Statistical Model Checking for Traffic Models

Thamilselvam B, Subrahmanyam Kalyanasundaram, Shubham Parmar, and M. V. Panduranga Rao

Department of Computer Science and Engineering, IIT Hyderabad, India.
Email: {`cs17resch11005, subruk, sm20mtech12002, mvp`}`@iith.ac.in`

Abstract. Statistical Model Checking (SMC) is a popular technique in formal methods for analyzing large stochastic systems. As opposed to the expensive but exact model checking algorithms, this technique allows for a trade-off between accuracy and running time. SMC is based on Monte Carlo sampling of the runs of the stochastic system, and lends itself to stochastic discrete event simulators as well.

In this paper, we use SMC to analyze traffic models like car-following and lane-changing models. We achieve this through an integration of the SMC tool MultiVeStA with the discrete event simulation software for urban mobility, SUMO.

As illustration of the approach and the tool chain, we compare the car-following and lane-changing models against various performance parameters like throughput, emissions and waiting times. Importantly, the use of formal methods allows for formulating and evaluating complex queries that can be asked of the model. The results show the utility of such a tool chain in performing complex quantitative what-if analyses of various traffic models and policies.

Keywords: Statistical Model Checking, Traffic Modeling and Simulation, Car following and Lane changing models.

1 Introduction

Modeling and simulation is used extensively by traffic engineers for understanding and designing protocols and policies for maximizing traffic flow and minimizing jams and emissions. Microscopic traffic flow modeling and simulation, that operate at the granularity of individual vehicles have attracted attention in the recent past. The impact of how vehicles follow each other and respond to the changing dynamics of nearby vehicles, and the way vehicles change lanes in a moving traffic flow has been accepted to be significant. So much so that two important examples of microscopic traffic modeling and simulation, the class of car-following models and lane-changing models, have been studied extensively in the past. Several mathematical models have been proposed, and several simulation tools have been developed on which these models have been implemented. The simulations are studied to draw inferences or validate various predictions of the models. SUMO (Simulation of Urban MObility) is a popular tool that provides the facility to carry out such studies [12].

Given their importance, several studies have been conducted in the past to compare the relative merits of various car-following and lane-changing models [4, 10, 15, 16]. In this work, we study these models from the stand-point of (statistical) model checking.

Model checking is a branch of formal methods that is used for analysis of systems against desirable properties. In order to perform this analysis automatically, both the system under analysis and the properties need to be specified in a mathematically precise manner [6]. This technique can be applied for analyzing stochastic systems as well. Generically called probabilistic model checking, there are two important ways to accomplish this–exact model checking and statistical model checking [23]. Statistical model checking (SMC) has emerged as an attractive approach to quantitative analysis of large stochastic systems [2, 14, 18, 24]. While approaches like exact model checking that rely on expensive state space exploration are accurate, SMC offers a trade-off between accuracy and runtime [23]. An additional advantage that we leverage in this paper is that this technique lends itself to black box systems and Stochastic Discrete Event Simulators. MultiVeStA [] is an example SMC tool that allows for easy integration with (stochastic) discrete event simulators. MultiVeStA supports both temporal logic queries (like PCTL and CSL), as well as (Multi)Quatex (quantitative temporal expression) queries.

In this work, we use statistical model checking to study traffic models. Our contributions are twofold:

1. We integrate the statistical model checker MultiVeStA with SUMO.
2. We analyze various car-following and lane changing models for performance parameters using this tool chain.

From the analysis that we report through four illustrative queries in this paper, the combination of the Intelligent-Driver-Model for car-following and SL2015 for lane-changing seems to outperform the other combinations.

While we focus our attention in this paper on car-following and lane-changing models, we believe that the integrated tool will be useful in in-depth analyses of other models and questions in traffic management and policy design.

We present the prerequisite background and the tools in the next section. Section 3 briefly discusses the tool integration details. In section 4, we discuss queries that illustrate the utility of the tool chain, along with results and discussions. We conclude the paper with a discussion on future directions.

2 Background

2.1 SMC and MultiVeStA

The "quantitative" variant of statistical model checking seeks to estimate the probability that a system (say, a stochastic discrete event simulator) satisfies a property stated formally (say, as a formula in a temporal logic). The SMC algorithm answers this through a Monte Carlo sampling based evaluation. Consequently, the running time depends on the desired level of confidence. Variants that answer qualitative queries – does the system satisfy the specification formula with probability at least θ have also been studied. For such applications, typically statistical techniques like hypothesis testing are employed.

MultiVeStA is a statistical model checking tool from the VeSta family. The first in the series, VeSta, is a tool that allows a variety of model specification formalisms like (discrete and continuous) Markov chains, and the executable specification language PMaude for probabilistic read-write theories [3, 19]. Property specification languages like the PCTL and CTL are supported, but importantly, Vesta supports the QUAntitative Temporal EXpressions language (QuaTEx). PVesTa improves the performance by distributing the simulations on difference processing units [5]. The tools in the VeSTa family work as long as discrete event simulations can be performed on the models and the probability measures are well defined on the paths of the model.

MultiVeStA builds on these tools and facilitates direct integration with discrete event simulators [17]. Additionally, it offers more sophisticated analysis capabilities like counter-factual analysis, and an enhanced interface [17]. The integrated tool chain has been made available for the interested reader at `https://github.com/ThamilselvamB/Multivesta-With-SUMO.git`

In the interest of space, we will not discuss details like the syntax and semantics of Quatex and Multiquatex. Instead, we will explain the semantics of the queries that we will use in the paper.

2.2 SUMO - Simulation of Urban Mobility

Simulation of Urban MObility (SUMO) is an open source tool for microscopic road traffic simulation. SUMO supports various traffic demand modeling and measurement of road network parameters like vehicle types, emission etc. We use the TRAffic Control Interface (Traci) to control the SUMO simulator. Traci is a Python package which interacts in an online manner and retrieves all objects involved in SUMO. SUMO also supports measurement and monitoring of a large number of traffic parameters including pollutant emissions of vehicles, and details of each vehicle's journey. Additionally, it allows simulation of various detectors and detector outputs–the lane area detector and loop detectors. Mainly SUMO has many car-following models as well as lane changing models. The theory behind the car-following model is that changes of velocity of one vehicle is dependent on the leading vehicles. The lane changing model defines a method of transferring a vehicle from one lane to adjacent lanes. We are examining these categories of traffic situations in simulation.

2.3 Car-following and Lane Changing models

Two of the most important dynamics of a vehicle on a road are movement along the longitudinal direction and lateral movement. Car-following models attempt to capture how a given vehicle follows the vehicle immediately ahead of it [4, 10, 16]. Similarly, lateral movement between lanes and sub-lanes are modelled by Lane-changing models [7,8]. We briefly discuss the models that are used in this work. These models are also supported by SUMO.

Car Following Models In modeling the car-following logic, various *motion parameters* like the accelerations, velocities and relative positions of the leading and the following car are relevant. Indeed, many models employ these heavily.

1. The **Krauss** [11] model calculates a vehicle's speed in relation to the vehicle in front of it. The primary objective in this model is to calculate a safe speed V_{safe} for a vehicle in relation to the vehicle ahead:
$$V_{\text{safe}} = V_l(t) + \frac{g(t) - V_l(t)t_r}{\frac{V_l(t)+V_f(t)}{2b} + t_r} ,$$
where $V_l(t)$ is speed of the leading vehicle at time t, $g(t)$ is gap to the vehicle ahead, t_r is driver's reaction time and b is maximum deceleration of the vehicle. By adhering to this speed, the vehicle remains "safe", and provides one car-following model.
2. The **Wiedemann** [22] model is a very popular psycho-physical car-following model. Based on the instantaneous values of the motion parameters, a car is in one of several regimes–for example, *following*, *cruising*, *approaching* or *emergency*. The driver is believed to behave differently in these regimes and the behavior, in terms of acceleration, deceleration or steady speed, is modeled accordingly.
3. The **Intelligent Driver Model (IDM)** [21] is a simple model that calculates the speed of the following vehicle based on the basic motion parameters:
$$\frac{dv}{dt} = a\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^{2}\right]$$
where
$$s^*(v, \Delta v) = s_0 + vT + \frac{v\Delta v}{2\sqrt{ab}} .$$
Here, v is the current speed of the vehicle, v_0 is the desired speed, $\frac{dv}{dt}$ is the proposed acceleration, a and b are the maximum acceleration and deceleration respectively, Δv difference in the speed of the current vehicle with the vehicle ahead, s_0 is the required minimum net distance desired between the vehicles, T a headway considered safe in terms of time and δ, is an acceleration exponent.

Lane Changing Models Lateral movement of vehicles between lanes are captured by the so-called lane-changing models. Modeling a vehicle's behaviour in its current lane is somewhat simpler because the only factors that matter are the preceding vehicle's speed and location. Lane changing, on the other hand, is more difficult because the decision to change lanes is based on several conflicting objectives. There are no analytic correlations that cover the complete lane switching procedure. Instead, it is usually depicted as a series of decision-making phases such as:

– Wishing to switch lanes.
– Choosing the target lane.
– Ensuring that lane change is feasible.
– Finally, the execution of lane change based on availability of gaps in the destination lane.

We discuss two important lane-changing models that are supported by SUMO.

1. **LC2013:** This model [13] considers three main reasons for a lane change: (i) Strategic: in order to avoid dead-ends, (ii) Cooperative: to allow a nearby vehicle to perform a lane change, and (iii) Tactical: to gain speed.
2. **SL2015:** This model [9] supports sub-lanes when more than one vehicle could be present in the same lane side-by-side (provided their dimensions permit). This model builds on LC2013 and includes parameters like lateral alignment, which determines the preference of staying in the middle of lane or any one of its side.

3 Integration of MultiVeStA and SUMO Simulator

As mentioned earlier, we use the MultiVesStA model checker. Therefore, we directly integrate the model checker with the traffic simulator SUMO instead of modeling using formalisms like Markov Chains or (probabilistic) rewrite theories. We now describe briefly the process of integration.

3.1 Initial Step

In order to integrate MultiVeStA with SUMO, one needs to extend the NewState class in MultiVeStA and create instances of SUMO simulator in the same class. Had SUMO been developed in Java, then extending the NewState class in MultiVeStA would have been easier since MultiVeStA is developed in Java. Since SUMO is developed in C++, one needs to create a wrapping method to have an interface between SUMO and MultiVeStA. This interface is provided by Traci (TRAffic Control Interface) which provides the necessary package for cross platform and cross language integration. The two ways to accomplish the integration are socket communication using Traci or using Traci API which is provided as a C++ library. The Traci socket communication results in communication overhead because of the protocol and server communication. We use the Traci API which can be linked with the client code, in our case MultiVeStA.

The SUMO source code is available from the SUMO website [1]. SUMO needs to be built with SWIG (Simplified Wrapper and Interface Generator). The Traci API Library will be available as the libsumo-version-SNAPSHOT.jar file in the bin folder. As mentioned before, we use this library as an interface between SUMO and MultiVeStA. After installing MultiVeStA, we have to load the SUMO library file and MultiVeStA NewState in the same class.

3.2 Integration

The class sumoState extends the NewState of MultiVeStA and is used to act as an interface between MultiVeStA and SUMO. In the constructor part, we pass the parameters to NewState of MultiVeStA (multivesta.jar) and load the SUMO library (libsumo-version-SNAPSHOT.jar) using JNI (Java Native Interface). After completing the initial steps, we override some of the methods in NewState class. The overrides are described below.
setSimulatorForNewSimulation(randomSeed): Since SUMO does not support reset simulation directly, we create generateRouteRandom(seed) function to fulfill the requirement of MultiVeStA. This function generates the route file for the SUMO simulator with a different random number for every run. This is equivalent to resetting the simulation with initial state.
performOneStepOfSimulation(): We call the function Simulation.step() to advance the simulation one step further.
performWholeSimulation(): To run the simulation until it reaches the state in which there is no vehicle in the simulation.
rval(int): This function is used to link observable quantities of SUMO to MultiVeStA. For example, the speed of the vehicle, number of vehicles loaded into the simulation, number of vehicles reached the destination etc. The rval() functions that are used in this work are listed here. The value returned by the function is given against the corresponding rval() entry.

```
rval(0)  - the current time
rval(3)  - the number of cars waiting
rval(4)  - the time loss of vehicle
rval(6)  - number of vehicles that reach their
           destination
rval(7)  - the CO2 emission
rval(10) - traffic volume at Intersection-1
rval(11) - traffic volume at Intersection-2
rval(12) - time at which emergency vehicle
           reaches its destination
rval(15) - current traffic load
rval(17) - time at which "normal" vehicle reaches the
           destination. In the experiment, this vehicle
           is started at same time and same location to
           emergency vehicle
rval(21) - returns  traffic load at the previous step
           of the simulation
```

Listing 1.1: rval() Method.

4 Simulation Experiments and Results

In all our experiments, we consider a topology of three intersections in a line (Fig. 3) and compare combinations of car-following and lane-changing models.

For the SUMO simulations, we use a heterogenic vehicles with different physical properties [20]. For emergency vehicles, which are central to Query 1 in section 4.2, we set the speedFactor as "1.9", jmDriveAfterRedTime "300", jmDriveAfterRedSpeed "5.56" are set. These configurations are available at the github repository for the tool. We mention here that the vehicles are introduced into the road network based on the Poisson distribution with different rates.

Listing 1.2 shows some of the important parameter values that we set for all our experiments.

```
-m data / cross.sumocfg
-l serversLists / oneLocalServer
-f quatex / exper1 . quatex
-bs 30 -a 0.1 -d1 2.0
-vp TRUE
-osws ONESTEP -sots 0 -sd sumoState
```

Listing 1.2: Parameters of MultiVeStA Client

Of particular note are the parameters 'a' and 'd', representing the α and δ values for confidence interval computation: The estimated value of the property in question lies within the interval $\pm\delta/2$ with probability at least $1-\alpha$. The parameter 'bs' stands for block size, and determines number of simulations after which inclusion in the confidence interval is checked.

4.1 Simple Queries

We begin by running some simple but useful Multiquatex queries that estimate vehicular CO_2 emissions (Listing 1.3) and throughput (Listing 1.4) (the number of vehicles that have reached their destination in the simulation).

```
expCo2Emission(x) = if ( s.rval(0) >= x )
    then  (s.rval(7))
 else # expCo2Emission((x)) fi ;
eval parametric(E[ expCo2Emission((k)) ],
k,1.0,1.0 ,100.0) ;
```

Listing 1.3: Expected CO2 emissions within simulation time.

```
expThroughput(x) = if ( s.rval(0) >= x )
    then  (s.rval(6))
 else # expThroughput((x)) fi ;
eval parametric(E[ expThroughput((k)) ],
k,1.0,1.0 ,160.0) ;
```

Listing 1.4: Expected throughput within simulation time.

The results of queries are shown in Fig 1 and Fig 2. As one would expect, the CO_2 increases steadily among all the car-following/lane-changing combinations at first, before plateauing. The throughput also increases with time, but the IDM-LC combination performs better.

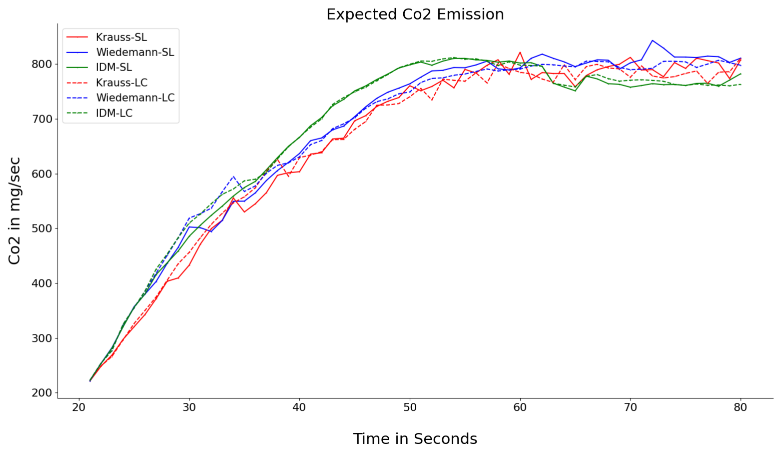

Fig. 1: Expected CO_2 Emissions.

Next we look at some queries which illustrate the usefulness of statistical model checking for analysing traffic models.

For the first three queries, we use two regimes of Poisson arrivals of the vehicles onto the road network. The first regime (at the rate of 20 vehicles per hour) is valid for the first 50 seconds, and the second one (200 vehicles per hour) is valid subsequently. For the last query (Query 4), we use four regimes (50,20,200,50 vehicles per hour for 0-50, 50-150, 150-200 and 200-400 minutes respectively), to simulate fluctuating traffic conditions.

4.2 Query 1: Behaviour on Emergency Vehicles

The first query that we ask is a natural question that arises in emergency situations. Suppose an "emergency vehicle" (say, an ambulance) and a normal vehicle of similar type, start at the same time from the same point. Which lane changing and car following model combination results in the emergency vehicle reaching the destination faster?

More precisely, what is the probability that the difference in the arrival times of the emergency vehicle and the normal vehicle is more that 20? Following is the MultiQuatex formulation that we use.

```
EVR(x) =
```

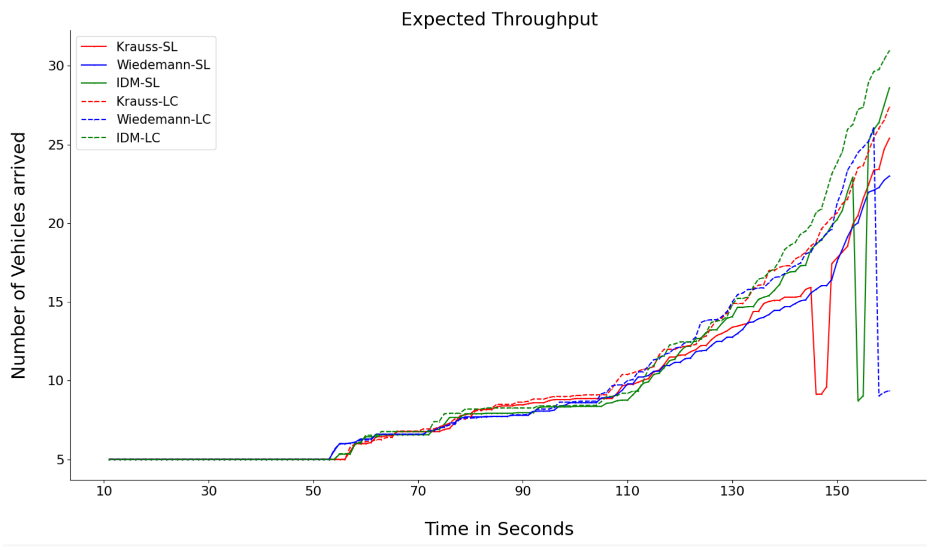

Fig. 2: Expected Throughput

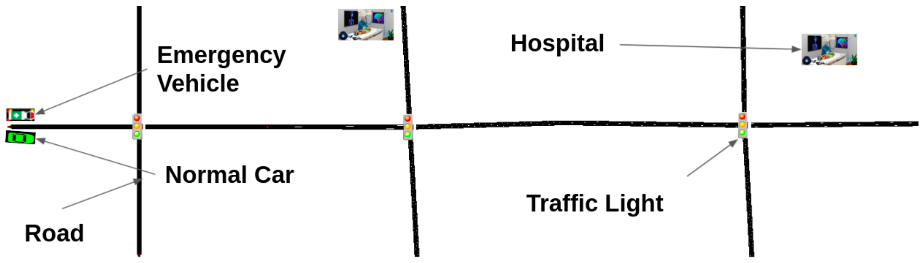

Fig. 3: Road Network with Hospital and Emergency Vehicle

```
if ( s.rval(0) >= x ) then
  ( if (s.rval(13) == 1 &&
    s.rval(16) == 1 && (s.rval(17) - s.rval(12)) < 20 )
    then (1)
    else (0) fi) else #EVR((x)) fi ;
eval
parametric( E[ EVR(k) ], k, 0.0, 1.0, 200.0) ;
```

Listing 1.5: Query for probability of emergency vehicle reached its destination faster than a “normal vehicle”

Figures 4 and 5 show the results of the query. The source-destination distance fixed for each of the plots. The X-axis shows the time. A higher time for the same distance travelled implies the presence of higher traffic.

As one would expect, when the distance between the source and destination is small, the probability that the emergency vehicle reaches significantly ahead of the normal vehicle is small. However, for scenarios of heavy traffic, (for higher time instances in Fig. 4), the probability that the emergency vehicle reaches much ahead is higher, given the relaxation in driving rules for such vehicles.

This probability increases with an increase in the distance from source to destination. This is observed in all the lane changing and car-following models. In fact, for longer distances, the probability that the emergency vehicle reaches much ahead of the normal vehicle approaches one for lighter traffic scenarios, see Fig. 5. For shorter distance, the Krauss-SL combination performs marginally better. However, as the distance increases, we see that the combination of IDM car following model and SL2015 lane changing model performs better for the light traffic regimes.

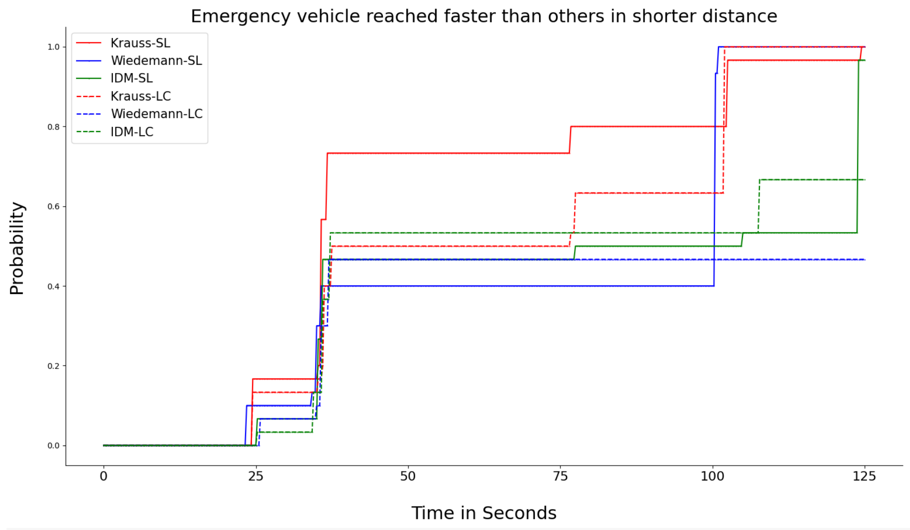

Fig. 4: Probability that Emergency Vehicle (ambulance) will reach its destination faster than others - Shorter Distance

4.3 Query 2: Traffic Load Comparison

Through this query, we demonstrate the use of the *Until* operator of temporal logic systems like PCTL and CSL, for analyzing traffic problems. The query that we show is merely illustrative, several other queries that enable insightful what-if analysis are possible.

Consider two intersections I_1 and I_2. Define “instantaneous traffic volume” at an intersection to be the instantaneous number of vehicles within 500 m of the

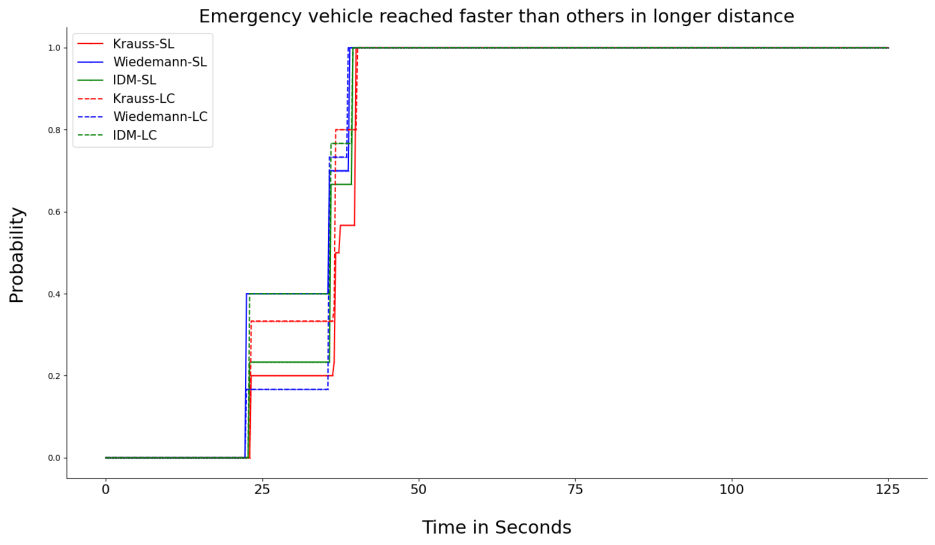

Fig. 5: Probability that Emergency Vehicle (ambulance) will reach its destination faster than others - Longer Distance

intersection in all four directions put together. Suppose we wish to ascertain that the traffic volume at intersection I_1 is less (denote it by the propositional formula ϕ_1), until the point the traffic volume is high at the intersection I_2 (denote it by the propositional formula ϕ_2). The temporal logic formula, involving the *Until* fragment, would be $\phi_1 U^{\leq \tau} \phi_2$, for different values of τ. The motivation behind such a query would be to ensure that both intersections are not heavily loaded at the same time. Following is the MultiQuatex formulation of the query:

```
t1Ut2(k,x,y) = if( s.rval(0) <= k) then
 if ( s.rval(11) > x )
   then (1)
 else if ( s.rval(10) <= y )
         then # t1Ut2((k),(x),(y))
       else  (0) fi fi else (0) fi ;
eval parametric(E[ t1Ut2((k),(20),(15)) ],
k, 1.0, 1.0, 200.0);
```

Listing 1.6: Parametric query using the Until Operator

Figure 6 shows the result of this query. The $x-$axis marks various values of τ. The probability is zero for lower values of τ because the traffic volume does not go beyond 15 within these time-steps. However, after a threshold, the traffic builds up at I_2 and we see different probabilities for the Multiquatex formula being true. We observe that the IDM-SL combination performs better. In general, for the same car-following model, the SL lane-changing model seems to perform better.

Even for higher values of τ, the probability for the Krauss-LC combination remains low. As the simulation proceeds, the traffic at intersection I_2 does increase beyond 20. However, the traffic load at I_1 also increases beyond 15, thus evaluating the formula to *false*.

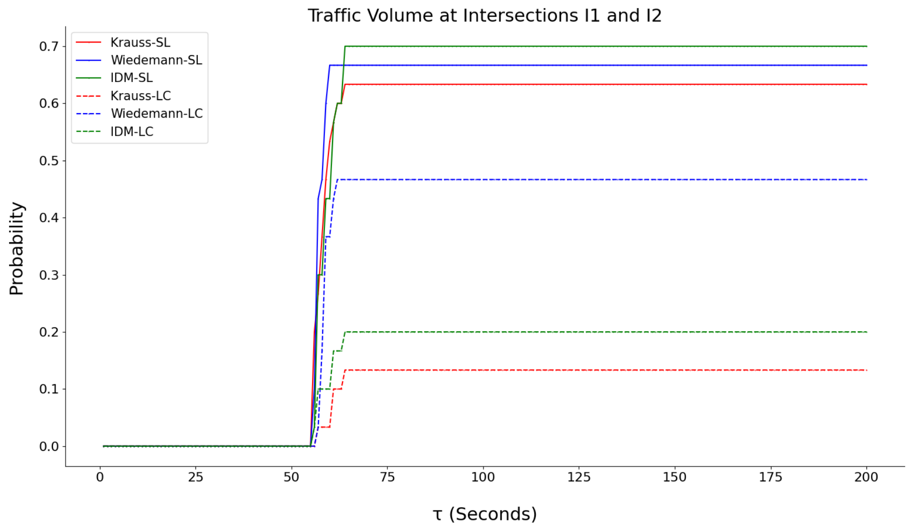

Fig. 6: Probability that the "traffic volume" at I_1 is less than 15 *Until* the traffic volume at I_2 is greater than 20.

4.4 Query 3: Load Conditions for Traffic Jams

Car-following and lane-changing models can differ in their ability of handling traffic loads without causing traffic jams.

Our next query analyzes this ability: What is the minimum traffic load that causes the number vehicles waiting at an intersection go above a threshold?

```
minJam(x,th) = if( s.rval(0) >= x) then
 if ( s.rval(3) > th ) then (s.rval(15))
 else (0) fi else #minJam((x),(th)) fi;
 wVeh(x) = if(s.rval(0)>= x) then
(s.rval(3)) else #wVeh((x)) fi;
eval parametric(E[ minJam((k,5))],
E[wVeh(k)], k, 1.0, 1.0, 200.0);
```

Listing 1.7: Minimum traffic load to jam traffic flow

Fig. 7 shows the results of this query. As before, the $x-$axis marks time, but we have two sets of curves for each car-following/lane-changing combination.

The dashed curves indicate the number of vehicles waiting at the intersections, while the solid curves indicate the total number of vehicles on the road network. For this query, IDM-LC2013 and IDM-SL2015 perform better–the number of vehicles on the road network is higher for the same approximately the same number of vehicles waiting at the intersections. We see dips in the number of vehicles occasionally as vehicles reach their destinations. Since these dips occur earlier, it also indicates that the IDM-SL2015 combination has higher throughput under the regimes in consideration.

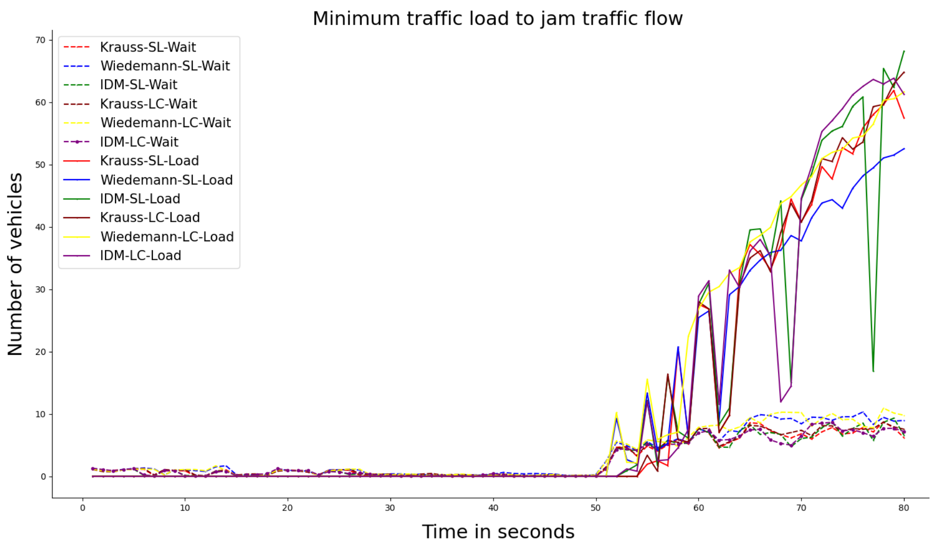

Fig. 7: Minimum traffic load to cause traffic jams

4.5 Query 4: Impending Drop in Traffic

The *neXt* operator of various temporal logics allows to query about the state of the system in the "next" step. In the context of traffic modeling and prediction, a natural question would be about the state of the traffic at the next step. We therefore ask the following query in Multiquatex: what is the probability that the traffic volume falls to 95% of the current volume in the next step?

```
TL_DropTo(x,p) =
if ( s.rval(0) >= x ) then
  (if(( s.rval(21) - s.rval(15))/s.rval(21) >= p)
      then (1)
    else (0) fi) else # TL_DropTo((x),(p)) fi ;
eval parametric( E[ TL_DropTo((x),(0.05)) ],
 x, 0.0, 10.0, 400.0);
```

Listing 1.8: Query for probability that the traffic volume drops to 95 percentage in the "next step"

For illustration, we fluctuate the traffic in SUMO to the effect. The results are shown in Fig. 8. Recall that for this experiment, we use four regimes of Poisson arrival of vehicles into the road network. Initially, since the rate of arrival of the vehicles is very small (20 vehicles per hour), and several of the vehicles reach their destinations, the traffic load dips with a high probability for all car-following/lane-changing combinations. Subsequently, a steadily increasing traffic volume results in a reduction in the probability that the volume drops to 95%. However, when the rate of injection of traffic drops in the next regime, this probability rises.

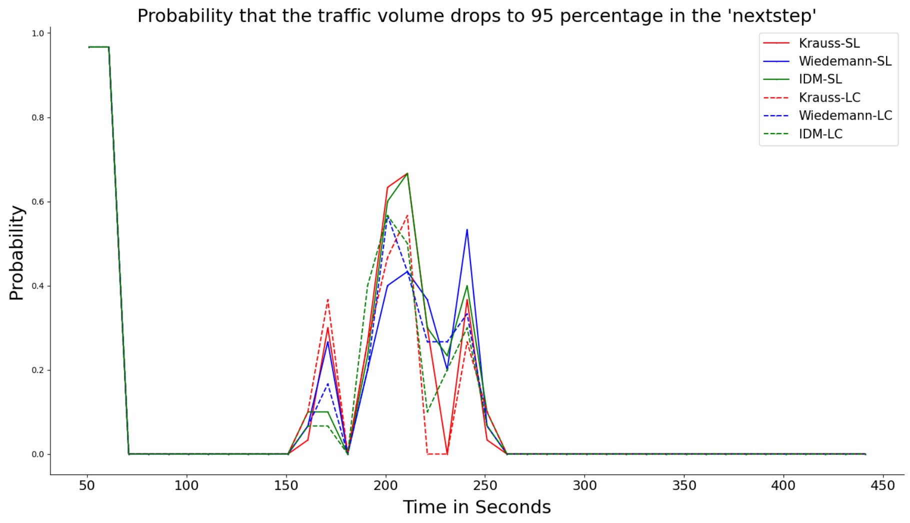

Fig. 8: Probability that the traffic volume drops to 95% in the "next step"

5 Conclusion and Future Work

In this paper, we introduce the technique of statistical model checking to traffic modeling and simulation. We demonstrate its potential by comparing combinations of various car-following and lane-changing models that are supported by SUMO. We believe that the tool chain described in the paper will help traffic engineers in analyzing micro-simulation models and performing what-if analyses.

Future work would be utilize this tool chain for a comprehensive analysis (in terms of queries) of various traffic models on realistic time-lines. A second important goal would be to validate the analysis on simulations of road-networks of cities in the real-world.

Acknowledgment

Thamilselvam B is supported by the project "M2Smart: Smart Cities for Emerging Countries based on Sensing, Network and Big Data Analysis of Multimodal Regional Transport System", JST/JICA SATREPS, Japan and Shubham Parmar is supported by the DST National Mission Interdisciplinary Cyber-Physical Systems (NM-ICPS), Technology Innovation Hub on Autonomous Navigation and Data Acquisition Systems: TiHAN Foundations at Indian Institute of Technology (IIT) Hyderabad.

References

1. Simulation of Urban MObility. `https://www.eclipse.org/sumo/`, accessed: 2021-08-06
2. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Transactions on Modeling and Computer Simulation (TOMACS) **28**(1), 6 (2018)
3. Agha, G.A., Meseguer, J., Sen, K.: Pmaude: Rewrite-based specification language for probabilistic object systems. Electron. Notes Theor. Comput. Sci. **153**(2), 213–239 (2006)
4. Ahmed, H.U., Huang, Y., Lu, P.: A review of car-following models and modeling tools for human and autonomous-ready driving behaviors in micro-simulation. Smart Cities **4**(1), 314–335 (2021). https://doi.org/10.3390/smartcities4010019
5. AlTurki, M., Meseguer, J.: Pvesta: A parallel statistical model checking and quantitative analysis tool. In: Corradini, A., Klin, B., Cîrstea, C. (eds.) Algebra and Coalgebra in Computer Science - 4th International Conference, CALCO 2011, Winchester, UK, August 30 - September 2, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6859, pp. 386–392. Springer (2011)
6. Baier, C., Katoen, J.P.: Principles of Model Checking (Representation and Mind Series). The MIT Press (2008)
7. Erdmann, J.: Lane-changing model in sumo. vol. 24 (05 2014)
8. Erdmann, J.: Sumo's lane-changing model. In: Behrisch, M., Weber, M. (eds.) Modeling Mobility with Open Data. pp. 105–123. Springer International Publishing, Cham (2015)
9. Erdmann, J.: Sumo's lane-changing model. In: Modeling Mobility with Open Data, pp. 105–123. Springer (2015)
10. Kanagaraj, V., Asaithambi, G., Kumar, C.N., Srinivasan, K.K., Sivanandan, R.: Evaluation of different vehicle following models under mixed traffic conditions. Procedia - Social and Behavioral Sciences **104**, 390–401 (2013), 2nd Conference of Transportation Research Group of India (2nd CTRG)
11. Krauß, S., Wagner, P., Gawron, C.: Metastable states in a microscopic model of traffic flow. Physical Review E **55**(5), 5597 (1997)
12. Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., Wießner, E.: Microscopic traffic simulation using sumo. In: The 21st IEEE International Conference on Intelligent Transportation Systems. IEEE (November 2018)

13. Mintsis, E., Koutras, D., Porfyri, K., Mitsakis, E., Lücken, L., Erdmann, J., Flötteröd, Y.P., Alms, R., Rondinone, M., Maerivoet, S., Carlier, K., Zhang, X., Blokpoel, R., Harmenzon, M., Boerma, S.: Transaid deliverable 3.1 - modelling, simulation and assessment of vehicle automations and automated vehicles' driver behaviour in mixed traffic (09 2019)
14. Nimal, V.: Statistical approaches for probabilistic model checking. Ph.D. thesis, University of Oxford (2010)
15. Olstam, J., Tapani, A.: Comparison of car-following models (2004)
16. Pourabdollah, M., Bjärkvik, E., Fürer, F., Lindenberg, B., Burgdorf, K.: Calibration and evaluation of car following models using real-world driving data. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). pp. 1–6 (2017). https://doi.org/10.1109/ITSC.2017.8317836
17. Sebastio, S., Vandin, A.: Multivesta: statistical model checking for discrete event simulators. In: Horváth, A., Buchholz, P., Cortellessa, V., Muscariello, L., Squillante, M.S. (eds.) 7th International Conference on Performance Evaluation Methodologies and Tools, ValueTools '13, Torino, Italy, December 10-12, 2013. pp. 310–315. ICST/ACM (2013)
18. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: International Conference on Computer Aided Verification. pp. 202–215. Springer (2004)
19. Sen, K., Viswanathan, M., Agha, G.A.: VESTA: A statistical model-checker and analyzer for probabilistic systems. In: Second International Conference on the Quantitative Evaluaiton of Systems (QEST 2005), 19-22 September 2005, Torino, Italy. pp. 251–252. IEEE Computer Society (2005)
20. Thamilselvam, B., Kalyanasundaram, S., Rao, M.V.P.: Scalable coordinated intelligent traffic light controller for heterogeneous traffic scenarios using uppaal stratego. In: 2021 International Conference on COMmunication Systems NETworkS (COMSNETS). pp. 404–412 (2021). https://doi.org/10.1109/COMSNETS51098.2021.9352946
21. Treiber, M., Hennecke, A., Helbing, D.: Congested traffic states in empirical observations and microscopic simulations. Physical review E **62**(2), 1805 (2000)
22. Wiedemann, R.: Simulation des strassenverkehrsflusses. Institut fur Verkehrswesen der Universitat Karlsruhe (1994)
23. Younes, H.L.S., Kwiatkowska, M.Z., andcDavid Parker, G.N.: Numerical vs. statistical probabilistic model checking: An empirical study. In: Tools and Algorithms for the Construction and Analysis of Systems, 10th Intl. Conf., TACAS 2004, Held as Part of the Joint European Conf.s on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proc. pp. 46–60 (2004)
24. Younes, H.L.S., Simmons, R.G.: Probabilistic verification of discrete event systems using acceptance sampling. In: Computer Aided Verification, 14th Intl. Conf., CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proc. pp. 223–235 (2002)