Variable Speed Limit Simulation in SUMO

Till Wenke[a], Daria Eckert[a], Jure Vito Srovin[a], and Robin Van Heirseele[a]

[a]ERASMUS+ at University of Tartu, Estonia

The project repository and implementation can be found on GitHub

Simulating traffic with a goal of designing an effective and safe transportation system can reduce costs of testing different approaches. One of the things you can do with it, is improving traffic in specific scenarios and conditions. We tried different algorithms for variable speed limits on a highway lane merge in order to reduce the congestion caused by the bottleneck and improve safety of passengers.

Introduction

Lane merges are a common occurrence in our current automotive transportation system. These merges exist everywhere: from roads connecting small villages, to inner city roads and also on motorways often in combination with temporary building sites. Motorways are known for their ability to handle huge volumes of traffic and enabling the cars to travel at a high average speeds. Due to these characteristics, the way lane merges are implemented is quite important for the flow of traffic on motorways. A poorly executed merge could result in a massive capacity drop and generate bottlenecks. In order to enable a smoother flow through these merges, we want to investigate variable speed limits (VSL) for the cars as they approach the point at which lanes are merged. In this project we focused on 3-to-2 lane reduction merges. The main argument for this choice was that this type of merge is one of the most realistic scenarios and consequently has the most applications in practice. Other scenarios such as 3-to-1 lane merges are less common as they usually result from accidents and reduce the traffic flow dramatically. In order to work with these merges, we used Python and SUMO - an open-source microscopic traffic simulation tool - as our programming and simulation environment with the TraCI library providing the means for interaction between the two. SUMO enabled us to first set up a road and simulate a realistic congestion. Afterwards a more realistic driving behaviour was implemented.This SUMO visualisation allowed us to verify the congestion model. Different VSL algorithms were used in order to implement the varying speed limits. With the help of SUMO, these different algorithms could be visualised, helping us to better understand the reasoning and performance of these algorithms. In order to evaluate how the algorithms performed, several metrics were used. These metrics had the goal of evaluating the efficiency, safety and environmental impact of the different algorithms. In order to see the potential improvement, we always compared the resulting metrics for the algorithms with our baseline model.

Fig. 1. Highway merge simulation in SUMO during a congestion.

Creating the SUMO simulation

1. Setting up the road

In the simulation we set up a 3-to-2 lane reduction merge in SUMO. The road can intuitively be split up into two parts: before and after the merge. The part before the merge consists of nine connected 500 m road segments while the part after the merge only contains one individual 500 m road segment. With 300 m (segment 10) and 275 m (segment 10) the two segments right before the merge area are a little shorter. With the 11th segment being the area where congestions are detected.

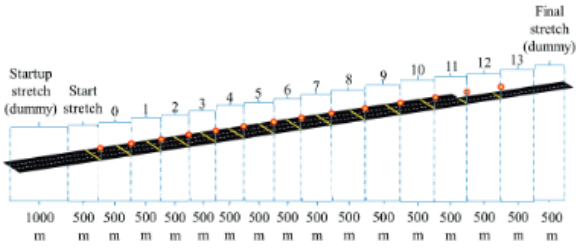

Fig. 2. Road model with multiple segments.

2. Realistic congestion

As already discussed in the introduction, the 3-to-2 lane reduction merge is already a quite realistic scenario for a motorway merge. This kind of merge also accounts for a more relaxed and thus more realistic merging behaviour of the simulated cars than for example a 2-to-1 lane merge. In order to create a realistic congestion, cars are being released at random locations on the startup stretch. If cars had only been released from one spot on the same lane, then SUMO wouldn't release all cars in the desired time for the simulation, which would have prevented congestion from emerging in the first place. Since traffic flow is definitely not constant, in reality there are "waves" of traffic on the roads. Besides that the flow of traffic has to be variable; so that congestion can emerge at times of peak flow and has a chance to be resolved. With the goal to recreate a realistic traffic flow one could model the input flow of cars as a sine curve or even more realistically as a Poisson distribution, which would be more representative for queueing systems and flow models. Nevertheless we decided to keep our first scenario simple and reproducable. Thus the vehicles were released at the flow rates shown in the figure 3. Overall the simulation runs for one hour. In the beginning there is a warm up period for the simulation environment where no vehicles are being realised. For the majority of the time just a few cars

are released which does not lead to a congestion on the road. This is followed by 15 minutes of a heavy congestion period. The goal of any controlling algorithm would be to spread out those approaching cars to minimise the amount of cars in the merging zone at the same time. (1)

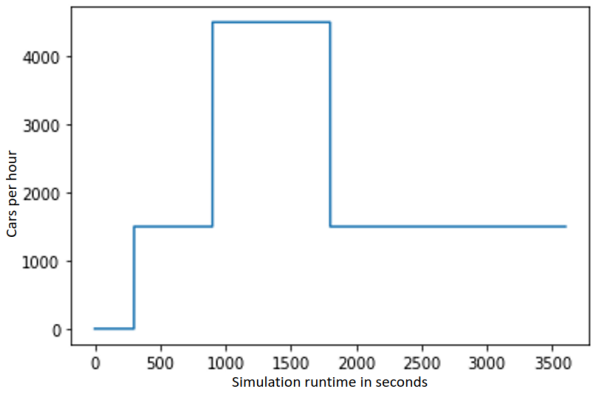

Fig. 3. Amount of vehicles released as function of the time.

3. Realistic driving behaviour

$$v_{safe}(t) = -\tau b + \sqrt{(\tau b)^2 + v_{leader}(t-1)^2 + 2bg_{leader}(t-1)} \quad [1]$$

A more realistic driving behaviour was achieved by three adjustments. First, a higher reaction time/ higher tau value was used for the cars in the function of their car-following model 1. This resulted in more even queue built-up. The second implementation was to lower the acceleration of the vehicles with the goal to more accurately represent a human driver. This adjustment resolved the problem of congestion being dissipating too quickly. Lastly we set the driving speed of the cars based on a normal distribution around the maximum speed to account for real-world drivers.

4. Verifying the congestion model

The congestion model can be verified in two main ways. Firstly, it can be verified intuitively , by visually analysing whether the congestion of the vehicles appears before the merge and the flow looks "normal" after the merge. We used this as a first sanity check. Although this method is not the most formal, that's why the second method definitely has some added value for verifying the congestion model. The second, more formal method uses the traffic flow diagram is visible in figure 4. In order to optimally make use of the fundamental diagram, the diagram can be improved by replacing density with occupancy. This is done because occupancy is a better measure for congestion for our scenario. The main difference between density and occupancy is that occupancy represents the portion of time a road spot is occupied by a vehicle which also takes the length of vehicles into account. That's why occupancy is more meaningful in a real world scenario with different types of cars than density. To verify that our model matched the pattern of the fundamental diagram, we collected the values for the flow-occupancy diagram using inductive loops. One inductive loop was used after the merge in order to measure the flow and four inductive loops were used before the merge to measure the occupancy. The maximum occupancy value over the four inductive loops was used for plotting the diagram.(2)

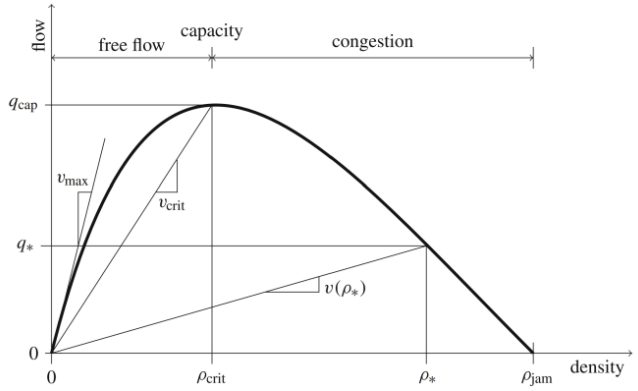

Fig. 4. Fundamental diagram (relationship between flow and density or occupancy).

Without the previously outlined adjustments to model the congestion, the recorded occupancy-flow diagram showed that the flow was plateauing with increasing occupancy, which indicates that a higher congestion would not have resulted in a capacity drop which showed us that the simulation was not yet realistic enough. After applying all the adjustments to the congestion model, we gained the following occupancy-flow plot which aligned with the theoretical knowledge and the known fundamental traffic diagram. It can be observed that the optimal occupancy for our specific merge scenario lies between 10% and 20% - we fixed it at 13%.

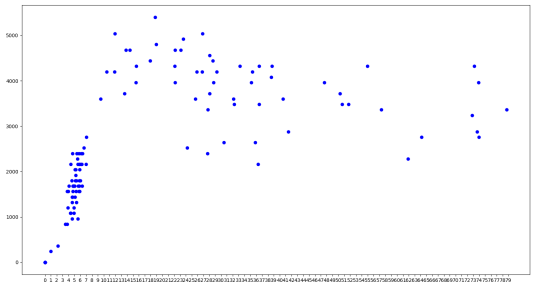

Fig. 5. Relationship between occupancy on the x axis in percentage and flow on the y axis in vehicles per hour.

Methods

We used three different VSL approaches for adjusting the speed limit of our motorway before the lane merge. We implemented the Mainstream Traffic Flow Control (MTFC) algorithm which uses occupancy at a specified bottleneck to adjust the speed limits. We also tried the Motorway Control System (MCS) that is currently implemented in Stockholm, Sweden. Those are both traditional rule-based approaches but they are still worth comparing as they rely on different traffic measurements and have different abilities to control the speed limit along the road. Finally we also tried to improve performance and safety of traffic using reinforcement learning. All of these approaches

will be described in the following sections. All of them have in common that they are able to adjust the speed limit on the road only every 30 seconds.

1. MTFC

Mainstream Traffic Flow Control is categorised as a homogenization algorithm as the speed limit is lowered with the objective of avoiding unstable traffic conditions.(3) For this algorithm the speed limit is set on the road segment 575m to 275m in front of the lane merge with the aim of bringing the occupancy around the bottleneck to a determined estimate of the critical occupancy and thereby avoiding a capacity drop. As determined earlier the critical occupancy is around 13% and the desired occupancy is set a little lower to 11% in order to avoid negative effects when it is slightly exceeded. The variable speed limit at time t is calculated with following function where b ranging from 0.2 to 1.0 is a portion of the default maximum speed limit of 120 km/h:

$$b(t) = b(t-1) + K'_I e_0(t) \quad [2]$$

K'_I is the integral gain (set to 0.005) and $e_0(t)$ is the occupancy error which is calculated as the difference between critical occupancy and measured occupancy at the bottleneck.(4)

2. MCS

Motorway Control System was originally developed as an incident detection system with the objective of warning drivers of incidents and decreasing the risk of further accidents.(4)

This algorithm uses a threshold for lowering and resetting the speed limit on different segments of the road. One VSL sign and one detector is used per segment of a road. The algorithm determines the new speed limits based on the mean speed averaged across all lanes in a given segment. This is used to calculate smoothed harmonic mean speed (1) with the following formula:

$$\frac{1}{\tilde{v}_{t,j}} = \alpha \frac{1}{v_{measured}} + (1-\alpha)\frac{1}{\tilde{v}_{t-1,j}} \quad [3]$$

The maximum speed is 120 km/h. If the smoothed harmonic mean speed drops under 45 km/h speed limit of 60 km/h is applied to the segment. Upstream segments speed limits are set to 100 km/h and 80 km/h respectively in order to make the reduction in speed safer. This is done for all downstream segments, which propagates the congestion down the road. If the smoothed harmonic mean speed increases over the limit of 45 km/h all speed limits are reset.(4) We also tried implementing it without the smoothed harmonic mean speed, using just mean speed instead and got better results. This adjusted version of MCS is named MCS2 in the following chapters.

3. Reinforcement learning

We wanted to examine whether using reinforcement learning could improve the prior rule-based control methods that set the speed limit based on a measure of the current congestion. As with the existing control mechanisms we could not see the desired results described in the *Results* section. Furthermore we wanted to provide a proof of concept as to whether it is worth pursuing reinforcement learning (RL) approaches for this particular narrow problem set especially as it is an emerging trend.(5)

A. Model. In order to provide a fair comparison between the approaches we provided the RL algorithm the same maximum occupancy around the merging area and ability to control the traffic flow by setting the speed limit for the road segment before the merging area. In order to enable a RL agent to interact with SUMO we had to build a custom OpenAI gym environment specialised for our purpose which was achieved by reusing most of the traditional VSL algorithms controlling the SUMO. For an environment design that could maintain optimal occupancy better than the MTFC algorithm we defined the following action space:

- 0 = Lower speed limit by 10 km/h.
- 1 = Keep current speed limit.
- 2 = Increase speed limit by 10 km/h.

Consequently the observation space was defined as the maximum occupancy from any of the inductive loops around the merging area ranging from 0% to 100% which was used as the input state for the function adjusting the speed limit. We chose the same occupancy as the input of the reward function as the defined goal was to keep it at the previously investigated 11%. As an RL approach we used a Deep Q Learning agent relying on a relatively small neural network with 723 parameters and two layers.

To yield an environment that would eventually result in a model that shows an improvement to the prevailing congestion we tried several reward functions. We also considered using the mean speed of cars over the whole road as the reward input as this has been defined as one of our objective metrics but results were generally worse than when occupancy was used. The first approach has been a binary reward function rewarding an occupancy around 11% with 1 and any other value with 0. We found out that due to the same rewards for most neighbouring values the algorithm could not effectively learn in which direction it should adjust to obtain a higher reward. As a consequence we focused on continuous reward function in the following. We considered the use of a gaussian curve with an expected value around 11% occupancy but most likely because of its flat tails it did not perform well for the same reason as the binary reward function. This means that during inference we could observe that the models tended to set the speed either to the lowest or highest possible limit and we did not achieve meaningful results. Finally we started to investigate reward functions that were assembled from simpler first and second order polynomial functions. We designed them in a way that the highest possible reward for one evaluation step would be 1 and came up with the following.

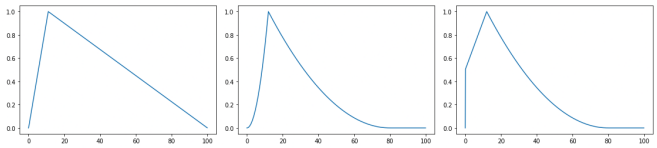

Fig. 6. The linear and quadratic reward functions.

The latter containing a combination of linear and quadratic functions achieved the best results for the mean speed of

the cars. This is most likely because it gave higher rewards to occupancy values below 11% in comparison to the other functions which rewards the model to achieve lower congestion. In addition it has to be kept in mind that we only gave a lower bound for the speed limit. The model could net at 10 km/h as we found out that when narrower bounds were in place the model would not learn them instead we gave it more freedom in its decisions. Consequently it decided to keep the default speed limit of 120 km/h and only increase it in the end. However the maximum speed of the cars was 200 km/h.

When evaluating the model it has to be kept in mind that it was only trained in one particular congestion scenario as a consequence it is very likely to overfit and not be applicable to other situations. Besides, we did not exhaust the full potential of the RL approach as it is likely to learn even better from a time series of occupancy that provides information about whether it is emerging or resolving in order to react differently. This indicates that as of now it is only able to react in the same way to situations that are in reality very different - but the same might be true for the previous algorithms that rely on information from a single time stamp.

Results

To evaluate the performance of the implemented VLS algorithms, they will be compared to each other alongside the baseline according to three main metric factors; efficiency, environmental impact, and safety. Different VSL algorithms are often designed to specifically improve one or more of these factors, typically the efficiency.(6) MCS, by contrast, primarily aims to improve driver safety along with efficiency. Comparing the simulations side by side in real time helps to show how the implemented algorithm varies the speed of the vehicles and its effect on the congestion zone. However, it is hard to evaluate the performance of the algorithms by visually comparing the simulations. Therefore, using these three metrics allows for a good comparison of the overall effectiveness of the algorithms.

Fig. 7. Congestion scenario at certain time depends on the control algorithm in use.

1. Efficiency

To evaluate efficiency, the mean speeds across all vehicles along the entire road are measured across the simulation period. The baseline (without implementing VLS) is shown in red. Here the dip observed is the congestion period. The lower the line the lower the minimal mean speed reached by the simulation. The higher and flatter the line, the more efficient the model is considered to be.

The only VLS implementation that managed to increase the mean speed above the baseline was the reinforcement learning approach. It had a minimal mean speed of only 19m/s. The RL approach was almost constantly above the baseline and had a much shorter congestion period in comparison to the other VSL implementations. Though there is an improvement, the difference in comparison to the baseline is still minimal. It must also be noted that the RL approach struggled to regulate its speed towards the end of the simulation period, hence the high fluctuation shown in the mean speed which is not ideal for a stable traffic condition.

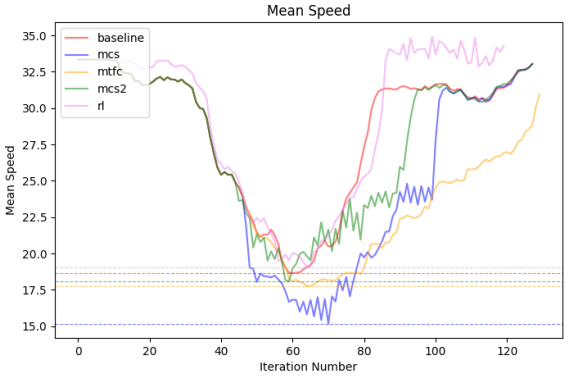

Fig. 8. The average mean speed during the simulation.

MCS2 (the slightly adapted form of MCS) performed the closest to the baseline. Both MTFC and MCS2 have mean speed levels that are slightly below the baseline when the queue is building up and until the flow is decreasing again. All VSL algorithms had a greater fluctuation in mean speed and a longer speed recovery period after peak congestion in comparison to the baseline. Overall, the VSL implementations didn't overly outperform the baseline for efficiency and in some cases performed slightly worse.

2. Emissions

In order to evaluate the environmental impact; the total CO_2 emissions across all edges were collected every 30 seconds during the entire simulation.

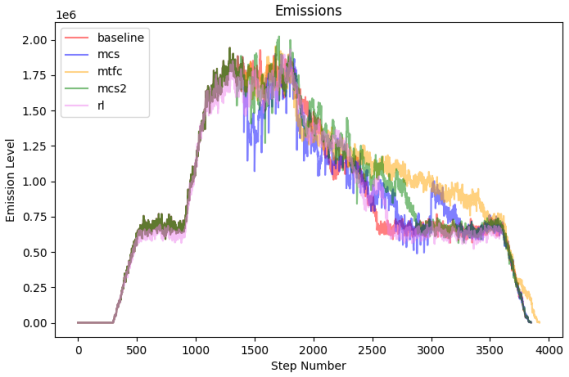

Fig. 9. Emissions produced during simulation.

As expected the emissions show a similar pattern to the mean speeds of the previous plot. The highest emissions are reached around the peak congestion period and reduce over time as the congestion begins to clear. As seen in the graph, all algorithms perform very similarly to both each other and to the baseline. MTFC has slightly higher emissions, particularly

in the second half of the simulation period. MCS2 on the other hand performs marginally better than the rest, having almost consistently the lowest emissions, particularly in the peak congestion period. These variances are still very small and overall, none of the algorithms stand out.

3. Safety

Safety was evaluated using the coefficient of variation in speed (CVS).(7) Each CVS value expresses the temporal variation in speed at a fixed position. For each segment of road j the CVS_j value is calculated as the standard deviation of speed divided by the mean speed for each lane i, averaged across all n lanes.

$$CVS_j = \frac{1}{n} \sum_{j=0}^{n} \frac{\sigma_i}{\overline{s_i}} \quad [4]$$

The higher the CVS value, the higher the variation in speed and consequently the less safe it is.(4) For each of the simulations, the CVS values were calculated for each of the 13 road segments of 500m in length at 30 seconds intervals across one hour long simulation period. These measurements have been plotted as heat maps. The road segments are plotted on the y axis, with the segments of road nearing the merge, closer to the x-axis. The time intervals are shown along the x-axis. The lighter the colour, the higher the CVS value.

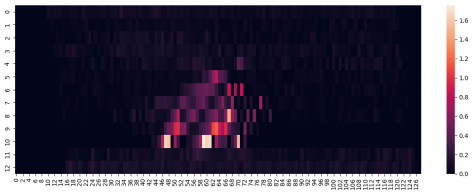

Fig. 10. Heatmap of emissions for baseline.

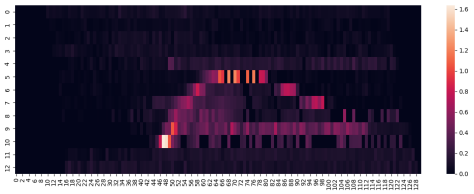

Fig. 11. Heatmap of emissions for MTFC approach.

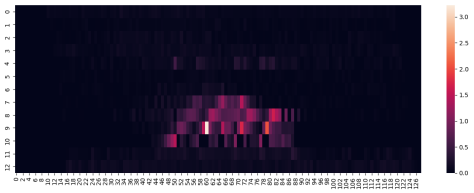

Fig. 12. Heatmap of emissions for MCS2 approach.

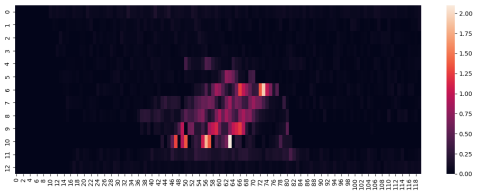

Fig. 13. Heatmap of emissions for RL approach.

Across all plots, higher CVS values can be observed on the later road segments around the bottleneck. The spread of speed variation and the range of CVS values vary slightly. To compare the maximum CVS values reached, a closer look at the scales is needed. The baseline has a maximum CVS value of around 1.75. Both MCS2 and RL reach higher CVS values, with around 3.2 and 2.1 respectively. Therefore, it performed worse than the base case with a slight decrease in safety. Though MCS has been known to improve safety in previously studied cases during this simulation it had the opposite effect.(8) MTFC on the other hand reached a lower CVS value, only reaching just over 1.6. Overall MTFC has lower CVS values across the simulation in comparison to both the base case and the other algorithms. This would suggest that MTFC made the best improvement to safety.

Moreover we can conclude from the heatmaps that only for MTFC half of the observed time some road segments were unsafe to some degree whereas for the other approaches this only applies to about one third of the time. This lets us conclude that we have a tradeoff between a low peak safety and only a short period of an unsafe road among our algorithms. Besides that we can observe that the MCS2 algorithm manages to keep the most road segments safe throughout the whole simulation and only the four segments in front of the lane merge have to be considered unsafe.

Overview

MCS is one of the traditional rule-based algorithms implemented. It has the lowest minimal speed limit before the merge of all the algorithms, which did result in lower overall traffic efficiency. MCS2 also had a slightly higher variation in speed around the merge zone which meant a marginally lower safety level than the baseline. However, MCS2 did have the lowest CO_2 emissions while the others made no noticeable improvements.

MTFC is another traditional rule-based algorithm which aims to prevent a potential breakdown before it occurs. MTFC performed the best for safety, making an improvement to the baseline. Once congestion around the merge had already occurred, MTFC took a little dip in overall efficiency but one only marginally behind the baseline. Suggesting that MTFC is perhaps most efficient when congestion is building up slowly but is not the best VSL algorithm for resolving existing congestion. The Reinforcement Learning approach is a reactive algorithm without set rules. Out of all the implemented VSL algorithms, the RL approach made the largest improvement to traffic efficiency. The quick speed limit response and large range of overall speeds did help to resolve the congestion quicker than the other algorithms but did result in reduced safety which was noticeably below that of the baseline. The

RL approach has potential but may need work to smooth the speed limit transitions and improve safety.

The RL also used the current occupancy to set the new speed limits. All three of the algorithms are reactive algorithms which react once a sufficiently high occupancy is reached and therefore all need to tackle the congestion once it has already formed. RL didn't manage to find a function that performed vastly better than the other algorithms, so perhaps predictive models would be a better approach to reducing overall congestion times. These would be worth investigating.

Conclusion

The aim of this project was to build a simulation for a motorway merge and to explore the potential improvements to congestion around the bottleneck through VSL algorithms. One of the great challenges of this project was to develop a realistic motorway merge scenario. To implement and evaluate VSL algorithms which aim to improve congestion around a bottleneck merge, a realistic congestion was needed. As previously explored, this involved additional manipulation of both input flow and adjustments to driver behaviour. So, though SUMO is already a very advanced traffic simulator, there are some limitations.

Once the simulation had been built, the two traditional rule-based algorithms MCS2 and MTFC, alongside the reinforcement learning approach were implemented and evaluated. Though for this simulated scenario, these implemented algorithms didn't make vast improvements compared to the base case, however, the slight improvements in some aspects indicate that VSL has potential. Each of the VSL algorithms performed better for different metrics. Different VSL algorithms are developed with individual goals to improve certain metrics. For each potential application it is important to consider these and to evaluate them for the desired metrics. For other motorway merge scenarios or other VSL algorithms there may be larger improvements, so that the additional effort and cost required to use VSL could be a good investment.

It would be interesting to expand the project to include other motorway merge scenarios (such as $4-3$ or $4-2$ lane reductions) and additional VSL algorithms to assess the impact of VSL algorithms on different congestion scenarios. This project (or an adaptation thereof) has great potential for accessing different VSL algorithms before investment. For example, when preparing for upcoming road works, planning new motorway lane reductions, or reassessing current motorway bottlenecks a comparative simulation could be a great tool.

1. Ellen Grumert. *Cooperative Variable Speed Limit Systems: Modeling and evaluation using microscopic traffic simulation*. Department of Science and Technology, Linkoping University, 2014.
2. Anupam Srivastava and Nikolas Geroliminis. Empirical observations of capacity drop in freeway merges with ramp control and integration in a first-order model. *Transportation Research Part C: Emerging Technologies*, 30:161–177, 2013. ISSN 0968-090X. . URL https://www.sciencedirect.com/science/article/pii/S0968090X13000363.
3. Rodrigo Castelan Carlson, Ioannis Papamichail, and Markos Papageorgiou. Local feedback-based mainstream traffic flow control on motorways using variable speed limits. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1261–1276, 2011. .
4. Ellen F. Grumert, Andreas Tapani, and Xiaoliang Ma. Characteristics of variable speed limit systems. *European Transport Research Review*, 10(2), 2018. .
5. Yuankai Wu, Huachun Tan, Lingqiao Qin, and Bin Ran. Differential variable speed limits control for freeway recurrent bottlenecks via deep actor-critic algorithm. *Transportation Research Part C: Emerging Technologies*, 117:102649, 2020. ISSN 0968-090X. . URL https://www.sciencedirect.com/science/article/pii/S0968090X20305647.
6. Zhibin Li, Pan Liu, Wei Wang, and Chengcheng Xu. Development of a control strategy of variable speed limits to reduce rear-end collision risks near freeway recurrent bottlenecks. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):866–877, 2014. .
7. Chris Lee, Bruce Hellinga, and Frank Saccomanno. Evaluation of variable speed limits to improve traffic safety. *Transportation Research Part C: Emerging Technologies*, 14(3): 213–228, 2006. ISSN 0968-090X. . URL https://www.sciencedirect.com/science/article/pii/S0968090X06000404.
8. Amir Samimi and Bruce Hellinga. Sensitivity of a real-time freeway crash prediction model to calibration optimality. *European Transport Research Review*, 4(3):167–174, 2012. .