations of this method describes the unsignalized junctions. These equations will take the number of legs into account when calculating the delays from the junction. The set of equations looks like the following (Formula 5 and 6):

$$d(x) = d \cdot Min\left[\left(\frac{x}{c \cdot w}\right), 1\right] \quad (5)$$

Where:

$$d = 0.05 \cdot \left(\frac{n \cdot (k-1)}{2} - p\right) \quad (6)$$

In which:

n = the number of links ending at the junction (-)
k = the number of links starting at the junction (-)
p = the amount of prohibited turns at the junction (-)

The main difference between the method of Aashtiani and Iravani and the method of Vasvári is the approach they take at the creating of equations. Whereas Vasvári is using a single equation, Aashtiani and Iravani use multiple. Both try to reach a model which represents the real-world situation in the best way possible while having a low amount of inputs. The third option that was used was the method, which was already programmed into OmniTRANS, which is described as Junction Modelling (DAT.Mobility, 2016).

Overview of the tests.

For the tests, the capacities of the junctions were set to Dutch standards. For this the capacities of Figure 1 were used. These values were used to calculate the delays using the different methods. First a single junction was tested, to see whether the delays would give a plausible representation of delays that could be the induced due to traffic volumes. For the junction, the different directional loads were added together. This was done to take cross traffic into account, which gave the tests starting values of 10 PCE/h per turn movement. This was increased by 10 PCE/h per turn until the maximum capacity was reached.

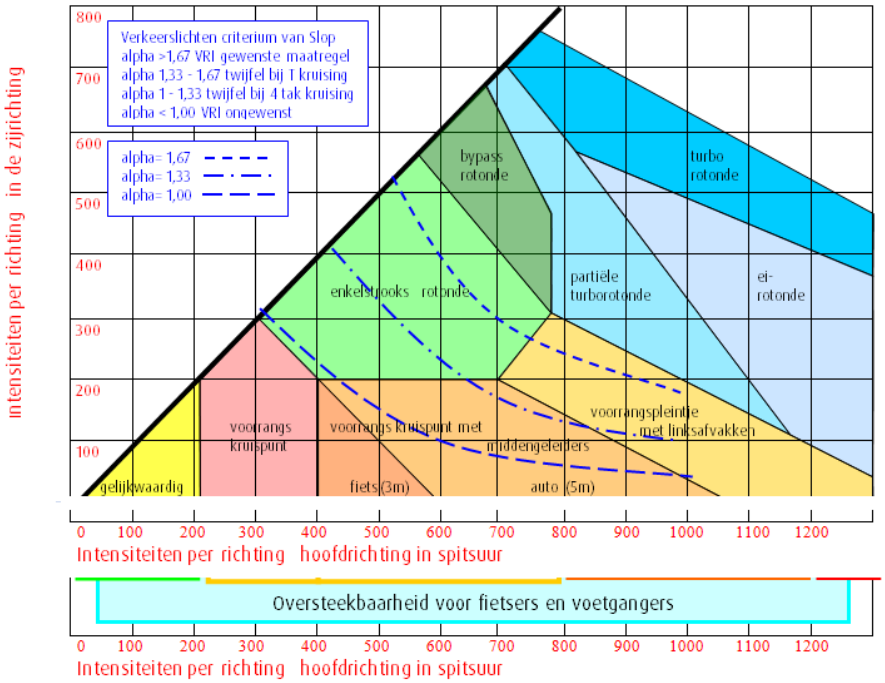

*Figure 1: The capacities per direction of different junction types (*Only available in Dutch) (Goudappel Coffeng, 2019)*

After these single junctions were analysed, the different junction delay functions were modelled in a static traffic model. The model to be used will be of the size of a city, the OmniTRANS tutorial model of Delft to be exact. This model is chosen due to the relatively small size, while having a large variety of junction types and a relatively short calculation time, as the model would have to be calculated with different methods. For the testing of the different methods, a custom job was created. This job can be found in Appendix B.

As the report is looking into some less accurate, but faster junction modelling techniques. This means that some assumptions had to be taken to explain some of the places where the model would become less accurate:

1. The way that traffic behaves is the same in the countries from the functions and The Netherlands. As the literature methods do not model Dutch traffic, but foreign.
2. The maximum capacity of a unsignalized junction is based on Figure 1, as this figure is also used when designing a junction of these loads.

Next to these assumptions, there were also a couple of constraints due to the calculation methods:

1. The capacity of the junction with signals is based on the lanes in the link and not the front lanes. Although this would give a less precise saturation capacity of the front lanes of the junction, these values were less complex to implement to the job.
2. The calculated delay of the junction is written on all the different turns of the junction, while normally the different turns would get different values due to the differences in load and priority settings.

After the methods are implemented and the main bugs are removed from the code, the three different methods will be compared with two different factors: (1) the convergence of the model, to see how many iterations there are needed to come to an equilibrium; and (2) the traffic assignment, which would indicate in what way the traffic would be distributed by the three methods. These two

factors will give an indication if the tested methods of junction delay functions are possible improvements for junction modelling, as is now implemented in OmniTRANS.

Results and discussion

Single junction results

The first results of the different methods were found while modelling a single junction of different types. This was done for four different types of junctions: equal priority, one priority direction, a roundabout and a signalized junction. These results were gathered using OmniTRANS, while using the different methods.

The results will be given in graphs and per different junction. In Appendix C tables can be found where the calculated delays are listed.

Equal priority junction

The first junction to be discussed is the equal priority junction. This junction is described in OmniTRANS and the method of Aashtiani and Iravani (1999). The method of Vasvári does not directly mention the equal junction, but does mention a method that has no signals, so that one is used for this intersection.

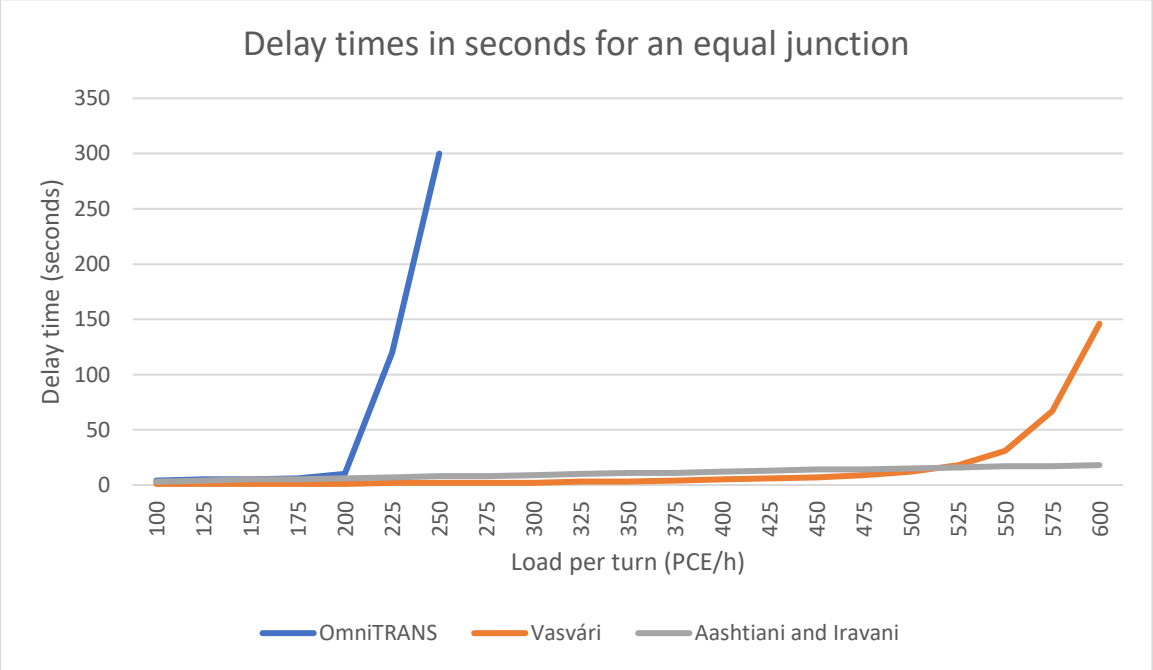

Figure 2: Delay times in seconds for an equal junction

What could be seen in Figure 2 is that the new methods will give a delay at significantly higher loads. This is undesirable, as this would mean that the traffic network could be overloaded with traffic at junctions. For example: if a road has a capacity of 1800 PCE/h and the junction can take 600 PCE/h for each turn, the total capacity of the junction would be not realistic at 7200 PCE/h.

To prevent this, Figure 1 was used for the capacity of the junctions and the loads of all junction arms were added together to get newly calibrated values which would give delays at a lower load on the junction. These values for the equal junction type could be seen in Figure 3:

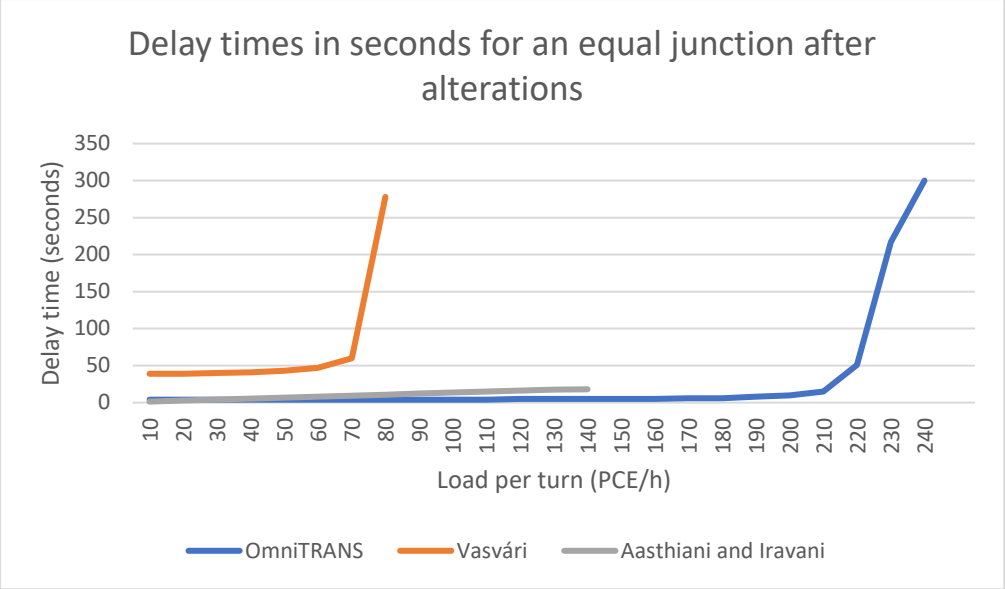

Figure 3: Delay times in seconds for an equal junction after alterations

In this second graph, it could be seen that the new methods will give delays at loads per turn lower than when the OmniTRANS method is used. This is more desirable than in the old situation, as the equal junctions are mostly situated in areas where people live, so people should not take those roads if they do not have to. For the other junctions without traffic signals only the graph after alterations will be given.

Priority junction

For the priority junction, Vasvári and the method in OmniTRANS have specific equations. The method of Aasthiani and Iravani would use the same equation set as for the priority junction, meaning that the new methods have the same values for both the equal junctions and the priority junctions. Which might result in different traffic assignment in a traffic model. The results of the three methods are shown in Figure 4.

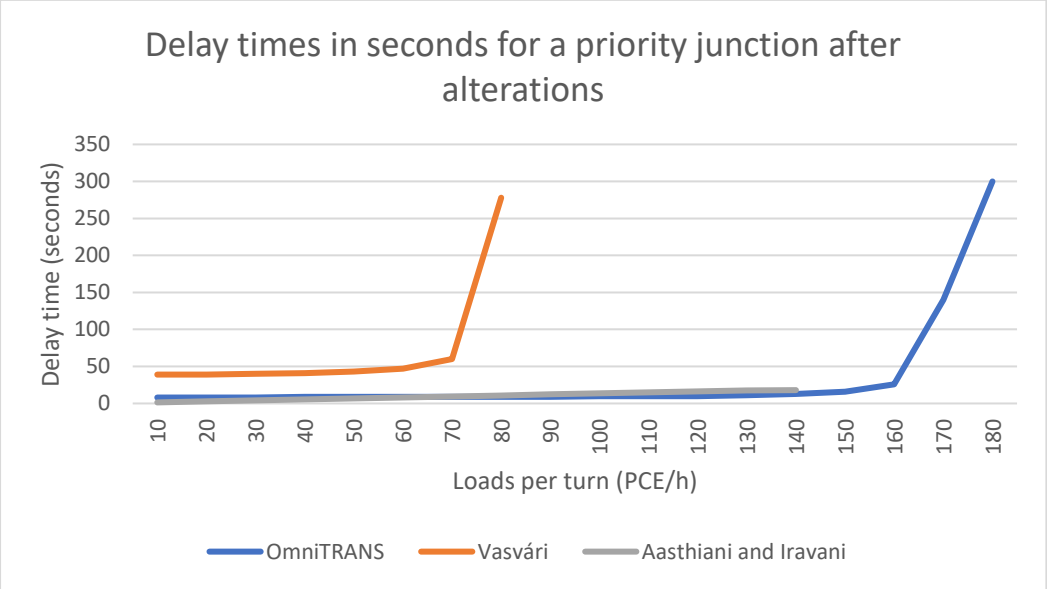

Figure 4: Delay times in seconds for a priority junction after alterations.

As could be seen in Figure 4, the method of Vasvári gives a maximum delay at an earlier moment then the OmniTRANS and the method of Aasthiani and Iravani. This could mean that the final equilibrium could be found faster, as the amount of delay is larger at a lower load. On the contrary, this could also give issues if there is more traffic that must use the junction. The results of this will become clear after testing the whole network.

Signalized junction.

For the signalized junctions the delay, given by the three methods, will be shown in Figure 5:

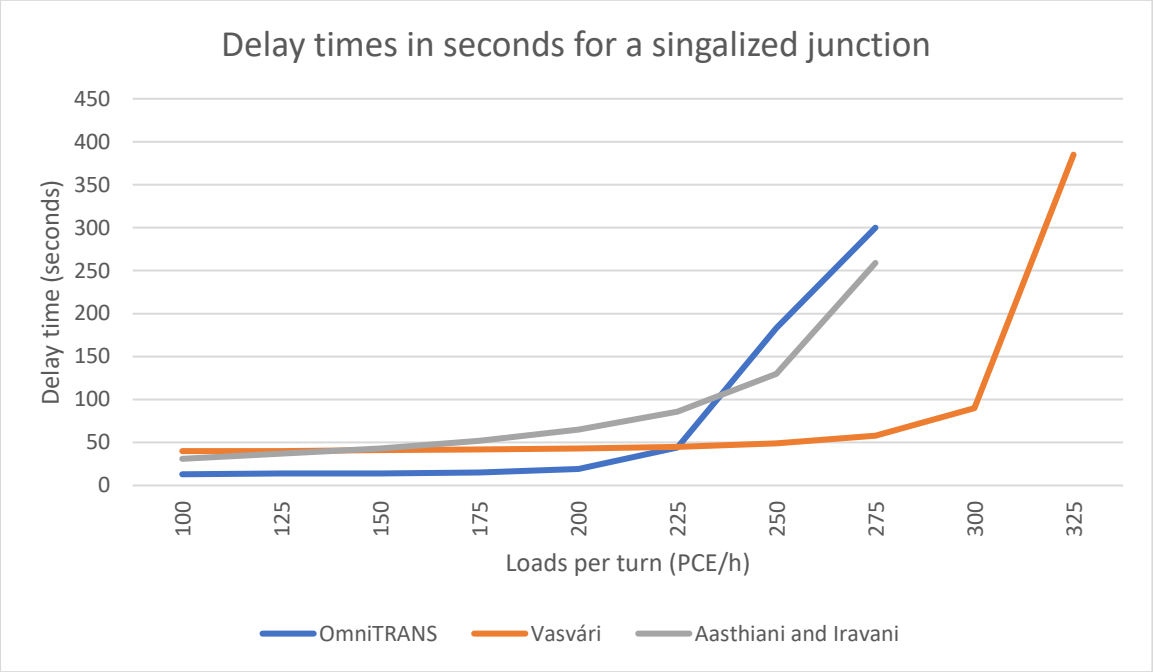

Figure 5: Delay times in seconds for a signalized junction

As could be seen in this figure, all the different methods will give a high delay within a range of 75 PCE/h per turn. This gives an indication that the estimated delay time at a signalized junction would be roughly the same for all the different methods. This makes that there were no alterations done to this type of junction.

Roundabout

The roundabout could not be found in the method of Aasthiani and Iravani. So, the unsignalized junction of this method will be used for roundabouts in the traffic model for this reason. The other two methods will be shown in Figure 6, where their delay times induced by the roundabout are shown.

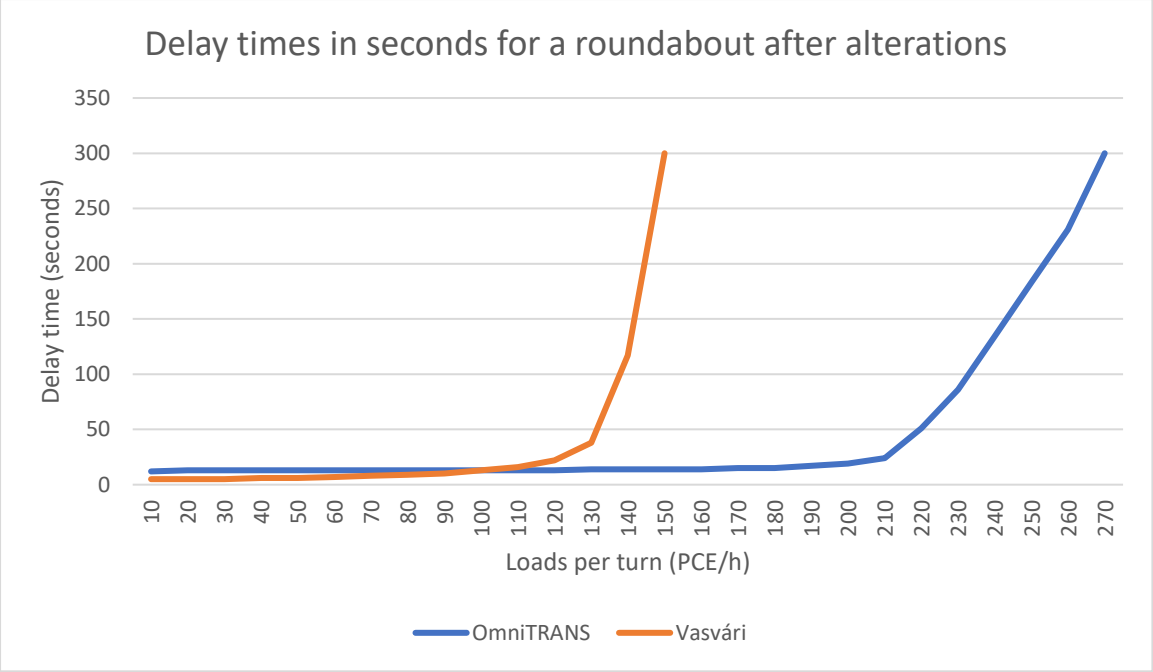

Figure 6: Delay times in seconds for a roundabout after alterations

For this last type of junction, the same observation can be made as before at the unsignalized junctions. The to be tested method gives a large delay at lower traffic loads then the method in OmniTRANS. As this is the case at most of the junctions, the expectation will be that this will not hinder the convergence or the traffic assignment in the traffic model. Therefore, there are no more alterations done to the equations of the different methods.

Network level

On the network level, different types of results can be found. As the network should have a single equilibrium in which all assigned traffic is stable. For this a couple of different indicators could be found. The first being also the lead focus, which is the convergence of the model. The second, which is for now less important, is the assignment of the traffic. The importance is lowered for the traffic assignment, as the methods will not be calibrated for the Dutch traffic system. This means that they will give different assignments compared to the OmniTRANS method, due to the different delay values. Still the traffic assignment can give an indication if the junction types would have the right effects, by looking if the major roads are given a higher load.

Convergence of the network

For the convergence of the network, the two types of junction delay functions were implemented, and a traffic assignment was run for the model of Delft. This is a tutorial network, meaning that the traffic assignment will not represent the real-world situation. However, the convergence should be able to be determined and hopefully would be faster compared to the method of OmniTRANS.

Method of Vasvári

The method of Vasvári was first tested for 20 iterations. After these 20 iterations, the method still had an average difference between iterations of roughly 6 PCE/h per direction of a link. This is quite a lot, but the amount was lowering over the 20 iterations that were done. The calculated average differences in cars per direction will be given in Figure 7, to show progression of the method towards the equilibrium.

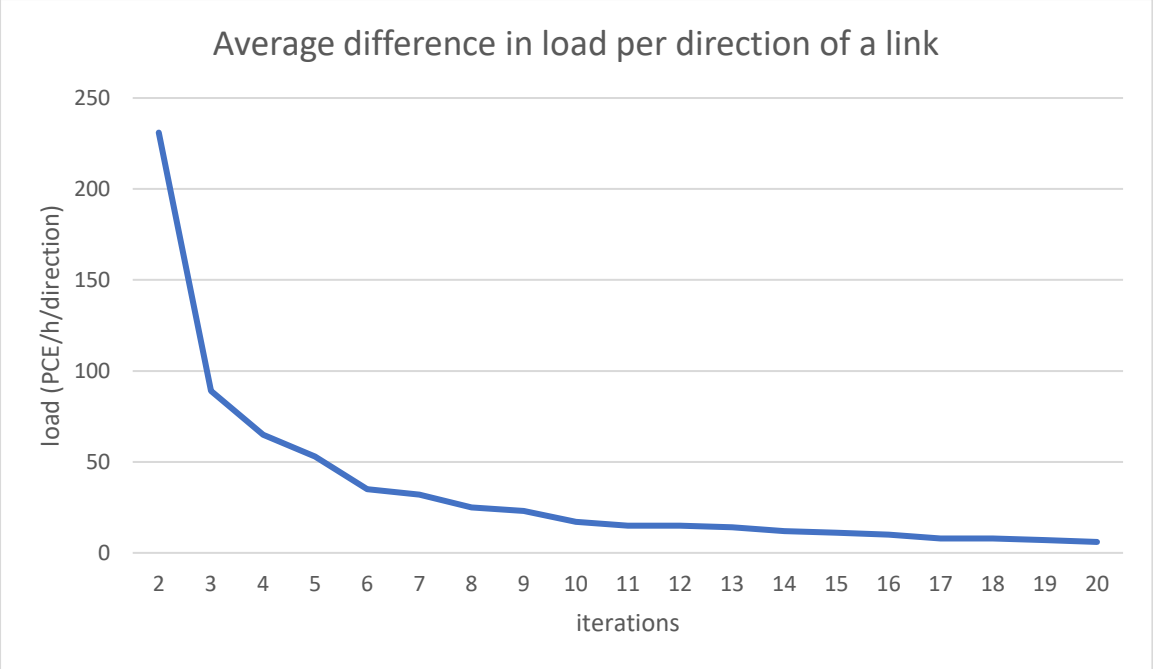

Figure 7: Average difference in load per direction of a link of the method of Vasvári

What could be seen here is that the average differences between directions drops rapidly at the start and starts to be less after reaching 12 alterations. This means that after these 12 iterations, the equilibrium state of the model would become close. Unfortunately, at the 20-iteration mark, the difference is more than 5 PCE/h per direction on a link. Therefore, the equilibrium is not yet reached. As the equilibrium is needed for the people using the traffic models, the model will need more than the 20 iterations to reach the equilibrium. To see the effect on a longer streak of iterations, the comparison with the different methods was used. This would be on 50 iterations, to get a clear view on the behaviour of the convergence and traffic assignment.

Method of Aasthiani and Iravani

The second new method was also tested, this method had some difficulties with the lower and less distinctive delay when the capacity of the junction was reached. This might be somewhat of a difficulty, but also this method shows converging properties. This is given in Figure 8, which shows the difference between iterations getting is getting smaller.

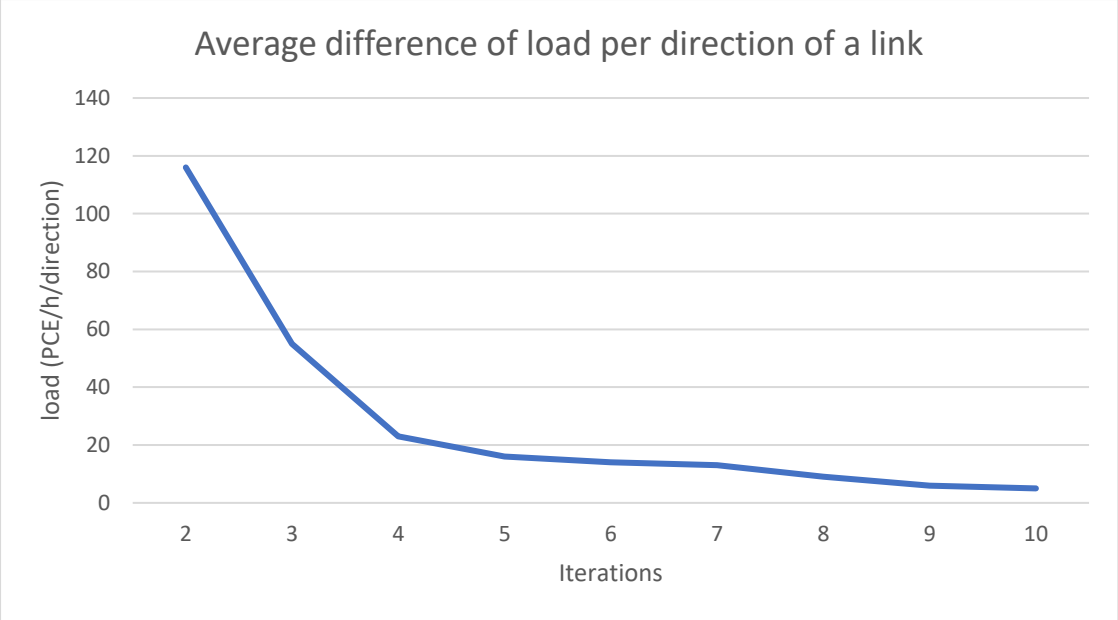

Figure 8: Average difference of load per direction of a link for the method of Aashtiani and Iravani

As could be seen the total amount of PCE/h that changes from links is also in this case dropping, but the starting point was already lower than the first of the methods. This could be because of the lower costs of the unsignalized junctions, giving a lower penalty when there is a lot of traffic. It is also possible to see is that after 10 iterations, the amount of changing PCE/h is already lower than the method of Vasvári after 20. The difference after 10 iterations would be 5 PCE/h, which gives a more promising convergence towards an equilibrium. This value is not yet getting towards zero, so also this method was extended to 50 iterations to get the result.

Overall comparison between methods

The overall comparison of the convergence of each of the methods shows that the current method in OmniTRANS has the highest rate of convergence. In Figure 9 the three methods are compared to one another, which will show the differences in speed at which the convergence takes place. The scale is made logarithmic, as this will give a better indication what happens when the model is reaching the equilibrium state.

This equilibrium is different for each of the methods, but this does not take away from the amount of iterations needed to get to this state. The Figure shows that the current method, implemented in OmniTRANS, has the fastest convergence of the different methods that were tested. This is due to the average difference between the iterations is reducing at the fastest rate, meaning this would be preferable as the acceptable equilibrium state would be available earlier.

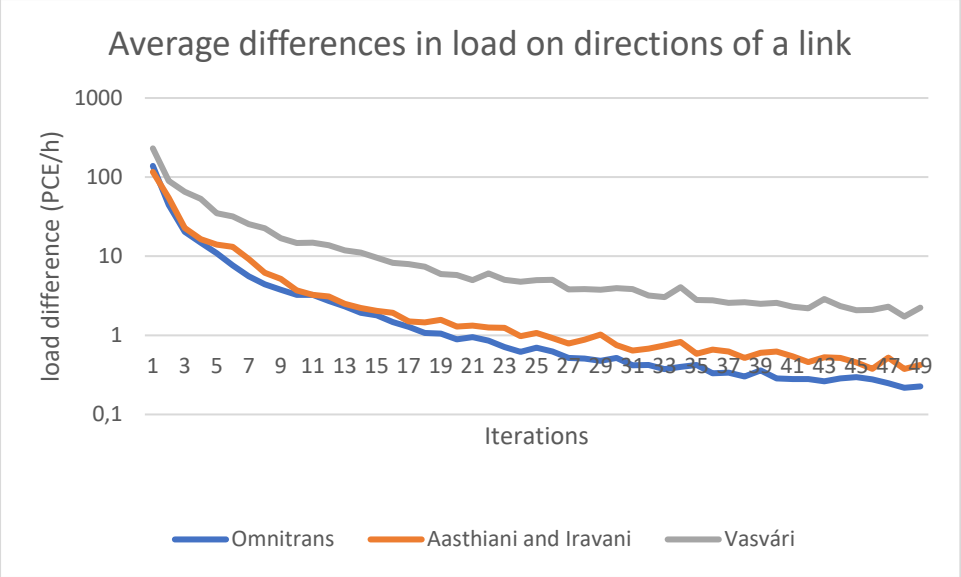

Figure 9: The average differences in load of all methods

So, as both new delay functions are, unfortunately, converging less quickly than the method that is used in OmniTRANS now. The new methods must be dropped, or changes must be made to increase their convergence. For the method of Vasvári, these changes could be made at the parameters that are used in the formulas. So the functions would lean more towards other traffic systems than just the Hungarian one. For the method of Aasthiani and Iravani, the major improvement could most likely be done by implementing an extra factor. This factor would then be used to increase the maximum delay at unsignalized junctions, as this maximum value is now significantly lower than the maximum delays of the other methods and types of junctions.

Traffic assignment

The three different methods have also given different distributions of traffic over the network. This makes that the two different traffic allocations differ from one another. This was to be expected, as the one of the methods is calibrated on the Hungarian traffic (Vasvári, 2015), the second on the Iranian network (Aashtiani & Iravani, 1999) (but also altered to fit the software) and the third is for the Dutch network (DAT.Mobility, 2016).

Although all traffic assignments were different, there is no conclusion when looking at the model of Delft. Because this model does not represent the real-life traffic generation and is not accurate enough for this scale. This is, as said before, due to the nature of being the tutorial network of the software, meaning that this model is purposely not made to represent the actual situation.

Still there is a comparison to be made between the two new methods, which could indicate the trends they would follow. This comparison is made in Figure 10, which shows the difference in the traffic assignment between the method of Vasvári and the method of Aasthiani and Iravani. As could be seen, the grey area is the again the load that is equal over both methods. At the green areas the load of the method of Aasthiani and Iravani is larger. The red areas indicate a large load for the method of Vasvári.

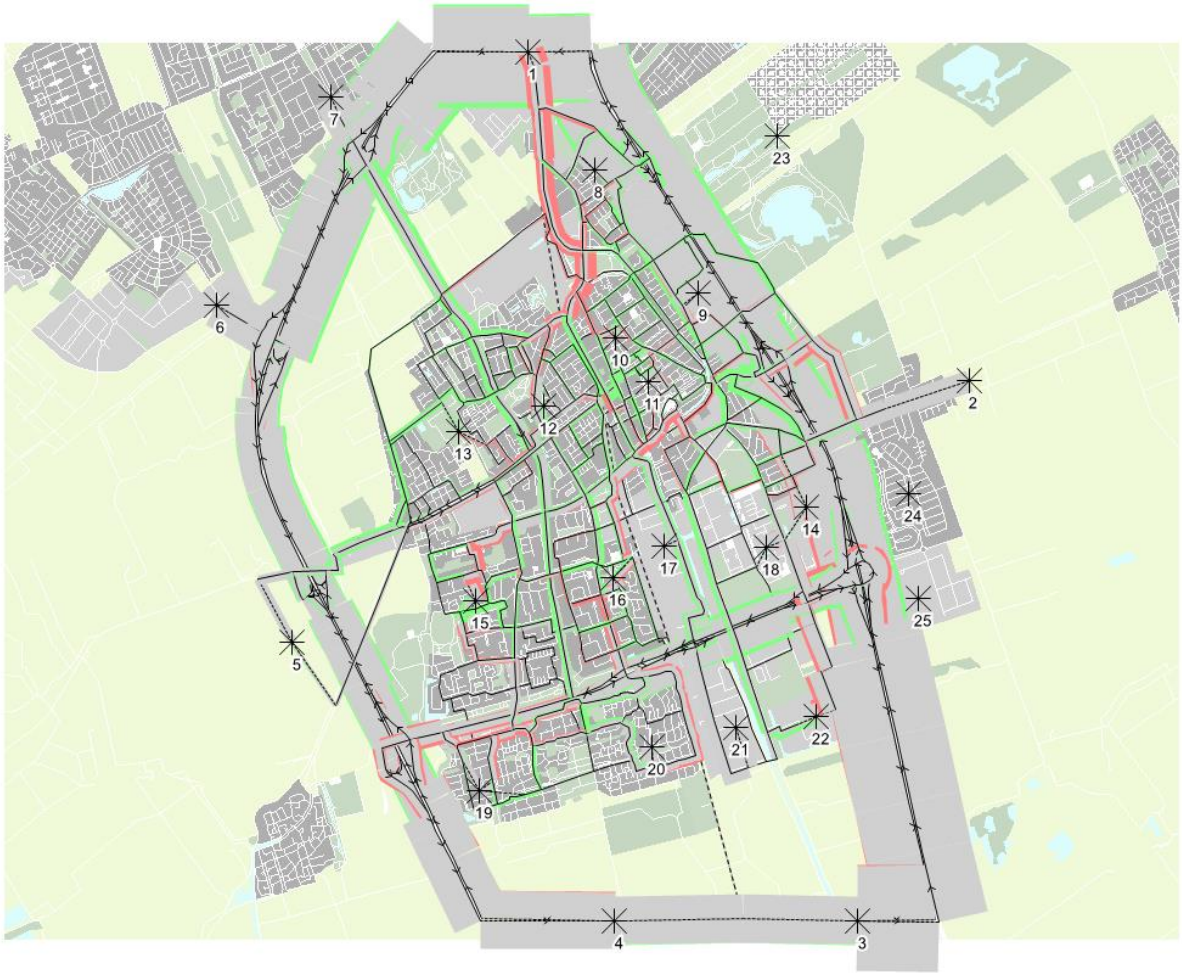

Figure 10: Differences in traffic Allocation of the two literature methods after 50 iterations

This shows that the different methods do in fact assign most of the traffic towards the highways and major roads, which would represent the real-world traffic routing in a city. As research did find out that this is the major trend in urban traffic networks and route choice decisions (Bonsall, 1992).

Conclusion

The new junction delay functions show positive convergence characteristics, which means that they could be implemented to model the junctions in large-scale static traffic models. The rate of this convergence will still differ per method. As the method of Vasvári will have a less stable result after 50 iterations than the method of Aasthiani and Iravani or OmniTRANS has now, this method would not be preferable to use over the other two. This makes that the current versions of these newly tested equations do not reach the rate of convergence and the stability in traffic assignment that the OmniTRANS method has, while the method of Aasthiani and Iravani does not trail far behind. Therefore, the current states of these methods are not able to increase the convergence of a large-scale traffic model.

This means that it must be mentioned that the wanted effect was not reached with this research, as it was hoped that the convergence of the model would increase. While it could be seen in Figure 9 that all the methods do show converging characteristics, they converge at a lower rate, meaning that they are for now no improvements in the large-scale traffic models.

This could be caused by a couple of factors. One of the main would be that they are more sensitive for the total load on a junction. So, if this could be altered to a maximum capacity at 250 PCE per turn movement, this could draw them towards the convergence rate of the current method that is implemented in OmniTRANS.

A second factor that could cause issues is one that could mainly be found in the method of Aasthiani and Iravani and would have to do with the fact that the maximum delay at unsignalized junctions is just 18 seconds. This is a small delay when comparing it to the 300 seconds that are programmed in the software now.

Another factor which could be a cause of the lower convergence, is the exactness of the different methods, although it was tried that during the testing the models had the most comparable quality of inputs. Due to time and skill constraints, the newly tested methods did not have the same quality of modelling compared to the current method of OmniTRANS. An example would be the front lanes that were not considered, it there were more front lanes than lanes in the link, it could influence the total delay of that intersection.

Even though there are no improvements in the convergence yet, the traffic assignment is already showing signs of a realistic load division. This means that the equilibrium could eventually be used to make advices for clients. This could only be feasible if the convergence is also increased by the proposed changes.

This means that the new methods, as used in this report, are not yet capable of increasing the convergence of the model. Nevertheless, this does not mean that they are not at all capable of doing this. As due to the different cost values, the costs of using different roads will also differ. Therefore, further research should be conducted to see if improving could pose for a possible increase of the convergence, while the assignment of traffic should remain at an acceptable level.

Recommendations

During the study some things were found that could use more attention than could be given in the available timeframe. The first of these would be to calibrate the methods that were found in this report to fit the Dutch traffic system, as it is proven that these would converge to a solution. This might thus be a good way to increase the accuracy of these two methods for the Dutch situation and possibly increase the convergence.

The second possibility for future research is to implement the method in a better way, making the inputs even more exact, meaning that the delays would be more precise. Giving the program the option to get better delays from the junctions to use in the routing of the vehicles.

The third would be different, as the next suggestion would be to do research into a possible junction delay function that is designed to work with the Dutch system. This is because there is a more than average amount of slow traffic mixed in the traffic in the Netherlands. For this to be implemented in a proper way, the design should be adapted to fit the situation.

If the method would be finalized and there is certainty that the method could indeed give a faster convergence after some changes to the methods and more testing, the method could be tested in combination with the link volume delay functions. This also effect the costs of the travelled distance.

These suggestions would investigate some of the aspects where the new methods could be improved to get better convergence in the large-scale strategic traffic model.

Bibliography

Literature

Aashtiani, H. Z., & Iravani, H. (1999). *Use of intersection delay functions to improve reliability of traffic assignment model.* Houston, Texas: Wilbur Smith Associates.

Bonsall, P. (1992). The influence of route guidance advic on route choice in urban networks. *Transportation 19*, 1-23. doi:https://doi-org.ezproxy2.utwente.nl/10.1007/BF01130771

DAT.Mobility. (2016). *Junction Modelling in OmniTRANS.* Deventer: DAT.Mobility.

Goudappel Coffeng. (2019). *Intensiteiten per rijrichting van een viertaks kruising.* Deventer.

Prasker, J. N., & Bekhor, S. (2004). Route Choice Models Used in the Stochastic User Equilibrium Problem: A Review. *Transport Reviews*, 437-463. doi:https://doi.org/10.1080/0144164042000181707

Vasvári, G. (2015). *Identification of junction specific volume-delay functions by methods of traffic simulation.* Budapest: Budapest University of Technology and Economics.

Webster, F. (1958). *Traffic Signal Setting.* Berkshire: Road Research Labatory Crowthorne.

Formulas

Tables

Figures

Figure 6 Own

Figure 7 Own

Figure 8 Own

Figure 9 Own

Figure 10 Own

Appendix A: The used values of the different parameters for the method of Vasvári

Junction type	α	β	φ_1	φ_2	φ_3	n	c
3L 2x2+2x1 sig	119.0	1.0043	45.5	1.920	1.000	2	255.0
3L 2x2+2x1 stop	60.0	1.0085	70.0	1.040	0.91	2	660.0
3L 2x2+2x2 sig	200.0	1.0026	100.0	1.500	1.000	2	205.0
3L 2x2+2x2 stop	70.0	1.0073	70.0	1.000	0.910	2	750.0
3L 2x1+2x1 sig & 4L 2x2+2x1 sig	41.0	1.0125	129.0	1.280	1.000	1	395.0
3L 2x1+2x1 stop & 4L 2x2+2x2 stop	44.0	1.0117	79.0	1.010	1.000	1	725.0
4L 2x1+2x1 sig	26.5	1.0197	156.5	1.245	1.050	1	212.5
4L 2x1+2x1 stop	20.5	1.0257	69.0	1.005	0.960	1	550.0
4L 2x2+2x1 stop	66.0	1.0077	93.5	1.005	0.985	2	312.5
4L 2x2+2x2 sig	60.5	1.0085	194.5	1.190	1.010	2	300.0
RA 1 11m	43.0	1.0120	304.5	1.015	1.015	1	387.5

Table A.1: The calculated parameters for the junctions (Vasvári, 2015)

Appendix B: Job used in the calculation of junction delays

```
# type of junction modelling
modelling = 1

# selecting where to write the results
if modelling ==1
  p,m,t,u,r,i = 1,11,10,1,3,1
elsif modelling ==2
  p,m,t,u,r,i = 1,11,10,1,2,1
end
  # description of the iterations and maximum iterations
maxIterations = 20
iteration = 1

# defining methods for repeated calculations
def variables(type,legs,lanes_main,lanes_secondary)  # the code which selects the different variables for the junction types for the method of Vasvári
   if type == 1 && legs == 3 && lanes_main == 1
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 129.0,1.0,41.0,1.0125,1.0,900.0
   elsif type == 1 && legs == 3 && lanes_main == 2 && lanes_secondary == 1 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 45.5,1.92,1.0,119.0,1.0043,2.0,900.0
   elsif type == 1 && legs == 3 && lanes_main == 2 && lanes_secondary == 2 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 100.0,1.5,1.0,200.0,1.0026,2.0,900.0
   elsif type == 2 && legs == 3 && lanes_main == 1 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 79.0,1.01,1.0,44.0,1.0117,1.0,800.0
   elsif type == 2 && legs == 3 && lanes_main == 2 && lanes_secondary == 1 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 70.0,1.04,0.91,60.0,1.0085,2.0,800.0
   elsif type == 2 && legs == 3 && lanes_main == 2 && lanes_secondary == 2 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 70.0,1.0,0.91,70.0,1.0073,2.0,800.0
   elsif type == 1 && legs == 4 && lanes_main == 1 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 156.5,1.245,1.05,26.5,1.0197,1.0,900.0
   elsif type == 1 && legs == 4 && lanes_main == 2 && lanes_secondary == 1 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 129.0,1.28,1.0,41.0,1.0125,2.0,900.0
   elsif type == 1 && legs == 4 && lanes_main == 2 && lanes_secondary == 2 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 194.5,1.19,1.01,60.5,1.0085,2.0,900.0
   elsif type == 2 && legs == 4 && lanes_main == 1 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 69.0,1.005,0.96,20.5,1.0257,1.0,800.0
   elsif type == 2 && legs == 4 && lanes_main == 2 && lanes_secondary == 1 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 93.50,1.005,0.985,66.0,1.0077,2.0,800.0
   elsif type == 2 && legs == 4 && lanes_main == 2 && lanes_secondary == 2 then
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 79.0,1.01,1.0,44.0,1.0117,1.0,800.0
   else
      phi1,phi2,phi3,alpha,beta,lanes,capacity = 304.5,1.015,1.015,43.0,1.0120,1.0,1700.0
   end
  return phi1,phi2,phi3,alpha,beta,lanes,capacity
end
 # the code which will give the loads on the different links

def loadcalculation(nodes,iteration,r)
   a,b,loadPer_link2,loads_each,loads= 0,0,[],[],[]
   while a < nodes.length
      junctionNode = nodes[a][0]
      loadPer_link = OtQuery.execute_to_a("SELECT l52.load
                  FROM link5_2data1 AS l52,
                      line AS l
                  WHERE l52.purpose = 1 AND l52.mode = 11 AND l52.time = 10 AND
                      l52.user = 1 AND l52.result = #{r} AND l52.iteration = #{iteration} AND
                      l52.linknr = l.linenr AND (l.pointnra = #{junctionNode} OR l.pointnrb = #{junctionNode})")

      if loadPer_link == []
         loadPer_link,loads_each[a] =0,0
      else
```

```
        while b < loadPer_link.length
          loadPer_link2[b] = loadPer_link[b][0]
          b+=1
        end
        loads_each[a] = loadPer_link2.sum
        loads[a] = loads_each[a]
      end
      b = 0
      a +=1
    end
  return loads
end

  # the code which will give the lanes, amount of legs and type of road
def lanecalculation(nodes,iteration,r)
    # Starts with defining several parameters which are used in the calculation (only for requirements from the code

legs,number,alternative,alt,alt2,alt3,calculation_lanesx,calculation_lanesxx,lanesPer_link,weightArray,roadtypesz,weight_o
utput= [],[],[],[],[],[],[],[],[],[],[],[]
  a=0
  while a< nodes.length
    junctionNode = nodes[a][0]
    linknr_link = OtQuery.execute_to_a("SELECT l52.linknr
          FROM link5_2data1 AS l52,
              line AS l
          WHERE l52.purpose = 1 AND l52.mode = 11 AND l52.time = 10 AND
              l52.user = 1 AND l52.result = #{r} AND l52.iteration =#{iteration} AND
              l52.linknr = l.linenr AND (l.pointnra = #{junctionNode} OR l.pointnrb = #{junctionNode})")
    linknr_link = linknr_link.uniq
    b= 0
    legs[a] = linknr_link.length
    while b < linknr_link.length
      number = linknr_link[b][0]
      lanesPer_link[b] = OtQuery.execute_to_a("SELECT lanes
                FROM link1_1data2
                WHERE linknr = #{number}")
      roadtypesz[b] = OtQuery.execute_to_a("SELECT typenr
                FROM link2_1data1
                WHERE serienr = 3 AND linknr = #{number}")
      f = 0
      while f < roadtypesz[b].length
        alternative = roadtypesz[b][0][0]
        f += 1
      end
      weightArray[b] = alternative

      c = 0
      weight = []
      while c < weightArray.length
        if weightArray[c] == 5 || weightArray[c] == 6 || weightArray[c] == 7
            weight[c] = 5.0
        elsif weightArray[c] == 2 || weightArray[c] == 3 || weightArray[c] == 4
            weight[c] = 4.0
        elsif weightArray[c] == 8 || weightArray[c] == 9 || weightArray[c] == 10
            weight[c] = 3.0
        else
            weight[c] = 2.0
        end
      c+= 1
      end
    b+=1
    end
    f,g = 0,0
```

```
      while g<lanesPer_link.length
        while f< lanesPer_link[g].length
          alternative = lanesPer_link[g][f][0]
          alt[f] = alternative
          f += 1
        end
        alt2[g] = alt
        g+=1
      end
      e=0
      alt2.each{|element|
      calculation_lanesx[e] = element.compact.uniq
      e+=1
      }
      calculation_lanesxx[a]=calculation_lanesx.flatten
      calculation_lanesxx[a]=calculation_lanesxx[a]-[0]
      if calculation_lanesxx[a].length >=5
        calculation_lanesxx[a]=calculation_lanesxx[a]-[calculation_lanesxx[a].min]
      end
      weight_output[a] = weight
      a+=1

    end
  return calculation_lanesxx,weight_output
end

###################################
#calculation of the junction delays
###################################

#start of the traffic assignment with junction costs.
while iteration <= maxIterations
 $Ot.indentStart "ITERATION #{iteration}"
 assign = OtTraffic.new
 assign.load = [p,m,t,u,r,iteration]
 assign.initialCost = [p,m,t,u,r,iteration-1] if iteration > 1
 assign.execute
 phi=1/iteration.to_f
 if(iteration>1)
  network = OtNetwork.new
  network.updateResults([p,m,t,u,r,(iteration-1)],[p,m,t,u,r,iteration],[p,m,t,u,r,iteration],1-phi,phi)
 end


 lanes_output=[]
  # Indexing parameters
 a,b,c,d,e = 0,0,0,0,0

 # parameters for determining loads and junction types
 loads_equal,loads_priority,loads_signals,loads_each,loadsPer_link2 = [],[],[],[],[]

 loadArray = OtQuery.execute_to_a("SELECT nodenr
             FROM node")
 equalJunctions = OtQuery.execute_to_a("SELECT nodenr
             FROM node1data1
             WHERE junctiontype = 1")
 priorityJunctions = OtQuery.execute_to_a("SELECT nodenr
             FROM node1data1
             WHERE junctiontype = 2")
 signals = OtQuery.execute_to_a("SELECT nodenr
             FROM node1data1
             WHERE junctiontype = 3")
 roundabout = OtQuery.execute_to_a("SELECT nodenr
```

```
            FROM node1data1
            WHERE junctiontype = 4")

 # calculation of the loads for each of the junction options
 loads_equal = loadcalculation(equalJunctions,iteration,r)
 loads_priority = loadcalculation(priorityJunctions,iteration,r)
 loads_signals = loadcalculation(signals,iteration,r)
 loads_RA = loadcalculation(roundabout,iteration,r)


##############################################################################################
# for equal and priority junctions
##############################################################################################
 #parameters used in calculation
 a,b,calculation_lanes,calculation_load,legs,junctiondelay,delay_output_equal,prohib,lanes_main,lanes_secondary =
0,0,[],0,[],[],[],0,[],[]

 #calculation of the number of lanes and type of roads for equal junctions
 calculation_lanes,weight_output = lanecalculation(equalJunctions,iteration,r)

 if modelling == 1
    while a < loads_equal.length
       legs = calculation_lanes[a].length
       if calculation_lanes[a].max >= 2
          lanes_main[a],lanes_secondary[a] = 2,2
       else
          lanes_main[a],lanes_secondary[a] = 1,1
       end
       type = 2 # no signals
          # loading of different parameters for the equation
       phi1,phi2,phi3,alpha,beta,lanes,capacity = variables(type,legs,lanes_main[a],lanes_secondary[a])
       calculation_load = loads_equal[a]/2
       distance = 1
       junctiondelay[a] = phi1 * distance * (phi2 + ((alpha**(2.0))*((phi3-(calculation_load/(lanes*capacity)))**(2.0)) +
(beta**2.0))**(1.0/2.0) - alpha * (phi3-(calculation_load/(lanes* capacity))) - beta)
          # in the software there is a value of 300 for a maximum delay, this is also done here
       if junctiondelay[a] >= 300
          junctiondelay[a] = 300
       end
       delay_output_equal[a] = [equalJunctions[a][0],junctiondelay[a]]   #output Array: [nodenr,delay (sec)]
       a += 1
    end
 elsif modelling == 2    # option for the second method
    while a < loads_equal.length
       legs = calculation_lanes[a].length
       capacity = 900
       calculation_load = loads_equal[a]/2
       d2 =  0.05 * ((legs*(legs-1.0)/2.0)-prohib)
       #delay calculation
       junctiondelay[a] =(d2 *[(calculation_load/(capacity*calculation_lanes[a].max)), 1.0].min)*60
       if junctiondelay[a] >= 300 #maximum delay
          junctiondelay[a] = 300
       end
       delay_output_equal[a] = [equalJunctions[a][0],junctiondelay[a]] #output Array: [nodenr,delay (sec)]
       a += 1
    end
 else
    writeln "No Junction modelling/opportunity for a new method."
 end

 #parameters for the priority junctions
 a, b,calculation_lanes,calculation_load,junctiondelay,delay_output_prio,prohib,lanes_main,lanes_secondary =
0,0,[],0,[],[],0,[],[]
```

```
 #calculation of the number of lanes and type of roads for priority junctions
 calculation_lanes,weight_output = lanecalculation(priorityJunctions,iteration,r)

 if modelling == 1
  while a < loads_priority.length
   legs = calculation_lanes[a].length
   if calculation_lanes[a].max >= 2
    lanes_main[a],lanes_secondary[a] = 2,2
   else
    lanes_main[a],lanes_secondary[a] = 1,1
   end
   type = 2 #no signals
    #parameters calculation
   phi1,phi2,phi3,alpha,beta,lanes,capacity = variables(type,legs,lanes_main[a],lanes_secondary[a])
   calculation_load = loads_priority[a]/2
   distance = 1
    #delay calculation
   junctiondelay[a] = phi1 * distance * (phi2 + ((alpha**(2.0))*((phi3-(calculation_load/(lanes*capacity)))**(2.0)) +
(beta**2.0))**(1.0/2.0) - alpha * (phi3-(calculation_load/(lanes * capacity))) - beta)
   if junctiondelay[a] >=300 #max delay
    junctiondelay[a] = 300
   end
   delay_output_prio[a] = [priorityJunctions[a][0],junctiondelay[a]] #output Array: [nodenr,delay (sec)]
   a += 1
  end
 elsif modelling == 2
  while a < loads_priority.length
   legs = calculation_lanes[a].length
   capacity = 900
   calculation_load = loads_priority[a]/2
   d2 =  0.05 * ((legs*(legs-1.0)/2.0)-prohib)
    # delay calculation
   junctiondelay[a] =(d2 *[(calculation_load/(capacity*calculation_lanes[a].max)), 1.0].min)*60
   if junctiondelay[a] >= 300 #max delay
    junctiondelay[a] = 300
   end
   delay_output_prio[a] = [priorityJunctions[a][0],junctiondelay[a]]   #output Array: [nodenr,delay (sec)]
   a += 1
  end
 else
  writeln "No Junction modelling/opportunity for a new method."
 end

##########################################################################################
# for signalized Junctions
##########################################################################################
 #parameters in calculation
 a,b,calculation_lanes,legs,junctiondelay,delay_output_sig,lanes_main,lanes_secondary = 0,0,[],[],[],[],[],[]
 #calculation of the number of lanes and types of roads for signalized junctions
 calculation_lanes,weight_output = lanecalculation(signals,iteration,r)

 if modelling == 1
  while a < loads_signals.length
   legs = calculation_lanes.length
   calculation_load = 0
   if calculation_lanes[a].max >= 2
    lanes_main[a],lanes_secondary[a] = 2,2
   else
    lanes_main[a],lanes_secondary[a] = 1,1
   end
   type = 1    #signalized
    #loading of parameters
   phi1,phi2,phi3,alpha,beta,lanes,capacity = variables(type,legs,lanes_main[a],lanes_secondary[a])
```

```
    calculation_load = loads_signals[a]/2
    distance = 1
     #delay calculation
    junctiondelay[a] = phi1 * distance * (phi2 + ((alpha**(2.0))*((phi3-(calculation_load/(lanes*capacity)))**(2.0)) +
(beta**2.0))**(1.0/2.0) - alpha * (phi3-(calculation_load/(lanes * capacity))) - beta)
    if junctiondelay[a] >=300 #maximum delay
     junctiondelay[a] = 300
    end
    delay_output_sig[a] = [signals[a][0],junctiondelay[a]]  #output Array: [nodenr,delay(sec)]
    a += 1
   end
  elsif modelling == 2
   exit_cap,f,calculation_load,cycle_time,red_time,junctiondelay1 = 900.0,0,[],[],[],[]
   loads_signals.each{ |index|
                calculation_load[f] = index/8.0
                f+=1
               }
   lanesPer_link_calc = calculation_lanes.flatten
   y,z= 0,0     #next loop was built in a different job, so the indexing parameters start at the end of the alphabet
   while z < signals.length
    y,junctiondelay1 = 0,[]
    while y < weight_output[z].length
     cycle_time[z] = 1.0+ ((weight_output[z].sum / 8.0)*(weight_output[z].length/4.0))
     red_time[z] = 1.2* cycle_time[z]* (1-((weight_output[z].length* weight_output[z][y])/(2*weight_output[z].sum)))
      #delay calculation
     junctiondelay1[y] =(((red_time[z]*red_time[z])/(2.0*cycle_time[z]*(1-
(calculation_load[z])/(exit_cap*lanesPer_link_calc[z])))))*60
     y+=1
    end
     #rewriting of the output to be used easier
    junctiondelay[z] = junctiondelay1
    junctiondelay[z] =junctiondelay[z].sum/junctiondelay[z].length
    if junctiondelay[z].abs >=300 # maximum delay
     junctiondelay[z]=300
    end
    delay_output_sig[z] = [signals[z][0],junctiondelay[z].abs]  #output Array : [nodenr,delay(sec)]
    z += 1
   end
  else
   writeln "No Junction modelling/opportunity for a new method."
  end

  ##################################################################################################
  # calculation of delay for roundabouts
  ##################################################################################################

  calculation_lanes,legs,junctiondelay,delay_output_RA,lanes_main,lanes_secondary,a = [],[],[],[],[],[],0
  calculation_lanes,weight_output = lanecalculation(roundabout,iteration,r)

  if modelling == 1
   while a < loads_RA.length
    legs = calculation_lanes[a].length
    if calculation_lanes[a].max >= 2
       lanes_main[a],lanes_secondary[a] = 2,2
    else
       lanes_main[a],lanes_secondary[a] = 1,1
    end
    type = 3
    phi1,phi2,phi3,alpha,beta,lanes,capacity = variables(type,legs,lanes_main,lanes_secondary)
    calculation_load = loads_RA[a]/2
    distance = 1
     #delay calculation
```

```
    junctiondelay[a] = phi1 * distance * (phi2 + ((alpha**(2.0))*((phi3-(calculation_load/(lanes*capacity)))**(2.0)) +
(beta**2.0))**(1.0/2.0) - alpha * (phi3-(calculation_load/(lanes * capacity))) - beta)
    if junctiondelay[a] >=300 #maximum delay
     junctiondelay[a] = 300
    end
    delay_output_RA[a] = [roundabout[a][0],junctiondelay[a]]          #output Array: [nodenr,delay (sec)]
    a += 1
  end
 elsif modelling == 2    # option for the second method
  while a < loads_RA.length
   legs = calculation_lanes[a].length
   capacity = 900
   calculation_load = loads_RA[a]/2
   d2 =  0.05 * ((legs*(legs-1.0)/2.0)-prohib)
    #delay calculation
   junctiondelay[a] =(d2 *[(calculation_load/(capacity*calculation_lanes[a].max)), 1.0].min)*60
   if junctiondelay[a] >= 300 #maximum delay
    junctiondelay[a] = 300
   end
   delay_output_RA[a] = [roundabout[a][0],junctiondelay[a]] #output Array: [nodenr,delay (sec)]
   a += 1
  end
 else
  writeln " No Junction modelling/opportunity for a new method."
 end

 delay_total = delay_output_RA + delay_output_sig + delay_output_equal + delay_output_prio
 p delay_total

 a=0
 delay_total.each{|nodedata|
          node = nodedata[0]
          delay = nodedata[1]
          turns = OtQuery.execute_to_a("SELECT tri.trilinenr
                    FROM node n, turn t, triline tri
                    WHERE n.nodenr = #{node} AND n.nodenr = tri.pointnrb AND tri.pointtypeb = 2 AND tri.trilinenr =
t.turnnr")
          turns=turns.flatten
          turns.each{ |turnnr|
            OtQuery.execute("INSERT INTO turn5data1 (turnnr,purpose,mode,time,\"user\",result,iteration,cost,turndelay)
                  VALUES (#{turnnr},#{p},#{m},#{t},#{u},#{r},#{iteration},#{delay/60.0},#{delay})
                  ON CONFLICT (turnnr,purpose,mode,time,\"user\",result,iteration) DO UPDATE
                  SET (cost,turndelay) = (#{delay/60.0},#{delay})")
                }
           }
 # collect and remove impossible turn delays
 turns5 = OtQuery.execute_to_a("SELECT turnnr,load,cost,turndelay FROM turn5data1 WHERE purpose = #{p} AND mode
= #{m} AND time = #{t} AND \"user\" = #{u} AND result = #{r} AND iteration = #{iteration}")
 turns5hash = {}
 turns5.each{ |turnnr,load,cost,turndelay|
  OtQuery.execute("UPDATE turn5data1 SET load = 0 WHERE turnnr = #{turnnr} AND purpose = #{p} AND mode = #{m}
AND time = #{t} AND \"user\" = #{u} AND result = #{r} AND iteration = #{iteration}") if load.nil?
  OtQuery.execute("UPDATE turn5data1 SET cost = 0 WHERE turnnr = #{turnnr} AND purpose = #{p} AND mode = #{m}
AND time = #{t} AND \"user\" = #{u} AND result = #{r} AND iteration = #{iteration}") if cost.nil?
  OtQuery.execute("UPDATE turn5data1 SET turndelay = 0 WHERE turnnr = #{turnnr} AND purpose = #{p} AND mode =
#{m} AND time = #{t} AND \"user\" = #{u} AND result = #{r} AND iteration = #{iteration}") if turndelay.nil?
  turns5hash[turnnr] = [(load.nil? ? 0 : load),(cost.nil? ? 0 : cost),(turndelay.nil? ? 0 : turndelay)]
        }
 turns = OtQuery.execute_to_a("SELECT turnnr FROM turn")
 turns.flatten!
 turns.each{ |turnnr|
  if turns5hash[turnnr].nil?
   OtQuery.execute("INSERT INTO turn5data1 (turnnr,purpose,mode,time,\"user\",result,iteration,load,cost,turndelay)
```

```
                VALUES (#{turnnr},#{p},#{m},#{t},#{u},#{r},#{iteration},0,0,0)
                ON CONFLICT (turnnr,purpose,mode,time,\"user\",result,iteration) DO NOTHING")
  end
      }


 iteration += 1
 $Ot.indentEnd
end
```

Appendix C: Calculated values for each junction type

Before alterations

Load per turn movement	Equal priority (Sec.)	Priority signs (Sec.) (main road, crossing road, side roads)	Traffic signals (Sec.)	Roundabout (Sec.)
100 PCE	4	1, 9, 10	22	13
125 PCE	5	1, 9, 11	24	14
150 PCE	5	1, 10, 16	25	14
175 PCE	6	1, 11, 279	31	15
200 PCE	10	1, 16, 300	47	19
225 PCE	120	1, 132, 300	113	44
250 PCE	300	1, 300, 300	300	183
275 PCE	-	-	-	300

Table D.1: calculated delays for the junction modelling module of OmniTRANS

Load per turn movement	Equal priority and priority junction	Signalized junction	Roundabout
100 PCE	1	40	5
125 PCE	1	40	5
150 PCE	1	41	6
175 PCE	1	42	6
200 PCE	1	43	6
225 PCE	2	45	7
250 PCE	2	49	7
275 PCE	2	58	8
300 PCE	2	90	8
325 PCE	3	385	9
350 PCE	3		10
375 PCE	4		11
400 PCE	5		13
425 PCE	6		15
450 PCE	7		17
475 PCE	9		21
500 PCE	12		28
525 PCE	18		41
550 PCE	31		78
575 PCE	67		305
600 PCE	146		

Table D.2: Calculated delays for the junction delay function of Vasvári

Load per turn movement	Equal priority and priority junction	Signalized junction
100 PCE	3	31
125 PCE	4	37
150 PCE	5	43
175 PCE	5	52
200 PCE	6	65
225 PCE	7	86
250 PCE	8	130
275 PCE	8	259
300 PCE	9	
325 PCE	10	
350 PCE	11	

375 PCE	11	
400 PCE	12	
425 PCE	13	
450 PCE	14	
475 PCE	14	
500 PCE	15	
525 PCE	16	
550 PCE	17	
575 PCE	17	
600 PCE	18	

TableD.3: Calculated delays for the junction delay function of Aashtiani and Iravani

After alterations

Load per turn movement	Equal priority and priority junction	Roundabout
10 PCE	39	5
20 PCE	39	5
30 PCE	40	5
40 PCE	41	6
50 PCE	43	6
60 PCE	47	7
70 PCE	60	8
80 PCE	278	9
90 PCE		10
100 PCE		13
110 PCE		16
120 PCE		22
130 PCE		38
140 PCE		117

Table D.4: Calculated delays for the junction delay function of Vasvári after alterations

Load per turn movement	Equal priority and priority junction
10 PCE	1,4
20 PCE	2,7
30 PCE	4,1
40 PCE	5,4
50 PCE	6,8
60 PCE	8,1
70 PCE	9,5
80 PCE	10,8
90 PCE	12,2
100 PCE	13,5
110 PCE	14,9
120 PCE	16,2
130 PCE	17,6
140 PCE	18

Table D.5: Calculated delays for the junction delay function of Aashtiani and Iravani