A framework for arterial traffic flow modeling - POLARIS

Vadim O. Sokolov, Xuesong Zhou, Pierre-Alain Langlois
Argonne National Laboratory, Arizona State University, École des Ponts ParisTech
vsokolov@anl.gov, pierre-alain.langlois@eleves.enpc.fr

Abstract

The increasing density of vehicles in urban areas and the improvement of automatic cars technology has brung the need for traffic flow modeling. On big networks, the current computation power does not already allows to use microscopic model. However, mesoscopic can be implemented thanks to power computing, Several models have been developed to do so. Among them, the reference is the Newell simplified kinetic wave model. This model allows to simulate very well the flow on highways. However, we aim in this paper at developing a mesoscopic model able to take into account the specificities of arterial roads. What's more, we will present a framework which allows to use this model properly.

Contents

I. Introduction

Polaris is a fully-integrated, agent-based simulation of both person travel and traffic operations. The model consists of a series of agent classes which implement events corresponding to typical components found in travel demand, network simulation and operations models. At the center of the model is a person-agent which represents the travelers in the system and their activity and travel planning behavior. The agent and their behaviors are implemented using a specific design pattern referred to as a computational process model. The computational process-type model is a natural fit for the interoperation of activity demand with the network simulation, as the focus is on the 'bottom-up' process of developing and implementing an activity-travel pattern from low-level decision models and behavior processes. In fact, in this instance, the network simulation actions become such low-level decision process, i.e. route choice becomes another decision of the agent constrained by the agent's context, and traffic simulation is implemented as a series of agent movements driven by another behavioral model, the bounded rationality route switching model. The person agents operate in an environment represented by the transportation network agents to handle movements through the system. A set of ITS components and an automated Traffic Management Center (TMC) agent controls the ITS system and monitors the network agents. The model allows to analyze the benefits of different network operational improvements.

Currently, the traffic simulator uses a variant of the Lighthill–Whitham–Richards (LWR)[4, 7] traffic flow model, which is a combination of a conservation law defined via a partial differential equation and a flow-density relation, which is called the fundamental diagram. The non-linear first-order partial differential equation describes the aggregate behavior of drivers. The model explicitly represents the dynamics of the primary variable of interest – traffic density which is a macroscopic characteristic of traffic flow and the control variable of interest in transportation system management strategies. Traffic density is defined as a number of vehicles per unit of length. The model is very well studied and was used in many transportation applications. Another benefit is that there exist an efficient numerical solution to the partial differential equation underlying the model. To numerically solve the partial differential equation we use the Newell's Simplified Kinematic Waves Traffic Flow model discretization scheme [5], which is a link-based solution method and has been recently recognized as an efficient and effective method for large-scale networks (Lu et al., 2013) and dynamic traffic assignment formulations (Zhang et al., 2013). A notable implementation of this model is in an open source dynamic traffic assignment tool – DTALite [8]. The model is used as the traffic simulation model agent in the POLARIS framework. The traffic simulation model includes a set of traffic simulation agents for intersections, links, and traffic controls as shown in Fig. 2. Given as input a set of travelers with route decisions and the traffic operation and control strategies in the network, the network simulation model agent simulates traffic operations and controls to provide capacities and driving rules on links and turn movements at intersections. With these capacity and driving rule constraints, link and intersection agents simulate the traffic flows using cumulative departures and arrivals as decision variables based on the Newell's Simplified model, which then determines the network performance for the route choice model, the demand model, and as well as the ITS model in the integrated framework. The traffic simulation model agents also produce a set of measures of effectiveness (MOE) such as average speed, density, and flow rate, as well as individual vehicle trajectories. The exact solution developed by Newell [6] is given by

$$T(x,n) = \max\left(T(x_u,n) + \frac{x - x_u}{u},\quad T\left(x_d, n - \rho_{jam}(x_d - x)\right) + \frac{x_d - x}{w}\right)$$

where $T(x,n)$ is the time when vehicle n crosses location x on the link, w is the shock wave propagation speed, u is the free flow speed and ρ_{jam} is the jam density of the road segment. Note, that w, u and ρ_{jam} are the parameters of the fundamental diagram. An event-base simulation scheme is implemented using Polaris' discrete event engine and traffic flow simulator is integrated with other transportation simulation components.

Polaris's traffic flow simulator aims at representing the evolution of the traffic distribution in a given network. More precisely, it takes as an entry a supply database which represents the network. The internal representation of the network is a graph whose links represent the road segments and whose nodes represent the road intersections. Moreover, details are included in the database such as the types of the links (arterial, minor, highway or expressway), the links' capacity, its max speed or it's number of lanes. Precisions about the list of parameters will be given later. The second entry of the model is a demand database. This database contains information about the vehicles traveling inside the network such as there entering time, there node of entry and there destination node. As for the supply database, precisions about the list of parameters will be given later.

The LWR model realistically reproduces traffic flows on highway roads, however, it is not usable for modeling arterial traffic flow. We introduce a new traffic flow model developed for Polaris to improve realism of traffic flows on arterial roads. The model uses an explicit representation of controlled intersections that are modeled either via a dynamically changing capacity parameter or when a smaller time step is used, states of the traffic signal are modeled explicitly. When intersection capacities are sued, at each time step, each road segment releases a certain number of cars, proportional to its capacity. This is tantamount to saying that whenever a car is not allowed to cross an intersection, it enters a queue. The first model given for the capacity computation is given by [1, p. 18-35]. This model is here given as a basis and will be improved to take into account other phenomenons in the network.

II. Improvement of the Newell's simplified kinetic wave model

I. Free-flow model

In the traveling areas, the car is not moving by changing its position in the queue : the density is supposed to be small enough for the cars to be able to drive in free-flow. However, in arterials, phases of acceleration and deceleration are the main regimes followed by the car. As a matter of fact, arterials are short and as a matter of fact, the cars can't ride at free-flow speed for long. The underlying question is : how to compute a given car's speed at each iteration ?

I.1 Model proposal

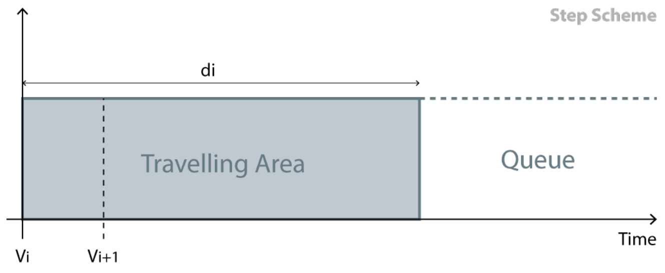

Figure 1: *Situation in the Traffic Area at step time i*

We will first introduce the notations used in the following parts :

- v_i : speed at step i
- a_i : acceleration at step i
- d_i : distance from the current car to its targeted queue at step i
- dt : time step
- $n_b(a)$: number of steps needed to stop the car with deceleration a
- $d_b(a)$: distance needed to stop the car with deceleration$a$
- v_{max} : maximum speed
- a_{dec} : maximum deceleration (positive)
- a_{max} : maximum acceleration (positive)

In this model, we assume that at every time step, we know the distance between the current car and its targeted queue, its current speed, its maximum acceleration, its maximum deceleration and the maximum speed authorized on the road segment. At each time step i, we want to compute the next speed v_{i+1}. We will proceed with a trapeze integration of the speed as represented in algorithm 1.

Algorithm 1 Speed updating

```
1: $\alpha \leftarrow f(d_i, v_i, dt, a_{dec}, a_{max})$
2: $v_{i+1} \leftarrow max(0, min(v_{max}, v_i + \alpha dt))$
3: $distance\_traveled += \frac{dt}{2}(v_i + v_{i+1})$
4: $v_i \leftarrow v_{i+1}$
```

As we can see, all we have to do is to find a model for the acceleration rate α. The first issue we have is that in the given parameters, the only two possibilities we have for the acceleration is maximum acceleration and maximum deceleration. Unfortunately, it is not relevant to model the drivers' behavior like that because it implies that the vehicle will always reach quicker the queue than in reality. As a consequence, it will put an important bias in the algorithm.

To address this concern, we will add three parameters :

- β which physically represents the mean/max deceleration rate.
  We have $\beta \in [0, 1]$.
- δ which behaves as follows :

$$\begin{cases} \delta = 0 \implies a = a_{max}(1 - \frac{v_i}{v_{max}}) & \text{in every acceleration phase} \\ \delta = 1 \implies a = a_{max} & \text{in every acceleration phase} \end{cases}$$

  We have $\delta \in [0, 1]$.
- $\gamma \in [0, 1]$ which will be a regularization parameter and will be discussed later.

The advantage of these parameters is that they allow to soften the drivers' behavior in the traveling area. The beta parameters allows to introduce the mean deceleration βa_{dec}, which will be used to soften braking. To soften the acceleration phase, we can have the idea to accelerate as much as possible when kicking off and to stop the acceleration when reaching the maximum speed with a linear model. However, this is not realistic because drivers will actually effectively reach the maximum speed in many cases. As a consequence, the correct formula for the acceleration will be

$$a = a_{max}\left(1 - \frac{v_i}{v_{max}}(1 - \delta)\right)$$

The last issue is that we don't have a criterion which allows to know when to accelerate, and when to brake. Of course this criterion is linked to the distance to the next queue. The criterion chosen works as follows. Let's call $d_b(a)$ the distance needed to stop the vehicle with constant acceleration $-a$. Then the criterion for braking is $d_i \leq d_b(\beta a_{max})$. The detailed procedure is presented in algorithm 2.

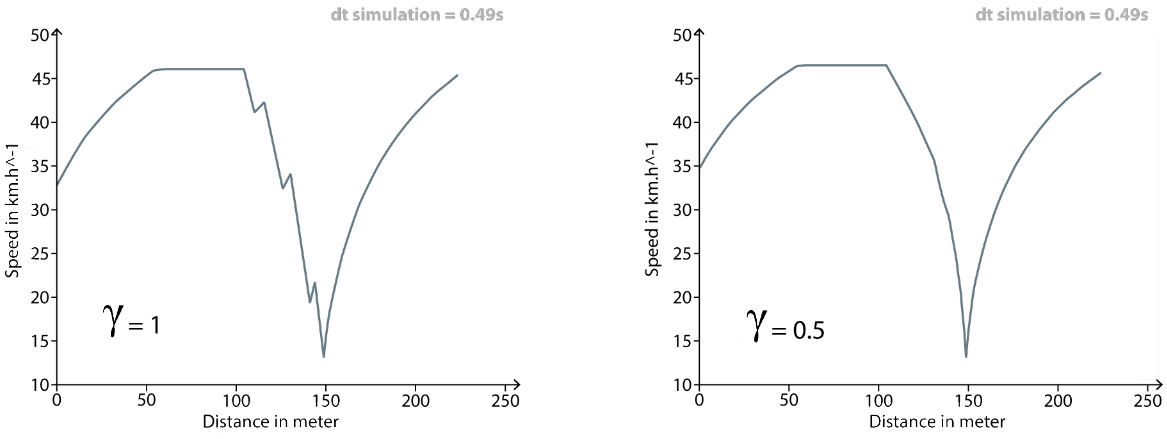

Figure 2: *Influence of gamma on the generated speed profiles*

Algorithm 2 Computing alpha

```
1: if $d_i \leq d_b(\beta a_{dec})$ then //Braking
2:     if $d_i \leq d_b(a_{dec})$ then //Emergency braking
3:         $\alpha \leftarrow -max\left(0, min\left(a_{dec}, \frac{{v_i}^2}{2d_i}\right)\right)$
4:     else
5:         $\alpha \leftarrow -\gamma\beta a_{dec}$
6: else // Accelerating
7:     if $v_{max} \leq v_i$ then
8:         $\alpha \leftarrow 0$
9:     else
10:        $\alpha \leftarrow a_{max}\left(1 - \frac{v_i}{v_{max}}(1-\delta)\right)$
   return $\alpha$
```

As we can see, an *Emergency braking* case has been introduced. In this case, we are supposed not to have enough distance to stop the car. As a consequence, we compute the constant acceleration needed to reach the queue with $v = 0$. It is

$$a = \frac{{v_i}^2}{2d_i}$$

Of course the alpha really used has to be between $-a_{dec}$ and 0 so we adjust the returned value in that way.
Otherwise, we can see that the parameter γ has been introduced in the non-emergency braking case. This situation corresponds to the case where a deceleration $-\beta a_{dec}$ is required to reach the queue with speed $v = 0$. However using this acceleration can lead to naturally undershooting the queue, which will lead to accelerate at the next step and create an hysteresis system. As a consequence, setting β between 0 and 1 allows to naturally overshoot and thus to avoid the hysteresis phenomenon as illustrated in Figure 2.

The last thing to discuss is the value of $d_b(a)$. Here we will try to take into account the fact that we have a discrete model.
Assuming that we are braking with the same deceleration a all the time, the speed is given by

$$\begin{cases} v_i \\ v_{j+1} = v_j - adt, \forall j >= i \end{cases}$$

Thus,

$$v_{n+i} = v_i - nadt$$

So formally,

$$n_b(a) = \left\lfloor \frac{v_i}{adt} \right\rfloor$$

We chose here the floor value because it implies that we under estimate the number of steps required to stop the car, so we under estimate the distance to stop the car, so we increase the chance to overshoot, so we reduce the chance of the system to enter in the hysteresis behavior we have described before.
Given this value, we have

$$d_b(a) = \frac{1}{2} \sum_{j=0}^{n_b(a)} (v_{i+j} + v_{i+j+1})dt$$

after calculation, we obtain

$$d_b(a) = v_i n_b(a)dt - \frac{1}{2} a n_b(a)^2 dt^2$$

I.2 Model calibration

Because of the difficulty to obtain a test database, the model validity has been calibrated to Vissim microscopic simulations. The choice of Vissim was made because it is a reference microscopic simulator and its validity has been widely proved. As an exemple, we show the calibration of our model on the default vehicle available on PVT Vissim 7.00 - 09. We choose to set up a 150m long road with a stop sign at length 100m from the beginning to simulate the cars' stop when it reaches the queue. After the car has stopped, it accelerates again until it leaves the road. We performed single car simulations. Because Vissim model is a stochastic model, the trajectories and speed profiles of each car going through the road is be the same. As a consequence, we have to calibrate our model's parameters to each trajectory. The initial speed and maximum speed can be easily computed from each Vissim speed profile. Meanwhile, the parameters β, γ, δ, a_{dec} and a_{max} are related to the vehicle and will be calibrated thanks to optimization.

Let's introduce new notations :

- Ω is the set of Vissim simulations
- max_s is the max iteration of simulation s
- $v^0_{s,i}$ is the speed value at iteration i of the vissim simulation s
- $v^1_{s,i}$ is the speed value at iteration i of the new model simulation corresponding to the Vissim simulation s
- n_s is the number of hysteresis cycles in the new model simulation corresponding to the Vissim simulation s

Parameter	Unit	Value
β	$\emptyset$	0.58034612
δ	$\emptyset$	0.38614873
γ	$\emptyset$	0.68841545
a_{dec}	$m.s^{-2}$	6.2254787
a_{max}	$m.s^{-2}$	3.30777879

Table 1: *Optimized parameters*

Here is the function we want to optimize :

$$f(\beta, \gamma, \delta, a_{dec}, a_{max}) = \sum_{s \in \Omega} \sum_{i=0}^{max_s} (v_{s,i}^0 - v_{s,i}^1)^2 + \lambda n_s$$

where λ is an optimization parameter which has to be taken big in order to avoid hysteresis cycles in the simulations. The model has no mathematical expression and it uses thresholds. As a consequence, optimizing f is a highly non linear problem. We tried several methods to solve the problem : L-BFGS-B, Nelder-Mead, Powell, Sequential quadratic programming, all included in the SciPy Python's package. The methods that seem to be the most successful for this problem are Nelder-Mead and Powell. Even if these methods don't handle boundary conditions on β, γ and δ, they do not give absurd results. We chose a time step $dt = 0.5s$, because it is the kind of time step that we use in the capacity-based library in Polaris.
Results of the optimization are presented in Table 1 and a representation can be seen on Figure 3. The data presented in Table 1 can be interpreted as a representation of a given car's behaviour in traveling areas. As a consequence, we can build several sets of data to represent several types of vehicles (sport vehicle, family vehicle for example) as long as we have corresponding data to calibrate the model.

As we can see, the new model is quite cogent with the Vissim simulation. However, there is a shift on the right during the deceleration phase. We can neglect this effect for two reasons. First of all, since the curves are parallel for each model, it does not change much in term of energy ; secondly, this gap narrows when the time step decreases. As a consequence, we have build a simple model to approximate the drivers' behavior in the traveling area.

II. Queuing Model

In the parts where queuing is involved, a given car's speed is mainly influenced by the front car. A lot of approaches have been developed to address this issue [3, 2].

In this part, we explain the model we will use to get one car's speed as a function of the front car's speed and position. The model is designed in order to be executed in constant time, and to feet the cars' behavior in an urban area. What's more, we want to use as less parameters as possible and to use the parameters we already have (a_{max} and a_{dec}).

The departure point of the model will be the Newell's car-following model [6]. In this model, a linear link is given between the front speed and the safety distance the current car should keep away from the front car :

$$d_i(t) = \tau_i v_{i+1}(t) + d_i^0$$

where

- d_i is the safety distance targeted by vehicle i.

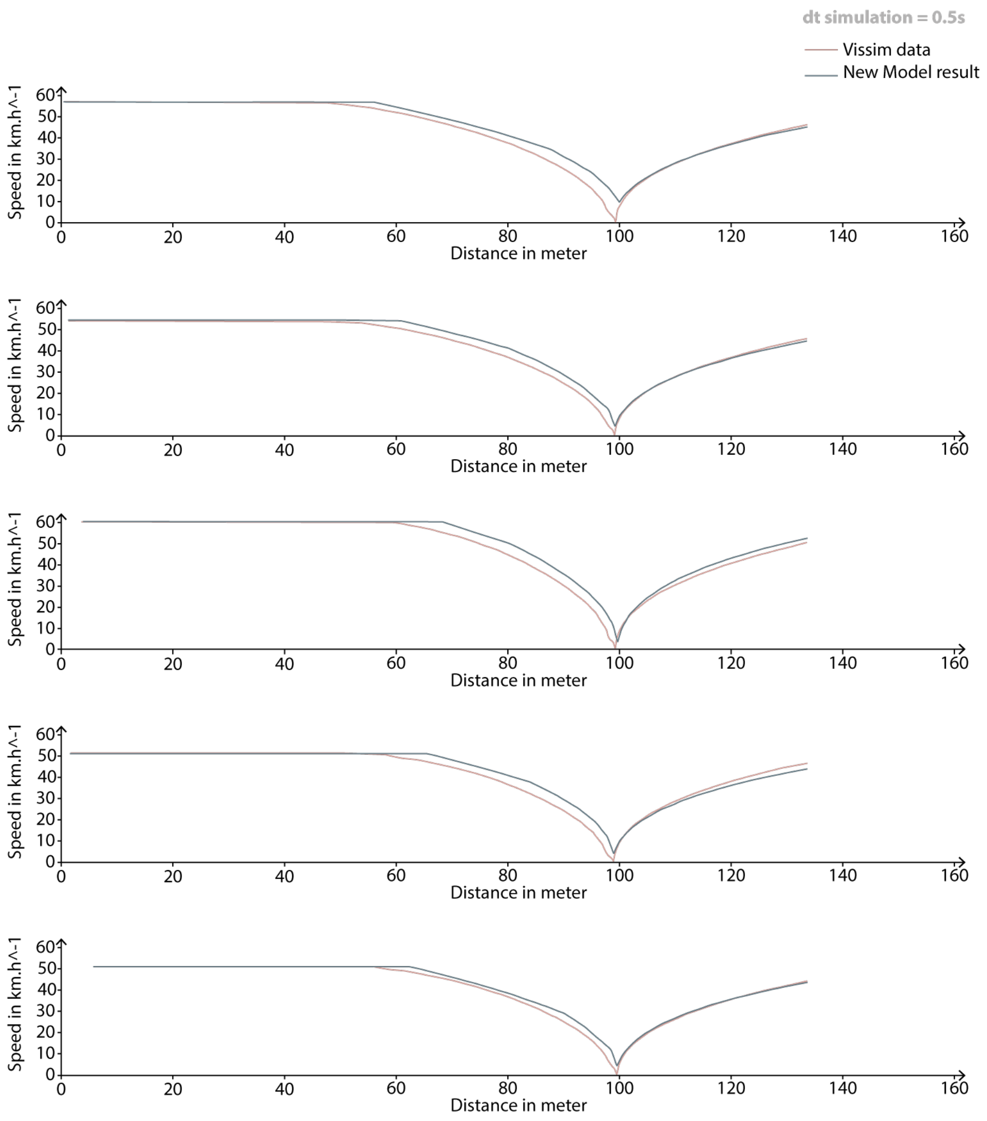

Figure 3: *Comparison between the model and Vissim simulation, optimized parameters*

- τ_i is the reaction time of vehicle i.
- v_i is the speed of vehicle i.
- d_i^0 is the safety distance of vehicle i when it is stopped.

However, this approximation is valid only when the two involved cars have the same speed. If the front cars brakes suddenly, it is possible that the current car won't have time to brake also for typical reaction time ($\approx$ 2s). As a consequence, the difference of speeds $(v_i - v_{i+1})$ should also be involved in the model :

$$d_i(t) = \tau_i v_{i+1}(t) + \tau_i'(t)(v_i - v_{i+1}) + d_i^0$$

We suggest the following method to estimate the new parameter $\tau_i'(t)$. Assuming a worst case approach, we will neglect the first and the third term. The worst case than can happen is when the two vehicles travel at speed v_{max} and that the leading vehicle brakes strongly. At this point, the difference $v_i - v_{i+1}$ can be close to v_{max}. Thus, the distance required for the following car to brake is

$$d = \frac{1}{2}\frac{v_{max}^2}{a_{dec}}$$

Identification gives then :

$$\tau' = \frac{1}{2}\frac{v_{max}}{a_{dec}}$$

In order to approach this distance in real time, we use a very simple controller described in Algorithm 3. The algorithm is equivalent to a P-Controller with coefficient 1 with boundaries to avoid unrealistic speeds and accelerations.

Algorithm 3 Speed updating in queuing model

```
1: $d_i \leftarrow \tau v_{i+1} + \frac{v_{max}}{2a_{dec}}(v_i - v_{i+1}) + d_i^0$
2: $\epsilon \leftarrow (x_{i+1} - x_i) - d_i$
3: $v_i \leftarrow \min(v_i + a_{max}dt, v_{max}, \max(v_i - a_{dec}dt, 0, \frac{\epsilon}{dt}))$
4: $x_i \leftarrow x_i + v_i * dt$
```

An example of the algorithm is given in Figure 4. The parameters used in this simulation are :

- $\tau = 2s$
- $d^0 = 5m$
- $a_{dec} = 6.23m.s^{-1}$
- $a_{max} = 3.31m.s^{-1}$

The model obtained allows to catch simply the reaction time of drivers and their reaction to emergency braking, which are two important aspects of the drivers in arterial roads. The worst case scenario we have suggested to find τ' can be replaced by a random variable whose distribution can be fitted to the behavior of real drivers to give more diverse and realistic car-following behavior.

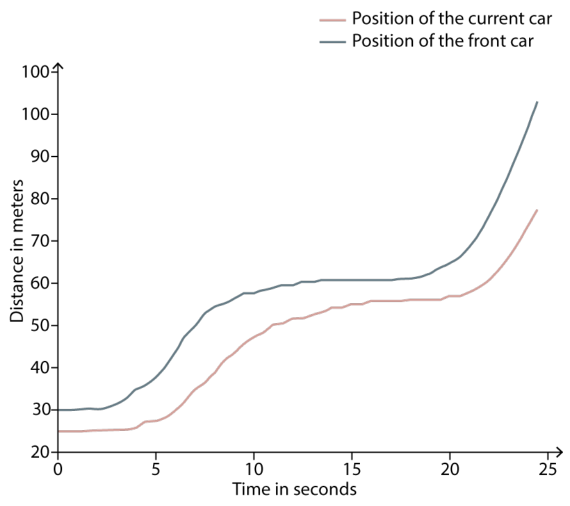

Figure 4: *Queue following algorithm example*

III. Framework presentation

I. Road Model

The most important feature of the new road model is its Junction Area (cf. Figure 5). This area is a matrix of individual queues which are all associated with a set of turning movements. In Figure 5, two pocket lanes are represented : one with a left turn and one with a right turn. As we can see, the right turn pocket lane is shorter than the other one. As a consequence, there is a natural two columns matrix of individual queues which appears. In each individual queue there is a set of turning movement associated. In the last columns, the set of turning movements leads directly to the corresponding next node. When a car reaches the end of a queue located in the first column, it can go to a neighbor lane. As an example, the queue located in position (2,0) authorizes left turn, going straight and right turn, which is the union of the authorized turning movements in the accessible queues : (1,1), (2,1) and (3,1). The pocket lanes are an exception to this rule : even if queue (1,1) is accessible from queue (0,0), we consider that a car going to a pocket lane has to stay on it. As a consequence, only the left turn is authorized in queue (0,0). Finally, the queue located in (4,0) does not really exist, as a consequence its turning movement set is empty.

A given individual queue will contain cars which follow a queuing dynamic model as described in II in addition to a set of turning as described above.

Finally, the traveling area and the common queue used in the first model are kept. The Road network is a set of roads, and it also contains a vector nodesToRoadId which allows to access a road from the id of its begin and end nodes. The whole structure is summarized in Figure 6.

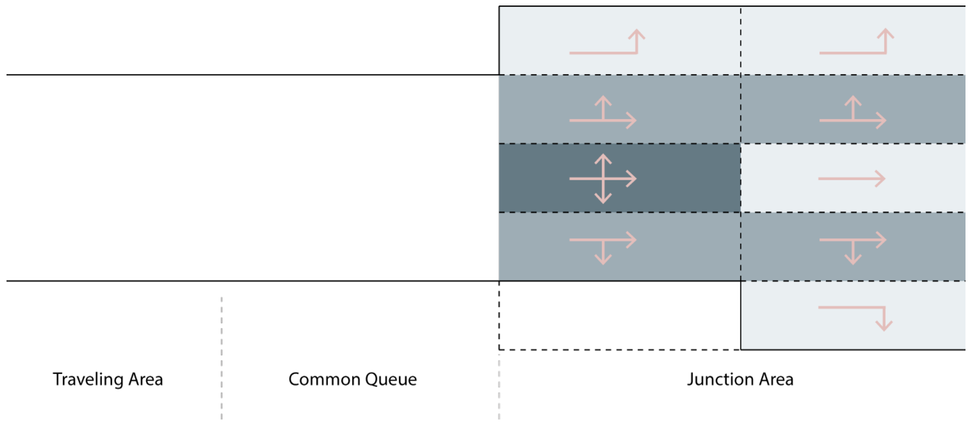

Figure 5: *New Road Model Scheme*

$$\text{Road Network}\begin{cases}[\text{Road}]\begin{cases}\text{Junction Area} \leftarrow [\text{Individual Queue}]\begin{cases}[\text{Car}]\\ [\text{Turning movement}]\end{cases}\\ \text{Common Queue}\\ \text{Traveling Area}\end{cases}\\ \text{vector}<\text{vector}<\text{int}>>\ \text{nodesToRoadId}\end{cases}$$

[object] : object set

Figure 6: *Network Structure Summary*

II. Intersection Model

The operation to transfer vehicles from junction areas to the next traveling areas are managed by the intersections which are located at each network node. The intersection manages the capacity of each turning movement and allows the cars to move. It matches an entering flow - the cars that enter the network in the current node and the entering roads - and an outgoing flow - the outgoing road and the exiting cars list. This set of flows is summarized on Figure 7. The intersections are able to manage the cars located in the junction area of the entering roads, and the cars located in the traveling area and the common queue of the outgoing road. The intersections are an important feature of the model because they are the link between the agent's paths and their movement.

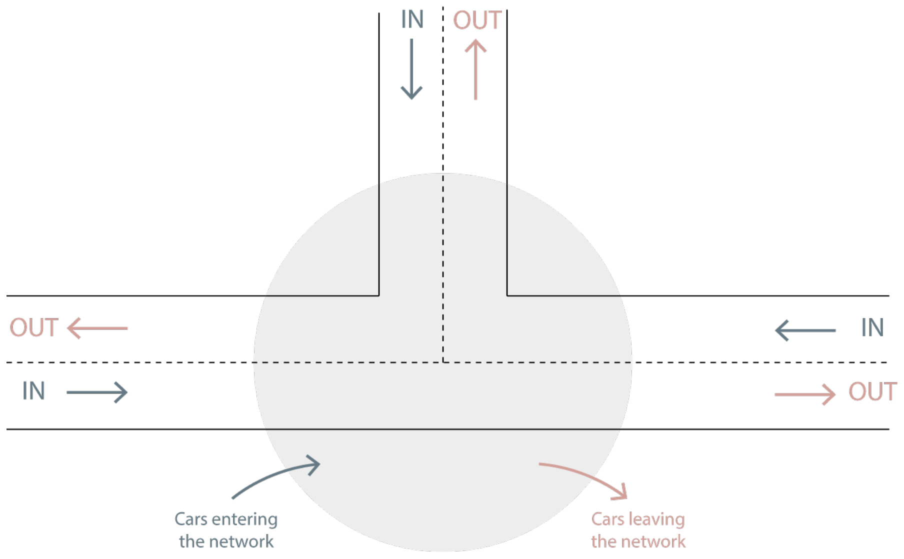

Figure 7: *Intersection Scheme*

References

[1] *HCM2010, Highway capacity manual*, volume 3. Transportation research board of the national academies, 500 Fifth Street, NW Washington, DC 20001, 2010.

[2] Johan Bengtsson. Adaptive cruise control and driver modeling, 2001. Licentiate Thesis.

[3] Mark Brackstone and Mike McDonald. Car-following: a historical review. *Transportation Research Part F: Traffic Psychology and Behaviour*, 2(4):181 – 196, 1999.

[4] M. J. Lighthill and G. B. Whitham. On kinematic waves. ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 229(1178):317–345, 1955.

[5] G.F. Newell. A simplified theory of kinematic waves in highway traffic, part i: General theory. *Transportation Research Part B: Methodological*, 27(4):281 – 287, 1993.

[6] G.F. Newell and Berkeley. Institute of Transportation Studies University of California. *A simplified car-following theory: a lower order model*. Research report (University of California, Berkeley. Institute of Transportation Studies). Institute of Transportation Studies, University of California at Berkeley, 1999.

[7] Paul I. Richards. Shock waves on the highway. *Operations Research*, 4(1):42–51, 1956.

[8] Xuesong Zhou and Jeffrey Taylor. Dtalite: A queue-based mesoscopic traffic simulator for fast model evaluation and calibration. *Cogent Engineering*, 1(1):961345, 2014.